

**Jean Patrick Figueiredo dos Santos**



# **Desenvolvendo Aplicativos com Lazarus**

## **Linux & Windows**

**1ª Edição  
2011**

## **Copyright © 2011 Jean Patrick Figueiredo dos Santos**

Este livro não pode ser reproduzido, mesmo parcial, por qualquer processo, sem autorização prévia do autor, em conformidade com a lei brasileira de direitos autorais (Lei 9610 de 19 de fevereiro de 1998).

Os nomes comerciais, marcas registradas de produtos e fotos dos mesmos, são usados nesta publicação apenas para fins editoriais, em benefício exclusivo do dono da marca registrada, sem nenhuma intenção de atingir seus direitos.

### **Direitos reservados por:**

Jean Patrick Figueiredo dos Santos

**Produção:** Jean Patrick Figueiredo dos Santos

**E-mail do Autor:** [orion.jean@hotmail.com](mailto:orion.jean@hotmail.com)

**Site:** [www.jpsoft.com.br](http://www.jpsoft.com.br)

# Índice

<b>Sobre o Autor.....</b>	<b>8</b>
<b>Introdução.....</b>	<b>9</b>
A Quem Interessa Este Livro.....	9
Como Está Organizado o Livro.....	10
Download do Código Fonte dos Projetos.....	13
<b>1 - Breve história do Free Pascal e da IDE Lazarus.....</b>	<b>14</b>
História do Free Pascal.....	14
História da IDE Lazarus.....	15
Licença do Lazarus e do Free Pascal.....	16
<b>2 - Obtendo as Compilações do Lazarus e do Free Pascal.....</b>	<b>18</b>
<b>3 - Instalando o Lazarus no Windows.....</b>	<b>21</b>
Instalação Padrão (Recomendado).....	21
Instalando usando o SVN (Avançado).....	23
<b>4 - Instalando o Lazarus no Linux.....</b>	<b>28</b>
Instalando usando a Central de Programas do Ubuntu.....	28
Instalando usando os Pacotes (Recomendado).....	29
Instalando usando o SVN (Avançado).....	31
<b>5 - Breve Comparativo Lazarus e Delphi 7.....</b>	<b>36</b>
Extensões de arquivos.....	36
Comparativo de ferramentas.....	36
Componentes de Terceiros Gratuitos.....	39
Velocidade de Compilação.....	40
<b>6 - Conhecendo e Configurando a IDE.....</b>	<b>41</b>
Mudando o idioma da IDE para Português do Brasil.....	41
Organizando as janelas da IDE.....	42
Outras Configurações.....	43
Conhecendo os Menus da IDE.....	46
Barra de Ferramentas Rápidas.....	65
Conhecendo as Paletas de Componentes.....	65
Teclas de Atalho do Lazarus.....	70
<b>7 - Instalando Novos Componentes.....</b>	<b>73</b>
Preparativos para Instalar os Componentes.....	73

Instalando os Componentes do PowerPDF .....	74
Instalando os Componentes do ZEOS.....	76
Instalando os Componentes do Fortes Report.....	78
Instalando os Componentes do LazReport e WebLaz.....	80
<b>8 - Programação Visual no Lazarus – Introdução.....</b>	<b>83</b>
Tipos de Projetos do Lazarus.....	83
Criando o Primeiro Programa.....	84
Diminuindo o Tamanho do Executável.....	90
Conhecendo o Explorador de Código.....	93
Conhecendo o Inspetor de Projetos.....	93
Conhecendo o Inspetor de Objetos, Propriedades e Eventos.....	93
Configurando Propriedades e Eventos dos Objetos.....	94
Criando o Segundo Programa.....	95
Alinhando Componentes.....	97
Bloco de Código.....	98
Linha de Comando.....	99
Comentários.....	99
Variáveis.....	99
Tipos de Variáveis.....	101
Constantes.....	102
Atribuição.....	103
Arrays ou Matrizes.....	103
Record ou Registro.....	104
Operadores Aritméticos.....	105
Operadores Lógicos.....	105
Operadores Relacionais.....	105
Procedures e Functions.....	106
Configurações Regionais.....	107
Função StrToInt.....	109
Função IntToStr.....	109
Função StrToFloat.....	109
Função FloatToStr.....	110
Função Date.....	111
Função Time ou Now.....	111
Função StrToDate.....	111
Função DateToStr.....	111
Função StrToTime.....	111
Função TimeToStr.....	111

Recursos do Editor de Código.....	111
<b>9 - Programação Visual no Lazarus – Estruturas de Controle...</b>	<b>114</b>
Estrutura TRY .. EXCEPT .. END.....	114
Estrutura IF ... THEN ... ELSE.....	114
Programa Exemplo – IF ... THEN ... ELSE.....	115
Estrutura WHILE .. DO.....	119
Programa Exemplo – WHILE ... DO.....	120
Estrutura REPEAT .. UNTIL.....	121
Programa Exemplo – REPEAT .. UNTIL.....	122
Estrutura FOR .. TO (DOWNT0) .. DO.....	123
Programa Exemplo – FOR .. TO (DOWNT0) .. DO.....	124
Estrutura CASE .. OF.....	126
<b>10 - Programação Visual no Lazarus – Projetos de Exemplo....</b>	<b>128</b>
Criando uma Calculadora Completa.....	128
Criando um Editor de Texto Simples.....	143
Criando um Visualizador de Imagens.....	154
<b>11 - Conceitos Básicos sobre Banco de Dados e SQL.....</b>	<b>166</b>
Tabelas.....	166
Chave Primária.....	166
Chave Estrangeira.....	166
Transações.....	166
Aplicativos de Banco de Dados.....	167
Criar e Gerenciar Banco de Dados.....	167
Linguagem SQL.....	168
Tipos de Campos.....	168
Criação de Tabelas.....	171
Comando SELECT.....	174
Comando INSERT.....	175
Comando UPDATE.....	176
Comando DELETE.....	176
<b>12 - Acessando Banco de Dados com SQLdb.....</b>	<b>178</b>
Modelo de Acesso a Banco de Dados.....	178
Visão Geral dos Componentes.....	179
O Componente TSQLConnector.....	180
Configurando Conexão a Bancos de Dados.....	181
Criando Bancos, Tabelas e Outros via Código.....	182
O Componente TSQLQuery.....	183

Personalizando o TSQLQuery.....	187
Executando Várias Instruções SQL com TSQLQuery.....	188
Filtrando Registros com TSQLQuery.Filter.....	189
Filtrando Registros com TSQLQuery.ServerFilter.....	190
Filtrando Registros com TSQLQuery.SQL.....	191
Localizando Registros com TSQLQuery.Locate.....	192
Formulário DataModule.....	193
TDatasource.....	193
TDBNavigator.....	194
TDBGGrid.....	194
TDBEdit.....	194
TDBMemo.....	195
TDBComboBox TDBListBox TDBRadioGroup TDBGroupBox....	195
TDBCheckBox.....	195
TDBLookupComboBox   TDBLookupListBox.....	195
TDBCcalendar.....	195
TDBImage.....	195
Criando um Controle de Cheques com SQLite3 e SQLdb.....	196
Relacionamento Mestre / Detalhe com SQLdb.....	216
<b>13 - Acessando Banco de Dados com ZEOS.....</b>	<b>220</b>
Visão Geral dos Componentes.....	220
Configurando Conexão Usando o TZConnection.....	221
O componente TZQuery.....	223
Criando um Cadastro de Produtos com Firebird e ZEOS.....	223
<b>14 - Criando Relatórios com LazReport.....</b>	<b>256</b>
Visão Geral dos Componentes.....	256
O Componente TfrReport.....	258
Barra de Ferramentas Objetos.....	260
Paleta de Alinhamento.....	261
Barra de Ferramentas Padrão.....	262
Barra de Ferramentas Texto.....	263
Barra de Ferramentas Retângulo.....	265
Barra de Status do Editor de Relatórios.....	265
Usando o Teclado e o Mouse no Editor de Relatórios.....	266
Trabalhando com Bandas (Seções de Relatório).....	268
O Objeto Caixa de Texto.....	270
A Ferramenta Realçar Atributo.....	270
Relatório do Controle de Cheques com TFrPrintGrid.....	270

Criando Relatórios para o Cadastro de Produtos.....	273
<b>Apêndice A - Configurando um Projeto.....</b>	<b>294</b>
Configurações da Aplicação.....	294
Configuração dos Formulários.....	295
Configurações de Versão.....	297
Configurando o Nome do Executável.....	298
<b>Apêndice B - Dicas de Programação Multiplataforma.....</b>	<b>299</b>
Gerenciamento de Arquivos e Pastas.....	299
Usando a diretiva {\$IFDEF}.....	301
Unidades Multiplataforma.....	302
TProcess para Executar Programas Externos.....	303
Visual Multiplataforma.....	304
<b>Apêndice C - Arquivos de Texto como Banco de Dados.....</b>	<b>306</b>
Componentes de Acesso a Arquivos de Texto.....	306
Desenvolvendo o Projeto de uma Agenda.....	306
<b>Apêndice D - Acessando Banco de Dados DBF.....</b>	<b>312</b>
Tipos de Tabelas.....	312
Criando Tabelas Via Código.....	312
Tipos de campos aceitos.....	314
Trabalhando com Índices.....	315
Manutenção de Tabelas e Índices.....	316
Cadastro de Clientes com DBF.....	317
<b>Apêndice E - Instalando Servidores de Banco de Dados.....</b>	<b>332</b>
Instalando o SQLite3 no Ubuntu e Derivados.....	332
Instalando o SQLite3 no Windows.....	334
Instalando o Firebird 2.1 ou 2.5 no Ubuntu e Derivados.....	334
Instalando o Firebird 2.1 ou 2.5 no Windows.....	335
Instalando o MySQL 5.1 no Ubuntu e Derivados.....	336
Instalando o MySQL 5.1 no Windows.....	337
Instalando o PostgreSQL 8.4 no Ubuntu e Derivados.....	339
Instalando o PostgreSQL 8.4 no Windows.....	343
<b>Links Úteis.....</b>	<b>345</b>
<b>Bibliografia.....</b>	<b>346</b>

## Sobre o Autor



Jean Patrick aprendeu a programar aos doze anos em Basic e DBase II Plus usando um computador MSX. Três anos mais tarde, em 1990, fez um curso de Clipper Summer 87 patrocinado pelo Instituto de Desenvolvimento Econômico e Social do Pará (antigo IDESP). Por muito tempo, programou por hobby e para pequena empresa de eletrônica da família.

Em 2000, passou a usar sua primeira distribuição Linux (Conectiva 5) em dual boot com o Windows.

Em meados de 2006, começou a estudar programação em Delphi e Kylix (descontinuada versão do Delphi para Linux), com o intuito de desenvolver sistemas comerciais. Como não tinha condições de comprar uma licença comercial do Delphi e ser totalmente contra o uso de cópias piratas, começou como desenvolvedor autônomo usando a recém lançada versão gratuita do Delphi, o Turbo Delphi Explorer 2006, e o servidor de banco de dados Firebird, também, gratuito. Um ano depois, conheceu o Lazarus e o Free Pascal, e passou a acompanhar o desenvolvimento destes e a fazer testes.

Atualmente, mantém o site **[www.jpsoft.com.br](http://www.jpsoft.com.br)** com dicas sobre Lazarus e alguns sistemas desenvolvidos nesta ferramenta. Também, trabalha na divisão de informática da prefeitura de sua cidade, desenvolvendo sistemas para Linux usando o Lazarus com um servidor de banco de dados PostgreSQL.



## Introdução

O Lazarus é um Software Livre maduro para o Desenvolvimento Rápido de Aplicações (RAD) com base no moderno compilador Free Pascal. Com esta ferramenta é possível o desenvolvimento de aplicações multiplataforma. Isto significa que podemos escrever um código e a partir deste compilar aplicativos para executarem em Windows, Linux, Mac OS X ou outra plataforma compatível. Por isso, o lema do Lazarus é: *“Escreva uma vez e compile em qualquer lugar”*.

Podemos, também, criar aplicativos para dispositivos móveis, bem como aplicações para Internet usando a tecnologia CGI ou FastCGI.

Este livro tem por objetivo ajudar o leitor, iniciante ou não em programação, a conhecer o Ambiente Integrado de Desenvolvimento (IDE) do Lazarus e a programação em Object Pascal usando o compilador Free Pascal.

Também, apresenta a IDE Lazarus como alternativa viável ao Delphi, com a vantagem de ser multiplataforma e gratuita.

Embora sejam mencionados alguns aspectos mais técnicos, o livro concentra-se no uso prático do Lazarus.

São abordados assuntos como: acesso a bancos de dados, geração de relatórios visualmente, compilação para Windows e Linux, entre outros temas.

### **A Quem Interessa Este Livro:**

Este livro é para aqueles que desejam aprender a desenvolver aplicações multiplataforma para desktop usando a IDE Lazarus para o compilador Free Pascal e, também, é indicado para

desenvolvedores Delphi, Visual Basic ou de outra linguagem que desejam migrar para Lazarus. Para compreender os assuntos abordados no livro, é necessário o leitor ter noções de algoritmo e alguma linguagem de programação. Portanto, se você não tem nenhum conhecimento nesta área, recomendo a leitura prévia de alguma apostila, livro ou site sobre estes assuntos.

O conteúdo deste livro é útil tanto para programadores iniciantes como para programadores experientes.

A didática é um dos pontos fortes deste livro. Os assuntos são considerados com clareza e objetividade, os códigos dos exemplos são, em sua maioria, comentados e os projetos desenvolvidos são explicados passo a passo para facilitar o aprendizado.

Todos os exemplos do livro são multiplataforma. Foram testados no Windows e no Linux.

### **Como Está Organizado o Livro:**

O livro é composto de catorze capítulos e cinco apêndices. Principalmente se você for iniciante, deve ler os capítulos na sequência (recorrendo aos apêndices quando necessário), pois o conhecimento apresentado em um capítulo requer o conhecimento adquirido em capítulos anteriores. Segue um resumo dos assuntos abordados no livro.

- **Capítulo 1 - Breve história do Free Pascal e da IDE Lazarus** – apresenta a história da criação do compilador Free Pascal, bem como a história do surgimento da IDE Lazarus. É abordado, também, o tipo de licenciamento usado pelo Free Pascal e pelo Lazarus.
- **Capítulo 2 - Obtendo as Compilações do Lazarus e do Free Pascal** – explica onde podemos baixar o Free Pascal e o Lazarus. Aborda os diversos pacotes disponíveis para os principais sistemas operacionais.

- **Capítulo 3 - Instalando o Lazarus no Windows** – explica o processo de instalação e pós-instalação do Free Pascal/Lazarus no Windows usando o instalador padrão, bem como baixar e compilar para Windows o Free Pascal/Lazarus a partir do repositório SVN.
- **Capítulo 4 - Instalando o Lazarus no Linux** – explica o processo de instalação e pós-instalação do Free Pascal/Lazarus no Linux (distribuição Ubuntu e derivados) usando os pacotes disponíveis, bem como baixar e compilar para Linux o Free Pascal/Lazarus a partir do repositório SVN.
- **Capítulo 5 - Breve Comparativo Lazarus e Delphi 7** – aborda as principais similaridades e diferenças entre a IDE do Delphi 7 e a IDE do Lazarus. Pontos como velocidade de compilação e componentes de terceiros disponíveis para as duas IDE's, também, são considerados.
- **Capítulo 6 - Conhecendo e Configurando a IDE** – aqui são explicados os principais recursos da IDE Lazarus. Considera, também, várias configurações que podem ser feitas para deixar a IDE mais intuitiva e inclui explicação sobre cada item dos onze menus disponíveis.
- **Capítulo 7 - Instalando Novos Componentes** – aborda todos os passos envolvidos na instalação de novas bibliotecas de componentes na IDE Lazarus. Apresenta passo a passo o processo de instalação de bibliotecas de terceiros e bibliotecas que acompanham o Lazarus, mas não são instaladas por padrão.
- **Capítulo 8 - Programação Visual no Lazarus - Introdução** – ensina como criar os primeiros projetos no Lazarus, diminuir o tamanho dos executáveis gerados, os recursos do editor visual de formulários, os recursos de edição de editor de códigos, bem como

apresenta uma introdução a linguagem Object Pascal do Free Pascal.

- **Capítulo 9 - Programação Visual no Lazarus - Estruturas de Controle** – apresenta as principais estruturas de controle da linguagem Object Pascal. Cada estrutura explicada acompanha um exemplo de programa desenvolvido passo a passo.
- **Capítulo 10 - Programação Visual no Lazarus - Projetos de Exemplo** – aqui são desenvolvidos passo a passo três projetos maiores usando diversos componentes disponíveis no Lazarus. Projetos: Calculadora Completa, Editor de Texto Simples e Visualizador de Imagens.
- **Capítulo 11 - Conceitos Básicos sobre Banco de Dados e SQL** – apresenta as principais estruturas da linguagem SQL para criação de tabelas, consultas e manutenção de dados. Aborda os tipos de dados e a construção de tabelas para os principais SGDB livres disponíveis.
- **Capítulo 12 - Acessando Banco de Dados com SQLdb** – ensina como usar os componentes da paleta SQLdb para conexão nativa há diversos bancos de dados. Apresenta os componentes da paleta Data Controls, o componente TDataSource, formatação de dados, controle de erros e o formulário Data Module. É, também, desenvolvido passo a passo um Controle de Cheques com um banco de dados SQLite3 e é explicado como realizar consulta e edição mestre/detalhe usando SQLdb.
- **Capítulo 13 - Acessando Banco de Dados com ZEOS** – apresenta a biblioteca de componentes ZeosLib como uma alternativa com mais recursos para conexão nativa com bancos de dados. Usando os componentes ZeosLib, é desenvolvido passo a passo um Cadastro de Produtos com foto (incluindo botões personalizados de navegação e edição) acessando um servidor de banco de dados Firebird 2.1. Em adição, ensina como usar o aplicativo multiplataforma FlameRobin para criar bancos, tabelas, índices e auto-numérico para o Firebird.
- **Capítulo 14 - Criando Relatórios com LazReport** – apresenta os principais recursos do gerador de relatórios

LazReport. Mostra como trabalhar com o componente TFrPrintGrid para criar rapidamente relatórios a partir dos dados exibidos em um TDBGrid. Ensina passo a passo como construir relatórios usando o Editor de Relatórios do LazReport, tendo como base os projetos Controle de Cheques e Cadastro de Produtos desenvolvidos nos capítulos anteriores.

- **Apêndice A - Configurando um Projeto** – ensina como configurar um projeto. Incluindo a escolha do ícone da aplicação, ordem de criação dos formulários, versionamento e etc.
- **Apêndice B - Dicas de Programação Multiplataforma** – apresenta diversos recursos disponíveis no Free Pascal/Lazarus para criação de código multiplataforma, bem como executar programas externos.
- **Apêndice C - Arquivos de Texto como Banco de Dados** – explica como pode-se usar os componentes do Lazarus para acessar um arquivo de texto como se fosse um banco de dados. É desenvolvido passo a passo o projeto de uma Agenda de Contatos usando arquivo texto com ordenação e pesquisa por nome.
- **Apêndice D - Acessando Banco de Dados DBF** – explica em detalhes como criar, acessar e dar manutenção a banco de dados DBF usando o componente TDbf do Lazarus e sem o uso da camada BDE (Borland DataBase Engine) . Usando o componente TDbf, é desenvolvido passo a passo um Cadastro de Clientes com foto.
- **Apêndice E - Instalando Servidores de Banco de Dados** – explica como instalar os servidores de bancos de dados PostgreSQL, MySQL, Firebird e o SQLite3 tanto no sistema operacional Ubuntu Linux com no Windows.

### **Download do Código Fonte dos Projetos:**

O código fonte dos projetos desenvolvidos no livro pode ser baixado no seguinte link:

*[http://www.jpsoft.com.br/livro\\_dal.zip](http://www.jpsoft.com.br/livro_dal.zip)*

# Breve história do Free Pascal e da IDE Lazarus

## História do Free Pascal:

O Free Pascal (também conhecido pela sigla FPC e anteriormente chamado FPK-Pascal) começou por volta da década de 90, depois que a Borland descontinuou os compiladores Turbo Pascal e Borland Pascal, prejudicando uma grande comunidade de programadores.

O código do FPC começou a ser escrito pelo estudante Florian Paul Klämpfl e daí vem a sigla FPK-Pascal, que em 1997 se tornou Free Pascal.

Nos primeiros estágios, o FPC era escrito em Pascal com o compilador Turbo Pascal, que é de 16bits, mas já produzia código 32bits. Dois anos depois, o compilador já era capaz de compilar a si próprio, no processo conhecido como *bootstrapping*, e que é utilizado até hoje para compilar o FPC.

Com o tempo, o Free Pascal mostrou-se exatamente o que a comunidade queria: um compilador Pascal de 32bits moderno, robusto, estável e confiável. Com isso, o Free Pascal começou a ganhar mais e mais adeptos e os grandes da programação em Pascal começaram a escrever seus programas com ele.

Hoje, o Free Pascal (FPC) é um compilador profissional de 32 e 64 bits (não se trata de um compilador acadêmico) e de código fonte aberto. Isto significa que qualquer programador do mundo pode contribuir para melhorá-lo e não é necessário pagar por uma licença para poder usá-lo. Ele executa em diversas plataformas, incluindo Windows, Linux e Mac OS X. É

compatível com o Turbo Pascal 7.0 e o Delphi 7, pois usa a linguagem Object Pascal, rotinas, classes e unidades com o mesmo nome e funções, mas possui outras rotinas e funções adicionais. Até mesmo é possível converter projetos Delphi para Free Pascal usando o Lazarus.

A partir da versão 2.4.2 do compilador, foi melhorada a compatibilidade com versões mais recentes do Delphi.

### **História da IDE Lazarus:**

O Lazarus começou em Fevereiro de 1999. Ele foi inicialmente fundado por três pessoas: Cliff Baeseman, Shane Miller e Michael A. Hess.

Os três participavam do projeto Megido (projeto iniciado em 1998 para ser um clone de código aberto do Delphi), que foi dissolvido. Frustrados, eles começaram o projeto Lazarus. Este teve um grato crescimento de apoiadores e desenvolvedores durante os anos seguintes.

Dos três fundadores, somente Michael A. Hess ainda está no projeto. O segundo mais antigo membro do grupo é Marc Weustink. Ele juntou-se ao projeto em Agosto de 1999. Seguindo-o veio Mattias Gaertner, em Setembro de 2000. Ambos têm sido os maiores contribuidores para o núcleo que faz o Lazarus existir.

Assim, o Lazarus é um Ambiente Integrado de Desenvolvimento (IDE) de código fonte aberto para o compilador Free Pascal. Inclui um editor com destaque de sintaxe, recurso para desenhar visualmente formulários e uma biblioteca de componentes altamente compatíveis com a biblioteca de componentes visuais do Delphi (VCL). A biblioteca de componentes do Lazarus (LCL) inclui equivalentes para os

principais controles da VCL tais como formulários, diálogos, caixas de texto, conexão com bancos de dados e outros, para criação da interface visual com o usuário (GUI).

O Lazarus tem por objetivo ser completamente independente de API (Interface de Programação de Aplicações).

Por exemplo, digamos que você está criando um produto no Windows usando os *widgets* (componentes visuais) padrão. E você quer criar uma versão para Linux. Primeiro você decide que *widget set* irá usar. Digamos, Gtk+. Então você copia o código para a sua máquina Linux, compila e vincula à unidade de interface Gtk+. Pronto, você vai agora criar uma versão Linux para o produto Windows sem qualquer código adicional.

**CURIOSIDADE:** Conforme registrado na Bíblia, Lázarô foi uma pessoa ressuscitada por Jesus Cristo. Portanto, o projeto é chamado Lazarus, pois ele foi iniciado ou ressuscitado da morte do projeto Megido.

### Baseado nas seguintes fontes:

[wiki.lazarus.freepascal.org/Overview\\_of\\_Free\\_Pascal\\_and\\_Lazarus/pt](http://wiki.lazarus.freepascal.org/Overview_of_Free_Pascal_and_Lazarus/pt)  
[pt.wikipedia.org/wiki/Free\\_pascal](http://pt.wikipedia.org/wiki/Free_pascal)  
[lazarusbrasil.org/historia.php](http://lazarusbrasil.org/historia.php)

### Licença do Lazarus e do Free Pascal:

Tanto o Lazarus como o Free Pascal, são desenvolvidos sob a licença GPL GNU (General Public Licence). Isto significa que ambos acompanham o código fonte e respeitam as quatro liberdades definidas pela Free Software Foundation:

1. A liberdade de executar o software, para qualquer uso;
2. A liberdade de estudar o funcionamento de um programa e de adaptá-lo às próprias necessidades;



3. A liberdade de redistribuir cópias do software;
4. A liberdade de melhorar o programa e de tornar as modificações públicas de modo que a comunidade inteira beneficie-se da melhoria.

**IMPORTANTE:** Os aplicativos criados com o Lazarus e o Free Pascal não precisam obrigatoriamente serem software livre (usando a licença GPL GNU). Podemos criar aplicações comerciais com código fonte fechado e cobrar por elas.

## Capítulo 2

---

### Obtendo as Compilações do Lazarus e do Free Pascal

Este livro foi totalmente baseado na versão 0.9.31 em desenvolvimento do Lazarus e no Free Pascal versão 2.4.2 , 2.4.3 ou 2.4.4.

Neste capítulo, abordaremos onde realizar na Internet o download do Lazarus e do Free Pascal.

Você pode baixar a IDE Lazarus neste link:

**<http://www.hu.freepascal.org/lazarus/> .**

A estrutura dos nomes dos arquivos de instalação do Lazarus, disponíveis no link citado acima, segue basicamente os seguintes padrões:

**IDE Lazarus:** lazarus-{versão}-{número da compilação}{versão do Free Pascal (só Windows)}{data da compilação}{plataforma operacional}.{extensão};

**Free Pascal:** fpc-{versão}{data da compilação}{plataforma operacional}.{extensão};

**Free Pascal Código Fonte:** fpc-src-{versão}{data da compilação}{plataforma operacional}.{extensão}.

No link **<http://www.hu.freepascal.org/lazarus/>** , você vai encontrar a IDE Lazarus e o compilador do Free Pascal para Win32, Win64, WinCE, Mac OS X PowerPC, Mac OS X i386, Linux i386, Linux x86\_64 e Linux AMD64, como você pode ver na figura seguinte:

## Desenvolvendo Aplicativos com Lazarus

Click below for fpc and Lazarus snapshot		
Lazarus fixes	source zip	lazarus-0.9.29-29671-20110227-src.zip
Lazarus fixes	source bz2	lazarus-0.9.29-29671-20110227-src.tar.bz2
Lazarus	source zip	lazarus-0.9.31-29876-20110317-src.zip
Lazarus	source bz2	lazarus-0.9.31-29876-20110317-src.tar.bz2
Lazarus fixes + fpc 2.4.2	win32	Lazarus-0.9.29-29679-fpc-2.4.2-20110227-win32.exe
Lazarus + fpc 2.4.2	win32	Lazarus-0.9.31-29878-fpc-2.4.2-20110317-win32.exe
Lazarus + fpc 2.4.2	win32 -> arm wince	Lazarus-0.9.31-29878-fpc-2.4.2-20110317-cross-arm-wince-win32.exe
Lazarus + fpc 2.4.3	win32	Lazarus-0.9.31-29879-fpc-2.4.3-20110317-win32.exe
Lazarus + fpc 2.4.3	win32 -> arm wince	Lazarus-0.9.31-29879-fpc-2.4.3-20110317-cross-arm-wince-win32.exe
Lazarus + fpc 2.4.2	win64	Lazarus-0.9.31-29879-fpc-2.4.2-20110317-win64.exe
Lazarus + fpc 2.4.3	win64	Lazarus-0.9.31-29879-fpc-2.4.3-20110317-win64.exe
fpc	Mac OS X PowerPC	fpc-2.4.2.powerpc-macosx.dmg
fpc sources	Mac OS X PowerPC	fpcsrc-2.4.2.powerpc-macosx.dmg
Lazarus fixes(needs fpc and fpcsrc)	Mac OS X PowerPC	lazarus-0.9.29.29671-20110227-powerpc-macosx.dmg
Lazarus (needs fpc and fpcsrc)	Mac OS X PowerPC	lazarus-0.9.31.29876-20110317-powerpc-macosx.dmg
fpc	Mac OS X i386	fpc-2.4.2.intel-macosx.dmg
fpc sources	Mac OS X i386	fpcsrc-2.4.2.intel-macosx.dmg
Lazarus fixes(needs fpc and fpcsrc)	Mac OS X i386	lazarus-0.9.29.29672-20110227-i386-macosx.dmg
Lazarus (needs fpc and fpcsrc)	Mac OS X i386	lazarus-0.9.31.29878-20110317-i386-macosx.dmg
fpc srpm	Linux	fpc-2.4.2-20110317.src.rpm
Lazarus srpm	Linux	lazarus-0.9.31.29879-20110317.src.rpm
fpc	Linux i386	fpc-2.4.2-20110317.i386.rpm
fpc sources	Linux i386	fpc-src-2.4.2-20110317.i386.rpm
Lazarus (needs fpc and fpc-src)	Linux i386	lazarus-0.9.31.29879-20110317.i386.rpm
fpc	Linux x86_64	fpc-2.4.2-20110317.x86_64.rpm
fpc sources	Linux x86_64	fpc-src-2.4.2-20110317.x86_64.rpm
Lazarus fixes (needs fpc and fpc-src)	Linux x86_64	lazarus-0.9.29.29679-20110227.x86_64.rpm
Lazarus (needs fpc and fpc-src)	Linux x86_64	lazarus-0.9.31.29879-20110317.x86_64.rpm
fpc debbs	Linux i386	fpc-2.4.2-0.i386.deb.tar
Lazarus debbs	Linux i386	lazarus-0.9.31.29879-20110317.i386.deb.tar
fpc debbs	Linux AMD64	fpc-2.4.2-0.amd64.deb.tar
Lazarus debbs	Linux AMD64	lazarus-0.9.31.29809-20110313.amd64.deb.tar

*Figura 2.1 – Lista de Downloads*

Os arquivos necessários para instalação do Lazarus e do Free Pascal, de acordo com o Sistema Operacional, são os seguintes (observe que no lugar do número e da data da compilação foi colocado a letra X, pois estes mudam diariamente):

Para o sistema operacional Windows 32bits (Ex.: XP):

*Lazarus-0.9.31-XXXXX-fpc-2.4.4-XXXXXXXXX-win32.exe*

Para o sistema operacional Windows 64bits (Ex.: Seven):

*Lazarus-0.9.31-XXXXX-fpc-2.4.4-XXXXXXXXX-win64.exe*

Para o sistema operacional Windows 32bits + suporte a Windows Mobile (também funciona no Seven):

*Lazarus-0.9.31-XXXXX-fpc-2.4.4-XXXXXXXXX-win32.exe*  
*Lazarus-0.9.31-XXXXX-fpc-2.4.4-XXXXXXXXX-cross-arm-wince-win32.exe*

Para o sistema operacional Linux 32bits (i386):

*fpc-2.4.4-XXXXXXXXX.i386.rpm*  
*fpc-src-2.4.4-XXXXXXXXX.i386.rpm*  
*lazarus-0.9.31.XXXXX-XXXXXXXXX.i386.rpm*

Para o sistema operacional Linux 64bits (x86\_64):

*fpc-2.4.4-XXXXXXXXX.x86\_64.rpm*  
*fpc-src-2.4.4-XXXXXXXXX.x86\_64.rpm*  
*lazarus-0.9.31.XXXXX-XXXXXXXXX.x86\_64.rpm*

**OBS.:** Você pode instalar a IDE Lazarus em quaisquer um dos outros sistemas operacionais e testar os exemplos do livro. Lembre-se, o lema do Lazarus é: *“Escreva uma vez e compile em qualquer lugar”*.

**NOTA:** Como alternativa, você pode, também, usar o seguinte link para baixar as compilações do Lazarus e do Free Pascal:  
<ftp://ftp.freepascal.org/pub/lazarus/snapshots/>

### Instalando o Lazarus no Windows

O Lazarus pode ser instalado no Windows 98SE, XP, Vista, Seven ou superior. Segue duas formas de realizar esta instalação:

#### Instalação Padrão (Recomendado):

A maneira mais fácil de instalar o Lazarus no Windows, é usando o instalador para esta plataforma, disponível no link citado no capítulo anterior.

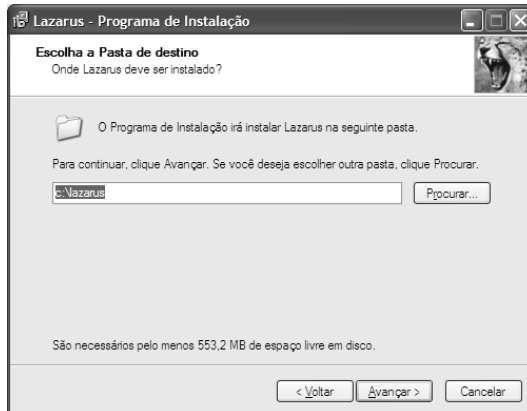
Ex.: **Lazarus-0.9.31-XXXXX-fpc-2.4.4-XXXXXXXXX-win32.exe**

Depois de executar o arquivo, selecione o idioma *Português (Brasil)* e surgirá a tela seguinte:



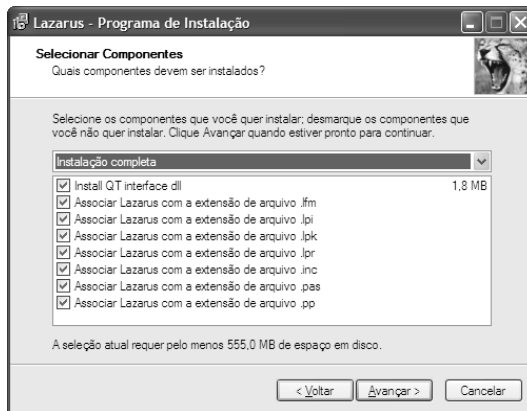
Figura 3.1 – Tela de Boas Vindas

Clique em avançar e na tela seguinte escolha a pasta de instalação do Lazarus. Dê preferência a pasta padrão e clique em avançar:



*Figura 3.2 – Escolha a Pasta de Destino*

Na tela seguinte escolha o tipo de instalação. Deixe em Instalação completa e todas as opções marcadas e clique em avançar:



*Figura 3.3 – Selecionar Componentes*

Nas próximas duas telas, escolha se você quer criar uma entrada no menu Iniciar e/ou um ícone no Desktop respectivamente, e na terceira seguinte clique em Instalar.



Depois da instalação completada, ao clicar no ícone do Lazarus, a IDE vai inicializar já com um novo projeto contendo um formulário vazio:

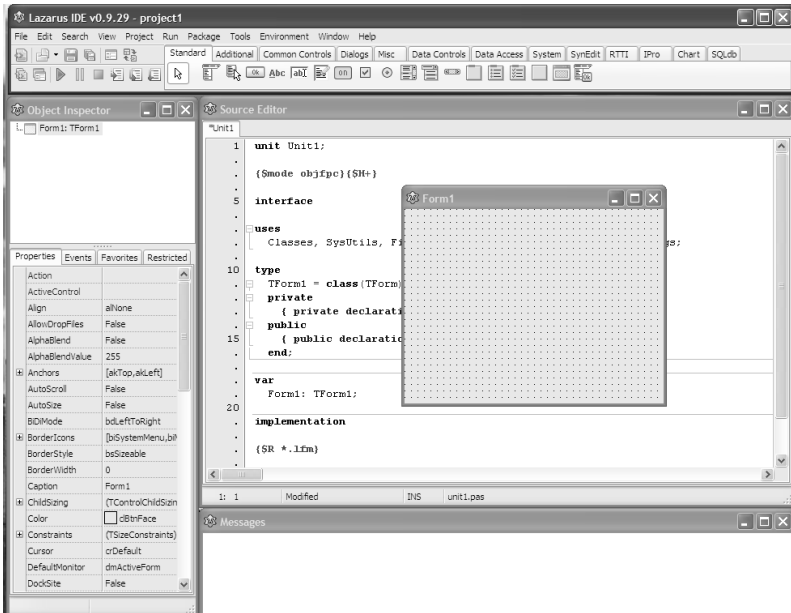


Figura 3.4 – IDE Lazarus no Windows XP

### Instalando usando o SVN (Avançado):

Para compilar e instalar o Lazarus e o Free Pascal a partir dos fontes disponíveis no repositório SVN, basta executar os procedimentos explicados a seguir:

**OBS.:** Se você pretende usar o Lazarus para programar para Windows Mobile, use os instaladores. Pois os passos mostrados aqui só incluem suporte a arquitetura Win32.

Primeiro, é preciso instalar o programa *TortoiseSVN*. Pode baixá-lo no link: <http://tortoisesvn.net/downloads.html>

Abra o *Windows Explorer*, e clique no menu *Arquivo > SVN Checkout...*

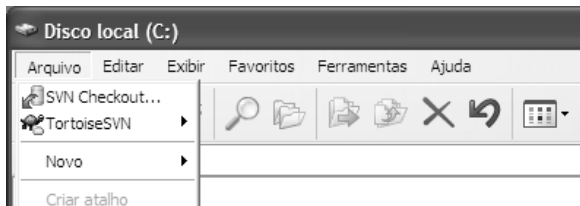


Figura 3.5 – Menu “SVN Checkout...” no Windows Explorer do XP

O diálogo que surge usaremos para baixar do SVN os arquivos necessários.

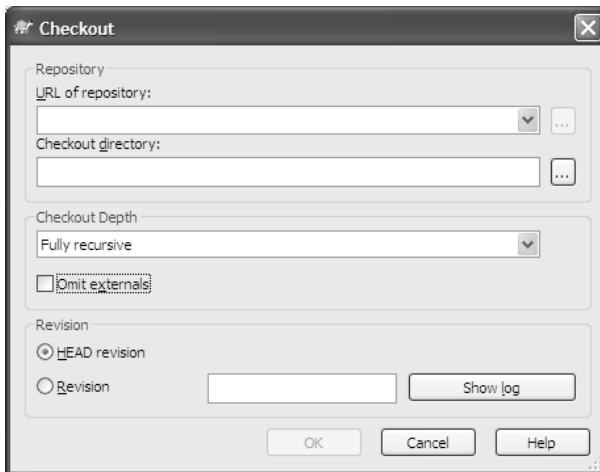


Figura 3.6 – Diálogo Checkout...



Em *URL of repository*: digite o caminho do SVN, em *Checkout diretory*: a pasta onde os arquivos serão salvos e clique em OK. Repita este processo para os seguintes caminhos:

Para baixar os arquivos necessários para compilação:

**URL of repository:** [http://svn.freepascal.org/svn/fpcbuild/branches/fixes\\_2\\_4/install/binw32](http://svn.freepascal.org/svn/fpcbuild/branches/fixes_2_4/install/binw32)

**Checkout Diretory:** `c:\freepascal\binutils\i386-win32\`

Para baixar o código fonte da versão em desenvolvimento do Free Pascal (neste caso 2.5.1):

**URL of repository:** <http://svn.freepascal.org/svn/fpc/trunk>

**Checkout Diretory:** `c:\freepascal\fpc\2.5.1\`

Para baixar o código fonte da versão em desenvolvimento do Lazarus:

**URL of repository:** <http://svn.freepascal.org/svn/lazarus/trunk>

**Checkout Diretory:** `c:\freepascal\laz`

Precisamos, agora, do compilador. Baixe-o usando o link: <ftp://ftp.freepascal.org/pub/fpc/dist/2.4.2/bootstrap/i386-win32-ppc386.zip>

Descompacte-o na pasta `c:\freepascal\binutils`.

Para iniciar a compilação do Free Pascal, é necessário criar um arquivo *BAT* com o nome *makefpc.bat* e salvá-lo em `c:\freepascal`. O arquivo deve ter o seguinte conteúdo (pode usar o *Bloco de Notas do Windows* para editar o arquivo):

```
@echo on
set myversion=2.5.1
set mypath=c:\freepascal\fpc\%myversion%
set mybinutils=c:\freepascal\binutils
set PATH=%mybinutils%\i386-win32;%mypath%\bin\i
386-win32;%PATH%
cd %mypath%
make clean all install INSTALL_PREFIX=%mypath%
PP=%mybinutils%\ppc386.exe
```

Salve e execute o arquivo. A compilação demora um pouco.

Agora, é necessário gerar um arquivo de configuração para o novo compilador criado. Este arquivo é o *fpc.cfg*, que estará direcionando para as pastas criadas. Assim, abra um *Prompt de Comando do Windows* e execute os dois comandos abaixo:

```
cd c:\freepascal\fpc\2.5.1\bin\i386-win32

fpcmkcfg -d basepath=c:\freepascal\fpc\2.5.1\bi
n\i386-win32 -o fpc.cfg
```

Certifique-se de que o arquivo *fpc.cfg* foi gerado na pasta *c:\freepascal\fpc\2.5.1\bin\i386-win32*.

Agora, para compilarmos o Lazarus, é necessário criar outro arquivo *BAT* com o nome *makelaz.bat* e salvá-lo em *c:\freepascal*. O arquivo deve ter o seguinte conteúdo:

```
@echo on
set myversion=2.5.1
set mypath=c:\freepascal\fpc\%myversion%
set mybinutils=c:\freepascal\binutils
set PATH=%mybinutils%\i386-win32;%mypath%\bin\i
386-win32;%PATH%
cd c:\freepascal\laz
```

```
make clean all OPT="-glw2"
```

Salve e execute o arquivo. A compilação demora um pouco.

Este arquivo só precisa ser usado uma vez. Para recompilar o Lazarus, use o menu *“Tools > Build Lazarus”* na própria IDE.

Para executar o Lazarus, siga para o caminho *c:\freepascal\lazarus* e execute o arquivo *lazarus.exe* .

# Instalando o Lazarus no Linux

Para instalações no Linux, recomendo usar o Ubuntu ou seus derivados. Foram feitos testes usando o Ubuntu 10.04, 10.10, 11.04 e Linux Mint 10.

## Instalando usando a Central de Programas do Ubuntu:

Seu computador deve estar conectado a internet.

Na área de trabalho do Ubuntu, clique no menu “*Aplicativos > Central de Programas do Ubuntu*”. Na parte superior à direita, no campo de pesquisa, digite *lazarus* e automaticamente será selecionado o pacote a ser instalado. Clique no botão Instalar.



*Figura 4.1 – Central de Programas do Ubuntu*

Terminada a instalação, a IDE estará disponível através do menu do Ubuntu: “*Aplicativos > Desenvolvimento > Lazarus*”.

**OBS.:** Esta é a versão 0.9.28-2 do Lazarus. Não é recomendado usar ela para acompanhar o livro, por ser muito antiga e não ser compatível com os novos temas do Ubuntu a partir da versão 10.04 (Ambiance e Radiance). A versão release 0.9.30 do Lazarus estará disponível no Ubuntu 11.10.

### Instalando usando os Pacotes (Recomendado):

Na sua “*Pasta pessoal*” no Ubuntu, crie uma pasta com o nome *pacotes\_lazarus*. Baixe e copie para esta pasta os três arquivos *RPM* necessários, de acordo com a arquitetura do Linux que está sendo usado, conforme já consideramos.

Exemplo de arquivos para Linux 32bits:

```
fpc-2.4.4-XXXXXXXXX.i386.rpm  
fpc-src-2.4.4-XXXXXXXXX.i386.rpm  
lazarus-0.9.31.XXXXXX-XXXXXXXXX.i386.rpm
```

Abra um terminal clicando no menu do Ubuntu: “*Aplicativos > Acessórios > Terminal*”. Em seguida, digite os seguintes comandos no terminal, lembrando de teclar *ENTER* no final de cada linha (informe sua senha de usuário se solicitado e aguarde o processamento de cada comando):

```
sudo apt-get update
```

```
sudo apt-get install alien
```

O comando a seguir instala pacotes adicionais necessários (só pressione a tecla *ENTER* depois de digitar *libvorbis-dev*):

```
sudo apt-get install libgtk2.0-dev liba52-0.7.4  
liba52-0.7.4-dev libdca-dev libdca0 libdts-dev
```

```
libmad0 libmad0-dev libmodplug-dev libogg-dev  
libvorbis-dev
```

Agora é preciso converter os pacotes *RPM*, que foram copiados para a pasta *pacotes\_lazarus*.

Esta conversão é necessária porque o Ubuntu trabalha com pacotes de instalação no formato *DEB*.

Então, primeiro vá para pasta *pacotes\_lazarus*, digitando no terminal (o til ( ~ ) é um atalho do Linux para “*Pasta pessoal*” do usuário):

```
cd ~/pacotes_lazarus
```

Agora, o comando a seguir pode demorar um bom tempo para terminar de executar (seja paciente). Se surgir a mensagem “*error: incorrect format: unknown tag*”, verifique se você baixou os pacotes de acordo com seu sistema Linux (32bits = i386 ou 64bits = x86\_64). Caso esteja certo, ignore o erro, que a conversão vai prosseguir normalmente. Segue o comando:

```
sudo alien -c *.rpm
```

Terminada a conversão dos três pacotes *RPM*, digite o comando abaixo para instalar os três pacotes *DEB* gerados:

```
sudo dpkg -i *.deb
```

Se correu tudo bem ☺, o Lazarus estará disponível através do menu do Ubuntu: “*Aplicativos > Desenvolvimento > Lazarus*”.

Ao iniciar o Lazarus pela primeira vez no Linux, pode aparecer o seguinte diálogo com os caminhos do Lazarus, Free Pascal e código fonte, basta clicar no botão *Start IDE* :

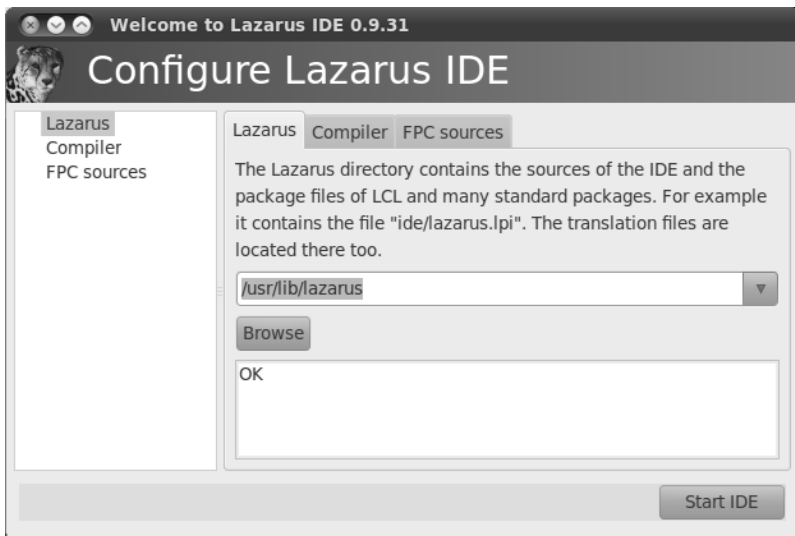


Figura 4.3 – Clique em Start IDE para Configurar o Lazarus

Assim, o Lazarus estará pronto para uso.

### Instalando usando o SVN (Avançado):

Para compilar e instalar o Lazarus e o Free Pascal no Linux, a partir dos fontes disponíveis no repositório SVN, siga os passos abaixo:

Primeiro, instalar o aplicativo necessário para baixar o código fonte e um compilador Free Pascal. Abra um *Terminal* e digite:

```
sudo apt-get install subversion fpc fpc-source
```

Agora execute o comando a seguir, para baixar o código fonte do Free Pascal do SVN (é só uma linha):

```
svn co http://svn.freepascal.org/svn/fpc/trunk  
~/fpc_fontes
```

Mude para pasta do código fonte com o comando abaixo:

```
cd ~/fpc_fontes
```

Se você estiver usando o Linux de 32bits, execute o comando de compilação abaixo:

```
make build OS_TARGET=linux CPU_TARGET=i386
```

Se você estiver usando o Linux de 64bits, execute o comando compilação abaixo:

```
make build OS_TARGET=linux CPU_TARGET=x86_64
```

Para gerar os binários para Linux de 32bits, execute o comando abaixo (é só uma linha):

```
make install OS_TARGET=linux CPU_TARGET=i386 PREFIX=~/fpc_svn
```

Para gerar os binários para Linux de 64bits, execute o comando abaixo (é só uma linha):

```
make install OS_TARGET=linux CPU_TARGET=x86_64 PREFIX=~/fpc_svn
```

Execute os comandos abaixo para remover o Free Pascal antigo e copiar os arquivos novos:

```
cd ~/fpc_svn/bin
```

```
sudo cp -v * /usr/bin
```

```
sudo rm -Rv /usr/lib/fpc/*
```



```
sudo cp -Rv ~/fpc_svn/lib/fpc/* /usr/lib/fpc

sudo ln -sf /usr/lib/fpc/2.5.1/ppc386 /usr/bin/ppc386

sudo rm -Rv /usr/share/fpcsrc/*

sudo mkdir /usr/share/fpcsrc/2.5.1

sudo cp -Rv ~/fpc_fontes/* /usr/share/fpcsrc/2.5.1
```

Agora é necessário criar o arquivo de configuração *fpc.cfg* . Assim, execute o comando abaixo:

```
sudo /usr/lib/fpc/2.5.1/samplecfg /etc
```

Precisamos editar o arquivo de configuração. Para isso, execute o comando:

```
sudo gedit /etc/fpc.cfg
```

No editor que surge, pressione *CTRL + H* e no diálogo seguinte, em “*Procurar por:*” digite **-Full** e em “*Substituir por:*” digite **-Fu/usr/lib/fpc/**, clique no botão “*Substituir todas*”, salve e feche o editor.

Voltando para o *Terminal*, execute o comando seguinte para baixar o código fonte do Lazarus (é só uma linha):

```
svn co http://svn.freepascal.org/svn/lazarus/trunk ~/lazarus_fontes
```

Mude para pasta do código fonte com o comando abaixo:

```
cd ~/lazarus_fontes
```

Para compilar e gerar o Lazarus, execute o comando abaixo:

```
make clean all
```

Os comandos seguintes criam links para o Lazarus:

```
sudo ln -s ~/lazarus_fontes/startlazarus /usr/bin/startlazarus
```

```
sudo ln -s ~/lazarus_fontes/lazarus /usr/bin/lazarus
```

Para criar um menu no Ubuntu para o Lazarus, clique no menu *Sistema > Preferências > Menu principal*. No diálogo que abrir, clique em *Desenvolvimento* (a esquerda) e clique no botão *Novo item*. Vai surgir o diálogo mostrado abaixo:



Figura 4.4 – Diálogo para Criar Item de Menu

Em **Nome:** digite *Lazarus*, em **Comando:** digite *startlazarus* e em **Comentário:** digite *IDE Lazarus*. Clique no botão com o ícone de lançador e localize o ícone do Lazarus no seguinte caminho: *[SUA\_PASTA\_PESSOAL]/lazarus\_fontes/images/ide\_icon48x48.png* e clique em *Abrir*. De volta ao diálogo, clique em *OK*.

Agora, o Lazarus estará disponível através do menu do Ubuntu: “*Aplicativos > Desenvolvimento > Lazarus*”.

### Breve Comparativo Lazarus e Delphi 7

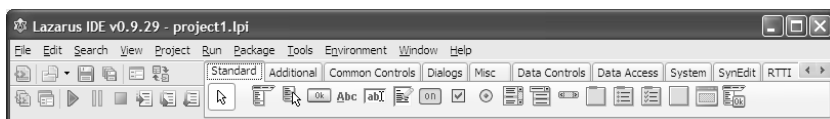
O Lazarus objetiva ser compatível, principalmente, com o Delphi 7. Então, vejamos algumas das diferenças e similaridades entre estas duas ferramentas.

Primeiramente, embora ambos usem praticamente a mesma sintaxe Object Pascal, não é possível abrir um projeto Delphi diretamente no Lazarus ou vice versa. Mas o Lazarus possui um recurso para converter (com limitações) projetos Delphi.

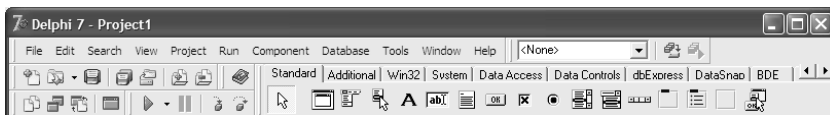
#### Extensões de arquivos:

Arquivo	Lazarus	Delphi
Projetos	.lpi e .lpr	.dpr
Pacotes	.lpk	.dpk
Formulários	.lfm ou .dfm	.dfm
Unidades	.pas ou .pp	.pas

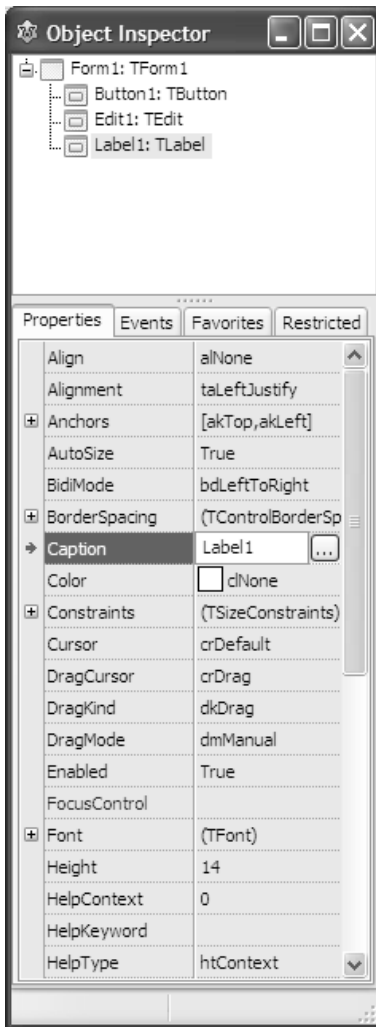
#### Comparativo de ferramentas:



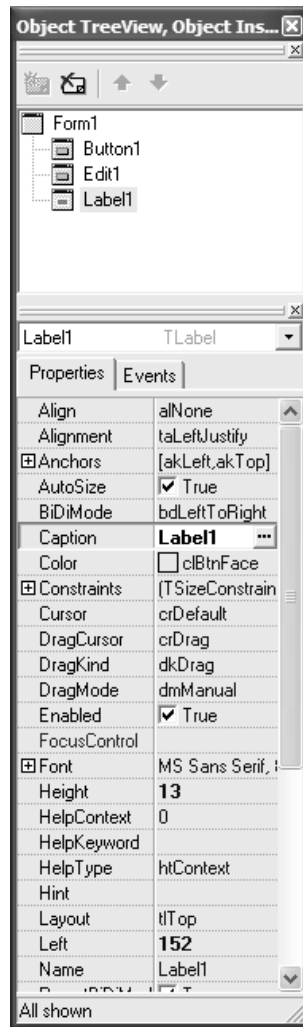
*Figura 5.1 – Menus, Ferramentas e Componentes do Lazarus*



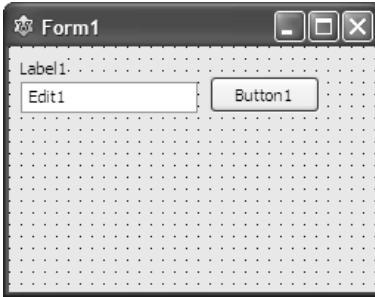
*Figura 5.2 – Menus, Ferramentas e Componentes do Delphi*



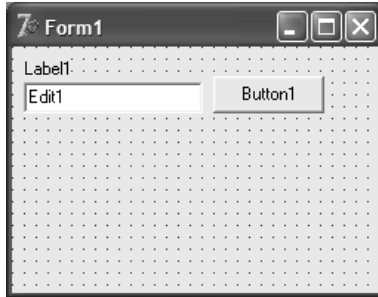
*Figura 5.3 – Object TreeView e Object Inspector do Lazarus*



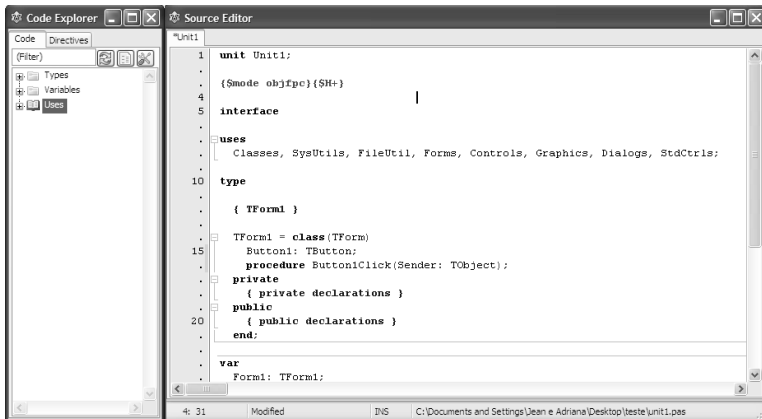
*Figura 5.4 – Object TreeView e Object Inspector do Delphi*



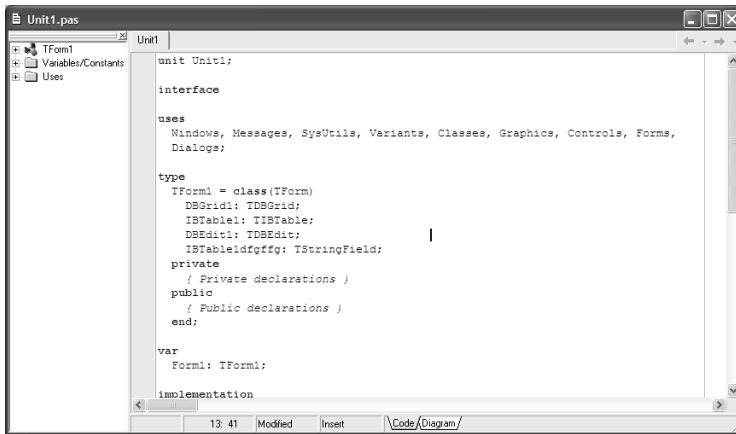
*Figura 5.5 – Exemplo de Formulário no Lazarus*



*Figura 5.6 – Exemplo de Formulário no Delphi*



*Figura 5.7 – Code Explorer e Source Editor do Lazarus*



*Figura 5.8 – Code Explorer e Source Editor do Delphi*

Para criar relatórios visualmente, o Delphi usa o QuickReport ou o Rave. Já o Lazarus, usa por padrão o LazReport (existe uma versão para Delphi chamada FreeReport) ou o FortesReport (também existe uma versão para Delphi).

