

FreeBoleto

Imprima você mesmo seus boletos bancários

Autor: Carlos Henrique Cantu – www.firebaseio.com.br – freeboleto.sf.net

Artigo originalmente publicado na revista ActiveDelphi.

Ooohhh não!!! Mais um componente de boleto???? Calma! Antes de tudo, permitam-me explicar o porquê de mais um componente de boleto para Delphi: Após testar diversos componentes disponíveis na internet, constatei que cada um deles tinha um problema que impedia ou dificultava o uso na forma que eu desejava.

O componente mais conhecido para essa finalidade é o *gbBoleto*. No entanto, encontrei dois problemas nele:

1. Dependência interna para o gerador de relatórios *QuickReport*, que eu felizmente não uso.
2. Aparentemente o projeto andava (ou ainda anda) meio abandonado e confuso. Quando testei, percebi que a geração dos boletos para alguns bancos estava fora do padrão ou simplesmente geravam dados inconsistentes.

Sendo assim, decidi criar eu mesmo um componente para geração de boletos, que atendesse minhas necessidades: fosse confiável, leve e com o mínimo de dependências. Assim, nasceu o **FreeBoleto**.

O principal objetivo desse artigo, além de apresentar o componente aos leitores, é de atrair mais pessoas para o desenvolvimento de novas *units* para os bancos que ainda não são suportados, pois o FreeBoleto é um projeto de **código aberto, gratuito**, e que depende da colaboração dos desenvolvedores para evoluir.

O FreeBoleto, em sua versão atual, suporta a emissão de boletos para os seguintes bancos:

- Itaú
- NossaCaixa
- Santander Banespa
- Unibanco
- Real
- Bradesco
- Caixa Federal
- Banco do Brasil
- Santander

Uma das grandes vantagens do FreeBoleto é que o componente de geração dos dados de cobrança (*TFreeBoleto*) **não depende** de qualquer componente externo ou de geradores de relatórios! O pacote oferece um segundo componente (*TFreeBoletoImagem*) que permite a geração da imagem do boleto para **visualização, impressão** ou **gravação** (em formato *jpg*). Gravar a imagem em *jpg* pode ser útil para quem deseja enviar os boletos por e-mail, ou mesmo em *cgis* para uso em servidores web.

Note que o FreeBoleto não gera, pelo menos por enquanto, arquivos de transação padrão CNAB. Isso poderá ser implementado no futuro (conforme houver necessidade), e provavelmente será através de novas classes, independentes das atuais.

Licença de uso

1. O FreeBoleto pode ser distribuído e utilizado livremente com qualquer tipo de projeto, comercial ou não.
2. Componentes derivados do código do FreeBoleto não podem ser vendidos, devem manter os créditos originais, e devem estar compatíveis com essa licença.
3. Qualquer alteração ou melhoria no código do FreeBoleto deve ser enviada ao autor para ser avaliada e, se possível, incorporada ao código oficial do componente.
4. A inclusão do suporte de novos bancos ao FreeBoleto deve ser notificada ao autor, enviando juntamente o código da *unit* do banco em questão. A criação de uma nova *unit* de suporte a um novo banco deverá ser feita em conjunto com a criação dos testes unitários (DUNITs) necessários para garantir o correto funcionamento das rotinas.
5. O autor não se responsabiliza por qualquer dano ou qualquer outro tipo de problema originado pela utilização desse componente, se isentando de qualquer responsabilidade sobre a utilização do mesmo.

Instalando o FreeBoleto

A instalação do FreeBoleto é muito fácil e rápida! A primeira coisa que deve ser feita é baixar o componente do site oficial: <http://freeboleto.sourceforge.net/>

Depois, basta abrir o arquivo DPK correspondente a sua versão do Delphi, compilar, e instalar na palheta de componentes (ver **figura 1**). O FreeBoleto roda em qualquer versão do Delphi (Win32) a partir do Delphi 7. O pacote atualmente traz os DPKs para Delphi 7, Delphi 2005 e Delphi 2006. Apesar de nunca ter testado, acredito que o componente compile também no Delphi 5 e no Delphi 2007, sem maiores problemas.



