

UNIVERSIDADE REGIONAL DE BLUMENAU
CENTRO DE CIÊNCIAS EXATAS E NATURAIS
CURSO DE CIÊNCIAS DA COMPUTAÇÃO – BACHARELADO

**AVALIAÇÃO DA MIGRAÇÃO DE UMA APLICAÇÃO
DELPHI 7 (WIN32) PARA DELPHI 8 (MICROSOFT .NET)**

TIAGO CONCEIÇÃO

BLUMENAU
2004

2004/2-45

TIAGO CONCEIÇÃO

**AVALIAÇÃO DA MIGRAÇÃO DE UMA APLICAÇÃO
DELPHI 7 (WIN32) PARA DELPHI 8 (MICROSOFT .NET)**

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 Ciências
da Computação — Bacharelado.

Prof. Maurício Capobianco Lopes - Orientador

**BLUMENAU
2004**

2004/2-45

AVALIAÇÃO DA MIGRAÇÃO DE UMA APLICAÇÃO DELPHI 7 (WIN32) PARA DELPHI 8 (MICROSOFT .NET)

Por

TIAGO CONCEIÇÃO

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

Presidente:

Prof. Maurício Capobianco Lopes. – Orientador, FURB

Membro:

Prof. Marcel Hugo, FURB

Membro:

Prof. Paulo R. Dias, FURB

Blumenau, 15 de fevereiro de 2005

Dedico este trabalho aos meus pais por todo o incentivo e apoio que me foi oferecido, ao meu orientador que acreditou na realização deste, e aos meus inseparáveis amigos(as).

AGRADECIMENTOS

À minha família, que sempre me apoiou em todos meus momentos, principalmente nos momentos difíceis.

Ao meu orientador, por acreditar e colaborar para a realização deste trabalho.

Aos meus amigos que compreenderam minha ausência em algumas festas, decorrente da realização deste.

RESUMO

Este trabalho apresenta a migração de uma aplicação desenvolvida em plataforma Win32 para .NET. A aplicação desenvolvida foi um sistema que controla as solicitações para programação e análise de empresas desenvolvedoras de software. Para o desenvolvimento do sistema em Win32 foi utilizado o Delphi 7 e para migrar para .NET foi utilizado o Delphi 8. A principal finalidade deste trabalho é apresentar uma análise dos principais pontos da migração e fazer uma comparação da performance da aplicação desenvolvida nas duas plataformas. Como resultado da migração pode-se avaliar que não é uma tarefa complexa. Sendo que a fase de compilação necessitou de poucas modificações no código fonte para mantê-lo compatível nas duas versões do Delphi. Entretanto foi difícil descobrir as causas dos erros ocorridos tanto na compilação quanto na execução devido a carência de documentação relativa aos mesmos. O comparativo de performance demonstrou que a versão Win32 chega a ser, em alguns casos, até 4 vezes mais rápida que a versão .NET, sendo este um aspecto relevante a ser avaliado na hora de migrar uma aplicação. O Delphi 8 demonstrou ser uma alternativa rápida e simples para os desenvolvedores Delphi Win32 manterem seus aplicativos atualizados tecnologicamente.

Palavras chaves: Delphi; Microsoft .NET Framework; Migração de sistemas.

ABSTRACT

This work presents an application migration between Win32 and .NET platform. As a result of the developed application, a requirements control system of programming and analysis for developing softwares companies has been developed. For developing in the Win32 system, Delphi 7 has been used, whereas Delphi 8 was the choice for migration to .NET. The main work objective is presents the strong points analysis of migration, besides performance comparing between both platforms. As a result of this migration, could be evaluated that it is not a complex task. The compilation phase needed of only some changes in the source code, to keep it compatible in both Delphi versions. However, it was difficult trapping the error causes occurred in compilation time and also in runtime, because there were no documentation enough related to that errors. The performance comparing has shown that in some cases, the Win32 application version can be up to four times faster than the .NET version, doing it a relevant aspect to be evaluated when migrating an application. Delphi 8 also has shown being a simple and fast choice for Delphi developers to keep their applications technologically updated.