### Componentes de Terceiros Gratuitos:

Segue algumas importantes bibliotecas de componentes de terceiros, com versões para Lazarus e Delphi:

**ZEOS:** A biblioteca ZeosLib contém componentes para conexão nativa aos principais bancos de dados existentes (Firebird, SQLite, MySQL, Oracle, PostgreSQL, MSSQL, etc). Agiliza a programação, deixa a instalação do seu sistema fácil e o mais importante, deixa seu sistema mais rápido. É OpenSource.

**ACBR:** Conjunto de componentes que permite acesso direto a Impressoras Fiscais, Gavetas de Dinheiro, Impressoras de Cheque, Transferência Eletrônica de Fundos (TEF), Monitor de Bombas de Combustível, Display de Mensagens, etc, sem

DLL's, interagindo direto na porta serial (ou outra porta que esteja ligado o equipamento). É OpenSource. Mais informações, acesse o site: [acbr.sourceforge.net/drupal/](http://acbr.sourceforge.net/drupal/) .

**ONGUARD:** Conjunto de componentes para proteger executáveis, criar versões demo e ou limitar a execução de um programa em rede ou por período. É OpenSource. Veja um exemplo de uso no link: <http://www.activedelphi.com.br/module.s.php?op=modload&name=News&file=article&sid=413> .

**POWER PDF:** Biblioteca de componentes para desenvolver visualmente relatórios diretamente em PDF. É OpenSource.

**OBS.:** Todos os pacotes de componentes citados, quando instalados no Lazarus, funcionarão tanto no Windows como no Linux.

### Velocidade de Compilação:

No passado, a compilação dos executáveis no Lazarus para Windows demorava muito mais do que no Delphi. Hoje a velocidade é quase igual. Basta seguir a seguinte técnica para compilar: pressione *Ctrl + F9* e depois de compilado, pode executar com *F9*. Também, no Lazarus, no menu *Run (Executar)*, temos a opção *"Quick compile" (Compilação rápida)*.

**ATENÇÃO:** Veja no *Capítulo 8* como resolver a questão de executáveis grandes no Lazarus.

Como vimos nesta breve comparação, os programadores Delphi, praticamente, se sentirão em casa trabalhando com o Lazarus. Evidentemente, o Lazarus tem suas manhas e macetes de uso, que são aprendidos com o tempo.



### Conhecendo e Configurando a IDE

O Lazarus depois de instalado corretamente, possui por padrão uma boa configuração. Mas podemos melhorar a IDE. E a medida que formos fazendo isso, vamos conhecer as principais características dela.

#### Mudando o idioma da IDE para Português do Brasil:

Clique no menu “*Tools > Options ...*”, e no diálogo que se abrirá (ajuste o tamanho para ver todas as opções), clique em *Desktop* na lista da esquerda, e, na direita, em *Language*, escolha *Portuguese [pb]*, clique no botão OK e reinicie a IDE.

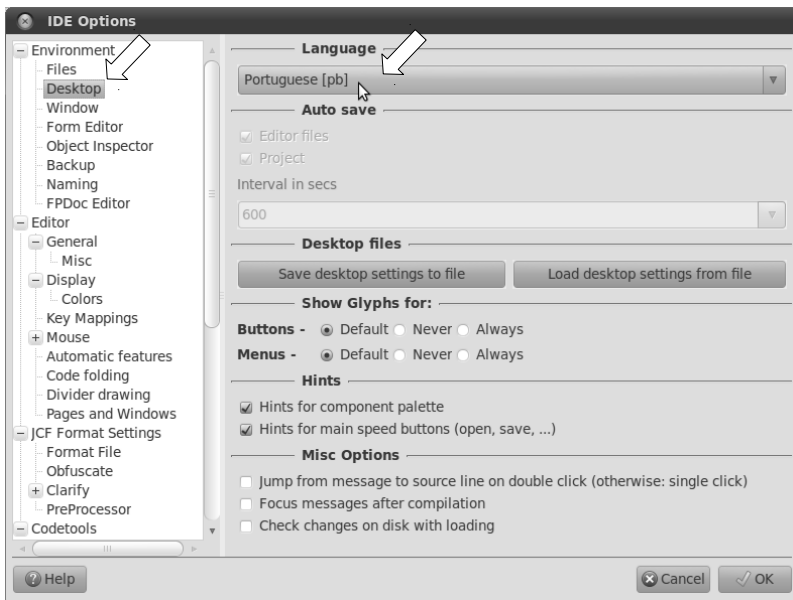


Figura 6.1 – Mudando o idioma para Português do Brasil

A partir deste ponto do livro, serão usadas referências a IDE em Português do Brasil, e as figuras e os exemplos terão como base o Lazarus para Linux. Mas, pode-se usar o Windows, pois serão citados quaisquer detalhes em relação a este sistema.

### Organizando as janelas da IDE:

Ao iniciar o Lazarus pela primeira vez, as janelas aparecem desorganizadas. A organização mostrada abaixo foi feita com a definição de vídeo 1280x800.

Também, foram adicionados dois novos diálogos, o *Explorador de Código* e o *Inspetor de Projetos*, que podem ser selecionados no menu *Exibir* e no menu *Projetos*, respectivamente.

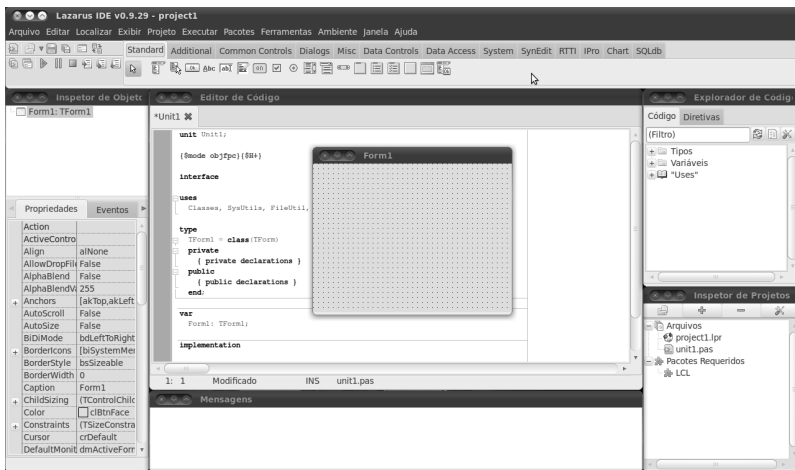
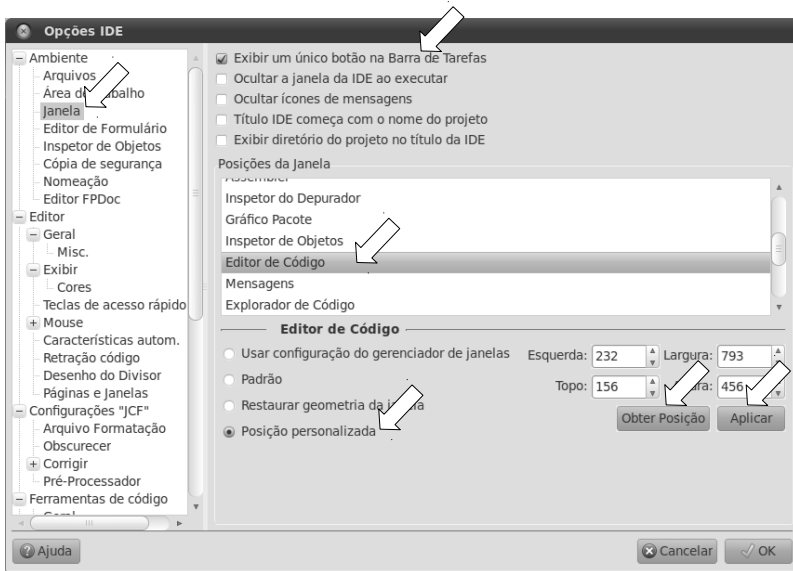


Figura 6.2 – Janelas Organizadas

Para que a janela do *Editor de Código* não mude de posição quando você fechar e reabrir a IDE, clique no menu *Ferramentas > Opções ...* ;

No diálogo que se abrirá, clique em *Janelas*, na lista da esquerda, e, na direita, em *Posições das Janelas*, escolha *Editor de Código*, marque *Posição personalizada*, clique no botão *Obter Posição*, clique no botão *Aplicar* e, aproveitando esta mesma tela, marque, bem acima, a opção “*Exibir um único botão na Barra de Tarefas*” (esta opção é muito útil se for usar o Lazarus no Linux) e clique em OK.



*Figura 6.3 – Configurando a posição do Editor de Código e a exibição de um único botão na Barra de Tarefas*

### Outras Configurações:

Abra novamente o diálogo “*Opções IDE*” (menu *Ferramentas > Opções ...*).

Na lista de categorias, à esquerda, você pode querer alterar as seguintes configurações:

**Ambiente => Arquivos:** Se você não quiser que na inicialização do Lazarus seja aberto o último projeto visualizado, desmarque “*Abrir o último projeto ao iniciar*”. Se quiser que apareça um diálogo de informações durante a compilação, marque “*Exibir diálogo de compilação*” e pode, também, marcar “*Auto fechar o diálogo de compilação*”.

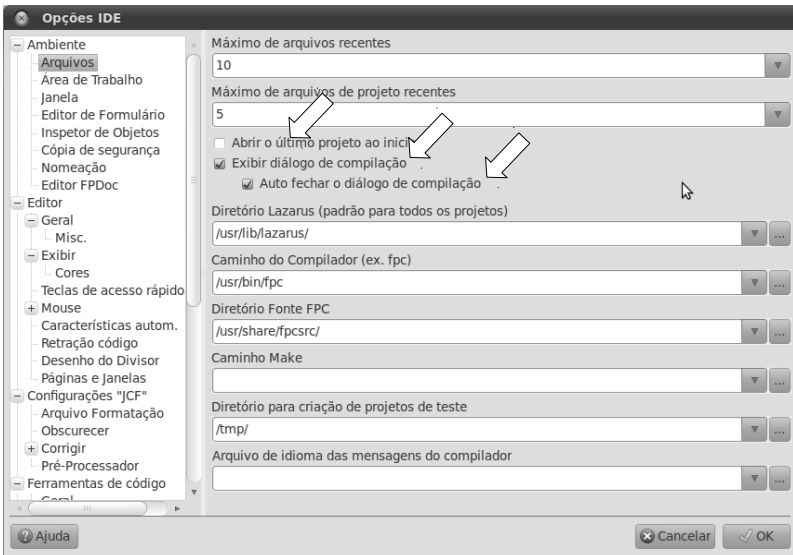


Figura 6.4 – Tela Ambiente => Arquivos

**Editor => Geral:** Tem a opção “*Tabulações endentam blocos*”, que, quando marcada, aumenta o recuo de um bloco de texto selecionado teclando *TAB* ou diminui teclando *SHIFT + TAB*. Também, recomendo deixar o valor 2 em “*Largura tabulações*”.

**Editor => Exibir:** Aqui você pode alterar o tamanho das fontes do editor de código em “*Altura da fonte do Editor*”. Se desejar ver o número das linha do editor, marque “*Exibir número de linhas*”.

**Editor => Exibir => Cores:** Aqui você pode definir o estilo de cores de destaque de sintaxe. Inclusive você pode mudar do estilo Default para o estilo Delphi.



Figura 6.5 – Estilo de Cores

**Editor => Complementos e Dicas:** Aqui é importante marcar *“Auto remover métodos vazios”*. Também, você pode diminuir o tempo para aparecer o auto completar código em *“Atraso para dicas e caixa de conclusão”* (sugiro deixar no mínimo).

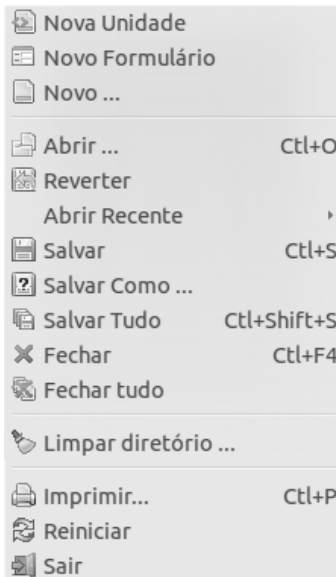
**Ferramentas de código => Espaço:** Para o código autogerado ficar mais legível, marque *símbolo* em *“Inserir espaço na frente de”* e em *“Inserir espaço após”*.

**Depurador => Geral:** Quando você interrompe a depuração de um projeto, aparece uma mensagem de parada um pouco incômoda. No próprio diálogo da mensagem já tem um botão para ela não aparecer novamente. Porém, se quiser desativá-la logo, desmarque *“Exibir mensagem na parada”*.

O diálogo *“Opções IDE”* possui várias outras possíveis configurações para o Lazarus. Fica a seu critério estudá-las.

### Conhecendo os Menus da IDE:

#### MENU ARQUIVO:



*Figura 6.6 – Menu Arquivo*

**Nova Unidade:** Cria um arquivo para nova unidade (Pascal Source).

**Novo Formulário:** Cria um novo formulário: uma janela visual e seu arquivo fonte Pascal associado.

**Novo ...:** Abre uma caixa de diálogo com uma variedade de novos tipos de arquivos e projetos para criar.

**Abrir ...:** Abre uma caixa de diálogo que habilita você a navegar pelos arquivos e escolher um arquivo existente para abrir.

**Reverter:** Remove as mudanças durante a edição e restaura o arquivo para seu estado original.

**Abrir Recente:** Fornece uma lista dos últimos arquivos abertos, possibilitando reabri-los facilmente.

**Salvar:** Salva o arquivo corrente (visível no editor), utilizando seu nome original de arquivo. Se este não tiver um nome, o sistema irá perguntar pelo nome que você deseja salvar (como no Salvar Como ...).

**Salvar Como ...:** Permite que você escolha um diretório e um nome de arquivo para salvar o arquivo atual.

**Salvar Tudo:** Salva todas as alterações feitas em todos os arquivos abertos no Editor de Código, utilizando os nomes originais dos arquivos. Se algum arquivo não tiver um nome, o sistema irá perguntar pelo nome que você deseja salvar (como no Salvar Como ...).

**Fechar:** Fecha o arquivo atual, perguntando se salva todas as mudanças do editor.

**Fechar tudo:** Fecha todos os arquivos abertos no Editor de Código, perguntando se salva as alterações para cada arquivo.

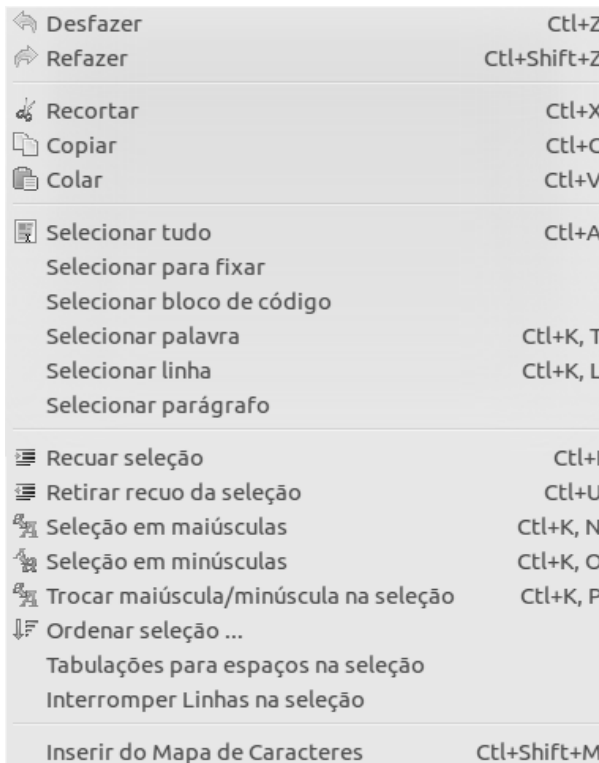
**Limpar diretório ...:** Disponibiliza um diálogo com uma série de filtros editáveis para remoção de arquivos do corrente diretório. Muito utilizado para remoção de arquivos .bak e outros temporários.

**Imprimir ...:** Permite imprimir o arquivo em uso no Editor de Códigos.

**Reiniciar:** Reinicia a IDE. Muito útil quando você instala novos componentes.

**Sair:** Sai do ambiente Lazarus, depois de perguntar se salva todos os arquivos editados.

### MENU EDITAR:



*Figura 6.7 – Menu Editar*

**Desfazer:** Desfaz a última ação de edição.



**Refazer:** Refaz a última ação de edição que foi revertida pelo Desfazer.

**Recortar:** Remove o texto ou item selecionado e o coloca na área de transferência.

**Copiar:** Faz uma cópia do texto selecionado, deixando o original no lugar, e coloca uma cópia na área de transferência.

**Colar:** Coloca o conteúdo da área de transferência na posição do cursor. Se texto tem sido selecionado na posição do cursor, o conteúdo da área de transferência irá substituir o texto selecionado.

**Selecionar tudo:** Seleciona todo texto.

**Selecionar para fixar:** o mesmo.

**Selecionar bloco de código:** o mesmo.

**Selecionar palavra:** o mesmo.

**Selecionar linha:** o mesmo.

**Selecionar parágrafo:** o mesmo.

**Recuar seleção:** Adiciona um número específico de espaços antes do texto selecionado.

**Retirar recuo da seleção:** Remove um número específico de espaços antes do texto selecionado.

**Seleção em maiúsculas:** Passa todo o texto selecionado para caixa alta, letras maiúsculas.

**Seleção em minúsculas:** Passa todo o texto selecionado para caixa baixa, letras minúsculas.

**Trocar maiúsculas/minúsculas na seleção:** Alterna entre maiúsculas e minúsculas o texto selecionado.

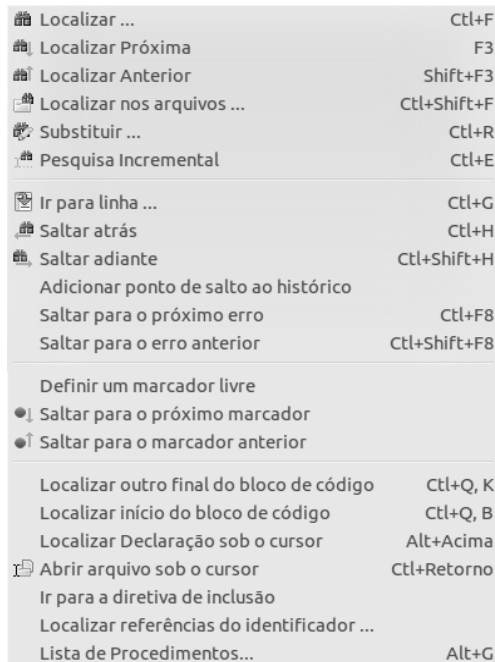
**Ordenar seleção ...:** Ordena linhas (ou palavras ou parágrafos) alfabeticamente; podendo ser: ascendente ou descendente; e sensível a caixa ou não. No meio do código fonte do programa, é claro, não faz sentido, mas se você tem uma lista em que você precisa ordená-la, você utilizará esta ferramenta.

**Tabulações para espaços na seleção:** Converte as tabulações (TAB) no texto selecionado em espaços. O número de espaços não tem quantidade fixa, mas é o número necessário para preencher a largura equivalente da tabulação.

**Interromper Linhas na seleção:** Se alguma linha no texto selecionado for maior que 80 caracteres ou o número especificado no diálogo “*Opções IDE*”, então a linha é quebrada no limite de palavras e continua na próxima linha.

**Inserir do Mapa de Caracteres:** Insere símbolos não existentes no teclado, como caracteres acentuados. Mostra um diálogo com o mapa de caracteres.

### MENU LOCALIZAR:



*Figura 6.8 – Menu Localizar*

**Localizar ...:** Similar a facilidade encontrada na maioria dos editores de texto. Uma caixa de diálogo aparece disponibilizando a entrada de um texto para procurar.

**Localizar Próxima, Localizar Anterior:** Procurar novamente pelo anterior texto encontrado, na especificada direção.

**Localizar nos arquivos ...:** Procura pelo texto nos arquivos. Mostra um diálogo com opções de procura.

**Substituir ...:** parecido com “Localizar”. Exibe um diálogo com lugares para entrar com o texto para procurar e substituir, e opções de procura.

**Pesquisa Incremental:** Procura pelo texto enquanto você está colocando os caracteres de procura. Exemplo: depois de você escolher “*Pesquisa Incremental*”, se você pressionar “p” o primeiro “p” irá ser marcado. Se, em seguida, você apertar “r”, o editor irá procurar o próximo “pr” e marcá-lo. E assim por diante.

**Ir para linha ...:** Move o cursor de edição para uma linha específica no arquivo.

**Saltar atrás:** Move de volta o arquivo para a próxima marcação (precisa ter usado “Adicionar ponto de salto ao histórico”). Irá mover para marcações em outros arquivos no editor.

**Saltar adiante:** Move para próxima marcação.

**Adicionar ponto de salto ao histórico:** Adiciona marcações ou pontos de pulo para o arquivo.

**Saltar para o próximo erro, Saltar para o erro anterior:** Move o cursor para linha do próximo erro reportado ou do anterior.

**Definir um marcador livre:** Define um marcador de texto, com o número de marcação seguinte ao último marcador. Dica: por meio do menu pop-up do editor (clique com o botão direito do mouse no texto) você tem mais opções de criação e navegação de marcadores de texto.

**Saltar para o próximo marcador, Saltar para o marcador anterior:** Salta para o marcador de texto seguinte ou anterior, respectivamente.

**Localizar outro final do bloco de código:** Se posicionado em um início de bloco, encontra o correspondente fim e vice-versa.

**Localizar início do bloco de código:** Move para o início do procedimento ou função em que o cursor encontra-se.

**Localizar Declaração sob o cursor:** Encontra o lugar em que o identificador selecionado é declarado. Pode ser o mesmo arquivo ou outro aberto no editor; se o arquivo não está aberto, ele será aberto (então se um procedimento ou função está declarado, por exemplo, no arquivo `classes.inc`, este irá ser aberto no editor).

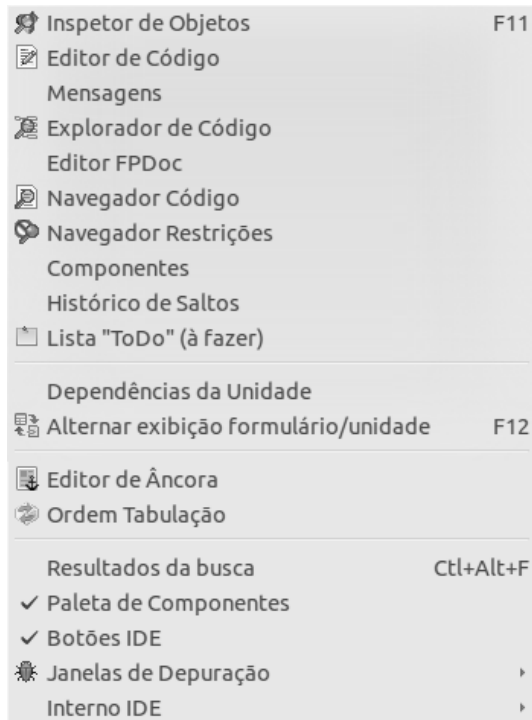
**Abrir arquivo sob o cursor:** Abre o arquivo, cujo o nome é selecionado no cursor. Útil para olhar arquivos incluídos ou que contêm outras unidades usadas no projeto.

**Ir para diretiva de inclusão:** Se o cursor for posicionado em um arquivo que esteja incluído em um outro arquivo, vai ao lugar no outro arquivo que chamou o arquivo incluído.

**Localizar referências do identificador ...:** Produz uma lista de todas as linhas no arquivo atual, ou o projeto atual ou em todos os arquivos anexados, em que um identificador é mencionado.

**Lista de procedimentos...:** Produz uma lista de todos os procedimentos e funções no arquivo atual, com os números de linha onde estão definidos.

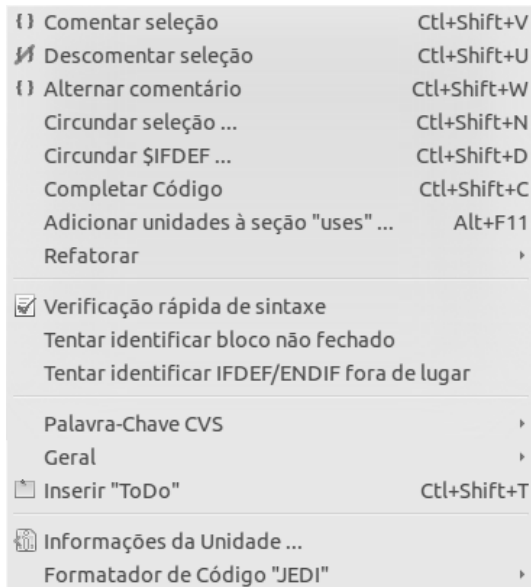
### MENU EXIBIR:



*Figura 6.9 – Menu Exibir*

Seria redundante comentar cada item deste menu. Pois, basicamente cada item exibe um diálogo correspondente ao nome do item (Por exemplo, o item *Mensagens* exibe o diálogo de mensagens de erro e de compilação). Estes diálogos são bem intuitivos e proveem recursos que facilitam a criação do código ou sua análise e, também, do design dos formulários. Mais a frente, comentaremos sobre alguns deles.

### MENU FONTE:



*Figura 6.10 – Menu Fonte*

**Comentar seleção:** Torna o texto selecionado um comentário inserindo antes de cada linha os caracteres: `//` .

**Descomentar seleção:** Remove as marcas de comentários.

**Alternar comentário:** Comenta ou descomenta um bloco selecionado.

**Circundar seleção ...:** Disponibiliza um menu flutuante com opções para incluir o texto selecionado dentro de uma das opções(begin ... end; try ... except; try ... finally; repeat ... until; { ... } etc).

**Circundar \$IFDEF...:** Circunda um bloco selecionado com diretivas \$IFDEF. Importante em programação multiplataforma.

**Completar Código:** Muito utilizado pelo desenvolvedor; permite completar o código de um método, procedimento ou função, declarado na interface. Ao chamar esta rotina, o “esqueleto” do procedimento ou função é criado na seção implementação (implementation) e o cursor posicionado neste.

**Adicionar unidades à seção “uses” ...:** Adiciona unidades do projeto à clausula *uses* da unidade de código atualmente exibida no editor de código.

**Refatorar:** Possui um submenu com ferramentas para refinar o código.

**Verificação rápida de sintaxe:** Verifica erros de sintaxe no código.

**Tentar identificar bloco não fechado:** Verifica blocos de código sem fechamento (*begin* sem o *end*).

**Tenta identificar IFDEF/ENDIF fora de lugar:** o mesmo.

**Palavra-Chave CVS:** Insere no código palavras chaves para serem usadas com CVS.

**Geral:** Insere no código avisos GPL, LGPL e entre outros.

**Inserir “ToDo”:** Provê um diálogo para inserir comentário sobre pendencias de código.

**Informações da Unidade ...:** Exibe um diálogo com informações sobre o arquivo da unidade de código atualmente exibida no editor de código.



**Formatador de Código “JEDI”:** Provê ferramentas para formatar o texto do código tornando-o mais legível.

### MENU PROJETO:



*Figura 6.11 – Menu Projeto*

**Novo Projeto ...:** Cria um novo projeto. Exibe um diálogo que possibilita escolher entre diversos tipos de projetos.

**Novo projeto do arquivo ...:** Permite que você selecione um arquivo a partir do qual será criado um novo projeto.

**Abrir projeto ...:** Abra um projeto salvo.

**Abrir Projeto Recente:** Exibe uma lista com os projetos recentemente trabalhados.

**Fechar Projeto:** Fecha o projeto atual.

**Salvar Projeto:** Salva todos os arquivos do projeto.

**Salvar Projeto Como ...:** Permite salvar o projeto com outro nome.

**Publicar Projeto ...:** Cria uma cópia de todo o projeto para envio.

**Inspetor de Projetos:** Exibe um diálogo com a estrutura do projeto.

**Opções de Projetos ...:** Exibe um diálogo com diversas opções de configurações do projeto.

**Adicionar arquivo editor ao Projeto:** Adiciona ao projeto o arquivo atualmente exibido no editor de código.

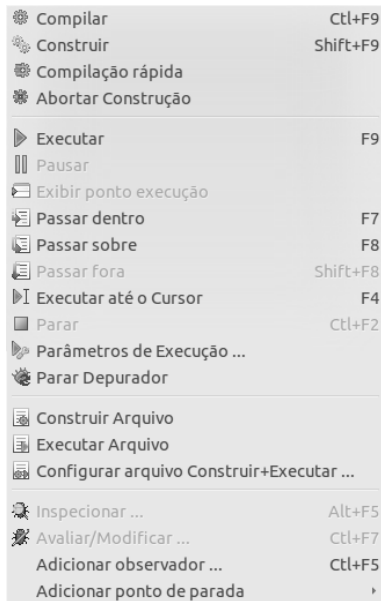
**Remover do Projeto ...:** Exibe um diálogo com arquivos que podem ser removidos do projeto.

**Unidades ...:** Exibe um diálogo com as unidades de código do projeto.

**Formulários ...:** Exibe um diálogo com os formulários do projeto.

**Exibir Fonte:** Exibe o código fonte do projeto.

### MENU EXECUTAR:



*Figura 6.12 – Menu Executar*

**Compilar:** Compila as alterações no projeto e gera o executável.

**Construir:** Recompila todo o projeto e gera o executável.

**Compilação rápida:** Efetua uma compilação rápida.

**Abortar Construção:** Interrompe a compilação do projeto.

**Executar:** Compila o projeto e inicia a execução do mesmo vinculado a IDE Lazarus.

**Pausar:** Suspende a execução do programa.

**Exibir ponto de execução:** Coloca o cursor no ponto de execução corrente.

**Passar dentro:** Executa o programa um passo de cada vez, até um ponto marcado no código fonte.

**Passar sobre:** Executa o programa até a instrução marcada no código, ignora esta instrução e prossegue com a execução do programa.

**Passar fora:** Executa o programa um passo de cada vez, até um ponto marcado no código fonte incluindo unidades de código dependentes.

**Executa até o Cursor:** Executa o programa até a posição do curso no código e continua a execução clicando no botão Executar.

**Parar:** Interrompe a execução do programa.

**Parâmetros de Execução ...:** Exibe um diálogo que permite configurar parâmetros e variáveis de ambiente para o programa em execução.

**Parar Depurador:** Interrompe a verificação de erros em tempo de execução.

**Construir Arquivo:** Compila apenas a unidade de código atualmente exibida no editor de código.

**Executar Arquivo:** Executa apenas a unidade de código atualmente exibida no editor de código.

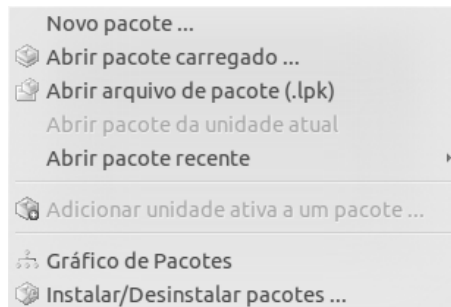
**Configurar arquivo Construir+Executar ...:** Exibe um diálogo que permite configurar a construção e/ou execução de um único arquivo.

**Inspecionar ...:** Exibe diálogo “Inspetor do Depurador” de erros.

**Avaliar/Modificar ...:** Exibe diálogo “Avaliar/Modificar” do depurador de erros.

**Adicionar observador ...:** Exibe diálogo “Propriedades observador” do depurador de erros.

### MENU PACOTE:



*Figura 6.13 – Menu Pacote*

**Novo pacote ...:** Cria um novo pacote de componentes.

**Abrir pacote carregado ...:** Exibe uma lista de pacotes instalados, possibilitando abrir um ou mais deles.

**Abrir arquivo de pacote (.lpk):** Permite abrir um arquivo de pacote.

**Abrir pacote da unidade atual:** Abre o pacote de componentes ao qual pertence a unidade de código exibida no editor de código.

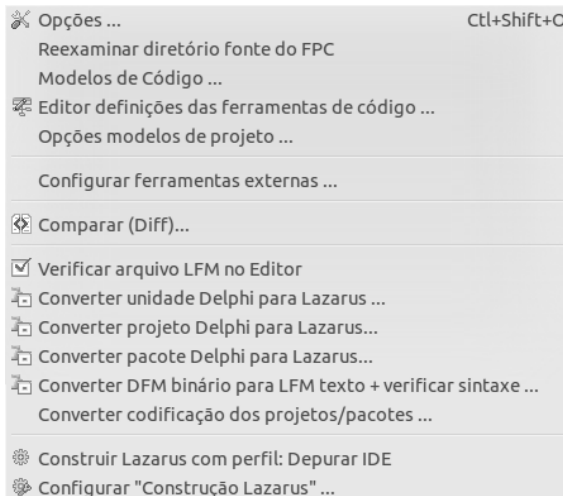
**Abrir recente Pacote:** Exibe uma lista com os pacotes abertos recentemente.

**Adicionar unidade ativa a um pacote ...:** Adiciona o arquivo da unidade de código exibida no editor de código.

**Gráfico de Pacotes:** Exibe uma árvore hierárquica dos pacotes atualmente instalados e suas dependências.

**Instalar/Desinstalar pacotes ...:** Exibe um diálogo que permite instalar novos pacotes (geralmente pacotes adicionais presentes na distribuição do Lazarus, mas não instalados por padrão) ou desinstalar pacotes.

### MENU FERRAMENTAS:



*Figura 6.14 – Menu Ferramentas*

**Opções ...:** Exibe um diálogo que permite realizarmos diversas configurações da IDE Lazarus.

**Reexaminar diretório fonte do FPC:** Reexamina o diretório que possui os arquivos de código fonte do Free Pascal para efetivar alguma atualização.

**Modelos de Código:** Exibe um diálogo para criar/editar modelos de código.

**Editor definições das ferramentas de código ...:** Exibe um diálogo com todas as definições do Free Pascal, diretório fonte Lázaró, outros diretórios e diretórios de pacotes.

**Opções modelos de projeto:** Exibe um diálogo para definição do caminho dos modelos de projeto.

**Configurar ferramentas externas ...:** Exibe um diálogo que permite adicionar diversas ferramentas externas (geralmente macros) ao conjunto de ferramentas do Lazarus.

**Comparar (Diff)...**: Exibe um diálogo que permite a comparação entre os dois arquivos em busca de diferenças.

**Converter unidade Delphi para Lazarus ...:** Exibe um diálogo que permite converter unidades de código escritas no Object Pascal do Delphi em unidades de código Free Pascal/Lazarus.

**Converter projeto Delphi para Lazarus ...:** Exibe um diálogo que permite converter um projeto desenvolvido no Delphi em um projeto Lazarus.

**Converter pacote Delphi para Lazarus ...:** Exibe um diálogo que permite converter um pacote de componentes

desenvolvido para o Delphi em um pacote de componentes compatível com o Lazarus.

### **Converter DFM binário para LFM texto + verificar sintaxe**

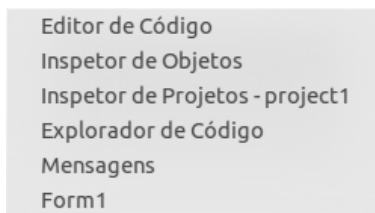
...: Exibe um diálogo que permite converter arquivos DFM do Delphi em arquivos LFM do Lazarus.

**Converter codificação dos projetos/pacotes ...:** Converte a codificação da tabela de caracteres dos arquivos de projetos ou pacotes.

**Construir Lazarus com perfil: Depurar IDE:** Constrói a IDE Lazarus com a configuração atual.

**Configurar “Construção Lazarus” ...:** Exibe um diálogo com opções de configuração da construção da IDE Lazarus.

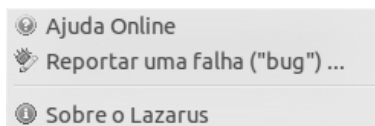
### **MENU JANELAS:**



*Figura 6.15 – Menu Janelas*

Exibe lista das janelas abertas.

### **MENU AJUDA:**



*Figura 6.16 – Menu Ajuda*



**Ajuda Online:** Abre a ajuda online no navegador padrão.

**Reportar uma falha (“bug”) ...:** Exibe a página Wiki do Lazarus sobre como relatar um bug.

**Sobre o Lazarus:** Exibe um dialogo com algumas informações sobre Lázaro.

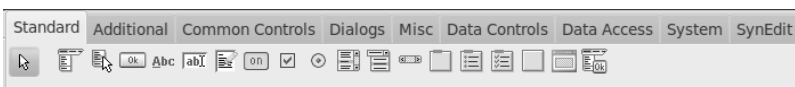
### Barra de Ferramentas Rápidas:



*Figura 6.17 – Barra de Ferramentas*

	Nova Unidade de Código		Abrir Arquivos
	Salvar Arquivo Atual		Salvar Todos os Arquivos
	Novo Formulário		Alternar Formulário/Unidade
	Exibir Unidades do Projeto		Exibir Formulários do Projeto
	Executar Projeto		Pausar Depuração
	Parar Depuração		Passar Para Próxima Linha
	Passar sobre próxima Linha		Executar até retorno da função

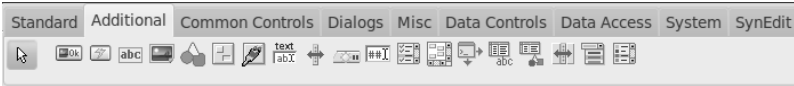
### Conhecendo as Paletas de Componentes:



*Figura 6.18 – Paleta Standard*

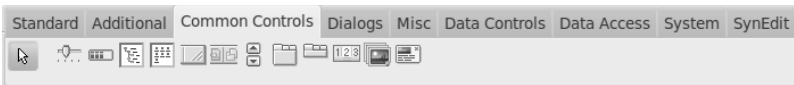
**Componentes básicos:** TMainMenu, Tpop-upMenu, TButton, TLabel, TEdit, TMemo, TToggleBox, TCheckBox, TRadioButton,

TListBox, TComboBox, TScrollBar, TGroupBox, TRadioGroup, TCheckGroup, TPanel, TFrame e TActionList .



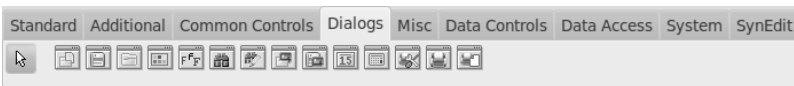
*Figura 6.19 – Paleta Additional*

**Componentes Adicionais:** TBitBtn, TSpeedButton, TStaticText, TImage, TShape, TBevel, TPaintBox, TNotebook, TLabelledEdit, TSplitter, TTrayIcon, TMaskEdit, TCheckListBox, TScrollBar, TApplicationProperties, TStringGrid, TDrawGrid, TPairSplitter, TColorBox e TColorListBox.



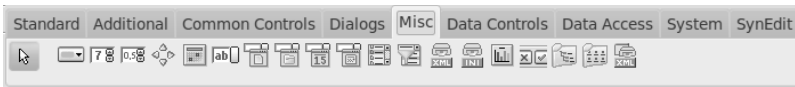
*Figura 6.20 – Paleta Common Controls*

**Componentes de Controles Comuns:** TTrackBar, TProgressBar, TTreeView, TListView, TStatusBar, TToolBar, TUpDown, TPageControl, TTabControl, THeaderControl, TImageList e Tpop-upNotifier.



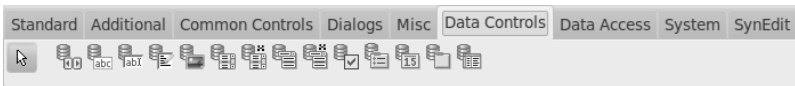
*Figura 6.21 – Paleta Dialogs*

**Componentes de Diálogos:** TOpenDialog, TSaveDialog, TSelectDirectoryDialog, TColorDialog e TFontDialog, TFindDialog, TReplaceDialog, TOpenPictureDialog, TsavePictureDialog, TCalendarDialog, TCalculatorDialog, TPrinterSetupDialog, TPrintDialog e TPageSetupDialog.



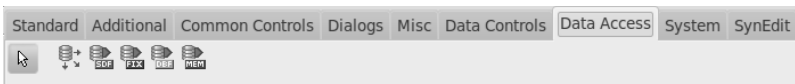
*Figura 6.22 – Paleta Misc*

**Outros Componentes:** TColorButton, TSpinEdit, TFloatSpinEdit, TArrow, TCalendar, TEditButton, TFileNameEdit, TDirectoryEdit, TDateEdit, TCalcEdit, TFileListBox, TFilterComboBox, TXMLPropStorage, TIniPropStorage, TBarChart, TButtonPanel, TShellTreeView, TShellListView e TIDDialogLayoutStorage.



*Figura 6.23 – Paleta Data Controls*

**Componentes de Controle para Dados:** TDBNavigator, TDBText, TDBEdit, TDBMemo, TDBImage, TDBListBox, TDBComboBox, TDBCheckBox, TDBRadioGroup, TDBCalendar, TDBGGroupBox e TDBGGrid.



*Figura 6.24 – Paleta Data Access*

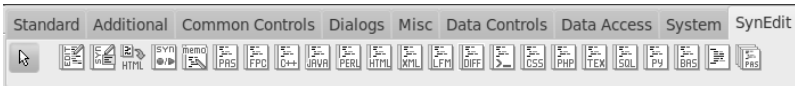
**Componentes de Acesso a Dados:** TDataSource, TMemDataSet, TSdfDataSet, TFixedFormatDataSet e TDbf.



*Figura 6.25 – Paleta System*

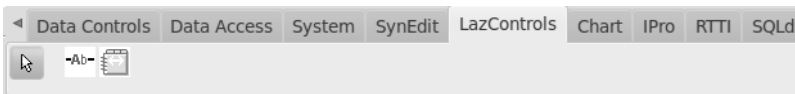
**Componentes de Sistema:** TTimer, TIdleTimer, TLazComponentQueue, THTMLHelpDatabase, THTMLBrowserHelpViewer, TAsyncProcess, TProcessUTF8,

TProcess, TSimpleIPCClient, TSimpleIPCServer, TXMLConfig e TEventLog.



*Figura 6.26 – Paleta SynEdit*

*Componentes para Editores de Código:* TSynEdit, TSynAutoComplete, TSynExporterHTML, TSynMacroRecorder, TSynMemo, TSynPasSyn, TSynFreePascalSyn, TSynCppSyn, TSynJavaSyn, TSynPerlSyn, TSynHTMLSyn, TSynXMLSyn, TSynLFMSyn, TSynDiffSyn, TSynUNIXShellScriptSyn, TSynCssSyn, TSynPHPSyn, TSynTeXSyn, TSynSQLSyn, TSynPythonSyn, TSynVBSyn, TSynAnySyn, TSynMultiSyn.



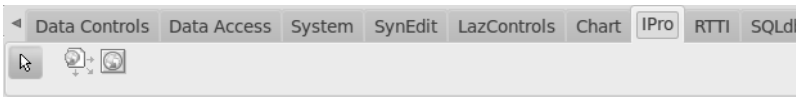
*Figura 6.27 – Paleta LazControls*

*Componentes de Controle do Lazarus:* TDividerBevel e TExtendedNotebook.



*Figura 6.28 – Paleta Chart*

*Componentes para Gráficos:* TDbChartSource, TChart, TListCartSource, TRandomChartSource, TDateTimeIntervalChartSource, TUserDefinedChartSource, TCalculatedChartSource, TChartToolset, TChartAxisTransformations, TChartStyles e TChartLegendPanel.



*Figura 6.29 – Paleta IPro*

*Componentes IPro: TIpFileDataProvider, TIpHtmlPanel.*



*Figura 6.30 – Paleta RTTI*

*Componentes RTTI: TTIEdit, TTIComboBox, TTIButton, TTICheckBox, TTILabel, TTIGroupBox, TTIRadioGroup, TTICheckGroup, TTICheckListBox, TTIListBox, TTIMemo, TTICalendar, TTImage, TTIFloatSpinEdit, TTISpinEdit, TTITrackBar, TTIProgressBar, TTIMaskEdit e TTIColorButton, TMultiPropertyButton, TPropertyGrid e TTIGrid.*



*Figura 6.31 – Paleta SQLdb*

*Componentes para Conexão a Bancos de Dados: TSQLQuery, TSQLTransaction, TSQLScript, TSQLConnector, TPQConnection, TOracleConnection, TODBCConnection, TMySQL40Connection, TMySQL41Connection, TMySQL50Connection, TSQLite3Connection e TIBConnection.*

Como você deve ter notado, o Lazarus possui muitos recursos e componentes para criação de todo tipo de aplicações. E como você verá no próximo capítulo, o Lazarus pode tornar-se ainda mais turbinado com a adição de componentes que vem com ele, mas não são instalados por padrão, e componentes de terceiros que encontramos na Internet gratuitamente.

### Teclas de Atalho do Lazarus:

Basicamente, o Lazarus tem as mesmas teclas de atalho do Delphi 7. Segue as que são usadas com mais frequência:

**CTRL + S** – Salva as alterações do arquivo corrente.

**CTRL + SHIFT + S** – Salva as alterações de todos os arquivos abertos no Editor de Código.

**F2** – Abre um diálogo com opções para renomear uma variável ou objeto.

**F12** – Alterna o foco entre o *Editor de Código* e o *Design do Formulário* e vice-versa.

**F11** – Muda o foco para o Inspetor de Objetos.

**CTRL + SETAS** (as setas para esquerda, cima, baixo, direita) – Quando um componente no formulário está selecionado, pode ser movido de posição com esta combinação de teclas.

**SHIFT + SETAS** (as setas para esquerda, cima, baixo, direita) – Quando um componente no formulário está selecionado, pode ser redimensionado com esta combinação de teclas.

**CTRL + F2** – Se acontecer de, ao executarmos nosso aplicativo, ele travar, não podendo ser fechado. Basta clicar na janela principal do Lazarus e usar **CTRL + F2** para encerrar o processo do aplicativo.

**CTRL + SHIFT + C** – Quando estamos na seção interfaces criando o cabeçalho de um procedimento e usarmos esta

combinação de teclas, o Lazarus cria automaticamente o “esqueleto” do procedimento na sessão *implementation*.

**CTRL + ESPAÇO** – Chama o painel de Auto Completar Código. Por exemplo, se você digitar “m” e pressionar CTRL + Espaço vai aparecer uma lista com todos os métodos e funções disponíveis que comecem com esta letra.

**CTRL + SHIFT + ESPAÇO** – Se você estiver digitando um procedimento ou função, e não souber os parâmetros que ela aceita, basta posicionar o cursor entre os parênteses do procedimento ou função e pressionar estas teclas e o Lazarus mostrará os parâmetros aceitos.

**CTRL + J** – Mostra uma lista de estruturas de código prontas para uso.

**CTRL + W** – Mostra uma lista com todas as palavras usadas no seu código.

**CTRL + /** – Comenta ou Descomenta a linha sob o cursor ou um bloco de texto selecionado (funciona só no Linux).

**CTRL + F** – Abre o diálogo de pesquisa e/ou substituição de texto.

**CTRL + D** – Formata o código da *unit* atual, ou seja, endenta o código no padrão Object Pascal.

**CTRL + F12** – Mostra as unidades de código associadas ao projeto. Possibilitando selecionar para edição.

**SHIFT + F12** – Mostra os formulários associados ao projeto. Possibilitando selecionar para edição.

**ALT + F11** – Diálogo para inserir *unit's* do projeto na *unit* em exibição no *Editor de Código* (só na versão 0.9.31 ou superior).



# Instalando Novos Componentes

No capítulo anterior, vimos que o Lazarus possui um leque vasto de componentes. Mas você deve ter notado a falta de componentes para geração visual de relatórios. E, embora o Lazarus venha com a paleta SQLdb, para conexões com bancos de dados, esta se torna um pouco trabalhosa para projetos grandes. Aí que entra o ZEOS, uma paleta de componentes para conexão a diversos bancos de dados com muitos recursos que facilitam e agilizam o desenvolvimento de aplicativos. Neste capítulo, vamos aprender a instalar estes e outros componentes no Lazarus. (É importante a instalação do ZEOS, PowerPDF e LazReport, pois serão usados no livro).

## Preparativos para Instalar os Componentes:

Primeiramente, o processo de instalação dos componentes é o mesmo no Lazarus para Linux ou para Windows.

Mas há um detalhe quanto a instalação do Lazarus no Linux. Devido ao modo recomendado para instalar o Lazarus 0.9.31 no Ubuntu Linux (usando os pacotes *RPM* convertidos para *DEB*), precisamos mudar as permissões da pasta de instalação do Lazarus para leitura e escrita.

Para isso, certifique-se de que o Lazarus esteja fechado e na Área de Trabalho do Ubuntu, abra um terminal clicando no menu *Aplicativos > Acessórios > Terminal*, digite o comando abaixo e pressione *ENTER* (digite sua senha de usuário, se for perguntado).

```
sudo chmod -R 777 /usr/lib/lazarus
```

Vamos colocar as pastas dos componentes de terceiro em uma pasta chamada *lazcomponentes* (pode ser qualquer outro nome, **mas não use espaços, letras acentuadas, traço e caracteres especiais, pode usar underline “\_”**). No Linux, crie esta pasta na sua *Pasta pessoal* e, se você estiver no Windows, crie a pasta no disco do sistema ( Por exemplo, C: ).

Agora, estamos prontos para iniciar as instalações.

### Instalando os Componentes do PowerPDF:

É necessário instalarmos este conjunto de componentes, pois o recurso do LazReport para exportar relatórios para PDF, depende dele.

Primeiro, baixe o PowerPDF no link: [http://sourceforge.net/projects/lazarus-ccr/files/PowerPDF/PowerPDF%20v0.9.6/powerpdf\\_0.9.6\\_20101201.tar.gz/download](http://sourceforge.net/projects/lazarus-ccr/files/PowerPDF/PowerPDF%20v0.9.6/powerpdf_0.9.6_20101201.tar.gz/download)

Descompacte dentro da pasta *lazcomponentes* .

Estando na IDE Lazarus, clique no menu “*Arquivo > Fechar tudo*” (se perguntado se quer salvar o projeto, não salve).

Clique no menu “*Pacotes > Abrir arquivo de pacote (.lpk)*”.

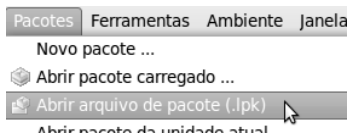


Figura 7.1 – Abrir Pacote de Componentes

Localize na pasta do PowerPDF (que foi descompactada dentro da pasta *lazcomponentes*) o arquivo *pack\_powerpdf.lpk* . Abra o arquivo.

Vai surgir o diálogo mostrado abaixo:

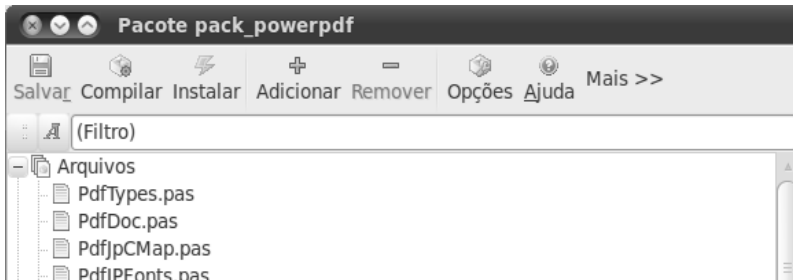


Figura 7.2 – Compilação e Instalação do PowerPDF

Clique no botão *Compilar*. Na janela de mensagens do Lazarus, irá aparecer uma série de mensagens de compilação. Quando surgir a mensagem “*Compiling package pack\_powerpdf 0.9.6*” *completado* (ou algo parecido), a compilação do pacote terminou. Agora, você pode clicar no botão *Instalar*. Vai surgir a seguinte mensagem:

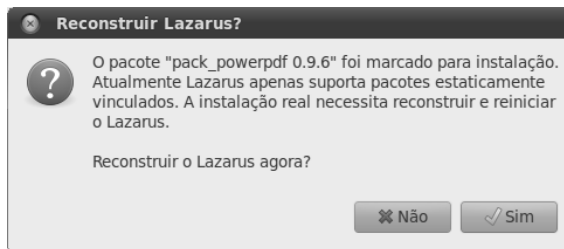
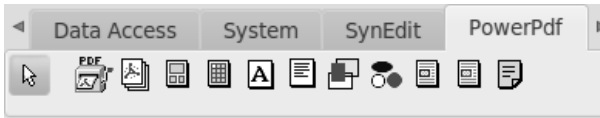


Figura 7.3 – Reconstruir Lazarus

A mensagem já diz tudo, então, clique em *Sim*.

Novamente, aparecerá algumas mensagens na janela de mensagens. Se depois que surgir a mensagem “*IDE*” *completado*, a IDE não reiniciar automaticamente, clique no menu *Arquivo > Reiniciar*. Se for perguntado se deseja salvar o projeto, não salve.

Depois que o Lazarus reiniciar, você terá a disposição a paleta abaixo:



*Figura 7.4 – Paleta do PowerPdf*

*Componentes do PowerPDF:* TPRReport, TPRPage, TPRLayoutPanel, TPRGridPanel, TPRLabel, TPRText, TPRRect, TPREllipse, TPRIImage, TPRJpegImage, TPRAnnotation.

Na pasta do PowerPDF, você encontra exemplos de uso dos componentes e manuais de referência.

### **Instalando os Componentes do ZEOS:**

O procedimento de instalação do ZEOS é parecido com a instalação do PowerPDF, mas fique atento aos detalhes que diferem.

