UNIVERSIDADE REGIONAL DE BLUMENAU

CENTRO DE CIÊNCIAS EXATAS E NATURAIS

CURSO DE SISTEMAS DE INFORMAÇÃO – BACHARELADO

COMPONENTE DE GERAÇÃO DE BOLETO BANCÁRIO EM DELPHI

JONAS RICARDO VIEL

BLUMENAU
2009

2009/1-08

JONAS RICARDO VIEL

COMPONENTE DE GERAÇÃO DE BOLETO BANCÁRIO EM DELPHI

Trabalho de Conclusão de Curso submetido à Universidade Regional de Blumenau para a obtenção dos créditos na disciplina Trabalho de Conclusão de Curso II do curso de Sistemas de Informação— Bacharelado.

Prof. Adilson Vahldick – Orientador

BLUMENAU
2009

2009/1-08

COMPONENTE DE GERAÇÃO DE BOLETO BANCÁRIO EM DELPHI

Por

JONAS RICARDO VIEL

Trabalho aprovado para obtenção dos créditos na disciplina de Trabalho de Conclusão de Curso II, pela banca examinadora formada por:

Presidente:	Prof. Adilson Vahldick, M. Sc. – Orientador, FURB
Membro:	Prof. Paulo Fernando da Silva, M. Sc. - FURB
Membro:	Prof. Francisco Adell Péricas, M. Eng. - FURB

Blumenau, 08 de Julho de 2009

Dedico este trabalho as pessoas que me ajudaram diretamente na realização deste.

AGRADECIMENTOS

À minha família, que mesmo longe, sempre esteve presente.

A minha noiva, Poliana Maria Xavier, pelo apoio e compreensão que teve comigo durante esse semestre.

Ao meu orientador, Adilson Vahldick, por ter acreditado na conclusão deste trabalho.

Os bons livros fazem “sacar” para fora o que a pessoa tem de melhor dentro dela.

Lina Sotis Francesco Moratti

RESUMO

Este trabalho apresenta o desenvolvimento de dois componentes para geração e validação de boletos bancários de cobrança. Os componentes foram desenvolvidos em Delphi, utilizando *SQLConnection* para tratar de conectividade. No componente de geração do boleto de cobrança, são populadas as informações tais como número da agência bancária, conta corrente, nosso número, valor do documento, valor dos descontos, instruções de cobrança, e é retornado o boleto de cobrança gerado. O outro componente é um painel com dois campos para digitação do código de barras ou linha digitável do boleto de cobrança e faz a consistência desses campos, inclusive extraindo dados como nome do banco, representação numérica do boleto de cobrança, valor do documento, nosso número, data de vencimento e código do cedente.

Palavras-chave: Delphi. Componentes. Boleto de cobrança.

ABSTRACT

This work presents the development of two components for generation and validation of collection banking tickets. The components were developed in Delphi, using SQLConnection to negotiate of connectivity. In the component of generation of the collection ticket, they are populated the such information as number of the bank agency, checking account, our number, value of the document, value of the discounts, collection instructions, and is the generated collection ticket comes back. The other component is a panel with two fields for fingering of the code of bars or digitable line of the collection ticket and he/she makes the consistency of those fields, besides extracting data as name of the bank, numeric representation of collection tickets, value of the document, our number, expiration date and the grantee's code.

Word-key: Delphi. Components. Collection ticket.

LISTA DE ILUSTRAÇÕES

LISTA DE SIGLAS

DER – Diagrama Entidade Relacionamento

DD – Dicionário de Dados

EAN – European Article Numbering Association

FEBRABAN – Federação Brasileira de Bancos

FURB – Fundação Universidade Regional de Blumenau

POO – Pascal Orientado para Objetos

RF – Requisito Funcional

RNF – Requisito Não Funcional

UCC – *Uniform Code Council*

UPC – Universal Product Code

SUMÁRIO

1 INTRODUÇÃO

A cada dia que passa, as pessoas buscam comodidade, segurança e rapidez. Nas últimas décadas, as necessidades de aperfeiçoamento das condições de vida tem se concentrado nas melhores maneiras de se obter e utilizar informações e sistemas.

De acordo com Moretti (1999, p. 09), em 1974, os fabricantes e distribuidores de doze países europeus formaram um conselho para examinar a possibilidade de desenvolver um sistema padronizado de numeração de artigos para Europa, semelhante ao sistema de Código Universal de Produtos (UPC), já estabelecido nos Estados Unidos pelo *Uniform Code Council* (UCC). Como resultado, foi criada em 1977 uma entidade sem fins lucrativos, a *European Article Numbering Association* (EAN), começando com isso disponibilizar a leitura de produtos pelo código de barras.

Conforme Silva (1993, p. 12), atualmente, grande parte das transações em organizações varejistas e atacadistas é feita através do código de barras, permitindo o registro rápido e preciso de movimentos de venda e a gestão dos estoques, garantindo a melhor produtividade e qualidade.

Conforme Moretti (1999, p. 11), o código de barras é utilizado como padrão internacionalmente em 90 países: gera informações instantâneas e é uma linguagem comum no intercâmbio de informações entre parceiros comerciais.

Conforme GS1 Brasil (2008), o código de barras é uma forma de representar a numeração, que viabiliza a captura automática dos dados por meio de leitura óptica nas operações automatizadas.

A empresa Edusoft possui um sistema para a área da educação, gerenciando escolas e universidades. Ela tem como objetivo específico atender a parte acadêmica e financeira de uma instituição de ensino. Nessa empresa são desenvolvidos boletos de cobrança para vários bancos.

A cobrança bancária é feita através de boletos que substituem duplicatas, notas promissórias, letras de câmbio, recibos ou cheques e são aceitos como documento formal nos processamentos das transações bancárias. Os valores resultantes da operação de cobrança são automaticamente creditados na conta corrente do cliente.

Os dados dos títulos a serem cobrados são passados aos bancos através de meios magnéticos ou diretamente, via computador. O banco ou o cliente emitem os boletos de

cobrança aos sacados que poderão fazer o pagamento tanto pelo *Bankline*, como em agências e casas lotéricas. O procedimento de geração não pode ocasionar erros de informações nos boletos de cobrança, pois este fato poderá acarretar inclusive o crédito cobrado na conta corrente de outro cliente, resultando, entre outros transtornos para o sacado, de ter que entrar em contato com o creditado para negociar a devolução dos valores.

Com o propósito de agilizar o processo de validação dos boletos de cobrança e oferecer a garantia da aprovação dos mesmos na primeira vez, este trabalho propõe-se a criar um componente que auxiliará na validação e geração dos códigos de barras, sendo configurável para qualquer banco.

1.1 OBJETIVOS DO TRABALHO

O objetivo desse trabalho foi desenvolver um componente em Delphi para geração e validação de código de barras de boletos de cobrança para qualquer banco.

Os objetivos específicos do trabalho são:

a) desenvolver um componente para o Delphi, onde possa ser feito a geração do boleto de cobrança em lotes conforme regras de formatação do banco;
b) desenvolver um componente para o Delphi, onde possa ser feita a validação pela leitura magnética do código de barras e linha digitável.

1.2 ESTRUTURA DO TRABALHO

Este trabalho está estruturado em quatro capítulos.

No primeiro capítulo apresenta-se a introdução, objetivos e estrutura do trabalho.

No segundo capítulo, são apresentados assuntos relacionados a boletos de cobrança, documentação da FEBRABAN, componentização em Delphi, assim como um levantamento de trabalhos correlatos.

No terceiro capítulo é apresentado o processo de desenvolvimento da aplicação, sua estrutura, desenvolvimentos e interface, a metodologia e a aplicação dos componentes para geração e validação do boleto de cobrança.

No quarto capítulo é apresentada uma conclusão sobre os componentes desenvolvidos assim como o uso de componentização.

2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo apresenta conceitos referentes a cobrança bancária, componentização em Delphi e trabalhos correlatos.

2.1 COBRANÇA BANCÁRIA

Esta seção apresenta conceitos referentes ao código de barras, boletos de cobrança, tipos de cobrança e linha digitável.

2.1.1 CÓDIGO DE BARRAS

Na sociedade atual, o desejo das pessoas é sempre por mais comodidades e segurança. Nos supermercados, por exemplo, o funcionário do caixa procurava a etiqueta de preço de cada item e digitava o valor em sua máquina registradora, fazendo a soma. Muito usual também era ver uma empresa do comércio “fechada para inventário”, visto que o controle era praticamente todo manual e demandava muito tempo, espaço e pessoas. Apenas as lojas menores podiam se dar ao “luxo” de conhecer mais de perto os clientes: anotavam-se em sua ficha, ou na caderneta, os produtos comprados e os pagamentos realizados (GROSSMAN, ZYNGIER, 1991, p. 7).

De acordo com Moretti (1999, p. 12), o conceito básico para o funcionamento de um sistema de captura de dados por código de barras reside na aplicação correta de três elementos:

a) símbolo ou código de barras;
b) leitor (decodificação inclusa) do código;
c) processador da informação.

São estes três elementos combinados que responderão pelo sucesso da aplicação e o uso correto dos mesmos disponibilizará a informação certa na hora certa.

No quadro 1 é ilustrado a montagem do código de barras para a cobrança.

Posição	Tamanho	Conteúdo
01 – 03	3	identificação do banco
04 – 04	1	código da moeda (9 – real)
05 – 05	1	dígito verificador geral do código de barras
06 – 09	4	fator de vencimento
10 – 19	10	valor do documento
20 – 44	25	Campo livre

Fonte: FEBRABAN (2008, p.40).

Quadro 1 – Composição do código de barras.

O código de barras para a cobrança contém 44 posições dispostas da seguinte forma:

a) da posição um a três com tamanho de três posições é informado o código do banco. Cada banco possui seu próprio código do banco nunca sendo o mesmo código para bancos diferentes;
b) na posição 4 com tamanho de uma posição é enviado o código da moeda, que para esse caso é sempre enviado o valor padrão nove indicando a moeda *real*;
c) na posição 5 com tamanho de uma posição é enviado o dígito verificador geral do código de barras calculado conforme módulo 11;
d) da posição 6 a posição 9 do código de barras é informado o fator de vencimento, que nada mais é que o resultado da subtração entre a data do vencimento do título e a DATA BASE, fixada em 07.10.1997 (03.07.2000 retrocedidos 1000 dias do início do processo);
e) na posição 10 a 19 é informado o valor do documento;
f) da posição 20 a 44 é chamado de campo livre, onde cada banco com seu tipo de cobrança escolhido decidirá o que deve ser mandado nessas 25 posições do campo livre.

