UNIVERSIDADE REGIONAL DE BLUMENAU

CENTRO DE CIÊNCIAS EXATAS E NATURAIS

CURSO DE SISTEMAS DE INFORMAÇÃO – BACHARELADO

FERRAMENTA PARA MANUTENÇÃO, DOCUMENTAÇÃO E PADRONIZAÇÃO DE INTERFACES PARA O AMBIENTE DELPHI

RODRIGO ZIMMERMANN

BLUMENAU
2006

2006/2-09

RODRIGO ZIMMERMANN

FERRAMENTA PARA MANUTENÇÃO, DOCUMENTAÇÃO E PADRONIZAÇÃO DE INTERFACES PARA O AMBIENTE 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. Everaldo Artur Grahl, Mestre

BLUMENAU
2006

2006/2-09

FERRAMENTA PARA MANUTENÇÃO, DOCUMENTAÇÃO E PADRONIZAÇÃO DE INTERFACES PARA O AMBIENTE DELPHI

Por

RODRIGO ZIMMERMANN

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

Presidente:	Prof. Everaldo Artur Grahl, Mestre - FURB
Membro:	Prof. Francisco Adell Péricas , Mestre - FURB
Membro:	Prof. Wilson Pedro Carli, Mestre - FURB

Blumenau, 14 de Dezembro 2006

Dedico este trabalho a minha esposa Silvana que tem muita paciência em me repartir com um computador e vários clientes e aos meus pais Maria e Gilberto que me incentivaram e apoiaram para seguir este carreira.

AGRADECIMENTOS

À Deus, por ter me dado sabedoria.

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

Tudo que acontece em nossas vidas pode ter dois sentidos, pode ser sorte ou azar, depende que lado você olha.

Rodrigo Zimmermann

RESUMO

O processo de desenvolvimento de sistemas tem se especializado dia-a-dia. Neste contexto, o uso de ferramentas para automatização, documentação e padronização de interfaces torna-se cada vez maior. O presente trabalho apresenta uma ferramenta que cria e gerencia um dicionário de dados no *MySQl* para que componentes escritos em Delphi, interpretem este dicionário e gerem em tempo de execução interfaces para cadastro e pesquisa de registros padronizadas. A ferramenta permite a criação de tabelas, campos e relacionamentos no banco *MySQl* , gerenciando algumas regras de negócio e documentação do sistema.

Palavras-chave: Padronização, Interface, Documentação, Componentes Delphi .

ABSTRACT

The systems process development if has specialized day-by-day. In this context, the use of tools for automatization, documentation and standardization of interfaces becomes each bigger time. The present work presents a tool that creates and manages a data dictionary in the *MySQl* so that component written in Delphi, they interpret this dictionary and they generate in execution time interfaces for register in cadastre and research of standardized registers. The tool allows the creation of tables, fields and relationships in the *MySQl* database, managing some rules of business and documentation of the system.

Key-Words: Standardization, Interface, Documentation, Delphi Components.

LISTA DE QUADROS

LISTA DE FIGURAS

LISTA DE SIGLAS

ADO - *ActiveX Data Objects*

DDL - *Data Definition Language*

DER – Diagrama entidade relacionamento

MDAC – *Microsoft Data Access Components*

RAD - *Rapid Application Development*

SQL – *Structured Query Language*

UML - *Unified Modeling Language*

XML - *eXtensible Markup Language*

SUMÁRIO

1 INTRODUÇÃO

Quando uma *software house* desenvolve e presta serviços de manutenção em um aplicativo, muitas funcionalidades se repetem ao longo da aplicação. Isto acaba se tornando um problema quando o sistema fica mais complexo, gerando uma grande possibilidade do sistema ficar desatualizado, caso não possua uma boa documentação.

Cousin e Collofello (1992) acreditam que as equipes de manutenção poderiam resolver melhor todos os problemas de manutenção de software se dispusessem de documentação atualizada, treinamento contínuo e ferramentas automatizadas. Uma combinação destes três itens poderia ser o melhor caminho para a continuidade da qualidade do software.

Focando mais diretamente a documentação do software, muitos esforços tem sido feitos na tentativa de melhorar o processo de documentação de software. No estudo de Garland (1991) é discutido um método para se incorporar informações sobre a solução de problemas numa base de dados que contém todo o acompanhamento da manutenção do software.

Segundo Forward (2002), documentação de software pode ser definida como um artefato cuja finalidade seja comunicar a informação sobre o sistema de software ao qual ele pertence. Atualmente a documentação de alterações e correções que um software sofre durante seu ciclo de vida são armazenadas em lugares distintos, impedindo uma uniformidade das informações e funções do sistema, tanto para a equipe que dará o suporte a esta aplicação quanto para o usuário.

Outro aspecto a ser considerado no desenvolvimento de softwares é a criação de interfaces, que traduzem a informação para as pessoas através de métodos que devem ser facilmente percebidos por um de nossos sentidos. Só assim essa informação nos chega de forma clara e objetiva. Segundo Minasi (1994), o crescimento da popularidade de ambientes gráficos como *X windows* da *UNIX*, *OS/2*, o *Macintosh*, e é claro o mais popular de todos, o Microsoft *Windows*, forçou muitos programadores a mover-se para a programação visual utilizando ambientes de desenvolvimento *Rapid Application Development* (RAD), como Delphi ou o Visual Basic, tornando assim o trabalho para desenvolvimento de interfaces mais produtivo, mais ainda não automatizado.

Segundo o modelo de qualidade de software MPS.BR (2006), no processo que se refere a solução técnica, as interfaces devem ser planejadas com base em critérios pré definidos, e a documentação também deve ser mantida e distribuída de acordo com critérios

pré definidos.

Dentro deste contexto, considerando a necessidade específica de uma empresa de software em agilizar a construção de interfaces, centralizar a documentação e automatizar o processo de manutenção, desenvolveu-se este TCC aplicado para a empresa Rz Sistemas que tem como principal foco o desenvolvimento e manutenção de sistemas de gestão empresarial. A Empresa Rz Sistemas utiliza para a implementação de seus aplicativos componentes escritos em Delphi que foram utilizados neste trabalho para a interpretação das informações contidas no dicionário de dados.

1.1 OBJETIVOS DO TRABALHO

O objetivo deste trabalho foi o desenvolvimento de uma ferramenta que gere um dicionário de dados no *MySQl*, para que componentes desenvolvidos em Delphi consigam gerar interfaces já contendo a documentação.

Os objetivos específicos são:

a) documentar informações relativas a base de dados que serão utilizadas pelo usuário final nesta ferramenta;
b) gerar informações suficientes para que componentes escritos em Delphi consigam gerar as janelas de cadastros e pesquisas automaticamente sem intervenção de programação, apenas com regras definidas na ferramenta;
c) criar um banco de dados no *MySQl* com as definições inseridas na ferramenta.

1.2 ESTRUTURA DO TRABALHO

Este trabalho está dividido em forma de capítulos descritos a seguir.

O primeiro capítulo expõe na introdução uma justificativa que originou este trabalho como também uma síntese do que será tratado no desenvolvimento do trabalho e os objetivos a serem alcançados.

O segundo capítulo explica mais detalhadamente qual a fundamentação teórica e prática para o desenvolvimento deste trabalho.

O terceiro capítulo explica qual a metodologia utilizada para o desenvolvimento do trabalho.

O quarto capítulo descreve sinteticamente como criar um aplicativo para cadastro de clientes, representantes e cidades usando a ferramenta desenvolvida neste trabalho, além disto é demonstrado um caso típico de manutenção no software e como a ferramenta trata esta situação

O quinto capítulo expõe as considerações finais após o desenvolvimento do trabalho e algumas sugestões para sua continuação.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo é apresentada a situação atual da empresa em relação ao seu processo de desenvolvimento e após a situação pretendida.

2.1 SITUAÇÃO ATUAL

Em um sistema de gestão de grande porte, grande parte do tempo é dispendido na criação de interfaces , campos, tabelas, busca de dados e algumas regras que se repetem ao longo de todo o sistema, isto consome um tempo considerável no desenvolvimento, manutenção e muitas linhas de código desnecessárias, causando assim vários "*bugs*". Numa primeira análise feita na empresa Rz Sistemas levantou-se que 70% dos erros que acontecem na atualização dos sistemas são derivados da falta de campos nas bases de dados dos clientes. Isto se deve ao fato que às vezes o programador cria campos diretamente no banco de dados e eventualmente esquece de atualizar na documentação os novos campos criados e qual sua função na aplicação.

Segundo Liesenberg (2006) em sistemas interativos, para os quais o objetivo é produzir interfaces que tenham o usuário como ponto central, a alocação de recursos financeiros e de tempo costuma ser da ordem de 50% do total alocado durante a fase de desenvolvimento. O código resultante também pode ser, muitas vezes, atribuído em torno de 50% para a interface e 50% para o componente que implementa a aplicação propriamente dita do sistema. Uma redução de custos e tempo de projeto, contudo, é possível com ferramentas que automatizem o trabalho de desenvolvimento.

Liesenberg (2006) afirma que uma interface não se restringe apenas a uma mera questão de acabamento e sofisticação de um sistema interativo realizado, depois de pronta a parte que implementa a funcionalidade de tal sistema. A construção de uma interface deve ser realizada com técnicas adequadas. Além do mais, tempo e recursos devem ser previstos e alocados de acordo. As técnicas e processos tradicionais da área de Engenharia de Software, infelizmente, não se mostram eficazes para o desenvolvimento de interfaces. Em função disto, modelos e técnicas específicas de produção foram e continuam sendo propostos e o processo de desenvolvimento da interface costuma ser realizado à parte do processo de

desenvolvimento do componente responsável pela funcionalidade do sistema à qual a interface dá acesso.

Wasserman (1994) apresentou algumas tendências para ambientes de desenvolvimento de software, e também aponta como tendência natural a codificação rápida. Para isto serão necessárias várias ferramentas para acelerar a programação, a fim de que haja maior concentração de esforços e recursos nas fases anteriores análise e projeto.

Além disto Wasserman (1994) citou a importância de um dicionário de dados, o qual permitirá o armazenamento das informações a serem compartilhadas. O dicionário de dados deverá estabelecer padrões a serem seguidos pelas diversas ferramentas e deverá ser aberto o suficiente para que todas as ferramentas possam interagir com ele.

A manutenção do software existente pode ser responsável por mais de 70% de todo o esforço despendido por uma organização de software. A porcentagem continua a se elevar à medida que mais software é produzido. No horizonte, pode-se prever uma organização de software baseada na manutenção que não mais pode produzir novo software, porque está gastando todos os seus recursos disponíveis mantendo um software antigo (PRESSMAN, 1995).

O modelo de melhoria de processo de software brasileiro (MPS.BR) abrange questões relativas a melhoria da qualidade de software principalmente focada a micro, pequenas e médias empresas. Analisando mais detalhadamente o modelo MPS.BR, no item que se refere ao processo de solução técnica, foram encontradas questões relativas à padronização de interfaces e documentação de sistemas, justificando a construção de uma ferramenta que apóie a criação de interfaces padronizadas juntamente com sua documentação.

Outro fato relevante é que caso este sistema não seja de uso exclusivo de uma organização haverá necessidades de alterações no código fonte da aplicação para dar suporte a algumas regras às vezes específicas de um cliente.

Imagine que um determinado campo em um cadastro tenha um valor mínimo. Suponha que este valor mínimo tenha que ser alterado porque um dos usuários deste sistema tem uma necessidade específica. As alternativas poderiam ser:

a) criar uma regra de validação dentro do banco de dados: se o sistema suportar múltiplo bancos de dados os técnicos terão que conhecer o funcionamento de cada gerenciador de banco de dados para definir esta regra. Caso for implementada no banco, a mensagem de erro do valor mínimo será apresentada em inglês, o que dificulta a compreensão do usuário final. Cada banco de dados vai exibir esta informação de maneira diferente;

b) criar uma regra dentro do código fonte da aplicação. E esta alternativa acrescentaria mais linhas de código no sistema que poderá apresentar problemas futuros. Se outro cliente precisar de um valor diferente terá que ser criada uma outra regra , o que levará novamente a possível inserção de um novo "*bug*".

Uma visível solução para este problema poderia ser implantar esta regra em um dicionário de dados que pertencerá a base de dados do cliente final. Assim, qualquer técnico do suporte que fornecer manutenção conhecerá o meio de como fazer o processo. A mensagem será padronizada em português, pois não será o banco de dados que apontará o erro e sim o componente que gerencia a comunicação entre banco de dados e a aplicação. Além disto, o sistema não precisa ser recompilado para suportar uma simples regra desta.

Mesmo a programação com uma ferramenta que otimiza muito o trabalho como o Delphi, montar uma interface de cadastro complexa exige um tempo considerável. Este processo atualmente acontece em várias etapas:

a) criação de uma ou várias tabelas que se relacionam entre si em uma base de dados, usando geralmente uma ferramenta distinta para cada banco de dados;
b) criação de diversas conexões com as tabelas usando no mínimo dois componentes para acessar uma única tabela;
c) criação de todos os campos com seus rótulos e campos de edição;
d) criação de diversos mecanismos de busca nestas janelas para otimizar a localização de registros;
e) criação de campos onde são digitados códigos e o sistema mostra o registro relacionado;
f) criação de regras utilizando programação para validação dos dados;
g) criação de máscaras e formatos de exibição para os dados.

Após este processo é criada alguma documentação explicando a função dos campos e das regras existentes.

2.2 SISTEMA PROPOSTO

Aproveitando a experiência deste autor que desde 1997 desenvolve aplicações para gestão empresarial utilizando o Delphi, para amenizar os problemas citados pensou-se numa ferramenta que possua informações suficientes para:

a) definir tabelas, documentando a finalidade desta dentro do aplicativo;
b) definir campos, com atributos como seu tamanho, valor máximo, mínimo, se é requerido, indexado, se será usado para localização e definir a função dele no sistema;
c) definir algumas regras que vão futuramente automatizar a criação da aplicação final;
d) definir regras de relacionamento entre as tabelas;
e) definir o tipo de objeto que será atribuído ao campo para criação da interface final;
f) definir os formulários de cadastros da aplicação, onde será possível definir a forma com o qual o registro será incrementado, o menu que ele se encaixara no sistema, as tabelas relacionadas com o formulário;
g) documentar as alterações e correções que foram feitas em cada *release* do aplicativo;
h) criar uma base de dados e um dicionário de dados com as definições inseridas na ferramenta

Usando as informações e regras definidas nesta ferramenta, um conjunto de objetos escritos em Delphi interpretará estas informações, montando assim de maneira automatizada estes formulários, vinculando automaticamente a documentação, seguindo assim a recomendação do modelo (MPS.BR) que fala no item relativo a solução técnica que as interfaces devem ser projetadas de acordo com padrões pré-definidos além da documentação ser distribuída também de acordo com padrões pré-definidos.

3 DESENVOLVIMENTO DO TRABALHO

O desenvolvimento do trabalho descreve todo o processo de construção da ferramenta, iniciando pela especificação onde foi utilizada a ferramenta *Enterprise Architect* para fazer os casos de uso e diagrama de atividades juntamente com a ferramenta *Db Designer* que foi utilizada para fazer o Diagrama de entidade relacionamento físico (DER). Por último são exibidos alguns trechos de código que foram utilizados para a implementação da ferramenta.

3.1 REQUISITOS DO SISTEMA

Após entrevistas feitas com a equipe de suporte e desenvolvimento da empresa Rz Sistemas, levantou-se os requisitos funcionais e não funcionais necessários para o desenvolvimento da ferramenta, o qual está apresentado no quadro 01 e quadro 02 respectivamente.

Requisitos Funcionais
RF01: A ferramenta deverá permitir o cadastro de informações relativas ao projeto.
RF02: A ferramenta deverá permitir o cadastro de tabelas, juntamente com a documentação da mesma dentro do sistema.
RF03: A ferramenta deverá permitir a definição dos campos de uma tabela bem como todas as informações relevantes para a criação e a documentação do mesmo.
RF04: A ferramenta deverá gerar um *script* de atualização do sistema sendo que este arquivo deverá ser compactado de forma a reduzir o tempo de transferência dele da internet para a máquina que será atualizada.
RF05: A ferramenta deverá permitir registrar as alterações que a versão do sistema contém em relação a verão anterior.
RF06: A ferramenta deverá gerar um arquivo de saída no formato XML com as informações armazenadas na ferramenta.
RF07: A ferramenta deverá permitir o cadastro de formulários com as tabelas e campos relacionados com o formulário.