Primeiro, baixe o ZEOS no link: <http://sourceforge.net/project/s/zeoslib/files/Zeos%20Database%20Objects/zeosdbo-6.6.6-stable/ZEOSDBO-6.6.6-stable.zip/download>

Dentro da pasta *lazcomponentes* crie uma pasta chamada *zeos*, descompacte o ZEOS dentro desta nova pasta.

Se não estiver executando, execute o Lazarus.

Clique no menu “Arquivo > Fechar tudo” (se perguntado se quer salvar o projeto, não salve).

Clique no menu *Pacotes > Abrir arquivo de pacote (.lpk)*.

No diálogo seguinte, localize na pasta *zeos* a pasta *packages*, abra esta, e dentro dela, abra a pasta *lazarus* e localize o arquivo *zcomponent.lpk*. Abra o arquivo. Vai surgir o diálogo de instalação de pacotes.

Clique no botão *Compilar*. Na janela de mensagens do Lazarus, irá aparecer uma série de mensagens de compilação. Quando surgir a mensagem “*Compiling package zcomponent 6.6.6*” *completado* (ou algo parecido), a compilação do pacote terminou. Agora, você pode clicar no botão *Instalar*. Vai surgir a seguinte mensagem:

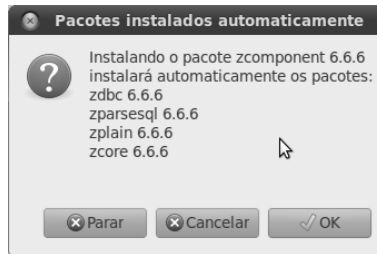


Figura 7.5 – Pacotes Auxiliares

Clique em *OK*, para confirmar a instalação dos pacotes auxiliares. Vai surgir a mensagem para reconstruir o Lazarus. Clique em *Sim*.

Novamente, aparecerá algumas mensagens na janela de mensagens. Se depois que surgir a mensagem “*IDE*” *completado*, a IDE não reiniciar automaticamente, clique no menu *Arquivo > Reiniciar*. Se for perguntado se deseja salvar o projeto, não salve.

Depois que o Lazarus reiniciar, você terá a disposição a paleta abaixo:

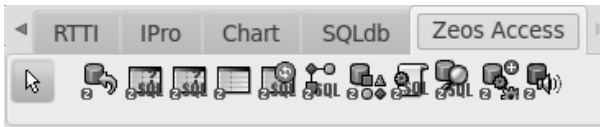


Figura 7.6 – Paleta do Zeos Access

*Componentes do ZEOS:* TZConnection, TZReadOnlyQuery, TZQuery, TZTable, TZUpdateSQL, TZStoredProc, TZSQLMetadata, TZSQLProcessor, TZSQLMonitor, TZSequence, TZIBEventAlerter

### Instalando os Componentes do Fortes Report:

O Fortes Report para Lazarus, durante a edição deste livro, encontra-se na versão 3.24-LCL-R2 que funciona muito bem no Linux e no Windows (o uso desta biblioteca de componentes não é abordado neste livro, a instalação dela é importante para desenvolvedores Delphi que já a usavam e estão migrando para o Lazarus).

Primeiro, baixe o Fortes Report em: <http://sourceforge.net/projects/fortes4lazarus/files/fortesreport-3.24-LCL-R2.zip/download>

Dentro da pasta *lazcomponentes* crie uma pasta chamada *fortesreport* e descompacte o pacote dentro desta nova pasta.

Se não estiver executando, execute o Lazarus.

Clique no menu “Arquivo > Fechar tudo” (se perguntado se quer salvar o projeto, não salve).

Clique no menu *Pacotes > Abrir arquivo de pacote (.lpk)*.

No diálogo seguinte, localize na pasta *fortesreport* (que foi criada dentro da pasta *lazcomponentes*) o arquivo

*fortes324forlaz.lpk* . Abra o arquivo. Vai surgir o diálogo de instalação de pacotes.

Clique no botão *Compilar*. Na janela de mensagens do Lazarus, irá aparecer uma série de mensagens de compilação. Quando surgir a mensagem *“Compiling package fortes324forlaz 3.24.2” completado* (ou algo parecido), a compilação do pacote terminou. Agora, você pode clicar no botão *Instalar*. Vai surgir a mensagem para reconstruir o Lazarus. Clique em *Sim*.

Novamente, aparecerá algumas mensagens na janela de mensagens. Se depois que surgir a mensagem *“IDE” completado*, a IDE não reiniciar automaticamente, clique no menu *Arquivo > Reiniciar*. Se for perguntado se deseja salvar o projeto, não salve.

Depois que o Lazarus reiniciar, você terá a disposição a paleta abaixo:



*Figura 7.7 – Paleta do Fortes Report*

*Componentes do Fortes Report:* TRLReport, TRLBand, TRLDetailGrid, TRLGroup, TRLSUBDetail, TRLLabel, TRLLAngleLabel, TRLDDBTtext, TRLMemo, TRLDDBMemo, TRLRichText, TRLDDBRichText, TRLImage, TRLDDBImage, TRLSystemInfo, TRLDdraw, TRLPanel, TRLDDBResult, TRLBarcode, TRLDDBBarcode, TRLPreView, TRLExpressionParser, TRLDraftFilter, TRLRichFilter, TRLHTMLFilter, TRLPDFFilter, TRLXLSFilter, TRLPreviewSetup, TRLGraphicStorage, TRLPrintDialogSetup e TRLSaveDialogSetup.

### Instalando os Componentes do LazReport e WebLaz:

Além do LazReport, vamos instalar um complemento para poder exportar os relatórios para PDF (é necessário ter instalado o pacote PowerPDF). Juntamente, também, instalaremos o pacote WebLaz que possibilita criar aplicativos para Internet usando as tecnologias CGI e FastCGI (não abordado neste livro, mas se quiser aprender veja o site: <http://lazarus-cgi.co.cc/>).

Não vamos precisar baixar o LazReport e nem o WebLaz, pois eles já vêm com Lazarus, mas não são instalados por padrão.

Estando no Lazarus, clique no menu “Arquivo > Fechar tudo” (se perguntado se quer salvar o projeto, não salve).

Clique no menu *Pacotes > Instalar/Desinstalar pacotes ...*



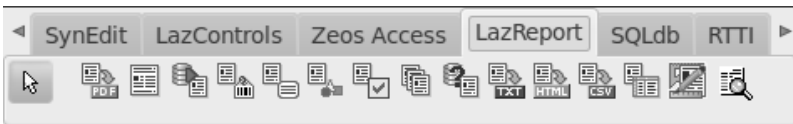
Figura 7.8 – Diálogo de Instalar/Desinstalar pacotes



No dialogo mostrado, para instalar um ou mais componentes, primeiro, selecione-o(s) na lista da direita. Então, selecione *lazreport 0.9.8*, *lazreportpdfexport 0.7* e *weblaz 0.0* . Clique em “*Instalar Seleção*” e clique em “*Salvar e reconstruir IDE*”.

Na tela seguinte clique em *Continuar*. Se depois que surgir a mensagem “*IDE*” *completado*, a IDE não reiniciar automaticamente, clique no menu *Arquivo > Reiniciar*. Se for perguntado se deseja salvar o projeto, não salve.

Depois de ter realizado todo o processo e o Lazarus reiniciar novamente, você terá a disposição a seguinte paleta:



*Figura 7.9 – Paleta do LazReport*

*Componentes do LazReport:* TfrTNPDFExport, TfrReport, TfrDBDataSet, TfrBarCodeObject, TfrRoundRectObject, TfrShapeObject, TfrCheckBoxObject, TfrCompositeReport, TfrUserDataSet, TfrTextExport, TfrHTMExport, TfrCSVExport, TfrPrintGrid, TfrDesigner e TfrPreview.

Se desejar desinstalar os componentes que vimos até aqui ou instalar outros, use o diálogo mostrado na *Figura 7.8*. À esquerda deste, há uma lista com os componentes instalados no Lazarus e a direita, componentes que podem ser instalados.

Para desinstalar um ou mais componentes, selecione-o(s) na lista da esquerda, clique em “*Desinstalar seleção*”. Para instalar um ou mais componentes, selecione-o(s) na lista da direita, clique em “*Instalar Seleção*”.

Para prosseguir, clique em “*Salvar e reconstruir IDE*”. Na tela seguinte clique em *Continuar*. Se depois que surgir a mensagem “*IDE*” *completado*, a IDE não reiniciar automaticamente, clique no menu *Arquivo > Reiniciar*. Se for perguntado se deseja salvar o projeto, não salve.

Para instalar outros componentes através de seus arquivos com extensão *lpk*, basta seguir os passos que consideramos neste capítulo.

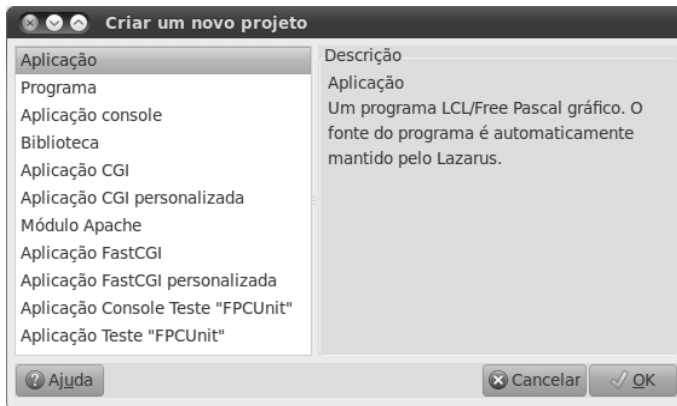
# Programação Visual no Lazarus

## Introdução

Neste capítulo vamos considerar o básico da programação com o Object Pascal do Free Pascal ao passo que aprenderemos a criar interfaces visuais com o Lazarus.

### Tipos de Projetos do Lazarus:

Quando iniciamos o Lazarus, é criado um projeto com um formulário (janela comum) e sua unidade de código associada. Para criarmos um novo projeto, basta clicar no menu “*Projeto > Novo Projeto ...*” e vai abrir o diálogo abaixo:



*Figura 8.1 – Diálogo Criar um novo projeto*

Os principais tipos de projetos são:

**Aplicação:** Projeto com interface gráfica usando a biblioteca de componentes do Lazarus. Este é o tipo de projeto que usaremos no livro.

**Programa:** Projeto de um programa Pascal simples.

**Aplicação console:** Cria um programa com uma nova classe derivada da *TCustomApplication*, que fornece uma série de funcionalidades que tornam a programação em linha de comando muito fácil. Por exemplo, verificar os parâmetros passados na linha de comando, verificar as variáveis de ambiente e gerenciar exceções (erros).

**Biblioteca:** Criar bibliotecas de funções. No Windows, estas têm a extensão “dll”, e no Linux “so”.

**Aplicação CGI:** Cria um programa CGI (Common Gateway Interface) em Free Pascal usando módulos web.

**Aplicação FastCGI:** Cria um programa FastCGI (Common Gateway Interface) em Free Pascal usando módulos web.

### **Criando o Primeiro Programa:**

Para organizar os exemplos que vamos desenvolver no decorrer do livro, crie uma pasta com o nome “*livro\_projetos*”. Se você estiver usando o Linux, crie a pasta dentro de sua *Pasta pessoal*, e, se você estiver usando o Windows, crie a pasta no disco do sistema.

Então, comecemos iniciando o Lazarus. Depois de iniciado, automaticamente temos um projeto base com um formulário e uma unidade de código associada a ele.

Pressione algumas vezes a tecla *F12* e veja que o *Design de Formulário* e o *Editor de Código* se alternam.

A unidade de código associada ao formulário é mostrada a seguir:

```
1.  unit Unit1;
2.
3.  {$mode objfpc}{$H+}
4.
5.  interface
6.
7.  uses
8.      Classes, SysUtils, FileUtil, Forms,
9.      Controls, Graphics, Dialogs;
10.
11. type
12.     TForm1 = class(TForm)
13.     private
14.         { private declarations }
15.     public
16.         { public declarations }
17.     end;
18.
19. var
20.     Form1: TForm1;
21.
22. implementation
23.
24. {$R *.lfm}
25.
26. end.
```

Vamos analisar o código acima:

**Linha 1:** temos ao lado da palavra **unit** o nome da unidade de código do formulário, neste caso *Unit1*. Se você salvasse agora com este nome, seria gerado dois arquivos, um com o nome de *Unit1.pas* e outro com o nome de *Unit1.lfm* . É claro que quando você for salvar os arquivos do seu projeto, deve dar nomes significativos.

**Linha 3:** Aqui temos diretivas para o compilador, indicando que vamos utilizar a linguagem Object Pascal do Free Pascal.

**Linha 5:** A palavra **interface** indica o início da seção de interface onde são definidas as funções, procedimentos, tipos e variáveis que serão usadas pelo formulário associado a *unit* e, também, poderão ser vistos por outras unidades (*unit's*) da aplicação.

**Linhas 7 a 9:** A cláusula **uses**, contém outras unidades de código que fornecem funções, procedimentos e/ou componentes para usarmos na programação do formulário. Quando inserimos outros componentes no formulário, o Lazarus automaticamente adiciona outras *unit's*, se necessário, para podermos manipular estes novos componentes.

**Linhas 11 a 17:** A palavra **type** inicia a definição do formulário. Temos a definição de uma classe *TForm1* que é derivada da classe base *TForm*. Aqui, também, vão ficar as definições dos componentes adicionados no formulário. Os procedimentos, funções, variáveis e constantes do formulário podem ser declarados como: **Private:** Os campos de dados, métodos (procedimentos e funções de um objeto) e objetos declarados nessa área só poderão ser acessados pela própria *unit*. **Public:** Os campos de dados, métodos e objetos declarados nessa área poderão ser acessados tanto pela própria *unit* como por outras *unit's*.

**Linhas 19 e 20:** A cláusula **var** contém a definição de variáveis globais. Veja que é definida uma variável de nome *Form1* que pode ser vista em outras *unit's*, caso a *Unit1* seja incluída na cláusula **uses** destas.

**Linha 22:** A palavra chave **implementation** delimita a segunda seção da *unit*, onde definimos a codificação dos procedimentos e das funções da *unit*.

**Linha 24:** A diretiva **{\$R \*.lfm}** faz a associação da *unit* com seu respectivo formulário e não deve ser modificada.

**Linha 26:** Ao Final da *unit*, temos uma linha com **end.** . Ele é o marcador de final de arquivo. Qualquer coisa colocada após esta linha será ignorada.

Temos, também, o arquivo com extensão *lpr*, que é responsável pela execução do programa. No momento, ele está com o nome *project1.lpr* e para exibi-lo clique no menu *Projeto > Exibir Fonte*.

Segue abaixo a descrição do mesmo:

```
1.    program Project1;
2.
3.    {$mode objfpc}{$H+}
4.
5.    uses
6.        {$IFDEF UNIX}{$IFDEF UseCThreads}
7.        cthreads,
8.        {$ENDIF}{$ENDIF}
9.        Interfaces, // this includes the LCL
        widgetset
10.
11.        Forms, Unit1
12.        { you can add units after this };
13.
14.    {$R *.res}
15.
16.    begin
17.        RequireDerivedFormResource := True;
18.        Application.Initialize;
```

```
19.      Application.CreateForm(TForm1, Form1);
20.      Application.Run;
21.      end.
```

**Linha 1:** Nome do aplicativo que será executado.

**Linhas 5 a 12:** As *unit's* que compõem o aplicativo.

**Linha 14:** Diretiva de compilação **{\$R \*.res}** que refere-se ao arquivo de recursos mantido pelo Lazarus.

**Linhas 16 a 21:** Aqui ocorre a execução do programa: Essa seção inicializa o aplicativo, instância os formulários e em seguida executa o aplicativo.

Vamos, agora, colocar um componente no formulário. Mude para *Unit1*, clicando na guia correspondente no *Editor de Código*, e pressione *F12* até aparecer o formulário.


Na paleta *Standard*, clique no componente *TButton* e clique no meio do formulário. É criado um botão de nome *Button1*.

Observe que o *Button1* está com alças de seleção. Você pode usar estas para redimensioná-lo, e mantendo pressionado o botão esquerdo do mouse em cima do componente é possível arrastá-lo de posição.

Dê um duplo clique sobre o *Button1* e escreva o código em negrito na posição mostrada abaixo:

```
procedure TForm1.Button1Click(Sender: TObject);
begin
    ShowMessage('Meu Primeiro Programa!');
end;
```



Teste o programa clicando no botão executar  da barra de ferramentas do Lazarus. Se estiver no Windows, para compilar mais rápido clique em *CTRL + F9* e depois *F9*.

Clique no *Button1* e aparecerá a mensagem “*Meu Primeiro Programa!*”. Clique em *OK* e feche a janela do seu programa.

Para salvar o projeto, crie na pasta *livro\_projetos* uma pasta com o nome *exemplo8\_1*, volte a IDE do Lazarus e clique no menu *Arquivo > Salvar tudo*. Salve na pasta *exemplo8\_1* o *projeto1.lpi* com o nome *exemplo8\_1.lpi* e a *Unit1.pas* com o nome *uprincipal.pas*.

**Lembrete:** Os nomes de pastas, sub-pastas e nomes de arquivos não podem conter: espaços, traços, acentos e caracteres especiais; podem conter letras, underline “\_” e/ou números.

Execute novamente o programa e feche-o. Abra a pasta *exemplo8\_1* e consideremos os arquivos criados:



*Figura 8.2 – Pasta do exemplo8\_1*

<b>PASTA</b>	<b>DESCRIÇÃO</b>
backup	Arquivos de backup do projeto
lib	Contém os arquivos compilados do projeto
<b>ARQUIVO</b>	<b>DESCRIÇÃO</b>
exemplo8_1	Este é o executável. No Windows tem extensão .EXE (Ex.: exemplo8_1.exe)
exemplo8_1.ico	Ícone da Aplicação
exemplo8_1.lpi	Arquivo do tipo XML com as configurações do projeto
exemplo8_1.lpr	Arquivo com o código de execução do Aplicativo. Similar ao <i>dpr</i> do Delphi
exemplo8_1.res	Arquivo de recursos
uprincipal.lfm	Arquivo com as definições do formulário
uprincipal.pas	Arquivo com a codificação do formulário

### **Diminuindo o Tamanho do Executável:**

Além da diferença no tempo de compilação no Windows (já vimos como resolver isso, e no Linux a compilação é rápida), outra reclamação constante dos usuários do Delphi ao começar a usar o Lazarus é o tamanho do executável gerado.

Em primeiro lugar, os executáveis binários do Linux, geralmente, são maiores do que os do Windows. Isto é uma característica do Linux e não do Lazarus.

O primeiro programa que criamos ficou com aproximadamente 19MB no Linux e aproximadamente 12MB no Windows. De qualquer forma, está estupidamente grande para um programa que só tem uma janelinha e um botão.

Consideremos duas maneiras de diminuir o tamanho do executável.

**OBS.:** Se você fechou o projeto *exemplo8\_1*, pode reabri-lo clicando no menu “*Projeto > Abrir Projeto ...*”, localize o arquivo *exemplo8\_1.lpi* e abra-o.

**Primeira:** Com o seu projeto aberto na IDE Lazarus, clique no menu *Projeto > Opções do Projeto ...*. No diálogo que abrir, na lista da direita, selecione *Vinculado* e na esquerda marque “*Usar arquivo depuração externo gdb (-Xg)*”, marque “*Usar estas opções como padrão para novos projetos*”, clique em OK, execute novamente seu projeto para recriar o executável, abra a pasta do mesmo e veja que tamanho tem agora o executável.

No caso do *exemplo8\_1*, o executável no Linux ficou com 4,6MB e no Windows com 1,54MB.

De fato, melhorou bastante o tamanho do executável. Mas, agora, a cada compilação do projeto, dentro da pasta é criado um arquivo com extensão *gdb* (este sim é grande) usado pela IDE para verificação de erros em tempo de execução. Este arquivo é temporário e pode ser excluído quando quiser, sua aplicação não depende dele para executar.

**Segunda:** O mesmo efeito acima pode ser obtido usando um aplicativo chamado “*strip*” em linha de comando, com a vantagem de não ser gerado um arquivo *gdb* externo.

Para executar o *strip* no projeto *exemplo8\_1*, estando no Linux, basta abrir um terminal e digitar o comando abaixo:

```
strip ~/livro_projetos/exemplo8_1/exemplo8_1
```

Se estiver no Windows, abra um *Prompt de Comando* e digite os dois comandos abaixo:

```
cd \lazarus\fpc\2.4.4\bin\i386-win32
```

```
strip C:\livro_projetos\exemplo8_1\exemplo8_1.exe
```

Depois de aplicar uma das duas sugestões aqui apresentadas, é possível diminuir mais ainda o executável gerado no Lazarus. Para isso, usamos um aplicativo em linha de comando chamado “*upx*”.

Se você está usando o Ubuntu Linux, provavelmente não tem o *upx* instalado. Para instalá-lo digite em um terminal o comando abaixo:

```
sudo apt-get install upx
```

Após instalado, você pode aplicá-lo no projeto *exemplo8\_1* usando o comando abaixo:

```
upx ~/livro_projetos/exemplo8_1/exemplo8_1
```

Para aplicar o *upx*, se estiver no Windows, execute os dois comandos abaixo em um *Prompt de Comando*:

```
cd \lazarus\fpc\2.4.4\bin\i386-win32
```

```
upx C:\livro_projetos\exemplo8_1\exemplo8_1.exe
```

Depois de executar o *upx* no projeto *exemplo8\_1*, ele ficou com o tamanho de 1,6MB no Linux e de 591KB no Windows.

Note, também, que a medida que seu projeto vai crescendo, o executável gerado pelo Lazarus cresce muito pouco de tamanho. Representando mais um ganho.

**CONCLUSÃO:** De, aproximadamente, 19MB, o tamanho do executável caiu para 1,6MB no Linux, e de, aproximadamente, 12MB caiu para 591KB no Windows, e sem prejuízo para o executável.

### Conhecendo o Explorador de Código:

Clique no menu *Exibir > Explorador de Código* . Esta janela contém todos os tipos, classes, propriedades, métodos, variáveis globais, rotinas globais e interfaces contidos na *unit* selecionada.

O *Explorador de Código*, também, permite que você navegue diretamente para as declarações que são apresentadas nele, bastando dar um duplo clique com o mouse sobre a declaração desejada.

### Conhecendo o Inspetor de Projetos:

Clique no menu *Projeto > Inspetor de Projeto* . São mostradas todas as unidades de código que compõem o seu projeto. Possui recurso de inserir ou excluir arquivos do projeto.

Mostra, também, as bibliotecas de componentes usadas pelo projeto, com opção de adicionar outras ou excluí-las do projeto.

### Conhecendo o Inspetor de Objetos, Propriedades e Eventos:

Geralmente à esquerda da IDE, nós temos a janela do *Inspetor de Objetos* (se ela não estiver visível, pressione *F11*).

Nesta janela temos uma guia mostrando as *Propriedades* (características; como, por exemplo, título, tamanho, fonte, cor, etc.) do componente selecionado (que pode ser um botão, uma caixa de texto, o próprio formulário, etc.) e outra guia mostrando os *Eventos* do objeto selecionado (Eventos são respostas as ações do objeto; como, por exemplo, o que fazer quando um botão é clicado).

### Configurando Propriedades e Eventos dos Objetos:

Se não estiver aberto, localize e abra o projeto *exemplo8\_1.lpi* clicando no menu *Projeto > Abrir Projeto ...*.

Clique no fundo do formulário, e no *Inspetor de Objetos*, na guia *Propriedades*, procure pela propriedade *Caption*, clique na caixa de texto ao lado dela e mude o conteúdo de “*Form1*” para “*Meu Primeiro Programa*”. Você acaba de mudar o título da janela do seu formulário. Experimente alterar outras propriedades.

Todo objeto tem um nome para referenciá-lo no código. Mude o nome do formulário de *Form1* para *fPrincipal* através da propriedade *Name*.

Através da propriedade *Color*, mude a cor do formulário para *clNavy* (Azul Marinho).

Agora, clique no botão (*Button1*) para selecioná-lo e mude as seguintes propriedades:

**Caption** → *Clique*

**Name** → *btMensagem*

Ainda no *Inspetor de Objetos*, clique na guia *Eventos*.

Observe que já havíamos criado um procedimento para responder ao evento *OnClick*. Este é o evento padrão do objeto *TButton*, que é criado quando damos um duplo clique sobre ele em modo de design.

O evento *OnClick* corresponde ao que deve acontecer quando o usuário clicar no botão durante a execução do aplicativo.

Para criar ou se posicionar nos eventos, basta, no *Inspetor de Objetos*, dar um duplo clique na caixa de texto ao lado do evento ou um clique para selecionar e outro no botão de reticências ( ... ), ao lado. Então, clique no botão de reticências ( ... ) ao lado do evento *OnClick*.

Veja que o cursor foi posicionado no código exatamente onde está o procedimento *OnClick* do botão *btMensagem*.

Todo código que estiver entre **begin** e **end** é o que será executado quando o botão for clicado. Neste caso, mostra uma mensagem.

Mude o que estiver entre o **begin** e o **end**, para:

```
ShowMessage('Seja Bem Vindo ao Lazarus!' + #13 +  
#13 + InputBox('Nome', 'Digite seu nome:', ''));
```

Execute e teste o programa. Não se preocupe em entender o código agora.

### Criando o Segundo Programa:

Crie uma pasta na pasta *livro\_projetos* com o nome *exemplo8\_2*. No Lazarus, crie um novo projeto do tipo Aplicação. Salve na pasta *exemplo8\_2* o projeto com o nome *exemplo8\_2.lpi* e a *unit* com o nome *uprincipal.pas*.

Mude as seguintes propriedades do formulário:

**Caption** → *Exemplo 8.2*

**Name** → *fPrincipal*

Coloque no formulário um *TLabel* (mostra um rótulo de texto), um *TEdit* (provê um espaço para o usuário digitar um texto) e um *TButton* da paleta *Standard*. Selecione o *TLabel*, e mude as propriedades:

**Caption** → *Digite seu Nome:*

**Name** → *lbNome*

Selecione o *TEdit*, e mude as propriedades:

**Name** → *edNome*

**Width** → *220*

**Text** → (apague o conteúdo)

Selecione o *TButton*, e mude as propriedades:

**Caption** → *Mostrar*

**Height** → *27*

**Name** → *btMostrar*

Posicione os objetos para que fiquem como mostrado na figura:



*Figura 8.3 – Formulário do exemplo8\_2*



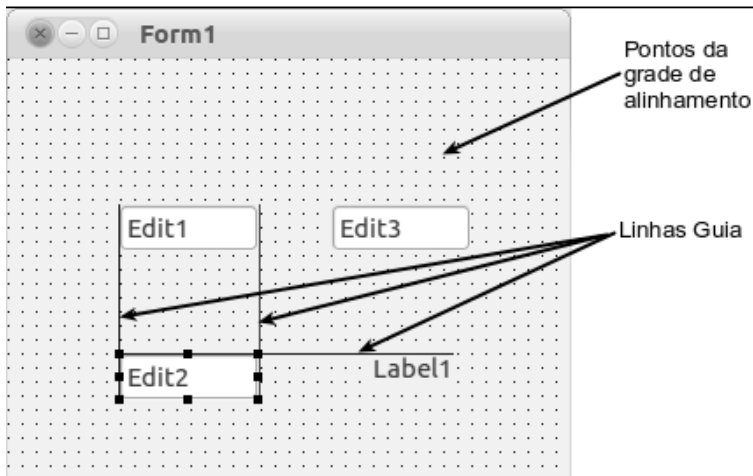
Selecione o botão *btMostrar* (*TButton*) e crie o evento *OnClick* dele. Entre o **begin** e o **end**, do procedimento gerado, digite o código abaixo:

```
ShowMessage('Seu nome é: ' + edNome.Text);
```

Execute e teste o programa. Mude outras propriedades.

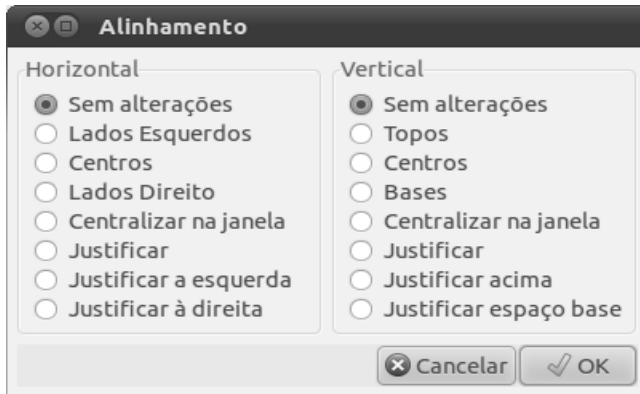
### Alinhando Componentes:

O editor de formulários do Lazarus permite posicionar componentes através de uma grade de alinhamento. Também, possui linhas guias de posicionamento. Observe a figura abaixo:



*Figura 8.4 – Grade e Guias de Posicionamento*

Podemos, também, alinhar componentes usando o diálogo Alinhamento:



*Figura 8.5 – Diálogo de Alinhamento*

Usando o mouse, selecione um grupo de componentes, clique com o botão direito do mouse sobre este e no menu pop-up escolha Alinhamento. Experimente as diversas opções de alinhamento (Veja Figura 8.5).

---

A seguir, consideraremos o básico da programação em Object Pascal:

### **Bloco de Código:**

Um bloco de código é delimitado por **begin** e **end**.

Ex.:

```
begin  
    {comandos}  
end;
```

### Linha de Comando:

O fim de uma linha de comando é indicado por ponto e vírgula. Com exceção de comandos delimitadores de sessão ou bloco de código, como é o caso do **begin**.

Ex.:

```
begin
  ShowMessage('IDE Lazarus');
  ShowMessage
    ('Ubuntu 10.04');
end;
```

### Comentários:

Use comentários para documentar seu código. O Free Pascal aceita os seguintes tipos de delimitadores de comentários:

Ex.:

```
// Comentário de uma linha

{ Comentário
  com várias
  linhas }

(* Outro tipo de
  comentário
  com várias
  linhas *)
```

### Variáveis:

Uma variável é um espaço nomeado da memória para guardar um tipo de valor. Você pode declarar (criar na memória) variáveis em Object Pascal da seguinte forma:

[nome da variável] : [tipo da variável] ;

As variáveis são declaradas na seção **var** de um programa ( *.lpr* ), *unit*, procedimento (*procedure*) ou função (*function*) (Veja mais a frente o que são *procedures* e *function's*).

Em uma *unit*, nós temos uma seção **var** antes da seção **implementation** e podemos iniciar outra logo após. As variáveis declaradas nestas seções **var** ficam acessíveis em todas as *procedures* e *functions* da *unit*. São variáveis globais.

As variáveis locais são declaradas na seção **var** de uma *procedure* ou *function*. Ficam acessíveis apenas em sua respectiva *procedure* ou *function*. Passam a existir na memória quando a *procedure* ou *function* é chamada e deixam de existir na memória quando a *procedure* ou *function* termina de executar.

O nome de uma variável pode ter até 255 caracteres, tem que começar com uma letra, não pode conter caracteres acentuados, espaço em branco e ser única em âmbito global e local. O nome pode conter números e underline “\_” e não pode ser uma palavra reservada (begin, end, etc.).

Ex.:

```
procedure Teste;  
var  
    Sistema: string; // Variável local do tipo texto  
begin  
    Sistema := 'Ubuntu Linux'; // Atribuindo um valor  
    ShowMessage(Sistema); // Mostrando o valor  
end;
```

**Tipos de Variáveis:**

Tipos de números inteiros:

<b>Tipo</b>	<b>Intervalo</b>
Byte	0 .. 255
Shortint	-128 .. 127
Smallint	-32768 .. 32767
Word	0 .. 65535
Integer	Mapeia para Longint em Object Free Pascal
Cardinal	Mapeia para Longword
Longint	-2147483648 .. 2147483647
Longword	0 .. 4294967295
Int64	-9223372036854775808 .. 9223372036854775807
QWord	0 .. 18446744073709551615

Tipos de números reais:

<b>Tipo</b>	<b>Intervalo</b>
Real	Depende da Plataforma Operacional
Single	$1.5^{-45}$ .. $3.4^{38}$
Double	$5.0^{-324}$ .. $1.7^{308}$
Extended	$1.9^{-4932}$ .. $1.1^{4932}$
Comp	$-2^{64}+1$ .. $2^{63}-1$
Currency	-922337203685477.5808 .. 922337203685477.5807

Existe o tipo **Booleano** que aceita *True* (Verdadeiro) e *False* (Falso). Tipos para data e hora: **TDate**, **TTime** e **TDateTime**.

Há o tipo **Char** que aceita um caractere. O valor de uma variável do tipo *Char* deve ser declarado entre aspas simples.

O tipo **String** serve para armazenar um texto com tamanho limitado apenas pela memória do computador. O valor de uma variável do tipo *string* deve ser declarado entre aspas simples.

Estes são os tipos mais usados.

### Constantes:

As constantes são declaradas com um valor que não muda durante a execução do programa. A declaração tem a sintaxe abaixo:

[nome da constante] = [valor] ;

As constantes são declaradas na seção **const** de um programa, *unit*, procedimento (*procedure*) ou função (*function*).

Uma seção **const** deve ser aberta antes, depois ou na mesma posição de uma seção **var**. Podem ser globais ou locais.

Ex.:

#### **const**

```
nome = 'Adriana'; // Constante do tipo string
idade = 35; // Constante de número inteiro
compras = 150.25; // Constante de número real
```

### Atribuição:

Usa-se `:=` (dois pontos e igual) para atribuir um valor a uma variável ou propriedade de um objeto.

Ex.:

```
procedure Teste;  
var  
    nome: string;  
    idade: integer;  
    compras: double;  
    letra: char;  
begin  
    nome := 'Jean Patrick';  
    idade := 35;  
    compras := 150.25;  
    edNome.Text := nome;  
    letra := 'A';  
end;
```

### Arrays ou Matrizes:

Um array é uma coleção de variáveis do mesmo tipo (integer, double, string, etc), identificadas por um índice. Um array é declarado do mesmo modo que as variáveis:

```
var  
    Aluno: array[0..15] of string;
```

O código anterior, declara um array de nome Aluno composto de 16 (0 a 15) variáveis do tipo string. Podemos ler ou atribuir um valor para cada variável através do nome do array seguido do índice numérico entre colchetes. Veja os exemplos:

```
Aluno[0] := 'Jean'; { Atribui o texto Jean a  
variável de índice 0. }
```

```
Aluno[9] := 'Adriana'; { Atribui o texto Adriana a
variável de índice 9. }
Aluno[15] := Aluno[0]; { A variável de índice 15
recebe o valor da variável de índice 0. }
```

### Record ou Registro:

A estrutura *record* define um novo tipo de variável composto por variáveis filhas que podem ser de tipos diferentes. Um tipo *record* é declarado dentro de uma seção *type*. Por exemplo:

```
procedure Escola;
// Cria um novo tipo record de nome Alunos
type
    Alunos = record
        Nome: string;
        Nota: double;
        Ativo: boolean;
    end;
// Cria uma variável de nome Aluno e do tipo Alunos
var
    Aluno: Alunos;
begin
    // Atribui valores as variáveis filhas de Aluno
    Aluno.Nome := 'Adriana';
    Aluno.Nota := 10.00;
    Aluno.Ativo := True;
    // Mostra o valor da variável filha Nome da
    // varável Aluno
    ShowMessage('Nome da Aluna: ' + Aluno.Nome);
end;
```

Seguindo o exemplo acima, em vez de criar uma variável do tipo *Alunos*, poderíamos criar um *array* do tipo *Alunos* assim:

```
var
    Aluno: array[0..15] of Alunos;
```



E usar desta maneira:

```
Aluno[0].Nome := 'Bethoven';
Aluno[0].Nota := 8.50;
Aluno[0].Ativo := True;
Aluno[1].Nome := 'Olívia';
Aluno[1].Nota := 9.50;
Aluno[1].Ativo := False;
```

E assim sucessivamente até chegar no maior índice definido.

### Operadores Aritméticos:

<b>+</b>	Adição (em strings concatena)
<b>-</b>	Subtração
<b>*</b>	Multiplicação
<b>/</b>	Divisão entre números reais
<b>DIV</b>	Divisão entre números inteiros
<b>MOD</b>	Resto da divisão

### Operadores Lógicos:

<b>AND</b>	E	lógico
<b>OR</b>	OU	lógico
<b>XOR</b>	OU EXCLUSIVO	lógico

### Operadores Relacionais:

Estes operadores são utilizados nas tomadas de decisões, os principais são os seguintes:

<b>=</b>	igual
<b>&lt;&gt;</b>	diferente
<b>&gt;</b>	maior que
<b>&lt;</b>	menor que
<b>&gt;=</b>	maior ou igual que

**<=**    menor ou igual que

### Procedures e Functions:

Trata-se de blocos de código que são executados apenas quando requisitados através do nome da respectiva *procedure* (procedimento) ou *function* (função).

As *procedures* não retornam valores como resposta, mas as *function's* sim. Ambas podem, ou não, possuir parâmetros (variáveis) que serão usados no código executado.

Sintaxe:

```
procedure [nome_da_procedure]([parâmetros_opcionais]
);
begin
    {comandos}
end;
```

```
function [nome_da_função]([parâmetros_opcionais]):[
tipo_de_retorno];
begin
    {comandos}
    Result := [valor_de_retorno] ;
end;
```

Ex.:

```
procedure FazNada;
var
    x, y, z: integer;
begin
    x := 10;
    y := 20;
    z := x + y;
end;
```

```
procedure FazNada2(z: integer);  
var  
    x, y: integer;  
begin  
    x := 10;  
    y := z + x;  
end;  
  
function Soma10(y: integer):integer;  
var  
    x: integer;  
begin  
    x := 10; // variável desnecessária, só exemplo  
    Result := x + y; // ou poderia ser 10 + y  
end;
```

### Configurações Regionais:

Especialmente se o seu aplicativo for executar no Linux, é necessário configurar algumas variáveis para exibição correta de datas, valores no formato de moeda, nomes das semanas e nome dos meses no padrão brasileiro.

O ideal é que estas variáveis sejam redefinidas no evento *OnCreate* do formulário principal da sua aplicação. Observe o código em negrito:

```
procedure TForm1.FormCreate(Sender: TObject);  
begin  
    // -----  
    // Configura a exibição de moeda no formato  
    // brasileiro. Ex.: R$ 1.500,45  
    // -----  
    CurrencyString := 'R$';  
    CurrencyFormat := 2;  
    DecimalSeparator := ',';  
    ThousandSeparator := '.';  
    // -----
```

```
// Configura a exibição de data no formato
// brasileiro.
// Ex.:
// 28/03/2011
// Segunda, 28 de Março de 2011
// -----
DateSeparator := '/';
ShortDateFormat := 'dd/mm/yyy';
LongDateFormat := 'dddd, dd "de" mmmm "de" yyy';
// -----
// Configura os nomes longos e curtos dos meses e
// das semanas.
// -----
LongMonthNames[1] := 'Janeiro';
LongMonthNames[2] := 'Fevereiro';
LongMonthNames[3] := 'Março';
LongMonthNames[4] := 'Abril';
LongMonthNames[5] := 'Maio';
LongMonthNames[6] := 'Junho';
LongMonthNames[7] := 'Julho';
LongMonthNames[8] := 'Agosto';
LongMonthNames[9] := 'Setembro';
LongMonthNames[10] := 'Outubro';
LongMonthNames[11] := 'Novembro';
LongMonthNames[12] := 'Dezembro';
ShortMonthNames[1] := 'Jan';
ShortMonthNames[2] := 'Fev';
ShortMonthNames[3] := 'Mar';
ShortMonthNames[4] := 'Abr';
ShortMonthNames[5] := 'Mai';
ShortMonthNames[6] := 'Jun';
ShortMonthNames[7] := 'Jul';
ShortMonthNames[8] := 'Ago';
ShortMonthNames[9] := 'Set';
ShortMonthNames[10] := 'Out';
ShortMonthNames[11] := 'Nov';
ShortMonthNames[12] := 'Dez';
LongDayNames[1] := 'Domingo';
LongDayNames[2] := 'Segunda';
```

```
LongDayNames[3] := 'Terça';
LongDayNames[4] := 'Quarta';
LongDayNames[5] := 'Quinta';
LongDayNames[6] := 'Sexta';
LongDayNames[7] := 'Sábado';
ShortDayNames[1] := 'Dom';
ShortDayNames[2] := 'Seg';
ShortDayNames[3] := 'Ter';
ShortDayNames[4] := 'Qua';
ShortDayNames[5] := 'Qui';
ShortDayNames[6] := 'Sex';
ShortDayNames[7] := 'Sáb';
end;
```

### Função StrToInt:

Converte uma string (texto) em número inteiro.

Ex.:

```
var
  x: integer;
  y: string;
begin
  y := '25';
  x := StrToInt(y);
end;
```

Se a string não poder ser convertida, um erro ocorrerá.

### Função IntToStr:

Faz o inverso da função *StrToInt*.

### Função StrToFloat:

Converte uma string (texto) em um número real.

Ex.:

```
var
  x: double;
  y: string;
begin
  y := '25.45';
  x := StrToFloat(y);
end;
```

Se o separador decimal não for o ponto ou for redefinido para outro caractere, a string a ser convertida deve seguir a configuração.

Ex.:

```
var
  x: double;
  y: string;
begin
  // Definindo separador decimal para virgula
  DecimalSeparator := ',';
  // Definindo separador de milhar para ponto
  ThousandSeparator := '.';
  y := '25.45'; // usando ponto
  x := StrToFloat(y); // Vai gerar um erro
  y := '25,45'; // usando virgula
  x := StrToFloat(y); // Não dá erro
  x := x + 15.45; // Número continua usando ponto
end;
```

**OBS.:** Mesmo trocando o separador decimal, a representação de um número real sempre usará o ponto.

### Função FloatToStr:

Faz o inverso da função *StrToFloat*.

### **Função Date:**

Retorna a data atual.

### **Função Time ou Now:**

As duas funções retornam a hora ou a data e hora atual.

### **Função StrToDate:**

Converte uma string (texto) de data em um tipo TDateTime.

### **Função DateToStr:**

Faz o inverso da função *StrToDate*.

### **Função StrToTime:**

Converte uma string (texto) de hora em um tipo TDateTime.

### **Função TimeToStr:**

Faz o inverso da função *StrToTime*.

### **Recursos do Editor de Código:**

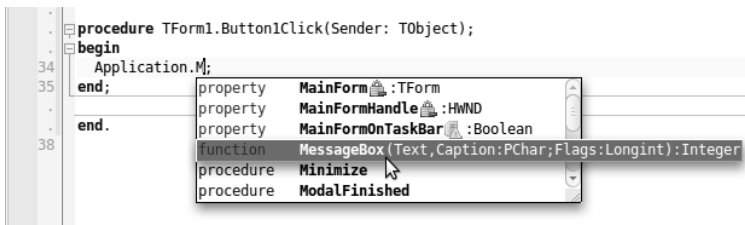
O editor de códigos do Lazarus possui muito mais recursos do que o editor de códigos do Delphi 7. Estes recursos são similares aos encontrados em versões mais recentes do Delphi. A mais comum destas funcionalidades é o complemento de código, que está presente no Delphi 7, mas no Lazarus é bem mais refinado. Por exemplo, digite no editor do Lazarus o seguinte código e pressione CTRL + ESPAÇO:

```
Application.M
```

Surgirá, como mostrado abaixo, um menu pop-up com sugestões para completar o código. No caso, mostra propriedades, funções, métodos e etc, pertencentes ao objeto *Application* que comecem com a letra “M”. Ao passo que prosseguimos com a digitação da palavra, as opções são filtradas automaticamente.

Além de outros detalhes, especialmente, observamos que é exibido, também, os parâmetros recebidos por funções e procedimentos, e no caso das funções podemos ver o tipo de valor de retorno.

Outro aspecto é que este menu pop-up surge automaticamente quando digitamos o nome de um componente (objeto) seguido de ponto ( . ) para chamar propriedades, procedimentos ou funções pertencentes ao componente.

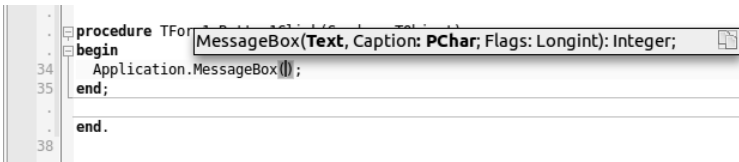


*Figura 8.6 – Menu Pop-up para Completar Código*

Podemos saber facilmente que parâmetros são aceitos por determinada função ou procedimento de um componente, bastando posicionar o cursor entre os parêntesis da função ou procedimento e pressionar CTRL + SHIFT + ESPAÇO para surgir uma caixa de dica os parâmetros usados.

Ao passo que formos digitando os parâmetros, o parâmetro em edição é destacado na caixa de dica.

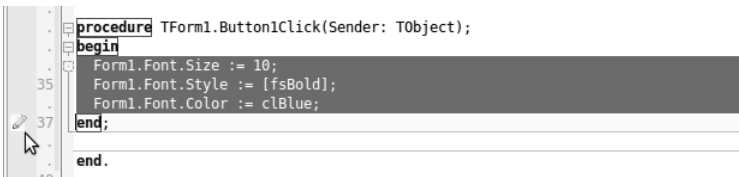




*Figura 8.7 – Dica de Parâmetros*

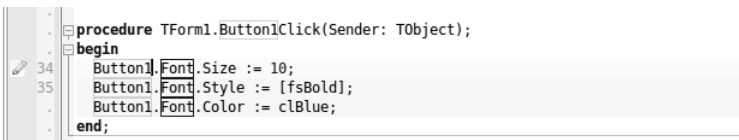
Observe que na caixa de dica há um botão copiar que permite inserir o modelo de parâmetros entre os parêntesis da função ou procedimento.

Outro recurso do editor do Lazarus permite a edição de palavras que se repetem pelo código. Veja no exemplo abaixo que foram selecionadas três linhas onde a palavra Form1 se repete.



*Figura 8.8 – Editar Palavras Iguais*

No fim do bloco selecionado, nota-se a esquerda um ícone de lápis. Ao clicar sobre este, as palavras que se repetem ficam contornadas por um retângulo. Ao editar uma destas palavras todas as outras iguais são editadas. Observe no exemplo abaixo que foi trocado Form1 para Button1.



*Figura 8.9 – Depois de Editado*

# Programação Visual no Lazarus

## Estruturas de Controle

Consideraremos agora as principais estruturas de controle do Object Pascal do Free Pascal. Se você está acostumado com o Delphi, a sintaxe dos comandos é a mesma nos dois. Para fixar os conceitos desenvolveremos alguns exemplos. Antes de prosseguir, talvez queira recapitular sobre propriedades e eventos no capítulo anterior.

### Estrutura TRY .. EXCEPT .. END:

Monitora o código entre TRY e EXCEPT. Caso ocorra um erro neste código, a execução do programa é desviada para a execução do código entre EXCEPT e END. Se não ocorrer um erro o código entre EXCEPT e END não é executado.

Sintaxe:

```
try
    {código monitorado}
except
    {código para tratar um erro}
end;
```

Veja um exemplo de uso mais a frente neste capítulo.

### Estrutura IF ... THEN ... ELSE:

Estrutura de decisão. Executa um comando ou uma cadeia de comandos se uma determinada condição for verdadeira. Se for falsa, opcionalmente, pode ser executado um outro comando ou cadeia de comandos.

Em pseudo linguagem – Portugal:

*SE {condição verdadeira} ENTÃO {faça} SENÃO {faça}*

A sintaxe desta estrutura em Object Pascal pode ser:

```
if (condição) then {uma única linha de comando};
```

ou

```
if (condição) then begin  
    {várias linhas de comando}  
end;
```

ou

```
if (condição) then  
    {uma única linha de comando}  
else  
    {uma única linha de comando};
```

ou

```
if (condição) then begin  
    {várias linhas de comando}  
end else begin  
    {várias linhas de comando}  
end;
```

### Programa Exemplo – IF ... THEN ... ELSE:

Crie uma pasta na pasta *livro\_projetos* com o nome *exemplo9\_1*. No Lazarus, crie um novo projeto do tipo Aplicação. Salve na pasta *exemplo9\_1* o projeto com o nome *exemplo9\_1.lpi* e a *unit* com o nome *uprincipal.pas*.

Mude o nome do formulário para *fPrincipal* e o *Caption* para *Exemplo 9.1* .

Dê um duplo clique no fundo do formulário. Automaticamente será criado o evento *OnCreate* para o mesmo. Digite o código em **negrito** na posição indicada:

```
procedure TfPrincipal.FormCreate(Sender: TObject);  
begin  
    DecimalSeparator := ',';  
    ThousandSeparator := '.';  
end;
```

Volte ao formulário (pressione F12) e coloque dois *TLabel*, dois *TEdit* e dois *TButton* da paleta *Standard*.

Mude o nome do primeiro *TEdit* para *edNum1* e do segundo para *edNum2* . Apague o conteúdo da propriedade *Text* dos dois *TEdit's* .

Altere o *Caption* do primeiro *TLabel* para “Número 1” e do segundo para “Número 2” (sem as aspas).

Modifique o nome dos botões para *btMaior* e *btMenor*, e os *Caption's* para *Maior* e *Menor*, respectivamente.

Organize os componentes no formulário para ficarem como mostra a próxima figura:



*Figura 9.1 – Programa Exemplo 9.1*

Crie o evento *OnClick* para o botão *btMaior* e digite o código em negrito nas posições indicadas (a escrita dos comentários é opcional):

```
procedure TfPrincipal.btMaiorClick(Sender:
TObject);
var
    // variáveis do mesmo tipo podem ser declaradas
    // na mesma linha, separadas por virgula
    num1, num2: double;
begin
    try
        // Início do código monitorado
        num1 := StrToFloat(edNum1.Text);
        num2 := StrToFloat(edNum2.Text);
        if (num1 = num2) then begin
            // Os comandos abaixo até o end; , só serão
            // executados se os dois números forem iguais
            ShowMessage('Os dois números são iguais');
            exit; // interrompe a execução da procedure
        end;
        // ----- Início do if -----
        if (num1 > num2) then
            // A mensagem a baixo só é mostrada se
            // o primeiro número for maior que o segundo
            ShowMessage('O primeiro número é o maior!')
        else
            // A mensagem a baixo só é mostrada se
            // o segundo número for maior que o primeiro
            ShowMessage('O segundo número é o maior!');
        // ----- Fim do if -----
        // fim do código monitorado
    except
        // Início do código para tratar erros
        ShowMessage('Digite números válidos!');
        // Fim do código para tratar erros
    end;
end;
```

**NOTA:** Sintaxe para pegar o valor da propriedade de um componente:

**[variável] := [nome\_componente].[nome\_propriedade] ;**

E para atribuir um novo valor a propriedade de um componente:

**[nome\_componente].[nome\_propriedade] := [novo\_valor] ;**

Crie o evento *OnClick* para o botão *btMenor* e digite o código em negrito nas posições indicadas:

```
procedure TfPrincipal.btMenorClick(Sender:
TObject);
var
    num1, num2: double;
begin
    try
        num1 := StrToFloat(edNum1.Text);
        num2 := StrToFloat(edNum2.Text);
        if (num1 = num2) then begin
            ShowMessage('Os dois números são iguais');
            exit;
        end;
        if (num1 < num2) then
            ShowMessage('O primeiro número é o menor!')
        else
            ShowMessage('O segundo número é o menor!');
        except
            ShowMessage('Digite números válidos!');
        end;
    end;
```

Execute e teste o programa.

**ATENÇÃO:** Durante a execução do programa, se você digitar em um dos *TEdit's* um texto, isso vai resultar num erro. A mensagem 'Digite números válidos!' vai aparecer, porém se você estiver executando o programa através do Lazarus,

aparecerá primeiro a mensagem do depurador, que é responsável por fazer o monitoramento dos erros que ocorrem durante o teste do programa. Para continuar a execução do programa, clique no botão *Continuar*.

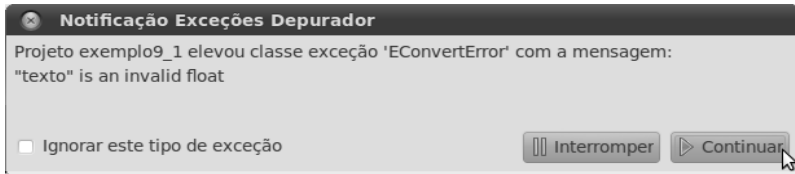


Figura 9.2 – Mensagem do Depurador de Erros

### Estrutura WHILE .. DO:

Repete uma linha ou várias linhas de código, enquanto uma determinada condição for verdadeira. O comando **BREAK** causa uma saída imediata do laço de repetição, passando a executar os comandos seguintes, fora do WHILE, se houverem. Já o comando **CONTINUE** retorna o fluxo do laço para o início do WHILE.

Em pseudo linguagem (Portugol):

ENQUANTO {condição verdadeira} FAÇA {isso}

Sintaxe:

```
while (condição) do {único comando};  
  
while (condição) do begin  
    {vários comandos}  
end;
```

### Programa Exemplo – WHILE ... DO:

Crie uma pasta na pasta *livro\_projetos* com o nome *exemplo9\_2*. No Lazarus, crie um novo projeto do tipo Aplicação. Salve na pasta *exemplo9\_2* o projeto com o nome *exemplo9\_2.lpi* e a unit com o nome *uprincipal.pas*.

Mude o nome do formulário para *fPrincipal* e o *Caption* para *Exemplo 9.2*.

No formulário, coloque dois *TLabel*, um *TEdit*, um *TMemo* (este componente pode mostrar um texto com várias linhas) e um *TButton* da paleta *Standard*. Altere o nome do *TEdit* para *edLimite* e apague o conteúdo da propriedade *Text*. Mude o *Caption* do primeiro *TLabel* para “Números Pares de 2 até:” e do segundo para “Limite de 1000”. Troque, também, a propriedade *Color* deste para *clRed* (vermelho). Modifique o nome do botão para *btMostrar*, e o *Caption* para *Mostrar*. Mude o nome do *TMemo* para *meListagem*, a propriedade *ReadOnly* para *True* e *ScrollBars* para *ssVertical*.

Organize os componentes no formulário para ficarem como mostra a próxima figura:



Figura 9.3 – Exemplo 9.2



No evento *OnClick* do botão, digite o código em negrito nas posições indicadas:

```
procedure TfPrincipal.btMostrarClick(Sender:
TObject);
var
    limite, contador: integer;
begin
    contador := 2; // Inicializa a variável contador
    // Limpa as linhas do TMemor
    meListagem.Lines.Clear;
    try
        limite := StrToInt(edLimite.Text);
        if (limite > 1000) then begin
            ShowMessage('Ultrapassou o Limite de 1000!');
            Exit; // Sai da procedure
        end;
        while (contador <= limite) do begin
            meListagem.Lines.Add(IntToStr(contador));
            // Incrementa de 2 em 2 a variável contador
            Inc(contador, 2);
        end;
    except
        ShowMessage('Digite um Limite Válido!');
    end;
end;
```

### Estrutura REPEAT .. UNTIL:

Similar ao WHILE .. DO, mas testa a condição no fim do laço de repetição. Pelo menos uma vez os comandos do laço serão executados.

Aceita BREAK e/ou CONTINUE (Veja Estrutura WHILE .. DO).

Em pseudo linguagem (Portugol):

FAÇA {isso} ENQUANTO {condição verdadeira}

Sintaxe:

```
repeat
    {vários comandos}
until (condição);
```

### Programa Exemplo – REPEAT .. UNTIL:

Crie uma pasta na pasta *livro\_projetos* com o nome *exemplo9\_3*. No Lazarus, crie um novo projeto do tipo Aplicação. Salve na pasta *exemplo9\_3* o projeto com o nome *exemplo9\_3.lpi* e a unit com o nome *uprincipal.pas*. Mude o nome do formulário para *fPrincipal* e o *Caption* para *Exemplo 9.3*.