Figura 1. Componentes do FreeBoleto na palheta do Delphi 7

Vale mencionar que o componente *TFreeBoleto* - responsável por gerar as informações de cobrança - não utiliza componentes visuais, portanto é bastante provável que ele possa ser compilado e instalado no *Kylix* ou mesmo no *Delphi for .Net*.

Informações básicas sobre boletos

Para aqueles que não estão familiarizados com o tema, um boleto nada mais é do que um título de cobrança, que pode ser emitido por você (cedente) ou pelo próprio banco.

Para o banco (“entidade financeira altamente lucrativa”) poder processar corretamente os pagamentos, é necessário que você solicite ao seu gerente a ativação da “cobrança

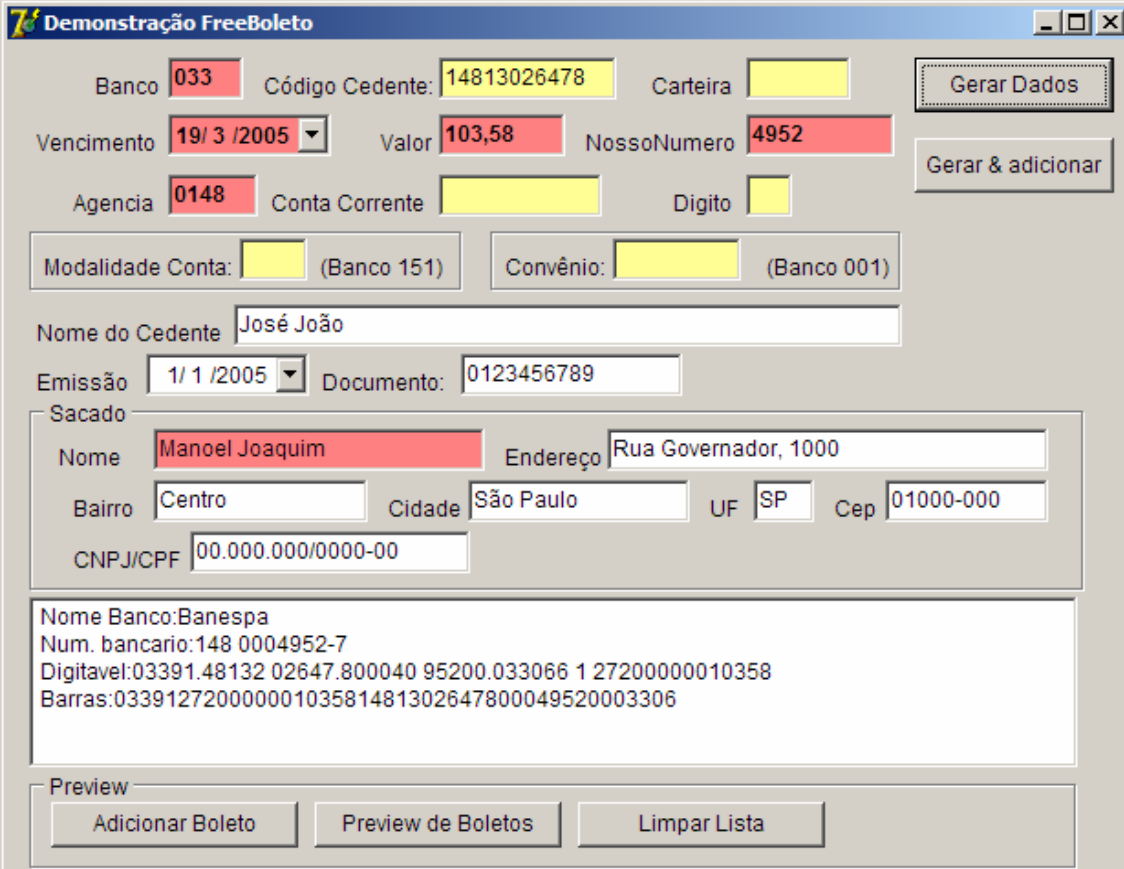
através de boletos” nas contas correntes que desejar. Serão solicitadas algumas informações, entre elas se a *carteira* utilizada será com ou sem registro. Carteiras *com registro* são aquelas onde os títulos são gerados (registrados) pelo banco, portanto o banco sabe da existência dos mesmos, e poderá inclusive enviá-los para protesto, caso não sejam pagos até o vencimento e assim for instruído. Carteiras *sem registro* são aquelas onde o banco só toma conhecimento da existência do boleto quando o mesmo é pago.