Quadro 01- Requisitos funcionais

O Quadro 02 lista os requisitos não funcionais previsto para o sistema.

Requisitos não funcionais
RNF01: O sistema deverá ser desenvolvido no ambiente de programação Delphi 7, utilizando banco de dados *MsAccess* para armazenar os dados e criar uma base de dados no banco *MySql*.
RNF02: O Sistema deverá interpretar o padrão de configuração dos aplicativos utilizados pela Rz Sistemas para realizar a atualização dos componentes.

Quadro 02 - Requisitos não funcionais

3.2 ESPECIFICAÇÃO

A especificação foi desenvolvida na ferramenta CASE *Enterprise Architect*, foram utilizados alguns diagramas da *Unified Modeling Language* (UML).

3.2.1 CASOS DE USO

Na Figura 01 é apresentado o diagrama de Casos de uso contendo as funcionalidades que o programador terá com a ferramenta.

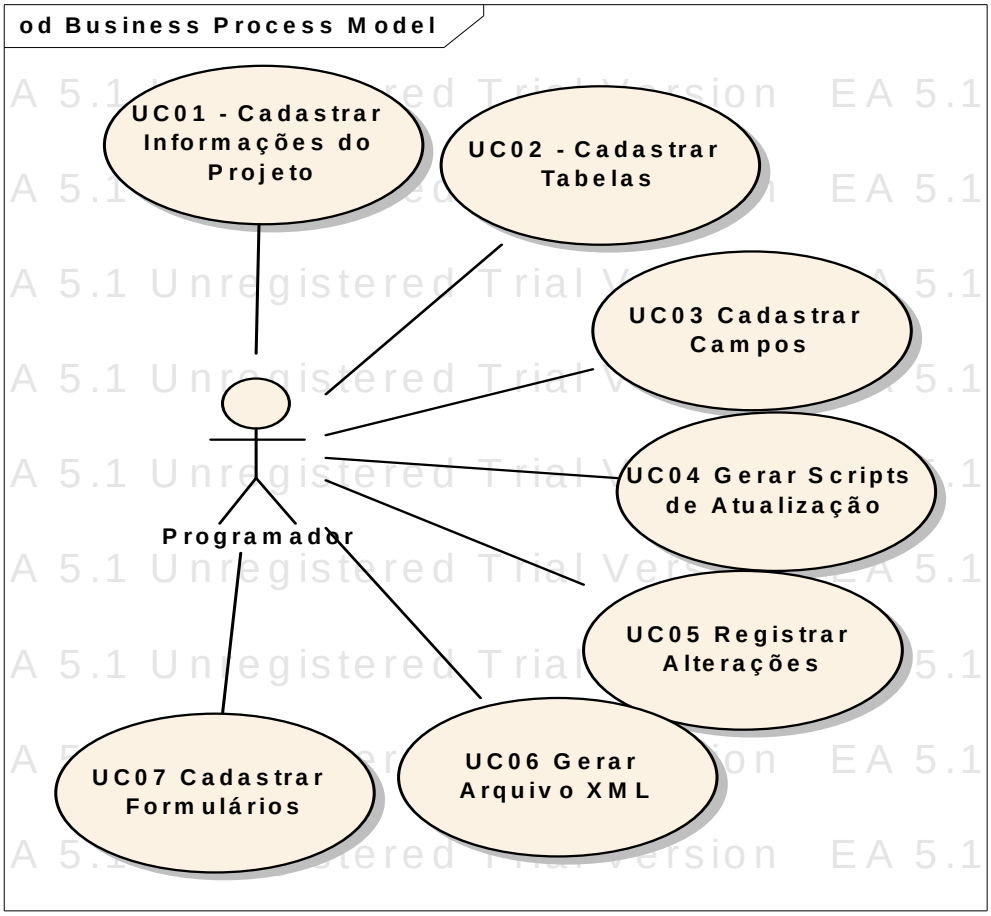

Figura 01- Diagrama de caso de uso.

Nos Quadros 03 até 09 são exibidos os detalhamentos dos casos de uso.

UC01 Cadastrar informações do projeto
Sumário: O programador abre um projeto salvo previamente, e informa os parâmetros solicitados pelo projeto. **Ator principal**: Programador. **Precondições**: Ter um *schema* criado no *MySQl* com o nome do projeto; **Fluxo principal:** a) programador abre o projeto clicando no ícone abrir; b) programador entra no menu utilitários informações do projeto; c) programador informa os dados solicitados pela ferramenta; d) programador grava as informações pressionando o botão gravar.

Quadro 03 - Cadastrar informações do projeto.

UC02 Cadastrar tabelas

Sumário: O programador poderá cadastrar as tabelas do sistema, sua finalidade dentro da aplicação, e a data de criação.

Ator principal: Programador.

Precondições: Ter configurado as informações do projeto;

Fluxo principal:

a) programador abre o projeto clicando no ícone abrir;

b) programador entra no menu cadastrar tabelas;

c) programador pressiona o botão “+” para adicionar uma tabela;

d) sistema limpa os campos que estão na tela;

e) programador alimenta os dados solicitados;

f) programador grava as informações pressionando o botão gravar.

Quadro 04 - Cadastrar tabelas.

UC03 Cadastrar campos.

Sumário: O programador poderá cadastrar os campos de uma tabela seus relacionamentos, atributos,valor mínimo dos campos, o componente que representara a informação, sua máscara de entrada, o formato da apresentação

Ator principal: Programador.

Precondições:

a) ter configurado as informações do projeto;

b) ter previamente criado a tabela.

Fluxo principal:

a) programador abre o projeto clicando no ícone abrir;

b) programador entra no menu cadastrar campos;

c) programador pressiona o botão “+” para adicionar um novo campo;

d) sistema limpa os campos que estão na janela;

e) programador alimenta os dados solicitados;

f) programador grava as informações pressionando o botão gravar.

Quadro 05 - Cadastrar campos.

UC04 Gerar scripts de atualização.

Sumário: o programador poderá gerar os scripts de atualização que conterão os executáveis e arquivos que compõe o sistema final**.**

Ator principal: Programador.

Precondições:

a) ter configurado as informações do projeto;

b) ter já compilado o sistema usando o Delphi.

Fluxo principal:

a) programador abre o projeto clicando no ícone abrir;

b) programador pressiona o botão "+ arquivo" para adicionar um novo arquivo;

c) sistema abre a janela de seleção de arquivos;

d) programador seleciona o arquivo;

e) programador informa o diretório de destino do aplicativo;

f) programador informa o diretório aonde será gerado o *script*;

g) programador gera o *script* pressionando o botão copiar.

Quadro 06 - Gerar scripts de atualização.

UC05 Registrar alterações.

Sumário: programador poderá registrar alterações nos *releases* do sistema final**.**

Ator principal: Programador.

Precondições:

a) ter configurado as informações do projeto;

Fluxo principal:

a) programador abre o projeto clicando no ícone abrir;

b) programador seleciona a data que foi feita a alteração;

c) sistema exibe todas as alterações feitas naquela data;

d) programador pressiona o botão "+"

e) sistema limpa os campos que estão na janela;

c) programador informa os dados solicitados pela ferramenta;

e) programador pressiona o botão gravar.

Quadro 07 - Registrar alterações.

UC06 Gerar arquivos XML.
Sumário: o programador poderá exportar informações armazenadas em uma ou mais tabelas da ferramenta no formato XML. **Ator principal**: Programador. **Precondições**: a) ter configurado as informações do projeto; **Fluxo principal:** a) programador abre o projeto clicando no ícone abrir; b) programador vai ao menu utilitários exportar dados para xml; c) sistema abre a caixa de seleção para salvar o arquivo; e) programador pressiona o botão salvar.

Quadro 08 - Gerar arquivos XML.

UC07 Cadastrar formulários
Sumário: o programador poderá definir os formulários da aplicação, em que item do menu ele será incluído e a tabela utilizada e a forma de incremento da chave primária, poderá definir quais guias compõe o formulário, quais campos cada guia irá conter. **Ator principal**: Programador. **Precondições**: a) ter configurado as informações do projeto; b) ter criado a tabela que será utilizada no formulário; c) ter definido os campos da tabela que será utilizada no formulário. **Fluxo principal:** a) programador abre o projeto clicando no ícone abrir; b) programador entra no menu cadastrar formulários; c) programador pressiona o botão “+”; e) sistema limpa os campos do formulário; f) programador informa qual tabela ou tabelas irão serão utilizadas neste formulário.

Quadro 09 - Cadastrar formulários.

3.2.2 Diagrama de atividades

Na figura 02 encontra-se o diagrama de atividades contendo a seqüência com que o programador deverá usar a ferramenta desde o cadastro do projeto até a entrega do sistema para o cliente.

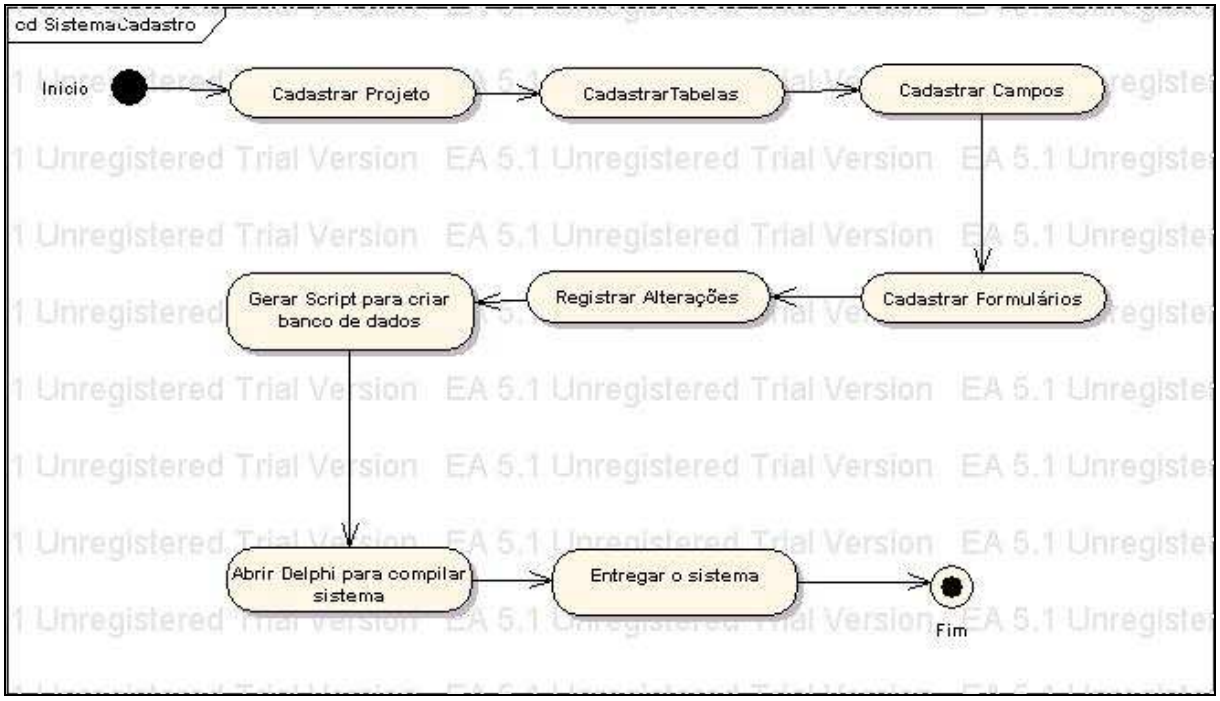

Figura 02 – Diagrama de atividades.

3.2.3 DIAGRAMA ENTIDADE RELACIONAMENTO LÓGICO

Na Figura 03 é apresentado o diagrama entidade relacionamento lógico da base de dados necessária para armazenar as informações para gerar o dicionário de dados.

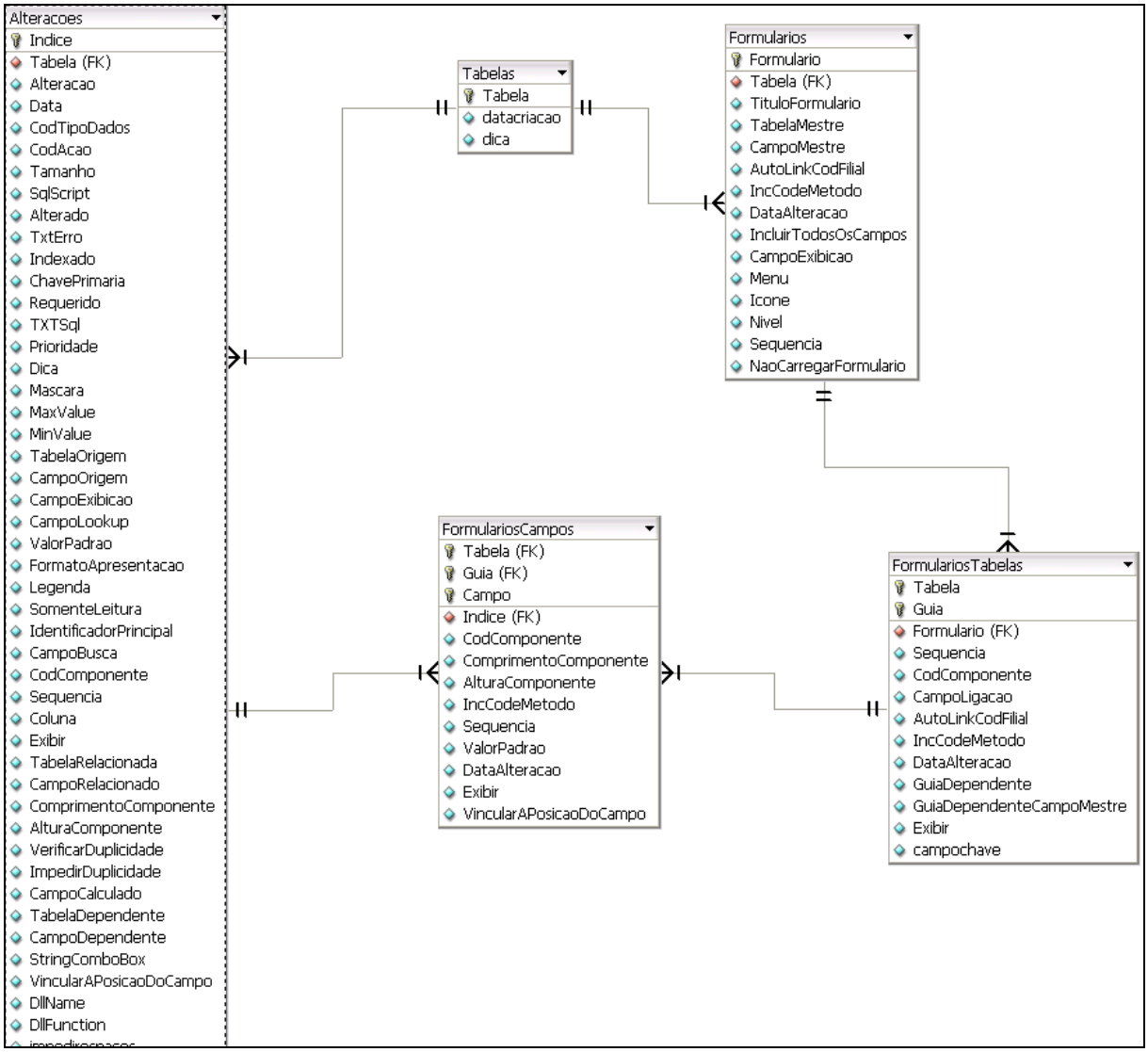

Figura 03 – Diagrama entidade relacionamento lógico

3.2.4 DIAGRAMA ENTIDADE RELACIONAMENTO FÍSICO

Na Figura 04 é apresentado o diagrama entidade relacionamento físico da base de dados necessária para armazenar as informações do dicionário de dados. As informações mais detalhadas sobre os atributos serão descritas no capitulo 4.2 que descreve a operacionalidade da ferramenta.

