



Contas a Pagar e Cobrança

Crie um sistema completo com Delphi, Firebird 2.0 e dbExpress - Parte 4

Chegamos a quarta parte do nosso mini-curso Sistema de Contas a Pagar e Cobrança onde estamos vendo alguns dos principais recursos em aplicações dessa natureza. Uma aplicação de contas a pagar e cobrança não se resume apenas em entradas e saídas de caixa. Deve-se pensar também em fluxo de caixa e geração de boletos bancários para cobrança, um dos carros-chefe da aplicação.

Para iniciarmos nosso artigo, veremos rapidamente o que foi criado no exemplo anterior:

- Cadastro de contas a receber;
- Manutenção do Data Module principal para criação da tabela Contas_Receber;
- Criação do recurso de duplicação de contas parceladas;

Daremos continuidade ao curso entendendo e desenvolvendo outro recurso bastante importante: Geração de texto para Cobrança Escritural. Nesse artigo veremos como fazer a geração do arquivo

texto para envio à instituição bancária que por sua vez se encarregará de gerar os boletos e envio dos mesmos efetuando desta forma a cobrança bancária.

Alterando o banco de dados de exemplo

Diferentemente dos artigos anteriores não precisaremos criar tabelas. Dessa vez criaremos uma *Stored Procedure* selecionável que resultará os dados necessários para geração dos arquivos de remessa.

Uma *Stored Procedure* selecionável é basicamente uma função criada no banco de dados capaz de preencher um ou mais parâmetros de saída que posteriormente serão usados pela aplicação Delphi como se fosse uma tabela comum.

Nossa *Stored Procedure* fará um filtro na tabela de contas a receber usando como parâmetros de entrada duas datas. Essas datas serão usadas na cláusula *Where* da SP ("Stored Procedure") onde faremos o filtro no campo *DT_VECTO*, ou seja, filtraremos



Adriano Santos

(artes@doiscliques.com)

é desenvolvedor Delphi desde 1998. Professor e programador PHP. Bacharel em Comunicação Social pela Universidade Cruzeiro do Sul, SP. É colunista e membro da Comissão Editorial da revista ClubeDelphi. É também Editor Técnico da Revista ClubeDelphi. Mantém o blog Delphi to Delphi (www.delphitodelphi.blogspot.com) com dicas, informações e tudo sobre desenvolvimento Delphi.

todos os registros de contas a receber que estejam entre as duas datas recebidas.

Será levado em consideração também se o registro é diferente de "I" ("inativo") e "B" ("baixado"). Assim evitamos enviar um título inativo por meio de cancelamento/exclusão ou ainda que tenha sido liquidado ("baixado").

Para criar nossa SP selecionável abra o IBExpert e conecte-se a base de dados de exemplo SysPague. Em seguida clique com o botão direito em *Procedure* e selecione *New Procedure*. Repare que o IBExpert exibe, em seu editor, uma divisão onde encontramos os botões *Input Parameters*, *Output Parameters*, *Variables* e *Cursor*. E mais acima podemos ver uma área vazia com os títulos *Name*, *Type*, *Size*, *Scale*, *Default Source*, *Subtype*, *Charset* e *Description*. É nessa área que criaremos os parâmetros de entrada ("Input Parameters") e saída ("Output Parameters") que por sinal se assemelha bastante com a criação de campos em uma tabela.

Pois bem, clique em *Input Parameters* e com o botão direito do mouse na área vazia selecione a opção *Append parameter/variable* ou localize este botão logo acima dessa área e clique-o. Um novo parâmetro será criado, agora basta modificar seu *Name* para "DT_INICIO" e seu *Type* para "Timestamp" deixando o restante dos itens com seus valores padrão. Repita o processo e crie um parâmetro chamado "DT_FIM" também como "Timestamp" em seu *Type*.

Nesse momento acabamos de criar dois parâmetros de entrada que serão usados em nossa aplicação. Enviaremos para

Campo	Tipo/Tamanho
CNPJ	VarChar(18)
VALOR_REAL	Numeric(15,2)
DT_CADASTRO	TimeStamp
DT_VECTO	TimeStatmp
CODIGO	VarChar(20)
RAZAO	VarChar(150)
ENDERECO	VarChar(100)
BAIRRO	VarChar(50)
CIDADE	VarChar(50)
ESTADO	VarChar(2)
CEP	VarChar(9)

Tabela 1. Parâmetros de saída da SP

a *Stored Procedure* uma data de início e outra de fim, assim nossa SP selecionará automaticamente o que deverá ser enviado à instituição.

