Universidade Federal da Paraíba - UFPB
Centro de Ciências e Tecnologia - CCT
Departamento de Sistemas e Computação - DSC
Aluno Estagiário: Gustavo Eulalio Miranda Cabral Mat.: 29711172-3
Orientador Acadêmico: Walfredo Cirne Filho
Supervisor Técnico: Williams Alves Dantas

Relatório Final de Estágio

Biblioteca Setorial do CDSA. Maio de 2021.

Sumé - PB

Índice

1. APRESENTAÇÃO

1.1 Objetivos da Empresa

A Bytecom Sistemas Ltda. atua na informatização de indústrias/empresas do setor de alimentação. Os principais clientes da Bytecom são panificadoras, confeitarias, restaurantes, *fast-foods* e pizzarias.

1.2 Objetivos Acadêmicos

A cadeira Estágio Integrado oferece ao aluno a oportunidade de perceber como ou *se* os conhecimentos adquiridos ao longo de quatro anos de estudo podem ser aplicados no dia-a-dia das empresas, se o currículo estudado realmente tem aplicações fora da universidade.

Também é uma chance de o aluno se testar, vendo de consegue, durante o estágio, aplicar o conhecimento que tem, se consegue relacionar as situações surgidas no trabalho com os conhecimentos estudados no curso.

Por último, o aluno terá uma convivência no ambiente da empresa, tendo uma melhor visão de como a empresa funciona internamente, de quais são as pressões existentes no ambiente, quais as formalidades e as liberdades, questões de hierarquia, etc.

1.3 Constituição do Relatório

Este relatório está constituído da seguinte forma:

- Apresentação: onde fala-se dos objetivos da empresa e do objetivo acadêmico do estágio, além de apresentar a constituição do relatório;
- Ambiente do estágio: onde explica-se o ambiente de trabalho, citando aspectos positivos e negativos;
- Descrição do problema: descreve a necessidade/problema que a empresa desejava resolver, e que foi suprida(o) pelo estágio;
- Proposta de solução: como o aluno se propôs a sanar a necessidade / problema;
- Atividades desenvolvidas: descreve as atividades desenvolvidas pelo aluno, indicando métodos/técnicas utilizadas, tempos gastos e artefatos gerados em cada atividade.
- Conclusões e sugestões: comentários finais sobre o estágio e sugestões para o melhoramento da atividade.
- Anexos: Artefatos gerados durante o estágio, resumo metodológico, declaração de aprovação pela empresa, plano de estágio.

2. AMBIENTE DE ESTÁGIO

A empresa atualmente utiliza a estrutura do laboratório de prototipagem de software do Centro Softex Genesis de Campina Grande – Poligene. No total são 11 (onze) computadores Compaq Prossig 305 com processadores Pentium Celeron 466 MHz, 64 MB de memória RAM e 8.4 GB de HD, além de um Servidor Netfinity 3500 também IBM. Todos os computadores possuem acesso à Internet 24 horas por dia.

A equipe de desenvolvimento é formada por dois analistas formados e por alunos da graduação do curso de Ciência da Computação e do curso de Desenho Industrial da UFPB, campus II. Têm a supervisão de Robert Kalley Menezes, coordenador do Centro Softex Genesis de Campina Grande - Poligene, e professor do Departamento de Sistemas e Computação - DSC.

Aspectos Positivos:

- Ambiente descontraído e informal;
- Acesso irrestrito à internet, e conseguinte vasto acesso a informações;
- Proximidade à universidade.

Aspectos Negativos:

- Ausência dos superiores;
- Falta de metodologia de desenvolvimento na empresa;
- Curto tempo para a conclusão do estágio.

3. DESCRIÇÃO DO PROBLEMA

A empresa utiliza o InstallShield Express para instalar o seu produto Panificador. O InstallShield realiza uma boa instalação, mas permite pouca personalização do processo, além disso deixa a desejar na instalação do banco de dados, cuja instalação nas máquinas-cliente requer configuração especial.

O Panificador é uma aplicação que trabalha em rede, numa arquitetura *client/server*. Deve ser instalado em várias máquinas (tipicamente, os caixas da panificadora), com um banco de dados centralizado, a ser instalado no servidor.

O instalador deve copiar o banco de dados para o servidor, e configurar o SGBD de forma que o BD fique visível à aplicação. Também deve instalar a aplicação nas máquinas da rede selecionadas pelo cliente e configurá-las para acessar o BD no servidor (e isto o InstallShield não está fazendo).

4. PROPOSTA DE SOLUÇÃO

Modelar e implementar um sistema para realizar a instalação de forma mais otimizada possível, tendo em vista a estrutura da aplicação (*client/server*).

O instalador deve dar flexibilidade suficiente à empresa para permiti-la mostrar seu logotipo e outras imagens durante a instalação do produto, além de escolher que arquivos serão instalados, e em que máquinas do cliente (nas máquinas *client* ou na *server*).

Também deve ser fácil e claro de usar o suficiente para que o cliente possa utilizá-lo com autonomia, sem a ajuda de um técnico da empresa.

Pelo menos dois programas serão gerados nesse projeto, o configurador de instalação, e o instalador em si. O configurador será usado na empresa, para definir os arquivos, as figuras, textos, etc. da instalação. O instalador será rodado nas máquinas do cliente, para copiar os arquivos e fazer as configurações devidas.

O sistema será implementado com o Delphi 5.

5. ATIVIDADES DESENVOLVIDAS

O plano do estágio prevê um total de três iterações. Um cronograma inicial foi estipulado da seguinte forma.

SEG	TER	QUA	QUI	SEX
				[3/8] - Definição de um cronograma inicial. (este cronograma)
[6/8] 1ª iteração: - Conversa com usuários do sistema. (6h) - Definição de atividades mais importantes. (2h)	[7/8] - Definição de um cronograma mais preciso da primeira iteração. (4h)	[8/8] - Análise referente à primeira iteração. (6h)	[9/8] - Estudo da literatura referente aos problemas encontrados. (5h)	[10/8] - Estudo da literatura referente aos problemas encontrados. (7h)
[13/8] - Implementação e testes (8h)	[14/8] - Implementação e testes. (4h)	[15/8] - Implementação e testes. (6h)	[16/8] - Implementação e testes. (5h)	[17/8] - Implementação e testes. (7h)
[20/8] - Implementação e testes. (8h)	[21/8] - Implementação e testes. (4h)	[22/8] - Implementação e testes. (6h)	[23/8] - Implementação e testes. (5h)	[24/8] Final de 1ª iteração: - Implementação e testes. (7h)
[27/8] 2ª iteração	A segunda iteração não foi planejada em mais detalhes, apenas o tempo foi estipulado.			[14/9] Final da 2ª iteração
[17/9] 3ª iteração	A terceira iteração não foi planejada em mais detalhes, apenas o tempo foi estipulado.			[5/10] Final da 3ª iteração

Uma das pretensões do sistema era poder fazer a instalação em várias máquinas a partir de uma máquina, inclusive alterando o registro do Windows em cada uma das máquinas. O orientador acadêmico aconselhou que, em primeiro lugar, deveria ser feita uma pesquisa sobre a viabilidade disso, e, sendo viável, como fazê-lo. Quanto a gravar arquivos nas máquinas remotas, era possível desde que, no computador do qual se

desejava fazer a instalação, houvesse unidades de rede mapeadas para os computadores remotos. Além disso, o usuário deveria ter permissão de escrita em cada máquina remota. Via de regra, os técnicos da Bytecom encontram as redes configuradas corretamente neste sentido. Contudo, havendo alguma falha nesta configuração, ajustes manuais precisarão ser feitos.

Duas opções surgem daí: considerar que a rede sempre estará bem configurada, e fazer com que o programa detecte esses problemas e realize os ajustes necessários automaticamente.

A primeira opção, mais simples, funcionaria na maioria das vezes (segundo a experiência da empresa), mas, nas situações em que não funcionasse, poderia acarretar prejuízos à imagem da Bytecom.

A segunda opção, fazer com que o programa detecte estes problemas e faça os ajustes automaticamente adicionaria uma complexidade ao projeto que o tornaria impossível de ser realizado por uma pessoa no espaço de dois meses da duração deste estágio.

Quanto à alteração remota do registro, isso requeria uma pesquisa mais detalhada. Achei que não poderia fazer uma análise precisa sem antes saber se isso era possível. Passei então uma semana pesquisando sobre o registro do Windows.

Pesquisa sobre o registro:

Iniciei minha pesquisa em sites de foruns on-line, como o `http://inforum.insite.com.br/202/`, o `http://www.certto.com.br/forum/` e o `http://www.qualyinf.com.br/forum/`, onde recebi indicações para procurar sobre arquivos do tipo MSI (Windows Installer). Os arquivos MSI indicam ao Windows Installer (uma nova tecnologia da Microsoft, para facilitar as instalações e desinstalações no Windows) como instalar os arquivos necessários, assim como fazer as devidas modificações no registro.

Parecia promissor, no entanto, segundo o texto da página `https://iats.missouri.edu/servlets/knowledgebase/article/25822`, o Windows Installer é uma tecnologia relativamente nova e que ainda está sendo atualizada e expandida. Diz que as versões mais antigas podem acusar corrupção/conflitos nas entradas do registro. Além disso, algumas versões mais antigas do Windows 98 não vêm com o Windows Installer instalado. O usuário teria que baixar um *patch* para atualizar o Windows, de forma a usar esta tecnologia.