No formulário, coloque dois *TLabel*, um *TEdit*, um *TMemo* e um *TButton* da paleta *Standard*. Altere o nome do *TEdit* para *edLimite* e apague o conteúdo da propriedade *Text*. Mude o *Caption* do primeiro *TLabel* para “Números Ímpares de 1 até:” e do segundo para “Limite de 1000”. Troque, também, a propriedade *Color* deste para *clRed*. Modifique o nome do botão para *btMostrar*, e o *Caption* para *Mostrar*. Mude o nome do *TMemo* para *meListagem*, a propriedade *ReadOnly* para *True* e *ScrollBars* para *ssVertical*. Organize os componentes no formulário para ficarem como mostra a próxima figura:



Figura 9.4 – Exemplo 9.3

No evento *OnClick* do botão, digite o código em negrito nas posições indicadas:

```
procedure TfPrincipal.btMostrarClick(Sender:
TObject);
var
    limite, contador: integer;
begin
    contador := 1;
    mListagem.Lines.Clear;
    try
        limite := StrToInt(edLimite.Text);
        if (limite > 1000) then begin
            ShowMessage('Ultrapassou o Limite de 1000!');
            exit;
        end;
        repeat
            mListagem.Lines.Add(IntToStr(contador));
            // Incrementa de 2 em 2 a variável contador
            Inc(contador,2);
        until (contador > limite);
    except
        ShowMessage('Digite um Limite Válido!');
    end;
end;
```

### Estrutura FOR .. TO (DOWNTO) .. DO:

Laço com limite de repetição previamente definido por variável. Usando DOWNTO a contagem é em ordem decrescente.

Pode-se usar BREAK ou CONTINUE (Veja Estrutura WHILE .. DO).

Em pseudo linguagem (Portugol):

PARA {contagem\_inicial} ATE {contagem\_final} FAÇA {isso}

Sintaxe:

```
for [variável]:= [valor_inicial] to [valor_final] do  
  {comando}  
  
for [variável]:= [valor_inicial] to [valor_final] do  
begin  
  {comandos}  
end;
```

### Programa Exemplo – FOR .. TO (DOWNTO) .. DO:

Crie uma pasta na pasta *livro\_projetos* com o nome *exemplo9\_4*.

No Lazarus, crie um novo projeto do tipo Aplicação. Salve na pasta *exemplo9\_4* o projeto com o nome *exemplo9\_4.lpi* e a unit com o nome *uprincipal.pas*.

Mude o nome do formulário para *fPrincipal* e o *Caption* para *Exemplo 9.4*.

No formulário, coloque dois *TLabel*, um *TEdit*, um *TMemo* e um *TButton* da paleta *Standard*. Altere o nome do *TEdit* para *edLimite* e apague o conteúdo da propriedade *Text*.

Mude o *Caption* do primeiro *TLabel* para “*Tabuada de Multiplicar de:*” e do segundo para “*Limite de 10*”. Troque, também, a propriedade *Color* deste para *clRed*.

Altere o nome do botão para *btMostrar*, e o *Caption* para *Mostrar*.

Mude o nome do *TMemo* para *meListagem*, a propriedade *ReadOnly* para *True* e *ScrollBars* para *ssVertical*. Organize os

componentes no formulário para ficarem como mostra a próxima figura:



*Figura 9.5 – Exemplo 9.4*

No evento *OnClick* do botão digite o código em negrito nas posições indicadas:

```
procedure TfPrincipal.btMostrarClick(Sender:
TObject);
var
    limite, contador: integer;
    l1, c1, s1: string;
begin
    mListagem.Lines.Clear;
    try
        limite := StrToInt(edLimite.Text);
        if (limite > 10) then begin
            ShowMessage('Ultrapassou o Limite de 10!');
            Exit;
        end;
        Str(limite : 2 , l1);
        // Str - Guarda o valor de uma variável
        // numérica em outra do tipo string (texto);
        // Sintaxe:
        // Str([var_num]:[tamanho],[var_texto]);
        for contador := 1 to 10 do begin
            Str(contador : 2 , c1);
```

```
    Str((limite * contador) : 2 , ns1);  
    mListagem.Lines.Add(l1 + ' X ' +  
        c1 + ' = ' + s1);  
end;  
except  
    ShowMessage('Digite um Limite Válido!');  
end;  
end;
```

Execute e teste o programa.

### Estrutura CASE .. OF:

Permite a execução seletiva de comandos. Similar a uma sequência de vários IF's.

Sintaxe:

```
case [variável_seletora] of  
    [caso 1]: {comando}  
    [caso 2]:  
        begin  
            {vários_comandos}  
        end;  
    [caso n]: {comando}  
[else]  
    {comandos}  
end;
```

Ex.:

```
case operacao of  
    '+': ShowMessage('Soma');  
    '-': ShowMessage('Subtração');  
    'x': ShowMessage('Multiplicação');  
    '÷': ShowMessage('Divisão');  
else  
    ShowMessage('Operação Inválida');
```

`end;`

**OBS.:** A variável seletora do comando `CASE` só pode ser do tipo *integer*, *char* (um só caractere) ou *boolean*.

No próximo capítulo será considerado o uso na prática do comando `CASE`.

Bem, consideramos as estruturas básicas de programação em Object Pascal e vimos alguns exemplos de uso através de pequenos programas visuais. No capítulo seguinte, vamos trabalhar exemplos mais complexos e extensos ao passo que consideraremos outros recursos do Free Pascal e do Lazarus.

# Programação Visual no Lazarus

## Projetos de Exemplo

Os exemplos que seguem ajudarão o leitor a ter uma melhor percepção da capacidade do Lazarus. Lembrando que todos os exemplos criados até aqui e os que seguem podem ser compilados e executados no Linux e no Windows.

### Criando uma Calculadora Completa:

A calculadora que vamos desenvolver é muito parecida a do Ubuntu 10.04, mas o modo de funcionamento dela é similar a calculadora do Windows. Depois de pronta, ela terá a aparência mostrada abaixo:



*Figura 10.1 – Calculadora*



Para começarmos, primeiro, crie uma pasta na pasta *livro\_projetos* com o nome *exemplo10\_1*. No Lazarus, crie um novo projeto do tipo Aplicação. Salve na pasta *exemplo10\_1* o projeto com o nome *exemplo10\_1.lpi* e a unit com o nome *ucalculadora.pas*.

Mude as seguintes propriedades do formulário:

**Name** → *fCalculadora*  
**Caption** → *Calculadora*  
**Position** → *poDesktopCenter*  
**Height** → *300*  
**Width** → *232*

Clique no sinal de mais ( + ) ao lado da propriedade **Constraints** e altere as seguintes propriedades filhas:

**MaxHeight** → *300*  
**MaxWidth** → *232*  
**MinHeight** → *300*  
**MinWidth** → *232*

Ainda no *Inspetor de Objetos* (pressione F11), acima das guias *Propriedades* e *Eventos*, temos um painel com todos os componentes (inclusive o próprio formulário) dispostos em forma de árvore hierárquica. Através deste painel, você pode selecionar componentes. Então, selecione o formulário *fCalculadora*, clicando sobre o nome deste, e na guia *Eventos*, crie o evento *OnCreate* e digite o código abaixo em negrito:

```
procedure TfCalculadora.FormCreate(Sender:
TObject);
begin
    DecimalSeparator := ',';
    ThousandSeparator := '.';
end;
```

Agora, pressione F12 até aparecer o formulário, e acrescente nele um TEdit da paleta Standard e mude as propriedades:

**Alignment** → *taRightJustify*

**Color** → *clBlack*

**Left** → 8

**MaxLength** → 15

**Name** → *edVisor*

**Text** → 0

**Top** → 8

**Width** → 216

Clique no sinal de mais ( + ) da propriedade **Font** e mude as propriedades filhas:

**Color** → *clLime*

**Size** → 18

Pressione F12 até exibir o código do formulário. Vamos criar algumas variáveis globais e procedimentos na seção **private**. Digite o código em negrito na posição indicada:

```
private
    signal, sinal_anterior, tecla_anterior: char;
    numero1, numero2: string;
    procedure digito(tecla: char);
    procedure operacao(operador: char);
    { private declarations }
public
    { public declarations }
end;
```

As variáveis *signal* e *sinal\_anterior*, servirão, respectivamente, para guardar o sinal da última e da penúltima operação digitada (seja no teclado ou em um botão da calculadora). A variável *tecla\_anterior* guardará o última tecla ou botão pressionado

(número, vírgula ou operação). E, por fim, as variáveis *numero1* e *numero2* guardarão os números entre uma operação para a posterior execução do cálculo.

Agora, posicione o cursor na linha da **procedure digito** e pressione CTRL + SHIFT + C. A estrutura da *procedure* será criada. Digite o código em negrito:

```
procedure TfCalculadora.digito(tecla: char);
var
  posicao: integer;
begin
  // Testa se a última tecla pressionada ou botão
  // foi de uma operação. Se foi, limpa o visor
  if (tecla_anterior in ['-','+', '/', 'x']) then
    edVisor.Text := '';
  // Se o visor tiver só um 0, limpa o visor
  if (edVisor.Text = '0') and ((tecla <> '0') and
    (tecla <> ',')) then edVisor.Text := '';
  // Evita que seja digitado mais de uma vírgula
  if (tecla = ',') and ((Pos(',',edVisor.Text) > 0)
    or (edVisor.Text='')) then exit;
  // Pega a posição do cursor no visor
  posicao := edVisor.SelStart;
  // Insere a tecla no visor na posição do cursor
  edVisor.Text := Copy(edVisor.Text,0,posicao) +
    tecla + Copy(edVisor.Text,posicao +
    1,Length(edVisor.Text));
  // Retorna o foco para o visor
  edVisor.SetFocus;
  // Deseleciona o texto do visor
  edVisor.SelLength := 0;
  // Reposiciona o cursor no visor
  edVisor.SelStart := posicao + 1;
  // Guarda a última tecla ou botão digitado
  tecla_anterior := tecla;
end;
```

Agora, posicione o cursor na linha da **procedure operacao** e pressione CTRL + SHIFT + C. Na estrutura criada, digite o código em negrito:

```
procedure TfCalculadora.operacao(operador: char);  
var  
    numero_tratado1, numero_tratado2: double;  
begin  
    // Guarda o sinal anterior  
    sinal_anterior := sinal;  
    // Guarda o sinal atual  
    sinal := operador;  
    edVisor.SetFocus;  
    edVisor.SelLength := 0;  
    edVisor.SelStart := Length(edVisor.Text);  
    // Testa se foi pressionada mais de uma vez  
    // seguida uma tecla ou botão de operação  
    if (tecla_anterior in ['+', '-', '/', 'x']) then  
    begin  
        tecla_anterior := sinal;  
        Exit; // Sai da procedure  
    end  
    else  
        tecla_anterior := sinal;  
        // Armazena o primeiro número ou o segundo se  
        // o primeiro já tiver sido armazenado  
        if (numero1 <> '') then  
            numero2 := edVisor.Text  
        else  
            numero1 := edVisor.Text;  
        // Testa se já tem dois números armazenados  
        if (numero1 <> '') and (numero2 <> '') then  
        begin  
            // Converte de texto para número real  
            numero_tratado1 := StrToFloat(numero1);  
            numero_tratado2 := StrToFloat(numero2);  
            // Efetua o calculo de acordo com a operação  
            // escolhida, converte o resultado para texto  
            // e joga no edVisor (visor da calculadora)
```

```
case sinal_anterior of
  'x': edVisor.Text :=
        FloatToStr(
          numero_tratado1*numero_tratado2);
  '/':
    begin
      if ((numero_tratado1 = 0) or
          (numero_tratado2 = 0)) and
          (sinal_anterior = '/') then
        begin
          ShowMessage('Divisão por Zero!');
          edVisor.Text := '';
          operador := '=';
        end
      else
        edVisor.Text :=
          FloatToStr(
            numero_tratado1/numero_tratado2);
      end;
  '+': edVisor.Text :=
        FloatToStr(
          numero_tratado1+numero_tratado2);
  '-': edVisor.Text :=
        FloatToStr(
          numero_tratado1-numero_tratado2);
end;
// Reinicia as variáveis
numero1 := edVisor.Text;
numero2 := '';
sinal_anterior := #0;
// Anula as variáveis caso o operador seja o =
if operador = '=' then begin
  tecla_anterior := #0;
  sinal := #0;
  if Length(edVisor.Text) > 14 then begin
    ShowMessage(
      'Resultado ultrapassou 14 dígitos!');
    // Zera tudo se passar de 14 dígitos
    tecla_anterior := #0;
```

```
sinal := #0;
sinal_anterior := #0;
numero1 := '';
numero2 := '';
edVisor.Text := '0';
edVisor.SetFocus;
edVisor.SelLength := 0;
edVisor.SelStart := 1;
end;
end;
end;
// Reposiciona o cursor no visor
edVisor.SelStart := Length(edVisor.Text);
end;
```

No evento OnKeyPress do **edVisor**, digite o código em negrito:

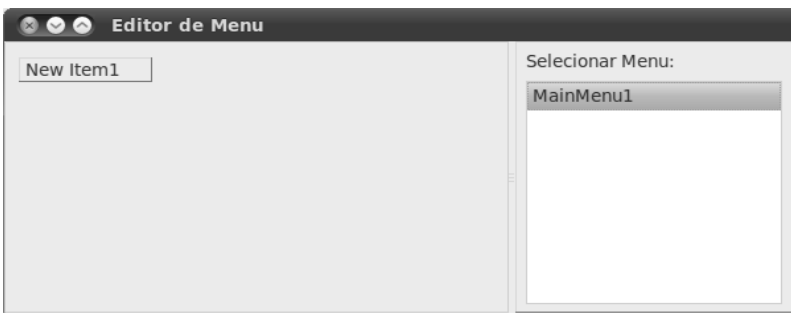
```
procedure TfCalculadora.edVisorKeyPress(Sender:
TObject; var Key: char);
begin
    // Testa se a tecla é um número ou vírgula
    if (key in ['0' .. '9', ',',']) then begin
        // Trata a tecla digitada através da
        // procedure digito
        digito(key);
        // Impede que a tecla seja repetida no visor
        key := #0;
    end else begin
        // Testa se foi digitada uma tecla de operação
        // e trata a situação através da
        // procedure operacao
        case key of
            '+' : operacao(key);
            '-' : operacao(key);
            '/' : operacao(key);
            'x', 'X', '*' : operacao('x');
            #13, '=' : operacao('=');
        end;
        // Testa se a tecla digitada é diferente de
```

```
// BACKSPACE - código ASCII = 8
if key <> #8 then key := #0;
end;
end;
```

No evento OnEnter do **edVisor**, digite o código em negrito:

```
procedure TfCalculadora.edVisorEnter(Sender:
TObject);
begin
    edVisor.SelLength := 0;
    edVisor.SelStart := 1;
end;
```

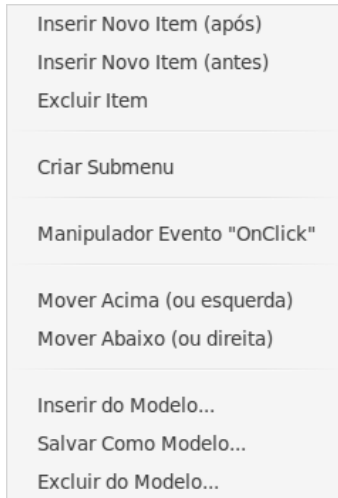
Agora, insira no formulário um componente *TMainMenu* da paleta *Standard*. Este é um componente representado apenas por um ícone no formulário em modo de design (é um componente não visual). O ícone fica oculto quando o programa estiver em execução (o que vai aparecer é o menu). Dê um duplo clique neste ícone para exibir o diálogo de edição do menu.



*Figura 10.2 – Janela do Editor de Menu*

Através deste diálogo podemos criar o menu principal de nossa aplicação (grave bem como abrir este diálogo, porque

vamos usá-lo bastante). Clique com o botão direito do mouse em “*New Item1*” e observe as opções do menu pop-up:



*Figura 10.3 – Menu pop-up*

Clique em “*Criar Submenu*”. Será criado o “*New Item2*” como submenu.

Clique em “*New Item1*” e mude, no *Inspetor de Objetos*, a propriedade *Caption* para *&Calculadora*.

O **&** torna a letra a direita um atalho de teclado para o menu, através da combinação ALT + letra.

Clique em “*New Item2*” e mude as propriedades:

**Name** → *mSair*

**Caption** → *Sair*

**ShortCut** → *Ctrl+Q*



A propriedade *ShortCut* configura um atalho de teclado para o item de menu.

Dê um duplo clique no item de menu “*Sair*” e no evento *OnClick* criado, digite o código:

```
Application.Terminate; //Encerra o programa
```

Abra novamente o *Editor de Menu*, clique com o botão direito do mouse no item “*Calculadora*” e, no menu pop-up, clique em “*Inserir Novo Item (após)*”.

Clique com o botão direito do mouse em “*New Item3*” e, no menu pop-up, clique em “*Criar Submenu*”.

Clique com o botão direito do mouse no item “*New Item4*” e, no menu pop-up, clique em “*Inserir Novo Item (após)*”.

Clique com o botão direito do mouse no item “*New Item5*” e, no menu pop-up, clique em “*Inserir Novo Item (após)*”.

Clique em “*New Item3*” e mude a propriedade *Caption* para *&Editar*. Agora, mude as propriedades dos outros itens:

New Item4:

**Name** → *mRecortar*

**Caption** → *Recortar*

**ShortCut** → *Ctrl+X*

New Item5:

**Name** → *mCopiar*

**Caption** → *Copiar*

**ShortCut** → *Ctrl+C*

New Item6:

**Name** → *mColar*

**Caption** → *Colar*

**ShortCut** → *Ctrl+V*

Agora, clique com o botão direito do mouse no item “*Editar*” e, no menu pop-up, clique em “*Inserir Novo Item (após)*”.

Clique com o botão direito do mouse em “*New Item7*” e, no menu pop-up, clique em “*Criar Submenu*”.

Clique em “*New Item7*” e mude a propriedade Caption para *Aj&uda*. Agora, mude as propriedades do “*New Item8*”:

**Name** → *mSobre*

**Caption** → *Sobre*

**ShortCut** → *Ctrl+U*

Como próximo passo, dê um duplo clique no item de menu “*Recortar*” e no evento *OnClick* criado digite o código:

```
edVisor.CutToClipboard; // Recorta texto do edVisor
```

No evento *OnClick* do item de menu “*Copiar*” digite:

```
edVisor.CopyToClipboard; // Copia texto do edVisor
```

No evento *OnClick* do item de menu “*Colar*” digite o código em **negrito** nas posições indicadas:

```
procedure TfCalculadora.mColarClick(Sender:
TObject);
var
    filtra_colagem: string;
    i: Integer;
```

```
begin
  // Testa se a última tecla pressionada ou botão
  // foi de uma operação. Se foi, limpa o visor e
  // cola o número
  if (tecla_anterior in ['-','+', '/', 'x']) then
    edVisor.Text := '';
  edVisor.PasteFromClipboard;
  filtra_colagem := edVisor.Text;
  edVisor.Text := '';
  // Tira tudo que não for número ou vírgula do
  // texto colado. E impede que a vírgula seja
  // repetida
  for i := 0 to Length(filtra_colagem) do begin
    if (filtra_colagem[i] in ['0'..'9',',']) then
      begin
        if (filtra_colagem[i] = ',') and
          (Pos(',',edVisor.Text) > 0) then
          continue;
        edVisor.Text :=
          edVisor.Text + filtra_colagem[i];
      end;
    end;
  end;
end;
```

No evento *OnClick* do item de menu “Sobre ...” digite o código em negrito:

```
procedure TfCalculadora.mSobreClick(Sender:
TObject);
begin
  Application.MessageBox(
    'Calculadora Básica' + #13 + #13 +
    'Desenvolvendo Aplicativos com Lazarus'
    + #13 + #13 + 'Exemplo 10.1', 'Sobre ...', 0
  );
  // O código ASCII #13 quebra a
  // linha do texto end;
```

Como próximo passo, insira no formulário um componente *TStatusBar* da paleta “*Common Controls*”. Você deve estar vendo agora uma barra de status no seu formulário. Dê um duplo clique nesta barra e você verá a janela mostrada abaixo:

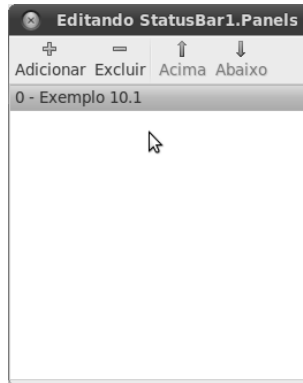


Figura 10.4 – Editor da Barra de Status

Através deste diálogo, podemos adicionar painéis (seções) na barra de status. Clique no botão com o sinal de + (Adicionar) e no Inspetor de Objetos mude as seguintes propriedades da seção criada e feche o diálogo:

**Alignment** → *taCenter*

**Text** → *Exemplo 10.1*

**OBS.:** Algumas propriedades, depois de alteradas, necessitam pressionar ENTER, clicar ou mudar para outra propriedade afim de validar a alteração.

Agora, adicione no formulário vinte *TButtons* e configure as propriedades *Height* e *Width* de cada um para 32 e 48, respectivamente. Organize os botões e mude a propriedade *Caption* de cada um conforme mostrado na Figura 10.1.

Para conseguir os caracteres dos botões  $\div$  e  $\pm$ , copie e cole de um editor de texto (Writer ou Word).

No evento *OnClick* de cada botão de número e o de vírgula, vamos digitar o mesmo código, mudando apenas o caractere enviado para *procedure digito*. Por exemplo, para o botão 5, o código da *procedure OnClick* deste fica assim:

```
digito('5');
```

No evento *OnClick* de cada botão de operação ( +, -,  $\div$ , x, =), chamaremos a *procedure operacao*, enviando o caractere correspondente a operação. Por exemplo, no evento *OnClick* do botão de + o código fica assim:

```
operacao('+');
```

**OBS.:** Para a operação de dividir, cujo botão tem o caractere  $\div$ , o código do *OnClick* fica assim: `operacao('/');`

Para o evento *OnClick* do botão “Bsp” digite o código em negrito:

```
procedure TfCalculadora.btBspClick(Sender:
TObject);
var
    posicao: integer;
begin
    // Pega a posição do cursor no edVisor
    posicao := edVisor.SelStart;
    // Elimina o caractere à esquerda do cursor
    edVisor.Text :=
        Copy(edVisor.Text,0,posicao - 1) +
        Copy(edVisor.Text,posicao +
            1,Length(edVisor.Text));
    // Reposiciona o cursor
    edVisor.SetFocus;
```

```
edVisor.SelLength := 0;  
edVisor.SelStart := posicao - 1;  
end;
```

Para o evento *OnClick* do botão “CE” digite o código:

```
edVisor.Text := '0'; // Zera o edVisor  
edVisor.SetFocus;  
edVisor.SelLength := 0;  
edVisor.SelStart := 1;
```

Para o evento *OnClick* do botão “Clr” digite o código:

```
// Zera todas as variáveis e o edVisor  
tecla_anterior := #0;  
sinal := #0;  
sinal_anterior := #0;  
numero1 := '';  
numero2 := '';  
edVisor.Text := '0';  
edVisor.SetFocus;  
edVisor.SelLength := 0;  
edVisor.SelStart := 1;
```

Para o evento *OnClick* do botão “±” digite o código:

```
// Acrescenta menos ( - ) ou tira  
if (pos('-',edVisor.Text) > 0) then  
    edVisor.Text :=  
        StringReplace(edVisor.Text, '-', '', [rfReplaceAll])  
else  
    if (edVisor.Text <> '') and  
        (edVisor.Text <> '0') then  
        edVisor.Text := '-' + edVisor.Text;  
// Reposiciona o cursor  
edVisor.SetFocus;  
edVisor.SelLength := 0;  
edVisor.SelStart := Length(edVisor.Text);
```

Está pronta a nossa calculadora. Você pode incluí-la nos seus futuros programas comerciais. Então, execute e teste o programa.

### Criando um Editor de Texto Simples:

O editor que criaremos é similar ao *Bloco de Notas* do Windows ou *gedit* do Ubuntu. Ao final do desenvolvimento, nosso editor vai ficar como mostrado na figura abaixo:

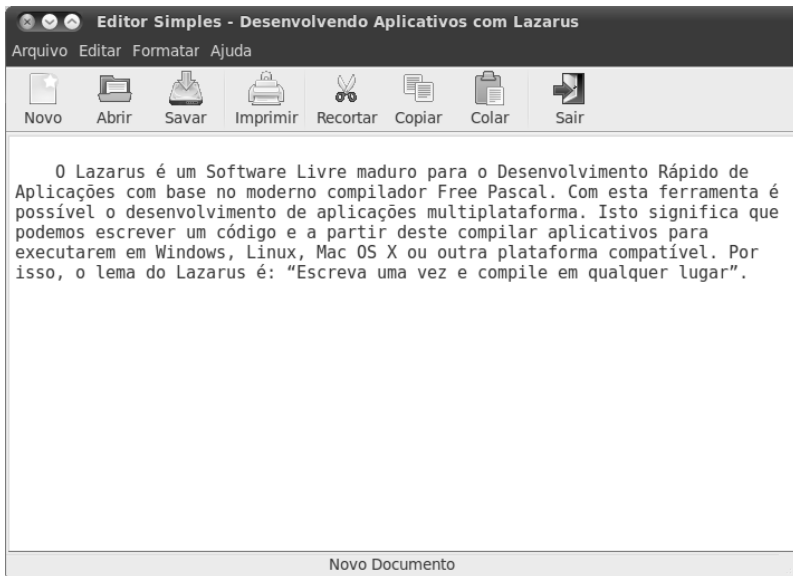


Figura 10.5 – Editor de Texto Simples

Crie uma pasta na pasta *livro\_projetos* com o nome *exemplo10\_2*. No Lazarus, crie um novo projeto do tipo Aplicação. Salve na pasta *exemplo10\_2* o projeto com o nome *editor\_simples.lpi* e a unit com o nome *ueditor.pas*.

Mude o nome do formulário para *fEditor* e o Caption para *“Editor Simples – Desenvolvendo Aplicativos com Lazarus”*. Também, mude as seguintes propriedades:

**Position** → *poDesktopCenter*

**Height** → 483

**Width** → 710

Baixe e descompacte o arquivo do link:

[http://www.jpsoft.com.br/downloads/icones\\_editor.zip](http://www.jpsoft.com.br/downloads/icones_editor.zip)

Agora adicione ao formulário dois *TImageList* da paleta *“Common Controls”*. Este componente armazena uma lista de imagens para serem usadas por outros componentes, como *TMainMenu* e *TToolBar* (Barra de Ferramentas).

Mude as propriedades Height e Width do *ImageList1* para 32, depois dê um duplo clique neste e vai surgir o diálogo abaixo:



Figura 10.6 – Editor “ImageList”



Clique no botão **Adicionar...**, localize o arquivo *icones32.png* na pasta *icones\_editor*, que foi descompactada, selecione-o e clique no botão **Abrir**. Vai surgir uma mensagem perguntando se você quer separar a imagem, clique em *Sim* e agora clique em *OK*.

Não é preciso mudar nenhuma propriedade do *ImageList2*. Dê um duplo clique sobre este componente, no diálogo que surge clique no botão **Adicionar...**, localize o arquivo *icones16.png* na pasta *icones\_editor*, que foi descompactada, selecione-o e clique no botão **Abrir**. Vai surgir uma mensagem perguntando se você quer separar a imagem, clique em *Sim* e agora clique em *OK*.

Insira no formulário um componente *TMainMenu* da paleta *Standard* e mude neste a propriedade *Images* para *ImageList2*.

Dê um duplo clique no componente *MainMenu1* e através do “*Editor de Menu*”, conforme já aprendido, crie os seguintes menus e itens de menu, e altere as propriedades como segue (as propriedades que estiverem em branco na tabela, significa que não devem ser alteradas):

Menu Arquivo (Caption → &Arquivo)				
Item	Name	Caption	ShortCut	ImageIndex
1	mNovo	Novo	Ctrl + N	1
2	mAbrir	Abrir	Ctrl + A	2
3	mSalvar	Salvar	Ctrl + S	4
4		-		
5	mImprimir	Imprimir	Ctrl + P	3
6		-		

7	mSair	Sair	Ctrl + Q	0
---	-------	------	----------	---

**Menu Editar (Caption → &Editar)**

Item	Name	Caption	ShortCut	ImageIndex
1	mRecortar	Recortar	Ctrl + X	6
2	mCopiar	Copiar	Ctrl + C	5
3	mColar	Colar	Ctrl + V	7
4		-		
5	mSelecionar	Selecionar Tudo	Ctrl + T	8

**Menu Formatar (Caption → &Formatar)**

Item	Name	Caption	ShortCut	ImageIndex
1	mFonte	Fonte ...	Ctrl + F	12
2	mCorFonte	Cor da Fonte ...		13
3	mCorPapel	Cor do Papel ...		13
4		-		
5	mEsquerda	Alinhar à Esquerda	Ctrl + L	10
6	mDireita	Alinhar à Direita	Ctrl + R	11
7	mCentro	Alinhar ao Centro	Ctrl + E	9
8		-		
9	mQuebrar	Quebrar Linha Automaticamente	Ctrl + O	

Menu Ajuda (Caption → <i>Aj&amp;uda</i> )				
Item	Name	Caption	ShortCut	ImageIndex
1	mSobre	Sobre ...	Ctrl + U	14

Quando se coloca um traço ( - ) no Caption de um *item de menu* é criado um separador.

No item de menu *mQuebrar*, mude a propriedade Checked para *True*.

Agora, coloque no formulário um *TToolBar* e mude as seguintes propriedades:

**AutoSize** → *True*

**ButtonHeight** → 56

**ButtonWidth** → 64

**EdgeBorders** > **edBottom** → *True*

**Images** → *ImageList1*

**ShowCaption** → *True*

Clicando com o botão direito do mouse no *ToolBar1*, é mostrado um menu pop-up com opções para inserir botões, separadores e etc. Insira e configure as propriedades dos itens mostrados na tabela abaixo:

Tipo	Caption	ImageIndex
Botão	Novo	4
Botão	Abrir	5
Botão	Salvar	7
Separador		
Botão	Imprimir	6

<b>Tipo</b>	<b>Caption</b>	<b>ImageIndex</b>
Separador		
Botão	Recortar	1
Botão	Copiar	0
Botão	Colar	2
Separador		
Botão	Sair	3

Como próximo passo, insira no formulário um componente *TStatusBar* da paleta “*Common Controls*”. Dê um duplo clique na barra de status do formulário.

Através do diálogo que surge, clique no botão com o sinal de + (Adicionar) e, no *Inspetor de Objetos*, mude as seguintes propriedades da seção criada e feche o diálogo:

**Alignment** → *taCenter*  
**Text** → *Novo Documento*

Insira um *TPanel* no formulário e mude as propriedades:

**Align** → *alClient*  
**BorderSpacing > Around** → 2  
**Color** → *clWhite*

Dentro do *TPanel* (Panel1) insira um *TMemo* e configure as propriedades:

**Align** → *alClient*  
**BorderSpacing > Around** → 5  
**BorderStyle** → *bsNone*  
**Lines** → (clique no botão de reticências ( ... ) e apague o texto)

**ScrollBars** → *ssBoth*

**Name** → *Documento*

Em qualquer posição no formulário, insira da paleta *Dialogs* um *TSaveDialog*, um *TColorDialog*, um *TFontDialog* e um *TOpenDialog*.

Pressione *F12* até exibir o código do formulário e na seção **uses** digite *Printers*, como mostrado abaixo (esta *unit* contém o objeto *printer* que possibilita enviarmos textos formatados para impressora):

```
uses
  Classes, SysUtils, FileUtil, Forms, Controls,
  Graphics, Dialogs, StdCtrls, PrintersDlgs, Menus,
  ComCtrls, ExtCtrls, Printers;
```

Dê um duplo clique no componente *MainMenu1*, dê um duplo clique no item *Novo* e digite o código em negrito para o evento *OnClick*:

```
procedure TfEditor.mNovoClick(Sender: TObject);
begin
  if MessageDlg(
    'Novo Documento',
    'Criar Novo Documento de Texto?',
    mtConfirmation,
    mbYesNo, ''
  ) <> 6 then exit;
  Documento.Clear; // Limpa o texto do TMemo
  StatusBar1.Panels[0].Text := 'Novo Documento';
end;
```

No evento *OnClick* do item de menu *Abrir* digite o código em negrito:

```
procedure TfEditor.mAbrirClick(Sender: TObject);
begin
    if OpenFileDialog1.Execute then
    begin
        Documento.Lines.LoadFromFile(
            OpenFileDialog1.FileName);
        StatusBar1.Panels[0].Text :=
            OpenFileDialog1.FileName;
    end;
end;
```

No evento *OnClick* do item de menu *Salvar* digite o código em negrito:

```
procedure TfEditor.mSalvarClick(Sender: TObject);
begin
    if SaveDialog1.Execute then
        Documento.Lines.SaveToFile(SaveDialog1.FileName);
end;
```

No evento *OnClick* do item de menu *Imprimir* digite o código em negrito:

```
procedure TfEditor.mImprimirClick(Sender: TObject);
var
    i: Integer;
begin
    if trim(Documento.Text) = '' then begin
        ShowMessage('Não há texto para impressão?');
        Exit;
    end;
    if MessageDlg('Imprimir',
        'Confirma a Impressão do Documento?',
        mtConfirmation,mbYesNo,'') <> 6 then exit;
    Printer.BeginDoc; // Inicia impressão
    for i := 1 to Documento.Lines.Count do
        Printer.Canvas.TextOut(1, (i - 1) *
            Printer.Canvas.TextHeight('Ág'),
```

```
// Comando que imprime as linhas
Documento.Lines[i-1]);
Printer.EndDoc; // Encerra impressão
end;
```

O código acima é apenas um exemplo simples de como podemos imprimir o texto de um TMemo. Assim, para a impressão sair corretamente, quebre as linhas do texto usando a tecla ENTER para se ajustarem a janela do editor no tamanho normal (sem estar maximizada).

No evento *OnClick* do item de menu *Sair*, digite o código seguinte entre o **begin** e o **end** da procedure:

```
Application.Terminate;
```

No evento *OnClick* do item de menu *Recortar*, digite o código seguinte entre o **begin** e o **end** da procedure:

```
Documento.CutToClipboard;
```

No evento *OnClick* do item de menu *Copiar*, digite o código seguinte entre o **begin** e o **end** da procedure:

```
Documento.CopyToClipboard;
```

No evento *OnClick* do item de menu *Colar*, digite o código seguinte entre o **begin** e o **end** da procedure:

```
Documento.PasteFromClipboard;
```

No evento *OnClick* do item de menu *“Selecionar Tudo”*, digite o código seguinte entre o **begin** e o **end** da procedure:

```
Documento.SelectAll;
```

No evento *OnClick* do item de menu “*Fonte ...*”, digite o código seguinte entre o **begin** e o **end** da procedure:

```
FontDialog1.Font := Documento.Font;
if FontDialog1.Execute then
    Documento.Font := FontDialog1.Font;
```

No evento *OnClick* do item de menu “*Cor da Fonte ...*”, digite o código seguinte entre o **begin** e o **end** da procedure:

```
ColorDialog1.Color := Documento.Font.Color;
if ColorDialog1.Execute then
    Documento.Font.Color := ColorDialog1.Color;
```

No evento *OnClick* do item de menu “*Cor do Papel ...*”, digite o código seguinte entre o **begin** e o **end** da procedure:

```
ColorDialog1.Color := Documento.Color;
if ColorDialog1.Execute then begin
    Documento.Color := ColorDialog1.Color;
    Panel1.Color := ColorDialog1.Color;
end;
```

No evento *OnClick* do item de menu “*Alinhar à Esquerda*”, digite o código seguinte entre o **begin** e o **end** da procedure:

```
Documento.Alignment := taLeftJustify;
```

No evento *OnClick* do item de menu “*Alinhar à Direita*”, digite o código seguinte entre o **begin** e o **end** da procedure:

```
Documento.Alignment := taRightJustify;
```

No evento *OnClick* do item de menu “*Alinhar ao Centro*”, digite o código seguinte entre o **begin** e o **end** da procedure:

```
Documento.Alignment := taCenter;
```



No evento *OnClick* do item de menu “*Quebrar Linha Automaticamente*”, digite o código seguinte entre o **begin** e o **end** da procedure:

```
if Documento.WordWrap then begin
    Documento.WordWrap := false; // Desativa quebra
    mQuebrar.Checked := false;
end else begin
    Documento.WordWrap := true; // Ativa quebra
    mQuebrar.Checked := true;
end;
```

No evento *OnClick* do item de menu “*Sobre ...*”, digite o código seguinte entre o **begin** e o **end** da procedure:

```
Application.MessageBox(
'Editor Simples' + #13 + #13 +
'Desenvolvendo Aplicativos com Lazarus' + #13 +
#13 + 'Exemplo 10.2', 'Sobre ...', 0);
```

Ligue os eventos *OnClick* dos botões da barra de ferramentas (*ToolBar1*) aos eventos *OnClick* dos itens de menu que exercem a mesma função.

Por exemplo, selecione o botão *Novo* e, no *Inspetor de Objetos*, na guia *Eventos*, clique ao lado do evento *OnClick*, clique no botão com a seta para baixo e selecione *mNovoClick* que é a *procedure* do evento *OnClick* do menu *Novo*.

Prossiga configurando os outros botões. Isso evita que um mesmo código seja repetido desnecessariamente. Depois, clique no objeto *OpenDialog1*, para selecioná-lo, e mude as seguintes propriedades:

**DefaultExt** → *.txt*

**Filter** → *txt*

Selecione o formulário *fEditor* e digite o código em negrito para o evento OnCreate deste:

```
procedure TfEditor.FormCreate(Sender: TObject);
begin
    {$IFDEF MSWINDOWS}
    mCorFonte.Visible := False;
    {$ENDIF}
end;
```

O “Apêndice - B” explica o que este código faz e quando ele é executado. Mas você pode descobrir sozinho compilando e executando o editor no Linux e no Windows. Assim, concluímos o nosso editor. Execute e teste o programa.

### Criando um Visualizador de Imagens:

O visualizador de imagens que desenvolveremos é similar ao exemplo encontrado na pasta do Lazarus. Mas, vamos usar alguns componentes especializados e um pouco de matemática para o posicionamento da imagem. Ao final do desenvolvimento, nosso programa ficará como mostrado na figura abaixo:



Figura 10.7 – Visualizador de Imagens

Crie uma pasta na pasta *livro\_projetos* com o nome *exemplo10\_3*. No Lazarus, crie um novo projeto do tipo Aplicação. Salve na pasta *exemplo10\_3* o projeto com o nome *visualizador.lpi* e a unit com o nome *uprincipal.pas*.

Mude o nome do formulário para *fVisualiza* e o Caption para “Visualizador de Imagens – Desenvolvendo Aplicativos com Lazarus”. Também, mude as seguintes propriedades:

**Position** → *poDesktopCenter*

**Height** → *500*

**Width** → *800*

Baixe o arquivo do link:

[http://www.jpsoft.com.br/downloads/icones\\_visualizador.png](http://www.jpsoft.com.br/downloads/icones_visualizador.png)

Adicione ao formulário um *TImageList* da paleta “Common Controls”.

Altere as propriedades Height e Width do *ImageList1* para 24, depois dê um duplo clique neste e no diálogo que surge, clique no botão **Adicionar...**, localize o arquivo *icones\_visualizador.png*, que foi baixado, selecione-o e clique no botão **Abrir**. Vai surgir uma mensagem perguntando se você quer separar a imagem, clique em *Sim* e no diálogo clique em *OK*.

Prosseguindo, insira no formulário um componente *TMainMenu* da paleta *Standard* e mude neste a propriedade *Images* para *ImageList1*.

Dê um duplo clique no componente *MainMenu1* e através do “Editor de Menu”, conforme já aprendido, crie os seguintes menus e itens de menu, e altere as propriedades como segue:

<b>Menu Arquivo (Caption → &amp;Arquivo / Name → mArquivo)</b>				
<b>Item</b>	<b>Name</b>	<b>Caption</b>	<b>ShortCut</b>	<b>ImageIndex</b>
1	mAbrir	Abrir Arquivo	Ctrl + A	0
2	mPasta	Selecionar Pasta	Ctrl + P	1
3		-		
4	mSair	Sair	Ctrl + S	5

<b>Menu Imagem (Caption → &amp;Imagem / Name → mImagem)</b>				
<b>Item</b>	<b>Name</b>	<b>Caption</b>	<b>ShortCut</b>	<b>ImageIndex</b>
1	mAjustar	Ajustar à Janela	Ctrl + J	11
2	mNormal	Tamanho Real	Ctrl + T	2
3	mMaisZoom	Mais Zoom	Ctrl + F9	3
4	mMenosZoom	Menos Zoom	Ctrl + F10	4
5		-		
6	mPrimeira	Primeira	Ctrl + Q	6
7	mAnterior	Anterior	Ctrl + W	9
8	mProxima	Próxima	Ctrl + E	8
9	mUltima	Última	Ctrl + R	7

<b>Menu Ajuda (Caption → Aj&amp;uda / Name → mAjudas)</b>				
<b>Item</b>	<b>Name</b>	<b>Caption</b>	<b>ShortCut</b>	<b>ImageIndex</b>
1	mSobre	Sobre ...	Ctrl + U	10

Agora, coloque no formulário um *TToolBar* e mude as seguintes propriedades:

**AutoSize** → *True*  
**ButtonHeight** → 32  
**ButtonWidth** → 32  
**EdgeBorders** > **edBottom** → *True*  
**Images** → *ImageList1*  
**ShowHint** → *True*

Como já considerado, clicando com o botão direito do mouse no *ToolBar1*, é mostrado um menu *pop-up* com opções para inserir botões, separadores e etc.

Usando este menu, insira e configure as propriedades dos catorze itens mostrados na tabela seguinte:

<b>Tipo</b>	<b>Name</b>	<b>Hint</b>	<b>ImageIndex</b>
Botão	btArquivo	Abrir Arquivo	0
Botão	btPasta	Selecionar Pasta	1
Separador			
Botão	btAjusta	Ajustar na Janela	11
Botão	btNormal	Tamanho Real	2
Botão	btMaisZoom	Mais Zoom	3
Botão	btMenosZoom	Menos Zoom	4
Separador			
Botão	btPrimeira	Primeira	6
Botão	btAnterior	Anterior	9
Botão	btProxima	Próxima	8
Botão	btUltima	Última	7
Separador			
Botão	btSair	Sair	5

Em seguida, insira no formulário um componente *TStatusBar* da paleta “*Common Controls*”. Dê um duplo clique na barra de status do formulário.

Através do diálogo que surge, clique no botão com o sinal de + (Adicionar) e, no *Inspetor de Objetos*, mude a seguinte propriedade da seção criada e feche o diálogo:

**Alignment** → *taCenter*

Insira no formulário um *TFileListBox* da paleta *Misc* e mude as propriedades:

**Align** → *alLeft*

**Mask** → *\*.bmp;\*.xpm;\*.png;\*.ico;\*.jpg;\*.jpeg;\*.jfif;\*.tif;\*.tiff;\*.gif*

**Name** → *ListaArquivos*

**Width** → *200*

Novamente, no formulário, adicione um componente *TSplitter* (barra divisória ajustável) da paleta *Additional* e altere as propriedades:

**Beveled** → *True*

**MinSize** → *200*

Em seguida um *TScrollBar* da paleta *Additional*. Este componente é um contêiner para outros componentes, similar ao *TPanel*. Altere as propriedades:

**Align** → *alClient*

**Name** → *sbVisor*

O próximo passo é inserir um *TImage* da paleta *Additional* dentro do *TScrollBar* (*sbVisor*). Altere as propriedades:

**Center** → *True*  
**Name** → *Imagem*  
**Proportional** → *True*  
**Stretch** → *True*  
**Top** → *0*  
**Left** → *0*

Da paleta *Dialogs*, coloque no formulário, um *TOpenPictureDialog* e mude seu nome para *AbreImagem*, em seguida, um *TSelectDirectoryDialog* e altere o nome deste para *SelecionaPasta*.

Vamos, então, codificar os eventos necessários. Primeiramente, crie uma variável global na unit *uprincipal* como indicado abaixo:

```
var  
    fVisualiza: TfVisualiza;  
    Opcao: Integer;
```

implementation

A variável *Opcao* será usada para armazenar um número indicador do último botão de ajuste de imagem que foi clicado. Esta informação é importante para poder ser feito o reajuste da imagem quando a janela for redimensionada.

Segue o código para cada evento *OnClick* dos botões do componente *TToolBar*:

Código do evento *OnClick* do botão *btArquivo*:

```
if AbreImagem.Execute then  
begin  
    try
```

```
// Carrega no TImage a imagem escolhida
Imagem.Picture.LoadFromFile(
    AbreImagem.FileName);
except
    ShowMessage(
        'Arquivo de imagem excluído ou inválido!');
    Exit;
end;
ListaArquivos.Directory :=
    ExtractFileDir(AbreImagem.FileName);
ListaArquivos.ItemIndex :=
    ListaArquivos.Items.IndexOf(ExtractFileName(
        AbreImagem.FileName));
// Executa o código do evento OnClick do botão
btAjusta
    btAjusta.Click;
// Escreve na barra de status o caminho e o nome
do
    // arquivo selecionado
    StatusBar1.Panels[0].Text := AbreImagem.FileName;
end;
```

Código do evento *OnClick* do botão *btPasta*:

```
if SelecionaPasta.Execute then
begin
    ListaArquivos.Directory :=
SelecionaPasta.FileName;
    if ListaArquivos.Count = 0 then
    begin
        // Limpa a imagem do TImage
        Imagem.Picture.Clear;
        StatusBar1.Panels[0].Text := '';
        Exit;
    end;
    ListaArquivos.ItemIndex := 0;
    try
```



```
Imagem.Picture.LoadFromFile(
    ListaArquivos.Directory +
    DirectorySeparator +
    ListaArquivos.GetSelectedText);
StatusBar1.Panels[0].Text :=
    ListaArquivos.Directory +
    DirectorySeparator +
    ListaArquivos.GetSelectedText;
except
    ShowMessage(
        'Arquivo de imagem excluído ou inválido!');
    StatusBar1.Panels[0].Text := '';
    Exit;
end;
btAjusta.Click;
end;
```

Código do evento *OnClick* do botão *btAjusta*:

```
sbVisor.VertScrollBar.Position := 0;
sbVisor.HorzScrollBar.Position := 0;
Imagem.Top := 0;
Imagem.Left := 0;
//Extensão da ScrollBar
sbVisor.VertScrollBar.Range := 0;
//Extensão da ScrollBar
sbVisor.HorzScrollBar.Range := 0;
Imagem.Width := sbVisor.Width;
Imagem.Height := sbVisor.Height;
Opcao := 1;
```

Código do evento *OnClick* do botão *btNormal*:

```
Imagem.Top := 0;
Imagem.Left := 0;
sbVisor.AutoScroll := True;
Imagem.Width := Imagem.Picture.Width;
Imagem.Height := Imagem.Picture.Height;
```

```
sbVisor.VertScrollBar.Position :=  
    trunc((sbVisor.VertScrollBar.Range -  
        sbVisor.VertScrollBar.Page) / 2);  
sbVisor.HorzScrollBar.Position :=  
    trunc((sbVisor.HorzScrollBar.Range -  
        sbVisor.HorzScrollBar.Page) / 2);  
Opcao := 2;
```

Código do evento *OnClick* do botão *btMaisZoom*:

```
sbVisor.VertScrollBar.Position := 0;  
sbVisor.HorzScrollBar.Position := 0;  
sbVisor.AutoScroll := True;  
Imagem.Width := Imagem.Width + 50;  
Imagem.Height := Imagem.Height + 50;  
Imagem.Top :=  
    trunc((sbVisor.Height - Imagem.Height) / 2);  
Imagem.Left :=  
    trunc((sbVisor.Width - Imagem.Width) / 2);  
Opcao := 3;
```

Código do evento *OnClick* do botão *btMenosZoom*:

```
sbVisor.VertScrollBar.Position := 0;  
sbVisor.HorzScrollBar.Position := 0;  
sbVisor.AutoScroll := True;  
Imagem.Width := Imagem.Width - 50;  
Imagem.Height := Imagem.Height - 50;  
Imagem.Top :=  
    trunc((sbVisor.Height - Imagem.Height) / 2);  
Imagem.Left :=  
    trunc((sbVisor.Width - Imagem.Width) / 2);  
Opcao := 4;
```

Código do evento *OnClick* do botão *btPrimeira*:

```
ListaArquivos.ItemIndex := 0;  
ListaArquivos.Click;
```

Código do evento *OnClick* do botão *btAnterior*:

```
if not (ListaArquivos.ItemIndex = 0) then
  ListaArquivos.ItemIndex :=
ListaArquivos.ItemIndex - 1;
ListaArquivos.Click;
```

Código do evento *OnClick* do botão *btProxima*:

```
if not(ListaArquivos.ItemIndex =
ListaArquivos.Count)
then ListaArquivos.ItemIndex :=
  ListaArquivos.ItemIndex + 1;
ListaArquivos.Click;
```

Código do evento *OnClick* do botão *btUltima*:

```
ListaArquivos.ItemIndex := ListaArquivos.Count - 1;
ListaArquivos.Click;
```

Código do evento *OnClick* do botão *btSair*:

```
Application.Terminate;
```

Para não repetir código, ligue os eventos *OnClick* dos itens de menu aos eventos *OnClick*, já criados, dos botões da barra de ferramentas (*ToolBar1*) que exercem a mesma função (Observe que fizemos o mesmo no desenvolvimento do *Editor Simples*, mas de modo inverso).

Segue o código do evento *OnClick* do item de menu *mSobre*:

```
Application.MessageBox(
  'Visualizador de Imagens' + #13 + #13 +
  'Desenvolvendo Aplicativos com Lazarus' + #13 +
  #13 + 'Exemplo 10.3', 'Sobre ...', 0);
```

Agora, vamos criar o código que vai manipular o evento *OnResize* do componente *TScrollBar*:

```
case Opcao of
  1: btAjusta.Click;
  2:
  begin
    sbVisor.VertScrollBar.Position :=
      trunc((sbVisor.VertScrollBar.Range -
        sbVisor.VertScrollBar.Page) / 2);
    sbVisor.HorzScrollBar.Position :=
      trunc((sbVisor.HorzScrollBar.Range -
        sbVisor.HorzScrollBar.Page) / 2);
  end;
  3, 4:
  begin
    Imagem.Top :=
      trunc((sbVisor.Height - Imagem.Height) / 2);
    Imagem.Left :=
      trunc((sbVisor.Width - Imagem.Width) / 2);
  end;
end;
```

O evento *OnResize* ocorre quando o *TScrollBar* é redimensionado. Visto que a propriedade *Align* dele está em *alClient* (ajustado ao espaço restante do formulário), quando o formulário é redimensionado o *TScrollBar*, também, é. Assim, o código anterior ajusta a imagem para o novo tamanho, de acordo com o último botão de ajuste pressionado no *TToolBar*.

Por último, criamos o evento *OnClick* para o componente *TFileListBox* (ListaArquivos):

```
if ListaArquivos.ItemIndex = -1 then Exit;
try
  Imagem.Picture.LoadFromFile(
    ListaArquivos.Directory +
    DirectorySeparator +
```

```
        ListaArquivos.GetSelectedText);  
    StatusBar1.Panels[0].Text :=  
        ListaArquivos.Directory + DirectorySeparator +  
        ListaArquivos.GetSelectedText;  
    if Opcao = 0 then btAjusta.Click;  
except  
    ShowMessage(  
        'Arquivo de imagem excluído ou inválido!');  
    StatusBar1.Panels[0].Text := '';  
end;
```

Concluimos aqui o último exemplo deste capítulo.

Observe que, em todos os exemplos, os códigos desenvolvidos não foram comentados exhaustivamente linha por linha. O objetivo disto é estimular você leitor iniciante a estudar os códigos apresentados, fazer pesquisas e modificações, verificar para que servem as outras propriedades dos componentes usados, avaliar o que poderia ser melhorado, executar e usar os exemplos de modo prático.

# Conceitos Básicos sobre Banco de Dados e SQL

Um banco de dados é um conjunto de informações que se relacionam. O acesso aos dados é feito por um software conhecido como Sistema Gerenciador de Banco de Dados (SGDB).

O modelo de organização dos dados mais usado é o relacional, baseado em tabelas.

### **Tabelas:**

É um agrupamento de dados dentro do banco de dados. As linhas representam registros e as colunas representam os campos.

### **Chave Primária:**

É um campo que torna única cada linha da tabela.

### **Chave Estrangeira:**

É um campo de uma tabela que faz referência ao campo chave primária de outra tabela. É usado para relacionar os dados das tabelas.

### **Transações:**

É um conjunto de procedimentos executados no banco de dados como uma única ação. Se algum procedimento do conjunto falhar, toda a transação é desfeita (rollback). Caso não haja falhas, a transação é efetivada (commit).

### **Aplicativos de Banco de Dados:**

Um aplicativo de banco de dados fornece a estrutura necessária para o armazenamento de informações. Os mais conhecidos no mundo do software livre são: PostgreSQL, Firebird e MySQL. Outros ideais para pequenos projetos são: SQLite3 e DBF.

### **Criar e Gerenciar Banco de Dados:**

A maneira mais fácil e prática de se criar e gerenciar um banco de dados é usando um aplicativo com interface visual. Estes aplicativos possuem recursos para criar as diversas estruturas de um banco de dados relacional (tabelas, índices, etc) e executar instruções SQL.

O PostgreSQL geralmente vem com o aplicativo pgAdmin III que é multiplataforma. Mas, um outro aplicativo muito bom é o EMS SQL Manager for PostgreSQL Freeware, só para Windows.

Para o Firebird recomendo o FlameRobin que é multiplataforma ou o IBExpert Personal, só para Windows. E, também, só para Windows, ainda temos o EMS SQL Manager for InterBase/Firebird Freeware.

Já o MySQL, possui dois aplicativos próprios, que são: MySQL Administrator e MySQL Query Browser, instalados separadamente, e são multiplataforma. Mas, também, há o EMS SQL Manager for MySQL Freeware, só para Windows.

E para o SQLite3 temos o SQLite Studio, que é multiplataforma.

Estes gerenciadores de banco de dados e aplicativos citados podem ser encontrados e baixados facilmente através da Internet. No Ubuntu, você pode encontrá-los na Central de Programas do Ubuntu. (Veja o *Apêndice - E*).

### Linguagem SQL:

A linguagem SQL (Structured Query Language - Linguagem Estruturada de Pesquisa) foi criada para ser uma linguagem padrão para consulta, atualização e manipulação de dados em um banco de dados relacional.

Portanto, podemos usar os comandos do SQL para extrair ou atualizar as linhas (registros) de uma ou mais tabelas que atendam as condições especificadas, manipulando, assim, somente os dados que sejam de nosso interesse.

A seguir consideramos como trabalhar com SQL nos seguintes gerenciadores de banco de dados: PostgreSQL, Firebird, MySQL e SQLite3.