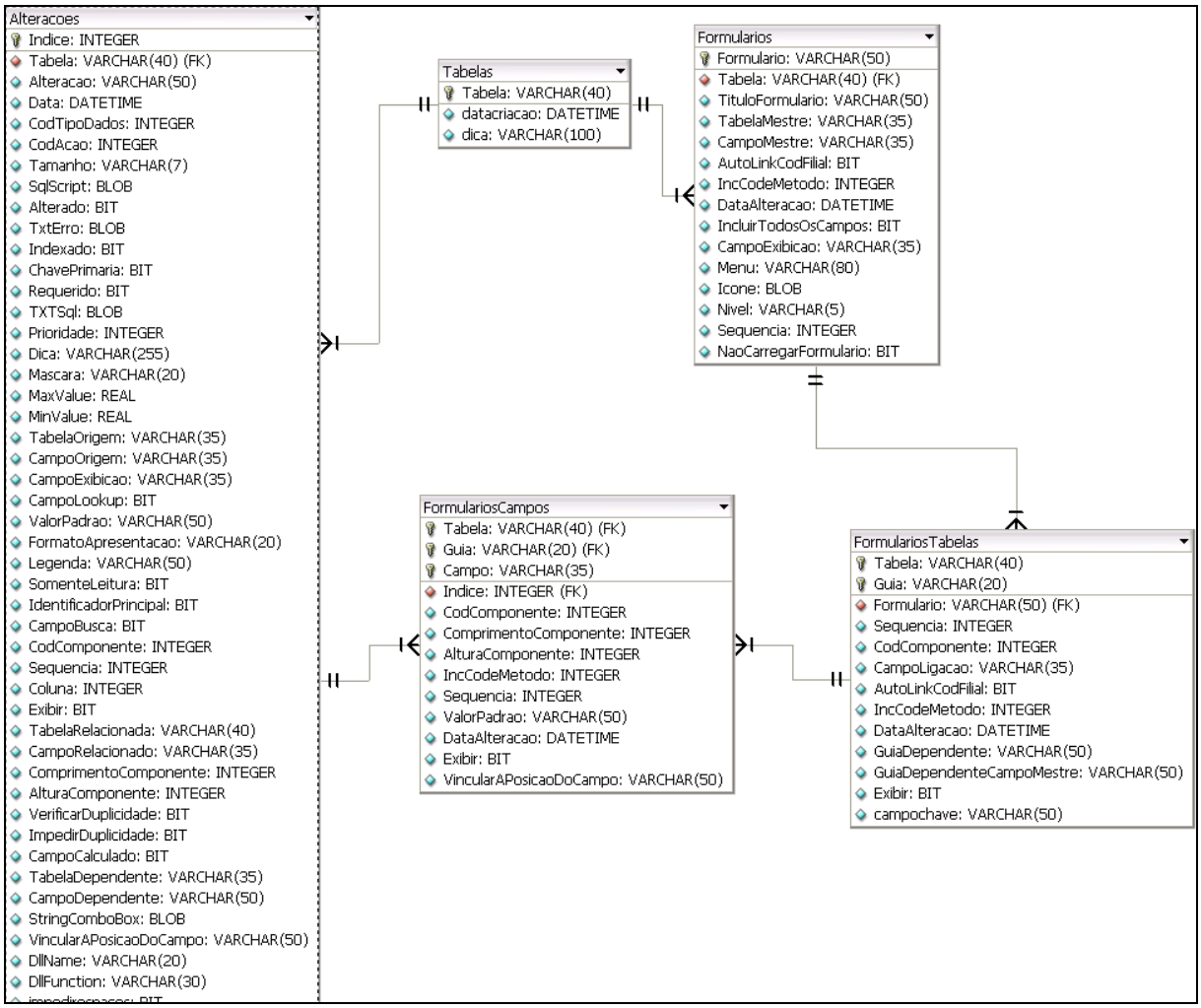

Figura 04 – Diagrama entidade relacionamento físico

3.3 IMPLEMENTAÇÃO

A seguir é descrito como foi implementada a ferramenta, detalhando as funções de acesso ao dicionário de dados, geração do script de atualização do sistema e a atualização dos arquivos no usuário final.

Por último é demonstrado como os componentes escritos em Delphi interpretam as informações geradas pela ferramenta e geram formulários em tempo de execução com as informações contidas no dicionário de dados do usuário final.

3.3.1 ACESSO AO DICIONÁRIO DE DADOS

A ferramenta construída neste trabalho tem que gravar os dados relativos ao projeto em um banco de dados que contém informações das tabelas, campos e formulários contidos no projeto. Para armazenar as informações foi escolhido o banco de dados Microsoft Access 97 por ser um banco facilmente portável entre as diversas versões do sistema operacional Windows. Para acessar os dados do Microsoft Access 97 foram escolhidos os componentes de acesso de dados *Data Access Objects* (ADO).

No quadro 10, segue a função que recebe o nome do arquivo do projeto, setando assim os componentes (ADO) para abrir o projeto selecionado.

```
procedure TD.OpenTables(FileName:String);
 Label
  Fim;
begin
  CloseDataBases;
//  Provider=Microsoft.Jet.OLEDB.4.0;User ID=Admin;Data Source=C:\Desenvolvimento\Rz_Builder\Upd_Base\Scripts\BluBordad

  Rz_DataBase_Update.Connected := false;
    Rz_DataBase_Update.ConnectionString := 'Provider=Microsoft.Jet.OLEDB.4.0;Data Source='+ FileName+';Persist Security
  Rz_DataBase_Update.Connected := true;

  RzSql(D.Aux,'select *from projeto',[]);

  AliasName     := d.Aux.fieldbyname('sistema').AsString;
  hostname      := 'localhost';
  UserName      := d.Aux.fieldbyname('username').AsString;
  Password      := d.Aux.fieldbyname('password').AsString;
```

Quadro 10 – função que seta os componentes ADO para o projeto selecionado.

3.3.2 GERAÇÃO DO SCRIPT DE ATUALIZAÇÃO

Para atualizar o sistema no usuário final é necessário que o programador gere um arquivo com todos os executáveis, dlls e qualquer outro arquivo que compõe o projeto.

Para isto foi utilizado o componente *Tarchiver* da biblioteca de componentes *Backuptools*, este componente compacta as informações e gera um arquivo único com todos os arquivos do projeto dentro dele.

No quadro 11 é demonstrado o código que percorre a lista de arquivos contida na tabela arquivos e gera o arquivo de *script* para atualização do projeto.

```
D.Arquivos.open;
D.Arquivos.First;
While not D.Arquivos.eof do begin
 SArquivo := D.Arquivos.fieldbyname('Arquivo').AsString;
  if GerarArquivo then begin
   D.Compact.Open;
     D.Compact.AddFile(SArquivo);
   D.Compact.Close;
  end
  else begin
   Destino := CbDestino.text +  Copy(SArquivo,3,Length(SArquivo));
   If not DirectoryExists(ExtractFileDir(Destino)) Then begin
     MsgE(['Caminho não encontrado ->',Destino]);
   end;
  end;

  If GerarArquivo then begin
    D.Compact.Close;
  end;

 d.Arquivos.Next;
End;
```

Quadro 11 – Geração do script de atualização.

3.3.3 ATUALIZAÇÃO DOS ARQUIVOS NO USUÁRIO FINAL

Quando o arquivo de script é descompactado no usuário final a ferramenta atualiza todos os arquivos do projeto em suas respectivas pastas que são definidas informando o parâmetro dirdestino na tabela arquivos.

No Quadro 12 é demonstrado como a rotina que realiza a atualização dos arquivos.

```
d.OpenTables(SExtractPath + '\' + sprojectName + '.dbu2');
D.Arquivos.Open;
While not d.Arquivos.Eof do begin
  Application.ProcessMessages;
  SExtractFileName     := d.Arquivos.fieldbyname('arquivo').AsString;
  LStatus.Caption      := 'Atualizando arquivo ' + ExtractFileName(SExtractFileName);
  SFileNameOrigem      := SExtractPath + '\' +  ExtractFileName(SExtractFileName);

  if SameText(d.Arquivos.fieldbyname('DirDestino').AsString,'ApDir') Then begin
    sFileDestino := RzGetSubDir(D.DirAlteracoes.ConfFileDir,2) + '\' + ExtractFileName(SExtractFileName);
  end
  else if SameText(d.Arquivos.fieldbyname('DirDestino').AsString,'DllDir') Then begin
    sFileDestino := D.DirAlteracoes.DllDir + '\' + ExtractFileName(SExtractFileName);
  end
  else if SameText(d.Arquivos.fieldbyname('DirDestino').AsString,'WinDir') Then begin
    sFileDestino := D.DirAlteracoes.WinDir + '\' + ExtractFileName(SExtractFileName);
  end
  else
   Erro('Não definido o Dir destino no arquivo '+d.Arquivos.fieldbyname('arquivo').AsString);

   if not RzCopyFile(SFileNameOrigem,sFileDestino) Then
     Erro(['Não é possível atualizar o arquivo "',sFileDestino,'"',#13,
       ' 1° verifique se o arquivo esta sendo usado por algum computador na rede ou aberto na maquina local',#13,
       ' 2° se nenhum computador esta usando o arquivo atual recomenda-se reinializar o servidor ou o computador principal',#13,
       ' A operação não pode proseguir']);

   DeleteFile(SFileNameOrigem);
  d.Arquivos.Next;
end;//fim while
```

Quadro 12 – Atualização dos arquivos do projeto.

Após a atualização dos arquivos do projeto é feita a atualização dos campos no dicionário de dados do cliente, bem como são adicionados novos campos e tabelas na base de dados do cliente.

Para atualização do dicionário de dados, criação de tabelas, campos e relacionamentos foram utilizados os componentes de acesso de dados Zeos devido a capacidade de acessar múltiplos bancos de dados, possibilitando assim que outros trabalhem implementem esta funcionalidade.

No quadro 13 é demonstrado como é executada a rotina de atualização dos campos no usuário final.

```
D.Alteracoes.First;
While Not D.Alteracoes.Eof Do begin
 If D.Alteracoes.fieldByName('Alterado').AsBoolean Then
   Goto NextAlt;

 D.Alteracoes.Edit;
  D.Alteracoes.fieldByName('TxtSql').Asstring  := '';
  D.Alteracoes.fieldByName('TxtErro').AsString := '';

 Case D.Alteracoes.fieldByName('CodAcao').AsInteger Of
   1: Alterado := CriarTabela;
   2: Alterado := DeleteTabela(D.Alteracoes.fieldByName('Alteracao').AsString);
   4: Alterado := AlterField(D.Alteracoes.fieldByName('Alteracao').AsString);
   5: Alterado := ExecScript;
   6: Alterado := CreateField(D.Alteracoes.fieldByName('Alteracao').AsString);
   7: Alterado := DeleteField(D.Alteracoes.fieldByName('Alteracao').AsString);
   8: Alterado := RenameField(D.Alteracoes.fieldByName('Alteracao').AsString);
  10: Alterado := CreateIntegridade(D.Alteracoes.fieldByName('Alteracao').AsString);
  11: Alterado := CreateIndex(D.Alteracoes.fieldByName('Alteracao').Asstring);
  14: Alterado := CreateIndex(var  TD.Alteracoes: TADOQuery - Dm.pas (32)eracao').AsString);
  15: Alterado := DeleteIntegridade(D.Alteracoes.fieldByName('Alteracao').AsString);
  16: Alterado := InsereUsuarioPadrao;
  17: Alterado := CriatabelaCampos;
 End;

 If Not (D.Alteracoes.State in [DsEdit,DsInsert]) Then
  D.Alteracoes.Edit;

  D.Alteracoes.fieldByName('Alterado').AsBoolean := Alterado;
 D.Alteracoes.Post;


 NextAlt:
   D.Alteracoes.Next;
End;
```

Quadro 13 - Atualização dos campos no usuário final.

No Quadro 14 é demonstrada como ferramenta executa a criação de um índice na base de dados do usuário final chamada na função *createindex* no Quadro 13 usando comandos *Data Definition Language* (DDL).

```
   If D.Alteracoes.FieldByName('ChavePrimaria').AsBoolean Then Begin
     UpdateExec('ALTER TABLE '+GetTableName+' add CONSTRAINT ' + IndexName +
     ' PRIMARY KEY ('+FieldName+')');
   End
   Else if D.Alteracoes.FieldByName('CodAcao').AsInteger = 14 Then//Indice Unico
     UpdateExec('CREATE unique INDEX ' + IndexName + ' ON ' + GetTableName + '('+FieldName+')')
   Else Begin
     UpdateExec('CREATE INDEX ' + IndexName + ' ON ' + GetTableName + '('+FieldName+')')
   End;
   Result := True;
 Except
   Result := False;
   GravaErro('Erro ao criar indice' + FieldName);
 End;
```

Quadro 14 - Alteração de um índice no usuário final.

3.3.4 CRIAÇÃO EM TEMPO DE EXECUÇÃO DOS FORMULÁRIOS.

Quando o usuário final executa uma aplicação escrita com o método de programação tradicional usando a ferramenta Delphi é chamado um formulário que está contido dentro do executável, no caso com o uso desta ferramenta componentes escritos em Delphi geram estes formulários em tempo de execução de acordo com os dados definidos na ferramenta.

O Quadro 15 demonstra um fragmento de código que cria guias que compõem um formulário. Para isto é criado o componente *trzscrollcadastros* descendente da classe *tscrollbox* que se encarrega de criar os campos daquela guia.

```
RzSql(AuxQ,'Select *from FormulariosTabelas Where Formulario = :D1 and Exibir = :D2 order by sequencia',[pFormulario,True]);
While not AuxQ.Eof do begin
 Pagina                      := TTabSheet.Create(CadForm);
 Pagina.Caption              := AuxQ.fieldByName('Guia').AsString;
 Pagina.PageControl          := CadForm.Paginas;
 ScrollCampos                := TRzScrollCadastros.create(CadForm);
 ScrollCampos.RzDataCenter   := RzDataCenter;
 ScrollCampos.TabelaPrincipal := CadForm.DDados;
 ScrollCampos.Parent         := Pagina;
 ScrollCampos.Align          := alClient;
 ScrollCampos.Formulario     := pFormulario;
 ScrollCampos.Name           := 'Scroll'+RetiraCaracteresEspeciais(AuxQ.fieldByName('Guia').AsString);
 ScrollCampos.Guia           := AuxQ.fieldbyname('Guia').AsString;
 ScrollCampos.GuiaDependente:= AuxQ.fieldbyname('GuiaDependente').AsString;
                               |
 ScrollCampos.DataSource     := ScrollCampos.CriaQuery(AuxQ.fieldByname('IncCodeMetodo').Value,
              AuxQ.FieldByName('AutoLinkCodFilial').Value,AuxQ.fieldByname('CampoLigacao').AsString,
              AuxQ.fieldByName('Tabela').AsString,AuxQ.fieldByname('CampoChave').AsString,CadForm.DDados,
              AuxQ.fieldByName('Tabela').AsString +CampoMestre);
 ScrollCampos.CriaCampos;
 AuxQ.Next;
end;
```

Quadro 15 - Criação do componente que gerencia a distribuição dos campos na guia do formulário.

No Quadro 16 é demonstrado como o componente trzscrollcadastros interpreta o dicionário de dados do usuário final e gera os campos.

```
While not Campos.Eof do begin
  sDescricao := SetCaptionComponente;

  Case Campos.FieldByName('CodComponente').AsInteger of
   1:begin
     CriaRzDbEdit(sDescricao,Campos.FieldByName('vincularaposicaodocampo').AsString);
   end;
   2:begin
     CriaRzDbDateEdit(sDescricao,Campos.FieldByName('vincularaposicaodocampo').AsString);
   end;
   3:begin
     CriaRzLookupCombo(sDescricao,Campos.FieldByName('vincularaposicaodocampo').AsString);
   end;
   4:begin
     CriaRzDbRadioGroup(sDescricao,Campos.FieldByName('vincularaposicaodocampo').AsString);
   end;
   5:begin
      CriaRzDbGrid(CriaGroupBox(Guia,0,0,0,0,alClient) ,DataSource);
   end;
   6:begin
     CriaRzDbCheckBox (sDescricao,Campos.FieldByName('vincularaposicaodocampo').AsString);
   end;
   7:begin
     CriaRzDbMemo(sDescricao,Campos.FieldByName('vincularaposicaodocampo').AsString);
   end;
   8:begin
     CriaRzDbComboBox(sDescricao,Campos.FieldByName('vincularaposicaodocampo').AsString)
   end;
   9:begin
     CriaRzDbEditCNPJ(sDescricao,Campos.FieldByName('vincularaposicaodocampo').AsString);
   end;
```

func TDataSet.FieldByName: function(const FieldName: String): TField - DB.pas (9302)

Quadro 16 - Criação dos componentes no formulário.