Key-Words: Delphi; Microsoft .NET Framework; Migration of Systems.

LISTA DE ILUSTRAÇÕES

Figura 1 - .NET Framework	14
Figura 2 – Carga, a compilação e a seqüência de execução da CLR	16
Figura 3 – Tipos na CTS	17
Figura 4 – IDE do Delphi 8	19
Figura 5 – Aplicações .NET no Delphi 8	20
Figura 6 – Diagrama de casos de uso	24
Quadro 1 – Apresentação dos casos de uso	25
Figura 7 – Diagrama de classes	26
Figura 8 – Diagrama de seqüência – Efetuar Solicitação	27
Figura 9 – Diagrama de seqüência – Revisar Solicitação	28
Figura 10 – Diagrama de seqüência – Implementar Solicitação	28
Figura 11 – Diagrama de seqüência – Testar Implementação	29
Figura 12 – Diagrama de seqüência – Gerar Solicitações	29
Figura 13 – Modelo Físico da Base de Dados	30
Quadro 2 – Estrutura da base de dados.....	32
Figura 14 – Exemplo da tela de cadastro simples (Cadastro de sistemas)	33
Figura 15 – Exemplo da tela de cadastro mestre/detalhe (Cadastro de clientes/Sistemas do cliente)	34
Figura 16 – Tela principal da aplicação.....	35
Figura 17 – Tela de cadastro de solicitações	36
Figura 18 – Tela para relacionar os solicitantes com a solicitação	36
Figura 19 – Tela para relacionar os sistemas à solicitação	37
Figura 20 – Tela de revisão da solicitação	37
Figura 21 – Tela de implementação da solicitação	38
Figura 22 – Tela de informações dos testes.....	39
Figura 23 – Tela opções do projeto (<i>Project Options</i>)	40
Figura 24 – Tela de mensagens de avisos.....	40
Quadro 3 – Diretivas de compilação	41
Figura 25 – Mensagens de erro da compilação	42
Figura 26 – Erro de arquivo não encontrado	42
Figura 27 – Informando o <i>Namespace</i> do <i>Rave</i>	43
Figura 28 – Comparativo de performance	46

LISTA DE TABELAS

Tabela 1 – Funcionalidades da tela de cadastro normal	33
Tabela 2 – Roteiro da avaliação	45
Tabela 3 – Quantidade de registros das tabelas	45
Tabela 4 – Características do computador.....	45

LISTA DE SIGLAS

HTML - Hypertext Markup Language

IDE - Integrated Development Environment

RAD - Rapid Application Development

VCL – Visual Class Library

XML - Extensible Markup Language

SUMÁRIO

1 INTRODUÇÃO.....	11
1.1 OBJETIVOS DO TRABALHO	12
1.2 ESTRUTURA DO TRABALHO	12
2 FUNDAMENTAÇÃO TEÓRICA	13
2.1 MICROSOFT .NET	13
2.1.1 DEFINIÇÃO DO .NET.....	13
2.1.2 .NET FRAMEWORK.....	13
2.1.2.1 Common Language Runtime (CLR)	14
2.1.2.1.1 Managed Modules.....	15
2.1.2.1.2 Assemblies	15
2.1.2.1.3 MSIL e JIT.....	16
2.1.2.2 Common Type System	17
2.1.2.3 Common Language Specification (CLS).....	18
2.1.2.4 .NET Framework Class Library (FCL)	18
2.2 BORLAND DELPHI 8.0 FOR THE MICROSOFT .NET FRAMEWORK	18
3 DESENVOLVIMENTO DO TRABALHO	22
3.1 REQUISITOS PRINCIPAIS DO PROBLEMA A SER TRABALHADO.....	22
3.2 ESPECIFICAÇÃO	22
3.2.1 DIAGRAMA DE CASOS DE USO	23
3.2.2 DIAGRAMA DE CLASSES	25
3.2.3 DIAGRAMA DE SEQÜÊNCIA.....	27
3.2.4 MODELO FÍSICO DA BASE DE DADOS.....	30
3.2.5 DICIONÁRIO DE DADOS.....	31
3.3 IMPLEMENTAÇÃO	32
3.3.1 TÉCNICAS E FERRAMENTAS UTILIZADAS.....	32
3.3.2 OPERACIONALIDADE DA IMPLEMENTAÇÃO.....	33
3.4 RESULTADOS E DISCUSSÃO	39
3.4.1 MIGRAÇÃO	39
3.4.2 COMPARATIVO DE PERFORMANCE	44
4 CONCLUSÕES.....	48
4.1 EXTENSÕES	49
REFERÊNCIAS BIBLIOGRÁFICAS	50

