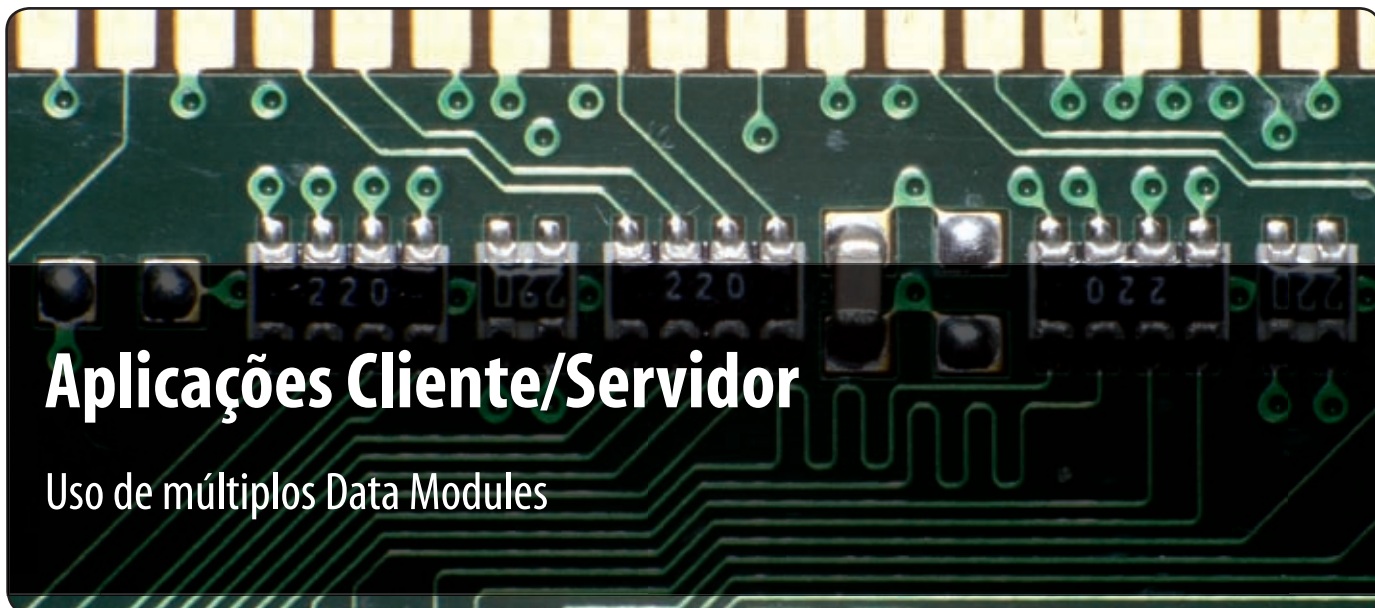


## Nesta seção você encontra artigos para iniciantes na linguagem Delphi



### Aplicações Cliente/Servidor

#### Uso de múltiplos Data Modules

No passado, principalmente quando se programava para MS-DOS, a quantidade de memória RAM disponível era irrisória, o que obrigava os programadores muitas vezes a usar técnicas complexas para economizar alguns poucos bytes de memória de leitura.

Atualmente os computadores possuem grande quantidade de memória RAM disponível, mas isto não é motivo para que se aloque memória desnecessariamente, já que nem sempre a aplicação rodará localmente. Por exemplo: uma aplicação instalada num servidor pode ser acessada simultaneamente por diversos usuários via Windows Terminal Services (WTS). Estes usuários podem estar em qualquer lugar do mundo, mas rodarão a aplicação diretamente no servidor. Se o sistema e o número de usuários forem grandes, e o desenvolvedor abusar da alocação de memória, pode haver problema de falta de memória mesmo em computadores

com mais de 1 GB de RAM.

Este artigo mostra passo a passo uma técnica simples e eficaz para criar *Data Modules* em tempo de execução em aplicações Cliente/Servidor, o que torna a inicialização da aplicação mais rápida e ao mesmo tempo consome menos memória RAM.