Agora vamos aos parâmetros de saída chamados de *Output Parameters*. São eles que nossa SP alimentará e que serão utilizados pelo nosso sistema como se fosse uma tabela comum do banco. Portanto clique no item *Output Parameters* e repita os passos anteriores incluindo os parâmetros da **Tabela 1**.

Modifique o nome da *Stored procedure* para "RetornarBoletos" na parte superior direita do IBExpert. Caso prefira, execute o *script* da **Listagem 1**.

Cobrança Escritural - CNAB

Para que possamos desenvolver a cobrança escritural, também chamada de CNAB, devemos primeiramente entender exatamente do que se trata e qual sua finalidade.

CNAB significa "Comissão de Tecnologia e Automação Bancária" e é basicamente um conjunto de regras estipuladas para geração de dados em formato texto. Esses dados são posteriormente enviados à instituição bancária para que sejam processados. Esse conjunto de regras,

também chamado de padrão, contém texto em forma de colunas que seguem regras definidas pela FEBRABAN ("Federação Brasileira dos Bancos"). O conjunto de regras estipuladas pela FEBRABAN é chamado de layout.

O layout do arquivo pode variar de banco para banco devido às particularidades de cada um, porém não é permitido que fuja demais do padrão estipulado.

Normalmente esses layouts são disponibilizados publicamente no website da instituição para que desenvolvedores possam adquirir e desenvolver suas próprias soluções. Contudo em alguns casos o layout só pode ser adquirido por meio de um correntista que possui usuário e senha de acesso ao portal do banco, por isso muitas vezes é necessário que o próprio cliente adquira o layout e o retransmita ao desenvolvedor responsável.

Os layouts CNAB seguem dois padrões:

- CNAB 240: Possui 240 posições, também chamado de colunas, em cada linha. Sendo que para cada boleto a ser gerado pela instituição há duas linhas de 240 colunas cada, ou seja, necessitamos gerar um boleto bancário para cada cliente e supondo que temos dez clientes, o arquivo deverá possuir vinte linhas de registro;

Listagem 1. Criação da tabela e índices Contas_Receber

```

SET TERM ^ ;

CREATE PROCEDURE RETORNARBOLETOS (
    DT_INICIO TIMESTAMP,
    DT_FIM TIMESTAMP)
RETURNS (
    CNPJ VARCHAR(18),
    VALOR_REAL DOUBLE PRECISION,
    DT_CADASTRO TIMESTAMP,
    DT_VECTO TIMESTAMP,
    CODIGO VARCHAR(20),
    RAZAO VARCHAR(150),
    ENDERECO VARCHAR(100),
    BAIRRO VARCHAR(50),
    CIDADE VARCHAR(50),
    ESTADO VARCHAR(2),
    CEP VARCHAR(9))
AS
begin
FOR SELECT
    CP.CODIGO, CP.CNPJ, CP.VLR_REAL,
    CP.DT_CADASTRO, CP.DT_VECTO, CL.RAZAO,
    CL.ENDERECO, CL.BAIRRO, CL.CIDADE,
    CL.ESTADO, CL.CEP
FROM
    CONTAS_RECEBER CP
    INNER JOIN CLIENTES CL
    ON (CP.CNPJ = CL.CNPJ)
WHERE
    (CP.DT_VECTO BETWEEN :DT_INICIO AND :DT_FIM) AND
    (CP.STATUS NOT IN ('I', 'B'))
INTO
    :CODIGO, :CNPJ, :VALOR_REAL, :DT_CADASTRO,
    :DT_VECTO, :RAZAO, :ENDERECO, :BAIRRO, :CIDADE,
    :ESTADO, :CEP
DO
    SUSPEND;
end^
    
```

• CNAB 400: São 400 colunas para cada linha do arquivo;

Atualmente o formato 240 está sendo gradativamente desativado pelas instituições por ter se tornado obsoleto, complexo e trabalhoso de lidar.

A finalidade do CNAB é poder efetuar a cobrança bancária de títulos, ou seja, após enviada ao banco a remessa é processada e boletos bancários são expedidos ao cliente. O arquivo de remessa possui entre as várias colunas dados como: valor, vencimento, agência e conta onde o valor pago será creditado, instruções de protesto e dados do sacado, entre outros.

Além do arquivo Remessa o banco disponibiliza o arquivo Retorno, que é justamente o contrário de remessa. Com o arquivo retorno podemos efetuar a baixa do título em nosso sistema informando valor pago, juros, mora, data de pagamento, agência e conta creditada, e etc.

De porte de todas essas informações fica fácil conferir o extrato bancário no início ou fim do dia através do sistema tendo como diferença pequenas taxas e tarifas, tais como: taxa de manutenção da conta corrente e CPMF.