Todo boleto possui um campo (numérico inteiro) chamado *Nosso Número*. Esse campo é que identifica o boleto no banco e na empresa. O ideal é que este número nunca se repita, caso contrário vai ser difícil identificar qual boleto foi pago.

Quando um boleto é pago, o banco fornece uma relação dos títulos que foram liquidados, contendo o *Nosso Numero*, valor, data de baixa, etc. Com isso, você poderá identificar facilmente os títulos pagos.

Usando o FreeBoleto

A melhor forma de aprender a usar o FreeBoleto é estudar o projeto de demonstração (demo.dpr) que acompanha o pacote (ver **figura 2**).



A imagem mostra a interface gráfica do programa "Demonstração FreeBoleto". O formulário contém os seguintes campos e botões:

- Banco:** 033 (vermelho)
- Código Cedente:** 14813026478 (amarelo)
- Carteira:** (campo vazio, amarelo)
- Vencimento:** 19/ 3 /2005 (menu suspenso)
- Valor:** 103,58 (vermelho)
- NossoNumero:** 4952 (vermelho)
- Agencia:** 0148 (vermelho)
- Conta Corrente:** (campo vazio, amarelo)
- Digito:** (campo vazio, amarelo)
- Modalidade Conta:** (campo vazio, amarelo) (Banco 151)
- Convênio:** (campo vazio, amarelo) (Banco 001)
- Nome do Cedente:** José João
- Emissão:** 1/ 1 /2005 (menu suspenso)
- Documento:** 0123456789
- Sacado:**
 - Nome:** Manoel Joaquim (vermelho)
 - Endereço:** Rua Governador, 1000
 - Bairro:** Centro
 - Cidade:** São Paulo
 - UF:** SP
 - Cep:** 01000-000
 - CNPJ/CPF:** 00.000.000/0000-00
- Nome Banco:** Banespa
- Num. bancario:** 148 0004952-7
- Digitavel:** 03391.48132 02647.800040 95200.033066 1 27200000010358
- Barras:** 03391272000000103581481302647800049520003306

Botões de ação:

- Gerar Dados
- Gerar & adicionar
- Adicionar Boleto
- Preview de Boletos
- Limpar Lista

Figura 2. Tela inicial do programa de demonstração do FreeBoleto

Os campos indicados com fundo vermelho são obrigatórios. Os campos com fundo amarelo podem ou não ser obrigatórios, dependendo do banco para o qual está sendo

gerado o boleto. Por exemplo, o campo “*Modalidade da Conta*” só é exigido nos boletos da NossaCaixa. Já o campo “*Convênio*” é necessário para o Banco do Brasil.

Infelizmente, os bancos não gostam de facilitar a vida dos programadores. Cada um deles tem suas próprias necessidades e podem exigir informações diferentes para gerar os boletos. Felizmente, a maior parte das informações é comum entre os bancos.

Clicando no botão “*Gerar Dados*”, o código apresentado na **listagem 1** será executado. Basicamente, o código associa as informações digitadas nos “*edits*” às suas respectivas propriedades no componente TFreeBoleto (no exemplo, chamado de “*B*”).