Como a idéia era fazer o processo todo de forma automática, sempre a partir de uma máquina para as demais, essa opção foi descartada, pois se o Windows Installer não estivesse presente em cada máquina, teríamos que instalá-lo manualmente.

Outra opção seriam os arquivos .REG, que contém instruções para modificações no registro. No entanto, esses arquivos teriam que ser executados nas diversas máquinas, o que iria de encontro ao nosso objetivo.

Através dos mesmos sites de foruns on-line, descobri que talvez fosse possível mexer no registro de outra máquina através do RegEdit (programa do Windows), e também que com o componente TRegistry do Delphi, eu poderia fazer as mesmas coisas que com o RegEdit. No entanto, isso só poderia ser feito com a ajuda de outro programa, que deveria estar presente nas máquinas remotas, e que permitiria que outra máquina alterasse o seu registro através da rede. Fiz alguns testes nas máquinas do laboratório, esperando que o software estivesse instalado por default, mas não obtive sucesso.

Outro site visitado foi o do Windows Guides Network, `http://www.winguides.com`, que fornece dicas sobre o registro do Windows. Algumas dessas dicas podem ser usadas para deixar o programa mais profissional.

Em outro site, o `http://www.easytweak.com/`, encontrei o 'Mystery Behind the Windows Registry', um arquivo no formato HTML Help File (CHM), que explica, em detalhes, a estrutura do registro do Windows.

Conclusão: Nenhuma das três alternativas encontradas foi capaz de resolver o problema de modificação de registros remotamente. Sem um *patch* ou um programa auxiliar nas máquinas remotas, não há como realizar as modificações no registro remotamente de maneira simples. No entanto, recolhi algumas informações úteis sobre o registro que poderei utilizar no futuro.

Decidi fazer a instalação da forma convencional, fazendo com que o usuário tenha de realizar a instalação em cada máquina. Entretanto, permanecia como requisito do programa realizar a configuração da aplicação com relação ao banco de dados.

Na semana seguinte, me foi pedido para documentar o código de forma que pudesse usar alguma ferramenta do tipo JavaDoc para gerar documentação em HTML.

Pesquisa sobre a ferramenta de documentação:

Eu já tinha conhecimento de uma ferramenta que poderia gerar os arquivos de documentação, o Time2Help. No entanto, nunca o tinha testado. Além do que era shareware e limitava o uso a 30 dias. Decidi então procurar alternativas.

Uma das alternativas era o DDoc, obtido no site `http://www.geocities.com/falcorus/ddoc.html`. Entretanto, a operação do programa era um pouco complicada, além do que o programa tinha alguns *bugs*, e os arquivos HTML gerados não tinham uma boa apresentação.

Entre as outras alternativas encontradas, achei alguns projetos comerciais (pagos), outros shareware, alguns que geravam estatísticas sobre o código (descartei-os devido ao cronograma, pois não haveria tempo para aprender a operá-los), e apenas um software freeware, o DelphiDoc. Este último foi o produto escolhido, tem alguns *bugs*, mas é de fácil uso e realiza a tarefa com eficiência suficiente para os nossos propósitos. Foi encontrado no site `http://www.softconsult.se/ delphidoc.html`.

Até então, o tempo tinha sido ocupado assim:

SEG	TER	QUA	QUI	SEX
[6/8]	Pesquisa sobre o registro do Windows (30h)			[10/8]
[13/8] - Pesquisa sobre ferramentas de documentação no Delphi. (8h)	[14/8] - Pesquisa sobre ferr. de doc. no Delphi. (4h)	[15/8] - Pesquisa sobre ferr. de doc. no Delphi. (4h) - Novo cronograma		

Novo cronograma:

Com quase duas semanas de atraso, um novo cronograma era necessário. Este ficou da seguinte forma:

SEG	TER	QUA	QUI	SEX
			[16/8] - Análise (5h)	[17/8] - Análise (4h) - Pesquisa sobre componentes ZIP (3h)
[20/8] - Pesquisa sobre componentes ZIP (8h)	[21/8] - Implementação e testes (4h)	[22/8] - Implementação e testes (6h)	[23/8] - Implementação e testes (5h)	[24/8] - Implementação e testes (7h)
[27/8] - Implementação e testes (8h)	[28/8] - Implementação e testes (4h)	[29/8] - Implementação e testes (6h)	[30/8] - Implementação e testes (5h)	[31/8] - Implementação e testes (7h)
[3/9] - Implementação e testes (8h)	[4/9] - Implementação e testes (4h)	[5/9] 2ª iteração - Atividades não definidas		
[18/9] 2ª iteração - Atividades não definidas		[19/9] 3ª iteração - Atividades não definidas		
[2/10] 3ª iteração - Atividades não definidas				

Pesquisa sobre componentes ZIP:

Entre os produtos encontrados, muitos eram programas de compressão de arquivos (como o gzip), não componentes. Era necessário um componente compatível com o Delphi. As soluções mais adequadas encontradas eram derivadas ou utilizavam o Info-Zip.

A solução escolhida foi o DelphiZip, utilizando o componente VCL ZipMaster e os DLLs do InfoZip (Zipdll.dll e Unzdll.dll). Este componente permite que se crie arquivos comprimidos no formato .ZIP com muita facilidade. Também permite a descompressão dos arquivos e a criação de arquivos auto-extraíveis. O DelphiZip pode ser encontrado na seguinte página: `http://www.geocities.com/SiliconValley/Network/2114/index.html`.

Artefatos gerados até então:

Na análise da primeira iteração, foram gerados os seguintes artefatos:

- Requisitos não-funcionais (Anexo IV)
- Requisitos funcionais (Anexo V)
- Diagrama do modelo arquitetural (Anexo VI)
- Diagramas de use-cases do Configurador de Instalação (Anexo VII)

- Diagramas de use-cases do Instalador (Anexo VIII)
- Diagrama de colaboração do Configurador de Instalação (Anexo IX)
- Diagrama de colaboração do Instalador (Anexo X)
- Diagrama de seqüência do Configurador de Instalação (Anexo XI)
- Diagrama de seqüência do Instalador (Anexo XII)
- Diagrama de classes do módulo de Configuração do Instalador (Anexo XIII)
- Diagrama de classes do módulo do Instalador (Anexo XIV)

O módulo de configuração funciona da seguinte forma: o programa principal passa todos os parâmetros necessários, como diretório do instalador do SGBD, arquivos a compactar (que o módulo repassa direto para o TZipMaster), e o diretório destino, que é onde os arquivos serão gravados. Depois de passar os parâmetros, o programa principal envia uma mensagem para o módulo realizar o processamento, e daí ele copia o instalador do SGBD, compacta os arquivos especificados, copia o instalador da aplicação para o diretório destino e cria um arquivo de guia de instalação, que o módulo de instalação usará para se guiar.

O módulo do instalador funciona similarmente. O programa principal cria o módulo de instalação, que lê o guia de instalação para saber o que vai ou não instalar. De acordo com as configurações do guia, pede os parâmetros necessários ao usuário -- diretórios de origem e de destino do BD e da aplicação e o diretório do instalador do SGBD -- e os vai passando para o módulo de instalação. Logo após envia comando para o módulo realizar a instalação, quando este descompacta os arquivos para os diretórios devidos e executa o instalador do SGBD.

Implementação:

Com a análise em mãos, parti para a implementação.

O primeiro obstáculo encontrado foi quanto a um componente visual que mostrasse a hierarquia de pastas do Windows (como o painel esquerdo do Windows Explorer). A intenção era que o usuário, quando procurando um diretório ou um arquivo, pudesse 'passear' livremente pelo computador, de modo similar ao Explorer. No entanto, o Delphi 5 não oferece um tal componente, apenas oferece uma combobox de drives e listboxes de diretórios e de arquivos, mas nada semelhante ao painel esquerdo do explorer.

Eu teria de fazer um componente para suprir essa necessidade ou ir procurar um já feito. As buscas eram muito infrutíferas, traziam sempre componentes semelhantes aos que já havia no Delphi, ou caixas de diálogo, o que não me interessava, ou eram componentes pagos em versão de avaliação e com limitações. Durante as buscas, encontrei alguns sites com dicas de como listas as unidades de rede, como ler os nomes das pastas do Windows, etc. Decidi então tentar implementar um

componente próprio, que permitisse a navegação pelas pastas. Passei cerca de 30 horas pesquisando sobre o assunto e fazendo testes, mas não obtive sucesso, o processo envolvia mexer com a API do Windows, e havia pouca documentação sobre o assunto no Delphi.

Voltei a procurar por componentes prontos. Os resultados das buscas eram ainda similares aos da busca anterior, os componentes sempre eram semelhantes aos que já havia no Delphi, ou eram caixas de diálogo, como o TdfsBrowseDirectoryDlg, que encontrei no site `http://www.1delphistreet.com/`, e outros pagos, como o TDirectoryTreeView, encontrado no site `http://www.delphi32.com/`.

Finalmente achei o TShellTree, que oferecia exatamente o que eu precisava e era oferecido como freeware para projetos não comerciais. Como já tinha gastado muito tempo nessa fase de procura, optei por usá-lo no projeto.

Por volta do dia 10 de setembro, problemas de ordem pessoal atrasaram o trabalho. Uma lente dos meus óculos quebrou, e tive que passar mais de uma semana com os óculos quebrados, o que diminuiu a minha produtividade.

2ª iteração:

Nesta data, eu já tinha começado a planejar a segunda iteração. Nela, eu iria fazer uma interface gráfica mais bem planejada para o projeto.