### Tipos de Campos:

Segue uma tabela com os tipos de campos mais usados:

<b>Campo</b>	<b>Descrição</b>	<b>Bancos</b>
VARCHAR(n)	Armazena linha de texto com tamanho variando até o limite especificado.	Firebird PostgreSQL MySQL SQLite3
CHAR(n)	Armazena texto com tamanho fixo, conforme especificado.	Firebird PostgreSQL MySQL SQLite3



<b>Campo</b>	<b>Descrição</b>	<b>Bancos</b>
DATETIME	Armazena data e hora.	PostgreSQL MySQL SQLite3
DATE	Armazena datas.	Firebird PostgreSQL MySQL SQLite3
TIME	Armazena hora.	Firebird PostgreSQL MySQL SQLite3
SMALLINT	Armazena números inteiros na faixa de -32768 .. 32767	Firebird PostgreSQL MySQL
INTEGER	Armazena números inteiros com precisão de 32bits.	Firebird PostgreSQL MySQL SQLite3
BIGINT	Armazena números inteiros com precisão de 64bits.	Firebird PostgreSQL MySQL
DOUBLE PRECISION	Ideal para armazenar valores monetários.	Firebird PostgreSQL MySQL SQLite3
FLOAT	Armazena números reais.	Firebird PostgreSQL MySQL

<b>Campo</b>	<b>Descrição</b>	<b>Bancos</b>
TEXT	Armazena textos grandes (memorando).	PostgreSQL MySQL SQLite3
BLOB	Armazena dados binários, como, por exemplo, imagens, textos e sons.	Firebird PostgreSQL MySQL SQLite3
BOOLEAN	Armazena informação lógica (True/False ou Verdadeiro/Falso).	PostgreSQL MySQL SQLite3

Um detalhe importante é que o SQLite3 não possui tipos de dados, mas apenas classes de armazenamento, que são: NULL, INTEGER, REAL, TEXT e BLOB. Os tipos de dados mencionados na tabela acima apenas fazem referência a uma das classes de armazenamento.

Outro ponto está relacionado ao Firebird que não tem o tipo BOOLEAN. Neste caso, podemos usar o tipo SMALLINT com os valores 0 para False (Falso) e 1 para True (Verdadeiro) para simular o tipo BOOLEAN.

Atente que em todos os servidores, os nomes dos campos e das tabelas não podem conter caracteres acentuados, caracteres especiais, espaços e cedilha. Devem começar com uma letra ou underline ( \_ ) e podem conter números, letras e/ou underline.

Note também que os nomes dos campos e das tabelas no MySQL são sensíveis a maiúsculas e minúsculas.

### Criação de Tabelas:

A sintaxe para criação de uma tabela no PostgreSQL, Firebird, MySQL ou SQLite3 (com diferença na declaração da chave primária neste último) segue basicamente a seguinte estrutura:

```
CREATE TABLE [nome_da_tabela] (  
  [nome_do_campo_1] [tipo_do_campo] [definições],  
  [nome_do_campo_2] [tipo_do_campo] [definições],  
  ...  
  [nome_do_campo_N] [tipo_do_campo] [definições],  
PRIMARY KEY ([lista_de_campos])  
);
```

Segue exemplo para cada banco de dados:

#### ➤ Exemplo em PostgreSQL:

```
CREATE TABLE EXEMPLO (  
  CODIGO SERIAL NOT NULL,  
  PRODUTO VARCHAR(100),  
  PRECO DOUBLE PRECISION,  
  ATIVO BOOLEAN,  
PRIMARY KEY(CODIGO)  
);
```

No PostgreSQL, o tipo SERIAL define um campo como auto-numérico. É o caso do campo CODIGO.

O parâmetro NOT NULL indica que o campo deve ter sempre um valor.

Observe que PRIMARY KEY(CODIGO) define o campo CODIGO como a chave primária da tabela EXEMPLO.

### ➤ Exemplo em Firebird:

```
CREATE TABLE EXEMPLO (  
  CODIGO BIGINT NOT NULL,  
  PRODUTO VARCHAR(100),  
  PRECO DOUBLE PRECISION,  
  ATIVO SMALLINT DEFAULT 1,  
  PRIMARY KEY(CODIGO)  
);
```

O parâmetro **DEFAULT** serve para indicarmos o valor padrão para o campo, caso não seja informado nenhum.

Para definirmos o campo **CODIGO** como auto-numérico, precisamos criar um **GENERATOR** (gerador de números sequenciais) e um **TRIGGER BEFORE INSERT** (procedimento que é disparado antes de um novo registro ser incluído na tabela).

Para facilitar a vida dos iniciantes, os aplicativos FlameRobin e IBExpert Personal, geram toda esta estrutura de forma automática. Veja abaixo o código em SQL, gerado pelo FlameRobin para criar o **GENERATOR** e o **TRIGGER** para o campo **CODIGO** da nossa tabela **EXEMPLO**:

```
CREATE GENERATOR GEN_EXEMPLO_ID;  
  
SET TERM !! ;  
CREATE TRIGGER EXEMPLO_BI FOR EXEMPLO  
ACTIVE BEFORE INSERT POSITION 0  
AS  
DECLARE VARIABLE tmp DECIMAL(18,0) ;  
BEGIN  
  IF (NEW.CODIGO IS NULL) THEN  
    NEW.CODIGO = GEN_ID(GEN_EXEMPLO_ID, 1);  
  ELSE  
    BEGIN
```

```
tmp = GEN_ID(GEN_EXEMPLO_ID, 0);  
IF (tmp < new.CODIGO) THEN  
    tmp = GEN_ID(GEN_EXEMPLO_ID, new.CODIGO-tmp);  
END  
END!!  
SET TERM ; !!
```

### ➤ Exemplo em MySQL:

```
CREATE TABLE EXEMPLO (  
CODIGO BIGINT NOT NULL AUTO_INCREMENT,  
PRODUTO VARCHAR(100),  
PRECO DOUBLE PRECISION,  
ATIVO BOOLEAN,  
PRIMARY KEY(CODIGO)  
);
```

Observe, neste exemplo em MySQL, que para tornar o campo CODIGO auto-numérico, basta acrescentar o parâmetro AUTO\_INCREMENT.

### ➤ Exemplo em SQLite3:

```
CREATE TABLE EXEMPLO (  
CODIGO INTEGER NOT NULL PRIMARY KEY,  
PRODUTO VARCHAR(100),  
PRECO DOUBLE PRECISION,  
ATIVO BOOLEAN  
);
```

No SQLite3 um campo definido como INTEGER e PRIMARY KEY, é automaticamente um campo auto-numérico.

Segue, agora, um resumo dos comandos principais da linguagem SQL para manipulação de tabelas de dados:

### Comando **SELECT**:

Usado para realizar consultas em uma ou mais tabelas. O resultado da consulta pode retornar registros ou não.

A sintaxe básica pode ser:

```
SELECT campos FROM tabela [WHERE condição] [ORDER BY  
campos [ASC] [DESC]]
```

Exemplo – 1:

```
SELECT * FROM EXEMPLO
```

É o mesmo que:

```
SELECT CODIGO, PRODUTO, PRECO, ATIVO FROM EXEMPLO
```

Um asterisco ( \* ) representa todos os campos da tabela. Os nomes dos campos informados devem ser separados por vírgula.

Exemplo – 2:

```
SELECT PRODUTO, PRECO FROM EXEMPLO ORDER BY PRODUTO  
ASC
```

Retorna todos os registros da tabela EXEMPLO, mas somente os campos PRODUTO e PRECO, ordenado por PRODUTO em ordem ascendente.

Exemplo – 3:

```
SELECT * FROM EXEMPLO WHERE CODIGO > 10
```

Retorna todos os registros que tenham o valor do campo CODIGO maior que 10.

Exemplo – 4:

```
SELECT * FROM EXEMPLO WHERE PRODUTO = 'DVD'
```

Retorna o registro que tem o conteúdo do campo PRODUTO igual a DVD.

Exemplo – 5:

```
SELECT * FROM EXEMPLO WHERE (CODIGO >= 10) AND  
(CODIGO <= 100)
```

Retorna todos os registros que tenham o valor do campo CODIGO maior ou igual 10 e menor ou igual a 100.

Exemplo – 6:

```
SELECT * FROM EXEMPLO WHERE PRODUTO LIKE 'D%'
```

Retorna todos os registros que tenham o valor do campo PRODUTO começando com a letra D.

### Comando INSERT:

Usado para inserir novos registros em uma tabela.

A sintaxe é:

```
INSERT INTO tabela (campos) VALUES (valores)
```

Exemplo:

```
INSERT INTO EXEMPLO (CODIGO, PRODUTO, PRECO, ATIVO)  
  VALUES (1, 'Computador', 1200.00, True)
```

O comando acima insere um novo registro na tabela EXEMPLO.

### Comando UPDATE:

Usado para editar registros em uma tabela.

A sintaxe é:

```
UPDATE tabela SET campo_1=valor_1[,  
[campo_2=valor_2],[campo_n=valor_n]] [WHERE  
condição]
```

Exemplo:

```
UPDATE EXEMPLO SET PRODUTO = 'Impressora', PRECO =  
200.00 WHERE CODIGO = 1
```

O comando acima edita os campos PRODUTO e PRECO do registro de CODIGO igual a 1 da tabela EXEMPLO.

### Comando DELETE:

Usado para excluir registros em uma tabela.

A sintaxe é:

```
DELETE FROM tabela [WHERE condição]
```



Exemplo:

```
DELETE FROM EXEMPLO WHERE CODIGO = 2
```

O comando acima exclui o registro da tabela EXEMPLO cujo valor do campo CODIGO é igual a 2.

### Acessando Banco de Dados com SQLdb

SQLdb é a paleta de componentes padrão do Lazarus para o acesso nativo a um grande número de bancos de dados SQL. É possível a conexão com os seguintes bancos: PostgreSQL, Oracle, Firebird, InterBase, MySql, SQLite3 e qualquer outro que tenha um driver ODBC.

#### Modelo de Acesso a Banco de Dados:

O esquema de acesso e manipulação de um banco de dados por um programa feito no Lazarus, usando a paleta nativa SQLdb, é realizado como mostrado abaixo:

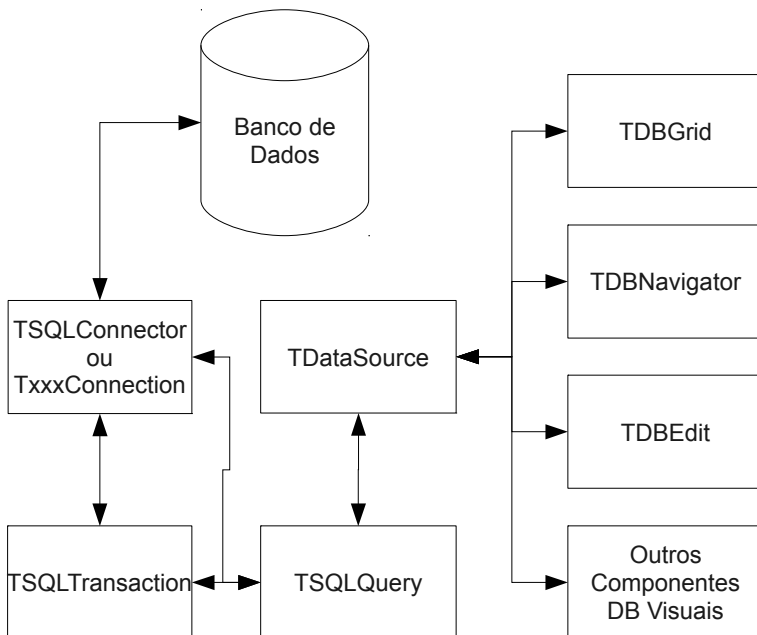














Figura 12.1 – Esquema de Acesso a Dados

### Visão Geral dos Componentes:

 <b>TSQLQuery</b>	Basicamente, responsável por realizar as consultas, inclusões, alterações e exclusões em uma tabela do banco.
 <b>TSQLTransaction</b>	Controla as transações, confirmando ou cancelando o salvamento físico das alterações propostas ao banco de dados.
 <b>TSQLScript</b>	Usado para executar no banco de dados um texto contendo uma sequência de comandos SQL.
 <b>TSQLConnector</b>	Um conector universal. Possui uma propriedade para informar o servidor de banco de dados que será acessado.
 <b>TPQConnection</b>	Estabelece conexão com um servidor de banco de dados PostgreSQL de qualquer versão.
 <b>TOracleConnection</b>	Estabelece conexão com um servidor de banco de dados Oracle de qualquer versão.
 <b>TODBCConnection</b>	Estabelece conexão com qualquer servidor de banco de dados por meio de seu driver ODBC instalado.
 <b>TMySQL40Connection</b>	Estabelece conexão com um servidor de banco de dados MySQL 4.0.

 <b>TMySQL41Connection</b>	Estabelece conexão com um servidor de banco de dados MySQL 4.1.
 <b>TMySQL50Connection</b>	Estabelece conexão com um servidor de banco de dados MySQL 5.0 ou 5.1.
 <b>TSQLite3Connection</b>	Estabelece conexão com um arquivo de banco de dados SQLite Versão 3.
 <b>TIBConnection</b>	Estabelece conexão com um servidor de banco de dados Firebird ou InterBase de qualquer versão.

### O Componente TSQLConnector:

Diferente dos outros conectores da paleta SQLdb, que geralmente só conectam com um respectivo servidor de banco de dados, este componente tem a capacidade de estabelecer conexão com diversos servidores de banco de dados e driver's ODBC.

Isso é possível através da propriedade *ConnectorType*. Esta propriedade aceita os seguintes parâmetros:

```
firebird  
mysql 4.0  
mysql 4.1  
mysql 5.0  
oracle  
odbc  
postgresql
```

### Configurando Conexão a Bancos de Dados:

Para estabelecer a conexão com um banco de dados usando os componentes da paleta SQLdb, precisamos de um conector *TxxxConnection* (*TIBConnection*, *TPQConnection*, etc) ou *TSQLConnector* e um componente *TSQLTransaction*. Na maioria dos conectores, você terá que, basicamente, configurar as seguintes propriedades:

**DatabaseName** – O nome do banco de dados a ser acessado. Se for um arquivo, deve incluir o caminho completo e o nome do arquivo (Ex.: */home/usuario/clientes.fdb*).

**HostName** – O IP da máquina onde está rodando o servidor do banco de dados. Se for a máquina local, o valor pode ser *localhost* ou *127.0.0.1*.

**Password** – A senha de acesso ao servidor do banco.

**Transaction** – Ligar a um componente *TSQLTransaction*.

**UserName** – Nome do usuário que tem acesso ao servidor do banco.

No componente *TSQLTransaction*, mude a propriedade *Action* para *caCommitRetaining*.

Para usar o conector *TSQLite3Connection*, que acessa bancos SQLite Versão 3, só precisamos informar o caminho e o nome do arquivo de banco de dados na propriedade *DatabaseName* (Ex.: */home/usuario/teste.db*).

Se o objetivo for acessar bancos de dados usando o conector *TODBCConnection* (não é o foco deste livro), além das

propriedades já mencionadas, e necessário preencher as propriedades *Driver* e *FileDSN* .

Para ativar a conexão com o banco, basta ativar o *TSQLTransaction* mudando a propriedade *Active* para *True*. A propriedade *Connected* do *TxxxConnection* associado é ativada (*True*) automaticamente.

Seguem-se exemplos de conexão via código (*scConexao* é um *TxxxConnection* qualquer e o *stTransacao* é um *TSQLTransaction*):

```
stTransacao.Active := True;  
// Automaticamente scConexao.Connected  
// muda para True
```

ou

```
scConexao.Connected := True;  
// Ou scConexao.Open;  
stTransacao.Active := True;
```

Desconectando no código:

```
scConexao.Connected := False;  
// Ou scConexao.Close;
```

### **Criando Bancos, Tabelas e Outros via Código:**

Usando um *TxxxConnection* (*TIBConnection*, *TPQConnection*, etc) é possível criar bancos através do método *CreateDB* para os seguintes servidores: Firebird, MySQL e PostgreSQL.

**OBS.:** Usando o conector *TSQLite3Connection*, ao conectar, se o banco SQLite3 não existir é criado automaticamente.

Por meio da função *ExecuteDirect* podemos enviar diretamente ao banco instruções para criar tabelas, índices, etc.

É necessário configurar todas as propriedades do *TxxxConnection* (ou *TSQLConnector*) e do *TSQLTransaction* para o banco a ser criado, conforme já considerado.

Veja como ficaria um exemplo de código para criar um banco, ativar a conexão e em seguida criar uma tabela:

```
// Cria o banco de acordo com a configuração feita
// nas propriedades do scConexao.
scConexao.CreateDB;
// Ativa a transação e consequentemente a conexão
stTransacao.Active := true;
// Executa instrução para criar uma tabela
scConexao.ExecuteDirect(
    'CREATE TABLE PRODUTOS(' +
    'CODIGO INTEGER NOT NULL,' +
    'PRODUTO VARCHAR(100),' +
    'PRECO DOUBLE PRECISION,' +
    'PRIMARY KEY(CODIGO)) '
);
// Confirma a instrução salvando no banco
stTransacao.CommitRetaining;
```

### O Componente TSQLQuery:

Componente usado para consulta e alteração de tabelas do banco de dados, chamar Stored Procedures e Funções SQL. Para seu funcionamento, é necessário ligar a propriedade *Database* deste há um componente *TxxxConnection* e a propriedade *Transaction* ao *TSQLTransaction* associado ao *TxxxConnection*. Por fim, configuramos a propriedade chamada *SQL*, que possui um botão de reticências ( ... ) que ao ser clicado abre um diálogo onde deve ser digitada a consulta (Ex.: **SELECT \* FROM NOME\_DA\_TABELA**).

Os comandos SQL para inserção, alteração e exclusão são criados internamente de modo automático. Mas se desejar personalizá-los, pode usar as propriedades *InsertSQL*, *UpdateSQL* e *DeleteSQL*.

Principais Propriedades do TSQLQuery:

Propriedade	Descrição
Active	Abre ou Fecha o acesso a tabela
BOF	Retorna <i>True</i> se estiver no início da tabela
DeleteSQL	Comando em SQL para excluir registros
EOF	Retorna <i>True</i> se estiver no fim da tabela
Filter	Expressão para filtrar registros na memória
Fitered	Ativa/Desative <i>Filter</i>
InsertSQL	Comando em SQL para inserir registros
ServerFilter	Expressão p/ trazer filtrados os registros do banco
ServerFiltered	Ativa/Desativa <i>ServerFilter</i>
SQL	Comando de consulta a dados em SQL
State	Retorna o estado da tabela
UpdateSQL	Comando em SQL para editar registros

Principais Métodos do TSQLQuery:

Metodo	Descrição
ApplyUpdates	Aplica atualizações pendentes na memória
Append	Coloca a tabela em modo de inserção
AppendRecord	Insere um registro na tabela com os valores



<b>Metodo</b>	<b>Descrição</b>
Cancel	Cancela edição ou inserção de registro
CancelUpdates	Cancela atualizações pendentes na memória
Close	Fecha o acesso a tabela
Delete	Exclui o registro atual
Edit	Coloca a tabela em modo edição p/ registro atual
ExecSQL	Usado quando a consulta não retorna dados
FieldByName	Acessa/Edita campo por nome p/ registro atual
First	Move para o primeiro registro da tabela
Insert	Similar ao <i>Append</i>
InsertRecord	Similar ao <i>AppendRecord</i>
IsEmpty	Retorna <i>True</i> se a consulta não retornou registros
Last	Move para o último registro
Locate	Localiza um registro
Next	Move para o próximo registro
Open	Abre o acesso a tabela
Post	Confirma Edição/Inserção de registro
Prior	Move para o registro anterior
Refresh	Atualiza a consulta aos dados do banco de dados

Todos os Eventos do TSQLQuery:

<b>Evento</b>	<b>Descrição</b>
AfterCancel	Ocorre após o método Cancel
AfterClose	Ocorre após o fechamento da tabela

<b>Evento</b>	<b>Descrição</b>
AfterDelete	Ocorre após o método Delete
AfterEdit	Ocorre após o método Edit
AfterInsert	Ocorre após o método Insert
AfterOpen	Ocorre após o método Open
AfterPost	Ocorre após o método Post
AfterScroll	Ocorre após trocar de registro
BeforeCancel	Ocorre antes da execução do método Cancel
BeforeClose	Ocorre antes do fechamento da tabela
BeforeDelete	Ocorre antes da execução do método Delete
BeforeEdit	Ocorre antes da execução do método Edit
BeforeInsert	Ocorre antes da execução do método Insert
BeforeOpen	Ocorre antes da execução do método Open
BeforePost	Ocorre antes da execução do método Post
BeforeScroll	Ocorre antes de trocar de registro
OnCalcFields	Usado p/ calcular valores dos campos calculados
OnDeleteError	Acionado quando há erro ao excluir registro
OnEditError	Acionado quando há erro ao editar registro
OnFilterRecord	Usado para realizar uma filtragem variável
OnNewRecord	Acionado no instante da inserção de um registro
OnPostError	Acionado quando ocorre um erro ao salvar registro
OnUpdateError	Acionado ao ocorrer um erro de atualização

No TSQLQuery, quando a propriedade *SQL* contiver uma consulta que retorna registros, use o método *Open* para disponibilizar os dados e quando a propriedade *SQL* contiver

um comando SQL de ação como DELETE, UPDATE, INSERT e etc, use o método *ExecSQL* para confirmar a execução dos comandos.

Para salvar fisicamente no banco de dados as alterações feitas numa tabela acessada pelo TSQLQuery, é necessário incluir no evento *AfterPost* e no evento *AfterDelete* do TSQLQuery o código exemplificado abaixo:

```
// Supondo que temos um objeto sQuery do
// tipo TSQLQuery e um objeto sTrans
// do tipo TSQLTransaction.
try
    sQuery.ApplyUpdates;
    sTrans.CommitRetaining;
except
    sQuery.CancelUpdates;
    sTrans.RollbackRetaining;
    sQuery.Refresh;
    ShowMessage (
        'Não foi possível salvar as alterações!');
end;
```

Após a chamada do método *ExecSQL* também é necessário a chamada do método *CommitRetaining* do *TSQLTransaction* para salvar no banco as alterações.

### Personalizando o TSQLQuery:

Como já mencionado, embora o TSQLQuery crie internamente os comandos de inserção, edição e exclusão para uma consulta SELECT digitada na propriedade SQL, podemos, se necessário, especificar manualmente estes comandos.

Para exemplificar, tomaremos por base uma tabela de nome ITENS\_VENDA com os seguintes campos: CODIGO,

PRODUTO, QUANT e PRECO . Na propriedade SQL de um TSQLQuery digitamos a seguinte instrução SELECT :

```
SELECT CODIGO, PRODUTO, QUANT, PRECO, (QUANT *  
PRECO) AS TOTAL FROM ITENS_VENDA
```

Note que o campo TOTAL é um campo calculado, não existe fisicamente, portanto não é atualizável pelo usuário. Precisamos então configurar as propriedades DeleteSQL, InsertSQL e UpateSQL do TSQLQuery da seguinte forma:

DeleteSQL:

```
DELETE FROM ITENS_VENDA WHERE CODIGO = :OLD_CODIGO
```

InsertSQL:

```
INSERT INTO ITENS_VENDA (CODIGO, PRODUTO, QUANT,  
PRECO) VALUES (:CODIGO, :PRODUTO, :QUANT, :PRECO)
```

UpdateSQL:

```
UPDATE ITENS_VENDA SET CODIGO = :CODIGO, PRODUTO  
= :PRODUTO, QUANT = :QUANT, PRECO = :PRECO WHERE  
CODIGO = :OLD_CODIGO
```

O prefixo OLD\_ acessa o valor do campo antes de ser modificado.

### **Executando Várias Instruções SQL com TSQLQuery:**

As propriedades SQL, *DeleteSQL*, *InsertSQL* e *UpdataSQL* do TSQLQuery aceitam múltiplas instruções de ação em SQL. Pode-se usar parâmetros nas instruções e estas devem ser separadas por ponto e vírgula.

Exemplo:

```
SqlQuery.SQL.Clear;  
SqlQuery.SQL.Add(  
    'UPDATE TABELA1 SET CAMPO1 = :V1;');  
SqlQuery.SQL.Add(  
    'INSERT INTO TB2 (CP1) VALUES (:V1);');  
SqlQuery.SQL.Add(  
    'INSERT INTO TB3 (CP2) VALUES (:V2);');  
SqlQuery.SQL.Add('DELETE FROM TABELA5;');  
SqlQuery.SQL.Add('UPDATE TABELA4 SET CP22 = :V2;');  
SqlQuery.Params.ParamByName('V1').Value := 'abc';  
SqlQuery.Params.ParamByName('V2').Value := 999;  
SqlQuery.ExecSQL;  
SqlTransaction.CommitRetaining;
```

Os comandos serão executados na ordem em que foram adicionados na propriedade *SQL* e salvos em uma única transação.

### Filtrando Registros com TSQLQuery.Filter:

Usando o TSQLQuery, podemos filtrar os registros trazidos do banco de dados para memória, colocando a propriedade *Filtered* para *True* e definindo em *Filter* a condição de filtragem. Isso pode ser feito no Inspetor de Objetos ou no código. Veja alguns exemplos de como fazer no código (sqTabela é um TSQLQuery):

Exemplo – 1:

```
// Ativando o filtro  
sqTabela.Filtered := True;
```

Exemplo – 2:

```
sqTabela.Filter := 'CODIGO > 10';
```

O código acima torna disponível apenas os registros cujo campo CODIGO tenham o valor acima de 10.

Exemplo – 3:

```
sqTabela.Filter := 'PRODUTO = ' + QuotedStr('D*');
```

Na filtragem acima só estarão disponíveis os registros cujo conteúdo do campo PRODUTO comece com a letra D .

### Filtrando Registros com TSQLQuery.ServerFilter:

Enquanto a propriedade *Filter* filtra os registros carregados na memória, a propriedade *ServerFilter* filtra no servidor do banco de dados. Assim, são carregados na memória só os registros resultantes da filtragem.

Para ativar o uso deste filtro, configure a propriedade *ServerFiltered* para *True*. É necessário usar a sintaxe SQL do servidor. Também, é preciso fechar a tabela, configurar o filtro e abrir a tabela.

Exemplo – 1:

```
sqTabela.Close; // Fecha o acesso à tabela
// Ativando o filtro
sqTabela.ServerFiltered := True;
```

Exemplo – 2:

```
sqTabela.Close;
sqTabela.ServerFilter := 'CODIGO > 10';
sqTabela.Open;
```

Exemplo – 3:

```
sqTabela.Close;  
sqTabela.ServerFilter := 'PRODUTO LIKE ' +  
    QuotedStr('D%');  
sqTabela.Open;
```

Observe que foi usada a sintaxe SQL.

### Filtrando Registros com TSQLQuery.SQL:

Podemos filtrar registros no componente TSQLQuery usando diretamente SQL. Nesta filtragem podemos usar ou não parâmetros configuráveis.

Usando o *Inspetor de Objetos*, digite a consulta com filtro na propriedade SQL do TSQLQuery ou faça isso no código:

Exemplo – 1:

```
sqTabela.Close;  
sqTabela.SQL.Text := 'SELECT * FROM EXEMPLO ' +  
    'WHERE (CODIGO > 10) AND (ATIVO = TRUE)';  
sqTabela.Open;
```

Exemplo – 2:

```
sqTabela.Close;  
sqTabela.SQL.Text := 'SELECT * FROM EXEMPLO ' +  
    'WHERE (CODIGO > :vCod) AND (ATIVO = :vAtivo)';  
sqTabela.Params.ParamByName('vCod').AsInteger :=  
    10;  
sqTabela.Params.ParamByName('vAtivo').AsBoolean :=  
    True;  
sqTabela.Open;
```

Acima foi usado parâmetros.

### Localizando Registros com TSQLQuery.Locate:

Podemos localizar um registro usando a função *Locate* presente no TSQLQuery (ou em qualquer descendente da classe TDataSet). A sintaxe desta função é:

```
Locate ({campo}, {valor}, [{parâmetros}]):boolean;
```

ou

```
Locate ({campo_1;campo_2;campo_n},  
VarArrayOf ([{valor_1,valor_2,valor_n}]),  
[{parâmetros}]):boolean;
```

Os parâmetros aceitos são: *loPartialKey* (similar ao LIKE do SQL) e *loCaseInsensitive* (pesquisa ignora maiúsculas e minúsculas). Para usar *VarArrayOf()*, incluir na seção *uses* da unit a unit *variants*. *Locate* retorna *True* se encontrar registro.

Exemplos:

```
sqTabela.Locate ('CODIGO', '10', []);
```

Localiza o registro de CODIGO igual a 10.

```
sqTabela.Locate ('PRODUTO', 'D',  
[loPartialKey, loCaseInsensitive]);
```

Localiza o primeiro registro cujo campo PRODUTO contenha a letra D, maiúscula ou minúscula.

Podemos também fazer pesquisas por mais de um campo. Veja um exemplo:

```
sqTabela.Locate ('PRODUTO; CODIGO',  
  VarArrayOf (['DVD', '10']), []);
```



**DICA:** Em todos os exemplos, os valores a serem pesquisados poderiam ser extraídos da propriedade *Text* de um *TEdit*.

### Formulário DataModule:

O DataModule é velho conhecido dos programadores Delphi e, também, está presente no Lazarus.

Um DataModule é um formulário, que consome menos recursos do sistema que um formulário comum, usado para concentrar alguns componentes não visuais da aplicação (aqueles que aparecem na forma de ícone), especialmente os componentes de acesso a dados. Isso ajuda na organização da aplicação e facilita a manutenção. Além de evitar a repetição desnecessária de componentes e código.

Para incluir um DataModule no projeto, clique no menu “*Arquivo > Novo ...*” e, no diálogo seguinte, selecione DataModule e clique em OK.

Para que um determinado formulário (form1) tenha acesso aos componentes de outro formulário (form2) (como o DataModule, por exemplo), é necessário acrescentar na cláusula **uses** do formulário (form1) o nome da unidade de código (Unit) do formulário a ser acessado (form2).

### TDatasource:

Possui uma propriedade *DataSet* que deve ser ligada a um componente descendente de um *TDataSet* (Ex.: *TSQLQuery*). Tem a função de ligar os dados de uma tabela do banco de dados aos componentes visuais que irão exibir e/ou manipular estes dados (por exemplo: *TDBGrid*, *TDBEdit*, *TDBNavigator*, etc).

Todos os componentes DB têm uma propriedade *DataSource* que deve ser ligada a um *TDataSource*. E, com exceção do *TDBGrid* e do *TDBNavigator*, também, possuem uma propriedade *DataField* onde informamos o campo da tabela que o componente vai manipular.

### **TDBNavigator:**

Este componente fornece um conjunto de botões para a execução das funções de manipulação de uma tabela de banco de dados.

### **TDBGrid:**

Este componente exibe os registros de uma tabela de banco de dados na forma de uma grade, similar a uma planilha do Calc do OpenOffice ou do Excel do MSOffice.

A propriedade *Columns* abre um diálogo para criar as colunas da grade. Cada coluna possui uma propriedade *FieldName* onde informamos o nome do campo a ser visualizado os dados.

### **TDBEdit:**

Ideal para campos de uma linha de texto, numéricos, datas e horas. Possui uma propriedade chamada *EditMask*, usada para formatar a digitação dos dados. Veja abaixo uma tabela de formatações para as situações mais comuns:

<b>Tipo de Informação</b>	<b>Mascara de Formatação</b>
DATA	99/99/9999;1;_
HORA	99:99;1;_

<b>Tipo de Informação</b>	<b>Mascara de Formatação</b>
FONE	(99)9999-9999;1;_
CEP	99.999-999;1;_
CPF	999.999.999-99;1;_
CNPJ	99.999.999/9999-99;1;_

**TDBMemo:**

Ideal para textos grandes.

**TDBComboBox | TDBListBox | TDBRadioGroup |  
TDBGroupBox:**

Ideal para campos com uma lista de opções definidas.

**TDBCheckBox:**

Ideal para campos lógicos (tipo Boolean).

**TDBLookupComboBox | TDBLookupListBox:**

Usados para manipular campos, cujo os valores devem ser escolhidos a partir dos dados extraídos de uma tabela auxiliar.

**TDBCalendar:**

Ideal para manipular campos de datas.

**TDBImage:**

Usado para manipular campos que armazenam imagens.

### Criando um Controle de Cheques com SQLite3 e SQLdb:

A melhor forma de fixarmos a maioria dos conceitos vistos até aqui é praticando. Desenvolveremos, então, um simples controle de cheques que ficará como mostrado na figura seguinte:

**Controle de Cheques**

Código: 00001    Data Cadastro: 09/05/2011    Conta: 12345-6    Agência: 987-6    Banco: EXEMPLO

Data Emissão: 09/05/2011    Número Cheque: 123    Descrição: COMPRA DE EXEMPLO    Valor: R\$ 1.234,56

Favorecido: PESSOA    Status: Compensado    Data Compensado: 10/09/2011

Filtrar por Status: Todos    Favorecido:    Filtar

Código	Data Cadastro	Conta	Agência	Banco
00001	09/05/2011	12345-6	987-6	EXEMPLO

Figura 12.2 – Tela do Controle de Cheques

Para começar, crie uma pasta na pasta *livro\_projetos* com o nome *ctrl\_cheques*. No Lazarus, crie um novo projeto do tipo Aplicação. Salve na pasta *ctrl\_cheques* o projeto com o nome *ctrl\_cheques.lpi* e a unit com o nome *uprincipal.pas*.

Mude o nome do formulário para *fPrincipal* e o Caption para “Controle de Cheques”. Também, mude as seguintes propriedades:

**Position** → *poDesktopCenter*

**Height** → 500

**Width** → 750

**Font > Size** → 9

Adicione ao projeto um formulário DataModule. Clique no menu *Arquivo / Novo ...* , selecione *Data Module* e clique em OK.

Pressione F12 até aparecer o Data Module. Provavelmente, ele estará bem pequeno. Redimensione-o e mude a propriedade Name para *dm* . Pressione CTRL + SHIFT + S e salve a *unit* com o nome *udados.pas* .

Em seguida, no *dm*, insira da paleta SQLdb os seguintes componentes: TSQLite3Connection, TSQLTransaction, TSQLQuery e um TDataSource da paleta Data Access..

Altere a propriedade Name do TSQLite3Connection para *scConexao* e na propriedade DatabaseName digite o caminho e o nome do banco (No Linux: */home/SEU\_NOME\_DE\_USUÁRIO/livro\_projetos/ctrl\_cheques/ctrl\_cheques.db* e no Windows: *C:\livro\_projetos\ctrl\_cheques\ctrl\_cheques.db*).

Segue as configurações para os outros componentes:

TSQLTransaction	
Propriedade	Valor
Action	caCommitRetaining
Database	scConexao
Name	stTransacao

<b>TSQLQuery</b>	
<b><i>Propriedade</i></b>	<b><i>Valor</i></b>
Database	scConexao
Filtered	True
Name	sqCheques
SQL	SELECT * FROM CHEQUES

<b>TDataSource</b>	
<b><i>Propriedade</i></b>	<b><i>Valor</i></b>
DataSet	sqCheques
Name	dsCheques

No Editor de Código clique na guia da unit *uprincipal*.

Para podermos acessar os componentes do Data Module (dm) no formulário fPrincipal, precisamos informar a unit do *dm* (udados) numa seção *uses*.

Assim, digite o código em negrito na posição indicada:

```
var
    fPrincipal: TfPrincipal;

implementation

uses
    udados;

{$R *.lfm}

{ TfPrincipal }
```

Pressione novamente F12 para exibir o formulário e insira neste um TDBGrid da paleta “Data Controls”. E altere as seguintes propriedades:

**Align** → *alBottom*  
**AlternateColor** → *clSkyBlue*  
**Anchor** > **akTop** → *True*  
**DataSource** → *dm.dsCheques*  
**Height** → *277*  
**Name** → *gdDados*  
**Options** > **dgRowSelect** → *True*  
**ReadOnly** → *True*

Em seguida coloque no formulário um TPanel e altere as propriedades:

**Align** → *alBottom*  
**BevelInner** → *bvLowered*  
**Caption** → *{deixe vazio}*

No Panel1 (TPanel) insira, na ordem, um TDBNavigator da paleta “Data Controls”, dois TLabel, um TComboBox, um TEdit e um TButton.

Selecione o *DBNavigator1*, mude as propriedades Name para *dbNavegador*, Height para *37*, ShowHints para *True* e na propriedade Hints clique no botão de reticências ( ... ) e, no diálogo que surge, digite o texto abaixo:

**Primeiro Registro**  
**Registro Anterior**  
**Próximo Registro**  
**Último Registro**  
**Novo Registro**  
**Excluir Registro**

**Editar Registro**  
**Salvar Alterações**  
**Cancelar Alterações**  
**Atualizar Dados**

Clique em OK para confirmar.

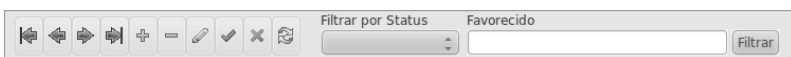
Selecione o componente ComboBox1 e altere a propriedade Name para *cbFiltrar* e mude a propriedade Style para *csDropDownList*. Para melhorar o visual no Linux, você pode mudar também as propriedades AutoSize para *False*, Height para 25 e Font > Size para 8. Agora, clique ao lado da propriedade Items e clique no botão de reticências ( ... ), no diálogo que surgir digite:

**a Compensar**  
**Compensado**  
**Cancelado**  
**Voltou**  
**Todos**

Clique em OK para confirmar.

Altere o Caption de cada TLabel para *Filtrar por Status* e *Favorecido*, respectivamente.

Modifique no TEdit (Edit1) as propriedades Name para *edFiltrar* , Anchors > akRight para *True* e Width para 283 . E no TButton (Button1) mude as propriedades Name para *btFiltrar* , Anchors > akLeft para *False* , Anchors > akRight para *True* , Caption para *Filtrar* e Width para 46. Organize os componentes como mostrado abaixo:



*Figura 12.3 – Painel de Navegação e Pesquisa*



Clique no fundo do formulário e, usando a guia Eventos do Inspetor de Objetos, crie o evento OnShow para o formulário fPrincipal e digite o código abaixo entre o **begin** e o **end** da procedure:

```
// Configurações regionais.
DateSeparator := '/';
ShortDateFormat := 'dd/mm/yyyy';
ThousandSeparator := '.';
DecimalSeparator := ',';
CurrencyFormat := 2;
CurrencyString := 'R$';
try // Início do bloco protegido.
    // Informamos o caminho e o nome do banco.
    // Se não existir é criado na pasta da aplicação.
    dm.scConexao.DatabaseName :=
        ExtractFilePath(ParamStr(0))+'ctrl_cheques.db';
    // Ativamos a transação e
    // consequentemente a conexão.
    dm.stTransacao.Active := True;
    // Sinaliza um comando SQL para criar a tabela
    // cheques, caso ela não exista.
    dm.scConexao.ExecuteDirect(
        'CREATE TABLE IF NOT EXISTS CHEQUES (' +
        'CODIGO INTEGER NOT NULL PRIMARY KEY,' +
        'DATACAD DATE,' +
        'CONTA VARCHAR(20),' +
        'AGENCIA VARCHAR(20),' +
        'BANCO VARCHAR(50),' +
        'DATAEMIS DATE,' +
        'NUMCHEQUE VARCHAR(20),' +
        'DESCRICAO VARCHAR(200),' +
        'FAVORECIDO VARCHAR(100),' +
        'VALOR DOUBLE PRECISION,' +
        'STATUS VARCHAR(12),' +
        'DATACOMP DATE);');
    // Executa o comando SQL no banco.
    dm.stTransacao.CommitRetaining;
    // Abre a consulta.
```

```
dm.sqCheques.Open;
except // Comandos para tratar erros.
// O objeto "e" conterá as informações do erro.
on e: Exception do begin
// Cancela qualquer instrução SQL pendente.
dm.stTransacao.RollbackRetaining;
// Mostra para o usuário o que ocorreu.
ShowMessage(
  'Não foi possível ativar o banco, ' +
  'o aplicativo será fechado!' + #13 + #13 +
  'CLASSE DO ERRO: ' + e.ClassName + #13 +
  'MENSAGEM: ' + e.Message);
// Esconde o formulário.
fPrincipal.Hide;
// Processa as mensagens pendentes da
// aplicação.
Application.ProcessMessages;
// Termina a aplicação.
Application.Terminate;
end;
end;
// Coloca o cbFiltra na posição Todos.
cbFiltra.ItemIndex := 4;
// Desativa a confirmação de exclusão de
// registro do TDBNavigator.
dbNavegador.ConfirmDelete := False;
```

Agora, execute o programa para que o banco de dados seja criado e em seguida encerre a execução. Agora que o banco foi criado podemos acessá-lo em tempo de projeto.

No Editor de Código, clique na guia da unit *udados* e acrescente na seção *uses* a unit *Dialogs* como mostrado a seguir:

```
unit udados;

{$mode objfpc}{$H+}
```

```
interface

uses
    Classes, SysUtils, sqlldb, db, sqlite3conn,
    FileUtil, Dialogs;

type
```

Esta unit é necessária para podermos usar no código do Data Module a função *ShowMessage* afim de mostrar mensagens de erro personalizadas para o usuário.

Pressione F12 para retornar ao formulário Data Module e selecione o componente *scConexao* e mude a propriedade *Conected* para *True*. Depois, selecione o componente *sqCheques* e mude a propriedade *Active* para *True*.

Dê um duplo clique no ícone do componente *sqCheques*. No diálogo que surgir, clique no botão com o sinal de mais ( + ) e no diálogo seguinte selecione todos os campos e clique no botão *Create*. O diálogo anterior ficará como mostrado abaixo:

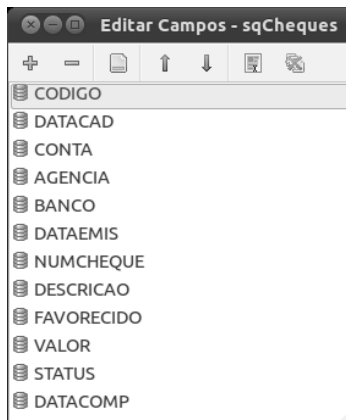


Figura 12.4 – Dialogo Editar Campos, preenchido

O que acabamos de fazer foi criar objetos do tipo `TField` para manipular cada campo da tabela `cheques`.

Então, continuando no diálogo acima, selecione o campo `CODIGO` e, no Inspetor de Objetos mude as seguintes propriedades:

**Alignment** → *taCenter*  
**DisplayFormat** → *00000*  
**ProviderFlags > pflnKey** → *True*  
**ProviderFlags > pflnUpdate** → *False*  
**Required** → *False*

As três últimas propriedades acima configuradas, informam ao `TSQLQuery` que o campo `CODIGO` é a chave primária da tabela, não precisa ser atualizado pelo usuário porque é auto-numérico e, conseqüentemente, não é obrigatório o seu preenchimento pelo usuário.

Agora, selecione `DATA CAD`, `CONTA`, `AGENCIA`, `DATA EMIS`, `NUMCHEQUE` e `DATA COMP`, e altere **Alignment** para *taCenter*. Depois, selecione `VALOR` e mude a propriedade **Currency** para *True* (mostra os números no formato de moeda).

Estando o campo `VALOR` ainda selecionado, no Inspetor de Objetos clique na guia **Eventos** e crie o evento *OnSetText* e acrescente o código em negrito mostrado abaixo:

```
procedure Tdm.sqChequesVALORSetText(Sender: TField;  
const aText: string);  
begin  
    try  
        if Trim(aText) = '' then  
            Sender.Value := null  
        else  
            Sender.Value := StrToFloat(aText);
```

```
except
    ShowMessage('Valor Inválido!');
    Exit;
end;
end;
```

Pressione F12, novamente dê um duplo clique no componente `sqCheques`, no diálogo `Editar Campos` selecione o campo `DATAEMIS`, no `Inspetor de Objetos` clique na guia `Eventos` e crie o evento *`OnSetText`* e acrescente o código em negrito mostrado abaixo:

```
procedure Tdm.sqChequesDATAEMISSetText(Sender:
TField; const aText: string);
begin
    try
        if aText = ' / / ' then
            Sender.Value := null
        else
            Sender.Value := StrToDate(aText);
    except
        ShowMessage('Data Inválida!');
        Exit;
    end;
end;
```

Para não repetir código, ligue o evento *`OnSetText`* do campo `DATAACAD` ao evento *`OnSetText`* do campo `DATAEMIS`.

Selecione o componente `sqCheques`, crie o evento *`OnNewRecord`* deste e digite o código em negrito na posição indicada abaixo:

```
procedure Tdm.sqChequesNewRecord(DataSet:
TDataSet);
begin
    sqCheques.FieldName('DATAACAD').AsDateTime :=
        Date;
```

```
sqCheques.FieldName('STATUS').AsString :=  
  'a Compensar';  
end;
```

Também, crie o evento *AfterPost* e digite o código em negrito na posição indicada abaixo:

```
procedure Tdm.sqChequesAfterPost(DataSet:  
TDataSet);  
begin  
  try  
    // Salva os dados no banco.  
    sqCheques.ApplyUpdates;  
    stTransacao.CommitRetaining;  
    sqCheques.Refresh;  
  except // Comandos para tratar erros.  
    // O objeto "e" conterá as informações do erro.  
    on e: Exception do  
      begin  
        sqCheques.CancelUpdates;  
        stTransacao.RollbackRetaining;  
        sqCheques.Refresh;  
        ShowMessage('Não foi possível alterar ' +  
          'o banco de dados!' + #13 + #13 +  
          'CLASSE DO ERRO: ' + e.ClassName + #13 +  
          'MENSAGEM: ' + e.Message);  
      end;  
    end;  
  end;  
end;
```

Agora, ligue o evento *AfterDelete* ao evento *AfterPost*.

No Data Module, selecione o componente *sqCheques* e mude a propriedade *Active* para *False*. Depois, selecione o componente *scConexao* e mude a propriedade *Conected* para *False*.

Retorne para o formulário fPrincipal. Você pode fazer isso de várias maneiras: 1) Clicando na guia com a unit do fPrincipal (uprincipal) e depois pressionar F12; 2) Clicando no botão Exibir Formulários na barra de ferramentas rápidas, selecionar fPrincipal e clicar em OK; 3) Pressionando SHIFT + F12, selecionar fPrincipal e clicar em OK.

Use estes métodos para navegar por qualquer formulário do projeto.

Agora, insira no fPrincipal doze TLabel, onze TDBEdit e um TDBComboBox.

Organize os componentes e mude a propriedade Caption dos TLabel's para ficarem como mostrado na figura seguinte:

A imagem mostra um formulário de controle de cheques com os seguintes campos:

Código	Data Cadastro	Conta	Agência	Banco
<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>
Data Emissão	Número Cheque	Descrição	Valor	
<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	
Favorecido	Status	Data Compensado		
<input type="text"/>	<input type="text"/>	<input type="text"/>		

*Figura 12.5 – Campos do Controle de Cheques*

Mude a propriedade DataSource de todos os TDBEdit's e TDBComboBox para *dsCheques*.

**DICA:** Para selecionar mais de um componente, afim de mudar propriedades em comum, clique no primeiro componente, segure a tecla SHIFT e clique nos componentes seguintes.

Em seguida, da esquerda para direita e de cima para baixo, mude a propriedade Name dos TDBEdit's para: *edCodigo*, *edDataCad*, *edConta*, *edAgencia*, *edBanco*, *edDataEmis*, *edNumCheque*, *edDescricao*, *edValor*, *edFavorecido* e

*edDataComp*. Altere a propriedade Name do TDBComboBox para *cbStatus* .

Modifique a propriedade DataField dos componentes *edCodigo*, *edDataCad*, *edConta*, *edAgencia*, *edBanco*, *edDataEmis*, *edNumCheque*, *edDescricao*, *edValor*, *edFavorecido*, *cbStatus*, *edDataComp* para *CODIGO*, *DATACAD*, *CONTA*, *AGENCIA*, *BANCO*, *DATAEMIS*, *NUMCHEQUE*, *DESCRICAO*, *VALOR*, *FAVORECIDO*, *STATUS* e *DATACOMP* , respectivamente.

Nos TDBEdit's *edCodigo* e *edDataCad*, também, mude as seguintes propriedades: Color para *clSilver*, ReadOnly para *True* e TabStop para *False*.

Selecione o componente *cbStatus* (TDBComboBox) e altere a propriedade Style para *csDropDownList*. Para melhorar o visual no Linux, você pode mudar também as propriedades AutoSize para *False*, Height para 25 e Font > Size para 8. Em seguida, clique ao lado da propriedade Items e clique no botão de reticências ( ... ), no diálogo que surgir digite:

**a Compensar**  
**Compensado**  
**Cancelado**  
**Voltou**

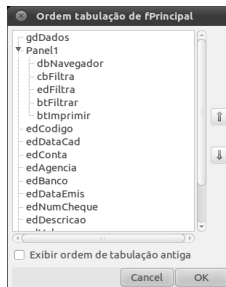
Clique em OK para confirmar.

Quando o programa estiver em execução e o usuário pressionar a tecla TAB, o foco do cursor é movido de um componente ao outro. Porém, este pode não ficar na ordem que desejamos. Por exemplo, estando o cursor no *edConta* e pressionarmos TAB o cursor deverá ficar posicionado no



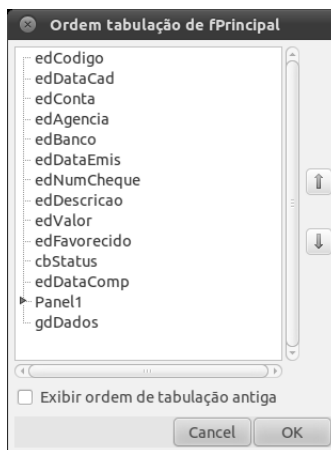
*edAgencia* e assim por diante. Portanto, como no Delphi, o Lazarus tem um recurso para editar a ordem do foco.

Para isso, basta clicar com o botão direito do mouse em qualquer parte dentro do formulário ou em um componente. No menu pop-up escolha a opção “Ordem de Tabulação ...” e vai surgir um diálogo similar ao mostrado abaixo:



*Figura 12.6 – Ordem de Tabulação (Foco)*

Usando as setas verdes, reposicione os componentes para que fiquem na seguinte ordem de foco e clique em OK:



*Figura 12.7 – Nova Ordem do Foco*

Agora, dê um duplo clique no gdDados (TDBGrid) e no diálogo que surge, adicione doze colunas na grade usando o botão *Adicionar*. No Inspetor de Objetos, mude as seguintes propriedades de cada coluna criada (não esqueça de selecionar no diálogo a coluna a ser alterada):

Coluna	Propriedade	Valor
0	Alignment	taCenter
	DisplayFormat	00000
	FieldName	CODIGO
	Title > Alignment	taCenter
	Title > Caption	Código
	Width	64
1	Alignment	taCenter
	FieldName	DATA CAD
	Title > Alignment	taCenter
	Title > Caption	Data Cadastro
	Width	100
2	Alignment	taCenter
	FieldName	CONTA
	Title > Alignment	taCenter
	Title > Caption	Conta
	Width	120

<b>Coluna</b>	<b>Propriedade</b>	<b>Valor</b>
3	Alignment	taCenter
	FieldName	AGENCIA
	Title > Alignment	taCenter
	Title > Caption	Agência
	Width	120
4	Alignment	taCenter
	FieldName	BANCO
	Title > Alignment	taCenter
	Title > Caption	Banco
	Width	300
5	Alignment	taCenter
	FieldName	DATAEMIS
	Title > Alignment	taCenter
	Title > Caption	Data Emissão
	Width	100
6	Alignment	taCenter
	FieldName	NUMCHEQUE
	Title > Alignment	taCenter
	Title > Caption	Núm. Cheque
	Width	100

<b>Coluna</b>	<b>Propriedade</b>	<b>Valor</b>
7	FieldName	DESCRICA0
	Title > Alignment	taCenter
	Title > Caption	Descrição
	Width	400
8	FieldName	FAVORECIDO
	Title > Alignment	taCenter
	Title > Caption	Favorecido
	Width	400
9	Alignment	taRightJustify
	FieldName	VALOR
	Title > Alignment	taCenter
	Title > Caption	Valor
	Width	130
10	Alignment	taCenter
	FieldName	STATUS
	Title > Alignment	taCenter
	Title > Caption	Status
	Width	100
11	Alignment	taCenter
	FieldName	DATAComp
	Title > Alignment	taCenter
	Title > Caption	Data Compensado
	Width	130

Terminamos aqui a parte visual do projeto. Vamos então para o restante da codificação.

Crie o evento *OnSelect* para o componente *cbFiltro* (TComboBox) que está no *Panel1* e digite o código abaixo em negrito:

```
procedure TfPrincipal.cbFiltroSelect(Sender:
TObject);
begin
    // Limpa o texto digitado em edFiltro.
    edFiltro.Text := '';
    // Cria o filtro de acordo com a opção
    // selecionada em cbFiltro.
    case cbFiltro.ItemIndex of
        0: dm.sqCheques.Filter :=
            'STATUS = ' + QuotedStr('a Compensar');;
        1: dm.sqCheques.Filter :=
            'STATUS = ' + QuotedStr('Compensado');;
        2: dm.sqCheques.Filter :=
            'STATUS = ' + QuotedStr('Cancelado');;
        3: dm.sqCheques.Filter :=
            'STATUS = ' + QuotedStr('Voltou');;
        4: dm.sqCheques.Filter := ''; // Todos.
    end;
end;
```

Crie o evento *OnClick* para o componente *btFiltrar* (Tbutton) que está no *Panel1* e digite o código abaixo em negrito:

```
procedure TfPrincipal.btFiltrarClick(Sender:
TObject);
begin
    // Deixa cbFiltro sem opção selecionada.
    cbFiltro.ItemIndex := -1;
    // Se o texto de edFiltro for vazio, limpa o
    // filtro. Se não, cria o filtro com o texto de
    // edFiltro.
```

```
if Trim(edFiltrar.Text) = '' then
  dm.sqCheques.Filter := '' else
  dm.sqCheques.Filter :=
    'FAVORECIDO = ' +
    QuotedStr(edFiltrar.Text + '*');
end;
```

Crie o evento *BeforeAction* para o componente *dbNavegador* (TDBNavigator) que está no *Panel1* e digite o código abaixo em **negrito**:

```
procedure
TfPrincipal.dbNavegadorBeforeAction(Sender:
TObject; Button: TDBNavButtonType);
begin
  // Se o botão de inserir registro for
  // pressionado, posiciona o cursor no componente
  // edConta antes de executar a ação do botão.
  if Button = nbInsert then edConta.SetFocus;
  // Se o botão de salvar o registro for
  // pressionado, verifica se foi preenchida
  // a Descrição antes de executar a ação do botão.
  if Button = nbPost then begin
    dbNavegador.SetFocus;
    if trim(edDescricao.Text) = '' then begin
      ShowMessage('Preencha a Descrição!');
      edDescricao.SetFocus;
      Abort; // Não use Exit.
    end;
  end;
  // Se o botão de excluir registro for
  // pressionado, pede confirmação do usuário antes
  // de executar a ação do botão.
  if Button = nbDelete then
    if MessageDlg('Confirmação',
      'Excluir Registro?',
      mtConfirmation,mbYesNo,'')
      ) <> 6 then Abort; // Não use Exit.
end;
```

O evento *BeforeAction* ocorre antes da ação do botão clicado no TDBNavigator ser enviada para o TSQLQuery (ou outro TDataSet descendente). Portanto, é útil para testarmos se os dados digitados estão corretos ou questionar o usuário.

O mesmo poderia ser feito usando os eventos do TSQLQuery que começam com a palavra *Before*.

Por fim, crie o evento *OnTitleClick* para o componente *gdDados* (TDBGrid) e digite o código abaixo em negrito:

```
procedure TfPrincipal.gdDadosTitleClick(Column:
TColumn);
begin
    dm.sqlCheques.Close;
    if Pos(Column.FieldName +
        ' ASC', dm.sqlCheques.SQL.Text) <= 0
    then
        dm.sqlCheques.SQL.Text :=
            'SELECT * FROM CHEQUES ORDER BY ' +
            Column.FieldName + ' ASC'
    else
        dm.sqlCheques.SQL.Text :=
            'SELECT * FROM CHEQUES ORDER BY ' +
            Column.FieldName + ' DESC';
    dm.sqlCheques.Open;
end;
```

O código acima ordena uma coluna da TDBGrid quando seu título é clicado. É feito um teste para saber se a coluna está em ordem ascendente ou descendente, a fim de inverter a ordenação.

Concluimos aqui o controle de cheques. Execute e teste o exemplo.