Nota: Nesse artigo veremos somente a geração do arquivo Remessa, porém a operação inversa segue basicamente a mesma idéia.

Nota: Para uma melhor compreensão do artigo é recomendada a leitura do artigo Sistema EDI na edição 84 da revista ClubeDelphi.

Como dito anteriormente, cada banco possui certas particularidades em relação ao padrão utilizado, portanto o mais sensato é encontrar o layout CNAB do banco desejado e fazer a leitura completa de todo o arquivo de ajuda que normalmente encontra-se em formato PDF ou DOC.

Além do layout as instituições costumam manter um departamento de suporte ao desenvolvedor para sanar possíveis dúvidas com o layout e/ou procedimentos de envio e retorno dos arquivos a serem trocados entre instituição financeira e cliente.

A grande maioria dos bancos disponibiliza uma área de testes para o de-

seenvolvedor efetuar os primeiros testes de carregamento e geração dos boletos bancários.

A geração destes arquivos de remessa pode ser feito utilizando-se de componentes de terceiros, mas por questões de didática e entendimento do mecanismo de desenvolvimento optei por fazer manualmente, até mesmo porque o processo é bastante simples.

Preparando o sistema

Antes de iniciarmos o desenvolvimento da caixa de diálogo e geração dos arquivos, faremos a preparação pra a leitura dos dados que será feita através da SP RetornarBoletos criada no início do artigo.

Abra o Delphi 7 e em seguida o projeto SysPague.dpr de nosso mini-curso. Após a abertura do projeto abra a Unit do Data Module principal e inclua um novo grupo de componentes para acesso a nossa SP.

Insira um componente do tipo SQL-Query (“sqlBoletos”) e mude sua propriedade SQLConnection para “sqlConexao”. Em seguida digite a instrução SQL a seguir na propriedade SQL do componente SQLQuery.

```
select * from RetornarBoletos(:DT_INICIO, :DT_FIM) order by CNPJ
```

Note que criamos dois parâmetros na instrução: DT_INICIO e a DT_FIM. Esses parâmetros são usados para selecionar quais os dados serão retornados pela *Stored Procedure* para que possamos então gerar o arquivo remessa. Em nossa SP todos os registros da tabela Contas_Receber que têm data de vencimento entre as DT_INICIO e DT_FIM e que seu status não seja “I” (inativo) ou “B” (baixado) serão retornados.

Poderíamos requerer outros dados, como por exemplo o CNPJ do cliente, assim poderíamos gerar o arquivo de remessa por cliente pagador. Para isso seria necessária a criação de um novo parâmetro de entrada na SP.

Feito isso devemos agora configurar os parâmetros criados pelo componente *SQLQuery*, por isso clique duas vezes na propriedade *Params* do *sqlBoletos* e veja que ambos os parâmetros foram criados.

Por se tratar de parâmetros do mesmo tipo, faremos a configuração de ambos ao mesmo tempo, por isso selecione-os simultaneamente usando a tecla CTRL e clicando com o mouse em cada um.

Agora pressione F11 para alternar para o *Object Inspector* e mude a propriedade *DataType* para “ftTimeStamp” e *ParamType* para “ptInput”.

Feito isso, insira todos os parâmetros de saída, que agora se tornarão campos, no *Fields Editor* do *sqlBoletos*. Para isso clique duas vezes no *sqlBoletos* e em seguida com o botão direito escolha a opção *Add all fields*.

Como pode notar não há segredos na criação de SP's selecionáveis. Faremos a leitura de sei *Result Set* (“resultado da consulta”) como se fosse uma tabela comum do banco de dados.

Insira agora um componente do tipo *DataSetProvider* (“dspBoletos”) da paleta *Data Access* e configure sua propriedade *DataSet* para “sqlBoletos”.

Coloque um *ClientDataSet* (“cndsBoletos”) também da paleta *Data Access* e ligue-o ao *DataSetProvider* usando a propriedade *ProviderName*. Em seguida clique com o botão direito no componente e selecione *Fetch Params* para que o *cndsBoletos* capture os parâmetros criados no *sqlBoletos*.

Use a opção *Add all fields* do *Fields Editor* para adicionar todos os campos da SP selecionável assim como fizemos no *sqlBoletos*. Se preferir modifique a propriedade *DisplayName* de cada campo adicionado conforme desejado. Essa modificação refletirá nos componentes visuais incluídos em tela tais como *Labels* e *DBGrids*. Veja como ficou nosso *Data Module* na **Figura 1**.