2.1.2 BOLETO DE COBRANÇA

Conforme a FEBRABAN (2008, p. 24), o boleto bancário é um título executivo de cobrança, pagável em qualquer agência bancária, *homebanking*, casas lotéricas, supermercados e agências dos correios existentes em território brasileiro. Os personagens

principais do relacionamento da cobrança através de boleto bancário são: banco, sacado e cedente.

a) banco: instituição financeira responsável pela cobrança, ou seja, desde que seja previamente contratado pelo cedente, o banco receberá o pagamento do referido documento de cobrança pago pelo sacado, e creditará a importância paga na conta corrente do cedente;
b) sacado: quem faz o pagamento do boleto de cobrança;
c) cedente: quem emite a cobrança, ou seja, aquele que irá receber a quantia cobrada.

O boleto deve conter duas partes que são o recibo do sacado e ficha de compensação. O layout e os dados a serem apresentados neste recibo serão definidos pelo cliente, salvo quando o mesmo for pré-estabelecido pelo banco. O recibo do sacado fica em poder do sacado para que ele possa, sempre que necessário, comprovar o pagamento do seu documento de cobrança (FEBRABAN, 2008).

Na confecção da ficha de compensação, o cedente poderá utilizar o layout que desejar, porém não pode deixar de ter os seguintes campos:

a) nome do cedente;
b) agência/ código do cedente;
c) valor do título;
d) vencimento;
e) nosso número;
f) nome do sacado;

A ficha de compensação fica em poder do banco para que seja encaminhado para a compensação bancária, onde o boleto é processado, devendo ter obrigatoriamente as seguintes dimensões:

a) altura - mínima de 95mm e máxima de 108mm;
b) largura - mínima de 210mm para o papel tipo A4.

2.1.3 TIPOS DE COBRANÇA

Os bancos dispõem em geral de dois tipos de cobrança, sem registro e registrada, sendo que esta nomenclatura pode variar de banco para banco.

As cobranças sem registro caracterizam-se pelo recebimento de cobranças sem o prévio registro dessas informações junto ao sistema bancário. Sendo assim, o banco irá receber as cobranças, creditar o valor correspondente e debitar automaticamente a taxa de cobrança. O próprio cedente pode efetuar o preenchimento, emitindo, enviando e especificando no boleto o banco cobrador. Assim sendo, o cedente poderá emitir os próprios títulos, e o banco na ocasião do recebimento dos mesmos, irá comunicar ao cedente, os dados referentes aos títulos recebidos e o valor creditado na conta corrente (FEBRABAN, 2008).

A cobrança registrada é aquela em que as informações a serem cobradas são enviadas previamente ao banco através de arquivos de remessa em diversos padrões, sendo o mais utilizado conhecido como cnab, que pode ser processado tais como: impressão e postagem de boletos, serviço de protesto, serviço de controle de recebimento da cobrança, solicitações de baixas e alterações entre outros.

Alguns bancos possuem o padrão CNAB 400 e outros CNAB 240, existindo também outros padrões de layouts. Para utilizar a cobrança, o cedente deve remeter o arquivo para o banco. Deve informar-se com seu gerente de contas qual é o tipo de arquivo adotado para sua conta (FEBRABAN, 2008).

2.1.4 LINHA DIGITÁVEL

No quadro 2 é apresentado um exemplo da composição da linha digitável do boleto de cobrança.

1º bloco	**2º bloco**	**3º bloco**	**dv geral**	**vencimento\valor**
10499.00127	00200.001287	70000.000128	1	10990000016000
Campo 1	Campo 2	Campo 3	Campo 4	Campo 5

Fonte: FEBRABAN (2008, p.45).

Quadro 2 – Composição da linha digitável do boleto de cobrança.

A linha digitável é formada pela seqüência de números que ficam na parte superior do boleto de cobrança, com os mesmos dados do código de barras, mas em posições diferentes, que servem para fazer o pagamento dos boletos de cobrança, quando a leitora de código de barras apresentarem problemas na leitura.

A montagem da linha digitável é padrão para todos os boletos de cobrança e funciona da seguinte forma:

a) 1º campo: composto por: código banco (posições 1 a 3 do código de barras), código da moeda (posição 4 do código de barras), as cinco primeiras posições do campo livre (posições 20 a 24 do código de barras) e dígito verificador deste campo;
b) 2º campo: composto pelas posições 6 a 15 do campo livre (posições 25 a 34 do código de barras) e dígito verificador deste campo;
c) 3º campo: composto pelas posições 16 a 25 do campo livre (posições 35 a 44 do código de barras) e dígito verificador deste campo;
d) 4º campo: dígito verificador geral do código de barras (posição 5 do código de barras);
e) 5º campo: composto pelo "fator de vencimento" (posições 6 a 9 do código de barras) e pelo valor nominal do documento (posições 10 a 19 do código de barras), com a inclusão de zeros entre eles ate compor as 14 posições do campo e sem edição (sem ponto e sem vírgula).

Quando se tratar de bloquetos sem discriminação do valor no código de barras a representação deve ser com zeros. Os três primeiros campos devem ser editados, após as cinco primeiras posições, com um ponto. Os dados da representação numérica não se apresentam na mesma ordem do código de barras, mas sim de acordo com a seqüência descrita acima. Os dígitos verificadores referentes aos campos 1, 2 e 3 não são representados no código de barras (FEBRABAN, 2008).

2.2 COMPONENTIZAÇÃO EM DELPHI

Conforme Anselmo (1997, p. 159), uma das maiores vantagens do Delphi sobre os demais concorrentes é o fato da geração de novos componentes (de novos objetos).

Componentes são como blocos de construção para as aplicações Delphi. Pode ser construído uma aplicação simplesmente adicionando estes blocos e modificando os eventos, propriedades ou métodos. Todos os componentes possuem duas propriedades em comum: `Name` e `Tag`. Alguns componentes estão distribuídos na *Component Pallete*. Componentes do

tipo `TApplication`, `TMenu`, `TMenuItem`, e `TScreen` são disponíveis apenas através de seu código. A *Component Pallete* são todas as abas onde ficam todos os componentes disponíveis para uso do programador no ambiente Delphi.

Para criação de novos componentes são utilizados os seguintes passos:

a) derivando os novos componentes de um componente já existente;

b) modificando um componente;

c) registrando um componente.

O componente é criado como uma *unit* separada de um projeto, podendo ser formado por uma ou mais *units*. Para usar o componente, selecione-o da *Component Pallete* e adicione-o ao formulário (ANSELMO, 1997, p. 160).

2.2.1 TCOMPONENT

Para criação de um novo componente utilizando diretamente o *Code Editor*, é necessário usar a *Component Expert*. Na verdade todos os componentes criados serão derivados de componentes já existentes, mesmo que seja necessário criar um componente sem eventos ou propriedades ele será herdado de uma classe já existente a `TComponent`.

A `TComponent` é uma classe inicial de componentes. Sob ela é que foi feita a árvore de componentes Delphi. Por exemplo, a classe `TControl` possui mais de 70 componentes descendentes, tais como: `TBitBtn`, `TButton`, `TCheckBox`, `TColorDialog`, `TComboBox`, `TForm`, `TFontDialog`, `TGroupBox`, `THeader`, `TImage`, `TLabel`, `TListBox`, `TMainMenu` e `TMediaPlayer`. E ainda pode derivar mais alguns descendentes (ANSELMO, 1997, p. 160).

2.2.2 UM COMPONENTE SIMPLES

Para criar um novo componente, deve-se abrir um novo projeto e selecionar a opção *File – New*, selecionar a página *New* e o item *Component*. Será mostrada a janela da *Component Expert* conforme Figura 1.

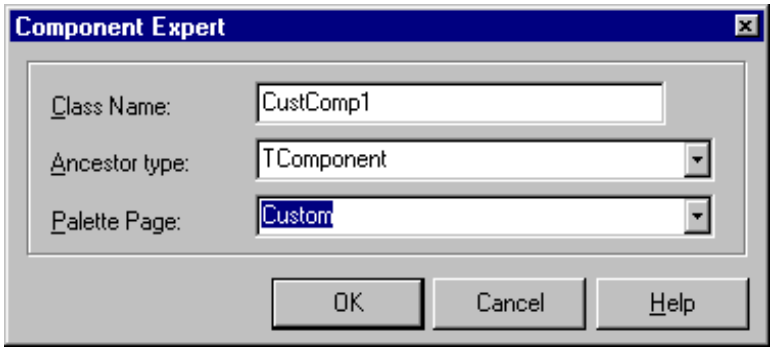

Figura 1 – Criação de novo componente

Após clicar no botão *OK* para aceitar a entrada, a *Component Expert* criará automaticamente o código para a chamada da *Unit1* conforme Quadro 3.

```
unit Unit1;

interface

uses
    Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms,
Dialogs;

type
  TCustComp1 = class(TComponent)
  private
      { Private declarations }
  protected
      { Protected declarations }
  public
      { Public declarations }
  published
      { Published declarations }
  end;

procedure Register;

implementation

procedure Register;
begin
    RegisterComponents('Custom', [TCustComp1]);
end;

end.
```

Quadro 3 – Criação de novo componente

2.2.3 CRIANDO COMPONENTES

No componente, pode-se adicionar uma propriedade que armazenará um valor inteiro, para esse processo, deve-se chamar novamente a unit do componente e inserir os códigos na declaração *private* conforme Quadro 4.

```
type
   TCustComp1 = class(TComponent)
   private
      fDemoProp:Integer;
```

Quadro 4 – Adicionando nova propriedade

No Quadro 5 é ilustrado a criação de uma propriedade onde o valor será lido e escrito através desta variável na declaração *published*.

```
  Published
      property DemoProp: Integer read fDemoProp write fDemoProp;
  end;
```

Quadro 5 - Declaração da propriedade

A seção *Public* (pública) abriga as variáveis, procedimentos ou funções que podem ser lidos e executados por quaisquer outras units que utilizem (através cláusula Uses) a unit em questão. Já a seção *Published* (Publicado) é utilizada para inserir propriedades ou eventos aos componentes. De modo semelhante aos já descritos uma propriedade pode abrigar uma lista de tipos definidos, para tanto na seção `type` pode ser definido o conjunto que abrigará os tipos conforme Quadro 6.

```
Type
   TDirecao = (drCima, drBaixo, drLado);
```

Quadro 6 - Criado novos tipos

Na seção `private` pode ser criado uma nova variável com base no tipo definido conforme ilustrado no Quadro 7.

```
Private
    fDemoProp:Integer;
    fNovaProp:TDirecao;
```

Quadro 7 - Definição do tipo

E finalmente na seção `published` é definido a propriedade conforme Quadro 8.

```
Published
    property DemoProp: Integer read GetDemoProp write SetDemoProp;
    property NovaProp: TDirecao read fNovaProp write fNovaProp;
```

Quadro 8 - Publicando as propriedades

No componente, pode ser criados eventos conforme ilustrado no quadro 9.

```
Type
  TResultValidacao = procedure (Sender: TObject; ANomBan, ARepresNum,
   AValor, ANosNum, ADvNosNum, ADatVct, ACedente: string) of object;
Private
  FResultValidacao : TResultValidacao
Published
  property OnResultValidacao : TResultValidacao read FResultValidacao
write FResultValidacao;
```

Quadro 9 - Criando evento no componente

Na seção `Published` é declarado o nome que irá aparecer na aba de eventos do componente.