**Relacionamento Mestre / Detalhe com SQLdb:**

Tomemos como exemplo duas tabelas, *vendas* e *itens\_venda*.

A tabela *vendas* tem os seguintes campos: CODIGO e DATA\_VENDA.

Já a tabela *itens\_venda* possui os campos: CODIGO, COD\_VENDA, PRODUTO, QUANT, PRECO e TOTAL.

Para cada venda registrada teremos muitos itens associados. O campo CODIGO da tabela *vendas* está associado ao campo COD\_VENDA da tabela *itens\_venda*.

Veja como exemplo as tabelas abaixo:

VENDAS	
CODIGO	DATA_VENDA
1	10/04/2011
2	11/04/2011

ITENS_VENDA					
CODIGO	COD_VENDA	PRODUTO	QUANT	PRECO	TOTAL
1	1	CD-R	2	1,00	2,00
2	1	DVD-R	4	1,50	6,00
3	1	DVD-RW	3	2,00	6,00
4	2	MOUSE	2	8,00	16,00
5	2	TECLADO	2	23,00	46,00

Observe que a venda de CODIGO 1 possui 3 itens e a venda de CODIGO 2 possui 2 itens.



Vejamos como mostrar o relacionamento acima usando os componentes da paleta SQLdb.

Temos, então, um conector TSQLConnector de nome *banco* e um TSQLTransaction de nome *trans* , interligados e devidamente configurados.

Um TSQLQuery de nome *sqVenda* (que mostrará a tabela *vendas*), com as seguintes propriedades configuradas:

**Database** → *banco*

**SQL** → *SELECT \* FROM VENDAS*

**Transaction** → *trans*

Um TDataSource de nome *dsVenda* com a propriedade DataSet apontando para *sqVenda*. E um TDBNavigator e um TDBGrid ligado ao *dsVenda*.

Outro TSQLQuery de nome *sqlItem* (que mostrará a tabela *itens\_venda*), com as seguintes propriedades configuradas:

**Database** → *banco*

**SQL** → *SELECT \* FROM ITENS\_VENDA*

**Transaction** → *trans*

Outro TDataSource de nome *dsItem* com a propriedade DataSet apontando para *sqlItem*. E um TDBNavigator e um TDBGrid ligado ao *dsItem*.

O componente *sqlItem* deve ser filtrado pelo registro selecionado em *sqVenda*.

Portanto, precisamos digitar o seguinte código no evento OnDataChange do *dsVenda* (TDataSource ligado ao *sqVenda*):

```
sqItem.Close;  
sqItem.SQL.Text :=  
  'SELECT * FROM ITENS_VENDA WHERE COD_VENDA = ' +  
  sqVenda.FieldName('CODIGO').AsString;  
sqItem.Open;
```

Para inserir novos registros na tabela *itens\_venda* com *sqItem*, temos de digitar o código seguinte no evento *OnNewRecord* deste:

```
sqItem.FieldName('COD_VENDA').Value :=  
  sqVenda.FieldName('CODIGO').Value;
```

Observe nas figuras seguintes como funcionaria este exemplo:



Figura 12.8 – Exibindo Itens da Venda de Código 1

Então, de acordo com a figura acima, estando a venda de código 1 selecionada na grade mestre, a grade detalhe é filtrada para exibir somente os itens da venda correspondente.

O mesmo ocorre quando selecionamos a venda de código 2. Note como ficaria a grade detalhe na figura a seguir:



*Figura 12.9 – Exibindo Itens da Venda de Código 2*







Visto que foi feita apenas uma explanação conceitual de como podemos criar um relacionamento Mestre/Detalhe com SQLdb.






Portanto, para praticar, tente criar o exemplo completo usando qualquer banco de dados.

### Acessando Banco de Dados com ZEOS

O pacote ZeosLib, ou simplesmente Zeos, é uma biblioteca de código aberto que permite acesso nativo a diversos tipos de banco de dados. Possui muito mais recursos do que a paleta SQLdb. Por isso, é recomendado para grandes projetos.

#### Visão Geral dos Componentes:

 <b>TZConnection</b>	Responsável por realizar a conexão a vários bancos de dados. Controla, também, as transações.
 <b>TZReadOnlyQuery</b>	Similar ao TZQuery, mas não realiza alterações no banco de dados, só realiza consultas.
 <b>TZQuery</b>	Basicamente, responsável por realizar consultas, inclusões, alterações e exclusões em uma tabela do banco.
 <b>TZTable</b>	Similar ao TZQuery, mas sempre carrega na memória todos os registros de uma tabela do banco.
 <b>TZUpdateSQL</b>	Ligado ao TZQuery, possibilita personalizar os comandos de inserção, alteração e exclusão.
 <b>TZStoredProc</b>	Usado para executar funções ou stored procedures do banco de dados.

 <b>TZSQLMetadata</b>	Usado para extrair informações referente as estruturas do banco de dados.
 <b>TZSQLProcessor</b>	Usado para executar no banco de dados um texto contendo uma sequência de comandos em SQL.
 <b>TZSQLMonitor</b>	Cria um log com todas as consultas enviadas e respostas do servidor de banco de dados.
 <b>TZSequence</b>	Usado para manipular geradores de números sequenciais. Por exemplo, um GENERATOR do Firebird.
 <b>TZIBEventAlerter</b>	Captura os alertas disparados por eventos no servidor InterBase ou Firebird.

### **Configurando Conexão Usando o TZConnection:**

O componente TZConnection é o responsável pelas conexões aos bancos de dados e, também, controla as transações. Basicamente, para acessarmos um banco de dados com este componente, precisamos configurar as seguintes propriedades:

**Database** – O nome do banco de dados a ser acessado. Se for um arquivo, deve incluir o caminho completo e o nome do arquivo (Ex.: /home/usuario/clientes.fdb).

**HostName** – O IP da máquina onde está rodando o servidor do banco de dados. Se for a máquina local, o valor pode ser *localhost* ou *127.0.0.1*.

**Password** – A senha de acesso ao servidor do banco.

**Port** – O número da porta de acesso usada pelo servidor do banco de dados. Na maioria das vezes é opcional (Portas padrão: PostgreSQL – 5432, Firebird – 3050 e MySQL – 3306).

**Protocol** – Aqui escolhemos o tipo de banco de dados que será acessado.

**TransactIsolation** – Tipo de isolamento das transações. Geralmente, deixa-se em *tiReadCommitted* .

**User** – Nome do usuário que tem acesso ao servidor do banco.

Para acessar um banco SQLite Versão 3, só precisamos informar o caminho e o nome do arquivo de banco de dados na propriedade *Database* (Ex.: */home/usuario/teste.db*), em Protocol escolher *sqlite-3* e em TransactIsolation escolher *tiReadCommitted* .

Para ativar a conexão com o banco de dados, basta mudar a propriedade *Connected* para *True*. Veja a seguir como ativar a conexão via código (*ZConecta* é um *TZConnection*):

```
ZConecta.Connect;
```

ou

```
ZConecta.Connected := True;
```

Para desativar:

```
ZConecta.Disconnect;
```

ou

```
ZConecta.Connected := False;
```

### O componente TZQuery:

Componente usado para consulta e alteração de tabelas do banco de dados, chamar Stored Procedures e Funções SQL. Para exercer esta função, é necessário ligar a propriedade *Connection* deste há um componente TZConnection. Feito isso, digitamos na propriedade *SQL* a consulta ou comando de ação (Ex.: `SELECT * FROM NOME_DA_TABELA`).

Tudo que foi considerado no capítulo anterior sobre o componente TSQLQuery da paleta SQLdb aplica-se ao TZQuery, com exceção das propriedades *ServerFilter*, *ServerFiltered*, *InsertSQL*, *DeleteSQL* e *UpdateSQL* que não existem no TZQuery (as três últimas estão disponíveis no componente *TZUpdateSQL*). Também, não existe o evento *OnUpdateError*, mas possui outros eventos que não encontramos no TSQLQuery.

Assim como no TSQLQuery, os comandos SQL para inserção, alteração e exclusão, em uma consulta, são criados internamente. Mas se desejar personalizá-los, deverá ligar a propriedade *UpdateObject* do TZQuery a um componente TZUpdateSQL.

Por padrão, quando o método *Post* ou *Delete* do TZQuery (ou TZTable) é chamado, automaticamente é feito um *CommitRetaining* salvando os dados no banco de dados. Neste aspecto, para o TZQuery funcionar como o TSQLQuery, mude para *True* a propriedade *CachedUpdates* do TZQuery e a propriedade *AutoCommit* do TZConnection para *False*.

### Criando um Cadastro de Produtos com Firebird e ZEOS:

No desenvolvimento deste exemplo, em vez de usarmos um TDBNavigator, usaremos botões personalizados para executar

os métodos do TZQuery para navegar, incluir, editar, salvar, cancelar e excluir.

Todas as pesquisas serão feitas diretamente com comandos SQL.

Usaremos o servidor de banco de dados Firebird 2.1, mas pode-se usar qualquer outro banco de dados, como o PostgreSQL ou MySQL.

**OBS.:** O ZEOS versão 6.6.6 não tem suporte ao Firebird 2.5. A versão 7 do ZEOS tem suporte, mas está em estado Alfa de desenvolvimento.

Para saber como instalar e configurar o servidor Firebird no Linux e no Windows, veja o *Apêndice – E*.

Devido ao sistema de permissões do Linux, precisamos dar permissões para o Firebird salvar os bancos em uma determinada pasta. Para isso, abra um terminal do Ubuntu em *Aplicativos > Acessórios > Terminal* e digite os dois comandos abaixo:

```
mkdir ~/bancos
```

```
sudo chown firebird:firebird -R ~/bancos
```

O primeiro comando cria uma pasta chama *bancos* dentro de sua “*Pasta pessoal*” e o segundo muda as permissões.

Para criar o banco, usaremos o programa FlameRobin (Veja instalação no *Apêndice – E*) que tem versão para Linux e Windows (pode-se usar, também, o IBExpert Personal que tem versão só para Windows, porém não será explicado seu uso).



Ao executar o FlameRobin, temos a tela mostrada abaixo:

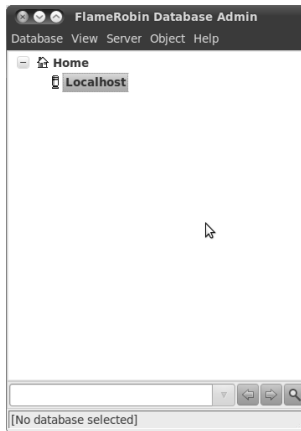


Figura 13.1 – Tela Principal do FlameRobin

Clique no menu *Database > Create new database...* , vai surgir o diálogo mostrado abaixo:

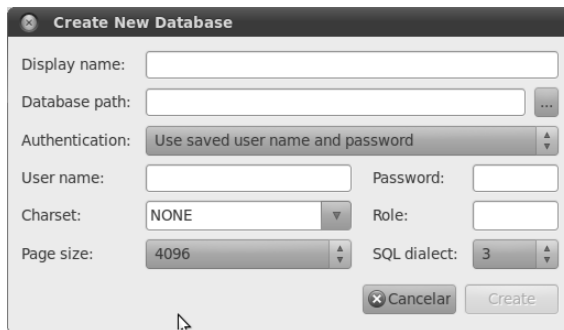


Figura 13.2 – Diálogo para Criar Novo Banco de dados

Em *Display name:* , digite o nome que deve aparecer no FlameRobin para referir-se ao banco. Neste caso, digite *Estoque* .

Em *Database path:* , digite o caminho e o nome do banco a ser criado. Neste caso, digite: */home/{seu\_nome\_de\_usuario}/bancos/estoque.fdb* (Ex.: */home/jean/bancos/estoque.fdb*). No Windows, você pode criar o banco na pasta raiz do sistema ou em qualquer outra pasta (Ex.: *C:\estoque.fdb* ou *C:\bancos\estoque.fdb*).

**DICA:** Você pode criar uma pasta dentro da pasta do seu projeto para ficar o arquivo do banco. Mas, se estiver no Linux, não esqueça de dar permissões para o Firebird acessar a pasta.

Em *User name:* , digite o nome do usuário que tem acesso ao banco. Neste caso, digite *SYSDBA* que é o padrão.

Em *Password:* , digite a senha do usuário está acessando o banco. Neste caso, a senha do usuário padrão é *masterkey* .

Em *Charset:* , escolhemos a tabela de caracteres a ser usada. Escolha *UTF8* por ser padrão dos novos sistemas operacionais.

Não é necessário alterar os outros parâmetros. Clique no botão *Create* .


Na janela do FlameRobin, expanda *Localhost* e *Estoque* clicando nos sinais de mais ( + ). Clique com o botão direito do mouse em *Tables* e clique em *Create new...* . Apague o conteúdo do editor e digite o código SQL abaixo:

```
CREATE TABLE PRODUTOS (  
    ID BIGINT NOT NULL,  
    DATACAD DATE,  
    REFERENCIA VARCHAR(15),  
    COD_BARRAS VARCHAR(30),  
    DESCRICAO VARCHAR(150),
```

```
QTD_ESTOQUE DOUBLE PRECISION,  
PRECO_CUSTO DOUBLE PRECISION,  
PRECO_VENDA DOUBLE PRECISION,  
FOTO BLOB SUB_TYPE 0,  
PRIMARY KEY (ID)  
);
```

Pressione F4 e depois F5 para criar a tabela PRODUTOS.

Dê um duplo clique na tabela *PRODUTOS*, na janela do FlameRobin.

Na janela seguinte, clique no ícone de lupa , o lado do campo *ID*. No diálogo que surgir, marque *Create new generator*, depois marque *Create trigger*, clique no botão *Execute* e, quando surgir o editor, pressione F5. Este procedimento torna o campo *ID* auto-numérico.

Com a janela da tabela PRODUTOS ainda aberta, clique no link *Indices*, depois em *Create new index*. No diálogo seguinte, em *Index name:* digite *IDX\_PROD\_REFERENCIA*, marque *Unique index*, selecione o campo REFERENCIA e clique no botão *Create* e, quando surgir o editor, pressione F5. Repita o processo para o campo COD\_BARRAS (porém, em *Index name:* digite *IDX\_PROD\_COD\_BARRAS*) e repita novamente o processo para o campo DESCRICAO (mas no caso deste, não marque *Unique index*).

**NOTA:** Os índices são importantes para agilizar as consultas e manter a integridade dos dados.

Com o banco e a tabela criados, vamos partir para o desenvolvimento do sistema.

Depois de pronto, o sistema de cadastro de produtos ficará como mostrado na figura seguinte:



Figura 13.3 – Tela do Cadastro de Produtos

Para começar, crie uma pasta na pasta *livro\_projetos* com o nome *cad\_produtos*. No Lazarus, crie um novo projeto do tipo Aplicação. Salve na pasta *cad\_produtos* o projeto com o nome *cad\_produtos.lpi* e a unit com o nome *u\_cad\_produtos.pas*.

Mude o nome do formulário para *fCadProdutos* e o Caption para “Cadastro de Produtos”. Também, mude as seguintes propriedades:

**Position** → *poDesktopCenter*

**Height** → 390

**Width** → 750

**Constraints > MaxHeight** → 390

**Constraints > MaxWidth** → 750

**Constraints > MinHeight** → 390

**Constraints > MinWidth** → 750

**Font > Size** → 9

Agora vamos acrescentar ao projeto um formulário DataModule. Clique no menu *Arquivo / Novo ...* , selecione *Data Module* e clique em OK.

Pressione F12 até aparecer o Data Module. Provavelmente, ele estará bem pequeno. Redimensione-o e mude a propriedade Name para *dm* . Pressione CTRL + SHIFT + S e salve a *unit* com o nome *u\_dados.pas* .

Em seguida, no *dm*, insira da paleta “Zeos Access” um TZConnection e um TZQuery.

Mude as seguintes propriedades:

TZConnection	
<b>Propriedade</b>	<b>Valor</b>
CharSet	UTF8
Database	<i>{caminho e arquivo do banco}</i>
HostName	localhost
Name	cnBanco
Password	masterkey
Port	3050
Protocol	Firebird-2.1
TransactIsolationLevel	tiReadCommitted
User	SYSDBA

TZQuery	
<b>Propriedade</b>	<b>Valor</b>
Connection	cnBanco
Name	tbProdutos
SortedFields	ID
SQL	SELECT FIRST 100 * FROM PRODUTOS ORDER BY ID DESC

Ainda no Data Module, insira, também, um TOpenPictureDialog da paleta Dialogs e mude a propriedade Name para opFoto, e insira um TImageList da paleta Common Controls.

Baixe o arquivo do link:

<http://www.jpsoft.com.br/downloads/navegador.png>

Mude as propriedades Height e Width do ImageList1 para 24, depois dê um duplo clique neste e vai surgir um diálogo, clique no botão **Adicionar...**, localize o arquivo *navegador.png* que você baixou, selecione-o e clique no botão **Abrir**. Surgirá uma mensagem perguntando se você quer separar a imagem, clique em Sim e agora clique em OK.

Dê um duplo clique no ícone do componente tbProdutos.

No diálogo que surgir, clique no botão com o sinal de mais ( + ) e no diálogo seguinte selecione todos os campos e clique no botão *Create*.

O diálogo anterior ficará como mostrado abaixo:

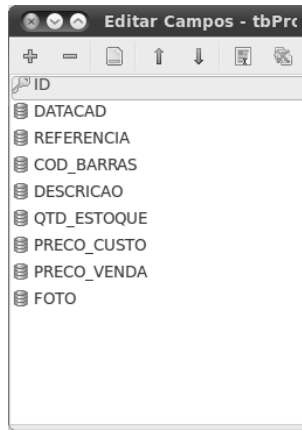


Figura 13.4 – Diálogo “Editar Campos”

No diálogo acima, selecione o campo ID e, no Inspetor de Objetos mude as seguintes propriedades:

**Alignment** → *taCenter*

**DisplayFormat** → *00000*

**ProviderFlags > pfInKey** → *True*

**ProviderFlags > pfInUpdate** → *False*

**Required** → *False*

Agora, selecione DATACAD e mude Alignment para *taCenter*. Depois, mude a propriedade Currency para *True* (mostra os números no formato de moeda) dos campos PRECO\_CUSTO e PRECO\_VENDA. Selecione o campo QTD\_ESTOQUE e na propriedade DisplayFormat digite *###,##0.00* . Estando o campo QTD\_ESTOQUE ainda selecionado, crie o evento *OnSetText* para ele e digite o código abaixo em negrito:

```
procedure Tdm.tbProdutosQTD_ESTOQUESetText(Sender:
TField; const aText: string);
begin
    try
```

```
if Trim(aText) = '' then
  Sender.Value := null
else
  Sender.Value := StrToFloat(aText);
except
  ShowMessage('Valor Inválido!');
  Exit;
end;
end;
end;
```

Abra novamente o diálogo “Editar Campos”, com um duplo clique no componente *tbProdutos*, selecione os campos PRECO\_CUSTO e PRECO\_VENDA e, no evento *OnSetText* destes, escolha *tbProdutosQTD\_ESTOQUESetText* . Pode fechar o diálogo.

Agora selecione o componente *cnBanco* e mude a propriedade Conected para *False*. O Data Module (*dm*) ficará similar ao mostrado abaixo:



Figura 13.5 – Data Module da Aplicação

Na unit *u\_dados* adicione na seção *uses* a unit *Dialogs* , como mostrado no código abaixo em **negrito**:



```
unit u_dados;
```

```
{ $mode objfpc } { $H+ }
```

```
interface
```

```
uses
```

```
Classes, SysUtils, DB, FileUtil, ExtDlgs,  
Controls, ZConnection, ZDataset, ZSqlUpdate,  
BufDataset, Dialogs;
```

```
type
```

Crie o evento *OnPostError* do componente *tbProdutos* e digite o código abaixo em negrito:

```
procedure Tdm.tbProdutosPostError(DataSet:  
TDataSet; E: EDatabaseError; var DataAction:  
TDataAction);  
begin  
    if Pos('IDX_PROD_REFERENCIA', E.Message) > 0 then  
        ShowMessage('Já existe um produto com a ' +  
            'referência ' + tbProdutosREFERENCIA.Value +  
            ' !')  
    else  
        if Pos('IDX_PROD_COD_BARRAS', E.Message) > 0  
            then ShowMessage(  
                'Já existe um produto com o código de ' +  
                'barras ' + tbProdutosCOD_BARRAS.Value +  
                ' !')  
        else  
            ShowMessage(  
                'Não foi possível incluir/alterar o ' +  
                'registro!' + #13+#13 + 'MENSAGEM: ' +  
                e.Message);  
        Abort;  
end;
```

Note que, na criação do banco, criamos dois índices únicos (*Unique index*) na tabela PRODUTOS. Um para o campo REFERENCIA com o nome IDX\_PROD\_REFERENCIA e outro para o campo COD\_BARRAS com o nome IDX\_PROD\_COD\_BARRAS. Isto significa que não pode haver dois registros com a mesma referência ou com o mesmo código de barras na tabela PRODUTOS. Se o usuário tentar fazer isso, um erro é lançado pelo servidor impedindo que o registro em duplicidade seja salvo.

Assim, o código digitado no evento *OnPostError*, trata estes erros mostrando mensagens mais amigáveis e claras para o usuário. Para saber qual foi o erro, testa-se a presença do nome do índice na mensagem de erro retornada. Se for outro erro não relacionado com os índices, é informado que não foi possível incluir ou alterar o registro e em seguida é exibida a mensagem retornada pelo servidor. De posse desta última informação, o usuário pode repassá-la ao suporte técnico ou ao desenvolvedor para análise e eventual correção do problema.

Agora, seguindo a mesma linha de raciocínio, crie o evento *OnDeleteError* do componente *tbProdutos* e digite o código abaixo em negrito:

```
procedure Tdm.tbProdutosDeleteError(DataSet:
TDataSet; E: EDatabaseError; var DataAction:
TDataAction);
begin
    ShowMessage (
        'Não foi possível excluir o registro!' +
        #13 + #13 + 'MENSAGEM: ' + e.Message);
    Abort;
end;
```

Para podermos acessar os componentes do Data Module (*dm*) no formulário *fCadProdutos* , precisamos informar a unit

(*u\_dados*) numa seção *uses* da unit *u\_cad\_produtos* abaixo da palavra *implementation* (poderia também ser na seção *uses* de *interface*). Também, para criarmos o código que ordena as colunas do TDBGrid (mais a frente) precisamos adicionar a unit *ZAbstractRODataset* :

Portanto, no Editor de Códigos, mude para guia da unit *u\_cad\_produtos* e digite o código em negrito na posição indicada:

```
var
    fCadProdutos: TfcadProdutos;

implementation

uses
    u_dados, ZAbstractRODataset;

{$R *.lfm}

{ TfcadProdutos }
```

Pressione F12 para exibir o formulário *fCadProdutos* e adicione um TToolBar, da paleta “Common Controls” e altere as seguintes propriedades:

**AutoSize** → *True*  
**ButtonHeight** → *48*  
**ButtonWidth** → *48*  
**EdgeBorders** > **edBottom** → *True*  
**Height** → *52*  
**Images** → *dm.ImageList1*  
**ShowCaption** → *True*

Clicando com o botão direito do mouse no *ToolBar1*, é mostrado um menu pop-up com opções para inserir botões,

separadores e etc. Insira dez botões e configure as propriedades dos itens de acordo com a tabela abaixo:

Botão	Name	Caption	Enabled	ImageIndex
1	btPrimeiro	Primeiro	False	0
2	btAnterior	Anterior	False	1
3	btProximo	Próximo	False	2
4	btUltimo	Último	False	3
5	btNovo	Novo	True	4
6	btEditar	Editar	False	5
7	btSalvar	Salvar	False	6
8	btCancelar	Cancelar	False	7
9	btExcluir	Excluir	False	8
10	btAtualizar	Atualizar	True	9

Adicione um TPanel ao formulário e mude a propriedade Align para *alBottom* e apague o conteúdo da propriedade Caption.

Também, coloque no formulário um TPageControl, da paleta “Common Controls”, e mude a propriedade Aling para *alClient* e TabStop para *False* .

Clique com o botão direito do mouse sobre o componente PageControl1 (TPageControl) e no menu que surgir clique na opção “Adicionar Página”. Repita para adicionar outra página.

Clique na página TabSheet1 e mude a propriedade Caption para *Dados*. Clique em TabSheet2 e mude o Caption para *Listagem*.

No Panel1 (TPanel) insira, na ordem, um TLabel, um TEdit, um TButton, um TBevel da paleta Additional, dois TRadioButton um acima do outro, outro TEdit, e mais dois TButton.

Mude o Caption do TLabel (Label1) para *Filtrar por Descrição* .  
Altere a propriedade Name do TEdit (Edit1) para *edFiltrar* , a propriedade Width para *263* , TabStop para *False* , MaxLength para *150* e apague o conteúdo da propriedade Text .

Para o TButton que ficará logo à direita do componente *edFiltrar* , modifique sua propriedade Name para *btFiltrar* , Caption para *Filtrar* , TabStop para *False* e Width para *47* .

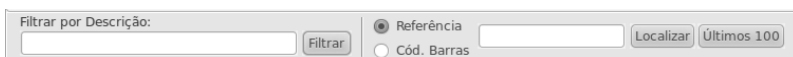
O TBevel deve ficar posicionado logo após o botão *btFiltrar* .  
Altere a propriedade Height para *40* , Shape para *bsLeftLine* , Top para *5* e Width para *5* .

No RadioButton1 mude o Caption para *Referência* , Checked para *True* e o Name para *rbRef* . E no RadioButton2 , mude o Caption para *Cód. Barras* e o Name para *rbCbr* .

Já no segundo TEdit (Edit2), altere a propriedade Name para *edLocalizar* , a propriedade Width para *140* , TabStop para *False* , MaxLength para *15* e apague o conteúdo da propriedade Text .

Os dois últimos botões devem ter a propriedade Name definida para *btLocalizar* e *btUltimos100* , o Caption para *Localizar* e *Últimos 100* , o Width para *62* e *83* e TabStop para *False* e *False*, respectivamente.

Organize os componentes como mostrado abaixo:



*Figura 13.6 – Painel de Pesquisas*

Insira no formulário fCadProdutos um TDataSource da paleta “Data Access” e altere as propriedades:

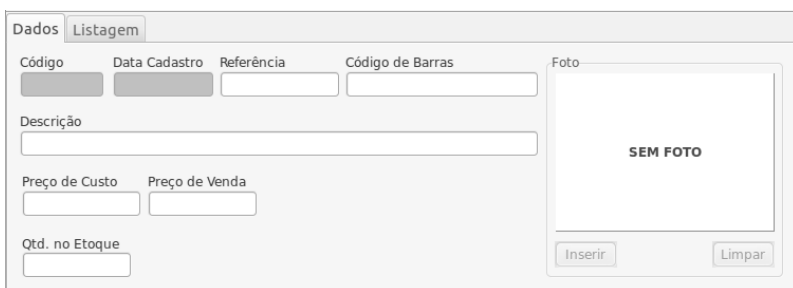
**Name** → *dsProdutos*

**DataSet** → *dm.tbProdutos*

Agora, na página *Dados* (TabSheet1) do PageControl1, insira oito TLabel e onze TDBEdit . Insira, também, um TGroupBox (mude as propriedades: Caption para *Foto*, Height para 212 e Width para 225) e dentro deste um TPanel (propriedades: Caption para *SEM FOTO*, Color para *clWhite*, Height para 150 e Width para 207).

Dentro do TPanel do TGroupBox insira um TDBImage e altere as propriedades: Align para *alCliente*, DataSource para *dsProdutos*, DataField para *FOTO*, Name para *imFoto* e Stretch para *True* . Abaixo do TPanel do TGroupBox , insira dois TButton .

Organize os componentes e mude a propriedade Caption dos TLabel's e TButton's para ficarem como mostrado na figura seguinte:



*Figura 13.7 – Campos do Cadastro de Produtos*

Mude a propriedade DataSource de todos os TDBEdit's para dsProdutos.

**LEMBRETE:** Para selecionar mais de um componente, afim de mudar propriedades em comum, clique no primeiro componente, segure a tecla SHIFT e clique nos componentes seguintes.

Em seguida, da esquerda para direita e de cima para baixo, mude a propriedade Name dos TDBEdit's para: *edCodigo*, *edDataCad*, *edReferencia*, *edCodBarras*, *edDescricao*, *edPcCusto*, *edPcVenda* e *edQtdEstoque*. Altere a propriedade Name do botão Inserir para *btInserir* e a do botão Limpar para *btLimpar*.

Mude a propriedade DataField dos componentes *edCodigo*, *edDataCad*, *edReferencia*, *edCodBarras*, *edDescricao*, *edPcCusto*, *edPcVenda* e *edQtdEstoque* para *ID*, *DATA CAD*, *REFERENCIA*, *COD\_BARRAS*, *DESCRICAO*, *PRECO\_CUSTO*, *PRECO\_VENDA* e *QTD\_ESTOQUE*, respectivamente.

Nos TDBEdit's *edCodigo* e *edDataCad*, também, mude as seguintes propriedades: Color para *clSilver*, ReadOnly para *True* e TabStop para *False*.

No PageControl1, clique na guia Listagem e dentro desta insira um TDBGrid e mude as seguintes propriedades:

**Align** → *alClient*

**AlternateColor** → *clSkyBlue*

**DataSource** → *dsProdutos*

**Name** → *gdDados*

**Options > dgRowSelect** → *True*

**ReadOnly** → *True*

Agora, dê um duplo clique no *gdDados* (TDBGrid) e no diálogo que surge, adicione oito colunas na grade usando o botão

*Adicionar.* No Inspetor de Objetos, mude as seguintes propriedades de cada coluna criada (não esqueça de seleccionar no diálogo a coluna a ser alterada):

Coluna	Propriedade	Valor
0	Alignment	taCenter
	DisplayFormat	00000
	FieldName	ID
	Title > Alignment	taCenter
	Title > Caption	Código
	Width	90
1	Alignment	taCenter
	FieldName	DATA CAD
	Title > Alignment	taCenter
	Title > Caption	Data Cad.
	Width	90
2	Alignment	taCenter
	FieldName	REFERENCIA
	Title > Alignment	taCenter
	Title > Caption	Referência
	Width	120
3	Alignment	taCenter
	FieldName	COD_BARRAS
	Title > Alignment	taCenter
	Title > Caption	Cód. Barras
	Width	150



<b>Coluna</b>	<b>Propriedade</b>	<b>Valor</b>
4	Alignment	taLeftJustify
	FieldName	DESCRICAO
	Title > Alignment	taCenter
	Title > Caption	Descrição
	Width	300
5	Alignment	taRightJustify
	FieldName	PRECO_CUSTO
	Title > Alignment	taCenter
	Title > Caption	Preço Custo
	Width	120
6	Alignment	taRightJustify
	FieldName	PRECO_VENDA
	Title > Alignment	taCenter
	Title > Caption	Preço Venda
	Width	120
7	Alignment	taCenter
	FieldName	QTD_ESTOQUE
	Title > Alignment	taCenter
	Title > Caption	Qtd. Estoque
	Width	100

No PageControl1, clique na guia Dados. Terminamos aqui a parte visual do projeto. Vamos então para o restante da codificação.

Crie o evento *OnClick* para o botão *btPrimeiro* do componente *TToolBar* e digite o código abaixo em negrito:

```
procedure TfCadProdutos.btPrimeiroClick(Sender:
TObject);
begin
    dm.tbProdutos.First;
end;
```

Crie o evento *OnClick* para o botão *btAnterior* do componente *TToolBar* e digite o código abaixo em negrito:

```
procedure TfCadProdutos.btAnteriorClick(Sender:
TObject);
begin
    dm.tbProdutos.Prior;
end;
```

Crie o evento *OnClick* para o botão *btProximo* do componente *TToolBar* e digite o código abaixo em negrito:

```
procedure TfCadProdutos.btProximoClick(Sender:
TObject);
begin
    dm.tbProdutos.Next;
end;
```

Crie o evento *OnClick* para o botão *btUltimo* do componente *TToolBar* e digite o código abaixo em negrito:

```
procedure TfCadProdutos.btUltimoClick(Sender:
TObject);
begin
    dm.tbProdutos.Last;
end;
```

Crie o evento *OnClick* para o botão *btNovo* do componente *TToolBar* e digite o código abaixo em negrito:

```
procedure TfCadProdutos.btNovoClick(Sender:
TObject);
begin
    btUltimos100.Click;
    PageControl1.PageIndex := 0;
    edReferencia.SetFocus;
    dm.tbProdutos.Append;
    dm.tbProdutosDATACAD.Value := date;
    dm.tbProdutosPRECO_CUSTO.Value := 0.00;
    dm.tbProdutosPRECO_VENDA.Value := 0.00;
    dm.tbProdutosQTD_ESTOQUE.Value := 0.00;
end;
```

Crie o evento *OnClick* para o botão *btEditar* do componente *TToolBar* e digite o código abaixo em negrito:

```
procedure TfCadProdutos.btEditarClick(Sender:
TObject);
begin
    PageControl1.PageIndex := 0;
    edReferencia.SetFocus;
    dm.tbProdutos.Edit;
end;
```

Crie o evento *OnClick* para o botão *btSalvar* do componente *TToolBar* e digite o código abaixo em negrito:

```
procedure TfCadProdutos.btSalvarClick(Sender:
TObject);
var
    reg: string;
begin
    if Trim(edDescricao.Text) = '' then
    begin
        ShowMessage('Preencha a Descrição!');
    end;
```

```
    Exit;
end;
if dm.tbProdutos.State = dsEdit then
    reg := dm.tbProdutosID.AsString;
dm.tbProdutos.Post;
dm.tbProdutos.Refresh;
if reg = '' then
    dm.tbProdutos.Last
else
    dm.tbProdutos.Locate('ID', reg, []);
end;
```

Crie o evento *OnClick* para o botão *btCancelar* do componente *TToolBar* e digite o código abaixo em negrito:

```
procedure TfCadProdutos.btCancelarClick(Sender:
TObject);
begin
    dm.tbProdutos.Cancel;
end;
```

Crie o evento *OnClick* para o botão *btExcluir* do componente *TToolBar* e digite o código abaixo em negrito:

```
procedure TfCadProdutos.btExcluirClick(Sender:
TObject);
begin
    if Application.MessageBox('Excluir o Produto?',
    'Confirmação', 6) <> 6 then Exit;
    dm.tbProdutos.Delete;
    if dm.tbProdutos.IsEmpty then
    begin
        btPrimeiro.Enabled := False;
        btAnterior.Enabled := False;
        btProximo.Enabled := False;
        btUltimo.Enabled := False;
        btEditar.Enabled := False;
        btExcluir.Enabled := False;
    end;
```

```
end;  
end;
```

Crie o evento *OnClick* para o botão *btAtualizar* do componente *TToolBar* e digite o código abaixo em negrito:

```
procedure TfCadProdutos.btAtualizarClick(Sender:  
TObject);  
begin  
    dm.tbProdutos.Refresh;  
end;
```

Selecione o componente *dsProdutos*, crie o evento *OnChange* para este e digite o código abaixo em negrito:

```
procedure TfCadProdutos.dsProdutosDataChange  
(Sender: TObject; Field: TField);  
begin  
    if not (dm.tbProdutos.State in  
        [dsInsert, dsEdit]) then  
        begin  
            btPrimeiro.Enabled := True;  
            btAnterior.Enabled := True;  
            btProximo.Enabled := True;  
            btUltimo.Enabled := True;  
        end;  
    if dm.tbProdutos.BOF then  
        begin  
            btPrimeiro.Enabled := False;  
            btAnterior.Enabled := False;  
        end;  
    if dm.tbProdutos.EOF then  
        begin  
            btProximo.Enabled := False;  
            btUltimo.Enabled := False;  
        end;  
end;
```

O código do evento *OnChange* controla a ativação ou desativação dos botões de navegação.

Selecione o componente *dsProdutos*, crie o evento *OnChange* para este e digite o código abaixo em negrito:

```
procedure TfCadProdutos.dsProdutosOnChange
(Sender: TObject);
begin
    if dm.tbProdutos.State in [dsInsert, dsEdit] then
        begin
            btPrimeiro.Enabled := False;
            btAnterior.Enabled := False;
            btProximo.Enabled := False;
            btUltimo.Enabled := False;
            btNovo.Enabled := False;
            btEditar.Enabled := False;
            btSalvar.Enabled := True;
            btCancelar.Enabled := True;
            btExcluir.Enabled := False;
            btAtualizar.Enabled := False;
            btInserir.Enabled := True;
            btLimpar.Enabled := True;
        end
    else
        begin
            btPrimeiro.Enabled :=
                not (dm.tbProdutos.IsEmpty);
            btAnterior.Enabled :=
                not (dm.tbProdutos.IsEmpty);
            btProximo.Enabled :=
                not (dm.tbProdutos.IsEmpty);
            btUltimo.Enabled :=
                not (dm.tbProdutos.IsEmpty);
            btEditar.Enabled :=
                not (dm.tbProdutos.IsEmpty);
            btExcluir.Enabled :=
                not (dm.tbProdutos.IsEmpty);
            btNovo.Enabled := True;
        end
    end;
end;
```

```
    btSalvar.Enabled := False;  
    btCancelar.Enabled := False;  
    btAtualizar.Enabled := True;  
    btInserir.Enabled := False;  
    btLimpar.Enabled := False;  
end;  
end;
```

Crie o evento *OnClick* para o botão *btInserir* (usado para inserir a foto do produto) e digite o código abaixo em negrito:

```
procedure TfCadProdutos.btInserirClick(Sender:  
TObject);  
begin  
    if dm.opFoto.Execute then  
    begin  
        try  
            imFoto.Picture.LoadFromFile(  
                dm.opFoto.FileName);  
        except  
            ShowMessage('Arquivo de Imagem Inválido!');  
        end;  
    end;  
end;  
end;
```

Crie o evento *OnClick* para o botão *btLimpar* (usado para excluir a foto do produto) e digite o código abaixo em negrito:

```
procedure TfCadProdutos.btLimparClick(Sender:  
TObject);  
begin  
    if dm.tbProdutosFOTO.AsVariant = Null then  
        Exit;  
    if Application.MessageBox(  
        'Deseja mesmo excluir a foto do produto?',  
        'Confirmação', 6) <> 6 then Exit;  
    imFoto.Picture.Clear;  
    dm.tbProdutosFOTO.AsVariant := Null;
```

end;

Crie o evento *OnClick* para o botão *btFiltrar* do *Panel1* e digite o código abaixo em negrito:

```
procedure TfCadProdutos.btFiltrarClick(Sender:
TObject);
begin
    dm.tbProdutos.Close;
    dm.tbProdutos.SQL.Text :=
        'SELECT * FROM PRODUTOS ' +
        'WHERE UPPER(DESCRICAO) LIKE UPPER(' +
        QuotedStr(edFiltrar.Text + '%') + ')';
    if trim(edFiltrar.Text) = '' then
    begin
        dm.tbProdutos.SQL.Text :=
            'SELECT FIRST 100 * FROM PRODUTOS ' +
            'ORDER BY ID DESC';
        dm.tbProdutos.SortedFields := 'ID';
        dm.tbProdutos.SortType := stAscending;
        StatusBar1.Panels[0].Text :=
            'Visualizando os Últimos 100 Registros';
    end;
    dm.tbProdutos.Open;
    if trim(edFiltrar.Text) <> '' then
        StatusBar1.Panels[0].Text :=
            'Filtrando por Descrição. Registros ' +
            'Retornados: ' +
            IntToStr(dm.tbProdutos.RecordCount);
end;
```

Crie o evento *OnClick* para o botão *btLocalizar* do *Panel1* e digite o código abaixo em negrito:

```
procedure TfCadProdutos.btLocalizarClick(Sender:
TObject);
begin
    edFiltrar.Text := '';
```



```
if rbRef.Checked then
  dm.tbProdutos.Locate(
    'REFERENCIA', edLocalizar.Text, [])
else
  dm.tbProdutos.Locate(
    'COD_BARRAS', edLocalizar.Text, []);
StatusBar1.Panels[0].Text :=
  'Visualizando Registro com Ref.: ' +
  dm.tbProdutos.FieldName(
    'REFERENCIA').AsString +
  ' e Cód. Barras: ' +
  dm.tbProdutos.FieldName(
    'COD_BARRAS').AsString;
end;
```

Crie o evento *OnClick* para o botão *btUltimos100* do *Panel1* e digite o código abaixo em negrito:

```
procedure TfCadProdutos.btUltimos100Click(Sender:
TObject);
begin
  dm.tbProdutos.Close;
  dm.tbProdutos.SQL.Text :=
    'SELECT FIRST 100 * FROM PRODUTOS ' +
    'ORDER BY ID DESC';
  dm.tbProdutos.SortedFields := 'ID';
  dm.tbProdutos.SortType := stAscending;
  dm.tbProdutos.Open;
  edFiltrar.Text := '';
  edLocalizar.Text := '';
  StatusBar1.Panels[0].Text :=
    'Visualizando os Últimos 100 Registros';
end;
```

Crie o evento *OnEnter* para o componente *edFiltrar* do *Panel1* e digite o código abaixo em negrito:

```
procedure TfCadProdutos.edFiltrarEnter(Sender:
TObject);
begin
    TEdit(Sender).Color := clSkyBlue;
end;
```

Crie o evento *OnExit* para o componente *edFiltrar* do *Panel1* e digite o código abaixo em negrito:

```
procedure TfCadProdutos.edFiltrarExit(Sender:
TObject);
begin
    TEdit(Sender).Color := clDefault;
end;
```

Ligue os eventos *OnEnter* e *OnExit* do componente *edLocalizar* do *Panel1* aos eventos *OnEnter* e *OnExit* do componente *edFiltrar* do *Panel1* (Veja um exemplo na figura abaixo):

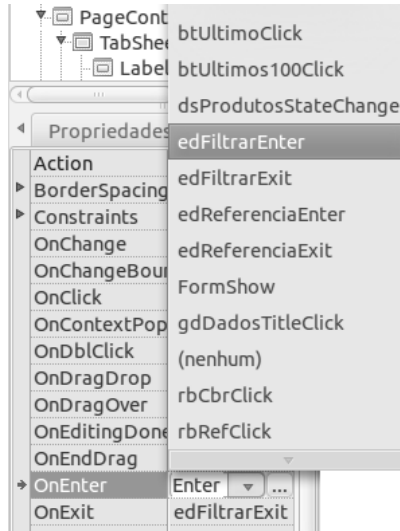


Figura 13.8 – Ligando o evento *OnEnter* do componente *edLocalizar* ao evento *OnEnter* do componente *edFiltrar*

Crie o evento *OnClick* para o componente *rbCbr* do *Panel1* e digite o código abaixo em negrito:

```
procedure TfCadProdutos.rbCbrClick(Sender:
TObject);
begin
    btUltimos100.Click;
    edLocalizar.MaxLength := 30;
end;
```

Crie o evento *OnClick* para o componente *rbRef* do *Panel1* e digite o código abaixo em negrito:

```
procedure TfCadProdutos.rbRefClick(Sender:
TObject);
begin
    btUltimos100.Click;
    edLocalizar.MaxLength := 15;
end;
```

Crie o evento *OnKeyPress* para o componente *edPcCusto* da guia Dados e digite o código abaixo em negrito:

```
procedure TfCadProdutos.edPcCustoKeyPress(Sender:
TObject; var Key: char);
{ $IFDEF LINUX }
    var
        posicao: Integer;
{ $ENDIF }
begin
    { $IFDEF LINUX }
        if key = ',' then begin
            posicao := TDBEdit(Sender).SelStart;
            TDBEdit(Sender).Text :=
                Copy(TDBEdit(Sender).Text, 0, posicao) +
                ',' + Copy(TDBEdit(Sender).Text, posicao +
                    1, Length(TDBEdit(Sender).Text));
            TDBEdit(Sender).SelStart := posicao + 1;
        end;
    { $ENDIF }
end;
```

```
    end;  
    {$ENDIF}  
end;
```

**NOTA:** O código do evento *OnKeyPress* do componente *edPcCusto* só é necessário no Linux. Acontece que o *TDBEdit*, no Linux, não aceita a digitação da virgula para casas decimais. Não sei se isto é um bug do Lazarus, mas esta é uma forma de resolver. Talvez, em versões futuras do Lazarus, este código não seja mais necessário. Observe que o código é genérico, só precisamos implementar uma vez na unit e ligar o *OnKeyPress* que tem a implementação aos eventos *OnKeyPress* dos outros *TDBEdit*'s que precisam do mesmo tratamento.

Agora, ligue o evento *OnKeyPress* dos componentes *edPcVenda* e *edQtdEstoque* da guia Dados ao evento *OnKeyPress* do componente *edPcCusto* da guia Dados.

Crie o evento *OnEnter* para o componente *edReferencia* da guia Dados e digite o código abaixo em negrito:

```
procedure TfCadProdutos.edReferenciaEnter(Sender:  
TObject);  
begin  
    TDBEdit(Sender).Color := clYellow;  
end;
```

Crie o evento *OnExit* para o componente *edReferencia* da guia Dados e digite o código abaixo em negrito:

```
procedure TfCadProdutos.edReferenciaExit(Sender:  
TObject);  
begin  
    TDBEdit(Sender).Color := clDefault;  
end;
```

Selecione os componentes *edCodBarras*, *edDescricao*, *edPcCusto*, *edPcVenda* e *edQtdEstoque* da guia Dados e ligue seus eventos *OnEnter* e *OnExit* aos eventos *OnEnter* e *OnExit* do componente *edReferencia*, respectivamente.

Clique na guia Listagem, selecione a TDBGrid *gdDados*, crie o evento *OnTitleClick* desta e digite o código abaixo em negrito:

```
procedure TfCadProdutos.gdDadosTitleClick(Column:
TColumn);
begin
  Application.ProcessMessages;
  if ((dm.tbProdutos.SortedFields =
    column.fieldname) and (dm.tbProdutos.SortType =
    stAscending)) then
    dm.tbProdutos.SortType := stDescending
  else
    if ((dm.tbProdutos.SortedFields =
    column.fieldname) and
    (dm.tbProdutos.SortType = stDescending)) then
      dm.tbProdutos.SortType := stAscending;
  if (dm.tbProdutos.SortedFields <>
    column.fieldname) then
    begin
      dm.tbProdutos.SortedFields := column.fieldname;
      dm.tbProdutos.SortType := stAscending;
    end;
    dm.tbProdutos.First;
  end;
```

O código acima permite ordenar (alternadamente em ordem crescente e decrescente) por uma coluna da *gdDados* clicando no título da coluna.

Clique na guia Dados.

Crie o evento *OnClose* para o formulário *fCadProdutos* e digite o código abaixo em negrito:

```
procedure TfCadProdutos.FormClose(Sender: TObject;  
var CloseAction: TCloseAction);  
begin  
    dm.tbProdutos.Close;  
    dm.cnBanco.Disconnect;  
end;
```

Crie o evento *OnKeyPress* para o formulário *fCadProdutos* e digite o código abaixo em negrito:

```
procedure TfCadProdutos.FormKeyPress(Sender:  
TObject; var Key: char);  
begin  
    if Key = #13 then  
    begin  
        SelectNext(activecontrol, True, True);  
        Key := #0;  
    end;  
end;
```

O código acima permite mudar o foco dos componentes usando a tecla ENTER em vez da tecla TAB.

Crie o evento *OnShow* para o formulário *fCadProdutos* e digite o código abaixo em negrito:

```
procedure TfCadProdutos.FormShow(Sender: TObject);  
begin  
    { $IFDEF LINUX }  
    CurrencyString := 'R$';  
    CurrencyFormat := 2;  
    DecimalSeparator := ',';  
    ThousandSeparator := '.';  
    DateSeparator := '/';  
    ShortDateFormat := 'dd/mm/yyyy';
```

```
{ $ENDIF }
// O código acima só é necessário no Linux para
// configurar a exibição de números formatados
// para moeda e para exibição correta de datas.
try
    // Define a tabela de caracteres
    // usada no banco.
    dm.cnBanco.Properties.Add('CODEPAGE=UTF8');
    dm.cnBanco.Connect;
    dm.tbProdutos.Open;
except // Comandos para tratar erros.
    // O objeto "e" conterá as informações do erro.
    on e: Exception do
        begin
            // Mostra para o usuário o que ocorreu.
            ShowMessage(
                'Não foi possível ativar o banco, ' +
                'o aplicativo será fechado!' + #13 + #13 +
                'CLASSE DO ERRO: ' + e.ClassName + #13 +
                'MENSAGEM: ' + e.Message);
            // Esconde o formulário.
            fCadProdutos.Hide;
            // Processa as mensagens
            // pendentes da aplicação.
            Application.ProcessMessages;
            // Termina a aplicação.
            Application.Terminate;
        end;
    end;
    edReferencia.SetFocus;
    rbRef.TabStop := False;
end;
```





Concluimos aqui o projeto. Execute e teste. Estude os códigos dos eventos. Observe que a estrutura do programa pode ser adaptada para criar todo tipo de formulário de cadastro (Funcionários, Fornecedores, Contas, etc). Além disso, para criar consultas Mestre/Detalhe com ZEOS, pode-se seguir o mesmo princípio apresentado no capítulo anterior.

### Criando Relatórios com LazReport







O ponto alto no desenvolvimento de aplicativos comerciais é, sem dúvida, a geração de relatórios. É o que vai agregar mais valor ao aplicativo. Porém, é a parte que demanda maior trabalho se não tivermos a disposição uma boa ferramenta geradora de relatórios. Certamente, o Lazarus sai-se bem neste quesito.



O LazReport é o gerador visual de relatórios, padrão do Lazarus. Ele é baseado no código fonte do FreeReport, que é a versão Open Source do FastReport. Atualmente, o LazReport possui recursos a mais em relação ao FreeReport. Porém, relatórios criados no FreeReport ou no FastReport não podem ser abertos pelo LazReport e vice-versa.

#### Visão Geral dos Componentes:

 <b>TfrReport</b>	Gerador de Relatório. Em tempo de design, um duplo clique neste, abre a janela de designer do relatório.
 <b>TfrTNPDFExport</b>	Torna possível salvar o relatório no formato PDF.
 <b>TfrTextExport</b>	Torna possível salvar o relatório no formato texto.
 <b>TfrHTMExport</b>	Torna possível salvar o relatório no formato HTML.

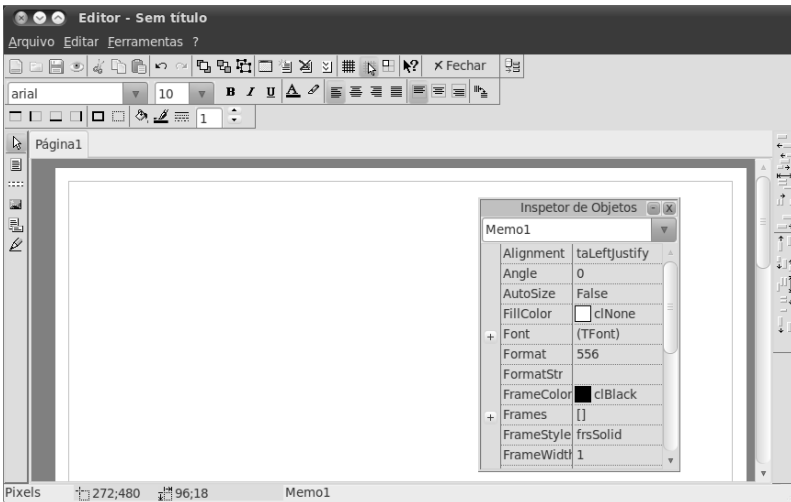


 <b>TfrCSVExport</b>	Torna possível salvar o relatório no formato CVS (Planilha).
 <b>TfrDBDataSet</b>	Necessário quando os dados do relatório são provenientes de um banco de dados.
 <b>TfrUserDataset</b>	Necessário quando os dados do relatório são provenientes de um array, arquivo texto, etc.
 <b>TfrCompositeReport</b>	Necessário quando desejamos combinar vários relatórios diferentes em um só.
 <b>TfrDesigner</b>	Necessário quando queremos permitir que o relatório seja alterado em tempo de execução.
 <b>TfrPrintGrid</b>	Usado para imprimir facilmente o conteúdo exibido por um componente TDBGrid.
 <b>TfrPreview</b>	Usado para criação de um visualizador personalizado de relatórios.
 <b>TfrBarCodeObject</b>	Provê campo para exibição de Códigos de Barras.
 <b>TfrRoundRectObject</b>	Provê retângulo com cantos arredondados para exibição de textos.

 <b>TfrShapeObject</b>	Provê componente para exibição de figuras geométricas (retângulo, círculo, triângulo, etc).
 <b>TfrCheckBoxObject</b>	Provê campo de checagem.

### O Componente TfrReport:

Inicie um novo projeto no Lazarus, e no formulário insira um TfrReport da paleta LazReport. Dê um duplo clique sobre este componente e você verá o Editor de Relatórios:



*Figura 14.1 – Tela Principal do Editor de Relatórios do LazReport*

Observe que algumas ferramentas do Editor de Relatórios do LazReport são as mesmas encontradas em editores de texto como LibreOffice Writer e Microsoft Word.

O editor possui um Inspetor de Objetos próprio que pode ser acessado a qualquer momento pressionando-se a tecla F11. Com ele, podemos alterar diversas propriedades dos componentes da barra de ferramentas a esquerda e propriedades das bandas (seções) do relatório.

Para configurar o tamanho da página, margens, colunas e etc, clicamos no menu “*Arquivo > Opções página*” .

No menu “*Arquivo > Opções de Relatório*” podemos escolher a impressora padrão, versão e propriedades do arquivo do relatório.







Através do menu “*Ferramentas > Opções...*” podemos configurar o Editor de Relatórios. Observe a figura seguinte:







*Figura 14.2 – Diálogo Opções de Relatório*

**Barra de Ferramentas Objetos:**

A barra de ferramentas a direita contém os componentes necessários para o desenvolvimento dos relatórios. Veja as tabelas abaixo:











Ferramenta	Descrição
	Selecionar Objetos
	Inserir Caixa de Texto
	Inserir Banda (Seção de Relatório)
	Inserir Figura
	Inserir Sub Relatório
	Desenhar Linhas

Através dos componentes TfrBarCodeObject, TfrRoundRectObject, TfrShapeObject e TfrCheckBoxObject temos, também, os seguintes objetos na barra de ferramentas:

Ferramenta	Descrição
	Inserir CheckBox
	Inserir Shape (retângulo, elipse, triângulo, etc)
	Inserir Retângulo Arredondado com Sombra
	Inserir Código de Barras

**Paleta de Alinhamento:**

A paleta de alinhamento facilita muito o posicionamento dos objetos que compõem um relatório. Observe a função de cada botão:

<b>Botão</b>	<b>Descrição</b>
	Alinha pela borda esquerda os componentes selecionados
	Alinha horizontalmente pelo eixo central dos componentes selecionados
	Alinha horizontalmente no centro da página o componente ou grupo de componentes selecionados
	Distribui horizontalmente os componentes selecionados para que fiquem com a mesma distância um do outro
	Alinha pela borda direita os componentes selecionados
	Alinha pela borda superior os componentes selecionados
	Alinha verticalmente pelo eixo central dos componentes selecionados
	Alinha verticalmente no centro da página o componente ou grupo de componentes selecionados
	Distribui verticalmente os componentes selecionados para que fiquem com a mesma distância um do outro
	Alinha pela borda inferior os componentes selecionados

**Barra de Ferramentas Padrão:**

Segue descrição de cada botão da barra de ferramentas padrão:



*Figura 14.3 – Barra de Ferramentas Padrão*

Botão	Descrição
1	Criar um novo relatório em branco.
2	Abrir um arquivo de relatório.
3	Salvar alterações.
4	Visualizar o relatório.
5	Recorta para área de transferência os componentes selecionados.
6	Copia para área de transferência os componentes selecionados.
7	Cola no relatório o conteúdo da área de transferência.
8	Desfazer alterações.
9	Refazer alterações.
10	Coloca o(s) objeto(s) selecionado(s) a frente do(s) outro(s) objeto(s).
11	Coloca o(s) objeto(s) selecionado(s) a trás do(s) outro(s) objeto(s).
12	Seleciona todos os objetos que compõem o relatório.
13	Este recurso ainda não está totalmente funcional.
14	Adiciona uma nova página no relatório.
15	Exclui páginas do relatório.