Como estavam antes, o configurador e o instalador chamava uma nova janela para cada operação. O que fiz, para deixar uma interface mais agradável, foi criar uma única janela e mudar o conteúdo da janela dependendo do estágio de configuração/instalação em que o usuário estivesse.

Delphi é uma linguagem que tem suporte a eventos. Usei esta característica para realizar a mudança de interface de forma mais transparente. As alterações na janela são feitas no evento de mudança de estágio, dependendo do valor que a variável recebe, alguns componentes são mostrados/escondidos.

Além disso, aproveitei o mesmo efeito para realizar o processamento dos dados, sempre que se passa de um estágio a outro mais avançado, os dados daquela tela são armazenados num objeto -- TConfiguracao para o módulo de configuração e TInstalador para o módulo de instalação. Similarmente, quando se passa de um estágio para outro anterior, os dados são recuperados do objetos e mostrados na tela. Diagramas de classes da interface do instalador e do configurador estão nos anexos XV e XVI.

3ª iteração:

Na terceira iteração, eu precisava fazer o instalador oferecer várias opções de instalação ao usuário. O configurador precisaria preparar cada opção.

As operações, nessa fase são basicamente sobre o arquivo de guia. Enquanto anteriormente o arquivo de guia era único, nesta fase decidi dividi-lo em arquivo de guia mestre e arquivos de guia auxiliares. O arquivo de guia mestre diz quais os tipos instalações, e os arquivos de guia auxiliares que guardam as configurações de cada tipo de instalação.

O módulo de configuração terá que gerar o arquivo mestre e os auxiliares. Já o módulo de instalação deverá ler o arquivo mestre e mostrar as opções ao usuário. Dependendo da escolha deste, o programa procura o arquivo auxiliar correspondente e o lê.

Esta fase requer algumas mudanças na parte de gerenciamento de interfaces, mas apenas para inserir uma nova tela, a de escolha de tipos de instalação.

Uma tabela com os comandos usados nos arquivos de guia está no anexo XVII.

Outros artefatos gerados:

Os códigos dos testes das classes TConfiguracao e TGuia (do instalador) estão respectivamente nos anexos XVIII e XIX.

As documentações da API dessas duas classes estão nos anexos XX e XXI respectivamente. Além da documentação da classe TInstalador, no anexo XXII.

6. CONCLUSÕES E SUGESTÕES

Neste projeto, me senti muito a vontade de trabalhar usando o paradigma orientado a objetos. O fato de já ter trabalhado com Java me ajudaram ter mais facilidade para trabalhar neste paradigma, mas o Delphi não é a melhor ferramenta para isto. A estrutura da linguagem deixa o programador mal acostumado, com tendências a 'facilitar' o trabalho de outras formas.

Tive dificuldades em fazer uma análise independente da tecnologia, como descrevi no início do relatório, quando tive que pesquisar sobre componentes de compressão de arquivos antes de fazer a análise. Acho que isso deve ser mais trabalhado com os alunos.

Acho, também, que a cobrança de um relatório parcial é interessante pois ajuda muito na elaboração do relatório final.

Por fim, aconselho muito que os alunos façam um estágio. É uma experiência marcante, em que o aluno tem muito a ganhar.

7. Anexos

Anexo I

PLANO DE ESTÁGIO

PLANO DE ESTÁGIO INTEGRADO

1. Ambiente do Estágio

1.1 Dados da Empresa

Razão Social: ByteCom Sistemas Ltda.

CNPJ: 03.688.196/0001-47

Endereço: Rua Treze de Maio 294 sala 301 – Centro, Campina Grande-PB

Tele/Fax: (83)310-1438

E-mail: bytecom@uol.com.br

URL: http://www.bytecom.com.br

1.2 Nicho de Mercado

A ByteCom Sistemas é uma empresa encubada no CENTRO SOFTEX GENESIS de Campina Grande – POLIGENE que atua na informatização de indústrias/empresas do setor de alimentação. Os principais clientes da ByteCom Sistemas são panificadoras, confeitarias, restaurantes, *fast foods* e pizzarias.

1.3 Estado da informática na empresa

A empresa atualmente utiliza a estrutura do laboratório de prototipagem de software do Centro Softex Genesis de Campina Grande – Poligene. No total são 11 (onze) computadores Compaq Prossig 305 com processadores Pentium Celeron 466 MHz, 64 MB de memória RAM e 8.4 GB de HD, além de um Servidor Netfinity 3500 também IBM. Todos os computadores possuem acesso à Internet 24 horas por dia.

A equipe de desenvolvimento é formada por dois analistas formados e por alunos da graduação do curso de Ciência da Computação e do curso de Desenho Industrial da UFPB, campus II. Tem a supervisão de Robert Kalley Menezes, coordenador do Centro Softex Genesis de Campina Grande - Poligene, e professor do Departamento de Sistemas e Computação - DSC.

1.4 Dados do Estagiário

Nome: Gustavo Eulalio Miranda Cabral
Matrícula: 29711172-3
Endereço: R. Dr. Fco Lima Neto, 86, Bodocongó – Campina Grande – PB
Telefone: (0xx83) 333-1114
E-mail: gugaec@cgnet.com.br

2. Supervisão

2.1 Supervisor Técnico:

Nome: Williams Alves Dantas
E-mail: williamsad@zipmail.com.br

2.2 Supervisor Acadêmico:

Nome: Walfredo Cirne Filho
E-mail: walfredo@dsc.ufpb.br

3. Problema Objeto do Estágio

A empresa utiliza o InstallShield Express para instalar o seu produto Panificador. O InstallShield realiza uma boa instalação, mas permite pouca personalização do processo, além disso deixa a desejar na instalação do banco de dados, cuja instalação nas máquinas-cliente requer configuração especial.

O Panificador é uma aplicação que trabalha em rede, numa arquitetura *client/server*. Deve ser instalado em várias máquinas (tipicamente, os caixas da panificadora), com um banco de dados centralizado, a ser instalado no servidor.

O instalador deve copiar o banco de dados para o servidor, e configurar o SGBD de forma que o BD fique visível à aplicação. Também deve instalar

a aplicação nas máquinas da rede selecionadas pelo cliente e configurá-las para acessar o BD no servidor (e isto o InstallShield não está fazendo).

4. Proposta de Solução

Modelar e implementar um sistema para realizar a instalação de forma mais otimizada possível, tendo em vista a estrutura da aplicação (*client/server*).

O instalador deve dar flexibilidade suficiente à empresa para permiti-la mostrar seu logotipo e outras imagens durante a instalação do produto, além de escolher que arquivos serão instalados, e em que máquinas do cliente (nas máquinas *client* ou na *server*).

Também deve ser fácil e claro de usar o suficiente para que o cliente possa utilizá-lo com autonomia, sem a ajuda de um técnico da empresa.

Pelo menos dois programas serão gerados nesse projeto, o configurador de instalação, e o instalador em si. O configurador será usado na empresa, para definir os arquivos, as figuras, textos, etc. da instalação. O instalador será rodado nas máquinas do cliente, para copiar os arquivos e fazer as configurações devidas.

O sistema será implementado com o Delphi 5.

5. Atividades a serem desenvolvidas

5.1 Atividades

O projeto será realizado seguindo-se um desenvolvimento iterativo e incremental. Ao fim cada iteração do desenvolvimento intenta-se ter um sistema totalmente funcional com relação às características previstas para aquela iteração. Somente numa iteração final (ou numa iteração avançada) o sistema estará pronto para ser usado nos clientes.

Uma estimativa das atividades está descrita abaixo, e estas são realizadas a cada iteração.

[Obs.: 'Cliente', no contexto abaixo, indica os usuários do instalador, i.e., a empresa e os usuários do Panificador.]

1- Conversa com os clientes e (elaboração/revisão da) análise do sistema - 20h

2- Definição das atividades mais importantes, divisão das atividades por iterações e elaboração de um cronograma mais detalhado, ou revisão desta atividade - 5h

3- Estudo de literatura referente a problemas de implementação - 20h

4- Implementação e testes - 60h

5.2 Cronograma de atividades

ATIVIDADE Nº	DURAÇÃO EM HORAS											
	1ª iteração				2ª iteração				3ª iteração			
	20	5	20	60	20	5	20	60	20	5	20	60
1												
2												
3												
4												

Campina Grande, 3 de Agosto de 2001

Anexo II

DECLARAÇÃO

Declaramos, para os devidos fins, que o aluno Gustavo Eulalio Miranda Cabral, matrícula 29711172-3, do curso de Ciência da Computação da Universidade Federal da Paraíba, Campus II, Campina Grande, estagiou nesta empresa de 1° de agosto a 5 de outubro de 2001, e concluiu o seu estágio de forma satisfatória para a empresa.

Campina Grande, ___ de outubro de 2001

_____

Rodrigo Figueiredo de Albuquerque

Diretor, Bytecom Sistemas Ltda.

Anexo III

RESUMO METODOLÓGICO

O projeto foi realizado seguindo-se um desenvolvimento iterativo e incremental. Ao fim cada iteração do desenvolvimento tem-se um sistema totalmente funcional com relação às características previstas para aquela iteração. Somente numa iteração final (ou numa iteração avançada) o sistema estará pronto para ser usado nos clientes.

A cada iteração, as seguintes atividades ocorrem: obtenção de requisitos, análise do sistema, estudo de literatura necessária à implementação e implementação e testes.