No Quadro 17 é demonstrado como é criado um componente *Tdbedit* em tempo de execução chamado no quadro 16 com a função criarzdbedit.

```
Function TRzScrollCadastros.CriaRzDbEdit(pDescricao,pVincularAPosicaoDoCampo:String):TRzDbEdit;
Var
 DbEdit:TrzDbEdit;
 begin
   DbEdit               := TRzDbEdit.create(Self);
   DbEdit.DataSource    := DataSource;
   Dbedit.Parent        := Self;
   DbEdit.DataField     := Campos.fieldByname('Campo').AsString;
   DbEdit.Name          := 'RzDbEdit'+ DbEdit.DataField;

   DbEdit.Width         := SetComprimentoComponente(DbEdit.DataField);
   SetDimensoesComponente(DbEdit);



   if pVincularAPosicaoDoCampo = '' Then begin
    Inc(PosTop,DbEdit.Height+2);
    Dbedit.Top           := PosTop;
    DbEdit.Left          := PosLeft;
    CreateLabel(DbEdit,pDescricao);
   end
   else begin
      CreateLabel(DbEdit,pDescricao);
      SetPosComponenteVinculado(DbEdit,pVincularAPosicaoDoCampo);
   end;



   Result := DbEdit;
end;
```

Quadro 17 - Criação do componente em tempo de execução

4 ESTUDO DE CASO

Para atestar a funcionalidade da ferramenta desenvolveu-se um pequeno sistema para cadastro de clientes, representantes e cidades. O objetivo deste pequeno sistema é permitir ao usuário cadastrar clientes vinculados com o representante que o atende, a cidade que este cliente está estabelecido, além de permitir cadastrar vários endereços adicionais para o cliente.

Na figura 05 é demonstrado o diagrama de caso de uso do sistema em questão.

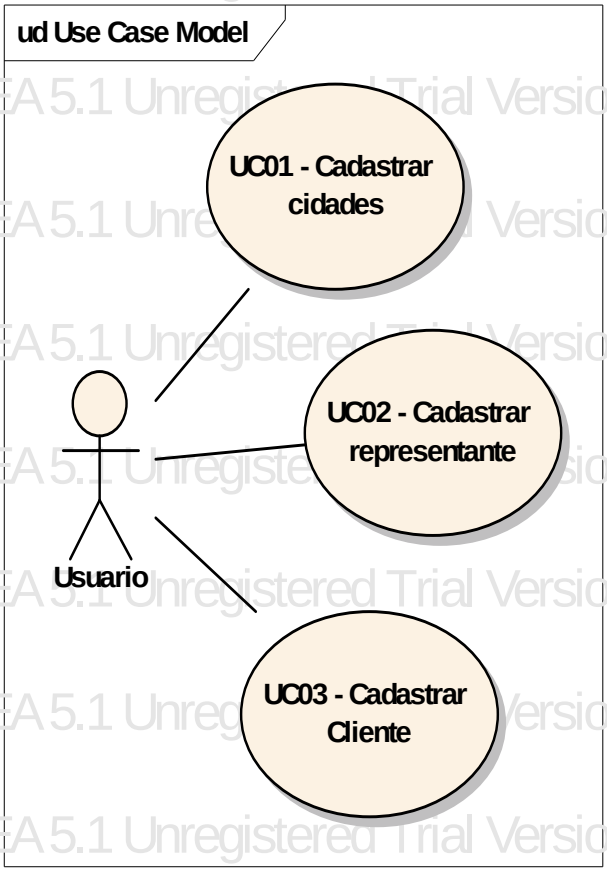

Figura 05 – Diagrama de caso de uso do sistema para cadastro de clientes.

4.1 DETALHAMENTO DOS CASOS DE USO

Nos quadros 18 até 20 são descritos os casos de uso do sistema para cadastro de clientes.

UC01 Cadastrar cidades

Sumário: o usuário cadastra as cidades que futuramente vão servir para alimentar o cadastro de clientes e representantes.

Ator principal: Usuário.

Precondições:

a) ter acesso ao sistema

Fluxo principal:

a) usuário abre o aplicativo;

b) usuário entra no menu cadastrar cidades;

c) usuário pressiona o botão “+” ;

d) sistema limpa os campos do formulário;

e) usuário informa os dados solicitados no cadastro;

f) usuário pressiona o botão gravar.

Quadro 18 - Cadastrar cidades.

UC02 Cadastrar representantes

Sumário: o usuário cadastra os representantes que futuramente vão servir para alimentar o cadastro do cliente.

Ator principal: Usuário.

Precondições:

a) ter acesso ao sistema

Fluxo principal:

a) usuário abre o aplicativo;

b) usuário entra no menu cadastrar representantes;

c) usuário pressiona o botão “+” ;

d) sistema limpa os campos do formulário;

e) usuário informa os dados solicitados no cadastro

f) usuário pressiona o botão gravar.

Quadro 19 - Cadastrar representantes.

UC03 Cadastrar Clientes
Sumário: o usuário cadastra o cliente vinculando-o a sua cidade, representante e informa os endereços adicionais. **Ator principal**: Usuário. **Precondições**: a) ter acesso ao sistema **Fluxo principal:** a) usuário abre o aplicativo; b) usuário entra no menu cadastrar clientes; c) usuário pressiona o botão “+” ; d) sistema limpa os campos do formulário; e) usuário informa os dados solicitados no cadastro; f) usuário pressiona o botão gravar.

Quadro 20 - Cadastrar clientes.

4.2 OPERACIONALIDADE

Nesta sessão será descrito passo a passo como se implementar o sistema proposto acima usando a ferramenta construída neste trabalho, juntamente com a informações dos campos de cada formulário para ajudar a compreender por completo os recursos da ferramenta.

4.2.1 Criando um projeto

Esta secção irá demonstrar as funcionalidades da janela cadastrar projetos. Para acessar esta janela deve-se ir ao menu cadastrar-> informações do projeto. Na Figura 06 está demonstrada a janela para cadastrar projeto.

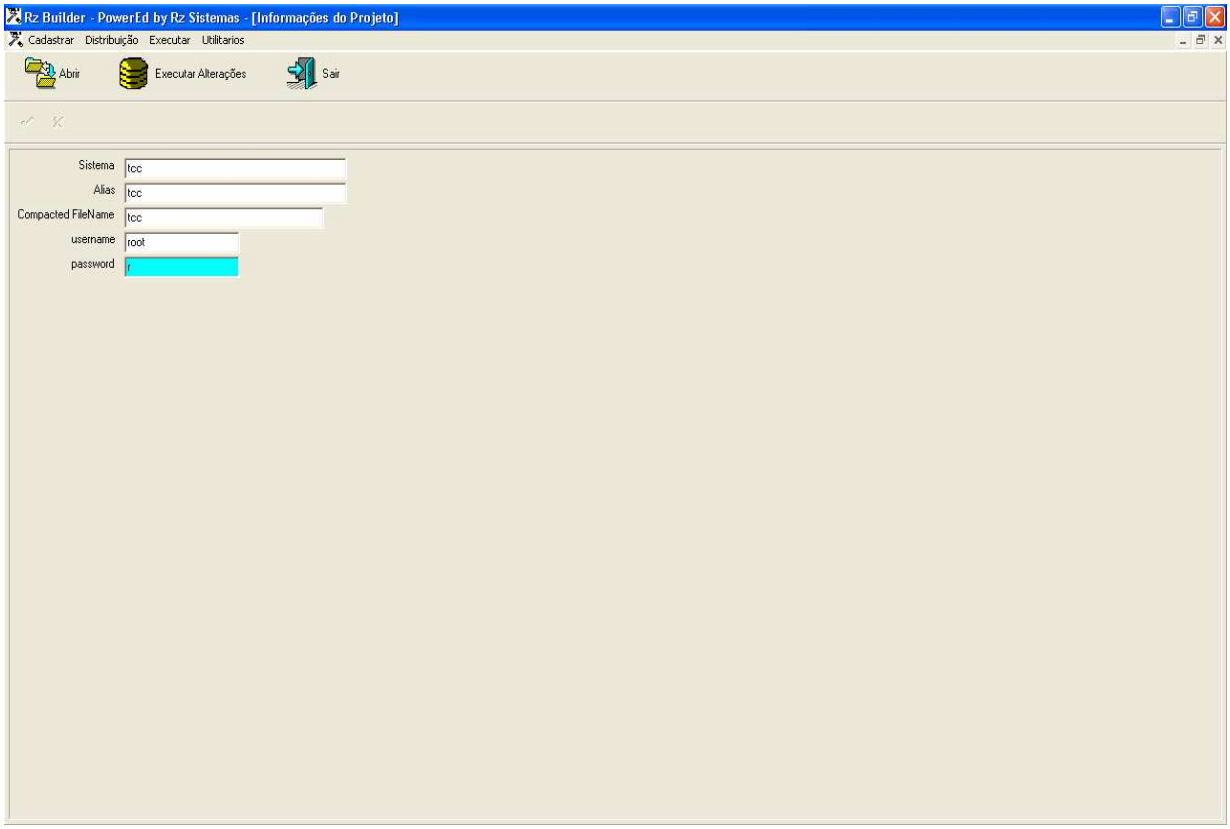

Figura 06 – Cadastrar projeto

A seguir é apresentado um detalhamento da finalidade dos campos exibidos na figura 06:

a) sistema: informa o nome do sistema que será tratado pelo projeto, neste caso tcc;
b) alias: informa qual o nome do alias ou *schema* no banco de dados onde serão criadas as tabelas e dicionário de dados do projeto, neste caso tcc;
c) compactedfilename: informa qual o nome do arquivo que será gerado para atualização do aplicativo, neste caso tcc;
d) username: informa o nome do usuário do banco de dados, neste caso “root”;
e) password: informa qual a senha para acesso ao banco de dados, neste caso “r”.

4.2.2 Cadastrar tabelas

Esta seção irá demonstrar as funcionalidades da janela cadastrar tabelas. Para acessar a janela cadastrar tabelas deve ir ao menu cadastrar-> tabelas.

Na Figura 07 está demonstrada a janela para cadastrar tabelas bem como suas

funcionalidades.

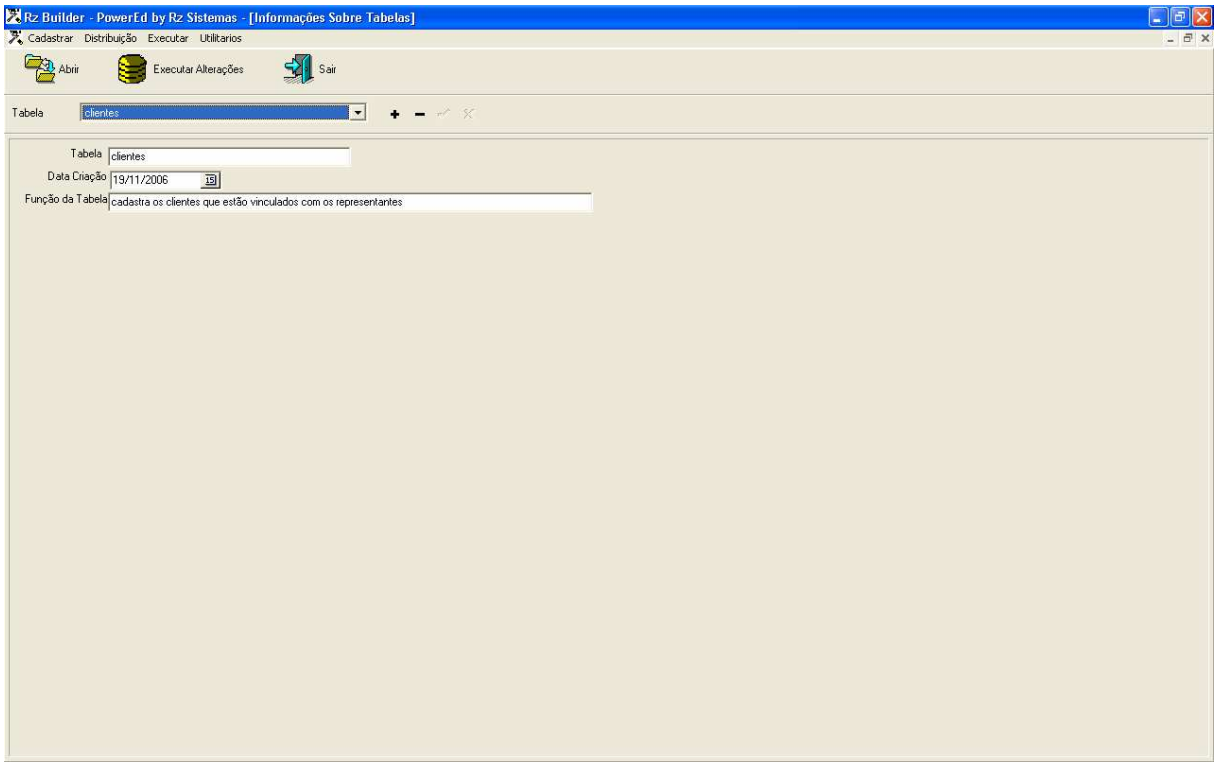

Figura 07 - Cadastrar tabelas

A seguir é apresentado um detalhamento da finalidade dos campos exibidos na figura 07:

a) tabela: determina o nome da tabela no sistema, neste caso clientes;
b) data criação: informa a data que a tabela foi criada dentro do sistema;
c) função da tabela: informa qual a finalidade da tabela dentro do sistema , neste caso armazena os clientes que estão vinculados com os representantes.

4.2.3 Cadastrar Campos

Esta secção irá demonstrar as funcionalidades da janela cadastrar campos. Para cadastrar os campos de uma tabela deve se acessar o menu cadastrar-> campos.

4.2.3.1 Cadastrar campos (guia padrão)

Na figura 08 encontra-se o formulário para cadastrar os campos de uma tabela bem como seus atributos, valores mínimos, máximos, componente utilizado para representação dos dados, máscara de entrada, formato da apresentação e relacionamento.

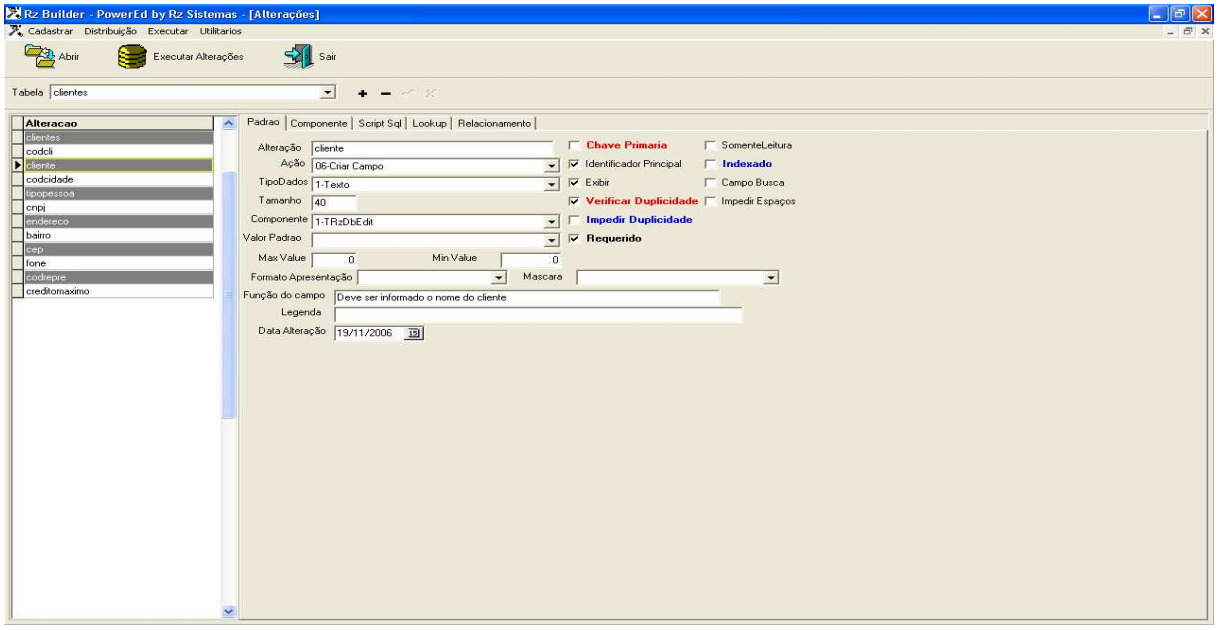

Figura 08 - Cadastrar campos (guia padrão)