Botão	Descrição
16	Abre o diálogo “Opções de Página”.
17	Exibe grade de alinhamento.
18	Alinha os componentes selecionados na grade de alinhamento.
19	Ajusta o tamanho dos componentes selecionados pela grade de alinhamento.
20	Ajuda não implementada.
21	Fecha o editor de relatórios.

### Barra de Ferramentas Texto:

Segue descrição de cada botão da barra de ferramentas de texto:



*Figura 14.4 – Barra de Ferramentas Texto*

Botão	Descrição
1	Selecionar tipo de fonte.
2	Selecionar tamanho da fonte.
3	Negrito.
4	Itálico.
5	Sublinhado.
6	Selecionar cor da fonte.
7	Ferramenta de realce condicional.
8	Alinhar à esquerda

Botão	Descrição
9	Centralizar.
10	Alinhar à direita.
11	Justificar (sem efeito).
12	Alinha o texto na parte superior do objeto.
13	Centraliza verticalmente o texto no objeto.
14	Alinha o texto na parte inferior do objeto.
15	Alterna entre texto normal e rotacionado em 90°.

Se pretende criar relatórios multiplataforma (Linux e Windows) é importante ter instalado no Linux as fontes padrões da Microsoft.

Para instalá-las no Ubuntu e derivados, digite o comando abaixo num terminal (se estiver usando o Linux, é recomendável que instale as fontes para acompanhar os exemplos do livro):

```
sudo apt-get install ttf-mscorefonts-installer
```

Quando no Linux, na barra de ferramentas Texto do Editor de Relatórios, não selecione uma fonte em tipo de fonte, digite o nome sem o texto entre colchetes, pois só assim ocorre a mudança de fonte. Por exemplo: em vez de selecionar *arial [monotype]* digite só *arial* .

Outras fontes aceitáveis: courier new, courier 10 pitch, impact, comic sans ms, georgia, symbol, times, webdings, verdana e entre outras.

Outra alternativa é selecionar o objeto que deseja mudar a fonte, pressione F11 para mostrar o “Inspetor de Objetos” e



neste clique no botão de reticências da propriedade *Font*. O diálogo de fonte é aberto e você pode selecionar a fonte normalmente.

### **Barra de Ferramentas Retângulo:**

Esta barra de ferramentas é usada para inserir bordas ou mudar a cor de fundo dos objetos selecionados. Segue descrição de cada botão:



*Figura 14.5 – Barra de Ferramentas Retângulo*

<b>Botão</b>	<b>Descrição</b>
1	Borda superior.
2	Borda esquerda.
3	Borna inferior.
4	Borda esquerda.
5	Todas as bordas.
6	Sem bordas.
7	Selecionar cor de fundo.
8	Selecionar cor da borda.
9	Selecionar tipo de borda.
10	Espessura da borda.

### **Barra de Status do Editor de Relatórios:**

A barra de status do editor de Relatórios é muito útil na hora de posicionar e ajustar objetos.



Figura 14.6 – Barra de Status do Editor de Relatórios

Item	Descrição
1	Unidade de medida usada.
2	Distância entre o lado esquerdo do objeto selecionado e a extremidade esquerda da página.
3	Distância entre o topo do objeto selecionado e a extremidade superior da página.
4	Largura do objeto selecionado.
5	Altura do objeto selecionado.
6	Nome do objeto selecionado.
7	Texto do objeto selecionado.

### Usando o Teclado e o Mouse no Editor de Relatórios:

Segue a descrição das principais teclas de atalho do Editor de Relatórios do LazReport:

Teclas	Descrição
Teclas de Setas	Move a seleção para o próximo objeto.
CTRL + Teclas de Setas	Move o objeto selecionado na direção da seta.
SHIFT + Teclas de Setas	Aumenta ou diminui a dimensão do objeto selecionado na direção da seta.

<b>Teclas</b>	<b>Descrição</b>
ENTER	Abre o editor do objeto selecionado.
DELETE	Apaga o objeto selecionado.
CTRL + ENTER	Confirma alterações e fecha o editor do objeto selecionado.
CTRL + 1 .. 9	Define a espessura da linha das bordas do objeto selecionado.
CTRL + Z	Desfaz a última ação.
CTRL + Y	Refaz ação cancelada.
CTRL + G	Alterna a exibição da grade de posicionamento.
CTRL + B	Alterna a opção de alinhar na grade.
CTRL + F	Adiciona bordas ao objeto selecionado.
CTRL + D	Retira as bordas do objeto selecionado.
CTRL + X	Recortar.
CTRL + V	Colar.
CTRL + C	Copiar.
CTRL + A	Seleciona todos os objetos na página.
CTRL + N	Cria novo relatório vazio.
CTRL + O	Abre um relatório salvo.
CTRL + S	Salva o arquivo de relatório.
CTRL + P	Visualiza o relatório.


Podemos usar o mouse para selecionar objetos, posicioná-los e redimensioná-los assim como faríamos com imagens em um editor de texto comum. Basta usar as alças de seleção que ficam nas extremidades do objeto quando este está selecionado.

Um duplo clique sobre um objeto abre o editor padrão deste. Um duplo clique na página abre o diálogo de configuração da página.

Podemos segurar a tecla SHIFT e clicar nos objetos com o botão esquerdo do mouse para selecioná-los simultaneamente.

Para dimensionar vários objetos selecionados, arraste a bolinha vermelha no canto inferior direito do grupo de objeto.

### Trabalhando com Bandas (Seções de Relatório):

Quando clicamos no botão Inserir Banda  e clicamos em seguida na página do relatório, surge o seguinte diálogo:



*Figura 14.7 – Diálogo para Inserir Nova Banda (Seção) no Relatório*

Segue descrição das principais Bandas de Relatório:

<b>Banda</b>	<b>Descrição</b>
Título Relatório	Impresso apenas no início da primeira pagina do relatório.
Sumário Relatório	Impresso apenas na última página.
Cabeçalho Página	Impresso no início de cada página (no caso da primeira página, após o Título se houver).
Rodapé Página	Impresso no fim de cada página.
Dados Mestre	Imprime os dados extraídos de uma tabela mestre em uma consulta Mestre/Detalhe.
Cabeçalho Mestre	Impresso logo antes da banda Dados Mestre.
Rodapé Mestre	Impresso logo após a banda Dados Mestre.
Dados Detalhe	Imprime os dados extraídos de uma tabela detalhe em uma consulta Mestre/Detalhe.
Cabeçalho Detalhe	Impresso logo antes da banda Dados Detalhe.
Rodapé Detalhe	Impresso logo após a banda Dados Detalhe.
Dados Subdetalhe	Imprime os dados extraídos de uma tabela subdetalhe em uma consulta Mestre/Detalhe/Subdetalhe.
Cabeçalho Subdetalhe	Impresso logo antes da banda Dados Subdetalhe.
Rodapé Subdetalhe	Impresso logo após a banda Dados Subdetalhe.

### O Objeto Caixa de Texto:



Utilizado para mostrar uma ou mais linhas de texto estático ou dinâmico. Pode ser ligado a um campo de uma tabela de banco de dados. Além disso, pode exibir o valor de uma variável ou mostrar o resultado de uma expressão.

### A Ferramenta Realçar Atributo:



Este recurso permite mudar a formatação da fonte ou a cor de fundo de um objeto *Caixa de Texto* ou objeto *Retângulo Arredondado* de acordo com uma expressão de condição.

### Relatório do Controle de Cheques com TFrPrintGrid:

O componente TFrPrintGrid permite imprimir o conteúdo de um TDBGrid. É ideal para relatórios simples.

Vamos desenvolver um relatório de exemplo para o aplicativo Controle de Cheques. Abra no Lazarus o projeto Controle de Cheques.

Insira no formulário principal um componente TFrPrintGrid da paleta LazReport e configure as seguintes propriedades:

**Caption** → *RELATÓRIO DO CONTROLE DE CHEQUES*

**DBGrid** → *gdDados*

**Orientation** → *poLandscape*

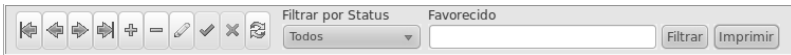
**Name** → *ImprimeGrade*

Também, insira no formulário o componente TfrTNPDFExport da paleta LazReport.

Agora, no Panel1 do formulário principal, insira um TButton e altere as propriedades:

**anchors > akLeft** → *False*  
**anchors > akRight** → *True*  
**Caption** → *Imprimir*  
**Name** → *btImprimir*  
**Width** → *63*

Reorganize os componentes no Panel1 para que fiquem como mostrado na figura abaixo:



*Figura 14.8 – Painel1 do Projeto Controle de Cheques com o novo botão Imprimir*

Crie o evento *OnClick* para o botão *btImprimir* e digite o código abaixo em negrito:

```
procedure TfPrincipal.btImprimirClick(Sender:
TObject);
begin
    // Configuração da fonte dos dados
    ImprimeGrade.Font.Name := 'arial';
    ImprimeGrade.Font.Color := clMaroon;
    ImprimeGrade.Font.Size := 9;
    // Configuração da fonte dos títulos
    ImprimeGrade.TitleFont.Name := 'arial';
    ImprimeGrade.TitleFont.Color := clNavy;
    ImprimeGrade.TitleFont.Size := 10;
    ImprimeGrade.TitleFont.Style := [fsBold];
    // Preview do relatório
    ImprimeGrade.PreviewReport;
end;
```

Por fim, selecione o componente *ImprimirGrade* (TFrPrintGrid), crie o evento *OnSetupColumn* e digite o código abaixo em negrito:

```
procedure TfPrincipal.ImprimeGradeSetupColumn
(Sender: TFrPrintGrid; const Column: TColumn; var
PrintColumn: boolean; var ColumnWidth: Integer);
begin
    // Garante que nenhuma coluna será impressa antes
    // de ser testada pelos if's abaixo.
    PrintColumn := False;
    // Cada if testa pelo título do TDBGGrid, qual
    // coluna está sendo impressa. Só as colunas que
    // corresponderem ao título pretendido é que
    // serão impressas.
    if Column.Title.Caption = 'Conta' then begin
        // Permite a impressão da coluna.
        PrintColumn := True;
        // Define a largura da coluna.
        ColumnWidth := 80;
    end;
    if Column.Title.Caption = 'Agência' then begin
        PrintColumn := True;
        ColumnWidth := 80;
    end;
    if Column.Title.Caption = 'Banco' then begin
        PrintColumn := True;
        ColumnWidth := 150;
    end;
    if Column.Title.Caption = 'Descrição' then begin
        PrintColumn := True;
        ColumnWidth := 300;
    end;
    if Column.Title.Caption = 'Favorecido' then begin
        PrintColumn := True;
        ColumnWidth := 200;
    end;
    if Column.Title.Caption = 'Valor' then begin
        PrintColumn := True;
        ColumnWidth := 100;
    end;
    if Column.Title.Caption = 'Status' then begin
        PrintColumn := True;
```



```
ColumnWidth := 80;  
end;  
end;
```

Como observado no código acima, podemos usar o evento *OnSetupColumn* para configurar a exibição das colunas do TDBGrid no relatório. A ordem de posicionamento das colunas no relatório segue a mesma ordem do TDBGrid.

Compile e execute o programa. Ao clicar no botão Imprimir, surgira o diálogo com a pré-visualização do relatório. A barra de ferramentas do “Preview de Relatórios” permite imprimir, salvar, mudar o zoom, etc. Clicando no botão Salvar, podemos, no diálogo que se abre, escolher salvar o relatório no formato PDF.



*Figura 14.9 – Relatório do Controle de Cheques*

### Criando Relatórios para o Cadastro de Produtos:

No tópico anterior, criamos um relatório para o aplicativo Controle de Cheques usando o componente TFrPrintGrid. Agora, vamos criar relatórios personalizados para o Cadastro de Produtos, usando o Editor de Relatórios do componente TfrReport.

Abra no Lazarus o projeto “Cadastro de Produtos”. No DataModule insira um TfrReport , um TfrDBDataSet e um TfrTNPDFExport .

Altere as seguintes propriedades do TfrReport :

**Dataset** → *frDBDataSet1*

**Name** → *frRelatorio*

**PreviewButons > pbFind** → *False*

**PreviewButons > pbHelp** → *False*

Em seguida, mude a propriedade DataSet do TfrDBDataSet para *tbProdutos* .

Dê um duplo clique no componente *frRelatorio* para abrir o Editor de Relatórios.

Clique no menu “Arquivo > Opções página” , na guia Opções deixe todas as margens com 20 milímetros e clique no botão OK. Veja a figura:

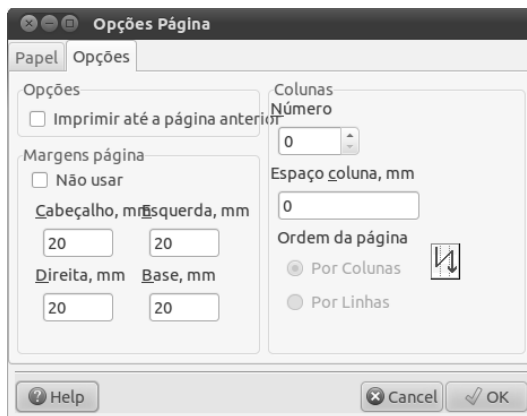


Figura 14.10 – Diálogo Opções de Página

Salve o relatório na pasta *cad\_produtos* com o nome *fichas\_produtos.lrf* . Periodicamente, salve as alterações durante o desenvolvimento do relatório.

De volta a página, use o botão “Inserir banda” da barra de ferramentas a esquerda e insira a banda “Título Relatório”.



*Figura 14.11 – Banda Título Relatório*

Usando o mouse, podemos arrastar a banda verticalmente. No entanto, a banda “Título Relatório”, sempre será impressa a partir da margem superior da página e somente na primeira página do relatório.

Observe as alças de seleção. Através delas podemos redimensionar a altura da banda.

Estando a banda selecionada, pressione F11. Surgirá o Inspetor de Objetos com as propriedades da banda selecionada. Altere a propriedade Height para 35 (pressione ENTER para validar a alteração).

**NOTA:** Use sempre o Inspetor de Objetos para alterar as propriedades dos objetos a seguir.

É na área cinza da banda que vamos posicionar os objetos que serão impressos neste setor. Então, clique no botão “Inserir caixa de texto” da barra de ferramentas, e clique sobre a área cinza da banda “Título Relatório”. Automaticamente, será aberto o editor padrão do objeto. Veja a figura a seguir:



Figura 14.12 – Editor de Texto do Objeto Selecionado

A medida que formos desenvolvendo os relatórios, entenderemos algumas funções deste diálogo. Então, digite na caixa “Texto”: *Fichas dos Produtos*. Clique no botão Ok.

Estando o objeto caixa de texto selecionado (se não estiver, selecione-o), pressione F11. No Inspetor de Objetos, clique ao lado da propriedade Font e clique no botão de reticências ( ... ). No diálogo de fonte, escolha Arial, tamanho 18, negrito e itálico. Clique em Ok. Altere, também, as seguintes propriedades:

**Alignment** → *taCenter*

**Height** → 30

**Layout** → *tlCenter*

**Width** → 920

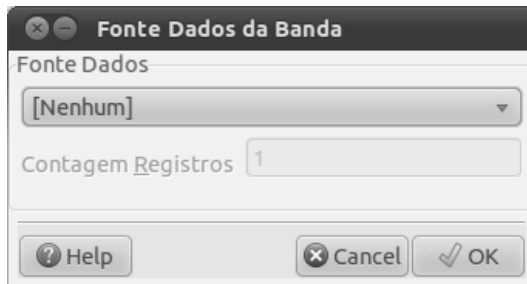
Usando o botão “Cor de fonte” da barra de ferramentas texto, altere a cor da fonte do texto do objeto selecionado para azul escuro. Posicione o objeto na banda “Título Relatório” como mostra a figura seguinte:



Figura 14.13 – Objeto Caixa de Texto

**NOTA:** Podemos visualizar, a qualquer momento, como está ficando o relatório clicando no botão “Visualizar relatório” ou através das teclas CTRL + P.

Insira, agora, uma banda “Dados Mestre” logo abaixo da banda “Título Relatório”. Note que ao inserir esta banda surge o diálogo abaixo:



*Figura 14.14 – Diálogo Fonte de Dados da Banda*

Em Fonte Dados escolha frDBDataSet1 e clique em OK.

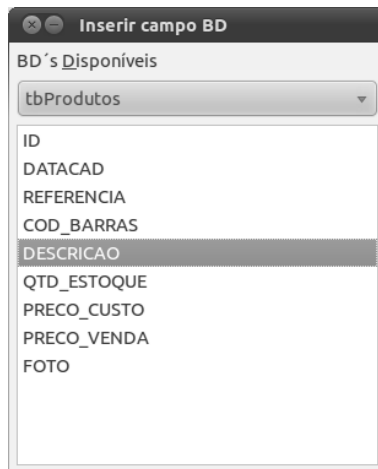
A banda “Dados Mestre” imprimirá os dados dos produtos. Altere a propriedade Height desta banda para 144. Execute os dois passos seguintes para criar as duas primeiras caixas de texto:

**1º Passo)** Adicione uma caixa de texto na banda “Dados Mestre” alinhada pelo topo com o texto *Descrição*. Mude a propriedade Height para 24 Width para 607, adicione bordas clicando no botão “Todas as bordas”, mude a cor de fundo com o botão “Cor plano de fundo” para turquesa, a cor da fonte para branca, fonte Arial tamanho 10 em negrito, e clique no botão que centraliza o texto na horizontal e no que centraliza na vertical (Figura 14.15).



*Figura 14.15 – Caixa de Texto Descrição*

**2º Passo)** Insira outra caixa de texto abaixo de *Descrição*. No diálogo que se abre, clique no botão “Campo DB” (Figura 14.12), no diálogo seguinte dê um duplo clique no campo DESCRICAO (Figura 14.16), retornando ao diálogo anterior, a caixa “Texto” fica com o conteúdo `[tbProdutos.”DESCRICAO”]` e clique em OK. Altere a propriedade Height para 24 Width para 607, adicione bordas clicando no botão “Todas as bordas”, mude a cor de fundo com o botão “Cor plano de fundo” para transparente, a cor da fonte para preto, fonte Arial tamanho 10, e clique no botão que alinha o texto para a esquerda na horizontal e no que centraliza na vertical (Figura 14.17).



*Figura 14.16 – Diálogo Inserir Campo*



Figura 14.17 – Caixa de Texto do Campo DESCRICAO

Repita seis vezes os dois passos acima identificados seguindo as alterações abaixo para cada repetição (Veja o posicionamento das caixas de texto na Figura 14.18):



Figura 14.18 – Posição das Caixas de Texto

1ª Repetição	1º Passo – Alterar:
	Texto <i>Código</i> , Width 101 e cor de fundo azul escuro
	2º Passo – Alterar:
	Campo <i>ID</i> , Width 101 e alinhamento horizontal centralizado

<b>2ª Repetição</b>	<b>1º Passo – Alterar:</b>
	Texto <i>Referência</i> , Width 141 e cor de fundo azul escuro
	<b>2º Passo – Alterar:</b>
	Campo <i>REFERENCIA</i> , Width 141 e alinhamento horizontal centralizado

<b>3ª Repetição</b>	<b>1º Passo – Alterar:</b>
	Texto <i>Código de Barras</i> , Width 213 e cor de fundo azul escuro
	<b>2º Passo – Alterar:</b>
	Campo <i>COD_BARRAS</i> , Width 213 e alinhamento horizontal centralizado

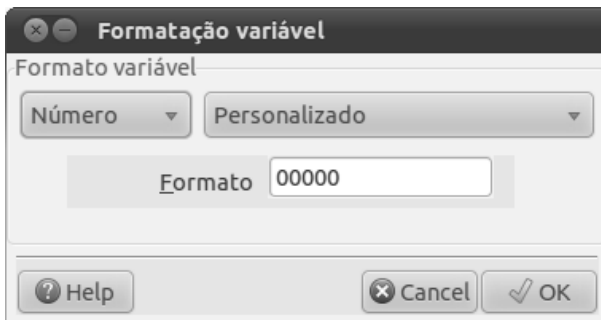
<b>4ª Repetição</b>	<b>1º Passo – Alterar:</b>
	Texto <i>Qtd. Estoque</i> , Width 101 e cor de fundo azul escuro
	<b>2º Passo – Alterar:</b>
	Campo <i>QTD_ESTOQUE</i> , Width 101 e alinhamento horizontal centralizado

<b>5ª Repetição</b>	<b>1º Passo – Alterar:</b>
	Texto <i>Preço Compra</i> , Width 177 e cor de fundo azul escuro
	<b>2º Passo – Alterar:</b>
	Campo <i>PRECO_CUSTO</i> , Width 177 e alinhamento horizontal a direita



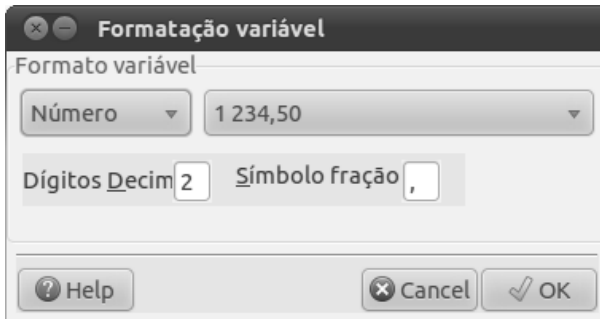
<b>6ª Repetição</b>	<b>1º Passo – Alterar:</b>
	Texto <i>Preço Venda</i> , Width 177 e cor de fundo azul escuro
	<b>2º Passo – Alterar:</b>
	Campo <i>PRECO_VENDA</i> , Width 177 e alinhamento horizontal a direita

Dê um duplo clique na caixa de texto do campo ID . No diálogo que se abre, clique no botão “Formato” (Figura 14.12) , deixe o diálogo seguinte configurado como mostra a Figura 14.19 e clique em OK, e em OK novamente.



*Figura 14.19 – Formatação da Caixa de Texto do Campo ID*

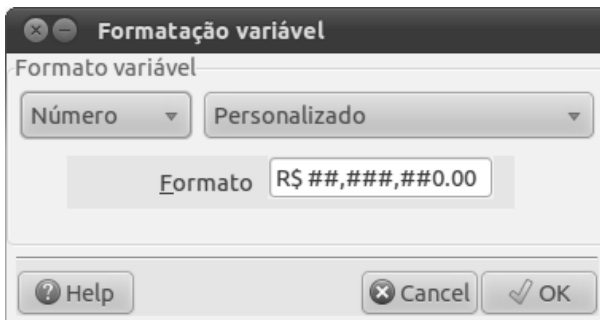
Dê um duplo clique na caixa de texto do campo QTD\_ESTOQUE . No diálogo que se abre, clique no botão “Formato” (Figura 14.12) , deixe o diálogo seguinte configurado como mostra a Figura 14.20 e clique em OK, e em OK novamente.



*Figura 14.20 – Formatação da Caixa de Texto do Campo QTD\_ESTOQUE*

Dê um duplo clique na caixa de texto do campo PRECO\_CUSTO . No diálogo que se abre, clique no botão “Formato” (Figura 14.12) , deixe o diálogo seguinte configurado como mostra a figura Figura 14.21 e clique em OK, e em OK novamente.

Dê um duplo clique na caixa de texto do campo PRECO\_VENDA . No diálogo que se abre, clique no botão “Formato” (Figura 14.12) , deixe o diálogo seguinte configurado como mostra a figura Figura 14.21 e clique em OK, e em OK novamente.



*Figura 14.21 – Formatação das Caixas de Texto dos Campos PRECO\_CUSTO e PRECO\_VENDA*

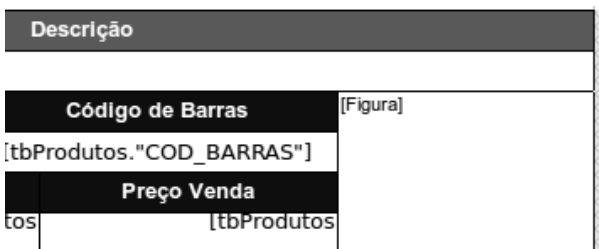
No espaço que ficou na banda “Dados Mestre”, insira um objeto imagem da barra de ferramentas a esquerda. Ao clicar na página surgirá o dialogo a seguir:



*Figura 14.22 – Diálogo Figura*

No diálogo Figura, marque “Estender” e clique no botão “Texto”. No diálogo que surge, clique no botão “Campo DB” (Figura 14.12), no diálogo seguinte, dê um duplo clique no campo FOTO e, de volta ao diálogo anterior, clique em OK.

Ajuste e redimensione o objeto imagem para ficar como mostrado na próxima figura:



*Figura 14.23 – Posição do Objeto Imagem*

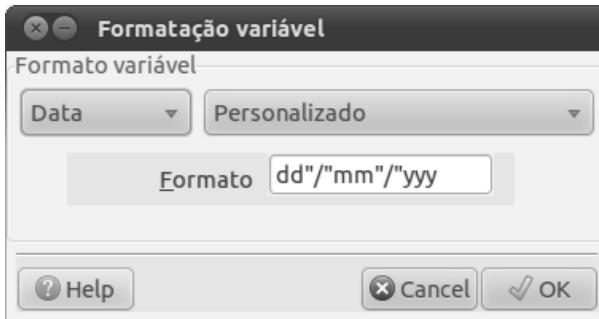
Agora, abaixo da banda “Dados Mestre” insira na página uma banda “Rodapé Página” e altere a propriedade Height para 23.

Nesta banda, insira duas caixas de texto alinhadas pela base da banda e uma em cada margem da página. Configure as caixas de texto para ficarem com fonte *Arial* tamanho 10, Width 160 e Height 18 . Na caixa da direita alinhe o texto a direita.

Dê um duplo clique na caixa de texto da esquerda. No diálogo que se abre, digite na caixa “Texto” a palavra *Data*: (deixe um espaço depois de dois pontos) e clique no botão “Variável” (Figura 14.12) . Outro diálogo é aberto onde podemos escolher entre variáveis que exibem informações sobre o relatório (Figura 14.24). Dê um duplo clique na variável “Data”. De volta ao diálogo anterior, a caixa “Texto” conterá *Data: [DATE]* . Agora, clique no botão “Formato” (Figura 14.12) , deixe o diálogo seguinte configurado como mostra a figura Figura 14.25 e clique em OK, e em OK novamente.

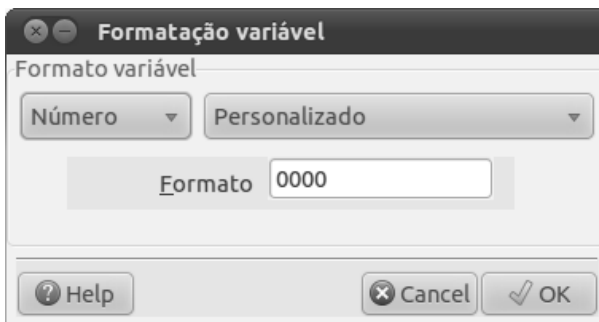


Figura 14.24 – Variáveis do Relatório



*Figura 14.25 – Formatar Variável [DATE]*

Dê um duplo clique na caixa de texto da direita. No diálogo que se abre, digite na caixa “Texto” a palavra *Página* - (deixe um espaço depois do traço) e clique no botão “Variável” (Figura 14.12) . Outro diálogo é aberto (Figura 14.24). Dê um duplo clique na variável “Página #”. De volta ao diálogo anterior, a caixa “Texto” conterá *Página - [PAGE#]* . Agora, clique no botão “Formato” (Figura 14.12) , deixe o diálogo seguinte configurado como mostra a figura Figura 14.26 e clique em OK, e em OK novamente.



*Figura 14.26 – Formatar Variável [PAGE#]*

A banda “Rodapé Página” deve ficar como mostra a Figura 14.27.



Figura 14.27 – Banda “Rodapé Página”

Salve as alterações. Aqui terminamos o primeiro relatório. Partiremos para o desenvolvimento de um segundo relatório para o projeto Cadastro de Produtos. Este relatório será uma lista de preços de venda.

No Editor de Relatórios clique no botão Novo. Configure a página para orientação Paisagem e as margens para 20mm.

Salve o relatório na pasta *cad\_produtos* com o nome *lista\_de\_precos.lrf*.

Agora, nós vamos usar um recurso interessante do Editor de Relatórios que visa agilizar o desenvolvimento do relatório.

Clique no menu “*Ferramentas > Ferramentas > Inserir campos DB*”. Surgirá um diálogo com os campos da tabela PRODUTOS, selecione os campos ID, REFERENCIA, COD\_BARRAS, DESCRICAO e PRECO\_VENDA. Marque as opções “Incluir Cabeçalhos” e “Incluir bandas” – Figura 14.28.

A página do relatório ficará como mostrado na Figura 14.29.

Insira borda em todas as caixas de texto, altere os tamanhos, as posições e os cabeçalhos conforme mostrado na Figura 14.30. Também, alinhe ao centro o texto das caixas de texto dos campos ID, REFERENCIA e COD\_BARRAS, dos campos RECO\_CUSTO e PRECO\_VENDA a direita. Altere as formatações dos campos como feito no relatório anterior. Por fim, troque o fundo das caixas de texto da banda “Cabeçalho Página” para azul escuro e fonte de cor branca.



Figura 14.28 – Diálogo Inserir Campos

Cabeçalho Página				
ID	REFERE	COD_BARRAS	DESCRICAO	PRE
Dados Mestre				
[tbPr	[tbPr	[tbPr	[tbPr	[tbPr

Figura 14.29 – Bandas Geradas Automaticamente

Cabeçalho Página				
Código	Referência	Cód. Barras	Descrição	Preço
Dados Mestre				
[tbProduto]	[tbProdutos]	[tbProdutos]	[tbProdutos."DESCRICAO"]	[tbProdutos]

Figura 14.30 – Caixas de Texto depois de Configuradas

Agora, podemos criar o título e o rodapé do relatório como visto no exemplo anterior – Figura 14.31.

Título Relatório	
LISTA DE PREÇOS	
Rodapé Página	
Data: [DATE]	Página - [PAGE#]

Figura 14.31 – Título e Rodapé do Relatório

Um recurso visual muito interessante é zebrar as linhas de dados do relatório. Para isso selecione a primeira caixa de texto da banda “Dados Mestre” e clique na ferramenta “Realçar atributos”. No diálogo que surge, em “Condição” digite *[LINE#] MOD 2 = 0*, desmarque “Negrito”, em “Plano de fundo” clique no botão “Cor ...” e escolha a cor amarela, e clique em OK – Figura 14.32.



Figura 14.32 – Diálogo Realçar Atributo

Repita o processo para as outras caixas de texto da banda “Dados Mestre”. Salve o relatório e feche o Editor de Relatórios. Mude a propriedade *Connected* do componente *cnBanco* para *False*.

Agora, que estamos com dois relatórios criados para o projeto Cadastro de Produtos, mas precisamos modificar nossa aplicação para acessar os relatórios.



Assim, insira mais um botão no *ToolBar1* do formulário *fCadProdutos* e configure as seguintes propriedades:

**Enabled** → *False*

**Caption** → *Imprimir*

**ImageIndex** → *10*

**Name** → *btImprimir*

Crie o evento *OnClick* para o botão *btImprimir* e digite o código em negrito mostrado a seguir:

```
procedure TfCadProdutos.btImprimirClick(Sender:
TObject);
begin
    try
        if (Application.MessageBox(
            '[Sim] para "Fichas dos Produtos".' +
            #13 + '[Não] para "Lista de Preços".' ,
            'Relatórios', 6) <> 6) then
            begin
                dm.frRelatorios.LoadFromFile(
                    'lista_de_precos.lrf');
                dm.frRelatorios.ShowReport;
            end else begin
                dm.frRelatorios.LoadFromFile(
                    'fichas_produtos.lrf');
                dm.frRelatorios.ShowReport;
            end;
        except
            ShowMessage(
                'Não foi possível gerar o relatório!');
        end;
    end;
```

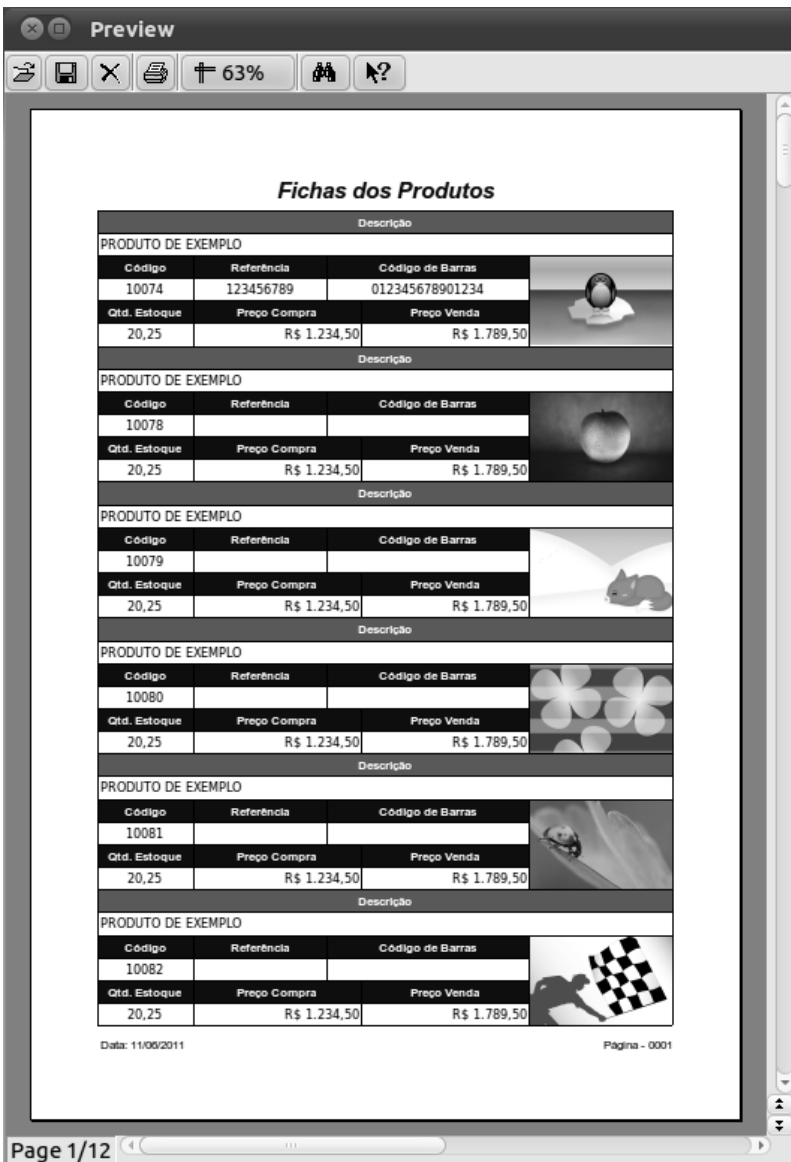
Altere o código do evento *OnStateChange* do componente *dsProdutos* conforme mostrado abaixo no texto em negrito:

```
procedure TfCadProdutos.dsProdutosStateChange
(Sender: TObject);
begin
    if dm.tbProdutos.State in [dsInsert, dsEdit] then
    begin
        btPrimeiro.Enabled := False;
        btAnterior.Enabled := False;
        btProximo.Enabled := False;
        btUltimo.Enabled := False;
        btNovo.Enabled := False;
        btEditar.Enabled := False;
        btSalvar.Enabled := True;
        btCancelar.Enabled := True;
        btExcluir.Enabled := False;
        btAtualizar.Enabled := False;
        btInserir.Enabled := True;
        btLimpar.Enabled := True;
        btImprimir.Enabled := False;
    end
    else
    begin
        btPrimeiro.Enabled :=
            not (dm.tbProdutos.IsEmpty);
        btAnterior.Enabled :=
            not (dm.tbProdutos.IsEmpty);
        btProximo.Enabled :=
            not (dm.tbProdutos.IsEmpty);
        btUltimo.Enabled :=
            not (dm.tbProdutos.IsEmpty);
        btEditar.Enabled :=
            not (dm.tbProdutos.IsEmpty);
        btExcluir.Enabled :=
            not (dm.tbProdutos.IsEmpty);
        btImprimir.Enabled :=
            not (dm.tbProdutos.IsEmpty);
        btNovo.Enabled := True;
        btSalvar.Enabled := False;
        btCancelar.Enabled := False;
        btAtualizar.Enabled := True;
    end
end;
```

```
        btInserir.Enabled := False;  
        btLimpar.Enabled := False;  
    end;  
end;
```

Execute e teste o programa e os relatórios.

**NOTA:** Para criar relatórios Mestre/Detalhe precisamos de dois TfrDBDataSet , um ligado a tabela mestre e outro ligado a tabela detalhe. No Editor de Relatórios insira uma banda “Dados Mestre” e ligue ao TfrDBDataSet mestre. Insira uma banda “Dados Detalhe” e ligue ao TfrDBDataSet detalhe.



*Figura 14.33 – Relatório Fichas dos Produtos*

Preview

89%

8?

### LISTA DE PREÇOS

Código	Referência	Cod. Barras	Descrição	Preço
10074	0123456789	012345678901234	PRODUTO DE EXEMPLO	R\$ 1.789,50
10078			PRODUTO DE EXEMPLO	R\$ 1.789,50
10079			PRODUTO DE EXEMPLO	R\$ 1.789,50
10080			PRODUTO DE EXEMPLO	R\$ 1.789,50
10081			PRODUTO DE EXEMPLO	R\$ 1.789,50
10082			PRODUTO DE EXEMPLO	R\$ 1.789,50
10083			PRODUTO DE EXEMPLO	R\$ 1.789,50
10084			PRODUTO DE EXEMPLO	R\$ 1.789,50
10085			PRODUTO DE EXEMPLO	R\$ 1.789,50
10086			PRODUTO DE EXEMPLO	R\$ 1.789,50
10087			PRODUTO DE EXEMPLO	R\$ 1.789,50
10088			PRODUTO DE EXEMPLO	R\$ 1.789,50
10089			PRODUTO DE EXEMPLO	R\$ 1.789,50
10090			PRODUTO DE EXEMPLO	R\$ 1.789,50
10091			PRODUTO DE EXEMPLO	R\$ 1.789,50
10092			PRODUTO DE EXEMPLO	R\$ 1.789,50
10093			PRODUTO DE EXEMPLO	R\$ 1.789,50
10094			PRODUTO DE EXEMPLO	R\$ 1.789,50
10095			PRODUTO DE EXEMPLO	R\$ 1.789,50
10096			PRODUTO DE EXEMPLO	R\$ 1.789,50
10097			PRODUTO DE EXEMPLO	R\$ 1.789,50
10098			PRODUTO DE EXEMPLO	R\$ 1.789,50
10099			PRODUTO DE EXEMPLO	R\$ 1.789,50
10100			PRODUTO DE EXEMPLO	R\$ 1.789,50
10101			PRODUTO DE EXEMPLO	R\$ 1.789,50
10102			PRODUTO DE EXEMPLO	R\$ 1.789,50
10103			PRODUTO DE EXEMPLO	R\$ 1.789,50
10104			PRODUTO DE EXEMPLO	R\$ 1.789,50
10105			PRODUTO DE EXEMPLO	R\$ 1.789,50

Data: 11/06/2011

Página - 0001

Figura 14.34 – Relatório Lista de Preços

### Configurando um Projeto

Quando estamos desenvolvendo um projeto, não basta nos preocuparmos só com os formulários e o código deste. É importante definir as características e a organização do projeto, como por exemplo: o ícone da aplicação, a versão, a ordem de criação dos formulários e o nome do executável. Vejamos como realizar estas configurações.

Primeiramente, abra qualquer um dos projetos de exemplo criados no livro e clique no menu “*Projeto > Opções de Projeto ...*”. Surgirá o diálogo “Opções do Projeto” que permite configurar nossa aplicação. Consideremos, então, as principais seções deste diálogo:

#### Configurações da Aplicação:

Selecione à esquerda a opção *Aplicação* :

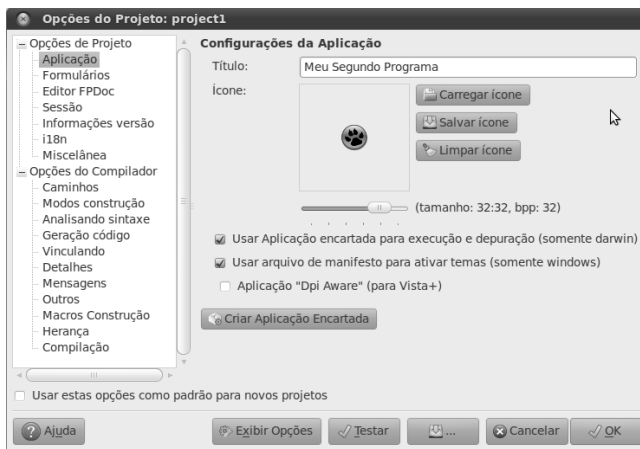
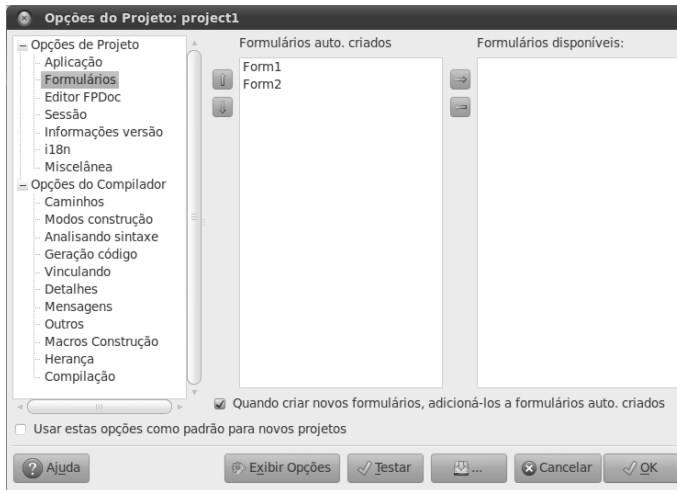


Figura A.1 – Configuração da Aplicação

Aqui você pode mudar o título da aplicação que aparecerá nas propriedades do arquivo (não é o nome do arquivo) e o ícone da aplicação. Este ícone aparecerá nas janelas e diálogos de mensagem da aplicação e, também, na barra de tarefas do sistema operacional.

### Configuração dos Formulários:

Selecione à esquerda a opção *Formulários* :



*Figura A.2 – Formulários do Projeto*

Aqui, você pode especificar quais os formulários da aplicação que serão criados automaticamente na memória e quais serão criados via código em alguma parte da aplicação. Você pode, também, alterar a ordem de auto criação dos formulários.

Todo novo formulário da aplicação é por padrão colocado para auto criação. Se desejar mudar isso, desmarque a opção “Quando criar novos formulários, adicioná-los a formulários auto. criados”.

O correto é auto criar somente o formulário principal de sua aplicação. Isso economiza memória e faz com que o programa inicie mais rápido.

Agora, se sua aplicação tiver um formulário com componentes de acesso a banco de dados (DataModule), este, também, deve ser colocado para auto criação. De preferência, primeiro que o formulário principal, afim de estar prontamente disponível para toda a aplicação.

Mas, como podemos chamar, no código, um formulário que não foi auto criado? Bem, digamos que sua aplicação tenha um formulário de nome *Form2* associado a uma *unit* de nome *Unit2*, que não foi auto criado e você deseja chamá-lo a partir de um botão no formulário principal.

Primeiro, na seção *implementation* do código do formulário principal, você deve chamar a *unit* do *Form2*, como mostra o código em negrito abaixo (isso seria necessário mesmo que *Form2* tivesse sido auto criado):

```
implementation

{$R *.lfm}

{ TFormPrincipal }

uses
    Unit2;
```

Agora, basta no evento *OnClick* do botão no formulário principal escrever o seguinte código:

```
Form2 := TForm2.Create(Self); { Cria uma instância  
do formulário na memória. }
```



```
Form2.ShowModal; { Mostra o formulário em modo  
exclusivo. Se Form2 tivesse sido auto criado,  
bastaria só esta linha para mostrá-lo. }  
Form2.Release; { Depois de fechado o formulário,  
libera a memória. Para ser novamente exibido, o  
Form2 teria de ser recriado com o comando:  
Form2 := TForm2.Create(Self) . }
```

### Configurações de Versão:

No diálogo “Opções do Projeto”, selecione à esquerda a opção “Informações versão”:

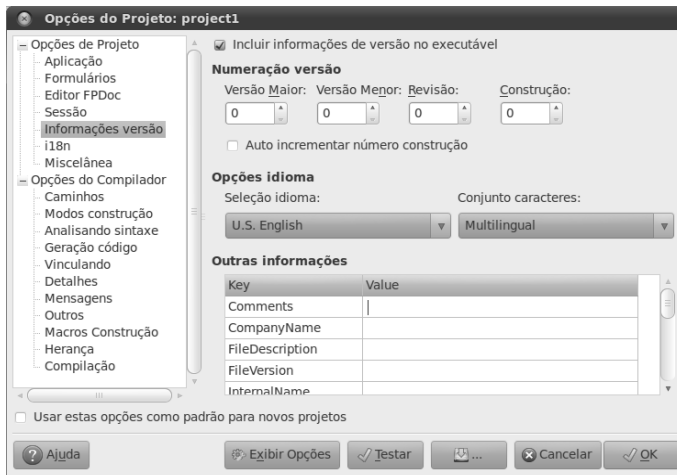


Figura A.3 – Controle de Versão

Aqui nesta seção é possível controlar a numeração da versão de sua aplicação. Informar o idioma de sua aplicação e outras informações.

### Configurando o Nome do Executável:

Selecione à esquerda a opção “Caminhos”. Na caixa de texto “Nome arquivo alvo (-o):”, você pode informar qual o nome do executável gerado para sua aplicação.

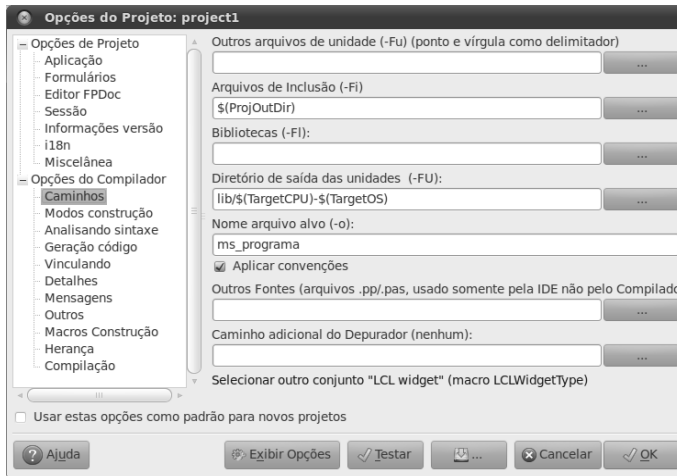


Figura A.4 – Nome do Executável Gerado

Estas são algumas das opções mais usadas. Tenha cuidado ao modificar outras opções, pois você poderia até mesmo desconfigurar a compilação do projeto, impedindo-o de ser executado.

### Dicas de Programação Multiplataforma

Como já considerado, o Free Pascal compila para diversos sistemas operacionais e arquiteturas de processamento. Evidentemente, estes não são iguais na apresentação visual, tabela de caracteres, acesso a arquivos e pastas, execução de programas externos e assim por diante.

Portanto, ao desenvolvermos um programa que deverá executar em mais de uma plataforma, devemos usar certas técnicas de codificação prevendo as diferentes características do ambiente de execução.

Este capítulo considerará técnicas de codificação para o desenvolvimento de sistemas que executarão em Linux e Windows. Sendo que, alguns dos recursos poderão ser usados para outros sistemas operacionais.

#### **Gerenciamento de Arquivos e Pastas:**

O delimitador de caminhos do Windows é o caractere “ponto e vírgula” ( ; ) e do Linux é o “dois pontos” ( : ).

O delimitador de diretórios ou pastas do Windows é o caractere “barra ao contrário” ( \ ) e do Linux o caractere “barra normal” ( / ).

O indicador de fim de linha de um arquivo texto no Windows é o conjunto de caracteres ASCII #13#10 e no Linux #10.

Para sanar estas diferenças, o Free Pascal (e consequentemente o Lazarus) possui uma série de constantes

que armazenam o delimitador de acordo com a plataforma em que o programa é executado.

**PathSep, PathSeparator:** delimitador de caminho do sistema operacional, para agrupar vários caminhos.

**PathDelim, DirectorySeparator:** delimitador de pastas do sistema operacional.

**LineEnding:** Sequência de caracteres que determina o fim de linha em um arquivo de acordo com o sistema operacional em execução.

Um exemplo de uso de uma constante multiplataforma:

```
'..' + PathDelim + 'agenda' + PathDelim + 'banco'
```

Resultado no Windows: ..\agenda\banco

Resultado no Linux: ../agenda/banco

Outra boa técnica é colocar e/ou salvar os arquivos que seu programa usará dentro da pasta do mesmo ou em pastas filhas.

Para extrair o caminho do seu programa, use o comando abaixo:

```
ExtractFilePath(Application.ExeName)
```

Agora, outro cuidado que devemos ter é relacionado ao fato que o sistema de arquivo do Linux é sensível a maiúsculas e minúsculas e o do Windows não. Assim, para não haver erro de acesso a arquivos e pastas, tanto no Windows como no Linux,

faça referência aos nomes de arquivos e pastas levando em conta as letras maiúsculas e minúsculas.

### Usando a diretiva `{$IFDEF}`:

Podemos usar a diretiva de compilação `{$IFDEF}` para executar, de maneira condicional, trechos de código, específicos para um sistema operacional. Observe os exemplos de sintaxe:

```
{$IFDEF LINUX}
    // Código(s) para executar no Linux
{$ELSE}
    // Código(s) para executar em outro sistema
{$ENDIF}

{$IFDEF MSWINDOWS}
    // Código(s) para executar no Windows
{$ELSE}
    // Código(s) para executar em outro sistema
{$ENDIF}

{$IFDEF LINUX}
    // Código(s) para executar no Linux
{$ENDIF}

{$IFDEF MSWINDOWS}
    // Código(s) para executar no Windows
{$ENDIF}
```

Vimos um pequeno exemplo de uso desta diretiva no desenvolvimento do “*Editor Simples*”. Veja a seguir o código do evento *OnCreate* do formulário do editor:

```
{$IFDEF MSWINDOWS}
    mCorFonte.Visible := False;
{$ENDIF}
```

O diálogo de fontes do Windows possibilita mudar a cor da fonte, mas o do Linux não tem esta opção. Para possibilitar a mudança de cor da fonte do editor no Linux, criamos uma entrada de menu só para esta função. Porém, não faria sentido esta entrada de menu ficar visível no Windows. Assim, o código acima, que esconde o menu cor da fonte, é executado somente quando o programa estiver rodando no Windows.

### Unidades Multiplataforma:

No delphi a unit *windows* provê funções que acessam a API (Interface de Programação de Aplicações) do Windows e, também, constantes pre definidas.

O Lazarus para Windows, também, possui esta unit, mas ela não é adicionada por padrão na seção *uses* do formulário, você tem que incluí-la manualmente.

É claro que a unit *windows* não está disponível no Lazarus para Linux. Porém, a maioria das funções da API do Windows estão disponíveis de uma maneira multiplataforma na unidade *LCLIntf*.

Portanto, ao chamar uma das funções da *LCLIntf*, caso esteja no Windows ela comunica-se com a API do Windows e no Linux é feito o direcionamento para a função correspondente na biblioteca GTK+ ou QT4.

As unit's *LCLType*, *LCLProc* e *LMessages*, também, disponibilizam outras constantes, funções e definições para substituir o uso da unit *windows*, *winproc*, etc .

Estas unit's, são especialmente úteis quando estamos convertendo um projeto do Delphi para o Lazarus com o objetivo de executá-lo tanto no Linux como no Windows.

### TProcess para Executar Programas Externos:

Quem trabalha com o Delphi sabe que para executar um programa externo (por exemplo: ter um botão no seu programa que ao ser clicado execute a calculadora do sistema operacional) basta usar a função *WinExec* ou *ShellExecute*.

O Lazarus, também, possui estas duas funções, mas somente para Windows. Basta declarar a unit *windows* na seção *uses*.

Mas, para executar um programa externo no Linux e no Windows usamos o *TProcess* que é multiplataforma.

Para usar esta classe é necessário declarar a unit *process* na seção *uses* do formulário.

De forma mais direta pode-se usar a função *ExecuteProcess*.

Exemplos:

```
ExecuteProcess('/usr/bin/gedit', '');
```

```
ExecuteProcess('C:\WINDOWS\notepad.exe', '');
```

Agora, veja um exemplo usando o *TProcess* junto com a diretiva *{IFDEF}* no evento *OnClick* de um botão:

```
procedure TForm1.Button1Click(Sender: TObject);  
var  
    processo: TProcess;  
begin  
    processo := TProcess.Create(nil);  
    {$IFDEF MSWINDOWS}  
    processo.CommandLine := 'notepad';  
    {$ENDIF}  
    {$IFDEF LINUX}
```

```
processo.CommandLine := 'gedit';
{$ENDIF}
processo.Execute;
processo.Free;
end;
```

O próximo exemplo mostra como pegar a saída de um comando no terminal Linux ou prompt do Windows e mostrar num *TMemo* (Memo1):

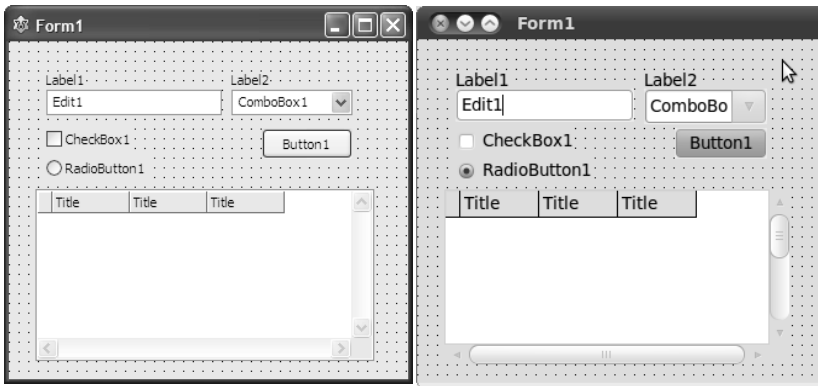
```
procedure TForm1.Button1Click(Sender: TObject);
var
    processo: TProcess;
begin
    processo := TProcess.Create(nil);
    processo.Options := [poUsePipes, poWaitOnExit];
    {$IFDEF MSWINDOWS}
    processo.CommandLine := 'dir \';
    {$ENDIF}
    {$IFDEF LINUX}
    processo.CommandLine := 'ls /';
    {$ENDIF}
    processo.Execute;
    Memo1.Lines.LoadFromStream(processo.Output);
    processo.Free;
end;
```

### Visual Multiplataforma:

Outro problema ao criar programas multiplataforma tem a ver com as diferenças nas interfaces visuais dos sistemas operacionais.

Por exemplo, veja a comparação de dois formulários, um foi criado no Ubuntu Linux e o outro no Windows XP:

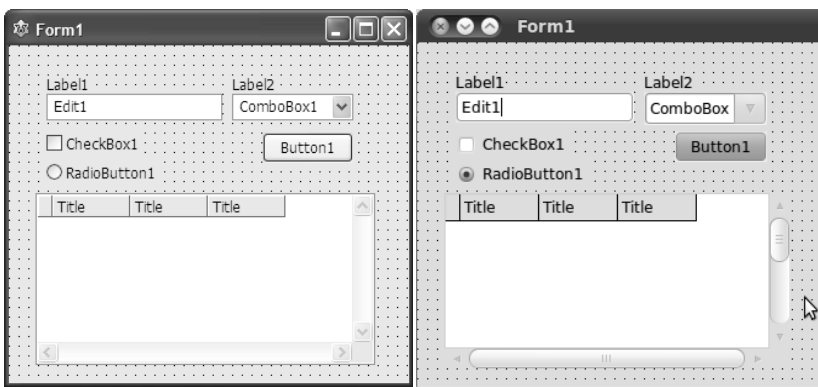




*Figura B.1 – Diferença Entre Fontes*

Há uma diferença significativa entre as fontes do Ubuntu e do Windows. No Ubuntu, a fonte padrão é a Sans de tamanho 10 e no Windows XP é a Tahoma de tamanho 8.