Criando a tela geração da remessa

Nossa tela de geração dos arquivos de remessa consiste apenas em receber o período desejado para geração do CNAB e a escolha de qual banco, agência e conta serão os mantenedores dos boletos.

Fixaremos a pasta de geração do arquivo usando a função *ExtractFilePath* passando como parâmetro o método *ExeName* do objeto *TApplication* mais o nome do arquivo de remessa, que nesse caso fixaremos como: “Remessa.txt”.

Fazendo isso garantimos que a pasta de geração seja sempre o local onde nossa

aplicação esteja rodando, ou seja, a pasta de instalação do executável. Mostraremos este caminho em um *Label* azul para que o usuário visualize e identifique rapidamente o caminho da remessa.

Vamos iniciar então a criação de um novo formulário e para isso selecione *File|New>Form*. Salve o formulário usando o menu *File|Save as* e atribua a ele o nome "uCobrancaEscritural.pas". Em seguida mude a propriedade *Name* do formulário para "frmCobrancaEscritural". Se desejar modifique sua largura *Width* e altura *Height* bem como as configurações de bordas: *BorderStyle* e *BorderIcons*. Uma prévia da tela pode ser vista na **Figura 2**.

Primeiramente adicione um componente *GroupBox* da paleta *Standard* e dentro dele insira dois *DateTimePicker* com os nomes "dtpDtInicio" e "dtpDtFim" respectivamente. Acima de cada um adicione dois *Labels* com os *Captions* de "Data de início" e "Data de fim" respectivamente. Inclua um novo *Label* entre *dtpDtInicio* e *dtpDtFim* com o *Caption* "a".

Após isso insira três novos *Labels* e mude o *Caption* de cada um para "Banco", "Agência" e "Conta". Eles servirão de nomes aos três *Edits* que serão adicionados agora, sendo que cada um será responsável por representar um campo da tabela CONTAS, ou seja, coloque estes três componentes e renomeie-os para "edtBanco", "edtAgencia" e "edtConta".

Após isso inclua um *SpeedButton* ("sbbSelegcionarBanco") e adicione uma imagem ao mesmo usando a propriedade *Glyph*. Se preferir copie o botão de alguma janela que já tenhamos adicionado uma instância desse componente, como "Cadastro de Clientes", por exemplo, e cole-o no formulário.

Ao centro da janela insira um *ProgressBar* ("pgbProgresso") da paleta Win32. Modifique sua propriedade *Step* para "1".

No rodapé da página adicione dois *Labels* sendo que o primeiro deverá ter o *Caption* alterado para "Salvar arquivo como:". Já no segundo *Label* ("lblCaminhoArquivo") cujo o *Caption* será "C:\Temp" configure as propriedades o e *AutoSize* para *False*. Redimensione a largura e altura do *lblCaminhoArquivo* de forma que ocupe a largura da janela e duas ou mais linhas de altura.

Adicione dois botões à tela configurando seus respectivos *Captions* para "Gerar" e "Cancelar". Modifique também a propriedade *Name* do botão "Gerar" para "btnGerar" e do botão "Cancelar" para "btnCancelar".

Codificando o formulário

A primeira providência a tomarmos em relação a codificação, será alterar a propriedade *Caption* do *lblCaminhoArquivo* em dinamicamente para que o usuário possa visualizar onde o arquivo será salvo, por isso entre no evento *OnShow* do formulário e apenas digite o código a seguir:

```
lblCaminhoArquivo.Caption :=
ExtractFilePath(Application.
ExeName)+'Remessa.txt';
```

Como mencionado anteriormente estamos concatenando o diretório de instalação do executável com o nome de arquivo que é fixado em "Remessa.txt".