#### Entendendo a arquitetura

*Data Module* nada mais é do que um repositório de componentes não visuais assim como o próprio nome sugere. Sua finalidade é alocar componentes de acesso a dados a fim de separar regras de negócio da interface. Com isso tem-se uma maior facilidade na manutenção e/ou em uma possível migração de Cliente/Servidor (duas camadas) para *n-tiers* (três ou mais camadas).

A **Figura 1** ilustra como fica a arquitetura de uma aplicação usando *Data Modules*, onde os componentes de acesso a dados e as regras de negócios estariam neles,



**Pablo Tøndolo de Vargas**

(pablodv@gmail.com)

é Acadêmico do Curso de Sistemas de Informação da Universidade Luterana do Brasil (ULBRA). Atua na área de desenvolvimento em Delphi para a empresa dotBR Soluções em TI.



**Fernando Sarturi Prass**

(fernando@dotbr.com.br)

é Mestre em Ciência da Computação pela UFSC. Professor da Universidade Luterana do Brasil (ULBRA) nos campus de Santa Maria e Cachoeira do Sul. Sócio-diretor da dotBR Soluções em TI ([www.dotbr.com.br](http://www.dotbr.com.br)), empresa que presta serviços de desenvolvimento de sistemas e de consultoria em Bancos de Dados e Metodologias de Desenvolvimento.

ficando somente na parte do formulário os componentes visuais e os que não acessam dados.

## Criando a Aplicação

Essa aplicação será constituída de um formulário principal e 4 *Data Modules*. Portanto crie uma nova aplicação utilizando o menu *File\New>Application*. Salve o projeto na pasta de sua preferência. Para a *Unit* dê o nome de “uPrincipal.pas” e para o projeto dê o nome de “Employee.dpr”, por último altere a propriedade *Name* do formulário para “frmPrincipal”.

## Configurando os Data Modules

Adicione um *Data Module* usando o menu *File\New>Data Module* e salve-o com nome de “uDMConexao”, altere a propriedade *Name* para “dmConexao”. Coloque o dmConexao para que seja criado antes do frmPrincipal. Para isso acesse *Project>Options* ou use o atalho *Ctrl + Shift + F11* e na aba *Forms* mude a ordem para que o dmConexao esteja primeiro que o frmPrincipal.

Por padrão, todos os formulários e *Data Modules* são criados automaticamente quando o programa é inicializado, para alterar isso vá em *Tools\Environment Options>Designer* e desmarque a opção *Auto create forms & data modules*. Outra forma de fazer a mesma coisa é a partir da janela *Project>Options*, após criar o