ANEXO A – Relação de componentes do Delphi 7 disponíveis no Delphi 8	52
--	----

1 INTRODUÇÃO

O acompanhamento das tendências tecnológicas na área de informática é um ponto vital para as empresas desenvolvedoras de *softwares*. A cada dia surgem novas tecnologias como, por exemplo, Microsoft .NET e Java 2 Enterprise Edition (J2EE), que se propõem a ser arquiteturas modernas para o desenvolvimento de *software*.

Esta dinâmica na oferta de tecnologia tem gerado grandes desafios para as empresas desenvolvedoras de *software*, pois se por um lado elas já têm sistemas desenvolvidos em tecnologias atuais, por outro lado manter os sistemas dentro dessas tecnologias pode defasá-los muito rapidamente.

Assim, neste trabalho pretende-se avaliar o impacto destas novas tecnologias, especificamente para o ambiente Delphi, que, a partir da versão 8, optou claramente por incorporar as características da plataforma .NET.

A plataforma .NET tem a proposta de proporcionar um ambiente de desenvolvimento avançado, disponibilizando recursos poderosos para uso dos desenvolvedores. Para isso, a Microsoft unificou todas as soluções de desenvolvimento dela nessa nova plataforma, além de melhorar bastante os recursos oferecidos. Pode-se dizer que o .NET *Framework* disponibiliza um ambiente de desenvolvimento multi-plataforma (em relação ao sistema operacional), multi-linguagem, orientada a objetos, e com uma grande e eficiente biblioteca de classes (D' Angelo, 2003b).

A Borland lançou o Delphi para .NET, para permitir que os desenvolvedores pudessem usar imediatamente suas habilidades com a estrutura e linguagem Delphi e aproveitar a maior parte dos investimentos prévios em código fonte de aplicação Delphi para começar a desenvolver aplicações para a plataforma .NET. O Delphi simplificou a migração de aplicações Win32 para a .NET de seus usuários de versões anteriores, tornando-se, assim, uma ferramenta de desenvolvimento competitiva na plataforma .NET (BORLAND, 2003b). Nesta nova versão do Delphi passou a ser possível o desenvolvimento de aplicações ASP.NET, permitindo construir serviços WEB XML e aplicações HTML dinâmicas de servidor, ou seja, as páginas HTML não estão prontas no servidor, mas elas são criadas dinamicamente pelas aplicações e serviços WEB.

Para aplicativos *Desktop*, existem extensões denominadas "Windows Forms", que são APIs utilizadas para se desenvolver o layout das telas do aplicativo. No caso de aplicações WEB (ASP.NET), por exemplo, essas APIs não são utilizadas. No entanto, existem APIs de auxílio no desenvolvimento de aplicações WEB denominadas WEBFORMS e WEBCONTROLS (D' Angelo, 2003a).

Neste sentido, o presente trabalho visa desenvolver uma aplicação para a plataforma Microsoft Win32 usando o Delphi 7 e migrar esta aplicação para a plataforma Microsoft .NET usando o Delphi 8, a fim de avaliar o impacto dessa mudança tecnológica. Logo após a migração, será demonstrada uma avaliação comparativa de performance das aplicações geradas.

A aplicação desenvolvida para esta avaliação é um controlador de Solicitações para Programação e Análise (SEPA) para empresas desenvolvedoras de *software*.

1.1 OBJETIVOS DO TRABALHO

O objetivo deste trabalho foi desenvolver uma aplicação para plataforma Microsoft Win32 usando o Delphi 7, para então migrar a aplicação para plataforma .NET da Microsoft usando o Delphi para .NET e comparar a performance das duas aplicações (Win32 e .NET).