A seguir é apresentado um detalhamento da finalidade dos campos exibidos na figura 08:

a) alteração: indica o nome do campo ou tabela que será criada ou alterada , neste caso cliente;
b) ação: indica qual a ação será executada com o registro selecionado, estão disponíveis as ações cria tabela, apagar tabela,executar script,criar campo, apagar campo, renomear campo, renomear tabela, criar estrutura básica das tabelas (dicionário de dados), neste caso a ação será criar campo ;
c) tipo de dados: indica qual o tipo de dados do campo, neste caso texto;
d) tamanho do campo: indica qual o tamanho do campo, neste caso 40;
e) componente: indica qual o componente será utilizado no formulário no qual este campo será usado (utilizado também como um atalho para não ter que entrar na guia componente), neste caso será colocado o componente como TRzDbEdit ;

f) valor padrão: indica caso haja necessidade de ao inserir um registro inicia-lo com um valor pré determinado;
g) max value: indica qual o valor máximo do campo, neste caso o valor do campo creditomáximo será 10000;
h) min value: indica qual o valor mínimo do campo, neste caso 0;
i) formato apresentação: indica qual o formato que o campo será exibido, neste caso a máscara será “#,##0.00;-#,##0.00” para o campo credito máximo ;
j) máscara: indica qual a mascara de entrada dos dados, no caso do campo fone a máscara será “!(9xx00)9000-0000;1;”;
k) função do campo: indica qual a função do campo no sistema, no caso do campo creditomaximo é que define o crédito máximo que o cliente possui;
l) legenda: indica qual o nome que será exibido em seu rótulo correspondente, no caso do campo codcidade a legenda deste campo deve ser “Cidade”;
m) data alteração: indica a data que foi criado ou alterado o campo para na hora da atualização, O sistema executar apenas as mudanças feitas a partir da data da ultima atualização;
n) chave primária: indica se um campo é chave primária da tabela, no caso da tabela clientes a chave primária é codcli;
o) identificador principal: indica se este campo contém a descrição do campo, no caso do cadastro de clientes o identificador principal seria o campo cliente pois é o nome do cliente em questão;
p) exibir: indica se o campo deve ser exibido no formulário de cadastro correspondente ou é apenas um campo interno do sistema;
q) verifica duplicidade: mesmo que o campo não seja a chave primaria o sistema irá verificar se já existe um registro com a mesma descrição cadastrada, um exemplo seria num cadastro de clientes verificar se o cnpj daquele cliente já esta cadastrado;
r) impedir duplicidade: neste caso seria impedido de se cadastrar um registro com o mesmo valor já atribuído a outro registro;
s) requerido: indica a obrigatoriedade de informar um valor para aquele campo, no caso foi definido que o campo cliente é um campo obrigatório;
t) somente leitura indica se aquele campo será somente para visualização impedindo sua alteração;
u) indexado: indica se o campo é indexado;
v) campo busca: indica se o campo será usado futuramente para localizar o registro,

no caso deste sistema o cliente poderá ser localizado tanto pelo nome, telefone ou cnpj.

4.2.3.2 Cadastrar campos (guia componente)

Na figura 09 encontra-se a guia componente do formulário para cadastrar campos que tem como principal finalidade a definição mais detalhada do componente que representará o campo.

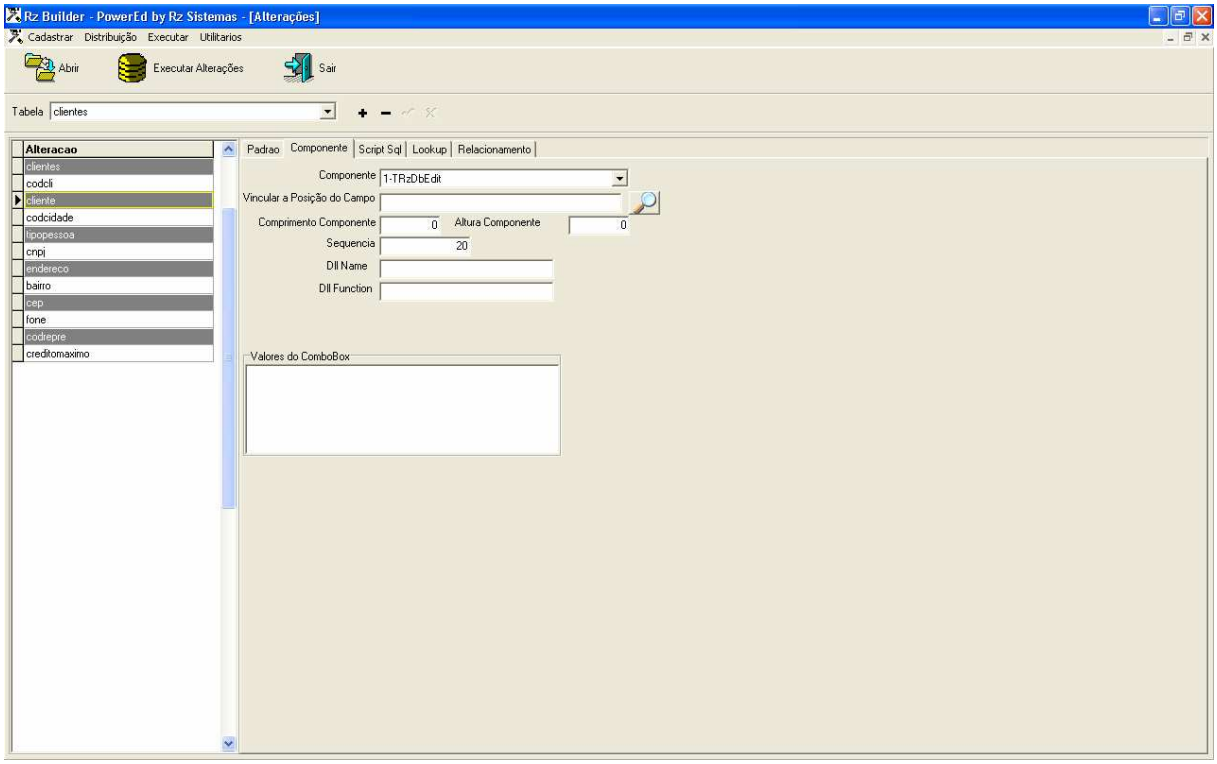

Figura 09- Cadastrar campos (guia componente)

A seguir é apresentado um detalhamento da finalidade dos campos exibidos na figura 09:

a) componente: indica qual o componente que será utilizado para representar a informação no formulário de destino, no caso do campo cliente foi escolhido o componente TRzDbEdit.
b) vincular a posição do campo: indica se o campo selecionado deve ficar vinculado a posição de algum outro campo no formulário;

c) comprimento do componente: indica qual comprimento o componente deve ter, caso não tenha nenhum valor padrão atribuído o sistema calcula o melhor tamanho para o componente;
d) altura componente: indica qual altura o componente deve ter, caso não tenha nenhum valor padrão atribuído o sistema calcula o melhor tamanho para o componente;
e) seqüência: indica qual a seqüência que este campo estará no formulário (ordem de tabulação) ;
f) dllname: caso haja necessidade de um tratamento mais complexo para o valor do campo é possível especificar o nome de uma biblioteca que o sistema chamará;
g) dll function: indica qual função será chamada na biblioteca.
h) valores do combobox: caso o componente for um tdbcombobox é possível definir quais os valores possíveis para aquele campo, no caso do cadastro de clientes o campo tipopessoa poderá conter os valores “F” ou “J”;

4.2.3.3 Cadastrar Campos (guia script sql)

Na figura 10 encontra-se a guia *script SQL* do formulário cadastrar campos, que tem como principal objetivo a definição de um *script SQL* manualmente.

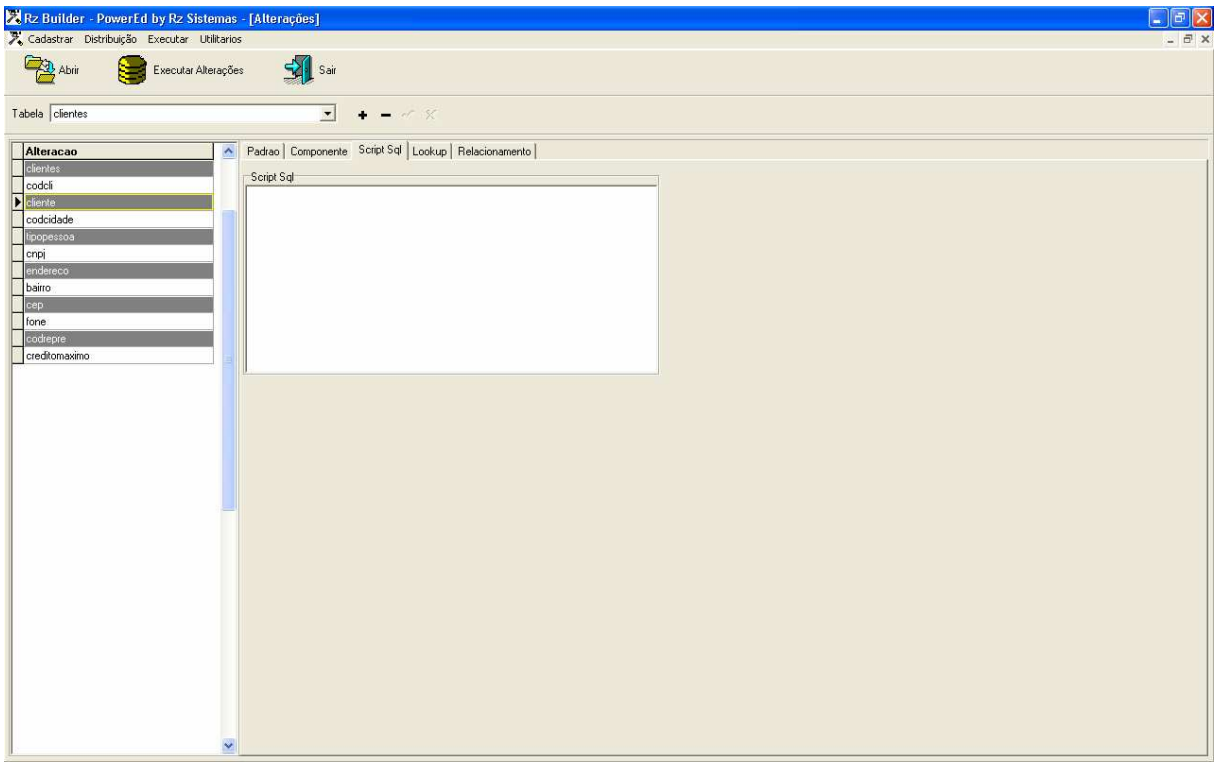

Figura 10 – Cadastrar campos guia script sql

4.2.3.4 Cadastrar Campos (guia lookup)

Na figura 11 encontra-se a guia *lookup* do formulário cadastrar campos que tem como principal finalidade a definição de um campo que é relacionado com uma tabela estrangeira e deverá exibir o nome no lugar do código relacionado.

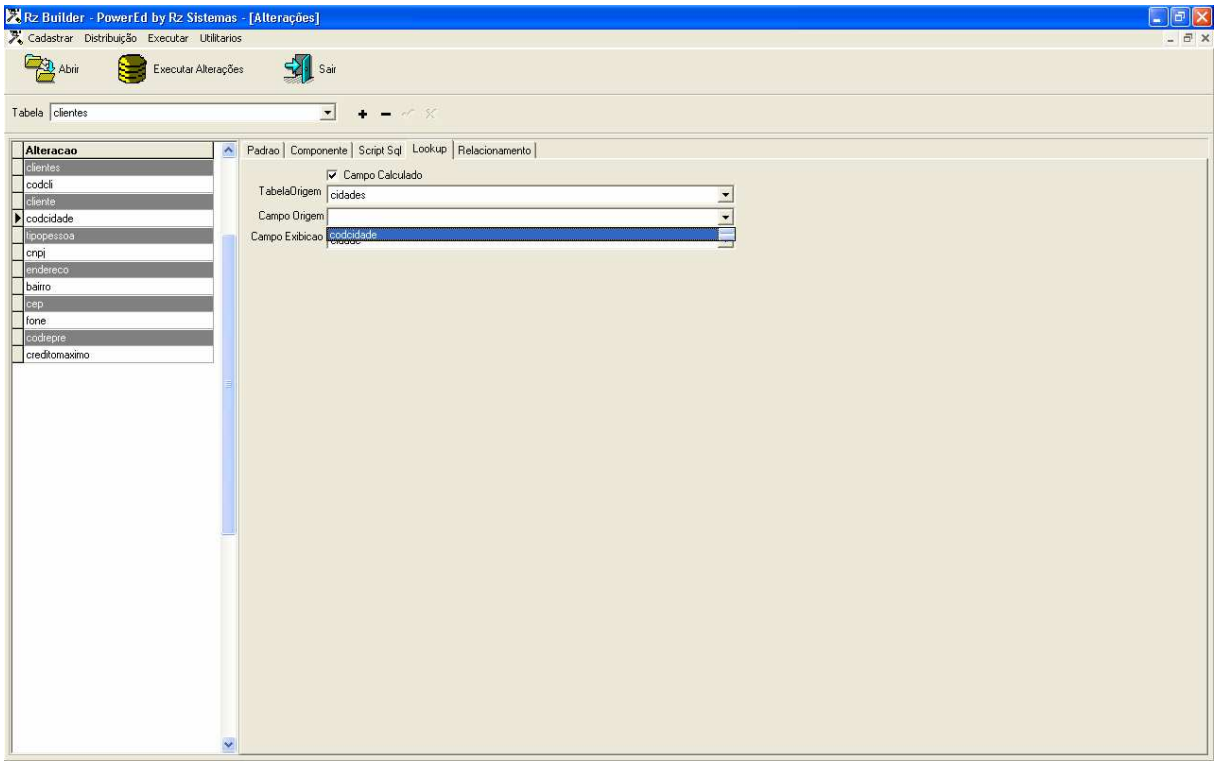

Figura 11 – Cadastrar campos guia lookup

A seguir é apresentado um detalhamento da finalidade dos campos exibidos na figura 11:

a) campo calculado: indica se o campo tem algum relacionamento com outra tabela no caso do campo codcidade este campo será habilitado;
b) tabela origem: indica qual tabela é relacionada com o campo especificado, , neste caso é cidade;
c) campo origem: indica qual a chave primaria da tabela relacionada se relaciona com o campo especificado, neste caso o campo origem é codcidade;
d) campo exibição: indica qual campo será exibido. neste caso sera relacionado o

campo codcidade da tabela clientes com o campo codcidade da tabela cidades e o valor que será exibido será o campo cidade.

4.2.3.5 Cadastrar Campos (guia relacionamento)

Abaixo segue um detalhamento da finalidade dos campos exibidos na figura 12 cuja finalidade principal é definir o relacionamento entre o campo e tabela que contém os dados.

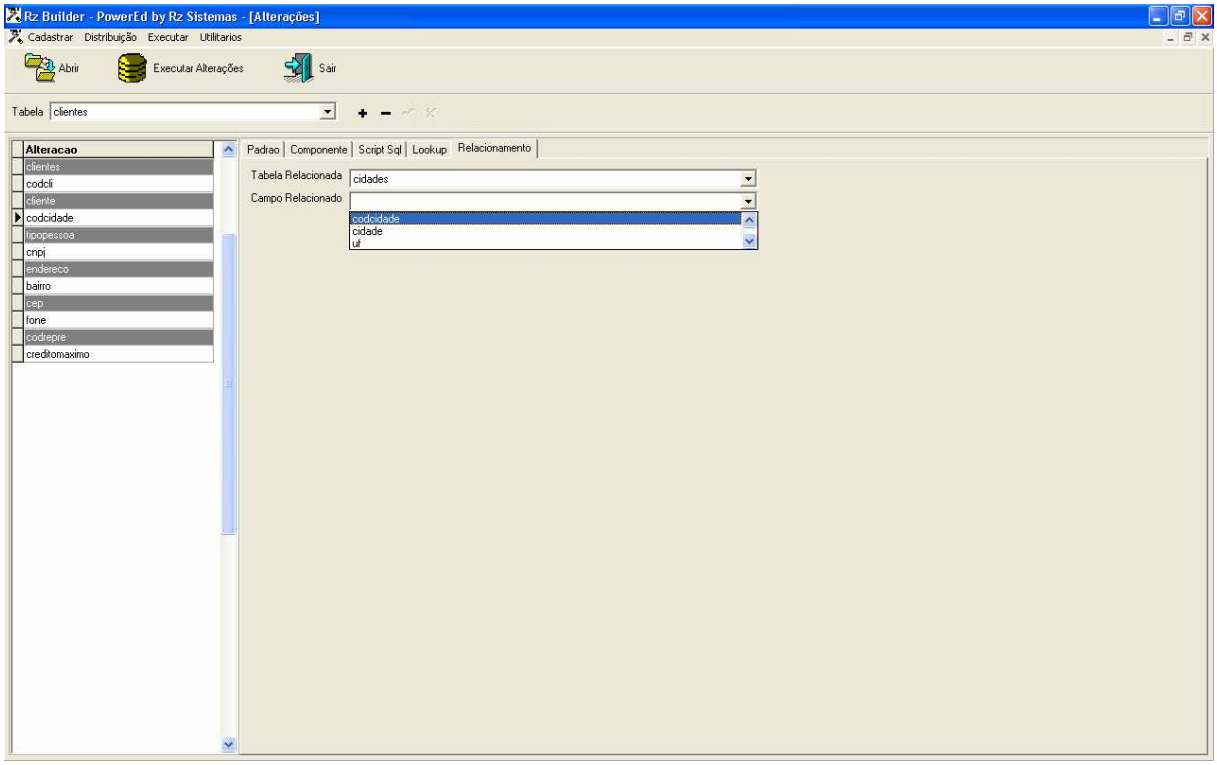

Figura 12 – Definir relacionamento

A seguir é apresentado um detalhamento da finalidade dos campos exibidos na figura 12:

a) tabela relacionada : indica qual é tabela que contem as informações relacionadas com o campo, neste caso a tabela relacionada é cidades;
b) campo relacionado: indica qual o campo que esta relacionado com o campo em questão neste campo será o campo codcidade.