2.3 TRABALHOS CORRELATOS

Moretti (1999) descreveu um trabalho de conclusão de curso para a Universidade Regional de Blumenau (FURB) sobre código de barras. O trabalho começa definindo os benefícios apontados no boleto de cobrança com código de barras, como a diminuição no tempo nas filas, maior controle tanto de pagamentos como de recebimentos, segurança e confiança nas informações obtidas.

A empresa Cobrebem disponibiliza uma ferramenta similar que faz a geração dos boletos de cobrança. Para poder usar o sistema de geração de boletos de cobrança, é cobrado uma licença de uso por cedente. O sistema Cobrebem não possui todos os tipos de cobrança requeridos pelo Banco do Brasil e Caixa Econômica Federal. O desenvolvimento de novos tipos de cobrança são feitos mediante solicitação, e está sujeito a pagamentos de taxas adicionais e restrição de tempo para a entrega (COBRE BEM, 2007).

3 DESENVOLVIMENTO

Este capítulo apresenta os requisitos dos componentes, bem como sua especificação, implementação e resultados obtidos. São mostrados diagramas de componentes e modelo de caso de uso. Também serão apresentadas explicações sobre o desenvolvimento dos componentes apresentados.

3.1 REQUISITOS PRINCIPAIS DO PROBLEMA A SER TRABALHADO

Os requisitos funcionais (RF) e não-funcionais (RNF) da suíte de componentes são:

a) disponibilizar uma interface para cadastrar o banco (RF);
b) disponibilizar uma interface para cadastrar tipos de cobrança (RF);
c) disponibilizar uma interface para cadastrar a montagem do código de barras (RF);
d) disponibilizar um componente visual para digitação dos dados da linha digitável ou código de barras (RF);
e) disponibilizar uma interface para receber tanto os valores da leitura magnética do código de barras como a linha digitável para fazer a validação (RF);
f) os componentes serão desenvolvidos para serem usados na ferramenta Delphi 6 (RNF);
g) o componente utilizará tecnologia SQLConnection para conectar-se a um banco de dados relacional (RNF).

3.2 ESPECIFICAÇÃO

Para a especificação foram efetuados diagramas pertencentes à linguagem de modelagem UML, sendo estes criados com a ferramenta Enterprise Architect. Tem-se nesta seção o modelo de caso de uso e o diagrama de componentes.

3.2.1 Modelo de casos de uso

Para Martins (2002, p. 45) o modelo de casos de uso é de extrema importância, pois força os desenvolvedores a modelar o sistema de acordo com o usuário, e não o usuário de acordo com o sistema. Esse diagrama é composto por casos de uso, atores e relacionamento entre eles.

Os casos de uso da suíte de componentes desenvolvidas são:

a) cadastrar banco (UC01);
b) cadastrar tipo de cobrança (UC02);
c) configurar regras de definição do código de barras (UC03);
d) fazer um formulário que popule dados para o componente (UC04);
e) fazer uma interface para receber tanto os valores da leitura do código de barras como a linha digitável para fazer a validação (UC05);
f) montar layout de um boleto bancário (UC06).

A Figura 2 apresenta o diagrama de casos de uso.

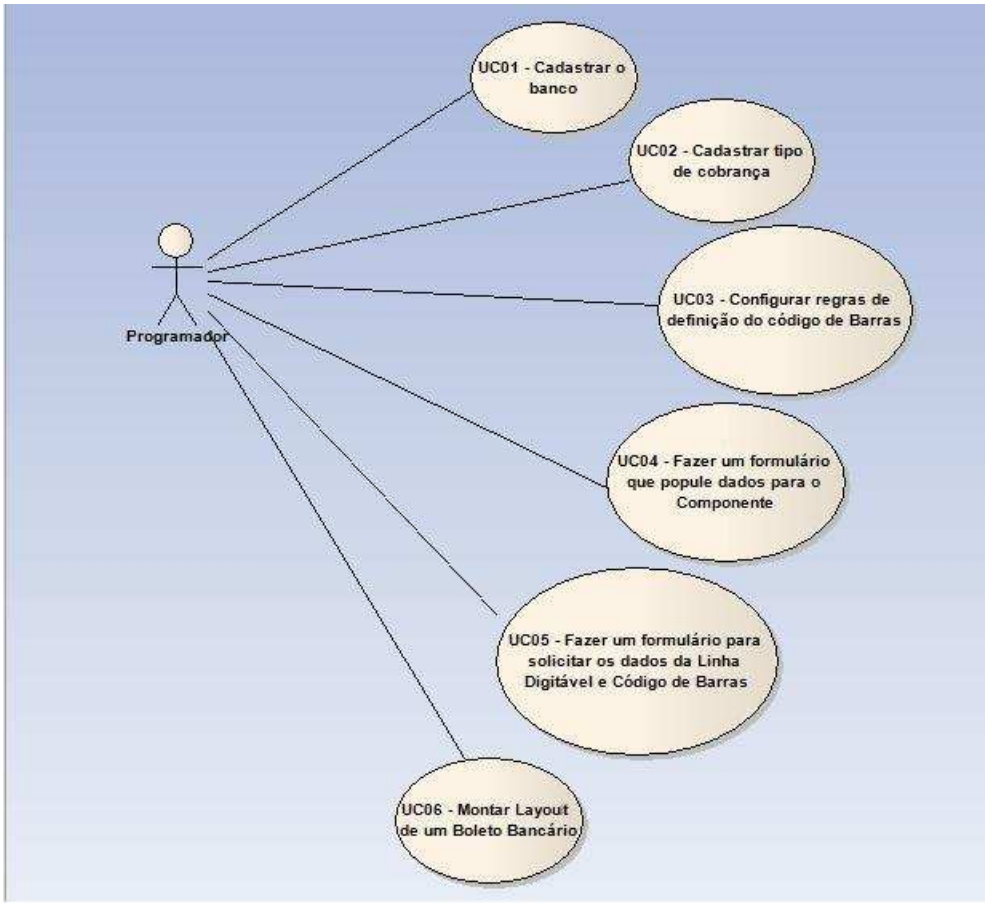

Figura 2 – Diagrama de casos de uso

O Quadro 10 apresenta o caso de uso cadastrar um banco.

UC01 – Cadastrar um banco

Sumário: Usuário cadastra o banco.

Ator Primário: Usuário.

Precondições: Estar conectado a uma base de dados com as tabelas necessárias para cadastramento do banco.

Fluxo Principal
1. O usuário informa um código único para o banco.
2. O sistema exibe tela de cadastramento do banco.
3. O usuário informa a descrição e o email referente ao banco a ser cadastrado.
4. O sistema grava o banco e o caso de uso termina

Pós condições: Banco cadastrado

Quadro 10 – Descrição do caso de uso UC01 – Cadastrar um banco

O Quadro 11 apresenta o caso de uso cadastrar tipo de cobrança.

UC02 – Cadastrar um tipo de cobrança

Sumário: Usuário cadastra o tipo de cobrança.

Ator Primário: Usuário.

Precondições: Estar conectado a uma base de dados com as tabelas necessárias para cadastramento do banco e ter pelo menos um banco cadastrado.

Fluxo Principal
1. O usuário informa um código único para o tipo de cobrança.
2. O sistema exibe tela de cadastramento do tipo de cobrança.
3. O usuário informa a descrição, agência, conta corrente, código do cedente, carteira, nosso número e banco referente ao tipo de cobrança a ser cadastrado.
4. O sistema grava o tipo de cobrança e o caso de uso termina

Pós condições: Banco cadastrado.

Quadro 11 – Descrição do caso de uso UC02 – Cadastrar um tipo de cobrança

O Quadro 12 apresenta o caso de uso configurar regras de definição do código de barras.

UC03 – Configurar regras de definição do código de barras

Sumário: O Programador configura as regras de definição do código de barras.

Ator Primário: Programador.

Precondições: Estar conectado a uma base de dados com as tabelas necessárias para cadastramento da regra de definição do código de barras e ter pelo menos um banco e um tipo de cobrança cadastrado.

Fluxo Principal
1. O programador informa o banco e o tipo de cobrança para definir a regra do código de barras.
2. O programador configura a nova regra conforme layout do banco, informando posição inicial e final de cada informação que deverá constar no código de barras.
3. O componente salva a regra.

Pós condições: Regra de definição do código de barras cadastrado

Quadro 12 – Descrição do caso de uso UC03 – Cadastrar regras de código de barras

O Quadro 13 apresenta o caso de uso fazer um formulário que popule dados para o componente.

UC04 – Fazer um formulário que popule dados para o componente.

Sumário: O Programador desenvolve um formulário onde popule os dados que o componente necessita para a geração do boleto bancário.

Ator Primário: Programador.

Precondições: Possuir pelo menos uma regra de definição do código de barras.

Fluxo Principal
1. O programador desenvolve um formulário para popular os dados que o componente necessita para a geração do boleto bancário.
2. O componente solicita os dados informados no formulário para geração do boleto bancário.
3. O programador informa esses dados para geração do boleto bancário.
4. O componente devolve os dados do boleto gerado.

Pós condições: Boleto bancário gerado.

Quadro 13 – Descrição do caso de uso UC04 –Popular dados para o componente

O Quadro 14 apresenta o caso de uso fazer um formulário para solicitar os dados da linha digitável e código de barras.

UC05 – Fazer o Formulário para solicitar os dados da linha digitável e código de barras.
Sumário: O Programador desenvolve um formulário onde recebe os dados da linha digitável e código de barras. **Ator Primário**: Programador. **Precondições**: Possuir pelo menos um boleto bancário gerado. **Fluxo Principal** 1. O programador desenvolve um formulário para receber as informações da leitura da linha digitável ou código de barras. 2. O programador associa o evento do componente com o formulário onde será mostrado o resultado das informações do validador. **Pós condições:** Dados da linha digitável e código de barras.

Quadro 14 – Descrição do caso de uso UC05 – formulário para solicitar os dados da linha digitável e código de barras.

O Quadro 15 apresenta o caso de uso montar Layout de um Boleto Bancário.