Os objetivos específicos do trabalho são:

- a) desenvolver a aplicação SEPA usando o Delphi 7;
- b) verificar a portabilidade da aplicação desenvolvida em Delphi 7 (plataforma Microsoft Win32) para Delphi 8 (plataforma Microsoft .NET), avaliando as facilidades e problemas desta migração;
- c) definir critérios de avaliação e fazer uma avaliação de performance das aplicações.

1.2 ESTRUTURA DO TRABALHO

O capítulo 2 apresenta a fundamentação teórica na qual este trabalho é embasado, com assuntos relativos a Microsoft .NET e o Delphi 8.

No capítulo 3 são abordados os processos envolvidos no desenvolvimento do presente trabalho, tais como, a definição dos principais requisitos, especificação apresentando os diagramas de casos de uso, diagrama de classes, diagramas de seqüência, modelo físico da base de dados e dicionário de dados, técnicas e ferramentas utilizadas, a operacionalidade do sistema, a migração do sistema, o comparativo de performance e uma explanação sobre os resultados obtidos.

O capítulo 4 apresenta as conclusões sobre os objetivos propostos, bem como uma discussão a respeito de propostas para trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo são apresentados os principais assuntos inerentes a este trabalho, tais como: Microsoft .NET e Delphi 8 para .NET.

2.1 MICROSOFT .NET

Nesta seção será dada a definição da plataforma .NET bem como o funcionamento do .NET *Framework*.

2.1.1 DEFINIÇÃO DO .NET

O .NET é uma plataforma de software que conecta informações, sistemas, pessoas e dispositivos, independentemente do sistema operacional, do tipo de computador, ou que linguagem de programação tenha sido utilizada na sua criação. A plataforma .NET conecta uma grande variedade de tecnologias de uso pessoal e de negócios, de telefones celulares a servidores corporativos, permitindo o acesso a informações importantes, onde e sempre que forem necessárias (MICROSOFT, 2003).

Desenvolvido sobre os padrões de *Web Services XML* (unidades de aplicações que provêm dados e serviços a outras aplicações), .NET possibilita que sistemas e aplicativos, novos ou já existentes, conectem seus dados e transações (MICROSOFT, 2003).

2.1.2 .NET FRAMEWORK

O .NET Framework é o núcleo tecnológico para fazer e executar aplicações .NET. Ele consiste basicamente em dois componentes: *Common Language Runtime (CLR)* e a *Framework Class Library (FCL)* (PACHECO, 2004, p. 6) que serão abordados mais adiante. Mas existem outros conceitos e definições que devem ser esclarecidos para se obter um melhor entendimento do funcionamento da arquitetura .NET.

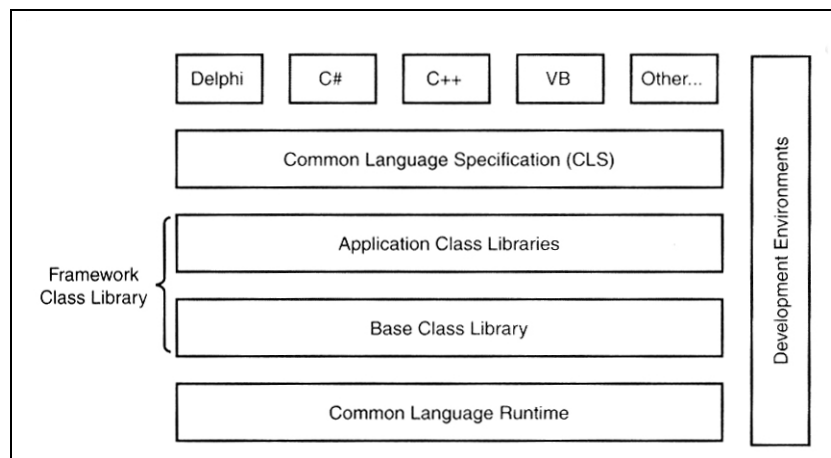
Abaixo será apresentada passo-a-passo, uma visão bem macro de como o .NET *Framework* funciona desde o desenvolvimento até a execução de uma aplicação .NET.