4.2.4 Cadastro de formulários

Esta secção irá demonstrar as funcionalidades da janela cadastrar formulários. Para cadastrar os formulários acessar o menu cadastrar-> formulários;

4.2.4.1 Cadastrar Formulários (guia cadastro)

Na figura 13 encontra-se a guia cadastro do formulário para cadastrar os formulários do projeto.

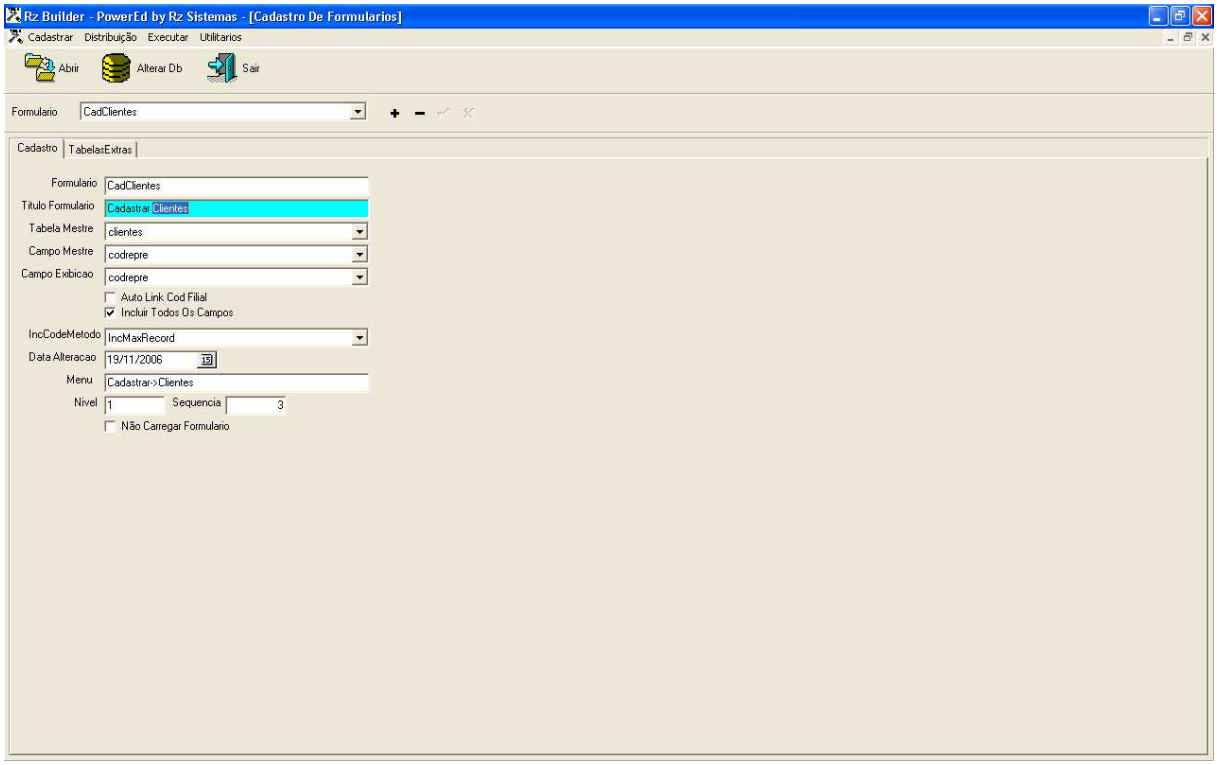

Figura 13 – Cadastrar formulários guia cadastro.

A seguir é apresentado um detalhamento da finalidade dos campos exibidos na figura 13:

a) formulário : indica o nome do formulário, neste caso foi definido o nome de cadclientes;
b) título do formulário: indica o título do formulário, neste caso foi colocado o título de Cadastro de Clientes;

c) tabela mestre: indica qual a tabela principal daquele formulário, neste caso foi definida a tabela clientes;
d) campo mestre: indica qual a chave primária daquela tabela, neste caso foi definido o campo codcli;
e) campo exibição: indica qual o campo que será exibido, neste caso foi escolhido o campo cliente;
f) auto link codfilial: indica se o sistema já deve atribuir ao campo codfilial (caso existir) na tabela com o código da filial que o usuário esta logado no momento;
g) inccodemetodo: indica o método de incremento da chave primária; estão disponíveis os sequintes métodos: incnome (não é feito incremento), incmaxrecord (pega o maior registro da tabela e incrementa 1), incmaxrecordfilial (pega o maior registro daquela tabela na filial logada no momento), para o cadastro de clientes foi definido o método incmaxrecord;
h) menu: indica em qual item do menu: o formulário deve ser incluído, neste caso foi definido o menu cadastrar->clientes;
i) nível: indica em que nível do menu a chamada ao formulário deve estar;
j) seqüência: indica dentro do nível do menu especificado qual a seqüência que o menu deve estar.

4.2.4.2 Cadastrar Formulários (guia tabela extra)

Na figura 14 encontra-se o formulário para cadastrar as guias adicionais que um formulário irá conter.

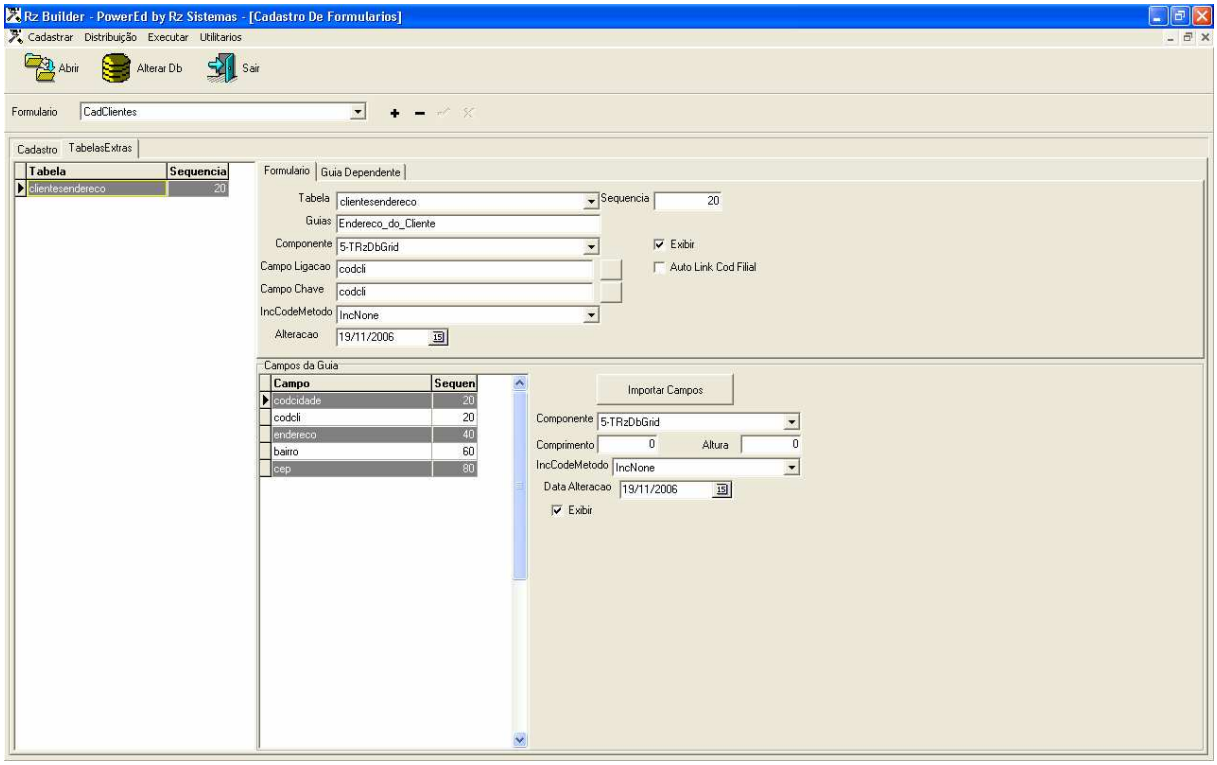

Figura 14 – Guias adicionais de um formulário.

A seguir é apresentado um detalhamento da finalidade dos campos exibidos na figura 14:

a) tabela : indica se existe mais alguma tabela vinculada com o formulário,neste caso foi vinculada a tabela clientesendereco;

b) seqüência: indica qual a seqüência que a guia vai ter no formulário, neste caso foi definida seqüência 1;

c) guia: indica o nome da guia, neste caso foi definida como sendo clientes_endereco;

d) componente: indica o componente que será utilizado para representar a guia naquele formulário, neste caso foi definido o componente TRzDbGrid;

e) campo ligação: indica qual o nome do campo que liga a tabela mestre da primeira guia com a guia secundária, neste caso foi definido o campo codcli;

f) inccodemetodo: indica o método de incremento da chave primária, Estão disponíveis os seguintes métodos: incnome (não é feito incremento), incmaxrecord (pega o maior registro da tabela e incrementa 1), incmaxrecordfilial (pega o maior registro daquela tabela na filial logada no momento);

g) campos da guia: indica quais campos serão exibidos na guia secundária.

4.2.5 Gerando o banco de dados da aplicação

Após a definição das tabelas, campos e formulários o programador poderá gerar o *script* da aplicação. Para isto deve-se ir ao botão executar alterações e pressionar o botão atualiza dados.

A figura 15 demonstra a janela que executa a alteração e a criação da base de dados no cliente.

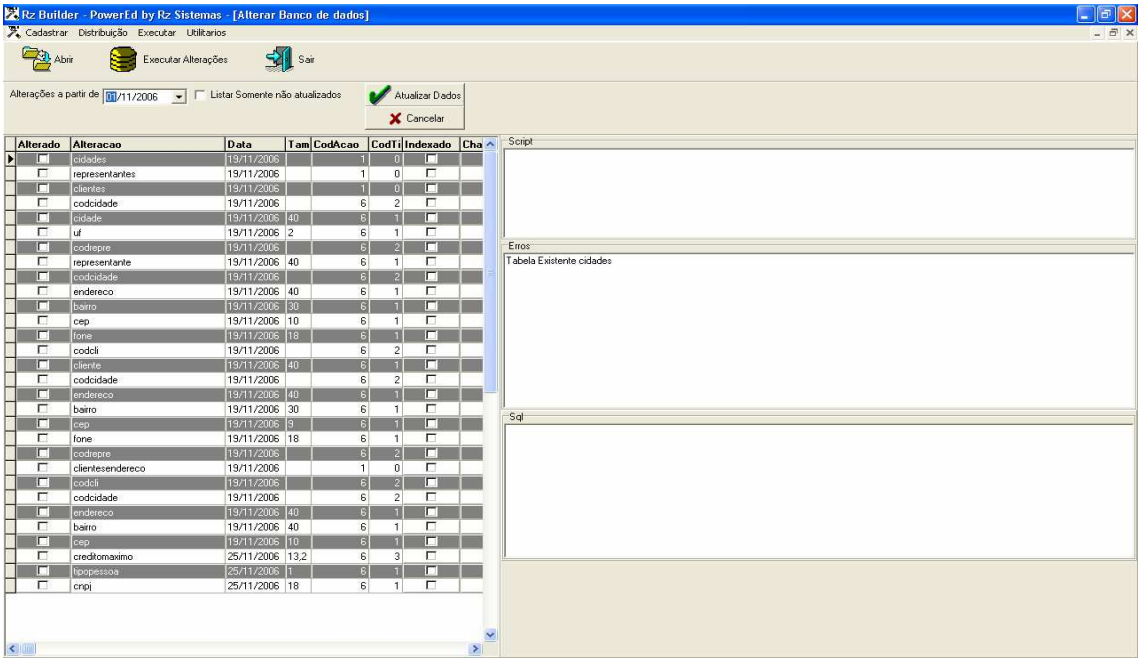

Figura 15 – Alteração e criação da base de dados.

4.2.6 Compilando a aplicação

Este trabalho não tem seu principal foco na biblioteca de componentes mais será demonstrada a praticidade que o programador terá quando criar a aplicação final.

Na Figura 16, encontra-se o formulário principal da aplicação juntamente com o componente TRzCadastroFormCreator que é responsável pela leitura do dicionário de dados e geração do menu que irá dar acesso aos formulários.

Para o funcionamento do componente basta definir qual o menu principal da aplicação e informar a base de dados que ele deverá buscar os dados.

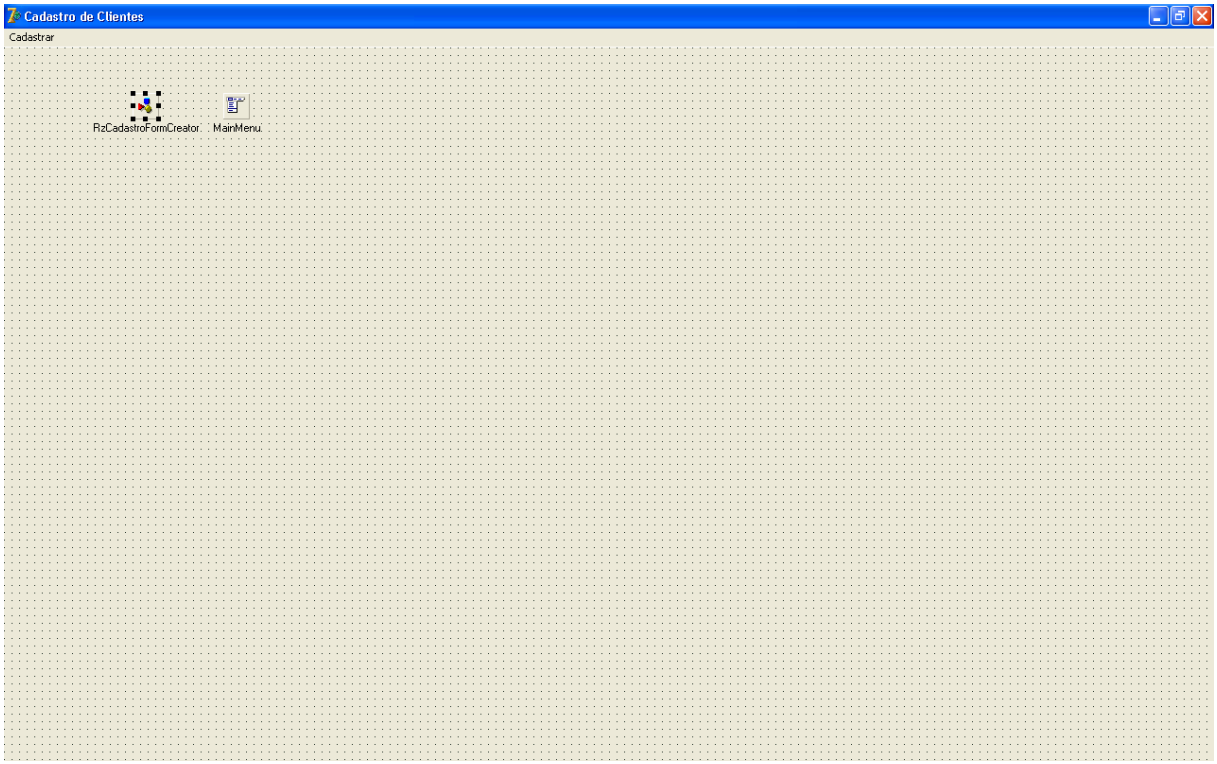

Figura 16 – Formulário principal da aplicação gerada dentro do ambiente Delphi.