UC06 – Montar Layout de um Boleto Bancário.
Sumário: O Programador disponibiliza um layout de um boleto bancário para receber os dados gerados do componente. **Ator Primário**: Programador. **Precondições**: Possuir os dados de um boleto bancário gerado. **Fluxo Principal** 1. O programador desenvolve um layout para receber as informações do boleto bancário gerado. 2. O componente apresenta os dados gerados para informar no layout desenvolvido. 3. O programador repassa os dados gerados pelo componente para o layout do boleto bancário da aplicação. **Pós condições:** Boleto bancário montado para ser confeccionado.

Quadro 15 – Descrição do caso de uso UC06 – montar Layout de um Boleto Bancário.

3.2.2 Diagrama Entidade Relacionamento

Segundo Martins (2006, p.180), o diagrama Entidade Relacionamento (DER), é o modelo diagramático que descreve o modelo de dados de um sistema com alto nível de abstração. Sua maior aplicação é para visualizar o relacionamento entre tabelas de um banco de dados, no qual as relações são construídas através da associação de um ou mais atributos destas tabelas.

A Figura 3 ilustra o diagrama de entidade relacionamento (DER) da aplicação, exibindo as entidades, seus respectivos atributos e os relacionamentos. O componente de geração de boletos necessita da criação das tabelas para o funcionamento do mesmo.

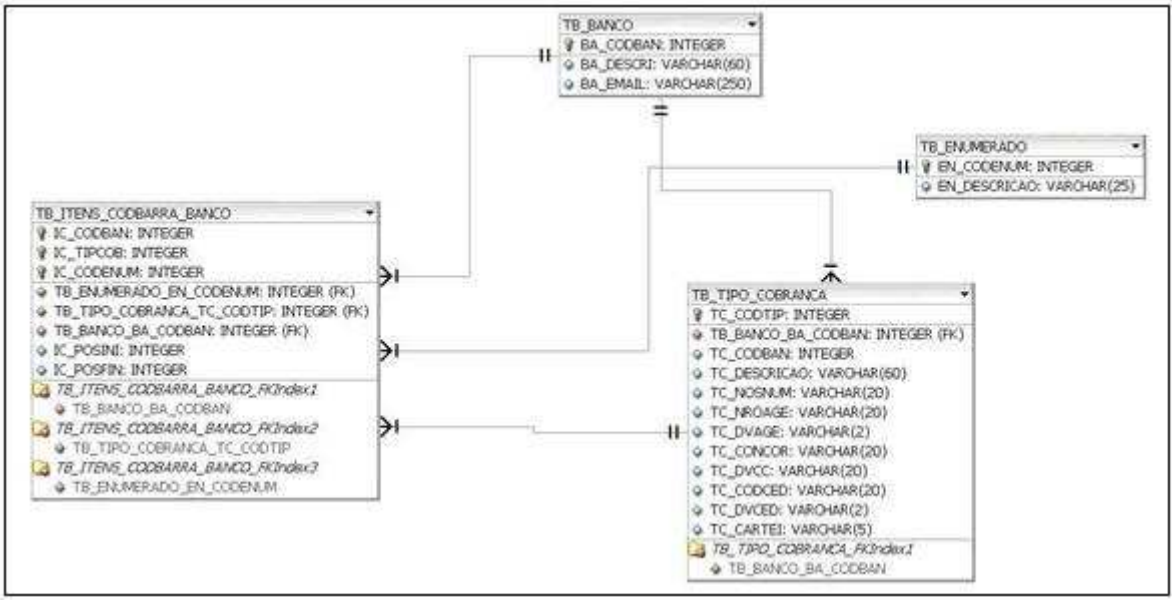

Figura 3 – Diagrama de entidade relacionamento da aplicação

3.2.3 Dicionário de Dados

Conforme Martins (2006, p.175), dicionário de dados (DD) é uma visão organizadora de todos os elementos de dados pertinentes ao sistema, com definições precisas e rigorosas, que complementam o modelo gráfico descrevendo e detalhando o conteúdo de cada componente do Diagrama Entidade Relacionamento (DER). Os quadros 16 a 19 descrevem o dicionário de dados e sua aplicação.

Descrição	**Cód. Atributo**	**Tipo**	**Explicação**
Código do banco	BA_CODBAN	Integer	Identificação do banco
Descrição	BA_DESCRI	Varchar(60)	Nome do banco
Email	BA_EMAIL	Varchar(250)	Email do banco

Quadro 16 – Descrição da Entidade Banco

Descrição	Cód. Atributo	Tipo	Explicação
Código do Enumerado	EN_CODENUM	Integer	Código do enumerado
Descrição	EM_DESCRICAO	Varchar(25)	Nome do enumerado

Quadro 17 – Descrição da Entidade Enumerado

Descrição	Cód. Atributo	Tipo	Explicação
Código do banco	IC_CODBAN	Integer	Chave Estrangeira entidade TB_BANCO
Código Tipo de Cobrança	IC_TIPCOB	Integer	Chave Estrangeira entidade TB_TIPO_COBRANCA
Código Enumerado	IC_CODENUM	Integer	Chave estrangeira entidade TB_ENUMERADO
Posição Inicial	IC_POSINI	Integer	Posição inicial do campo no código de barras
Posição Final	IC_POSFIN	Integer	Posição final do campo no código de barras

Quadro18 – Descrição da Entidade Itens_Codbarra_Banco

Descrição	Cód. Atributo	Tipo	Explicação
Código Tipo de Cobrança	TC_CODTIP	Integer	Chave Estrangeira entidade TB_TIPO_COBRANCA
Código do banco	TC_CODBAN	Integer	Chave Estrangeira entidade TB_BANCO
Descrição	TC_DESCRICAO	Varchar(60)	Descrição do tipo de cobrança
Nosso número	TC_NOSNUM	Varchar(20)	Nosso Número do banco
Número da agência	TC_NROAGE	Varchar(20)	Número da agência cobradora
Dígito da agência	TC_DVAGE	Varchar(2)	Dígito verificador da Agência
Conta corrente	TC_CONCOR	Varchar(20)	Conta corrente
Dígito conta corrente	TC_DVCC	Varchar(2)	Dígito da conta corrente
Código do cedente	TC_CODCED	Varchar(20)	Código do cedente
Dígito cedente	TC_DVCED	Varchar(2)	Dígito do código do cedente
Carteira	TC_CARTEI	Varchar(5)	Carteira

Quadro 19 – Descrição da Entidade Tipo_Cobranca

3.2.4 Diagrama de componentes

Conforme Martins (2006, p.169), o diagrama de componentes representa os elementos de softwares que serão criados como programas executáveis, tabelas de banco de dados, componentes OCX, applets e outros. O diagrama mostra a relação de interdependência e de comunicação entre estes elementos.

Na Figura 4 é apresentado o diagrama de componentes da aplicação.

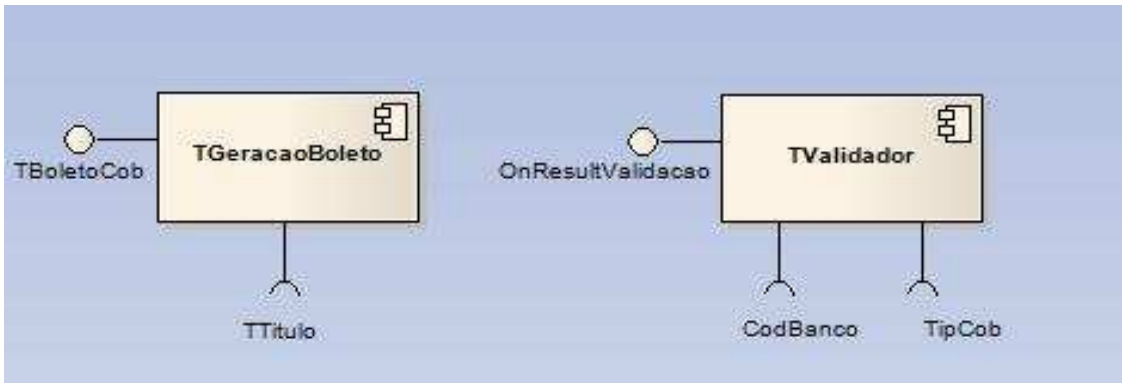

Figura 4 – Diagrama de componentes

O componente `TGeracaoBoleto` requer como entrada de dados a classe `TTitulo`. Essa classe `TTitulo` possui todas as informações para fazer a geração de um boleto bancário. Esse componente tem como saída a classe `TBoletoCob`, contendo todas as informações de um boleto de cobrança.

O componente `TValidador` requer como entrada de dados o código do banco e o tipo de cobrança. Com essas duas informações o componente carrega a regra de definição do código de barras e faz a leitura dos dados da linha digitável ou código de barras informado.

3.2.5 Diagrama de Classes

Conforme Martins (2006, p.159), o diagrama de classes é utilizado para representar a estrutura estática do sistema, composta pelas classes de negócio, classes de interface com o usuário e com outros sistemas, e as classes de controle, responsáveis pelo controle de transações. A figura 5 representa o diagrama de classe dessa aplicação.