Segundo Pacheco (2004, p. 12), são eles:

- a) escolher uma linguagem .NET (Delphi.NET, C#, C++, VB e outras) para escrever o programa;

- b) compilar o código para uma linguagem intermediária - *Intermediate Language (IL)* - residente em um módulo gerenciado (pacotes que contêm todas as informações necessárias para o seu funcionamento);
- c) ligar o módulo gerenciado a um *assembly* (empacotamento do módulo gerenciado no formato da plataforma de destino);
- d) gerar o *assembly* conforme a plataforma de destino;
- e) chamar o CLR, que carrega, compila, executa e controla o código.

Na Figura 1 são mostrados diversos componentes utilizados para fazer todo esse trabalho. Na linha de cima, as linguagens de programação .NET, logo abaixo a *Common Language Specification (CLS)*, que são as regras para as linguagens de programação .NET., a *Framework Class Library (FCL)*, a *Common Language Runtime (CLR)*, e os ambientes de desenvolvimentos tais como o Delphi para .NET e Visual Studio .NET. Todos esses componentes serão detalhados mais tarde.



Fonte: Pacheco (2004, p. 13)

Figura 1 - .NET Framework

2.1.2.1 Common Language Runtime (CLR)

O CLR é o elemento chave para o .NET *Framework*, pois ele é quem gerencia a execução das aplicações .NET e fornece os *runtime services* para estas aplicações. Quando o CLR é chamado, ele executa as operações requeridas para compilação, alocação de memória e gerenciamento do código da aplicação executada (PACHECO, 2004, p. 13).

O CLR funciona em uma representação intermediária do código original escrito em algum compilador de linguagem .NET, assim como o Delphi e o C#. Esse código

intermediário, chamado de *Microsoft Intermediate Language* (MSIL) ou simplesmente de IL, fica em um arquivo chamado de *Managed Module* (PACHECO, 2004, p. 13). *Managed Modules* contém outras informações usadas pelo CLR, que serão abordadas nas subseções a seguir.

2.1.2.1.1 Managed Modules

Os módulos gerenciados são gerados pelos compiladores de linguagens .NET. Um módulo gerenciado é um *Windows Portable Executable* (PE) e consiste em quatro partes:

- a) *PE Header*: consiste em um cabeçalho padrão Windows PE;
- b) *CLR Header*: consiste em um cabeçalho de informação do CLR para o uso do CLR;
- c) *metadata*: consiste em um metadados contendo tabelas de descrições dos tipos (classes, estruturas, tipos enumerados entre outros), membros (propriedades, métodos, eventos etc.) e referências para outros módulos;
- d) *managed code*: consiste em uma linguagem .NET comum gerada pelos compiladores de linguagem .NET. Esta linguagem é chamada de *Microsoft Intermediate Language*.

Mesmo que o módulo gerenciado seja um executável portátil, ele precisa do CLR para executar, ou seja, não pode ser executado independentemente, pois primeiro ele deve ser incorporado em um *assembly*.

2.1.2.1.2 Assemblies

Assemblies são unidades de empacotamento e de distribuição para aplicações .NET. Elas podem conter uma aplicação completa ou uma parte funcional de uma outra aplicação. Um *assembly* é de fato uma aplicação .NET.