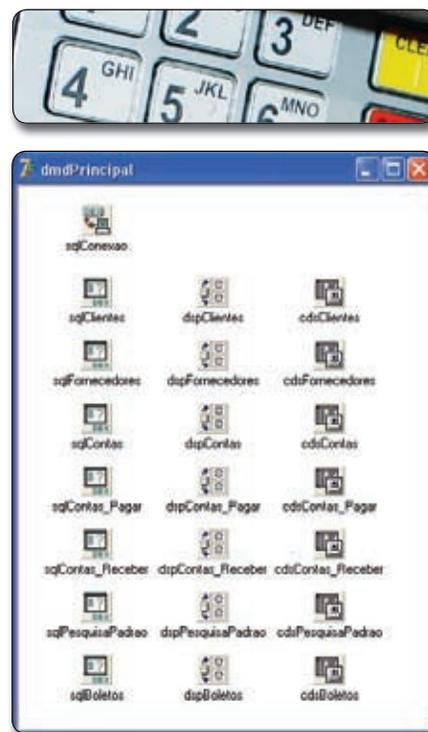


Figura 1. Data Module alterado

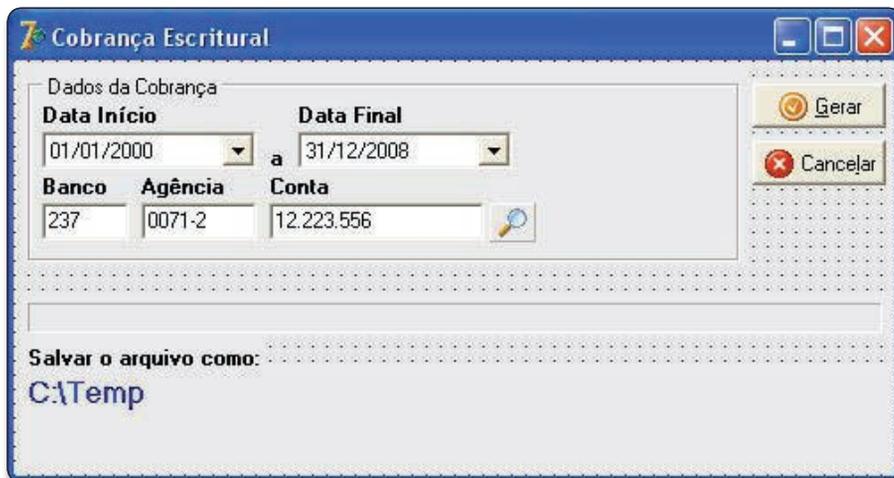


Figura 2. Tela de sugestão da Cobrança Escritural

Listagem 2. Botão de pesquisa de bancos

```
procedure TfrmCobrancaEscritural.
sbbSelegcionarBancoClick(Sender: TObject);
begin
frmPesquisaPadrao := TfrmPesquisaPadrao.Create(Self);
with frmPesquisaPadrao do
begin
Ftabela := 'Contas';
FDataSetAlvo := dmdPrincipal.cdsContas;
ShowModal;

if (ModalResult = mrOk) and not
(dstGenerico.DataSet.IsEmpty) then
begin
edtBanco.Text := dstGenerico.DataSet.FieldByName('BANCO').AsString;
edtAgencia.Text := dstGenerico.DataSet.FieldByName('AGENCIA').AsString;
edtConta.Text := dstGenerico.DataSet.FieldByName('CONTA').AsString;
end;
end;
FreeAndNil(frmPesquisaPadrao);
end;
```

Em seguida faremos a codificação do botão *sbbSelecionarBanco* que possui exatamente o mesmo método de pesquisa padrão encontrado nas janelas “Baixar contas a receber” e “Baixar contas a pagar”, ou seja, faz uso do formulário “Pesquisa padrão”.

Se preferir abra um dos formulários de baixa descritos anteriormente e copie o evento do botão de pesquisa para o *sbbSelecionarbanco*. O código deste botão está descrito na **Listagem 2**.

Nota: Não esqueça de que para funcionar a Pesquisa Padrão seu formulário deve ser adicionado ao Usos do formulário Cobrança Escritural e para isso será necessário pressionar ALT + F11 ou ir ao menu File|Use unit.

Nessa parte do artigo faremos uso também de algumas funções e procedimentos especiais para só então codificarmos

a geração da remessa propriamente dita. Essas funções e procedimentos especiais se fazem necessárias, pois devemos adequar nosso arquivo de remessa aos padrões de CNAB. Essas funções são:

- *LimparString*: Retira caracteres de uma *String* conforme o segundo parâmetro passado. Ideal para limpar máscaras em campos do tipo CPF, CEP e CNPJ;
- *PadL*: Alinha o texto à esquerda incluindo caracteres em branco à direita da *string*; Ideal para completar as posições à esquerda do texto que precisam ser preenchidas para se enquadrar com o número de caracteres solicitados pelo CNAB;
- *PadR*: Ao contrário de *PadL*, ou seja, alinha o texto à direita incluindo caracteres em branco à esquerda da *String*.
- *Replicar*: Repete um determinado caractere “n” vezes;
- *SoNumeros*: Retira máscaras de números deixando somente números inteiros; Para incluir as funções e procedimen-

tos acima, declare-as na seção *Public* do formulário conforme a seguir e pressione CTRL + SHIFT + C para que o Delphi crie para nós o cabeçalho das mesmas.

```
function LimparString(
  ATexto, ACaracteres: string): string;
function PadL(
  ATexto: string; ATamanho: Integer): string;
function PadR(
  ATexto: string; ATamanho: integer): string;
function Replicar
  (ATexto: string; AVEzes: Integer): string;
function SoNumeros(
  const ATexto: string): string;
```

Encontre na área *Implementation* do formulário o cabeçalho de cada função e codifique de acordo com a **Listagem 3** onde você pode ver o código de todos os procedimentos e funções.