Anexo IV
REQUISITOS NÃO-FUNCIONAIS

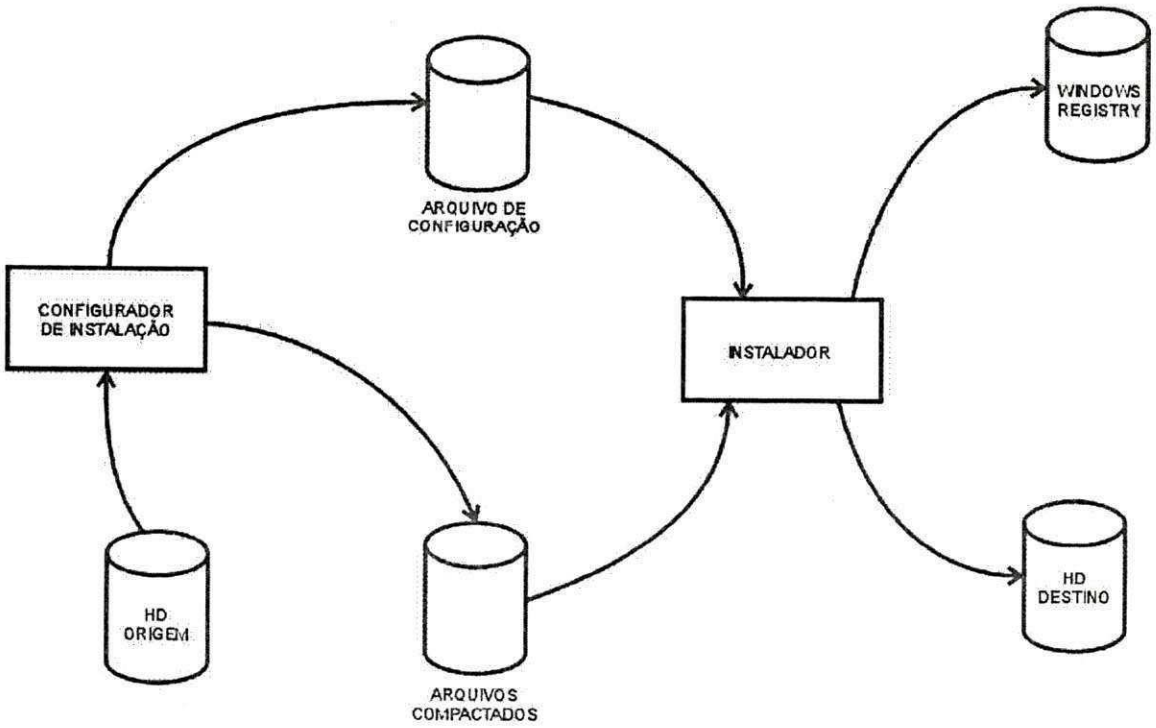
REQUISITO	DETALHES
FACILIDADE DE USO	Permitir o uso de mouse em operações de botões, escolha de diretórios e arquivos. Permitir o uso de teclado em todas as operações.
TIPO DE INTERFACE	Interface gráfica baseada em janelas. Uso do produto através de mouse e teclado.
USUÁRIO	O produto pode ser usado por leigos em operação com computadores.
VOLUME DE UTILIZAÇÃO	1 pessoa
HARDWARE E SOFTWARE	Servidor: IMB PC (ou compatível), com Windows NT/2000 e placa de rede Cliente: IMB PC (ou compatível), com Windows 9x/Me/NT/2000 e placa de rede (conectando-o ao servidor)
QUALIDADE / ROBUSTEZ	Cuidado para não apagar BD já existente.
TOLERÂNCIA A FALHA	Nenhum
DESEMPENHO	Nenhum
SEGURANÇA	Nenhum
COMPATIBILIDADE COM OUTROS PRODUTOS / VERSÕES	Verificar se pode sobrescrever o BD.
INTERNACIONALIZAÇÃO	Não há necessidade
CUSTOMIZABILIDADE	No módulo de *Configuração do Instalador*, o usuário poderá definir que tipos de instalação estarão disponíveis ao usuário. Adicionalmente, poderá também definir figuras e textos a serem mostrados durante a instalação. No módulo de *Instalação*, o usuário poderá escolher o tipo de instalação, assim como diretórios em que instalará o produto. A instalação do BD, no entanto, só poderá ser feita no disco principal.
DOCUMENTAÇÃO	Arquivo DOC (MS Word)
USO DE PADRÕES	Arquivos BMP para as imagens.
ASPECTOS LEGAIS	Pode haver complicações no uso dos compactadores de arquivos mais usados no mercado.
INTEGRAÇÃO COM OUTROS PRODUTOS	Instalador do Interbase.
INSTALAÇÃO	Descompactar arquivos no HD.
RISCOS ACEITÁVEIS	Não conseguir alterar o registro das máquinas cliente a partir do servidor automaticamente, sem outro programa instalado nas máquinas remotas. Neste caso, fazer a instalação máquina a máquina.

Anexo V
REQUISITOS FUNCIONAIS

- Módulo de Configuração da instalação:
  - Permitir escolha de arquivos a instalar (tanto arquivos de BD como da aplicação;
  - Compactar arquivos selecionados;
  - Opção "Instalar SGBD?";
  - Copiar instalador do SGBD, se for o caso;
  - Gravar arquivo de configuração da instalação.
- Módulo de Instalação:
  - Ler arquivo de configuração de instalação;
  - Permitir escolher diretórios de instalação dos arquivos da aplicação e do BD (no caso do BD, só pode ser em disco primário);
  - Opção de instalar o SGBD ou não.

Anexo VI

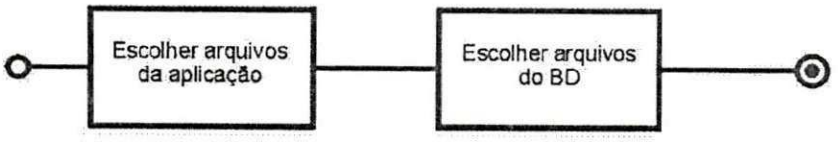
DIAGRAMA DO MODELO ARQUITETURAL



ANEXO VII

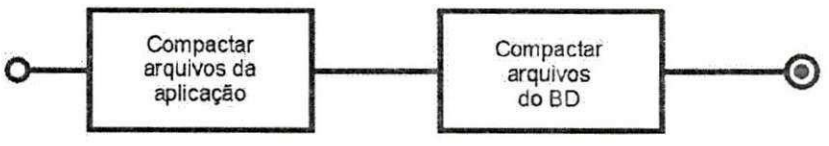
DIAGRAMAS DE USE-CASES DO CONFIGURADOR DE INSTALAÇÃO

ESCOLHER ARQUIVOS:

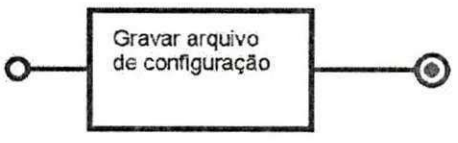


COPIAR INSTALADOR DE SGBD:

COMPACTAR ARQUIVOS:



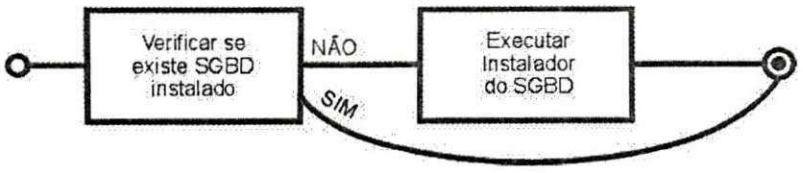
GRAVAR CONFIGURAÇÃO:



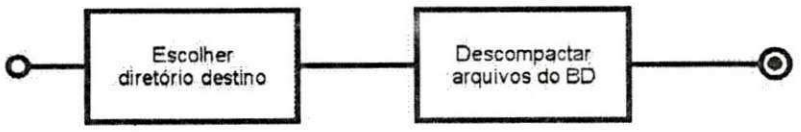
ANEXO VIII

DIAGRAMAS DE USE-CASES DO INSTALADOR

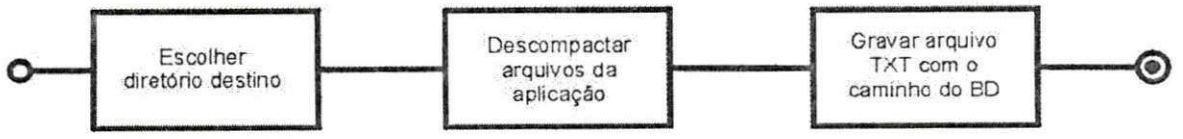
INSTALAR SGBD:



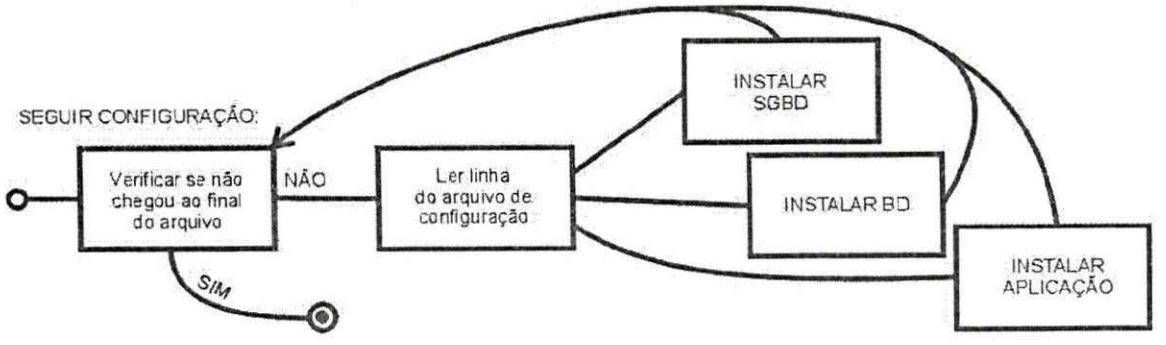
INSTALAR BD:



INSTALAR APLICAÇÃO:

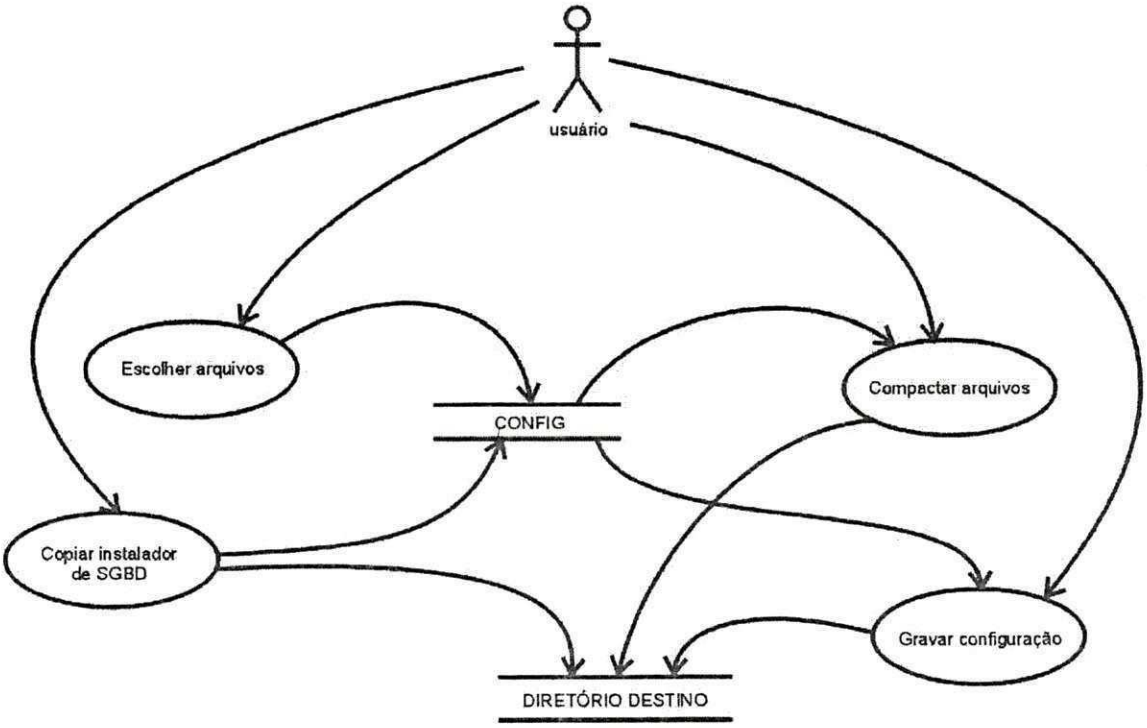


SEGUIR CONFIGURAÇÃO:

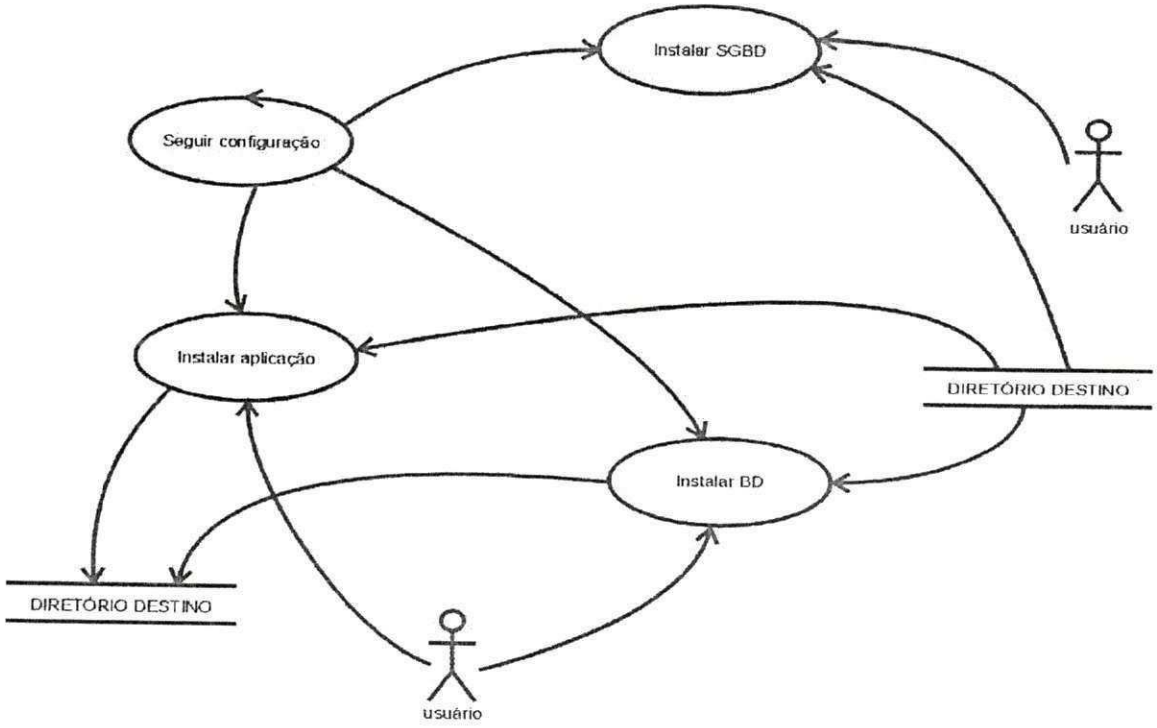


ANEXO IX

DIAGRAMA DE COLABORAÇÃO DO CONFIGURADOR DE INSTALAÇÃO

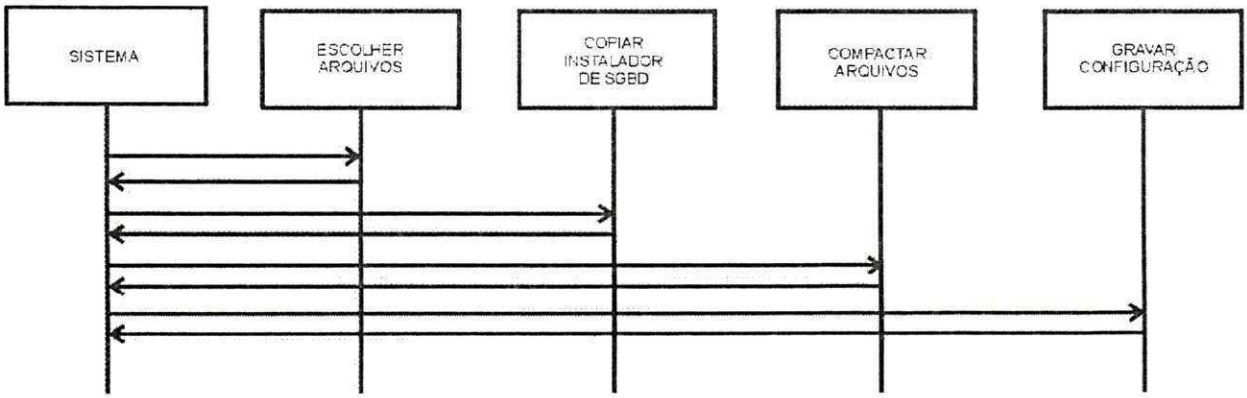


ANEXO X
DIAGRAMA DE COLABORAÇÃO DO INSTALADOR



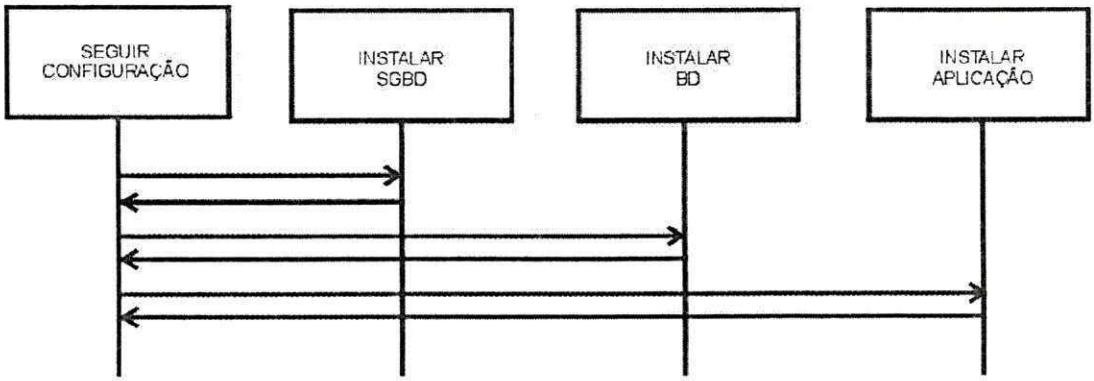
ANEXO XI

DIAGRAMA DE SEQÜÊNCIA DO CONFIGURADOR DE INSTALAÇÃO



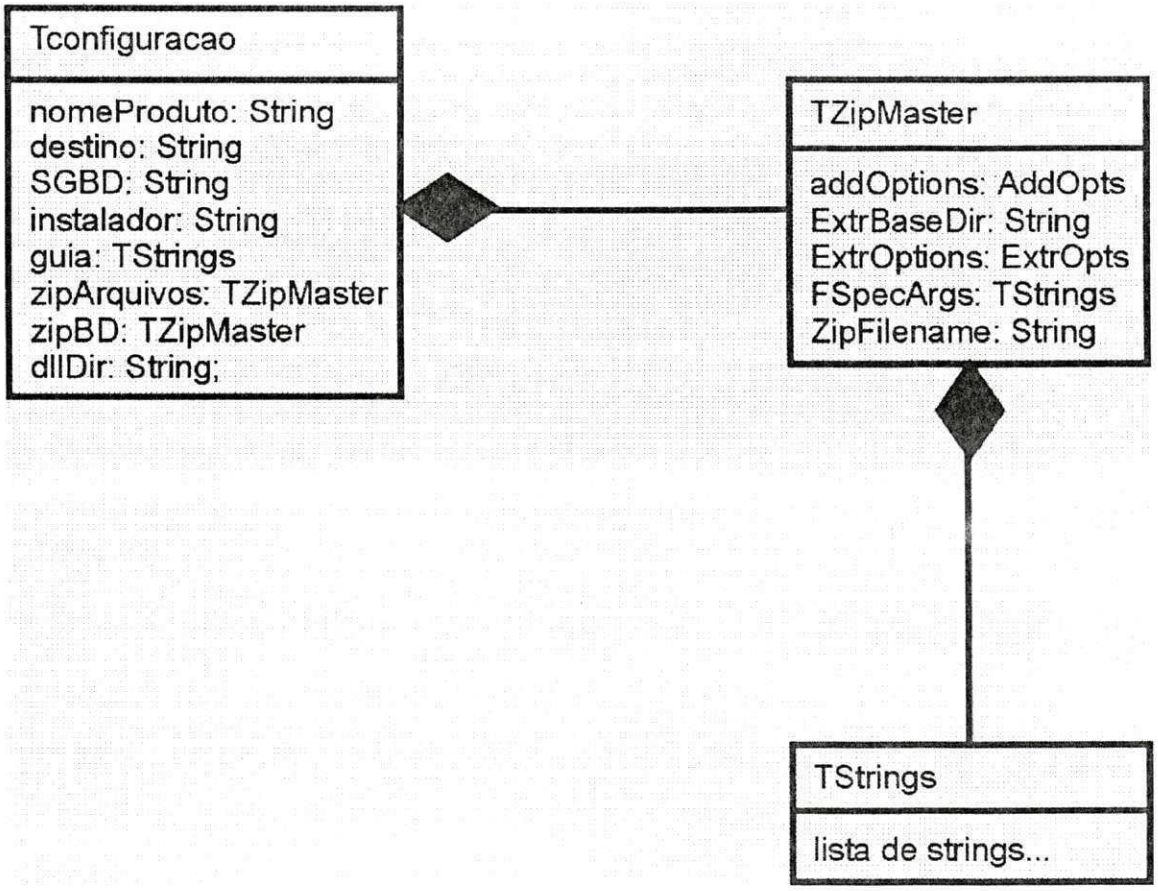
ANEXO XII

DIAGRAMA DE SEQÜÊNCIA DO INSTALADOR



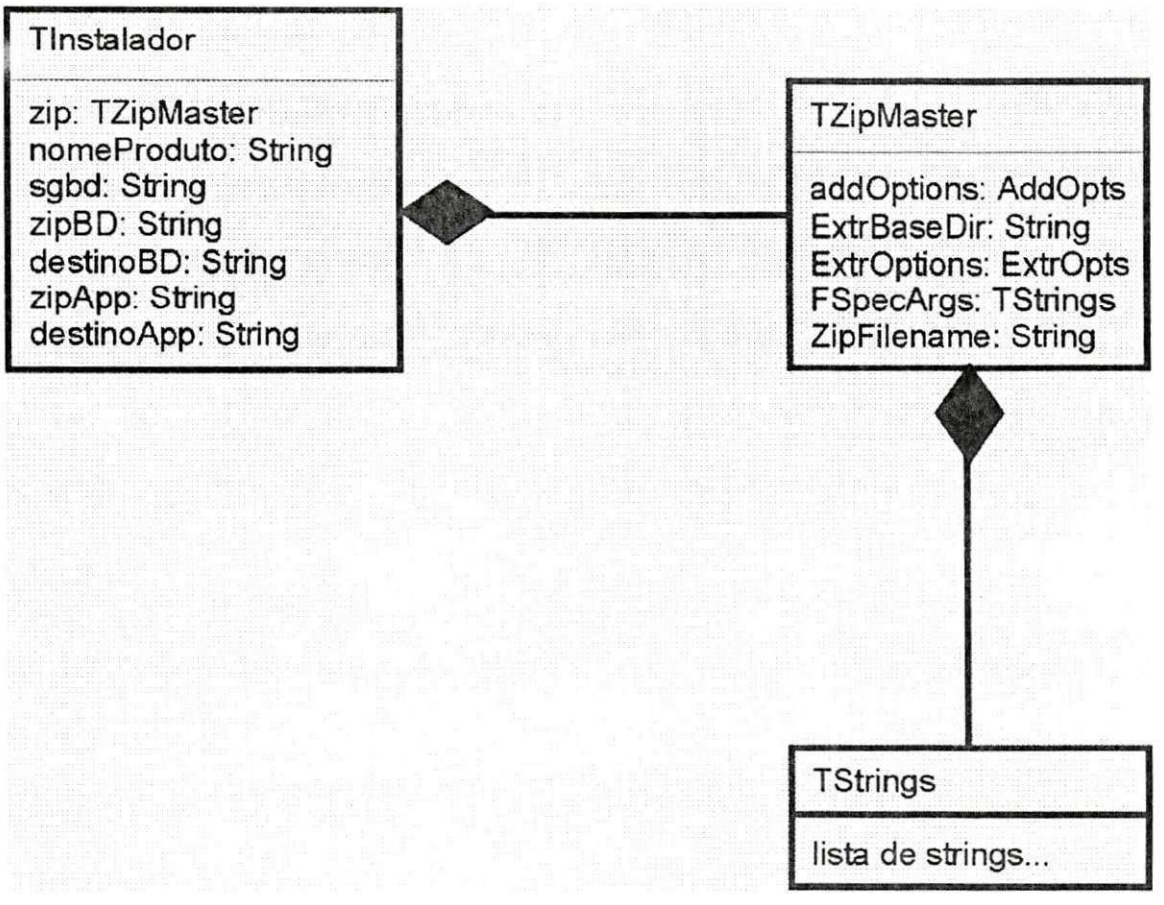
Anexo XIII

Diagrama de classes do módulo de Configuração do Instalador



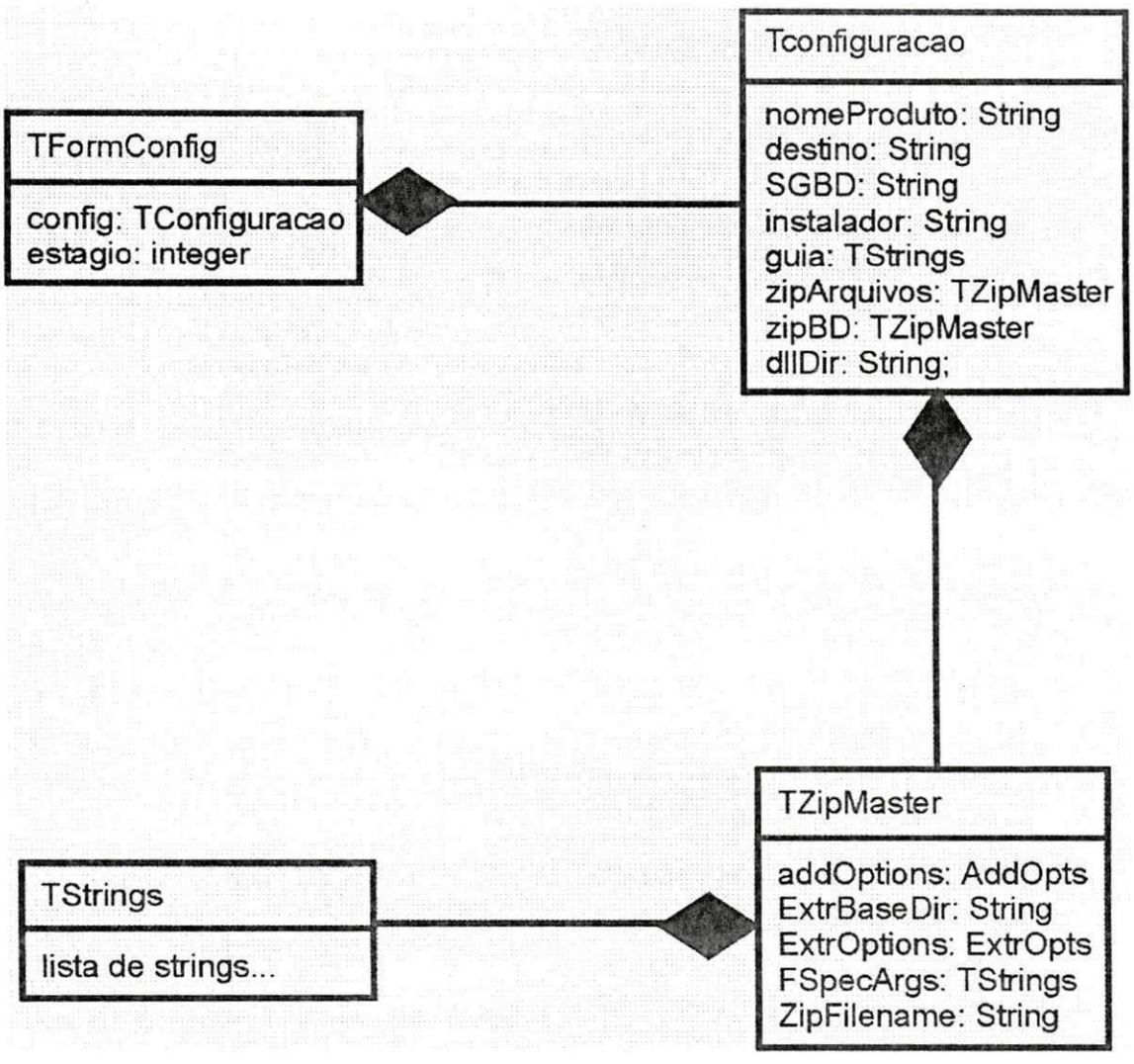
Anexo XIV

Diagrama de classes do módulo do Instalador



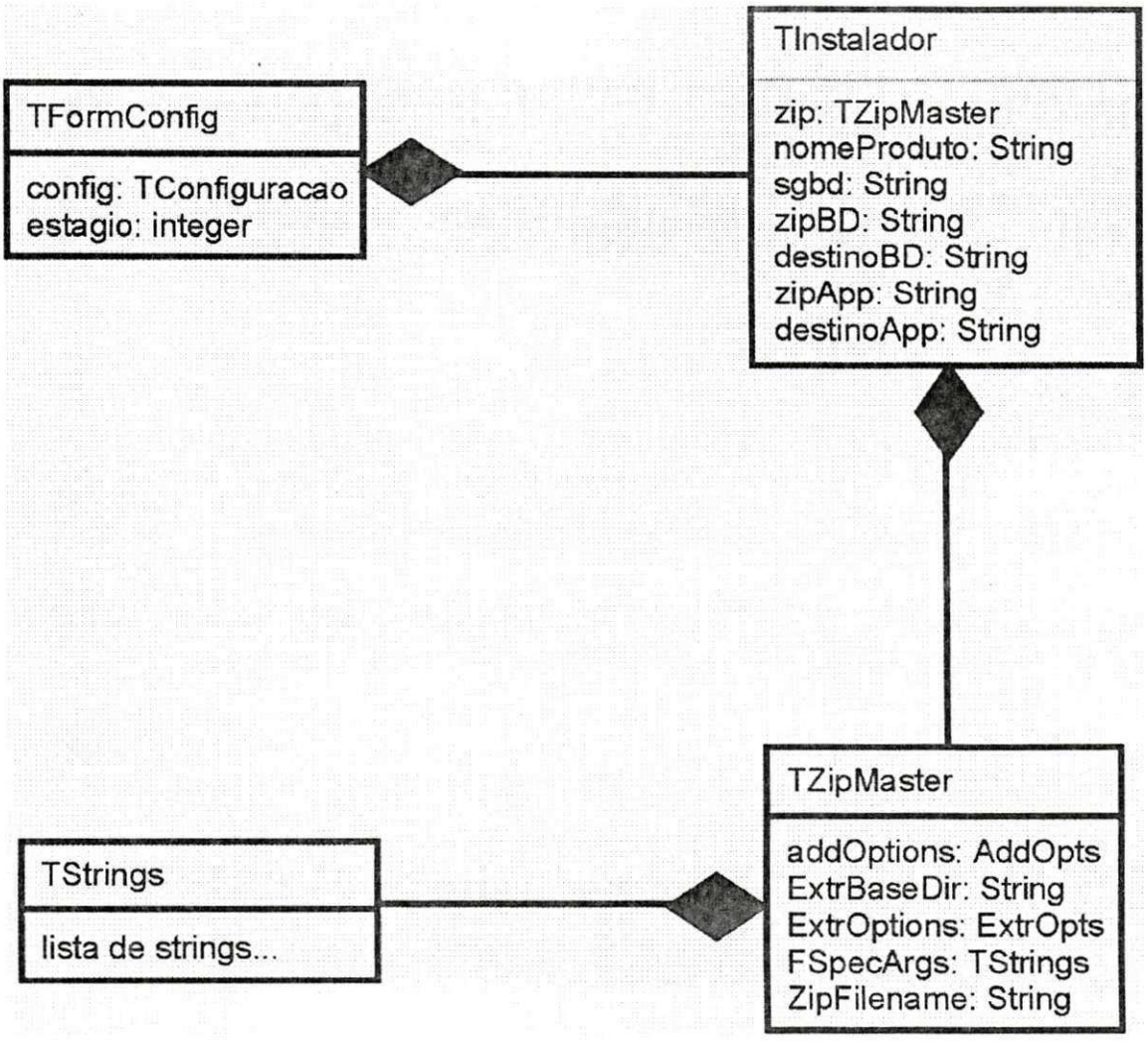
Anexo XV

Diagrama de classes do módulo Interface Gráfica de Configuração do Instalador



Anexo XVI

Diagrama de classes do módulo Interface Gráfica do Instalador



Anexo XVII

Tabela dos comandos do guia de instalação

Comando	Descrição
Ler: <nomeArquivo>	Comando da guia mestre, para o programa ler uma guia auxiliar no arquivo especificado.
NomeGuia: <nome>	Nome da guia, será apresentado como a descrição do tipo de instalação.
NomeProduto: <nome>	Define o nome do produto a ser instalado. Este nome aparecerá na tela de fundo do programa de instalação.
Instala: <opção>	Se opção: BD: instala os arquivos do BD App: instala os arquivos da aplicação SGBD: executa o instalador do SGBD

Anexo XVIII

Códigos de testes da classe TConfiguracao

```
unit TestaConfiguracao;

interface

uses Classes, TestFrameWork, Configuracao;

type
   TTestaConfiguracao = class(TTestCase)
   private
      config: TConfiguracao;
   protected
      procedure setUp; override;
      procedure tearDown; override;
   published
      procedure testaCreate;
      procedure testaGetESetNomeProduto;
      procedure testaGetESetArquivoInstalador;
      procedure testaGetESetGuia;
      procedure testaGetESetSGBD;
      procedure testaGetESetDestino;
   end;

implementation

{ TTestaConfiguracao }

procedure TTestaConfiguracao.setUp;
begin
   inherited;
   config := TConfiguracao.create;
end;