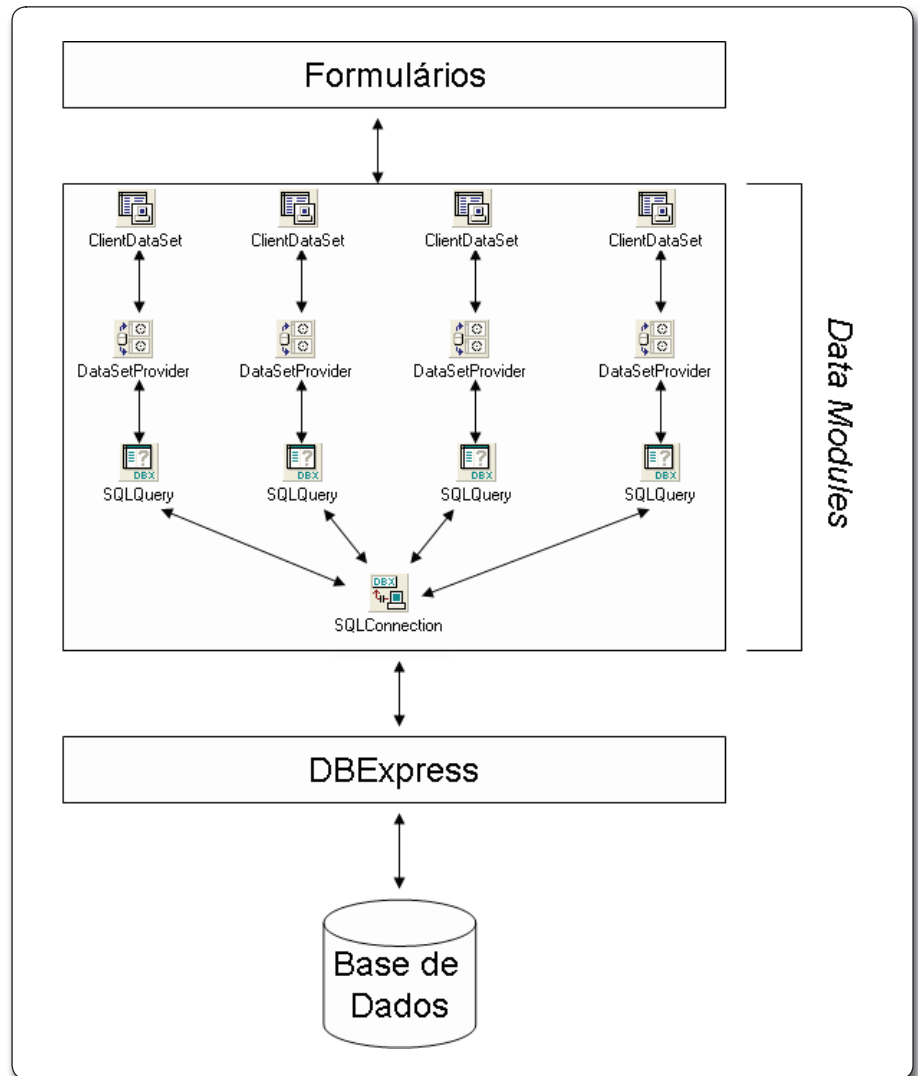


Figura 1. Arquitetura de uma aplicação com Data Module

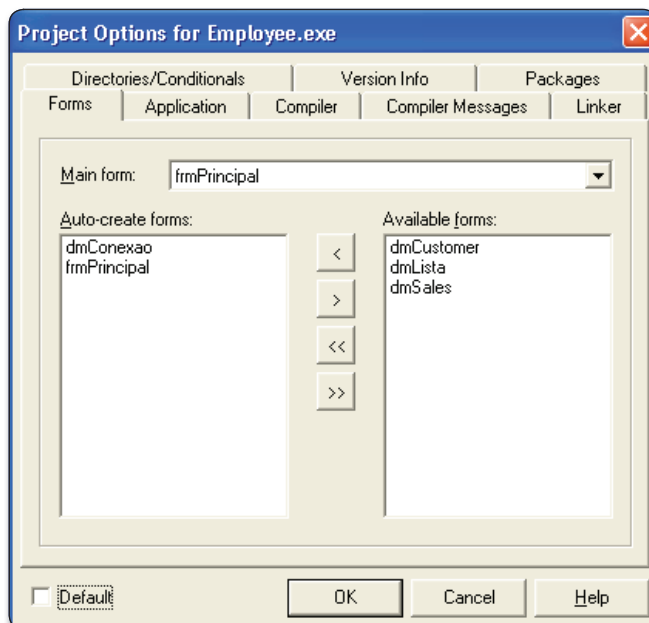


Figura 2. Opções do projeto com todos os Data Modules criados

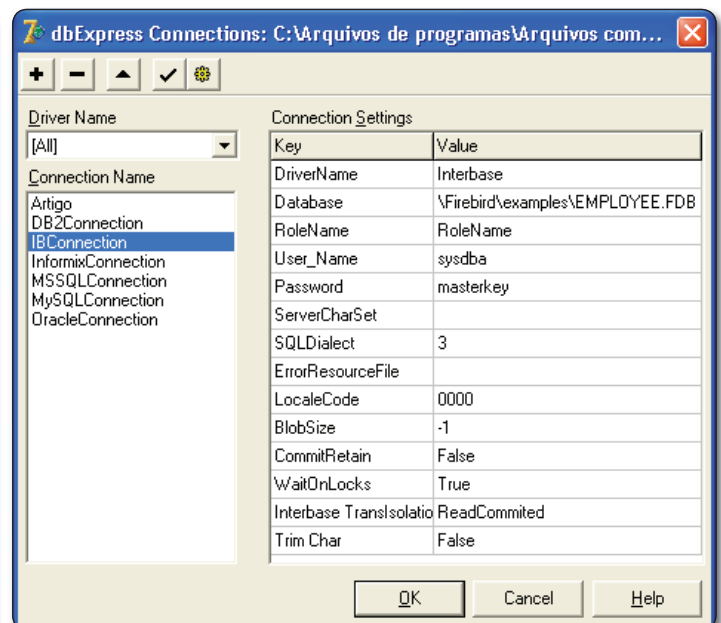


Figura 3. Exemplo de Conexão com o FireBird

formulário ou *Data Module* selecionar os que não serão criados com a aplicação e arrastá-los para o lado (*Available forms*). No final do desenvolvimento do aplicativo mostrado neste artigo, deverá ter-se a mesma configuração apresentada na **Figura 2**.