O método **LimparTudo** “*reseta*” o componente para seu estado inicial. Como podemos usar uma única instância do componente “*B*” para gerar diversos boletos, é necessário reiniciá-lo antes de gerar um novo boleto, garantindo que as informações anteriores foram descartadas.

O método **Preparar** deve ser chamado depois que as informações necessárias para gerar o boleto já foram atribuídas ao componente. Esse método é que vai gerar a informação da *Linha Digitável* (propriedade *LinhaDigitavel*) e do *Código de Barras* (propriedade *CodigoBarras*). Ele também faz a checagem se todas as informações essenciais para a geração do boleto para o banco em questão estão presentes e no formato/tamanho esperado.

Diferente do que muita gente imagina, para pagar um boleto via Internet só é necessário a informação da *Linha Digitável*. Ou seja, só é necessário imprimir o boleto no papel caso pretenda-se paga-lo diretamente nos bancos, ou nos caixas eletrônicos (com leitor de código de barras). A *linha digitável* contém todas as informações necessárias (ex: valor, vencimento, *Nosso Número*, etc) para a correta identificação e baixa do título pelo banco.

Por fim, os dados gerados são inseridos no TMemo, para que possam ser visualizados na tela.

```
procedure TForm1.Button1Click(Sender: TObject);  
begin  
  B.LimparTudo;  
  B.Cedente.Agencia := edAgencia.text;  
  B.Cedente.CodigoBanco := edBanco.text;  
  B.Moeda := '9';  
  B.Cedente.Nome := edNomeCede.text;  
  B.DataDocumento := trunc(dtpEmissao.date);  
  B.Documento := edDocumento.text;  
  B.Vencimento := dtpVencimento.date;  
  b.Cedente.CodigoCedente := edCodCedente.text;  
  b.Valor := StrToFloat(edValor.Text);  
  b.NossoNumero := edNNum.text;  
  B.Cedente.ContaCorrente := edContaCorrente.text;  
  if edDigitoCC.text <> '' then  
    B.Cedente.DigitoContaCorrente := edDigitoCC.text[1];  
  B.carteira := edCarteira.text;  
  B.Cedente.Banco151.ModalidadeConta := edModeloCarteira.text;  
  B.Cedente.Banco001.Convenio := edConvenio.text;  
  with B.sacado do  
    begin
```

```

Nome := edSacado.text;
Endereco := edEndereco.text;
Bairro := edBairro.text;
Cep := edCep.text;
Estado := edUF.text;
Cidade := edCidade.text;
CNPJ_CPF := edCNPJ.text;
end;
B.preparar;
memo1.Lines.clear;
with memo1.lines do
begin
Add('AgenCodCedente:' + b.DadosGerados.AgencCodigoCedente);
Add('Nome Banco:' + b.DadosGerados.NomeDoBanco);
Add('Num. bancario:' + b.DadosGerados.NumeroBancario);
Add('Digitavel:' + b.DadosGerados.LinhaDigitavel);
Add('Barras:' + b.DadosGerados.CodigoBarras);
end;
end;

```

Listagem 1. Código associado ao botão Gerar Dados

Após gerar as informações do boleto, podemos clicar no botão “Adicionar Boleto”, cujo código pode ser visto na **listagem 2**, e que basicamente cria um novo objeto do tipo *TFreeBoleto* (chamado simplesmente de “x”) na memória, e copia as informações do objeto “B” para ele. A cópia é feita através de uma rotina especial chamada *CloneProperties* (ver **Listagem 3**). Essa rotina pode ser usada sempre que se desejar copiar as propriedades de qualquer objeto para outro do mesmo tipo.