procedure TTestaConfiguracao.tearDown;
begin
   config.Free;
   inherited;
end;

procedure TTestaConfiguracao.testaCreate;
var
   config: TConfiguracao;
begin
   with config do
   begin
      check(getNomeProduto = '',
          'Erro na criação: nome do produto não vazio!');
      check(getArquivoInstalador = '',
          'Erro na criação: caminho do instalador não vazio!');
      check(getArquivos = nil,
          'Erro na criação: lista de arquivos não vazia!');
      check(getBD = nil,
          'Erro na criação: lista de arquivos de BD não vazia!');
      check(getGuia = nil,
          'Erro na criação: guia de instalação não vazio!');
   end;
end;
```

```
procedure TTestaConfiguracao.testaGetESetArquivoInstalador;
var
   arqInst: String;
begin
   arqInst := 'C:\lalala.exe';
   config.setArquivoInstalador(arqInst);
   check(config.getArquivoInstalador = arqInst,
               'Caminho do instalador: erro no get ou no set!');
end;

procedure TTestaConfiguracao.testaGetESetDestino;
var
   dest: String;
begin
   dest := 'Lalala';
   config.setDestino(dest);
   check(config.getDestino = dest,
            'Diretório destino: erro no get ou no set!');
end;

procedure TTestaConfiguracao.testaGetESetGuia;
var
   g: TStringList;
begin
   // Guia vazio
   g:=nil;
   config.setGuia(g);
   check(config.getGuia = nil,
               'Guia vazio: erro no get ou no set!');
   // Guia preenchido
   g := TStringList.Create;
   g.Add('a');
   g.Add('b');
   config.setGuia(g);
   check(config.getGuia = g,
               'Guia preenchido: erro no get ou no set!');
end;

procedure TTestaConfiguracao.testaGetESetNomeProduto;
var
   nome: String;
begin
   nome := 'Lalala';
   config.setNomeProduto(nome);
   check(config.getNomeProduto = nome,
               'Nome do produto: erro no get ou no set!');
end;

procedure TTestaConfiguracao.testaGetESetSGBD;
var
   sgbd: String;
begin
   sgbd := 'Lalala';
   config.setSGBD(sgbd);
   check(config.getSGBD = sgbd,
               'Caminho do SGBD: erro no get ou no set!');
end;

end.
```