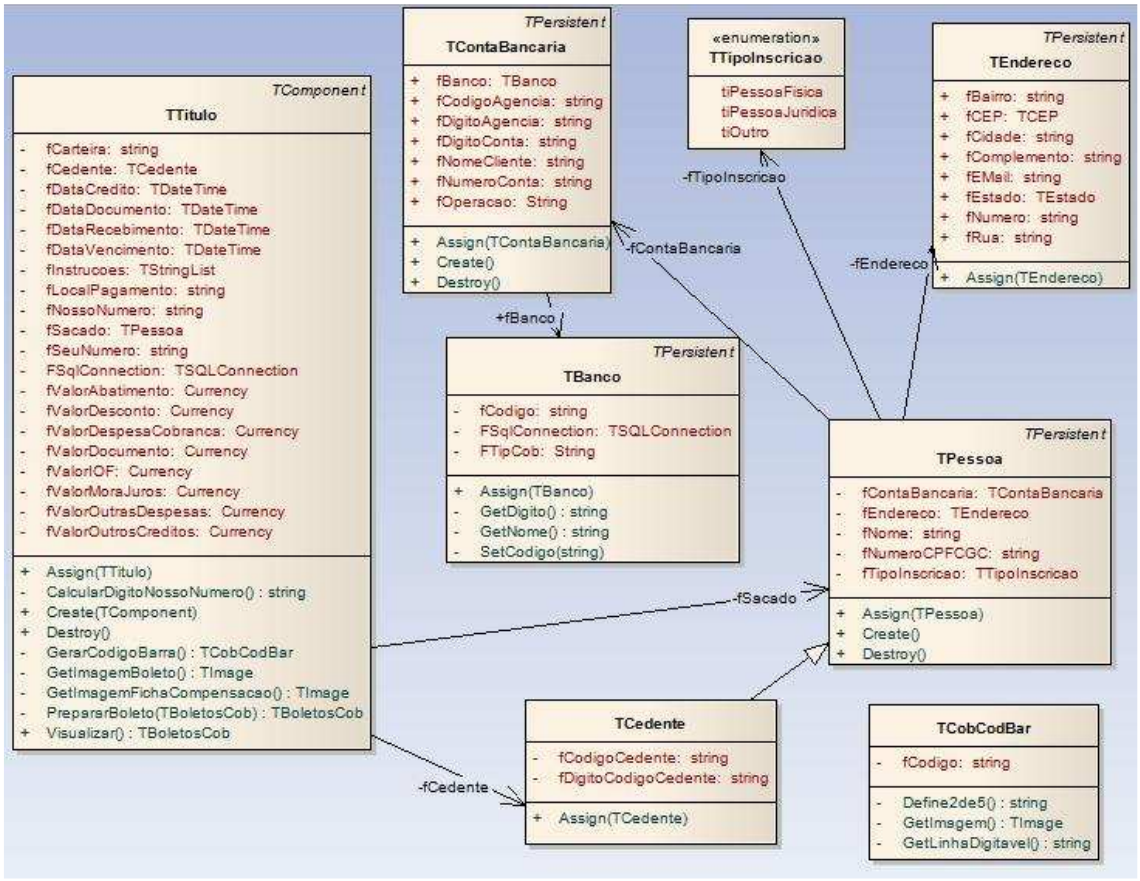

Figura 5 – Diagrama de classes

O componente `TGeracaoBoleto` utiliza as classes ilustradas na figura 5. Ele necessita que a classe `TTitulo` seja populada para poder gerar o boleto de cobrança. Um dos atributos da classe `TTitulo` é o `FCedente`, que é uma associação com `TCedente`. Um título não pode ser gerado se não possuir um cedente. A Classe `TCedente` possui herança da classe `TPessoa`, pois um cedente pode ser tanto uma pessoa física como uma pessoa jurídica. A classe `TPessoa` possui associação com `TContaBancaria`, onde necessita de uma conta bancária. Possui associação também com a classe `TTipoInscricao` podendo ser uma pessoa física, pessoa jurídica ou outros. A classe `TPessoa` possui associação também com a classe `TEndereco`, pois deve constar no atributo `FSacado` da classe `TTitulo` o endereço do sacado. A classe `TContaBancaria` tem associação com a classe `TBanco`, pois para ter uma conta bancária é necessário ter um banco.

Após a classe `TTitulo` ter recebido todas as informações necessárias e feito todos os cálculos para a geração do boleto de cobrança, é retornado um objeto chamado `TBoletosCob` com todas as informações conforme figura 6.

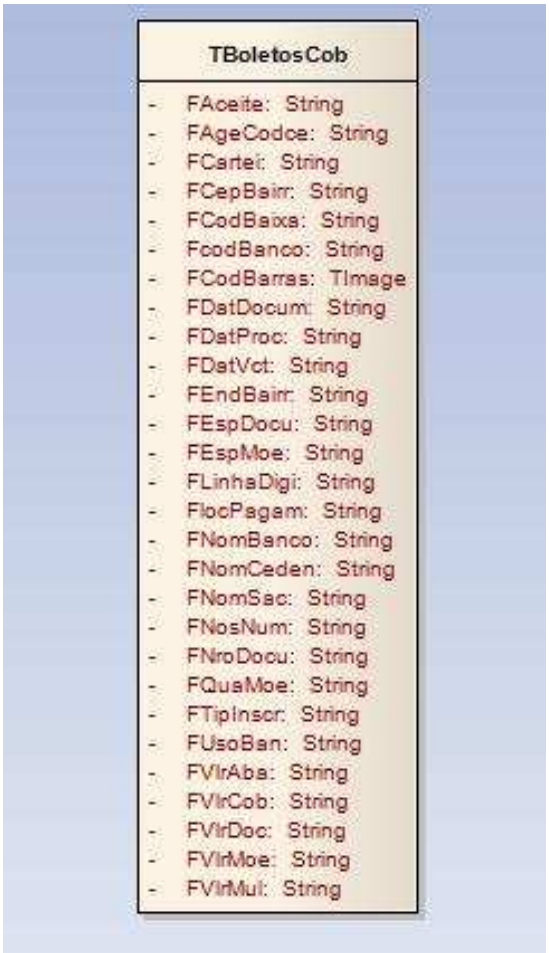

Figura 6 – Diagrama de classes boleto bancário.

O componente `TValidador` recebe o código de barras ou linha digitável e faz a leitura das informações conforme o banco e tipo de cobrança informados. A figura 7 ilustra o diagrama de classe do componente `TValidador`.

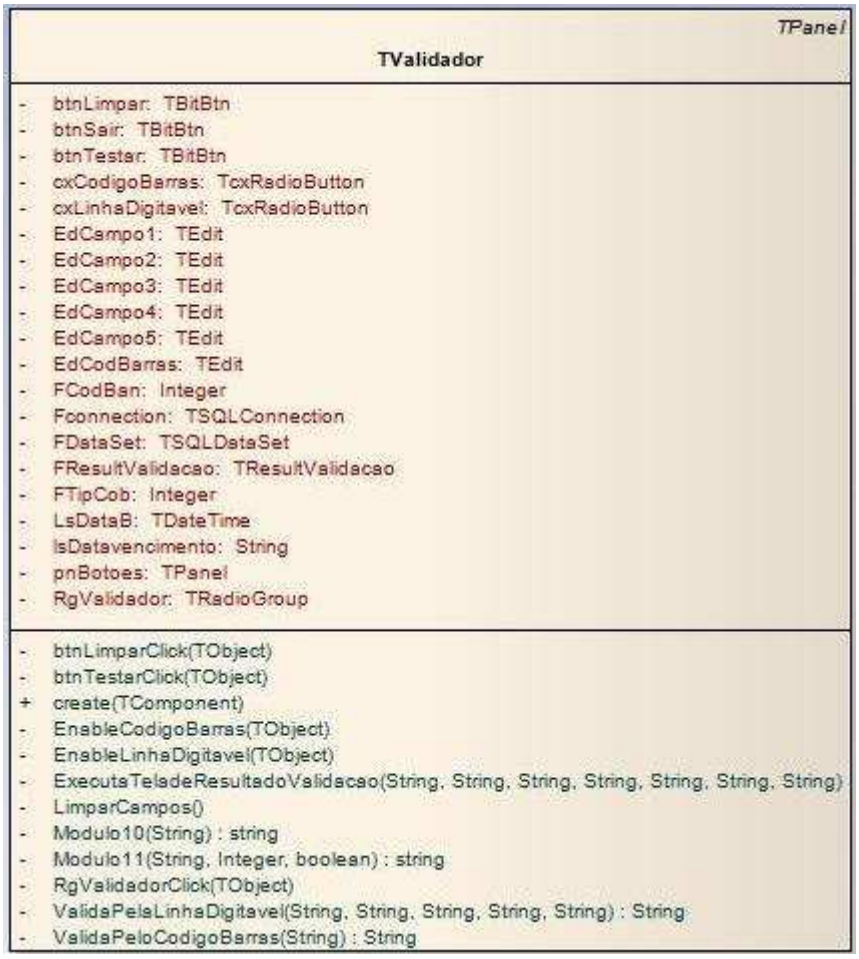

Figura 7 – Diagrama de classes TValidador.

3.3 IMPLEMENTAÇÃO

Esta seção descreve as ferramentas e técnicas utilizadas para o desenvolvimento do trabalho.

3.3.1 Técnicas e ferramentas utilizadas

Para a fase do desenvolvimento e implementação do sistema foram utilizadas ferramentas e técnicas que auxiliaram no decorrer do trabalho, mantendo as informações organizadas e consistentes, agilizando o processo do desenvolvimento.

3.3.1.1 Delphi

A suíte de componentes foi desenvolvida utilizando-se do ambiente de programação Delphi 6 da Borland, que segundo Anselmo (1997), Delphi é um ambiente de programação do Pascal Orientado para Objetos (POO ou Object Pascal) extendido. É um produto de alta performance que combina códigos totalmente compiláveis, ferramentas visuais e tecnologia para a composição de bases de dados escaláveis, e tem como uma forte característica a facilidade para um rápido desenvolvimento em plataforma Windows e aplicações Client/Server.

Para conexão com banco de dados relacional, é usado um componente do Delphi chamado de *SQLConnection*. O *SQLConnection* pode conectar com qualquer banco de dados relacional. Todo processo que envolve conexão com banco de dados na aplicação, é repassado como parâmetro o *SQLConnection* para o banco de dados a ser conectado.

3.3.2 Desenvolvimento do trabalho

Para demonstrar o desenvolvimento do trabalho, serão apresentadas a seguir algumas telas da aplicação dos componentes juntamente com algumas funções do código fonte.

No Quadro 20, mostra como funciona a lógica do cálculo da linha digitável.

```
function TCobCodBar.GetLinhaDigitavel : string;
var
   p1, p2, p3, p4, p5, p6,
   Campo1, Campo2, Campo3, Campo4, Campo5 : string;
begin
   p1 := Copy(Codigo,1,4);
   p2 := Copy(Codigo,20,5);
   p3 := Modulo10(p1+p2);
   p4 := p1+p2+p3;
   p5 := Copy(p4,1,5);
   p6 := Copy(p4,6,5);
   Campo1 := p5+'.'+p6;

   p1 := Copy(Codigo,25,10);
   p2 := Modulo10(p1);
   p3 := p1+p2;
   p4 := Copy(p3,1,5);
   p5 := Copy(p3,6,6);
   Campo2 := p4+'.'+p5;

   p1 := Copy(Codigo,35,10);
   p2 := Modulo10(p1);
   p3 := p1+p2;
   p4 := Copy(p3,1,5);
   p5 := Copy(p3,6,6);
   Campo3 := p4+'.'+p5;

   Campo4 := Copy(Codigo,5,1);

   Campo5 := Copy(Codigo,6,14);

   Result := Campo1 + ' ' + Campo2 + ' ' + Campo3 + ' ' + Campo4 + ' ' +
Campo5;
End;
```

Quadro 20 – Código fonte Lógica de geração linha digitável

Para montagem da linha digitável, é recebido como parâmetro a seqüência de caracteres lido no código de barras e destrinchado conforme regras da FEBRABAN.