Algumas considerações devem ser revistas antes de iniciarmos a codificação do layout. De acordo com o padrão CNAB:

- Todo campo de texto deve ser formatado com alinhamento à esquerda e preenchido com caracteres brancos à direita;
- Todo e qualquer campo numérico deve ser alinhado à direita precedido de zeros;
- Campos de valor, como juros, mora e valor do boleto, devem ser apresentados sem formatação monetária, ou seja, sem quaisquer separadores seja ele milhar ou decimal;
- Campos de data devem obedecer ao padrão estipulado no layout do banco podendo variar bastante de um banco para outro, ex: DDMMYYYY, DDMMYY, DMY e etc;

Muito bem, agora precisamos apenas codificar a parte mais importante de nosso exemplo que é o evento *OnClick* botão *btnGerar*. Por isso clique duas vezes no botão em questão e digite o código da **Listagem 4**.

Entendendo o código

O código de geração da remessa não possui, na verdade, nenhuma magia ou efeito especial de cinema. Nas primeiras linhas do código estamos declarando duas variáveis que receberão o endereço de memória do arquivo e o número da linha gerada.

Primeiramente atribuímos o valor dos parâmetros DT_INICIO e DT_FIM do *cdsBoletos*, presente no Data Module, o mesmo valor dos campos *dtpDtInicio* e *dtpDtFim* que são as datas de início e fim do período de vencimento dos boletos.

Listagem 3. Funções e procedimentos especiais

```
function TfrmCobrancaEscritural.
  LimparString(ATexto, ACaracteres: string): string;
var
  strAux      : string;
  I           : integer;
begin
  strAux := '';
  for I := 1 to Length(ATexto) do
    if Pos(Copy(ATexto, I, 1), ACaracteres) <= 0 then
      strAux := strAux + Copy(ATexto, I, 1);
  Result := strAux;
end;

function TfrmCobrancaEscritural.PadL(ATexto: string; ATamanho: Integer): string;
var
  I : integer;
begin
  Result := ATexto;
  for I := 1 to (ATamanho - Length(Result)) do
    Result := Result + ' ';
end;

function TfrmCobrancaEscritural.PadR(ATexto: string; ATamanho: integer): string;
var
  I : integer;
begin
  Result := ATexto;
  for I := 1 to (ATamanho - Length(Result)) do
    Result := ' ' + Result;
end;

function TfrmCobrancaEscritural.SoNumeros(const ATexto: string): string;
var
  I: Integer;
  S: string;
begin
  S := '';
  for I := 1 to Length(ATexto) do
    if (ATexto[I] in ['0'..'9']) then
      S := S + Copy(ATexto, I, 1);
  Result := S;
end;

function TfrmCobrancaEscritural.Replicar(ATexto: string; AVEzes: Integer): string;
var
  I: Integer;
begin
  Result := '';
  for I := 1 to AVEzes do
    Result := Result + ATexto;
end;
```

Em seguida abrimos o *ClientDataSet* que por sua vez executa a *Stored Procedure RetornarBoletos*. Caso o método *IsEmpty* do *cdsBoletos* retorne diferente de *True*, ou seja, se houverem dados no *Resultset* do *ClientDataSet* significa que existem boletos a serem gerados.

Verificamos então a existência do arquivo *Remessa.txt* no diretório da aplicação e o apagamos caso exista, assim criamos um arquivo novo no diretório.

Nota: Poderíamos criar uma função para retornar nomes aleatórios para que pudéssemos guardar o arquivo anterior, porém não há necessidade de se guardar os arquivos de remessa já enviados ao banco.

Os métodos *AssignFile* e *ReWrite*, criam e inicializam o arquivo *Remessa.txt* criando em memória uma instância deste arquivo para ser manipulado pelo sistema.

Um arquivo *CNAB* é composto por basicamente três seções:

- *Header*: Cabeçalho do arquivo contendo informações sobre o sacador;
- *Detail*: São os boletos em si. No padrão *CNAB 240* são duas linhas de *Detail* (“detalhe”) para cada boleto gerado;
- *Trailer*: Rodapé do arquivo, normalmente consta somente o número de registros gerados e a data;

A primeira parte do arquivo consiste em escrever o *Header* da remessa, ou seja, o cabeçalho do *CNAB*. Note que a última linha antes do fim do *Header*

usamos a função *WriteLn*. O *WriteLn* escreve o texto e quebra a linha enviando o cursor para a linha posterior ao passo que *Write* apenas escreve o texto sem mover o cursor.