Anexo XIX

Códigos de testes da classe TGuia

```
unit TestaGuia;

interface

uses Classes, TestFrameWork, Guia;

type
   TTestaGuia = class(TTestCase)
   private
      guia: TGuia;
      procedure criaArquivo;
      procedure criaArquivoVazio;
   protected
      procedure setUp; override;
      procedure tearDown; override;
   published
      procedure testaCreate;
      procedure testaGetComando;
   end;

implementation

uses SysUtils;

{ TTestaGuia }

procedure TTestaGuia.criaArquivo;
var
   fg: TextFile; // Arquivo-guia de instalação
begin
   inherited;
   assignFile(fg, 'guia.guia');
   rewrite(fg);
   writeln(fg, 'NomeProduto: Panificador');
   writeln(fg, 'InstalaApp');
   CloseFile(fg);
end;

procedure TTestaGuia.criaArquivoVazio;
var
   fg: TextFile; // Arquivo-guia de instalação
begin
   inherited;
   assignFile(fg, 'guia.guia');
   rewrite(fg);
   CloseFile(fg);
end;

procedure TTestaGuia.setUp;
begin
   inherited;
   criaArquivo;
   guia := TGuia.create;
end;

procedure TTestaGuia.tearDown;
```

```
begin
   guia.free;
   DeleteFile('guia.guia');
   inherited;
end;

procedure TTestaGuia.testaCreate;
begin
   check(guia <> nil, 'Objeto de configuração não criado!');
   guia.free;
   try
      criaArquivoVazio;
      guia := TGuia.create;
      check(guia = nil, 'Objeto de configuração criado com arquivo
de configuração vazio!');
   except
      criaArquivo;
      guia := TGuia.create;
   end;
end;

procedure TTestaGuia.testaGetComando;
var
   com: TComandoDeGuia;
begin
   // Testa comando com parâmetros
   com := guia.getComando(0);
   check( com.comando = 'NomeProduto',
          'Comando lido: "'+com.comando+
          '", devia ser: "NomeProduto"');
   check( com.param = 'Panificador',
          'Parâmetro lido: "'+com.param+
          '", devia ser: "Panificador"');
   // Testa comando sem parâmetros
   com := guia.getComando(1);
   check( com.comando = 'InstalaApp',
          'Comando lido: "'+com.comando+
          '", devia ser: "InstalaApp"');
   check( com.param = '',
        'Parâmetro lido: "'+com.param+'", devia ser vazio');
end;

end.
```

Anexo XX

API da classe TConfiguração

Class: TConfiguracao

```
TObject ----+
            |
      TConfiguracao
```

Description:
Classe que gerenciará o processo de configuração do instalador.

Unit:
Configuracao (Configuracao.pas)

Constructors:

constructor create;

Cria um objeto com os parâmetros vazios

Methods:

procedure addGuia(novaGuiaAux: TStrings);

Adiciona uma guia auxiliar

procedure clearGuiasAux;

Limpa o array de guias auxiliares

function criaGuiaAux(nome: String; Opcoes: TItensDeInstalacao): TStrings;

Cria arquivo de guia auxiliar com as opções especificadas

function getArquivoInstalador: String;

Lê caminho do arquivo instalador da aplicação

function getArquivos: TStrings;

Lê a lista de arquivos de aplicação

function getBD: TStrings;

Lê a lista de arquivos do BD

function getDestino: String;

Lê o caminho onde os arquivos de destino serão gravados

function getGuia: TStrings;

Lê guia de instalação

function getNomeProduto: String;

Lê o nome do produto

function getSGBD: String;

Lê o caminho do instalador do SGBD

procedure gravaGuia;

Grava o guia de instalação num arquivo

function Instala: boolean;

Compacta arquivos, realiza cópias, etc.

procedure setArquivoInstalador(arq: String);

Define o caminho do arquivo de instalação

procedure setArquivos(arqList: TStrings);

Define a lista de arquivos de aplicação

procedure setBD(arqList: TStrings);

Define a lista de arquivos do BD

procedure setDestino(dest: String);

Define o caminho onde os arquivos de destino serão gravados

procedure setDLLDir(dir: String);

Define o diretório do arquivo de DLL necessário para dezipar os arquivos

procedure setGuia(guiaInst: TStrings);

Define guia de instalação

procedure setNomeProduto(nome: String);

Define o nome do Produto

procedure setSGBD(caminhoSGBD: String);

Define o caminho dos arquivos de instalação do SGBD

Constructors:

constructor create;

Cria um objeto com os parâmetros vazios

Methods:

procedure addGuia(novaGuiaAux: TStrings);

Adiciona uma guia auxiliar

procedure clearGuiasAux;

Limpa o array de guias auxiliares

function criaGuiaAux(nome: String; Opcoes: TItensDeInstalacao): TStrings;

Cria arquivo de guia auxiliar com as opções especificadas

function getArquivoInstalador: String;

Lê o caminho do instalador da aplicação

Returns

caminho do arquivo instalador da aplicação

function getArquivos: TStrings;

Lê a lista de arquivos de aplicação

Returns

lista de arquivos

function getBD: TStrings;

Lê a lista de arquivos do BD

Returns

lista de arquivos do BD

function getDestino: String;

Lê o caminho onde os arquivos de destino serão gravados

Returns

diretório onde os arquivos serão gravados

function getGuia: TStrings;

Lê guia de instalação

Returns

guia da instalação

function getNomeProduto: String;

Lê o nome do produto

Returns

nome do produto

function getSGBD: String;

Lê o caminho do SGBD

Returns

caminho do instalador do SGBD

procedure gravaGuia;

Grava o guia de instalação num arquivo

function Instala: boolean;

Compacta arquivos, realiza cópias, etc.

Returns

Se copiou/compactou os devidos arquivos com sucesso

procedure setArquivoInstalador(arq: String);

Define o caminho do arquivo de instalação

Parameters

arq : caminho do arquivo instalador do produto

procedure setArquivos(arqList: TStrings);

Define a lista de arquivos do BD

Parameters

arqList : lista de arquivos da aplicação a serem copiados

procedure setBD(arqList: TStrings);

Define a lista de arquivos do BD

Parameters
arqList : lista dos arquivos de BD a serem comprimidos

procedure setDestino(dest: String);
Define o caminho onde os arquivos de destino serão gravados
Parameters
dest : diretório onde os arquivos serão gravados

procedure setDLLDir(dir: String);
Define o diretório do arquivo de DLL necessário para dezipar os arquivos

procedure setGuia(guiaInst: TStrings);
Define guia de instalação.

procedure setNomeProduto(nome: String);
Define o nome do Produto
Parameters
nome : nome do produto

procedure setSGBD(caminhoSGBD: String);
Define o caminho do SGBD
Parameters
caminhoSGBD : caminho do instalador do SGBD

Anexo XXI

API da classe TGuia

Class: TGuia

```
TObject ----+
            |
          TGuia
```

Description:
Classe que encapsula o guia de instalação

Unit:
Guia (Guia.pas)

Constructors:

constructor create(sArquivo: String);

Construtor

Methods:

function getComando(index: integer): TComandoDeGuia;

Retorna o comando e parâmetro com o índice indicado

Constructors:

```
constructor create(sArquivo: String);
```

Construtor

Methods:

```
function getComando(index: integer): TComandoDeGuia;
```

Retorna o comando e parâmetro com o índice indicado

Parameters

index : índice do comando a receber

Returns

Comando com o parâmetro

Anexo XXII

API da classe Instalador

Class: TInstalador

```
TObject ----+
            |
       TInstalador
```

Description:

Unit:
Instalador (Instalador.pas)

Methods:

function getDestinoApp: String;

Retorna o caminho onde os arquivos da aplicação serão descompactados

function getDestinoBD: String;

Retorna o caminho onde os arquivos do BD serão descompactados

function getNomeProduto: String;

Retorna o nome do produto

function getSGBD: String;

Retorna o caminho completo do instalador do SGBD

function getZipApp: String;

Retorna o caminho completo do ZIP dos arquivos da aplicação

function getZipBD: String;

Retorna o caminho completo do ZIP dos arquivos do BD

procedure Instala;

Descompacta arquivos e executa o instalador de SGBD

procedure setDestinoApp(caminho: String);

Define o caminho onde os arquivos da aplicação serão descompactados

procedure setDestinoBD(caminho: String);

Define o caminho onde os arquivos do BD serão descompactados

procedure setNomeProduto(nome: String);

Define o nome do produto

procedure setSGBD(caminho: String);

Define o caminho completo do instalador do SGBD

procedure setZipApp(caminho: String);

Define o caminho completo do ZIP dos arquivos da aplicação

procedure setZipBD(caminho: String);

Define o caminho completo do ZIP dos arquivos do BD

Methods:

function getDestinoApp: String;
Retorna o caminho onde os arquivos da aplicação serão descompactados
Returns
destino da descompactação dos arquivos da aplicação

function getDestinoBD: String;
Retorna o caminho onde os arquivos do BD serão descompactados
Returns
destino da descompactação dos arquivos do BD

function getNomeProduto: String;
Retorna o nome do produto
Returns
nome do produto

function getSGBD: String;
Retorna o caminho completo do instalador do SGBD
Returns
caminho do instalador do SGBD

function getZipApp: String;
Retorna o caminho completo do ZIP dos arquivos da aplicação
Returns
caminho do ZIP da aplicação

function getZipBD: String;
Retorna o caminho completo do ZIP dos arquivos do BD
Returns
caminho do ZIP do BD

procedure Instala;
Descompacta arquivos e executa o instalador de SGBD

procedure setDestinoApp(caminho: String);
Define o caminho onde os arquivos da aplicação serão descompactados
Parameters
caminho : destino de descompactação da aplicação

procedure setDestinoBD(caminho: String);
Define o caminho onde os arquivos do BD serão descompactados
Parameters
caminho : destino de descompactação do BD

procedure setNomeProduto(nome: String);
Define o nome do produto
Parameters
nome : nome do produto

procedure setSGBD(caminho: String);
Define o caminho completo do instalador do SGBD
Parameters
caminho : caminho do instalador do SGBD

procedure setZipApp(caminho: String);
Retorna o caminho completo do ZIP dos arquivos da aplicação
Parameters
caminho : caminho do ZIP da aplicação

procedure setZipBD(caminho: String);
Retorna o caminho completo do ZIP dos arquivos do BD
Parameters
caminho : caminho do ZIP do BD