O próximo passo é configurar a conexão com a base de dados. Para este aplicativo será usada a base de exemplo que acompanha o FireBird. Para isso adicione um componente *SQLConnection* ("sqlConexao"), que está na aba *dbExpress*, ao *sqlConexao* dê dois cliques no componente para configurar suas propriedades.

Crie uma nova conexão clicando no botão "+". Na caixa de diálogo que se abre selecione o driver Interbase em *Driver Name*, dê um nome a conexão em *Connection Name* e confirme.

**Nota:** O Delphi traz por padrão algumas conexões previamente configuradas, como é o caso de *IBConnection*.

Após criar a conexão configure a propriedade *DataBase* informando o endereço completo para o arquivo *EMPLOYEE.FDB*, normalmente localizado na pasta *C:\Arquivos de Programas\Firebird\_1\_5\Firebird\Examples\empbuild\*. A propriedade *User\_Name* é o usuário que será usado para conectar no servidor, por padrão é *SYSDBA* e a senha é *masterkey* (atributo *Password*). Mude o *SQLDialect* para 3 e pressione *OK*, a configuração deverá ficar semelhante a mostrada na **Figura 3**.

No *Object Inspector* mude as propriedades *LoginPrompt* para *False* e ative a conexão mudando para *True* a propriedade *Connected*. Pronto, a conexão com o servidor de banco de dados está estabelecida.

Adicione um novo *Data Module* ao proje-

to. Ele conterá a *query* que será usada para mostrar os dados da tabela *Country*. Salve-o com o nome de "uDMLista", mude a propriedade *Name* para "dmLista" e adicione uma referência a *Unit dmConexao* usando para isso o menu *File>Use Unit* ou o atalho *Alt + F11*. Agora adicione um componente do tipo *SQLQuery* ao *Data Module*, renomeie-o para "qryCountry" e na propriedade *SQLConnection* coloque a conexão criada no *dmConexao*. Coloque a seguinte consulta na propriedade *SQL*:

```
select * from COUNTRY
```

Ative e desative a *query* para verificar se a consulta inserida está correta. Feito isso adicione mais dois componentes: um *DataSetProvider*("dspCountry") alterando a propriedade *DataSet* para "qryCountry". Um *ClientDataSet*("cdsCountry") com a propriedade *DataSetProvider* apontada para o *dspCountry*. Dê dois cliques no *cdsCountry* e com o botão direito do mouse selecione a opção *Add all fields*, com isso estará adicionando todos os campos da tabela ao *Fields Editor*.

Adicione outro *SQLQuery* que será utilizado para mostrar os dados da tabela *Customer*. Mude as propriedades *Name* para "qryCustomerLista", *SQLConnection* para "dmConexao.sqlConexao", e para o *SQL* adicione a instrução a seguir:

```
select * from CUSTOMER
```

Adicione agora um *DataSetProvider*("dspCustomerLista") e mude seu *DataSet* para "qryCustomerLista". Finalmente adicione um *ClientDataSet*("cdsCustomerLista"). Seu *ProviderName* deverá ser apontado para *dspCustomerLista*. Por fim dê dois cliques no *cdsCustomerLista* e com o botão direito do mouse selecione a opção *Add all fields* (**Figura 4**).

Adicione mais um *Data Module* para a

tabela *Customer* da nossa base de dados. Salve com o nome de "uDMCustomer.pas" e mude a propriedade *Name* para "dmCustomer". Adicione uma referência ao *dmConexao* usando *Alt + F11* assim como foi feito anteriormente.