Para compatibilizar melhor o tamanho dos componentes e das fontes, antes de inserir componentes nos formulários de um projeto, mude a fonte destes no Ubuntu para Sans de tamanho 9 e no Windows XP para Tahoma de tamanho 9. Fica assim:



*Figura B.2 – Fontes Compatibilizadas*

### Arquivos de Texto como Banco de Dados

Vejamos como podemos usar um simples arquivo texto como se fosse um banco de dados. Desenvolveremos como exemplo uma agenda de contatos.

#### Componentes de Acesso a Arquivos de Texto:

O Lazarus possui dois componentes, o *TSdfDataSet* e o *TFixedFormatDataSet*, todos da paleta *DataAccess*, que possibilitam usar um arquivo de texto como se fosse uma tabela de banco de dados.

A diferença entre eles é que o primeiro usa um caractere (uma vírgula, por exemplo) para separar os campos da tabela e o outro usa campos de tamanho fixo. No exemplo que vamos desenvolver usaremos o *TSdfDataSet*.

#### Desenvolvendo o Projeto de uma Agenda:

Depois de pronta a nossa agenda vai se parecer com a mostrada abaixo:

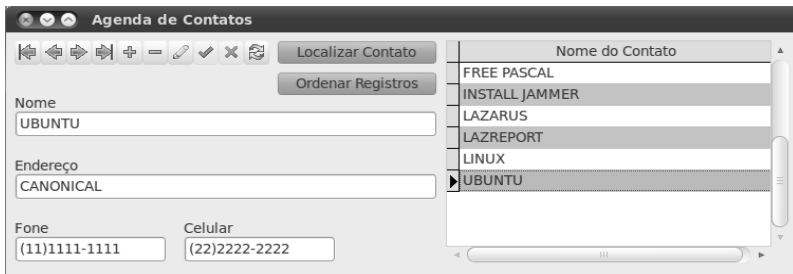


Figura C.1 – Agenda de Contatos

Crie uma pasta na pasta *livro\_projetos* com o nome *agenda\_texto*. No Lazarus, crie um novo projeto do tipo *Aplicação*. Salve na pasta *agenda\_texto* o projeto com o nome *agenda\_texto.lpi* e a unit com o nome *uprincipal.pas*.

Mude o nome do formulário para *fPrincipal* e a propriedade *Caption* para “*Agenda de Contatos*”. Também, mude as seguintes propriedades:

**Position** → *poDesktopCenter*

**Height** → 247

**Width** → 750

Agora adicione ao formulário dois *TButton* e quatro *TLabel* da paleta *Standard*, um *TDBNavigator*, quatro *TDBEdit* e um *TDBGrid*, todos da paleta “*Data Controls*”, e por fim um *TSdfDataSet* e um *TDataSource* da paleta “*Data Access*”.

Com exceção dos ícones do *SdfDataSet1* e do *Datasource1*, que podem ser deixados em qualquer posição no formulário, organize os outros componentes para ficarem como mostrado na *Figura C.1*.

Mude a propriedade *Caption* de cada *TLabel* para *Nome*, *Endereço*, *Fone* e *Celular*, respectivamente. Mude o *Caption* do *Button1* para “*Localizar Contato*” e do *Button2* para “*Ordenar Registros*”.

Altere a propriedade *Name* de cada *TDBEdit* para *edNome*, *edEndereco*, *edFone* e *edCelular*, respectivamente.

Selecione o componente *SdfDataSet1* e mude as seguintes propriedades:

**Delimiter** → ;

**FileMustExist** → *False*

Na propriedade **Schema** clique no botão de reticências ( ... ) e, no diálogo que surge, digite o texto abaixo:

**nome**  
**endereco**  
**fone**  
**celular**

Mude a propriedade **DataSource** dos componentes *edNome*, *edEndereco*, *edFone*, *edCelular*, *DBGrid1* e *DBNavigator1* para *Datasource1*.

Altere a propriedade **DataField** dos componentes *edNome*, *edEndereco*, *edFone* e *edCelular* para *nome*, *endereco*, *fone* e *celular*, respectivamente. Para cada alteração vai aparecer uma mensagem de erro, ignore-a. Esta mensagem aparece porque ainda não foi criado o arquivo de texto e as definições dos campos. O arquivo e as definições serão criadas quando o programa for executado pela primeira vez.

Modifique a propriedade **MaxLength** dos componentes *edNome*, *edEndereco*, *edFone* e *edCelular* para 100, 150, 13 e 13, respectivamente.

Selecione o *DBGrid1* e mude as propriedades:

**AlternateColor** → *clSkyBlue*  
**Anchor** > **akBottom** → *True*  
**Anchor** > **akRight** → *True*  
**ReadOnly** → *True*

Agora, dê um duplo clique no *DBGrid1* e no diálogo que surge clique em *Adicionar*, para adicionar uma coluna na grade. Na

janela *Inspetor de Objetos*, mude as seguintes propriedades da coluna criada:

**FieldName** → *nome*

**Title > Alignment** → *taCenter*

**Title > Caption** → *Nome do Contato*

**Width** → *300*

Selecione o *DBNavigator1*, e altere a propriedade *ShowHints* para *True*, na propriedade *Hints* clique no botão de reticências ( ... ) e, no diálogo que surge, digite o texto abaixo:

**Primeiro Contato**

**Contato Anterior**

**Próximo Contato**

**Último Contato**

**Novo Contato**

**Excluir Contato**

**Editar Contato**

**Salvar Alterações**

**Cancelar Alterações**

**Atualizar Dados**

Se necessário, altere a ordem do foco dos componentes ao teclar *TAB*. Para isso, clique com o botão direito do mouse em qualquer parte do formulário e no menu *pop-up* escolha “*Ordem de Tabulação...*” e use o diálogo que surgir.

Codifiquemos agora os quatro únicos eventos necessários para o funcionamento da agenda.

Crie o evento *OnCreate* para o formulário e digite o código abaixo entre o **begin** e o **end** da procedure:

```
// Cria o arquivo agenda na mesma pasta da
// aplicação
SdfDataSet1.FileName :=
    ExtractFilePath(Application.ExeName) + 'agenda';
SdfDataSet1.Open;
```

Já, no evento *OnClose* do formulário, digite o código seguinte entre o **begin** e o **end** da procedure:

```
SdfDataSet1.Close;
```

Agora, para o evento *OnClick* do Button1 digite o código em negrito nas posições indicadas:

```
procedure TfPrincipal.Button1Click(Sender:
TObject);
var
    consulta: String;
begin
    InputQuery('Localizar','Digite um nome para ' +
        'localizar:',consulta);
    if trim(consulta) = '' then exit;
    SdfDataSet1.First;
    while not(SdfDataSet1.EOF) do
        begin
            if Pos(UpperCase(consulta),UpperCase(
                SdfDataSet1.FieldByName(
                    'nome').AsString)) > 0 then break;
            SdfDataSet1.Next;
        end;
    if SdfDataSet1.EOF then ShowMessage('Contato ' +
        'não Localizado!');
end;
```

O código acima não é nenhum algoritmo elaborado de busca. Ele apenas faz uma busca sequencial por um nome. Para uma agenda com uns mil contatos, está de bom tamanho. É certo que a classe *TDataSet*, da qual a classe *TSdfDataSet*

descende, fornece a função *Locate* para uma busca mais eficiente de registros, mas ela não funciona para arquivos de texto comum, só para bancos de dados reais.

Continuando, para o evento *OnClick* do *Button2* digite o código em negrito nas posições indicadas:

```
procedure TfPrincipal.Button2Click(Sender:
TObject);
var
    ordem: TStringList;
begin
    SdfDataSet1.Close;
    //Estancia o objeto na memória
    ordem := TStringList.Create;
    ordem.LoadFromFile(ExtractFilePath(
        Application.ExeName) + 'agenda');
    ordem.Sorted := true;
    ordem.SaveToFile(ExtractFilePath(
        Application.ExeName) + 'agenda');
    //Destrói o objeto, liberando a memória
    ordem.Free;
    SdfDataSet1.Open;
end;
```

Como nós estamos trabalhando com um arquivo de texto comum e não com um banco de dados real, o código acima usa um objeto da classe *TStringList* como ponte para prover a capacidade de ordenação para nossa agenda. O texto do arquivo é carregado no objeto, depois é ordenado, através de um algoritmo de ordenação já implementado no objeto, e novamente salvo no disco para ser exibido por meio do *SdfDataSet1*.

A agenda está pronta. Execute, teste e divirta-se.

### Acessando Banco de Dados DBF

Bancos de dados DBF ainda são muito comuns e muitos programadores gostam de usá-los para pequenos projetos ou testes. Além disso, ainda há muitos projetos grandes legados que usam DBF.

O Lazarus tem um componente multiplataforma para criação e acesso a bancos DBF sem a necessidade do BDE (Borland DataBase Engine) que vem com o Delphi.

#### Tipos de Tabelas:

O Componente *TDbf* cria tabelas dos seguintes tipos:

Tipo 3 = dBase III Plus

Tipo 4 = dBase IV

Tipo 7 = Visual dBase VII

Tipo 25 = Visual FoxPro

#### Criando Tabelas Via Código:

Existem vários programas que podemos usar para criar tabelas DBF. Por exemplo, DataBase Desktop (vem com o Delphi), Excel, Calc do LibreOffice, etc. Mas, usando o *TDbf*, nós podemos criar as tabelas via código como mostrado no exemplo:

```
if not(FileExists('/home/usuario/clientes.dbf'))
then
begin
    Dbf1.FilePathFull := '/home/usuario/';
    Dbf1.TableLevel := 7;
```



```
Dbf1.TableName := 'clientes.dbf';
Dbf1.FieldDefs.Add('Id', ftAutoInc, 0, True);
Dbf1.FieldDefs.Add('Data', ftDateTime, 10, True);
Dbf1.FieldDefs.Add('Nome', ftString, 80, True);
Dbf1.FieldDefs.Add('Credito', ftFloat, 12, True);
Dbf1.FieldDefs.Items[3].Size := 10;
Dbf1.FieldDefs.Items[3].Precision := 2;
Dbf1.CreateTable;
end;
```

Neste código, *Dbf1* é um objeto da classe *TDbf*. Então, primeiro usa-se um *if* para testar se o arquivo *clientes.dbf* não existe no caminho especificado. Caso não exista será criado.

Na linha seguinte definimos o caminho do arquivo:

```
Dbf1.FilePathFull := '/home/usuario/';
```

Em seguida, definimos o tipo (ou nível) da tabela e, então, o nome da tabela que é o nome do arquivo.

```
Dbf1.TableLevel := 7;
Dbf1.TableName := 'clientes.dbf';
```

As próximas linhas de código definem as colunas ou campos da tabela:

```
Dbf1.FieldDefs.Add('Id', ftAutoInc, 0, True);
Dbf1.FieldDefs.Add('Data', ftDateTime, 10, True);
Dbf1.FieldDefs.Add('Nome', ftString, 80, True);
Dbf1.FieldDefs.Add('Credito', ftFloat, 12, True);
Dbf1.FieldDefs.Items[3].Size := 10;
Dbf1.FieldDefs.Items[3].Precision := 2;
```

Segue a sintaxe desta função:

```
Dbf1.FieldDefs.Add({nome_campo}, {tipo_campo},  
{tamanho_campo}, {se_é_obrigatório});
```

### Tipos de campos aceitos:

Abaixo temos os tipos de campos aceitos pelo *TDbf* :

**ftString**  
**ftSmallInt**  
**ftInteger**  
**ftWord**  
**ftBoolean**  
**ftFloat**  
**ftCurrency** (Nível de tabela 25)  
**ftBCD** (Nível de tabela 25)  
**ftDate**  
**ftDateTime**  
**ftBytes** (Nível de tabela 25)  
**ftAutoInc** (Nível de tabela 7 ou 25)  
**ftBlob**  
**ftMemo**  
**ftDBaseOle**  
**ftFixedChar**  
**ftWideString**  
**ftLargeInt**

Ao definirmos campos do tipo *ftFloat* e *ftCurrency*, devemos especificar o tamanho e a precisão (número de casas decimais). Se isso não for feito, o campo funcionará como se fosse do tipo inteiro.

Note, no código anteriormente mostrado, que o campo de nome *Credito* é o quarto a ser adicionado. Mas como a contagem dos campos começa com zero, o campo *Credito* recebe o índice 3 :

```
Dbf1.FieldDefs.Items[3].Size := 10;  
Dbf1.FieldDefs.Items[3].Precision := 2;
```

A primeira linha define o tamanho e a segunda a precisão do quarto campo adicionado.

Por fim, a linha abaixo cria a tabela (no caso, o arquivo *clientes.dbf*):

```
Dbf1.CreateTable;
```

### Trabalhando com Índices:

Em tabelas que conterão muitos dados, é importante criar índices para acelerar o acesso aos dados e as pesquisas usando filtros.

O TDbf trabalha com dois tipos de arquivos de índice. O tipo .NDX armazena um único índice e o tipo .MDX armazena vários índices (como se fosse uma coleção de arquivos .NDX).

Veremos apenas como usar arquivos de índice do tipo .MDX por serem mais práticos.

Veja o exemplo abaixo:

```
Dbf1.Exclusive := true;  
Dbf1.Open;  
Dbf1.AddIndex('indx_id', 'Id', [ixPrimary,  
                                ixUnique]);  
Dbf1.AddIndex('indx_nome', 'Nome',  
              [ixCaseInsensitive]);  
Dbf1.Close;  
Dbf1.Exclusive := false;
```

Colocamos a tabela em modo exclusivo e abrimos. E depois usamos a função *AddIndex* do *TDbf* para adicionar os índices. A sintaxe desta função é:

```
Dbf1.AddIndex({nome_indice},{nome_campo},  
              [{tipo_indice}]);
```

No exemplo anterior, usando esta função, primeiro adicionamos um índice do tipo chave primária única para o campo “Id”.

Depois, é adicionado, para o campo “Nome”, um índice não sensível a maiúsculas e minúsculas . Por fim, a tabela é fechada e tirada do modo exclusivo.

Automaticamente será criado um arquivo com extensão **.mdx** na mesma pasta e com o mesmo nome da tabela.

Quando a tabela é aberta ou fechada, o *TDbf* gerencia automaticamente o arquivo **.mdx** criado.

### Manutenção de Tabelas e Índices:

Quando excluímos registros de uma tabela DBF, estes não são excluídos fisicamente, mas apenas marcados para exclusão e por padrão ficam ocultos.

Para excluí-los definitivamente, execute o código abaixo:

```
Dbf1.Exclusive := True;  
Dbf1.Open;  
Dbf1.PackTable;  
Dbf1.Close;  
Dbf1.Exclusive := False;
```

Também, os índices precisam de manutenção.

Para executar esta tarefa, você pode usar o código seguinte:

```
Dbf1.Exclusive := True;  
Dbf1.Open;  
Dbf1.RegenerateIndexes;  
Dbf1.Close;  
Dbf1.Exclusive := False;
```

### Cadastro de Clientes com DBF:

Vamos por em prática tudo que consideramos até aqui sobre o componente *TDbf*. Criaremos um cadastro de clientes com foto.

Depois de pronto, o sistema de cadastro de clientes ficará como mostrado na figura abaixo:

*Figura D.1 – Cadastro de Clientes com TDbf*

Para começar, crie uma pasta na pasta *livro\_projetos* com o nome *cad\_cli\_dbf*. No Lazarus, crie um novo projeto do tipo *Aplicação*. Salve na pasta *cad\_cli\_dbf* o projeto com o nome *cad\_cli\_dbf.lpi* e a unit com o nome *u\_cad\_clientes.pas*.

Mude o nome do formulário para *fCadClientes* e o Caption para “*Cadastro de Clientes*”. Também, mude as seguintes propriedades:

**Position** → *poDesktopCenter*

**Height** → 366

**Width** → 750

**Constraints > MaxHeight** → 366

**Constraints > MaxWidth** → 750

**Constraints > MinHeight** → 366

**Constraints > MinWidth** → 750

**Font > Size** → 9

Agora adicione ao formulário um *TPanel*, da paleta *Standard*, e mude a propriedade *Align* para *alTop* e apague o conteúdo da propriedade *Caption*.

Adicione um segundo *TPanel* ao formulário e mude a propriedade *Align* para *alBottom* e apague o conteúdo da propriedade *Caption*.

Também, coloque no formulário um *TPageControl*, da paleta “*Common Controls*”, e mude a propriedade *Align* para *alClient*.

Clique com o botão direito do mouse sobre o componente *PageControl1* (*TPageControl*) e no menu que surgir clique na opção “Adicionar Página”. Repita para adicionar outra página.

Clique na página *TabSheet1* e mude a propriedade *Caption* para *Dados*. Clique em *TabSheet2* e mude o *Caption* para *Listagem*.

Insira no formulário, em qualquer posição, um componente *TDbf* e altere as seguintes propriedades:

**FilePath** → { clique no botão de reticências ( ... ), localize a pasta *cad\_cli\_dbf* e clique em Abrir }

**Filtered** → *True*

**FilterOptions** > **foCaseInsensitive** → *True*

**Name** → *tbClientes*

**TableLevel** → 7

**TableName** → *clientes.dbf*

Agora, insira no formulário, em qualquer posição, um componente *TDataSource* e mude as seguintes propriedades:

**DataSet** → *tbClientes*

**Name** → *dsClientes*

Novamente, insira no formulário, em qualquer posição, um componente *TOpenPictureDialog* e mude a propriedade **Name** para *opFoto* .

Usando a árvore hierárquica de componentes na parte superior da janela do *Inspetor de Objetos* (pressione *F11*), selecione o formulário *fCadClientes* . Na guia *Eventos*, crie o evento *onCreate* e digite o código abaixo em negrito:

```
procedure TfCadClientes.FormCreate(Sender:
TObject);
begin
    { $IFDEF LINUX }
        CurrencyString := 'R$';
        CurrencyFormat := 2;
        DecimalSeparator := ',';
        ThousandSeparator := '.';
        DateSeparator := '/';
        ShortDateFormat := 'dd/mm/yyyy';
    { $ENDIF }
    // O código acima só é necessário no Linux para
    // configurar a exibição de números formatados
    // para moeda e para exibição correta de datas.
```

```
if not(FileExists(ExtractFilePath(
  Application.ExeName) + 'clientes.dbf')) then
begin
  tbClientes.FilePathFull :=
    ExtractFilePath(Application.ExeName);
  tbClientes.TableLevel := 7;
  tbClientes.TableName := 'clientes.dbf';
  tbClientes.FieldDefs.Add('Id', ftAutoInc, 0,
    True);
  tbClientes.FieldDefs.Add('DataCad', ftDateTime,
    10, True);
  tbClientes.FieldDefs.Add('Nome', ftString, 100,
    True);
  tbClientes.FieldDefs.Add('Endereco', ftString,
    150, False);
  tbClientes.FieldDefs.Add('Bairro', ftString,
    80, False);
  tbClientes.FieldDefs.Add('Cidade', ftString,
    80, False);
  tbClientes.FieldDefs.Add('Estado', ftString, 2,
    False);
  tbClientes.FieldDefs.Add('CPF', ftString, 14,
    False);
  tbClientes.FieldDefs.Add('RG', ftString, 10,
    False);
  tbClientes.FieldDefs.Add('Fone', ftString, 13,
    False);
  tbClientes.FieldDefs.Add('Celular', ftString,
    13, False);
  tbClientes.FieldDefs.Add('Foto', ftBlob, 0,
    False);
  tbClientes.FieldDefs.Add('Credito', ftFloat,
    16, False);
  tbClientes.FieldDefs.Items[12].Size := 14;
  tbClientes.FieldDefs.Items[12].Precision := 2;
  tbClientes.CreateTable;
end;
if not(FileExists(ExtractFilePath(
  Application.ExeName) + 'clientes.mdx')) then
```



```
begin
    tbClientes.Exclusive := true;
    tbClientes.Open;
    tbClientes.AddIndex('indx_id', 'Id',
        [ixPrimary, ixUnique]);
    tbClientes.AddIndex('indx_nome', 'Nome',
        [ixCaseInsensitive]);
    tbClientes.AddIndex('indx_CPF', 'CPF',
        [ixUnique]);
    tbClientes.AddIndex('indx_RG', 'RG',
        [ixCaseInsensitive]);
    tbClientes.Close;
    tbClientes.Exclusive := false;
end;
tbClientes.Open;
end;
```

Execute o aplicativo (pressione *F9*) para criar os arquivos *clientes.dbf* e *clientes.mdx*, e feche-o. Agora podemos acessar o arquivo *dbf* em tempo de design.

Dê um duplo clique no ícone do componente *tbClientes* (*TDbf*). No diálogo que surgir, clique no botão com o sinal de mais ( **+** ) e no diálogo seguinte selecione todos os campos e clique no botão *Create*. O diálogo anterior ficará como mostrado abaixo:



Figura D.2 – Editor de Campos

No diálogo acima, selecione o campo ID e, no Inspetor de Objetos, deixe a propriedade `DisplayFormat` com o valor `000000` e a propriedade `Alignment` mude para *taCenter*. Selecione `DATA CAD` e mude `Alignment` para *taCenter*. Depois, selecione o campo `CREDITO` e mude a propriedade `Currency` para *True* (números em formato de moeda). Feche o diálogo.

No `Panel1` (*TPanel* superior) insira, na ordem, lado a lado, um *TLabel*, um *TEdit*, dois *TRadioButton*, um *TMaskEdit*, e um *TButton*.

Mude o `Caption` do *TLabel* (`Label1`) para *Filtrar Nome:*. Altere a propriedade `Name` do *TEdit* (`Edit1`) para *edFiltrar* e a propriedade `Width` para `310`.

No `RadioButton1` mude o `Caption` para *CPF*, `Checked` para *True* e o `Name` para *rbCPF*. E no `RadioButton2` mude o `Caption` para *RG* e o `Name` para *rbRG*.

Já no *TMaskEdit* (`MaskEdit1`), altere a propriedade `Name` para *edLocalizar*, `Width` para `117` e `EditMask` para `999.999.999-99;1;_`. E no *TButton* (`Button1`) mude a propriedade `Name` para *btLocalizar* e o `Caption` para *Localizar*.

Organize os componentes como mostrado abaixo:



*Figura D.3 – Painel Superior*

Agora, na página *Dados* (`TabSheet1`) do `PageControl1`, insira doze *TLabel*, onze *TDBEdit*, um *TDBComboBox*. Insira, também, um *TGroupBox* (mude as propriedades: `Caption` para *Foto*, `Height` para `209` e `Width` para `147`) e dentro deste um *TPanel* (propriedades: `Caption` para *SEM FOTO*, `Color` para *clWhite*, `Height` para `150` e `Width` para `130`). Abaixo do *TPanel* do *TGroupBox*, insira dois *TButton*.

Dentro do *TPanel* do *TGroupBox* insira um *TDBImage* (propriedades: *Align* para *alCliente*, *DataSource* para *dsClientes*, *DataField* para *FOTO*, *Name* para *imFoto* e *Stretch* para *True*) .

Organize os componentes e mude a propriedade *Caption* dos *TLabel's* e *TButton's* para ficarem como mostrado na figura seguinte:



*Figura D.4 – Página de Dados*

Mude a propriedade *DataSource* de todos os *TDBEdit's* e *TDBComboBox* para *dsClientes*.

**DICA:** Para selecionar mais de um componente, a fim de mudar propriedades em comum, clique no primeiro componente, segure a tecla SHIFT e clique nos componentes seguintes.

Em seguida, da esquerda para direita e de cima para baixo, mude a propriedade *Name* dos *TDBEdit's* para: *edCodigo*, *edDataCad*, *edNome*, *edEndereco*, *edBairro*, *edCidade*, *edCPF*, *edRG*, *edFone*, *edCelular* e *edCredito*. Altere a propriedade *Name* do *TDBComboBox* para *cbEstado*, a do botão *Inserir* para *btInserir* e a do botão *Limpar* para *btLimpar*.

Configure a propriedade `DataField` dos componentes `edCodigo`, `edDataCad`, `edNome`, `edEndereco`, `edBairro`, `edCidade`, `cbEstado`, `edCPF`, `edRG`, `edFone`, `edCelular` e `edCredito` para *ID*, *DATA CAD*, *NOME*, *ENDERECO*, *BAIRRO*, *CIDADE*, *ESTADO*, *CPF*, *RG*, *FONE*, *CELULAR* e *CREDITO*, respectivamente.

Nos *TDBEdit*'s `edCodigo` e `edDataCad`, também, mude as seguintes propriedades: `Color` para *clSilver*, `ReadOnly` para *True* e `TabStop` para *False*.

Selecione o componente `cbEstado` (*TDBComboBox*), mude a propriedade `Style` para *csDropDownList*. Agora, clique ao lado da propriedade `Items` e clique no botão de reticências ( ... ), no diálogo que surgir digite, uma por linha, as siglas dos estados do Brasil e clique em *OK*, como mostrado na figura seguinte . Se estiver usando o Linux, para melhorar o visual, também, mude as propriedades `AutoSize` para *False*, `Height` para 25 e `Font > Size` para 8.

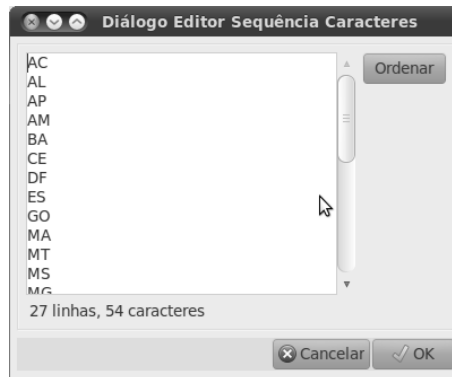


Figura D.5 – Sigla dos Estados para o `cbEstado`

No `PageControl1`, clique na guia *Listagem* e dentro desta insira um *TDBGrid* e mude as seguintes propriedades:

**Align** → *alClient*

**AlternateColor** → *clSkyBlue*

**DataSource** → *dsClientes*

**Name** → *gdDados*

**Options > dgRowSelect** → *True*

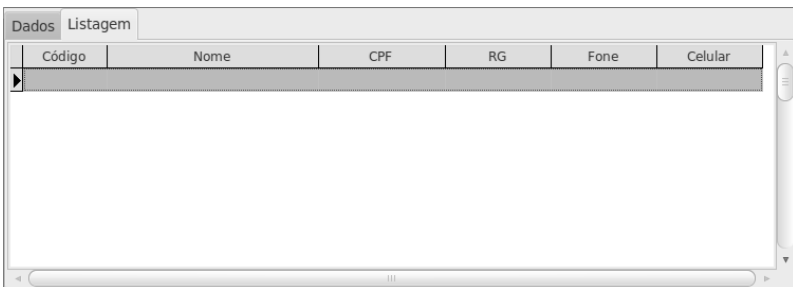
**ReadOnly** → *True*

Agora, dê um duplo clique no *gdDados (TDBGrid)* e no diálogo que surge, adicione seis colunas na grade usando o botão Adicionar. No *Inspetor de Objetos*, altere as propriedades de cada coluna criada:

Coluna	Propriedade	Valor
0	Alignment	taCenter
	DisplayFormat	000000
	FieldName	ID
	Title > Alignment	taCenter
	Title > Caption	Código
	Width	80
1	FieldName	NOME
	Title > Alignment	taCenter
	Title > Caption	Nome
	Width	200
2	Alignment	taCenter
	FieldName	CPF
	Title > Alignment	taCenter
	Title > Caption	CPF
	Width	120

Coluna	Propriedade	Valor
3	Alignment	taCenter
	FieldName	RG
	Title > Alignment	taCenter
	Title > Caption	RG
	Width	100
4	Alignment	taCenter
	FieldName	FONE
	Title > Alignment	taCenter
	Title > Caption	Fone
	Width	100
5	Alignment	taCenter
	FieldName	CELULAR
	Title > Alignment	taCenter
	Title > Caption	Celular
	Width	100

A grade ficará como mostrado na figura seguinte:



*Figura D.6 – Grade com as colunas configuradas*

Clique na guia *Dados*.

Agora, insira no painel inferior (Panel2) um *TDBNavigator* e dois *TButton*. Organize os componentes e altere a propriedade *Caption* dos *TButton* como mostrado abaixo:



*Figura D.7 – Painel Inferior*

Configure a propriedade *Name* dos *TButton* para *btCompactar* e *btReconstruir*, respectivamente.

Selecione o *DBNavigator1* e mude a propriedade *DataSource* para *dsClientes*.

Se necessário, altere a ordem do foco dos componentes ao teclar *TAB*. Para isso, clique com o botão direito do mouse em qualquer parte do formulário e no menu *pop-up* escolha “*Ordem de Tabulação...*” e use o diálogo que surgir.

Agora, vamos criar os eventos.

Crie o evento *OnShow* do formulário *fCadClientes* e digite:

```
edFiltrar.SetFocus;
```

Crie o evento *OnClose* do formulário *fCadClientes* e digite:

```
tbClientes.Close;
```

No evento *OnClick* do *rbCPF*, digite:

```
edLocalizar.EditMask := '999.999.999-99;1;_';
```

No evento *OnClick* do *rbRG*, digite:

```
edLocalizar.EditMask := ''; // duas aspas simples  
edLocalizar.MaxLength := 10;
```

No evento BeforePost do tbClientes, digite o código em negrito:

```
procedure TfCadClientes.tbClientesBeforePost  
(DataSet: TDataSet);  
begin  
    if Trim(edNome.Text) = '' then begin  
        ShowMessage('Digite o nome do cliente!');  
        Abort; //Não use Exit  
    end;  
end;
```

No evento OnNewRecord do tbClientes, digite o código em negrito:

```
procedure TfCadClientes.tbClientesNewRecord  
(DataSet: TDataSet);  
begin  
    tbClientes.FieldByName('DataCad').Value := date;  
    PageControl1.PageIndex := 0;  
    edNome.SetFocus;  
end;
```

No evento OnStateChange do dsClientes, digite o código em negrito:

```
procedure TfCadClientes.dsClientesStateChange  
(Sender: TObject);  
begin  
    if tbClientes.State in [dsEdit, dsInsert] then  
        begin  
            btInserir.Enabled := True;  
            btLimpar.Enabled := True;  
        end else begin  
            btInserir.Enabled := False;  
            btLimpar.Enabled := False;  
        end;
```



```
end;  
end;
```

No evento *OnChange* do *edFiltrar*, digite o código em negrito:

```
procedure TfCadClientes.edFiltrarChange(Sender:  
TObject);  
begin  
    if trim(edFiltrar.Text) <> '' then  
        tbClientes.Filter := 'Nome = ' +  
            QuotedStr(edFiltrar.Text + '*')  
        else  
            tbClientes.Filter := '';  
end;
```

No evento *OnClick* do *btInserir*, digite o código em negrito:

```
procedure TfCadClientes.btInserirClick(Sender:  
TObject);  
begin  
    if opFoto.Execute then begin  
        try  
            imFoto.Picture.LoadFromFile(opFoto.FileName);  
        except  
            ShowMessage('Arquivo de Imagem Inválido!');  
        end;  
    end;  
end;
```

No evento *OnClick* do *btCompactar*, digite o código em negrito:

```
procedure TfCadClientes.btCompactarClick(Sender:  
TObject);  
begin  
    tbClientes.Close;  
    tbClientes.Exclusive := True;  
    tbClientes.Open;  
    tbClientes.PackTable;
```

```
tbClientes.Close;  
tbClientes.Exclusive := False;  
tbClientes.Open;  
end;
```

No evento *OnClick* do btLimpar, digite o código em negrito:

```
procedure TfCadClientes.btLimparClick(Sender:  
TObject);  
begin  
    if Application.MessageBox('Deseja mesmo ' +  
        'excluir a foto do cliente?', 'Confirmação',6) <>  
        6 then exit;  
    imFoto.Picture.Clear;  
end;
```

No evento *OnClick* do btLocalizar, digite o código em negrito:

```
procedure TfCadClientes.btLocalizarClick(Sender:  
TObject);  
begin  
    if rbCPF.Checked then  
        tbClientes.Locate('CPF',edLocalizar.Text,[])  
    else  
        tbClientes.Locate('RG',edLocalizar.Text,[]);  
end;
```

No evento *OnClick* do btReconstruir, digite o código em negrito:

```
procedure TfCadClientes.btReconstruirClick(Sender:  
TObject);  
begin  
    tbClientes.Close;  
    tbClientes.Exclusive := True;  
    tbClientes.Open;  
    tbClientes.RegenerateIndexes;  
    tbClientes.Close;  
    tbClientes.Exclusive := False;
```

```
tbClientes.Open;  
end;
```

No evento OnKeyPress do edCredito, digite o código em negrito:

```
procedure TfCadClientes.edCreditoKeyPress(Sender:  
TObject; var Key: char);  
{ $IFDEF LINUX}  
    var  
        posicao: Integer;  
{ $ENDIF}  
begin  
    { $IFDEF LINUX}  
        if key = ',' then begin  
            posicao := TDBEdit(Sender).SelStart;  
            TDBEdit(Sender).Text :=  
                Copy(TDBEdit(Sender).Text, 0, posicao) +  
                ',' + Copy(TDBEdit(Sender).Text, posicao +  
                    1, Length(TDBEdit(Sender).Text));  
            TDBEdit(Sender).SelStart := posicao + 1;  
        end;  
    { $ENDIF}  
end;
```

**LEMBRETE:** O código acima só é necessário no Linux. Acontece que o *TDBEdit* (no caso, *edCredito*), no Linux, não aceita a digitação da virgula para casas decimais. Não sei se isto é um bug do Lazarus, mas esta é uma forma de resolver. Talvez, em versões futuras do Lazarus, este código não seja mais necessário. Observe que o código é genérico, só precisamos implementar uma vez na unit e ligar o OnKeyPress que tem a implementação aos eventos OnKeyPress dos outros TDBEdit's que precisam do mesmo tratamento.

O programa está finalizado. Estude o código, execute e teste.

### Instalando Servidores de Banco de Dados

Consideraremos como instalar e configurar os servidores de banco de dados mais usados e os aplicativos gerenciadores destes, tanto no Ubuntu Linux como no Windows.

#### Instalando o SQLite3 no Ubuntu e Derivados:

O SQLite3 já encontra-se instalado por padrão no Ubuntu, mas caso tenha sido desinstalado digite o comando abaixo num terminal:

```
sudo apt-get install libsqlite3-0
```

Precisamos criar um link simbólico para a biblioteca do SQLite3. Isso é necessário para que a biblioteca seja encontrada pelos componentes de conexão das paletas SQLdb e ZEOS. Então, digite o comando abaixo num terminal:

```
sudo ln -s /usr/lib/libsqlite3.so.0.8.6 /usr/lib/libsqlite3.so
```

Se o comando acima não funcionar, tente o seguinte:

```
sudo ln -s /usr/lib/i386-linux-gnu/libsqlite3.so.0.8.6 /usr/lib/libsqlite3.so
```

Um ótimo aplicativo gráfico para gerenciar bancos SQLite é o SQLite Studio. Para baixá-lo digite num terminal o comando seguinte:

```
wget http://sqlitestudio.one.pl/files/free/stable/linux32/sqlitestudio-2.0.12.bin ~/sqlitestudio
```

Dê permissão de execução ao arquivo com o comando abaixo:

```
chmod +x ~/sqlitestudio/sqlitestudio-2.0.12.bin
```

Para facilitar o acesso ao aplicativo crie um link simbólico com o comando a seguir:

```
sudo ln -s ~/sqlitestudio/sqlitestudio-2.0.12.bin /bin/sqlitestudio
```

Para criar um menu no Ubuntu para o SQLite Studio, clique no menu *Sistema > Preferências > Menu principal*. No diálogo que abrir, clique em *Desenvolvimento* (a esquerda) e clique no botão *Novo item*. Vai surgir o diálogo mostrado abaixo:



Figura E.1 – Diálogo para Criar Item de Menu

Em **Nome:** digite *SQLite Studio* e em **Comando:** digite *sqlitestudio*. Clique em **OK**.

Acesse o SQLite Studio através do menu do Ubuntu: *“Aplicativos > Desenvolvimento > SQLite Studio”*.

### Instalando o SQLite3 no Windows:

Baixe o SQLite3 para Windows do seguinte link: <http://www.sqlite.org/sqlite-shell-win32-x86-3070600.zip>

Descompacte o arquivo e copie o arquivo *sqlite3.dll* para pasta system32 do Windows.

Baixe o SQLite Studio para Windows do seguinte link: <http://sqlitestudio.one.pl/files/free/stable/windows/sqlitestudio-2.0.12.exe>

### Instalando o Firebird 2.1 ou 2.5 no Ubuntu e Derivados:

Para instalar o servidor Firebird 2.5 digite o seguinte comando num terminal:

```
sudo apt-get install firebird2.5-super
```

Se não estiver disponível a versão 2.5, instale a 2.1 com o comando seguinte:

```
sudo apt-get install firebird2.1-super
```

Para configurar o início automático do servidor e a senha do administrador SYSDBA digite num terminal o comando de acordo com a versão:

Para o Firebird 2.5:

```
sudo dpkg-reconfigure firebird2.5-super
```

Para o Firebird 2.1:

```
sudo dpkg-reconfigure firebird2.1-super
```

Siga os passos apresentados no terminal.

Por fim, precisamos criar o link simbólico necessário. Assim, digite num terminal:

```
sudo ln -s /usr/lib/libfbclient.so.2 /usr/lib/libfbclient.so
```

O comando seguinte instala o aplicativo FlameRobin para gerenciar o Firebird:

```
sudo apt-get install flamerobin
```

Devido ao sistema de permissões do Linux, precisamos dar permissões para o Firebird salvar os bancos em uma determinada pasta. Então, digite num terminal os comandos seguintes:

```
mkdir ~/bancos
```

```
sudo chown firebird:firebird -R ~/bancos
```

O primeiro comando cria uma pasta chama *bancos* dentro de sua *Pasta pessoal* e o segundo muda as permissões.

### Instalando o Firebird 2.1 ou 2.5 no Windows:

Baixe o Firebird 2.1 ou 2.5 para o Windows no seguinte link:  
<http://www.firebirdsql.org/index.php?op=files>

Execute o instalador e siga os passos até surgir a tela a seguir:

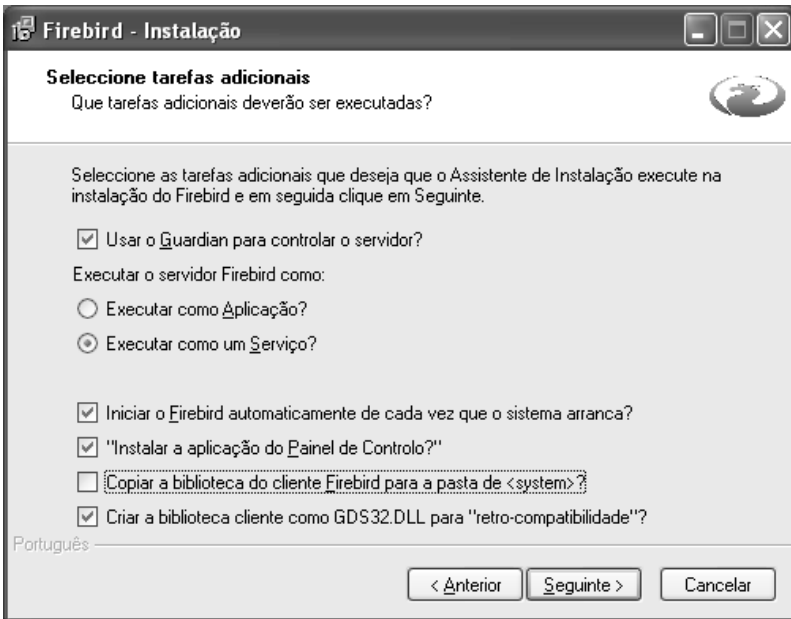


Figura E.2 – Tela para Selecionar Tarefas Adicionais

Marque a opção “Copiar a biblioteca do cliente Firebird para a pasta de <system>?” e prossiga com a instalação até o final.

Baixe o FlameRobin para Windows para gerenciar o Firebird no seguinte link: <http://www.flamerobin.org/>

### Instalando o MySQL 5.1 no Ubuntu e Derivados:

Instale o MySQL 5.1 executando o comando abaixo num terminal:

```
sudo apt-get install mysql-server-5.1
```

Durante a configuração do pacote, surgirá uma tela para configurar a senha do usuário *root* do MySQL.



Se necessitar reconfigurar a senha do usuário *root* do MySQL, digite o comando abaixo num terminal:

```
sudo dpkg-reconfigure mysql-server-5.1
```

Depois de configurar a senha, aguarde o servidor reiniciar.

Para instalar os aplicativos gráficos de gerenciamento do servidor execute o comando:

```
sudo apt-get install mysql-admin mysql-query-browser
```

Após a instalação crie um link simbólico com o comando abaixo:

```
sudo ln -s /usr/lib/libmysqlclient.so.16 /usr/lib/libmysqlclient.so
```

### Instalando o MySQL 5.1 no Windows:

Baixe o instalador do MySQL 5.1 para Windows usando o link:  
<http://dev.mysql.com/downloads/mysql/5.1.html>

Execute o instalador e execute os passos apresentados, mantendo as opções padrões.

Após a instalação, o instalador prossegue com a configuração do servidor. Siga os passos sem mudar as opções previamente selecionadas até chegar na tela mostrada a seguir:

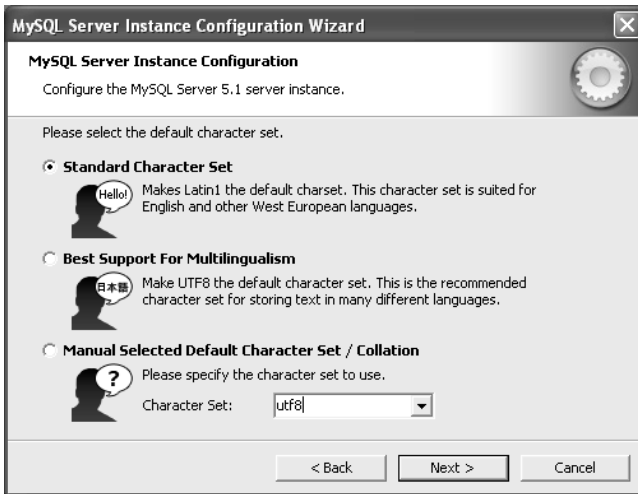


Figura E.3 – Tela para Selecionar a Tabela de Caracteres

Marque **“Manual Selected Default Character Set / Collation”** e em **Character Set:** escolha *utf8* . prossiga até surgir a tela abaixo:



Figura E.4 – Tela para Configurar a Senha do Administrador

Digite a senha do usuário *root* do servidor MySQL e prossiga com a configuração até o final.

Localize o arquivo *libmySQL.dll* no caminho “C:\Arquivos de programas\MySQL\MySQL Server 5.1\bin” e copie-o para a pasta *system32* do Windows.

Baixe os aplicativos de gerenciamento do MySQL no seguinte link: <http://mysql.cce.usp.br/Downloads/MySQLGUITools/mysql-gui-tools-5.0-r17-win32.msi>

Execute o instalador.

### Instalando o PostgreSQL 8.4 no Ubuntu e Derivados:

Podemos instalar o PostgreSQL 8.4 usando um instalador idêntico ao da versão para Windows ou através do repositório de pacotes do Ubuntu.

Vejamos primeiro como instalar usando o repositório. Abra um terminal e digite:

```
sudo apt-get install postgresql-8.4 pgadmin3
```

O comando acima instala o servidor PostgreSQL 8.4 juntamente com o aplicativo de administração pgAdmin III.

O usuário administrador padrão do PostgreSQL é o *postgres* . Precisamos configurar a senha deste. Para isso digite num terminal o seguinte comando:

```
sudo -u postgres psql template1
```

O comando acima loga no servidor PostgreSQL usando sua conta de administrador no Linux. Agora você está executando o

*psql* que é um aplicativo em linha de comando para administrar o servidor.

Agora digite o seguinte comando SQL:

```
ALTER USER postgres WITH PASSWORD 'sua_senha' ;
```

Troque '**sua\_senha**' pela nova senha entre aspas simples.

Feche o aplicativo *psql* digitando \q e pressione ENTER. A instalação e configuração está completa.

Agora vamos considerar como instalar o PostgreSQL 8.4 usando um instalador igual ao do Windows. Abra um terminal e digite o seguinte comando para baixar o instalador:

Para Linux 32 bits:

```
wget http://get.enterprisedb.com/postgresql/postgresql-8.4.8-1-linux.bin ~
```

Para Linux 64 bits:

```
wget http://get.enterprisedb.com/postgresql/postgresql-8.4.8-1-linux-x64.bin ~
```

Dê permissão de execução ao arquivo com o comando abaixo:

Para Linux 32 bits:

```
chmod +x ~/postgresql-8.4.8-1-linux.bin
```

Para Linux 64 bits:

```
chmod +x ~/postgresql-8.4.8-1-linux-x64.bin
```

Para executar o instalador digite:

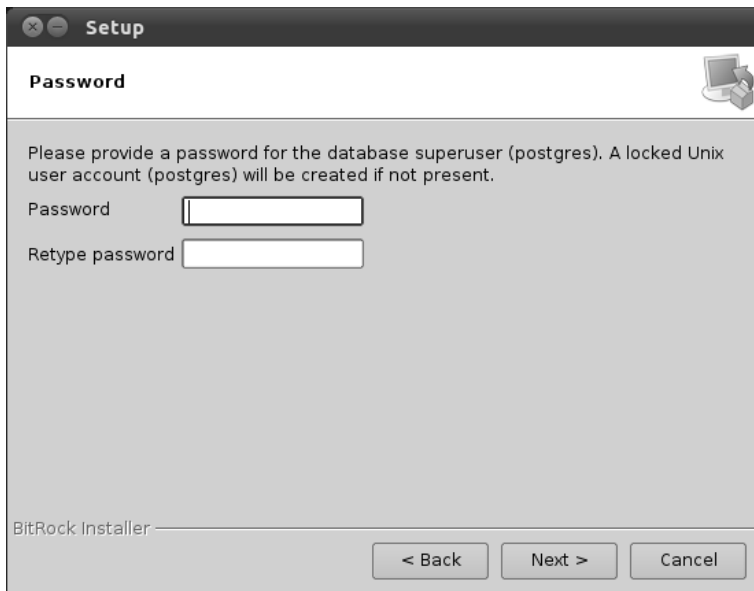
Para Linux 32 bits:

```
sudo ./postgresql-8.4.8-1-linux.bin
```

Para Linux 64 bits:

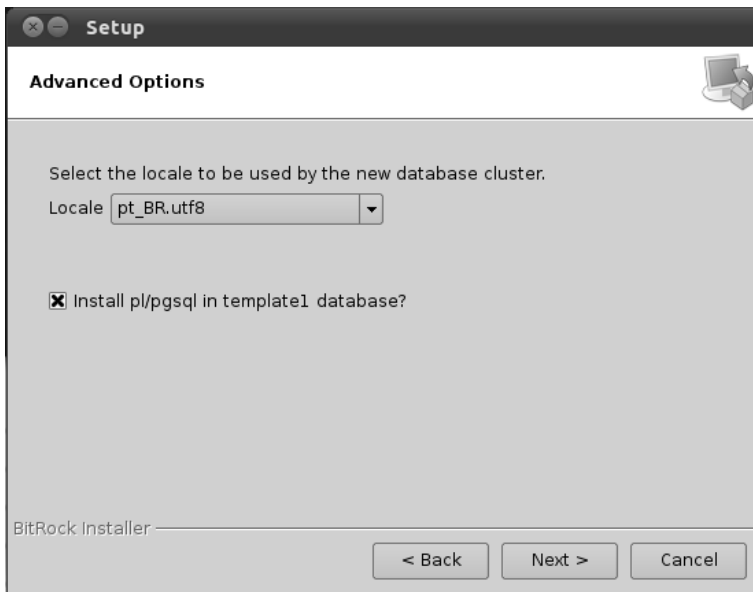
```
sudo ./postgresql-8.4.8-1-linux-x64.bin
```

Siga os passos do instalador até surgir a tela abaixo:



*Figura E.5 – Tela para Configurar a Senha do Administrador*

Configure a senha do administrador *postgres* e prossiga até a tela:



*Figura E.6 – Tela para Selecionar a Tabela de Caracteres*

Na tela anterior, escolha em **Locale** a opção *pt\_BR.utf8* e prossiga com a instalação até o final. Depois da instalação ser concluída, é mostrada uma tela com aplicativos que você pode baixar opcionalmente.

É preciso instalar a biblioteca *libpq* para usar o SQLdb ou ZEOS. Assim, digite os dois comandos a seguir num terminal:

```
sudo apt-get install libpq5
```

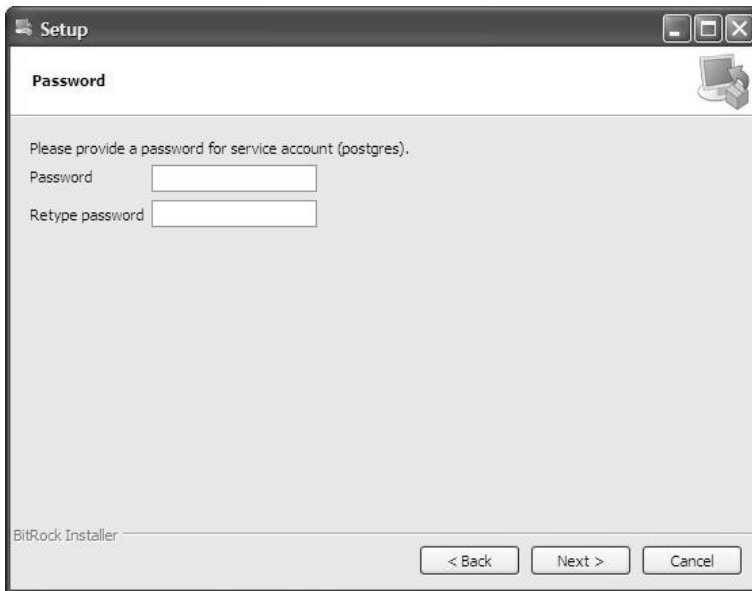
```
sudo ln -s /usr/lib/libpq.so.5.2 /usr/lib/libpq  
.so
```

Usando este instalador, já é instalado o aplicativo pgAdmin III .

### Instalando o PostgreSQL 8.4 no Windows:

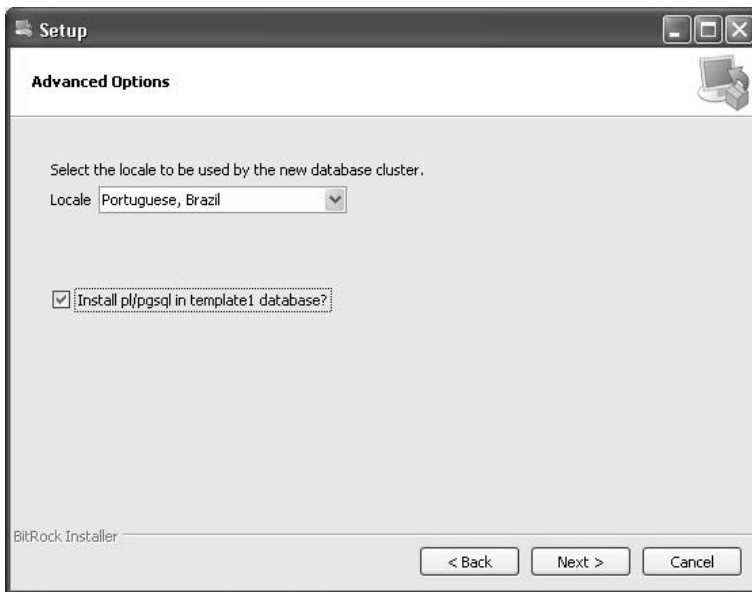
Baixe o instalador do PostgreSQL 8.4 para Windows do seguinte link: <http://get.enterprisedb.com/postgresql/postgresql-8.4.8-1-windows.exe>

Execute o instalador e siga os passos até a tela abaixo:



*Figura E.7 – Tela para Configurar a Senha do Administrador*

Configure a senha do administrador *postgres* e prossiga até ser mostrada a tela seguinte:



*Figura E.8 – Tela para Selecionar a Tabela de Caracteres*

Na tela acima, escolha em **Locale** a opção *Portuguese, Brasil* e prossiga com a instalação até o final.

Depois da instalação ser concluída, é mostrada uma tela com aplicativos que você pode baixar opcionalmente.

Usando este instalador, já é instalado o pgAdmin III para gerenciar o servidor e menus para iniciar ou parar o servidor.

Copie todos os arquivos com extensão **.dll** do caminho *C:\Arquivos de programas\PostgreSQL\8.4\bin* para a pasta *system32* do Windows.



## Links Úteis

Segue uma lista de links úteis sobre Free Pascal e Lazarus:

**<http://www.lazarus.freepascal.org/>** – Site oficial do Lazarus.

**<http://lazarusbrasil.org/>** – Site da comunidade brasileira de Lazarus e Free Pascal.

**[http://wiki.lazarus.freepascal.org/Lazarus\\_Documentation/pt](http://wiki.lazarus.freepascal.org/Lazarus_Documentation/pt)** – Site com documentação do Lazarus em Português.

**<http://www.lazarus-components.org/>** – Repositório de códigos e componentes para o Lazarus.

**<https://groups.google.com/group/lazarus-br?hl=pt>** – Grupo brasileiro de usuários do Lazarus.

**<http://www.lazarus.freepascal.com.br/>** – Fórum brasileiro sobre Free Pascal e Lazarus.

**<http://www.lazarusportugal.org/doku.php>** – Comunidade de utilizadores do Lazarus, Delphi e Pascal em Portugal.

**<http://silvioprogram.com.br/>** – Site com artigos, dicas e componentes para o Lazarus.

**<http://professorcarlos.blogspot.com/>** – Site com muitos artigos sobre o Lazarus, do iniciante ao avançado.

**<http://dicas4lazarus.blogspot.com/>** – Outro site com artigos, dicas e componentes para o Lazarus.

**<http://lazarus-cgi.co.cc/>** – Site sobre desenvolvimento de aplicativos web usando Lazarus.

## **Bibliografia**

Diversos Artigos – Autor: Jean Patrick – URL:  
<http://www.jpsoft.com.br>.

Diversos Artigos – Autor: Carlos Araújo – URL:  
<http://professorcarlos.blogspot.com/>.

Diversos Artigos – Autor: Sílvio Clécio – URL:  
<http://silvioprogram.blogspot.com/>.

Documentação Oficial do Lazarus – Autores: Comunidade de  
Usuários e Desenvolvedores do Lazarus – URL:  
[http://wiki.lazarus.freepascal.org/Lazarus\\_Documentation/pt](http://wiki.lazarus.freepascal.org/Lazarus_Documentation/pt).

Compilando o FPC/Lazarus a partir dos fontes do SVN  
(Windows) – Autor: Marcos Douglas – URL:  
<http://groups.google.com/group/lazarus-br/web/compilando-o-fpc-lazarus-a-partir-dos-fontes-do-svn-windows?version=5>.

Tutorial Zeos Especial – Autor: Michael – URL:  
[http://www.apostilasneweb.com.br/component/docman/doc\\_download/2020-tutorialzeosespecial-pt-br](http://www.apostilasneweb.com.br/component/docman/doc_download/2020-tutorialzeosespecial-pt-br).