Após feita a “clonagem” do componente “B” para o “X”, este último é adicionado à uma lista de boletos associada ao componente *Img* (do tipo *TFreeBoletoImagem*). O componente *Img* é responsável por gerar a imagem gráfica do boleto, possibilitando sua visualização, impressão ou gravação, bem como manter uma lista de boletos que poderão ser impressos de uma única vez, ou visualizados na tela de *preview*. Agora você entendeu porque todo esse trabalho de clonar os componentes, afinal, eles devem continuar existindo na memória para posterior impressão, até que não sejam mais necessários.

```

procedure TForm1.Button3Click(Sender: TObject);
var x:TFreeBoleto;
begin
x:=TFreeBoleto.create(nil);
CloneProperties(B,x);
Img.ListaBoletos.Add(x);
end;

```

Listagem 2. Código associado ao botão “Adicionar Boleto”

```

procedure CloneProperties(SourceComp, DestComp: TObject);
var
Propinfo: PPropInfo;
Properties: PPropList;
FCount: Integer;
FSize: Integer;
i: Integer;
PropName: String;
SourcePropObject: TObject;
DestPropObject: TObject;

```

```

begin
  FCount := GetPropList(SourceComp.ClassInfo, tkAny, nil);
  FSize := FCount * SizeOf(Pointer);
  GetMem(Properties, FSize);
  GetPropList(SourceComp.ClassInfo, tkAny, Properties);
  for i := 0 to FCount-1 do
    begin
      PropName := Properties^[i].Name;
      PropInfo := GetPropInfo(DestComp.ClassInfo, PropName);
      if (PropInfo = nil) or (UpperCase(PropName) = 'NAME') then
        Continue;
      case PropType(SourceComp, PropName) of
        tkInteger,
        tkWChar,
        tkSet,
        tkChar:
          SetOrdProp(DestComp, PropName, GetOrdProp(SourceComp, PropName));
        tkString, tkLString, tkWString:
          SetStrProp(DestComp, PropName, GetStrProp(SourceComp, PropName));
        tkEnumeration:
          SetEnumProp(DestComp, PropName, GetEnumProp(SourceComp, PropName));
        tkFloat:
          SetFloatProp(DestComp, PropName, GetFloatProp(SourceComp, PropName));
        tkClass:
          begin
            SourcePropObject := GetObjectProp(SourceComp, PropName);
            DestPropObject := GetObjectProp(DestComp, PropName);
            if (SourcePropObject <> nil) and (DestPropObject <> nil) and
              (SourcePropObject.ClassType.ClassParent.ClassName = 'TPersistent') then
              CloneProperties(SourcePropObject, DestPropObject)
            else
              SetObjectProp(DestComp, PropName, GetObjectProp(SourceComp, PropName));
            end;
          end;
        end;
      if Properties <> nil then FreeMem(Properties, FSize);
    end;
  end;
end;

```

Listagem 3. Código para copiar as propriedades de um objeto para outro

Após clicar no botão “Adicionar Boleto”, já é possível visualizar a imagem do boleto na tela. Para isso, basta clicar no botão “Preview de Boletos” (ver **figura 3**). Através da barra de navegação, podemos “percorrer” todos os boletos que tenham sido adicionados à lista, imprimi-los ou gravar suas imagens em arquivos.