A montagem do código de barras é feita com uma regra de definição do código de barras cadastrada pelo usuário nas telas de cadastro do próprio componente. Quando é disparada uma geração de boleto de cobrança, é informado como parâmetros necessários juntamente com a classe `TTitulo` o código do banco e o tipo de cobrança. O componente carrega o cadastro conforme parâmetros, e busca as informações necessárias da regra na classe `TTitulo` onde estão todas as informações necessárias para a geração do boleto de cobrança.

No Quadro 21 é ilustrada a função onde faz o cálculo do módulo 10, que é necessário para o cálculo dos dígitos verificadores da montagem da linha digitável.

```
function Modulo10(Valor: String) : string;
var
  Auxiliar : string;
  Contador, Peso : integer;
  Digito : integer;
begin
  Auxiliar := '';
  Peso := 2;
  for Contador := Length(Valor) downto 1 do
  begin
    Auxiliar := IntToStr(StrToInt(Valor[Contador]) * Peso) +
Auxiliar;
    if Peso = 1 then
       Peso := 2
    else
       Peso := 1;
  end;
  Digito := 0;
  for Contador := 1 to Length(Auxiliar) do
  begin
    Digito := Digito + StrToInt(Auxiliar[Contador]);
  end;
  Digito := 10 - (Digito mod 10);
  if (Digito > 9) then
     Digito := 0;
  Result := IntToStr(Digito);
end;
```

Quadro 21 – Código fonte Módulo 10

A Figura 8 esta ilustrando a tela do cadastro do banco. Essa tela faz parte da aplicação do componente `TGeracaBoleto`. Para cadastrar um novo banco é necessário informar o código do banco a ser cadastrado no campo *Código Banco*. Para alterar ou excluir algum banco, deve ser selecionado o banco no campo *Código Banco* ou pelo *Lookup* ao lado.

Figura 8 – Cadastro do banco

Na Figura 9 é a ilustração do cadastro do tipo de cobrança, que fica em outra aba dentro do cadastro do banco. Para cadastrar um novo tipo de cobrança é necessário informar o tipo de cobrança a ser cadastrado no campo *Tipo de Cobrança*. Para alterar ou excluir algum tipo de cobrança, deve ser selecionado o tipo de cobrança no campo *Tipo de Cobrança* ou pelo *Lookup* ao lado.

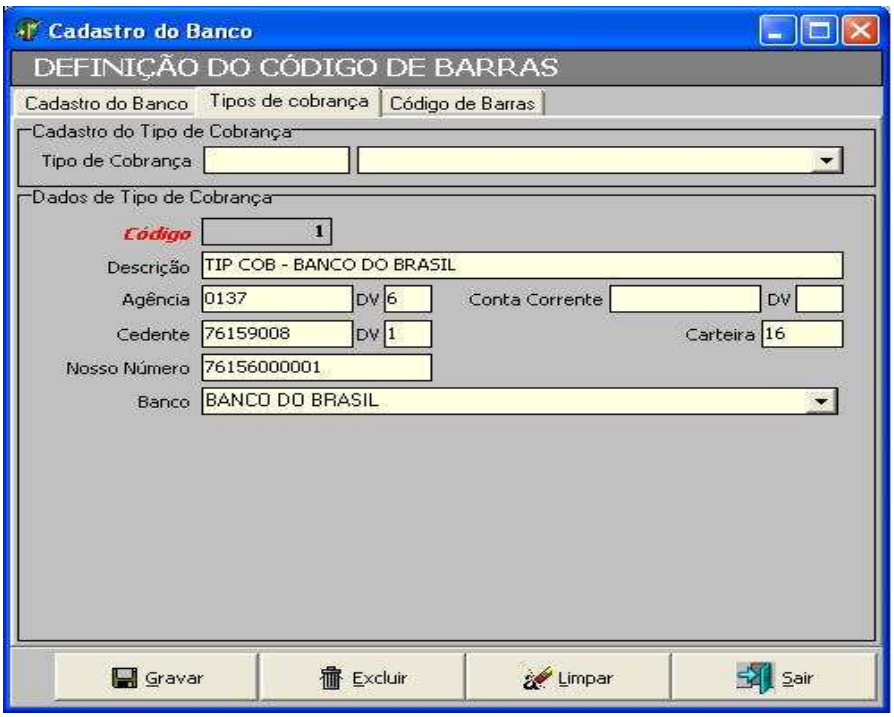

Figura 9 – Cadastro do tipo de cobrança

Na Figura 10 é feita à ilustração do cadastro de definição do código de barras. Para cadastrar uma nova regra de definição do código de barras, deve ser selecionado um banco e um tipo de cobrança ambos cadastrados nas abas anteriores. Cada banco poderá ter vários tipos de cobranças, mas somente será aceito o cadastro de uma regra de definição do código de barras para cada banco e tipo de cobrança.

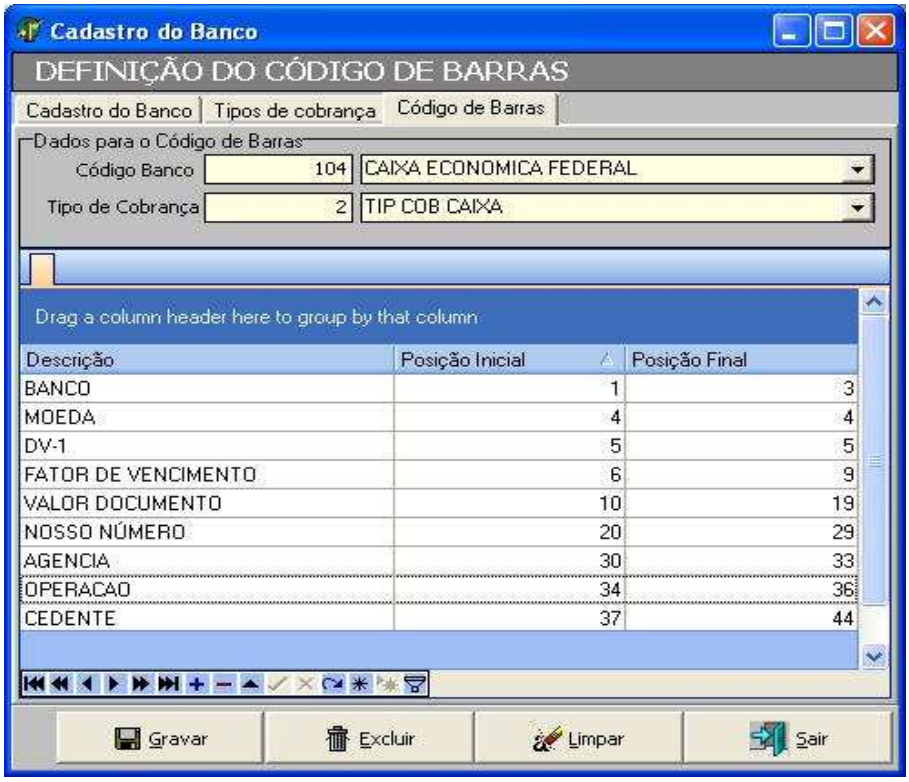

Figura 10 – Cadastro da regra de definição do código de barras

3.3.3 Operacionalidade da implementação

Para demonstrar o funcionamento da aplicação são apresentadas a seguir algumas funcionalidades passo a passo da utilização dos componentes simulando um caso real de utilização.

Para facilitar o entendimento, as principais telas da aplicação serão apresentadas com a explicação de suas funcionalidades. Na Figura 11, está ilustrando um novo formulário com o componente `TGeracaoBoleto` e o componente `SQLConnection`.

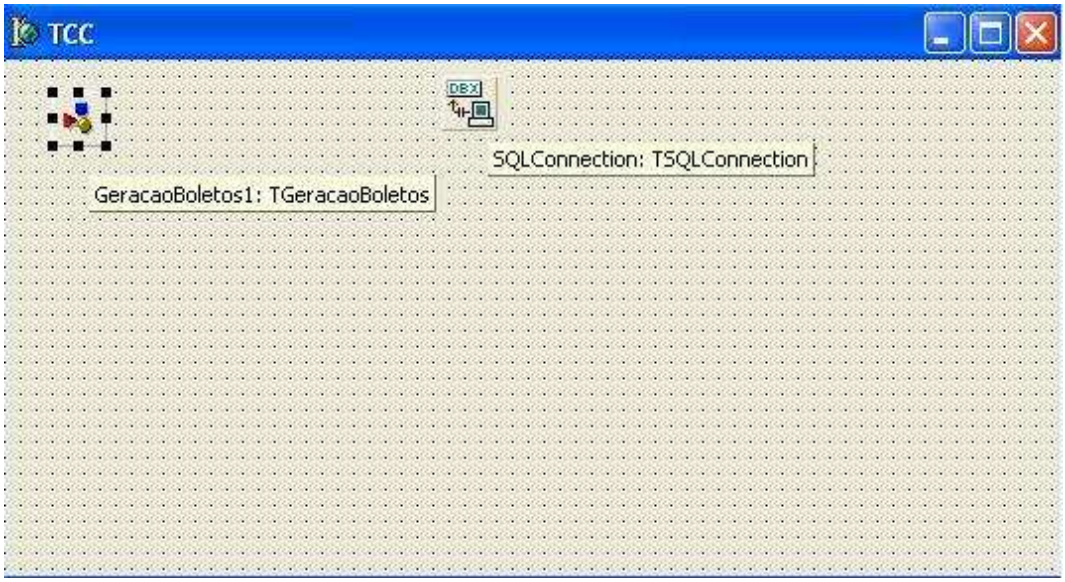

Figura 11 – Componente geração de boletos

O componente `GeracaoBoletos1` informado em tela necessita de um `SqlConnection`, podendo assim, fazer a conexão com qualquer banco de dados. A Figura 12 mostra como conecta o componente com o `SqlConnection`. O componente possui uma propriedade para ser informado o *`SqlConnection`*.