Nesse ponto serão adicionados os componentes que permitem inserir novos registros na tabela. Para isso adicione três componentes sendo: um *SQLQuery*("qryCustomer"), *SQLConnection* como "dmConexao.sqlConexao" e propriedade *SQL* com a instrução a seguir:

```
select * from CUSTOMER
```

Um *DataSetProvider*("dspCustomer") com a propriedade *DataSet* apontada para *qryCustomer* e por último um *ClientDataSet*("cdsCustomer") com sua propriedade *ProviderName* igual a *dspCustomer*.

Assim como nos demais *ClientDataSets* adicione todos os campos da tabela usando a opção *Add all fields* encontrado no menu de contexto do *Fields Editor*, ou seja, clique duas vezes no *cdsCustomer* e em seguida com o botão direito escolha a opção mencionada (**Tabela 1**).

No evento *BeforeOpen* do *cdsCustomer* digite:

```
{ Cria o dmLista }
dmLista := TdmLista.Create(nil);
{ Abre o cdsCountry }
dmLista.cdsCountry.Open;
```

E no evento *AfterClose* digite:

```
{ Fecha o cdsCountry }
dmLista.cdsCountry.Close;
{ Libera de memória o dmLista }
FreeAndNil(dmLista);
```

**Nota:** Não esqueça de adicionar a *Unit* do *Data Module dmLista* ao *Uses* do *Data Module dmCustomer*. Utilize *File>Use unit* ou *Alt + F11*.

Componente	Propriedade	Valor
SQLQuery1	Name	qryCustomer
	SQLConnection	dmConexao.SQLConnection1
	SQL	SELECT * FROM CUSTOMER
DataSetProvider1	Name	dspCustomer
	DataSet	qryCustomer
ClientDataSet1	Name	cdsCustomer
	DataSetProvider	dspCustomer

Tabela 1. Configuração dos componentes utilizados no dmCustomer

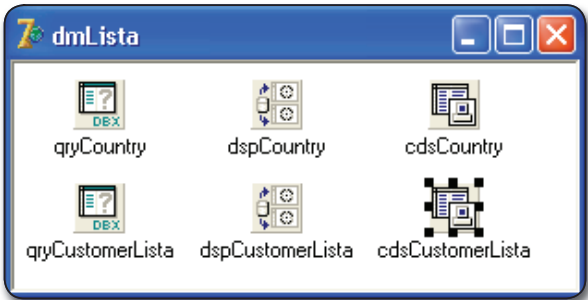


Figura 4. dmLista depois de configurado

Com estes dois eventos garante-se que o *dmLista* será criado quando for aberto o *cdsCustomer* e quando ele for fechado será liberado de memória, não permanecendo em memória quando desnecessário.

Adicione também a este *Data Module* mais um *SQLQuery* que desta vez será usado pelo *Generator CUST\_NO\_GEN*, que vai garantir que não se tenha chaves primárias repetidas para a tabela *Customer*, altere as propriedade *Name* para "qryGenCustomer", *SQLConnection* coloque a conexão criada no *dmConexao*, que neste caso é *sqlConexao*, e insira o seguinte select na propriedade *SQL*:

```
select GEN_ID(cust_no_gen,1) from
rdb$database
```

**Nota:** Generator é um objeto do banco de dados Firebird que, com o auxílio da função *GEN\_ID* também do banco, gera números sequenciais, os famosos Auto Incrementados.

De dois cliques na *query* e adicione todos os campos. No evento *AfterInsert* do *cds*-

*Customer* digite o código da **Listagem 1**.

Para tornar mais clara a utilização será criado agora mais um cadastro para a tabela *SALES* que tem relacionamento "n" para 1 com a tabela *CUSTOMER*. Seguindo a mesma idéia usada anteriormente para o

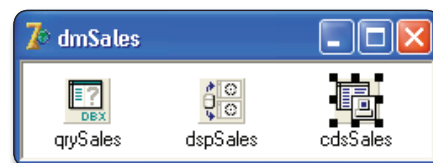


Figura 5. dmSales depois de configurado

#### Listagem 1. Código do evento AfterInsert do cdsCustomer

```
procedure TdmCustomer.cdsCustomerAfterInsert(DataSet: TDataSet);
begin
  { Abre a qryGenCustomer }
  qryGenCustomer.Open;
  { Atribui o valor do qryGenCustomerGEN_ID para o cdsCustomerCUST_NO }
  cdsCustomerCUST_NO.Value := qryGenCustomerGEN_ID.AsInteger;
  { Fecha a qryGenCustomer }
  qryGenCustomer.Close;
end;
```