4.2.7 Exportação dos dados para XML

Na figura 17, encontra-se a janela que exporta os dados da aplicação para o formato XML. Desta forma o programador poderá exportar os dados do projeto para o formato XML permitindo assim a compatibilidade com outros aplicativos.

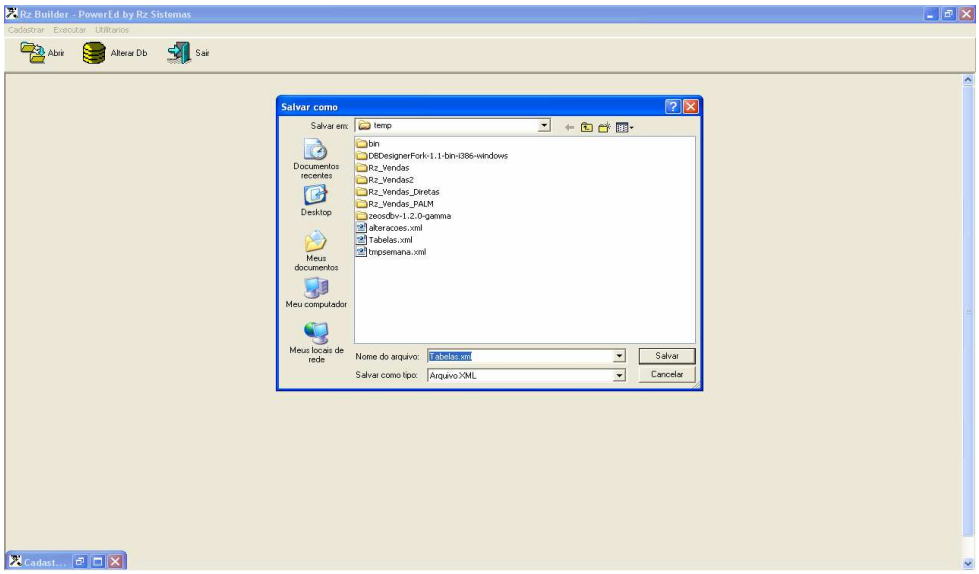

Figura 17 – Exportar dados da ferramenta para xml.

4.2.8 Executando uma manutenção no aplicativo.

Finalizando a demonstração da ferramenta será feita a manutenção da aplicação gerada com auxílio da ferramenta. Suponha que este aplicativo fosse entregue ao usuário final da aplicação e ele solicitasse o acréscimo de um campo adicional no cadastro de clientes denominado celular.

No método tradicional o programador executaria esta atualização conforme os passos descritos a seguir:

a) entraria o *MySQl Query Browser* onde está sendo criado o banco de dados e criaria o campo;
b) entraria no ambiente delphi;
c) acrescentaria o campo no dataset correspondente;
d) acrescentaria o componente responsável pela exibição da informação no formulário correspondente;
e) acrescentaria o rótulo do novo campo neste caso celular;
f) definiria a ordem de tabulação do componente no formulário;
g) definiria a máscara do campo;
h) compilaria a aplicação e executaria os testes necessários;
i) escreveria na documentação de alterações que foi criado um campo novo;
j) criaria um *script* SQL com as alterações efetuadas no banco;
k) após isto seria levado para o cliente o novo sistema e juntamente com o *script* SQL necessário para atualização;
l) colocaria o novo executável na pasta correspondente;
m) entraria no banco de dados e criaria o novo campo executando o script SQL;
n) executaria os testes para validar a aplicação no cliente.

Agora será feita a mesma manutenção utilizando a ferramenta:

a) abrir o projeto previamente salvo;
b) ir no menu cadastrar-> campos e criar o novo campo na tabela clientes, já definindo sua máscara;

c) ir no menu distribuição informar alterações de *release* descrevendo que foi criado um novo campo, conforme exibido na figura 18;

d) após isto acessar o menu distribuição-> criar arquivo de atualização deverá ser informado o caminho do executável e seu diretório de destino neste caso ApDir sendo que este parâmetro será interpretado pela ferramenta no cliente e copiando o arquivo para a pasta correspondente;

e) pressionar o botão copiar, neste momento será gerado um arquivo compactado com todas os arquivos necessários para o projeto, a Figura 19 demonstra o formulário para gerar o script de atualização;

f) após estes passos basta apenas levar o *script* para o cliente , executá-lo e a aplicação estará funcionando novamente.

Na figura 18 encontra-se o formulário para informar as alterações que foram realizadas nas versões do aplicativo, bem como suas principais alterações.

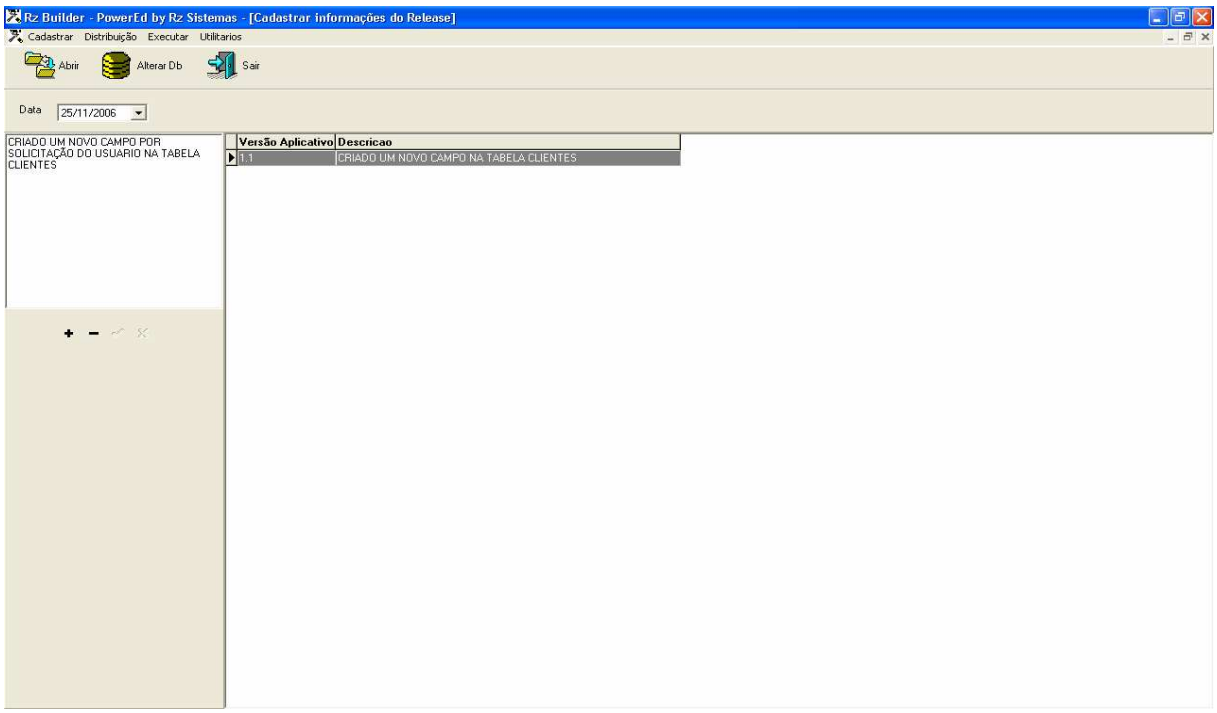

Figura 18 – Informar as alterações realizadas nas versões do aplicativo

A seguir é apresentado um detalhamento da finalidade dos campos exibidos na figura 18:

a) versão do aplicativo: indica a versão do executável que se refere a documentação;

b) descrição: informa uma descrição sintética da alteração implementada;

c) obs: descreve mais detalhadamente a alteração;

d) data: informa a data que foi realizada esta alteração.

Na figura 19 está o formulário que gera o script para atualização do sistema no usuário final.

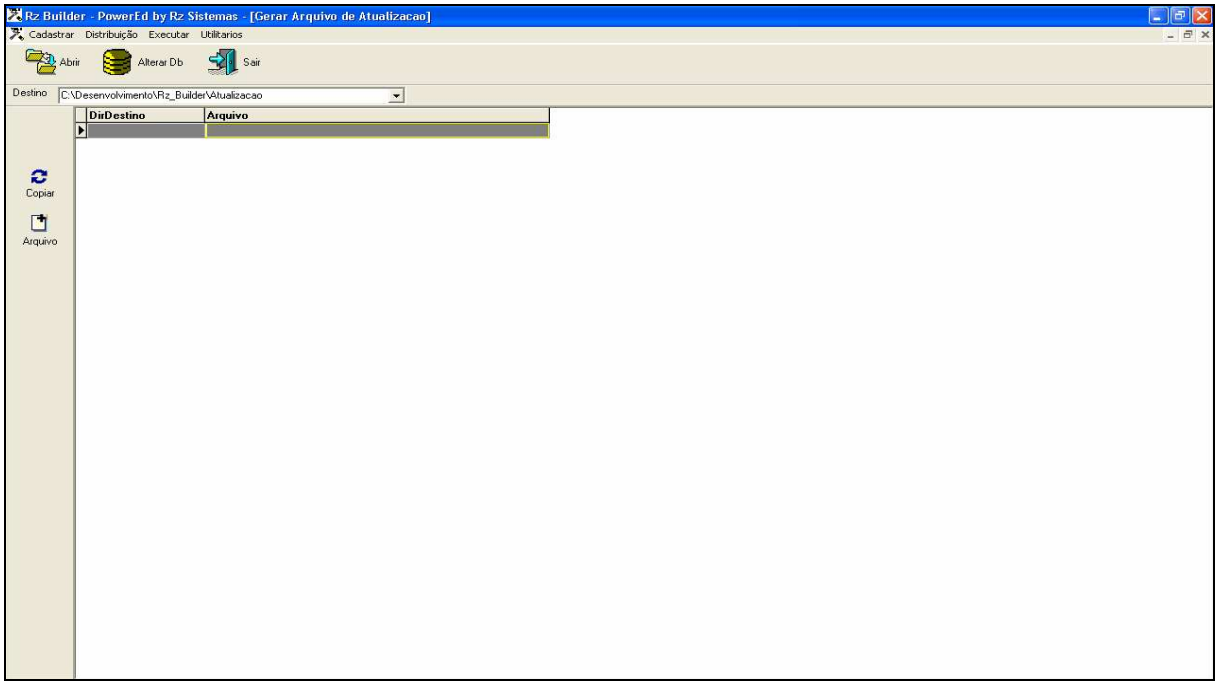

Figura 19 – Gerar script atualização

A seguir é apresentado um detalhamento da finalidade dos campos exibidos na figura 19:

a) dir destino: indica em qual diretório o sistema devera ser copiado na máquina do usuário final; estão disponíveis os seguintes diretórios apdir (diretório do executável principal da aplicação), dlldir (diretório onde ficam as dlls da aplicação) , windir (diretório do windows);
b) arquivo: indica qual o arquivo que será copiado;
c) botão copiar : gera o arquivo de atualização;
d) botão arquivo : insere um novo arquivo.

Os passos descritos anteriormente demonstram a integração e centralização que a ferramenta promove no desenvolvimento de uma aplicação, sendo que para gerar a aplicação demonstrada a seguir não foram necessárias nenhuma linha de código a não ser a utilizada para escrever os componentes.

Além disto demonstrou-se como a ferramenta reduz os passos para manutenção de um

sistema centralizando a documentação da aplicação e facilitando o trabalho do desenvolvedor e da equipe de suporte.

4.3 APLICAÇÃO GERADA

Na figura 20 encontra-se o formulário para cadastro de clientes gerado com as informações da ferramenta e interpretadas por componentes escritos em Delphi.

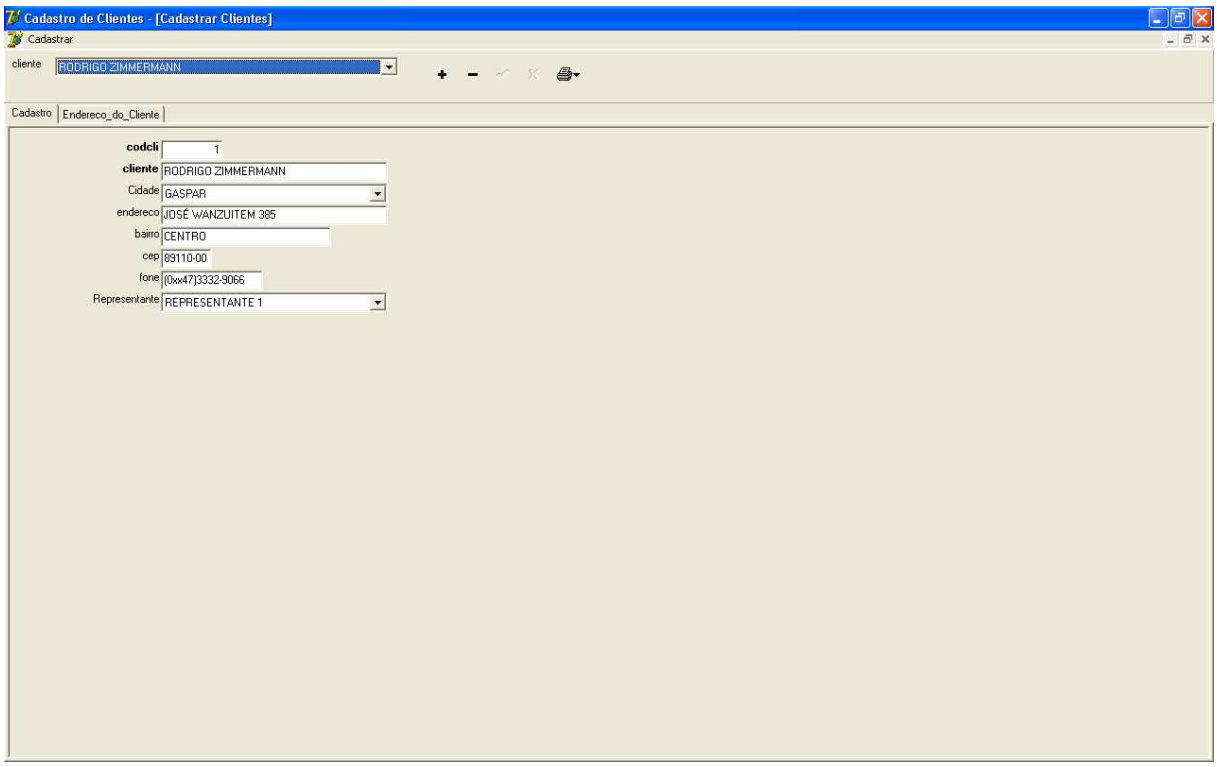

Figura 20 – Cadastro de clientes gerado pelos componentes.

Nota-se que os campos em negrito destacam informações obrigatórias e as máscaras do telefone e CEP são inseridas automaticamente devido as regras que foram inseridas na ferramenta.

O campo codcli também é incrementado automaticamente pois é fornecido no parâmetro inccodmetodo no cadastro de formulários, que está explicado com mais detalhes no item g da figura 13.

O campo representante, que tem seu nome no banco de dados como codrepre é criado já vinculado com a tabela representante. Seu rótulo é automaticamente alterado para representante pois este parâmetro é fornecido no campo legenda.

Na figura 21, é demonstrada a guia com os endereços adicionais do cliente do formulário para cadastro de clientes, juntamente com uma janela de pesquisa para localizar cidades, gerada pelos componentes escritos em Delphi que interpretam as informações geradas pela ferramenta, que neste caso indica que o campo codcidade deve buscar informações da tabela cidades.