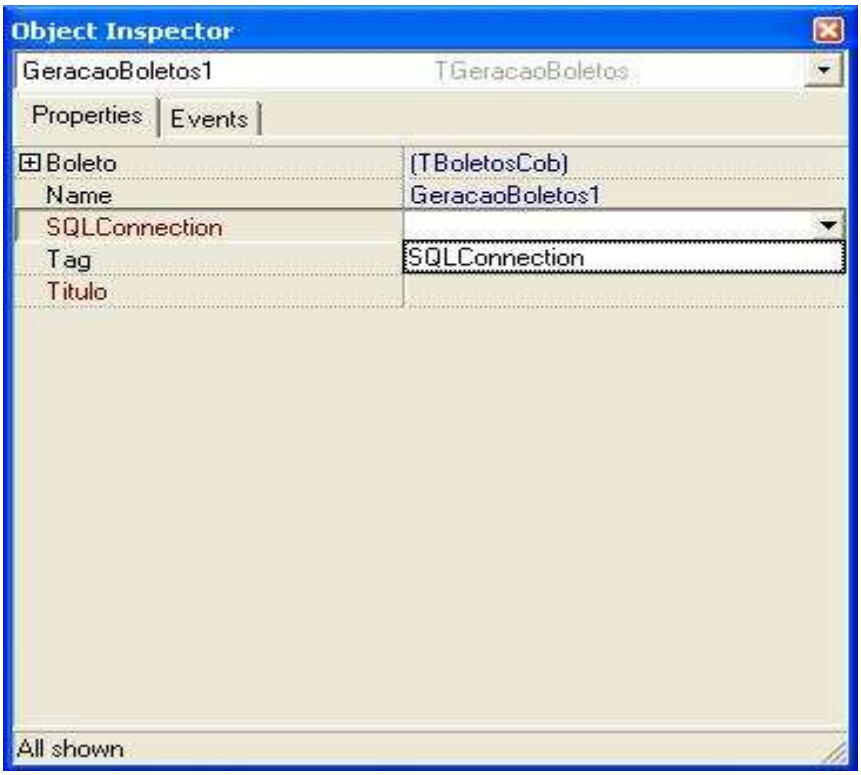

Figura 12 – Informar o SqlConnection no componente GeracaoBoletos1

Após fazer a conexão com o banco de dados, o componente faz chamada dos cadastros, conforme Quadro 22. O comando faz a chamada do cadastro do banco, onde nesse

mesmo formulário possui uma aba para cadastramento do tipo de cobrança e outra aba para cadastramento das regras de definição do código de barras.

```
procedure TFrm_GerarBoletos.CadastroBancoClick(Sender: TObject);
begin
  GeracaoBoletos1.ExecutaCadastroBanco;
end;
```

Quadro 22 – Fazer a chamada do cadastro do banco

A Figura 13 apresenta um protótipo de como seria usado o componente `TGeracaoBoletos`. Aqui são informados todos os valores que o componente necessita para fazer a geração do boleto de cobrança.

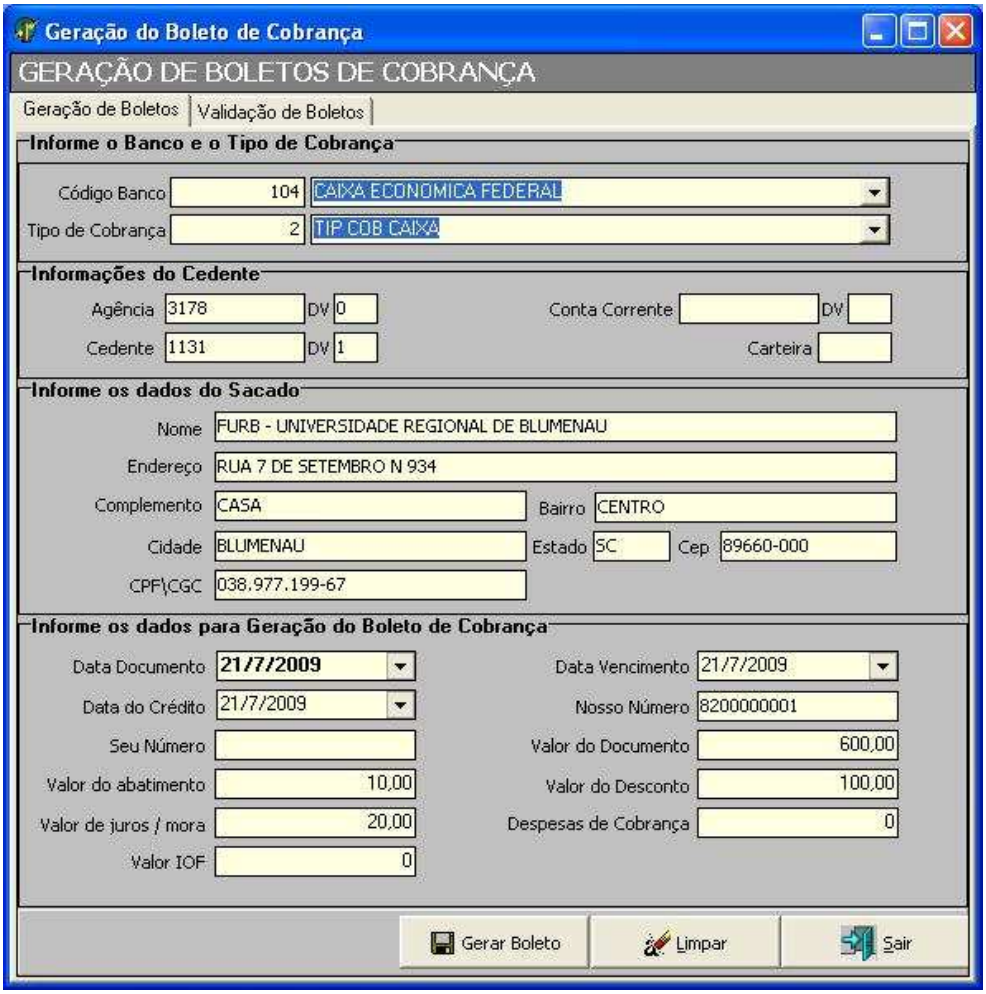

Figura 13 – Protótipo de um formulário usando o componente TGeracaoBoletos

No Quadro 23, este código fonte de como são populadas as informações que o componente necessita para poder fazer a geração do boleto de cobrança.

```
Procedure TFrm_GerarBoletos.CarregaDadosGeracaoBoleto;
Begin
  GeracaoBoletos1.Titulo.SqlConnection               := SQLConnection;
  GeracaoBoletos1.Titulo.Instrucoes.Clear;
  GeracaoBoletos1.Titulo.Carteira                    := EdCarteira.Text;
  GeracaoBoletos1.Titulo.Cedente.CodigoCedente       := EdCedente.Text;
  GeracaoBoletos1.Titulo.Cedente.DigitoCodigoCedente := EdDVCedente.Text;
  GeracaoBoletos1.Titulo.Cedente.Nome                := EdNome.Text;
  GeracaoBoletos1.Titulo.Cedente.ContaBancaria.Banco.SqlConnection := SQLConnection;
  GeracaoBoletos1.Titulo.Cedente.ContaBancaria.fBanco.Codigo  := CxBanco.EditValue;
 GeracaoBoletos1.Titulo.Cedente.ContaBancaria.fBanco.TipCob:=cxTipoCobranca.EditValue;
  GeracaoBoletos1.Titulo.Cedente.ContaBancaria.fCodigoAgencia := EdNroAgencia.Text;
  GeracaoBoletos1.Titulo.Cedente.ContaBancaria.fDigitoAgencia := EdDVAgencia.Text;
  GeracaoBoletos1.Titulo.Cedente.ContaBancaria.fNumeroConta:= EdContaCorrente.Text;
  GeracaoBoletos1.Titulo.Cedente.ContaBancaria.fDigitoConta:= EdDvContaCorrente.Text;
  GeracaoBoletos1.Titulo.Cedente.ContaBancaria.Operacao    := '870';
  GeracaoBoletos1.Titulo.Cedente.Endereco.fRua         := EdEndere.Text;
  GeracaoBoletos1.Titulo.Cedente.Endereco.fComplemento := EdComple.Text;
  GeracaoBoletos1.Titulo.Cedente.Endereco.fBairro      := EdBairro.Text;
  GeracaoBoletos1.Titulo.Cedente.Endereco.fCidade      := EdCidade.Text;
  GeracaoBoletos1.Titulo.Cedente.Endereco.fEstado      := EdEstado.Text;
  GeracaoBoletos1.Titulo.Cedente.Endereco.fCEP         := EdCep.Text;
  GeracaoBoletos1.Titulo.Cedente.NumeroCPFCGC          := EdCpfCgc.Text;
  GeracaoBoletos1.Titulo.Sacado.Nome                   := EdNome.Text;
  GeracaoBoletos1.Titulo.Sacado.NumeroCPFCGC           := EdCpfCgc.Text;
  GeracaoBoletos1.Titulo.Sacado.Endereco.Rua           := EdEndere.Text;
  GeracaoBoletos1.Titulo.Sacado.Endereco.Complemento   := EdComple.Text;
  GeracaoBoletos1.Titulo.Sacado.Endereco.fBairro       := EdBairro.Text;
  GeracaoBoletos1.Titulo.Sacado.Endereco.fCidade       := EdCidade.Text;
  GeracaoBoletos1.Titulo.Sacado.Endereco.fEstado       := EdEstado.Text;
  GeracaoBoletos1.Titulo.Sacado.Endereco.CEP           := EdCep.Text;
  GeracaoBoletos1.Titulo.DataCredito                   := dtCredito.Date;
  GeracaoBoletos1.Titulo.DataDocumento                 := DtDocumento.Date;
  GeracaoBoletos1.Titulo.DataVencimento                := DtVencimento.Date;
  GeracaoBoletos1.Titulo.NossoNumero                   := EdNosNum.Text;
  GeracaoBoletos1.Titulo.ValorDocumento                := VlDocumento.FloatNumber;
  GeracaoBoletos1.Titulo.ValorAbatimento               := VlAbatimento.FloatNumber;
  GeracaoBoletos1.Titulo.ValorDesconto                 := VlDesconto.FloatNumber;
  GeracaoBoletos1.Titulo.ValorMoraJuros                := VlJurMora.FloatNumber;
  GeracaoBoletos1.Titulo.Instrucoes.Add('Após o vencimento, Multa de 2 % mais Juros de
1% ao dia');
End;
```

Quadro 23 – Código fonte populando informações no componente

Esse procedimento é feito antes de chamar a função que gera o boleto de cobrança. No Quadro 24 exemplifica o funcionamento da rotina para fazer a geração do boleto bancário.

```
01 procedure TFrm_GerarBoletos.btnGravarClick(Sender: TObject);
02 Var
03   i : Integer;
04 begin
05  Try
06    GeracaoBoletos1.CriaTitulo;
07    CarregaDadosGeracaoBoleto;
08    FBoleto :=  GeracaoBoletos1.Titulo.Visualizar;
09    PopulaBoleto(FBoleto);
10    FRelBoleto.Preview;
11  Finally
12   GeracaoBoletos1.DestroyTitulo;
13  end;
14 end;
```

Quadro 24 – Código fonte comandos para geração do boleto bancário

O comando informado na linha 6 serve para criar a classe `TTitulo` dentro do

componente `TGeracaoBoleto`. O procedimento na linha 7 é um exemplo de popular as informações necessárias para fazer a geração do boleto de cobrança. A linha 8 faz todos os cálculos necessários, gera o boleto de cobrança e retorna um objeto do tipo `TBoletoCob`.
Esse objeto do tipo `TBoletoCob` é uma classe onde possui todos as informações para montar um boleto de cobrança.