Componente	Propriedade	Valor
SQLQuery1	Name	qrySales
	SQLConnection	dmConexao.SQLConnection1
	SQL	SELECT * FROM SALES
DataSetProvider1	Name	dspSales
	DataSet	qrySales
ClientDataSet1	Name	cdsSales
	DataSetProvider	dspSales

Tabela 2. Configuração dos componentes utilizados para o dmSales

## Impressão Rápida em Matriciais...

### RDprint 4.0

**O mais completo componente para impressão em MATRICIAIS !**  
**LIDERANÇA absoluta na sua categoria !**  
 Ideal para Notas Fiscais, Duplicatas, Boletos Bancários, etiquetas e relatórios em geral.

- Opção para impressão colorida
- Ajustes de margens para impressão gráfica
- Opção para ocultar a barra de progresso
- Variáveis PAGINAS, DATA, HORA e TÍTULO

**Novo form de SETUP com :**

- Mapeamento das impressoras e Modelos
- Seleção de páginas igual ao word (1-5,7,8)
- Opção para Inverter e Agrupar cópias na impressão

**Novo form de PREVIEW com:**

- Função para Procura de TEXTO no relatório
- ROLAGEM com salto automático de página
- ARRASTO da imagem do preview
- StatusBar com informações da impressão
- Novos ícones personalizados

**RDprint Setup**

Configuração da Impressão

Impressora: HP DeskJet 870Cxi

Modelo: Gráfico - Compatível com Windows

Intervalo de Páginas: ☐ Todas ☐ Página Atual ☒ Páginas: 1-5,7

Cópias: Número de Cópias: 3

Imprimir: ☐ Todas as páginas do intervalo

Opções: ☒ Visualizar ☒ Agrupar ☐ Ordem Inversa

Botões: [Ok] [Cancelar]

**Nova Versão!**

**Deltress Informática**

Fone/Fax (14) 3454-7880  
[www.deltress.com.br](http://www.deltress.com.br)

\* Disponível para Delphi 5, 6, 7, 2005 e 2006 (VCL)  
 \* Compatível com todas as versões do Windows  
 \* Imprime em portas LPT / COM e USB (modo gráfico)

Página: 2 de 19    87%    Impressora: HP DeskJet 870Cxi    Gráfico    \* O RDprint 4.0 não imprime gráficos !



#### Listagem 2. Código do botão Novo

```
procedure TfrmPrincipal.btnNovoClienteClick(Sender: TObject);
begin
  { Verifica se já foi instanciado a o dmCustomer }
  if dmCustomer = nil then
    { Instância a variável dmCustomer }
    dmCustomer := TdmCustomer.Create(nil);
    { Abre o cdsCustomer }
    dmCustomer.cdsCustomer.Open;
    { Insere um novo registro no cdsCustomer }
    dmCustomer.cdsCustomer.Insert;
end;
```

#### Listagem 3. Código do botão Salvar

```
procedure TfrmPrincipal.btnSalvarClienteClick(Sender: TObject);
begin
  { Verifica se já foi criada o dmCustomer }
  if dmCustomer <> nil then
    begin
      { Guarda uma linha no cdsCustomer }
      dmCustomer.cdsCustomer.Post;
      { Salva as alterações guardadas no cdsCustomer para o banco }
      dmCustomer.cdsCustomer.ApplyUpdates(0);
      { Libera da memória o dmCustomer. }
      FreeAndNil(dmCustomer);
    end
  else
    { Caso não exista em memória ainda o dmCustomer emite uma mensagem }
    ShowMessage('Você deve clicar em novo antes de salvar um registro');
end;
```

#### Listagem 4. Código do botão Cancelar

```
procedure TfrmPrincipal.btnCancelarClienteClick(Sender: TObject);
begin
  { Verifica se já foi criada o dmCustomer }
  if dmCustomer <> nil then
    begin
      { Cancela todas as alterações realizadas no cdsCustomer. }
      dmCustomer.cdsCustomer.Cancel;
      { Libera da memória o dmCustomer. }
      FreeAndNil(dmCustomer);
    end
  else
    { Caso não exista em memória ainda o dmCustomer emite uma mensagem. }
    ShowMessage('Você deve clicar no novo antes de cancelar um' + 'registro');
end;
```

cadastro da tabela *CUSTOMER* crie um novo *Data Module* salve com o nome de “uDMSales.pas” e mude a propriedade *Name* para “dmSales”. **Figura 5**

Adicione um componente *SQLQuery*, um *DataSetProvider* e um *ClientDataSet* configurando-os de acordo com a **Tabela 2**.