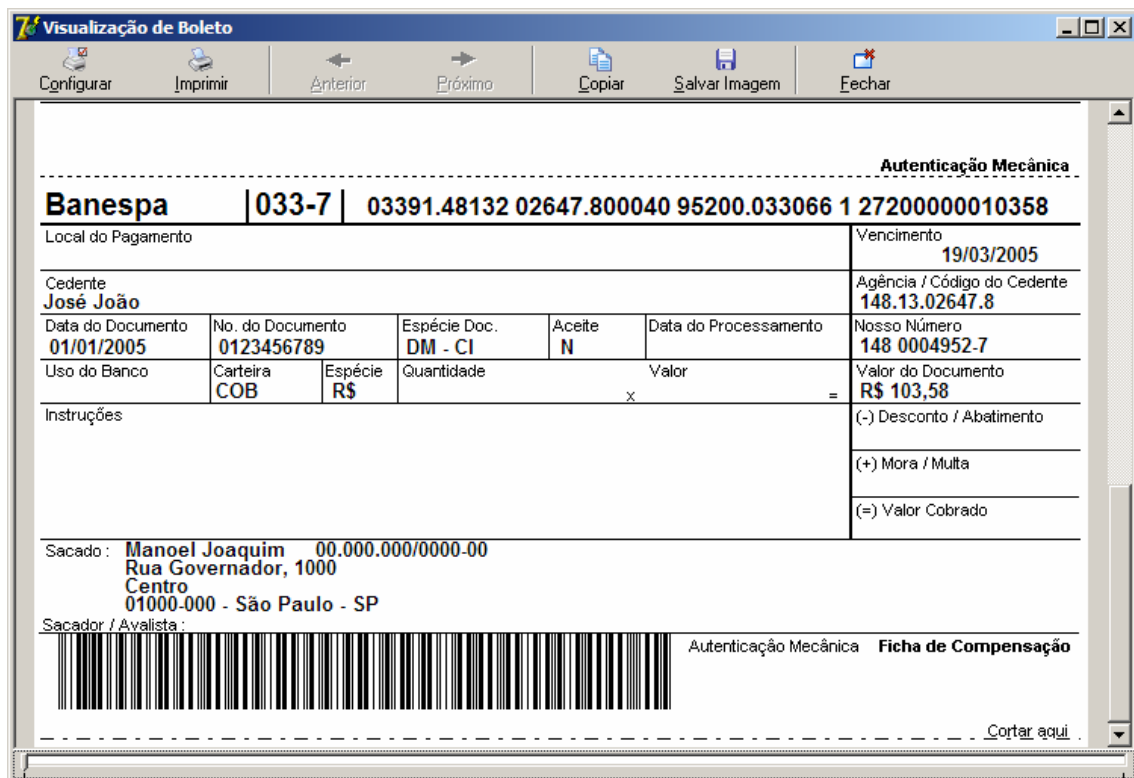


Figura 3. Imagem do boleto sendo visualizada na tela

Sendo assim, repetindo os passos anteriores, podemos gerar diversos boletos, adicioná-los a lista de boletos do componente `Img`, e depois visualizá-los na tela de *preview*. O botão “*Gerar e Adicionar*” pode ser usado para simplificar a operação, fazendo a função do botão “*Gerar*” e “*Adicionar Boleto*” em um único *click*.

O terceiro botão, “*Limpar lista*”, remove todos os componentes que foram adicionados à lista de boletos do componente `Img`, através da chamada `Img.ListaBoletos.Clear`;

O componente `Img` (classe `TBoletoImagem`) possui uma propriedade do tipo *boolean* chamada `DestruirBoletos`. Quando “*true*”, no momento em que a lista de boletos for destruída (através da chamada ao método `clear`), os boletos associados a lista serão também destruídos, liberando portanto a memória consumida. Se estiver “*false*”, em algum outro momento será necessário destruir os boletos para evitar os famosos (e indesejáveis) *memory leaks*.

Adicionando novos bancos

Se o seu banco preferido não faz parte da lista atual de bancos suportados pelo `FreeBoleto`, está aí a sua chance de contribuir para o projeto!

Criar uma nova *unit* para suportar um novo banco, do ponto de vista de programação, não é difícil. O pacote `FreeBoleto` já traz um modelo padrão de *unit* (no arquivo `uFreeBoletoBASE.pas`) que pode ser usada com base para gerar novas *units* para outros bancos. Leia os comentários do arquivo para entender como e o que deve ser feito. A primeira tarefa é solicitar ao banco o *manual de cobrança*, onde vai constar todas as

informações para gerar o *código de barras* e a *linha digitável* no padrão do banco em questão.