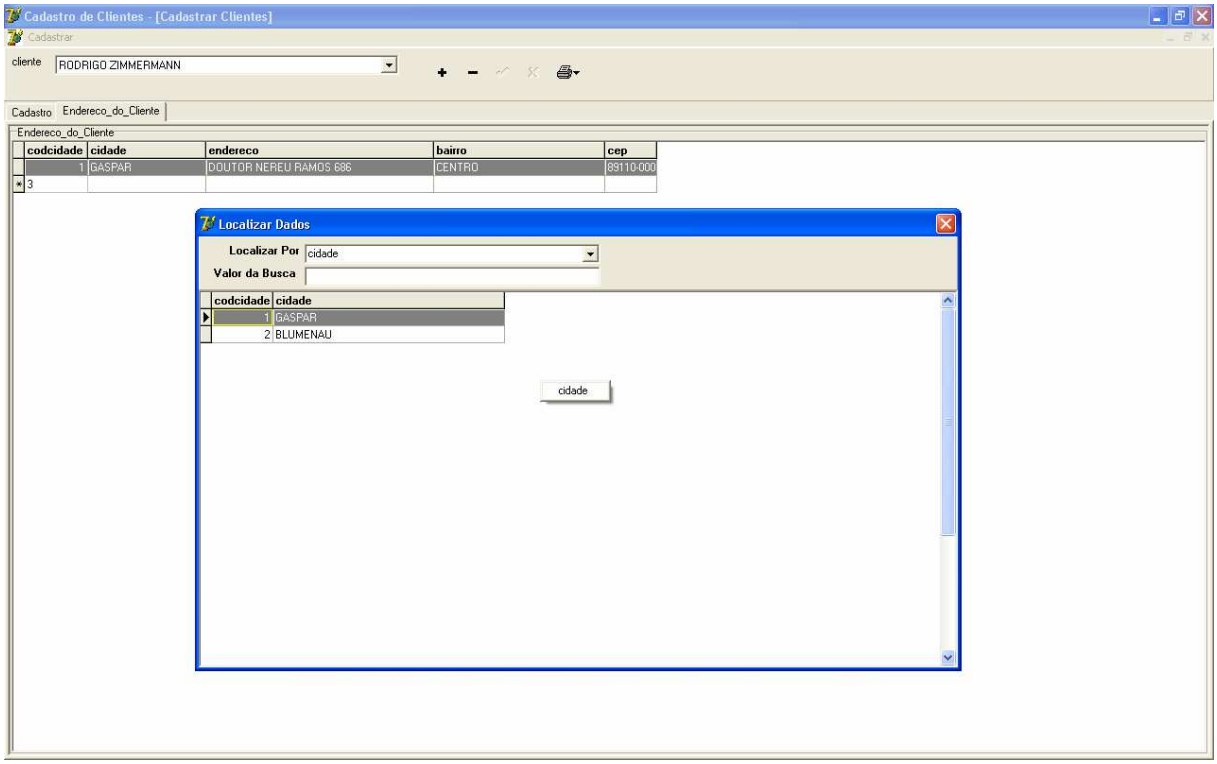

Figura 21 – Cadastro de clientes guia endereco_do_cliente.

Na figura 22, são demonstradas as tabelas criadas no *MySQl* com as informações alimentadas na ferramenta, este diagrama entidade relacionamento físico foi obtido usando a ferramenta *SQL Manager 2005 for MySQl*, usando a função de engenharia reversa disponível na mesma.

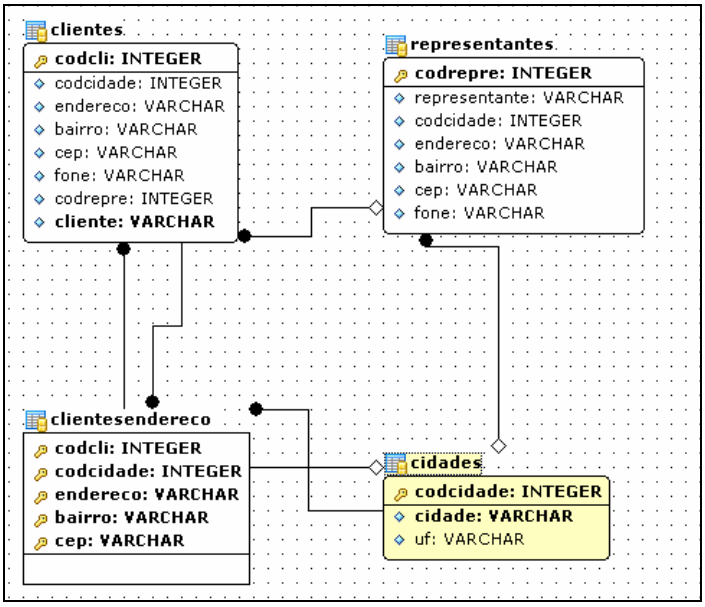

Figura 22 – Diagrama entidade relacionamento físico dos dados especificados na ferramenta.

Na Figura 23 está disposto o diagrama entidade relacionamento físico do dicionário de dados gerado pela ferramenta.

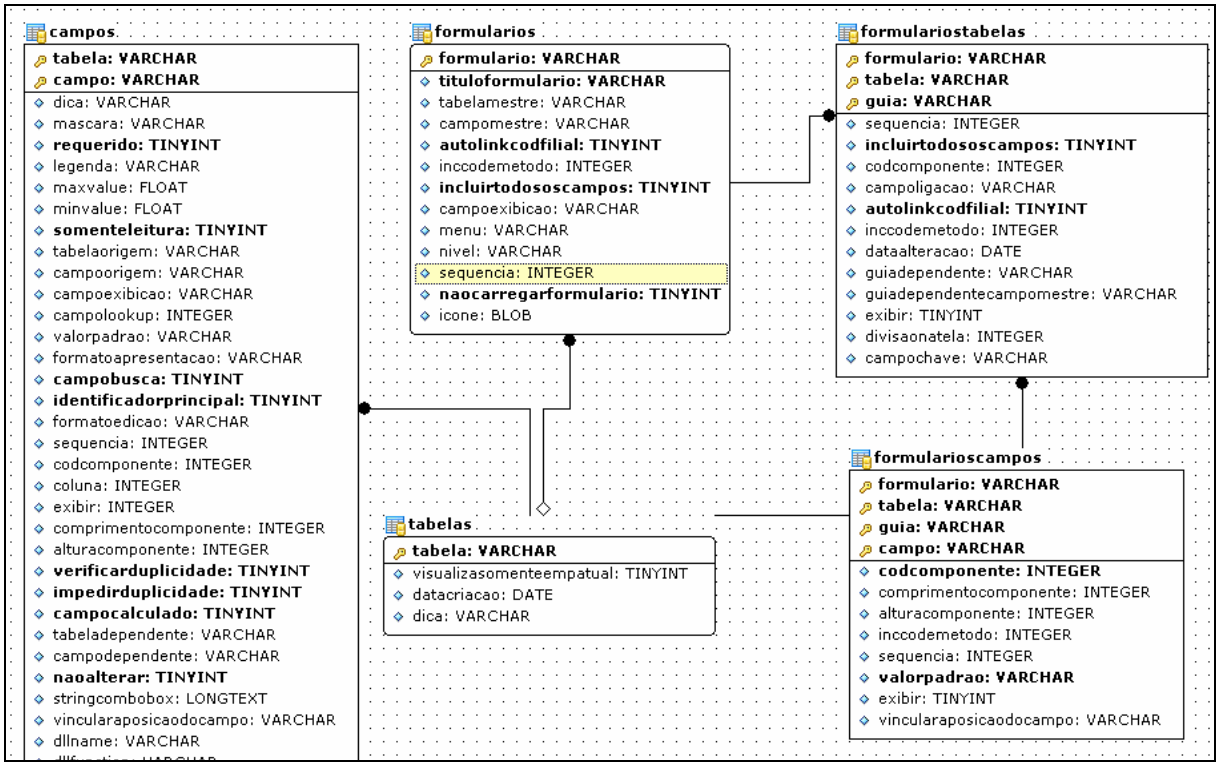

Figura 23 - Diagrama entidade relacionamento físico do dicionário de dados criado pela ferramenta.

4.4 RESULTADOS E DISCUSSÃO

Para o desenvolvimento desta ferramenta foi necessário a utilização de vários componentes de terceiros não distribuídos originalmente com o Delphi, entre eles se destacam-se:

a) backuptools componente utilizado para gerar o arquivo compactado que neste trabalho denominamos *script* de atualização do sistema, o qual contém os executáveis e o banco de dados do MsAccess;
b) ZeosDBO foi incorporado na ferramenta devido a sua grande portabilidade entre bancos distintos, possibilitando futuramente a implementação desta ferramenta para outros bancos de dados;
c) componentes da empresa Rz Sistemas, que gerenciam a informações para configuração de seus aplicativos, dentre eles o rzpathmanager que gerencia em que diretório os executáveis e arquivos do projeto devem ser copiados.

Comparando o trabalho atual com os trabalhos correlatos, ele é uma junção das funcionalidades implementadas por Werner (2004), Heiden (2005), Ferreira (2003) sendo que a seguir são apresentados os principais pontos que auxiliaram na construção deste trabalho.

Werner (2004) desenvolveu como trabalho de conclusão de curso uma ferramenta de gerenciamento do banco de dados Firebird, que tinha como objetivos permitir a manutenção da estrutura de metadados, permitir a inclusão/alteração/exclusão de dados na estrutura da base de dados, permitir ao usuário executar comandos SQL, permitir a manipulação de uma base de dados através de uma interface interativa. Como uma parte do trabalho se propõe à criação de bancos de dados este trabalho conteve algumas informações relevantes para o desenvolvimento desta ferramenta.

Heiden (2005) desenvolveu uma ferramenta de geração de código a partir de um dicionário de dados do *SQL Server* para tecnologia ASP.NET. Mesmo que a ferramenta deste trabalho não tivesse a intenção de gerar código, ela se propõe a gerar informações que alimentarão um dicionário de dados para que objetos implementados no Delphi gerem formulários de cadastros e interfaces.

Ferreira (2003), desenvolveu uma ferramenta para extração e documentação de rotinas de projetos de sistemas de informação. A ferramenta proposta por este trabalho não tem seu foco principal na documentação do código fonte da aplicação, mas uma parte deste trabalho focará a documentação das informações do banco de dados da aplicação.

O quadro comparativo 20 exibe as principais características dos trabalhos correlatos com o trabalho atual.

Funcionalidades	Atual TCC	Ferreira (2003)	Werner (2004)	Heiden (2005)
As páginas são geradas a partir de uma aplicação desktop				X
Compatível com o SGDB SQL Server 2000				X
Compatível com banco de dados *MySQl*	X			
Compatível com Microsoft Access	X			
Compatível com SGDB Firebird		X	X	
Especificar o tipo de acesso que terá cada tabela				X
Campos dinâmicos conforme tipo de dados	X			X
Validação de campos na inserção e alteração	X			X
Definir número de registro por página				X
É possível definir o modelo entidade relacionamento	X		X	
A Atualização do sistema é feita pela ferramenta	X			
A Documentação do sistema é armazenada juntamente com o dicionário de dados	X			
Ë possível criar,tabelas,campos e relacionamentos	X	X	X	
Ë possível criar formulários dinamicamente sem necessidade de recompilação	X			
É possível documentar o código fonte		X		

Quadro 20 - Funcionalidades especificas de cada trabalho.

4.4.1 Testes

Nos testes realizados com a ferramenta, a mesma mostrou-se bem estável, não apresentando problemas eminentes. O único erro aparente que a ferramenta apresenta acontece caso o *MDAC* não esteja instalado causando um erro logo na abertura da ferramenta, impedindo assim sua utilização, um dos possíveis erros que foram detectados nos testes seria caso a biblioteca lib*MySQl*41.dll não estivesse na pasta windows do computador que irá executar a aplicação, o outro erro que causaria o travamento da aplicação gerada seria caso o *MySQl* não estivesse rodando na rede no qual o sistema foi instalado.

5 CONCLUSÕES

Os objetivos propostos para este trabalho foram alcançados pois a ferramenta desenvolvida conseguiu criar tabelas, campos e relacionamentos no banco *MySQl* documentando sua finalidade dentro do aplicativo.

As informações inseridas na ferramenta foram suficientes para que os componentes escritos em Delphi conseguissem criar de maneira automatizada os formulários demonstrados na figura 20 e 21, além de definir como a chave primária foi incrementada, reduzindo a quantidade de linhas de código necessárias para gerar o sistema proposto.

As informações referentes aos *releases* e alterações do aplicativo podem ser facilmente alimentadas e consultadas conforme o formulário que está na figura 18.

Apesar da ferramenta criada atender de forma mais apropriada a realidade da empresa Rz Sistemas com poucos ajustes poderia ser estendida para outras empresas.

A ferramenta centraliza e automatiza o trabalho de desenvolvimento, suporte e manutenção de qualquer sistema de gestão, conforme Cousin e Collofello (1992) sugeriram. Além disto, a implantação desta ferramenta aliada a uma biblioteca eficiente atende os requisitos que o modelo (MPS.BR) sugere como imprescindíveis no processo de solução técnica, principalmente referente a padronização de interfaces e documentação.

Considerando a necessidade do mercado possuir ferramentas cada vez mais rápidas e eficazes para o desenvolvimento de aplicações, entende-se que este trabalho mostrou-se bastante eficiente pois automatiza e centraliza a documentação da base de dados, permite a criação e futura manutenção da aplicação, além de facilitar a padronização e diminuir o tempo para desenvolvimento e manutenção.

Visando atender os requisitos disponíveis no modelo de qualidade de software (MPS.BR) no item que se refere a solução técnica o uso de uma ferramenta que auxilie na distribuição e centralização da documentação é praticamente imprescindível para se aumentar a qualidade e eficiência de um sistema de gestão de alta confiabilidade.

Em uma *softwarehouse* que visa a produtividade e qualidade tendo seu foco em vender e prestar serviços de manutenção em sistemas de informações, o uso de uma ferramenta que centralize a criação de bases de dados, documentação auxiliando na automatização e padronização de interfaces que segundo Liesenberg (2006) são responsáveis por 50% dos recursos despendidos na fase de desenvolvimento, é fundamental para o sucesso desta empresa. Aumentando assim a qualidade em seus produtos, reduzindo custos para atender

tanto a equipe de desenvolvimento e suporte quanto os clientes finais dos *softwares* desta empresa.

5.1 EXTENSÕES

Como possíveis extensões para este trabalho destaca-se:

a) a implementação de componentes em outras linguagens que interpretem as informações inseridas na ferramenta e gerem interfaces em outros padrões;
b) a geração da base de dados e do dicionário de dados para outro banco de dados;
c) a implementação de componentes no Delphi for .NET para geração de formulários para WEB;
d) a implementação da biblioteca no próprio Delphi já que este trabalho não tratou especificamente da biblioteca.

REFERÊNCIAS BIBLIOGRÁFICAS

COUSIN L., COLLOFELLO J.S, **A task-based approach to improving the software maintenance process** - International Conference on Software Maintenance (ICSM) – IEEE Orlando, Flórida Nov., 9-12, 1992.

EMS SQL Management Studio. **Informações sobre o EMS SQL Management Studio**

FERREIRA, Lúcio Antonio Raupp **Ferramenta para extração e documentação rotinas de projetos de sistemas de informação**. 2003. 63f. Trabalho de Conclusão de Curso (Bacharelado em Ciência da Computação) - Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau.

FORWARD, Andrew**. Building and maintaining artefacts of comunication**, 2002. Tese de Mestrado, Universidade de Ottawa, Ottawa, Toronto, Canadá. Home page disponível em <http://www.site.uottawa.ca/~tcl/gradtheses/aforward/aforward_thesis.pdf> acesso em : 10/08/2006

GARLAND J. **Improved change tracking for software maintenance** - International Conference on Software Maintenance (ICSM) – IEEE Sorrento, Italy Oct., 15-17, 1991.

HEIDEN, André Cezar. **Ferramenta de geração de código a partir do dicionário de dados do Sql Server para tecnologia Asp.Net**. 2005 . 84p Trabalho de Conclusão de Curso (Bacharelado em Ciência da Computação) Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau.

LIESENBERG , Hans **Projeto e Construção de Interfaces de Usuário com Técnicas de Baixo Custo** <http://www.dcc.unicamp.br/~hans/mc750/projInt/palestra.html#biblio> acesso em : 25/12/2006

MINASI, Mard. **Segredos de Projeto de Interface Gráfica com o Usuário**. Rio de Janeiro, Infobook, 1994, 223 p.
MPS.BR. **Melhoria de processo do software brasileiro**. Softex Home Page disponível em <http://www.softex.br/media/MPSBR_em_portugues.pdf> acesso em 10/08/2006

MPS.BR. **Melhoria de processo do software brasileiro**. Softex Home Page disponível em <http://www.softex.br/media/MPSBR_em_portugues.pdf> acesso em 10/08/2006

PRESSMAN, Roger S. **Engenharia de software**. São Paulo: Makron Books, 1995.

WASSERMAN, **A. The next generation of integrated software development environments**. Anais. VIII Simpósio Brasileiro de Engenharia de Software. Curitiba, 25/10/1994 . (palestra convidada)

WERNER, Carlos Eduardo **Ferramenta de Gerenciamento para o banco de dados Firebird**. 50f. 2004. Trabalho de Conclusão de Curso (Bacharelado em Ciência da Computação) Centro de Ciências exatas e naturais, Universidade Regional de Blumenau.