Dê dois cliques no *cdsSales* com o botão direito do mouse selecione a opção *Add All Fields*. Para o evento *BeforeOpen* do *cdsSales* digite:

```
{ Cria o dmLista }
dmLista := TDMLista.Create(nil);
{ Abre o cdsCustomerLista }
dmLista.cdsCustomerLista.Open;
```

E para o evento *AfterClose* do *cdsSales* digite:

```
dmLista.cdsCustomerLista.Close;
{ Fecha o cdsCustomerLista }
FreeAndNil(dmLista);
{ Libera de memória o dmLista }
```

Com isto tem-se somente em memória o *dmLista* enquanto o *cdsSales* estiver aberto.

### Configurando o formulário principal

Abra o *frmPrincipal* adicione uma referência aos *Data Modules* *dmCustomer*, *dmLista* e *dmSales*. Adicione dois *GroupBox*s mudando a propriedade *Name* para “gbxCustomer” e “gbxSales” respectivamente. Adicione também 4 *DataSources* alterando a propriedade *Name* de cada um deles para *dsCustomer*, *dsSales*, *dsCountryLista* e *dsCustomerLista*. Altere a propriedade *DataSet* de cada um deles apontando-os para seus respectivos *ClientDataSets*. O *dsCustomer* receberá “dmCustomer.cdsCustomer”, para o *dsSales* coloque “dmSales.cdsSales”, no *dsCountryLista* receberá “dmLista.cdsCountry” e para o *dsCustomerLista* coloque “dmLista.cdsCustomerLista”.

Dentro do *gbxCustomer* adicione 10 *DBEdits*, 11 *Labels* e um *DBLookupComboBox*. Para os componentes *DBEdit* configure a propriedade *DataSource* para *dsCustomer* e a propriedade *DataField* faça com que cada um receba um campo diferente.

No *DBLookupComboBox* altere as seguintes propriedades: *Name* para “cbxCountry”, *ListSource* para “dsCountryLista”, *ListField* para “CURRENCY”, *KeyField* para “COUNTRY”, *DataSource* para *dsCustomer* e *DataField* para “COUNTRY”.

Dentro do “gbxSales” adicione 10 *DBEdits*, 11 *Labels* e um *DBLookupComboBox*. Altere a propriedade *DataSource* dos *DBE-*

The screenshot shows the 'frmPrincipal' application window with two main sections: 'Cadastrar' (Register) and 'Sales'. The 'Cadastrar' section has a 'Novo' button and a grid for entering customer data. The grid has columns for CUST\_NO, CUSTOMER, CONTACT\_FIRST, CONTACT\_LAST, ADDRESS\_LINE1, ADDRESS\_LINE2, CITY, PHONE\_NO, STATE\_PROVINCE, POSTAL\_CODE, COUNTRY, and a 'Salvar' button. The 'Sales' section also has a 'Novo' button and a grid for entering sales data. The grid has columns for PO\_NUMBER, CUST\_NO, AGED, ORDER\_STATUS, ORDER\_DATE, DATE\_NEEDED, DISCOUNT, SHIP\_DATE, QTY\_ORDERED, TOTAL\_VALUE, ITEM\_TYPE, and 'Salvar' and 'Cancelar' buttons. The 'COUNTRY' field in the Customer section is a dropdown menu.

Figura 6. frmPrincipal depois de alterado para o cadastro da tabela Customer

dit para *dsSales* e ajuste o *DataField* para que nenhum *DBEdit* fique com o mesmo campo. Já para o *DBLookupComboBox* altere as seguintes propriedades: *Name* para *cbxSales*, *ListSource* para *dsCustomerLista*, *ListField* para "CUSTOMER", *KeyField* para "CUST\_NO", *DataSource* para *dsSales* e *DataField* para "CUST\_NO".