Desabafo: Até hoje, não encontrei um manual de cobrança que prestasse! São todos mal escritos! Alguns apresentam inconsistências onde os próprios exemplos apresentados contradizem algumas informações do próprio manual. Em suma, um horror! Ligar no “0800” para tirar alguma dúvida é tão caótico quanto conversar com alguém que não fale a sua língua.

Uma das preocupações ao criar o projeto foi garantir que mudanças feitas por outros usuários não “quebrassem” o código já existente e testado. Para isso, criei alguns casos de teste, também chamados de “testes unitários”, ou DUnits. Eles executam as principais rotinas dos componentes, e verificam se a informação retornada/gerada por elas foi a esperada, ou seja, está corretada. Se alguém fizer alguma alteração que altere as informações essenciais geradas, o teste vai “quebrar”, indicando que um possível erro foi detectado.

Sendo assim, peço que todos aqueles que criarem *units* para novos bancos, que criem também os respectivos casos de teste. Quem nunca fez isso não perderá mais que 10 minutos para estudar e entender o código dos testes já existentes. É realmente muito simples! Apesar de ser uma tarefa “sacal”, garanto que é um trabalho importante e que pode poupar muitos problemas (e horas de *debug*) no futuro.

Depois, basta enviar o código gerado para o meu *e-mail*, para que seja incluído no projeto e assim permita que outras pessoas se beneficiem das melhorias. Aqueles que se tornarem colaboradores freqüentes, poderão ganhar direito de *commit* diretamente no repositório do SourceForge.

A *unit* *uFreeBoleto.pas*, que contém o código base do principal componente (*TFreeBoleto*), geralmente não precisa ser alterada, a não ser pela inclusão da *unit* de um novo banco na sua cláusula *uses*.

Os boletos seguem um padrão de formatação para o *Código de Barras e Linha Digitável*, definido pela *Febraban*. No entanto, este padrão possui um *Campo Livre*, que é usado pelos bancos para armazenar qualquer tipo de informação que lhes interesse.

As *units* responsáveis pela geração das informações específicas de cada banco (*Campo Livre*) são nomeadas seguindo o padrão *uFreeBancoXXX.pas*, onde *XXX* é o número oficial do respectivo banco no Brasil (ex: Banco do Brasil = 001, Banespa = 033, etc). Estas *units* também contem o código de consistência das informações necessárias para gerar o boleto, bem como as rotinas de formatação de alguns dados, seguindo o padrão de cada banco.

Suporte

Desculpe, eu **não tenho** condições para dar suporte aos usuários do FreeBoleto, seja ele via *e-mail* ou qualquer outro método.

Para isso, foi criada uma lista de discussão no YahooGroups, onde os usuários poderão trocar experiências e solucionar suas dúvidas e possíveis problemas, lembrando que o

suporte da lista é voluntário, portanto não reclame se sua dúvida demorar a ser respondida, pois os que estão ali ajudando o fazem por livre e espontânea vontade, sem ganhar por isso. O endereço da lista é <http://br.groups.yahoo.com/group/freeboleto>

Não enviem dúvidas para meu e-mail pessoal, pois elas serão ignoradas.

Conclusão

Espero que esse artigo tenha sido útil, e que você tenha gostado do FreeBoleto. Independente de decidir usar ou não o componente, o código serve como uma boa base para aqueles programadores que ainda não dominam alguns conceitos de *orientação à objetos*, como herança, etc.

A proposta do componente é ser simples, leve, rápido, confiável e versátil. Como qualquer projeto *Open Source*, ele depende da ajuda de todos para evoluir e crescer. E que venham novos (e bons) desenvolvedores para o projeto ;-)

Up the Irons!

Carlos H. Cantu

www.firebaseio.com.br / www.warmboot.com.br

blog.firebaseio.com.br / freeboleto.sf.net

www.firebirdnews.org / www.firebirddevelopersday.com.br