A linha 9, é um exemplo de procedimento que tem como parâmetro um objeto do tipo `TBoletoCob`, onde populo um layout de boleto com as informações recebidas do componente. As informações retornadas do componente podem ser usadas da forma que for necessário.

Na Figura 14 é ilustrado um exemplo de layout de boleto bancário. Esse layout faz parte de uma aplicação onde usa o componente `TGeracaoBoleto`. Ele foi populado com as informações recebidas da classe `TBoletoCob` do componente `TGeracaoBoleto`.

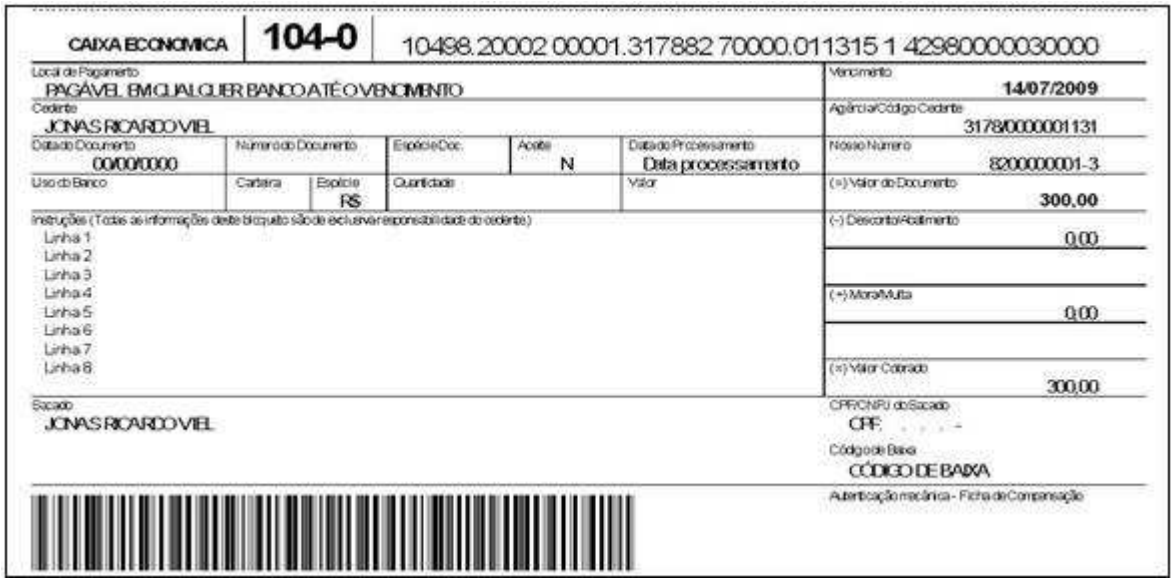

Figura 14 – Boleto bancário gerado

A figura 15, ilustra o componente `TValidador`. Ele necessita de dois parâmetros para fazer a validação que seria o código do banco e o tipo de cobrança, ambos os códigos devem estar cadastrados no sistema. O `TValidador` possui um evento chamado `OnResultValidacao`, onde retorna os dados referentes ao código a ser validado.

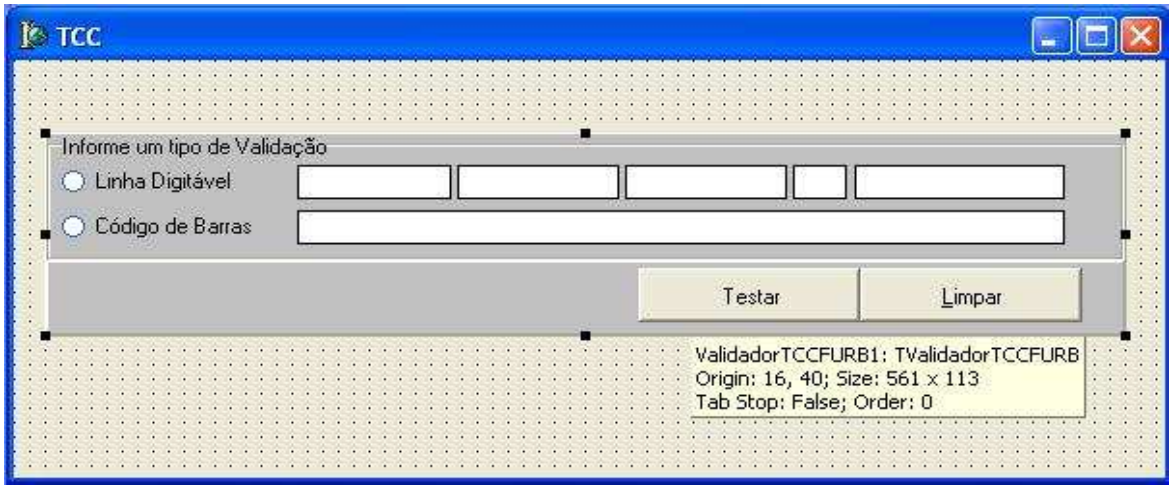

Figura 15 – Componente TValidador

O retorno das informações do teste pode ser através do evento `OnResultValidacao` ou por um formulário que também faz parte do componente. Se no evento `OnResultValidacao` não tiver nenhum procedimento que executa o evento, ele dispara o formulário com o resultado do teste. O evento serve para poder adaptar o resultado do teste em qualquer aplicação.

3.4 RESULTADOS E DISCUSSÃO

Para os testes dos componentes, foi montado um protótipo onde o mesmo popula as informações necessárias para o componente fazer a geração do boleto de cobrança. Foi criado um layout de boleto de cobrança, onde foi possível testar a operacionabilidade dos casos de uso apresentados na seção da especificação. Foram constatados que os componentes atenderam aos requisitos especificados.

O processo que faz a geração do boleto de cobrança mostrou-se bastante rápido e seguro. Foram cadastrados tipos de cobrança de bancos diferentes e feito a validação pelo segundo componente, e também pelos caixas eletrônicos na parte de pagamento de boletos com código de barras, ambos os testes apresentaram os mesmos resultados.

O desenvolvimento resultou em dois componentes de fácil conexão com banco de dados. A facilidade dá-se pelo fato de poder gerar boletos de cobrança para qualquer banco, podendo configurar o tipo de cobrança desejado usando qualquer tipo de banco de dados relacional.

O diferencial para este trabalho em relação aos trabalhos correlatos foi à geração e validação de boletos de cobrança em dois componentes, possibilitando à empresa o acesso completo desde a parte de geração até a validação dos boletos de cobrança. Foi disponibilizado nesse componente um ambiente para criação de novas regras de definição do código de barras, que serão disponibilizados para geração e validação sem custo algum para empresa.

No Quadro 25 são apresentadas as principais características entre o sistema de geração de boletos COBREBEN e gerador de boletos desenvolvido.

Comparação entre COBREBEN e o Gerador de Boletos		
	COBREBEN	**Gerador Boletos**
Faz geração de boletos de cobrança para qualquer banco	X	X
Pode ser cadastrado qualquer tipo de cobrança para geração do boleto de cobrança		X
Tem custo para adquirir uma licença de uso para gerar o boleto de cobrança	X	
Necessita de desenvolvimento para um novo tipo de cobrança	X	
Pode ser usado qualquer layout de boletos de cobrança		X

Quadro 25 – Comparação entre COBREBEN e Gerador de Boletos

É importante ressaltar que o sistema COBREBEN funciona com licenças de uso. Para cada cnpj, deve ser pago por uma licença de uso. O sistema COBREBEN possui modelos próprios de boletos de cobrança, não podendo adaptar as informações do boleto de cobrança gerado em qualquer layout de boleto de cobrança.

4 CONCLUSÕES

Após a análise do problema envolvendo a geração e validação de boletos de cobrança para qualquer banco, surgiu a idéia deste trabalho, sendo este o desenvolvimento de dois componentes, um para fazer a geração do boleto de cobrança e outro para a validação.

As telas de cadastro fazem parte do componente de geração do boleto de cobrança, onde possui também o cadastro da regra de definição do código de barras, podendo ser gerado boletos de cobrança conforme a definição que o banco necessita.

O desenvolvimento resultou em dois componentes de fácil configuração e acesso para fazer a geração e validação do boleto de cobrança. A flexibilidade dá-se pelo fato da mesma conectar com qualquer banco de dados relacional passando como parâmetro o `SqlConnection`, e fazer as chamadas das telas com uma única linha de comando.

Esse trabalho contribuiu para uma melhor praticidade na geração de boletos de cobrança para a empresa, tendo em vista que funciona para qualquer banco.

Finalizando, o sistema foi feito usando ferramenta Delphi, tendo como vantagem a praticidade de usar um componente com uma estrutura organizada, com classes e métodos bem definidos. A limitação desses componentes é ao fato que deve ser usado somente para aplicações desenvolvidas em Delphi.

4.1 EXTENSÕES

Sugere como extensão deste trabalho a implementação da geração de arquivos remessa, para suprir a necessidade de quando for necessário enviar ao banco todos os boletos de cobrança para registro ou para fazer a impressão.

Também podem ser migrados os componentes para outras plataformas, por exemplo, Java e .Net.

Outra sugestão seria permitir passar as regras do boleto do componente para a geração, em vez de usar banco de dados relacional.

REFERÊNCIAS BIBLIOGRÁFICAS

ANSELMO, Fernando Antônio. **Desvendando o caminho das pedras**. São Paulo: Borland, 1997.

FEBRABAN. **Layout padrão Febraban**. São Paulo**:** Federação Brasileira de Bancos, 2008.

COBRE BEM. Cobre Bem Tecnologia, 2007. Disponível em:
< http://www.cobrebem.com.br >. Acesso em 20 de mai. 2008

GROSSMANN, Fábio; ZYNGIER, Mauro L. **Código de Barras:** da teoria à prática**.** São Paulo: Nobel, 1991.

GS1 BRASIL - Associação Brasileira de Automação, 2008. Disponível em:
< www.gs1brasil.org.br>. Acesso em 11 abr. 2008.

MARTINS, José Carlos Cordeiro. **Gerenciando projetos de desenvolvimento de softwares com PMI, RUP e UML**. Rio de Janeiro: Brasport, 2006.

MORETTI, A. A. **Código de Barras**. 1999. 87 f. Trabalho de Conclusão de Curso (Bacharelado em Ciências da Computação) – Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau.

SILVA, Vera Lucia Pinheiro da. **Aplicações práticas do código de barras.** São Paulo: Nobel, 1993.