Após concluídas essas configurações o formulário principal deverá se parecer com a **Figura 6**.

## Codificando o exemplo

Faremos agora a parte de codificação do exemplo. Antes de codificar, insira 6 componentes *Button* sendo 3 no *GroupBox* superior com os nomes *btnNovoCliente*, *btnSalvarCliente* e *btnCancelarCliente*. Repita esses passos modificando os nomes dos 3 outros *Buttons* do *GroupBox* para *btnNovaVenda*, *btnSalvarVenda* e *btnCancelarVenda*.

Começaremos pelo botão novo chamado de *btnNovo*. Dê um duplo clique no *btnNovoCliente* que está dentro do *gbxCustomer* e insira o seguinte código da **Listagem 2**.

Para o evento *onClick* do *btnSalvar* que está dentro do *gbxCustomer* digite o código da **Listagem 3**.

Para o evento *onClick* do *btnCancelar* que está dentro do *gbxCustomer* digite o código da **Listagem 4**.

Em seguida codifique o *btnNovaVenda* que está dentro *gbxSales* digite o código da **Listagem 5**.

Já para o botão *btnSalvarVenda* que também está dentro do *gbxSales* digite o código da **Listagem 6**.

Por último o *btnCancelar* digite o seguinte código da **Listagem 7**.

### Listagem 5. Código do botão Nova Venda

```
procedure TfrmPrincipal.btnNovaVendaClick(Sender: TObject);
begin
  { Verifica se já foi criado o dmSales }
  if dmSales = nil then
    { Instancia a variável dmSales }
    dmSales := TdmSales.Create(nil);
    { Abre o cdsSales }
    dmSales.cdsSales.Open;
    { Insere um novo registro no cdsSales }
    dmSales.cdsSales.Insert;
end;
```

### Listagem 6. Código do botão Salvar Venda

```
procedure TfrmPrincipal.btnSalvarVendaClick(Sender: TObject);
begin
  { Verifica se já foi criado o dmSales }
  if dmSales <> nil then
    begin
      { Guarda uma linha no cdsSales }
      dmSales.cdsSales.Post;
      { Salva as alterações guardadas no cdsSales para o banco }
      dmSales.cdsSales.ApplyUpdates(0);
      { Fecha o cdsSales }
      dmSales.cdsSales.Close;
      { Libera da memória o dmSales }
      FreeAndNil(dmSales);
    end
  else
    { Caso não exista em memória ainda o dmCustomer emite uma mensagem }
    ShowMessage('Você deve clicar no novo antes de salvar um registro');
end;
```

### Listagem 7. Código do botão Cancelar Venda

```
procedure TfrmPrincipal.btnCancelarVendaClick(Sender: TObject);
begin
  { Verifica se já foi criada o dmSales }
  if dmSales <> nil then
    begin
      { Cancela todas as alterações realizadas no cdsSales }
      dmSales.cdsSales.Cancel;
      { Fecha o cdsSales }
      dmSales.cdsSales.Close;
      { Libera da memória o dmSales }
      FreeAndNil(dmSales);
    end
  else
    { Caso não exista em memória ainda o dmCustomer emite uma mensagem }
    ShowMessage('Você deve clicar no novo antes de cancelar um registro');
end;
```

## Conclusão

Embora os computadores de hoje possuam grande quantidade de memória RAM disponível, ainda é preciso que se tenha cuidado com o desenvolvimento, pois nem sempre as aplicações são executadas numa

rede local. Este artigo mostrou passo a passo uma técnica simples e eficaz para criar os componentes de acesso à dados em tempo de execução, o que torna a inicialização da aplicação mais rápida e ao mesmo tempo consome menos memória RAM. ●

+ de 80.000 membros cadastrados  
+ de 15.000 exemplos com fontes  
+ de 900 apostilas  
+ de 4.000 dicas  
Fórum Delphi  
Artigos

TOTALMENTE GRÁTIS

**www.delphi.eti.br**

Um dos maiores sites de apoio a desenvolvedores Delphi do Brasil!!!