Ao entrar no laço *while..do* estamos escrevendo o *Detail* do arquivo que é justamente o detalhe, ou detalhamento de cada boleto a ser gerado pela instituição financeira. Cada linha gerada é um boleto bancário que o banco se encarregará de gerar e enviar ao sacado. Após o término do laço escrevemos o *Trailer*, ou rodapé do arquivo remessa.

Esses são todos os passos efetuados no código visto agora a pouco. Como vimos não há segredos. Nas **Figuras 3 e 4** podemos ver o sistema em execução e uma prévia do arquivo carregado.

Listagem 4. Código do botão Gerar

```

procedure TfrmCobrancaEscritural.btnGerarClick(Sender: TObject);
var
  Arquivo: TextFile;
  Contador: Integer;
begin
  with dmdPrincipal, cdsBoletos do
  begin
    { Para uso da função de conversão
      DateTimeToSQLTimeStamp declare SQLTimSt }
    cdsBoletos.Params.ParamByName('DT_INICIO').
      AsSQLTimeStamp := DateTimeToSQLTimeStamp
      (dbeDtInicio.DateTime);
    cdsBoletos.Params.ParamByName('DT_FIM').
      AsSQLTimeStamp := DateTimeToSQLTimeStamp
      (dbeDtFim.DateTime);
    cdsBoletos.Open;
    if not cdsBoletos.IsEmpty then
    begin
      { Gera os arquivos para a cobrança escritural }
      if FileExists(ExtractFilePath(Application.
        ExeName)+'Remessa.txt') then
        DeleteFile(ExtractFilePath(Application.
          ExeName)+'Remessa.txt');
      { Cabeçalho / Header do Arquivo }
      Contador := 1;
      AssignFile(Arquivo, ExtractFilePath(
        Application.ExeName)+'Remessa.txt');
      Rewrite(Arquivo);
      Write (Arquivo, '01');
      Write (Arquivo, 'REMESSA');
      Write (Arquivo, '01');
      Write (Arquivo, 'COBRANCA');
      Write (Arquivo, edtAgencia.Text);
      Write (Arquivo, '00');
      Write (Arquivo, edtConta.Text);
      Write (Arquivo, Replicar(' ', 6));
      Write (Arquivo, 'CLUBEDELPHI LTDA. ');
      Write (Arquivo, '341');
      Write (Arquivo, 'BANCO ITAU SA ');
      Write (Arquivo, FormatDateTime('DDMMYYYY', Now));
      Write (Arquivo, Replicar(' ', 294));
      WriteLn(Arquivo, FormatFloat('000000', Contador));
      { Fim do Header do Arquivo }
      pgbProgresso.Position;
      pgbProgresso.Max := RecordCount;
      while not EOF do
      begin
        Inc(Contador);
        { Salva alguns dos principais dados no arquivo de remessa }
        Write (Arquivo, '1');
        Write (Arquivo, '02');
        Write (Arquivo, LimparString(FieldByName('CNPJ').AsString, '-'));
        Write (Arquivo, PadL(edtBanco.Text, 4));
        Write (Arquivo, '00');
        Write (Arquivo, PadL(LimparString(edtAgencia.Text, '-'), 6));
        Write (Arquivo, Replicar(' ', 4));
        Write (Arquivo, Replicar(' ', 4));
        Write (Arquivo, PadL('USO DA EMPRESA', 25));
        Write (Arquivo, PadL(Copy(FieldByName('CODIGO').AsString, 1, 8), 8));
        Write (Arquivo, Replicar('0', 12));
        Write (Arquivo, '109');
        Write (Arquivo, '00');
        Write (Arquivo, PadL(Copy(FieldByName('CODIGO').AsString, 1, 10), 10));
        Write (Arquivo, FormatDateTime('DDMMYY',
          FieldByName('DT_VECTO').AsDateTime));
        Write (Arquivo, FormatFloat('#####.##',
          FieldByName('VALOR_REAL').AsFloat));
        Write (Arquivo, '341');
        Write (Arquivo, PadL(LimparString(edtAgencia.Text, '-'), 6));
        Write (Arquivo, '01');
        Write (Arquivo, 'A');
        Write (Arquivo, FormatDateTime('DDMMYY',
          FieldByName('DT_CADASTRO').AsDateTime));
        Write (Arquivo, '02');
        Write (Arquivo, '09');
        Write (Arquivo, FormatFloat('#####.##', 0));
        Write (Arquivo, FormatDateTime('DDMMYY',
          FieldByName('DT_VECTO').AsDateTime));
        Write (Arquivo, FormatFloat('#####.##', 0));
        Write (Arquivo, FormatFloat('#####.##', 0));
        Write (Arquivo, '02');
        Write (Arquivo, PadL(LimparString(edtAgencia.Text, '-'), 6));
        Write (Arquivo, PadL(Copy(FieldByName('RAZAO')
          .AsString, 1, 30), 30));
        Write (Arquivo, Replicar(' ', 10));
        Write (Arquivo, PadL(Copy(FieldByName('ENDERECO').AsString, 1, 40), 40));
        Write (Arquivo, PadL(Copy(FieldByName('BAIRRO').AsString, 1, 12), 12));
        Write (Arquivo, LimparString(FieldByName('CEP').AsString, '-'), 8);
        Write (Arquivo, PadL(Copy(FieldByName('CIDADE').AsString, 1, 15), 15));
        Write (Arquivo, FieldByName('ESTADO').AsString);
        Write (Arquivo, Replicar(' ', 42));
        WriteLn(Arquivo, FormatFloat('000000', Contador));
        Next;
      pgbProgresso.StepIt;
    end;
    { Rodapé / Trailer do Arquivo }
    Inc(Contador);
    Write (Arquivo, '9');
    Write (Arquivo, Replicar(' ', 393));
    WriteLn(Arquivo, FormatFloat('000000', Contador));
    { Fim do Header do Arquivo }
    CloseFile(Arquivo);
    MessageDlg('Arquivo gerado com sucesso!', mtInformation, [mbOk], 0);
  end
  else
  begin
    MessageDlg('Nada selecionado!', mtWarning, [mbOk], 0);
    dbeDtInicio.SetFocus;
  end;
end;
end;
end;

```

Últimas considerações

Além de todas essas informações que vimos há ainda que se pensar e lembrar que existem diversas carteiras de cobrança a serem utilizadas. As carteiras de cobranças são as regras básicas de como o banco fará a execução da cobrança junto ao cliente de seu usuário.

Também é importante salientar as formas de cobrança que são com e sem registro. Sem registro e com registro são na verdade algumas das modalidades de cobrança. Em nosso caso estamos utilizando cobrança com registro o que significa que estamos registrando previamente os dados de cobrança no banco de dados da instituição através do envio de arquivos texto.

Para mais detalhes consulte o gerente de sua conta corrente para que ele possa lhe passar todas as informações necessárias a respeito do assunto.

Conclusão

Finalizamos a quarta parte deste minicurso gerando a cobrança escritural, CNAB, onde tivemos a oportunidade de ver como fazer na prática a geração de um arquivo remessa. Esses são apenas os passos mais importantes da geração, porém o desenvolvimento como um todo requer um grau de atenção ainda maior para que todas as regras impostas no layout sejam cumpridas.

Sempre que for preciso consulte o suporte técnico do banco para qual esteja desenvolvendo o CNAB para que não fiquem dúvidas durante a programação do processo.

Um forte abraço e até a próxima. ●



Figura 3. Arquivo gerado com sucesso

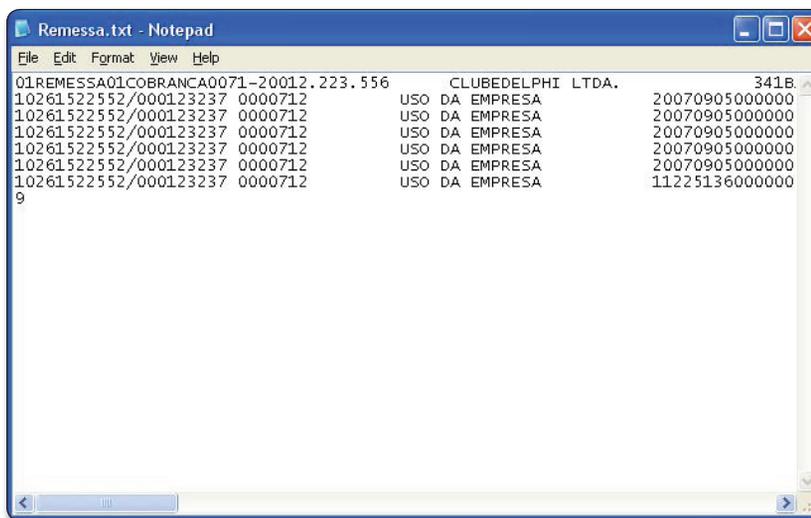


Figura 4. Remessa gerada com sucesso