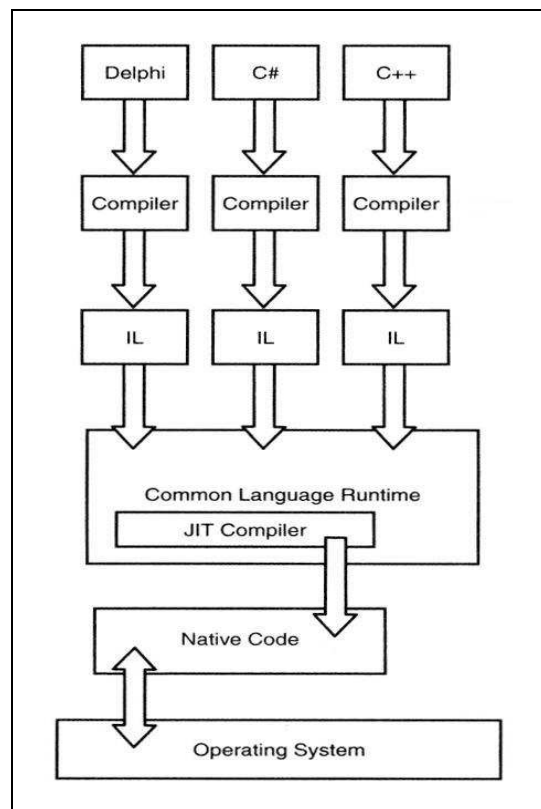
Os *assemblies* podem conter um ou mais *managed modules*. Eles podem ser comparados com os clássicos executáveis Win32 de extensão “.exe” e com as *Dynamic Link Libraries* (DLLs) de extensão “.dll”, só que eles possuem maior segurança no acesso ao código, na definição de tipos, no gerenciamento de memória e no controle de versão.

2.1.2.1.3 MSLI e JIT

Microsoft Intermediate Language (MSIL) é uma linguagem intermediária gerada pelos compiladores de linguagem .NET, tais como Delphi.NET e C#.

Através da MSIL é possível criar aplicações .NET escritas em diferentes linguagens .NET. Em outras palavras, é possível escrever um código usando o Visual Basic .NET e usar diretamente no Delphi.NET ou no C#.

A MSIL funciona como um código pré-compilado. Esse código é compilado pelo compilador *Just-In-Time (JIT)* quando a aplicação é executada. A Figura 2 mostra a carga, a compilação e a seqüência de execução da CLR.



Fonte: Adaptado de Pacheco (2004, p. 15)

Figura 2 – Carga, a compilação e a seqüência de execução da CLR

Primeiramente é escrito um código em uma linguagem .NET que é pré-compilada gerando uma IL. Quando o código é executado, a CLR executa várias tarefas importantes, tais como carregar o código, alocar e desalocar memória e executar a verificação de tipo. A classe carregada está preparada para a compilação. As classes só são carregadas e compiladas, à medida que a aplicação necessite delas. Após o código estar preparado, o compilador JIT

compila a linguagem intermediária para o código nativo da máquina. O JIT só compila os métodos na medida em que a aplicação pedir e somente se o código já não tenha sido compilado. Uma vez compilado, o código é executado pela CLR.

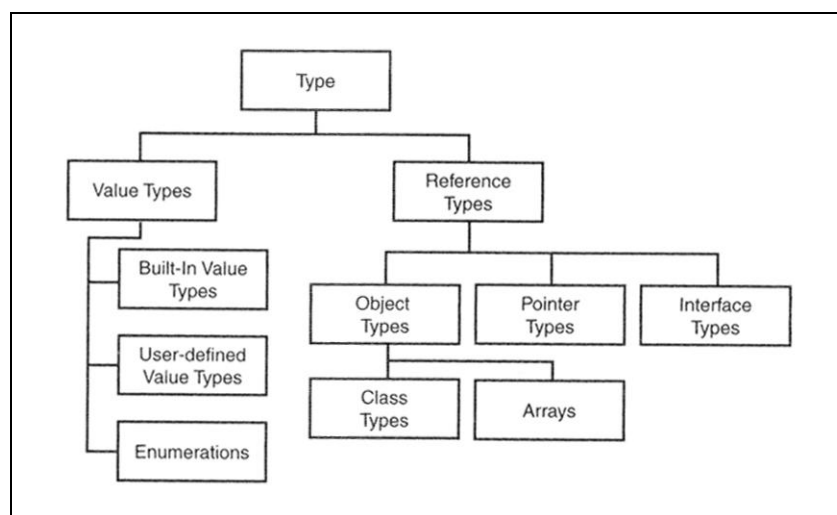
2.1.2.2 Common Type System

Common Type System (CTS) é um sistema de tipos comuns dentro da CLR. A CTS garante que todas as classes sejam compatíveis umas com as outras, descrevendo os tipos em uma forma comum, permitindo a interoperabilidade dos tipos em diferentes linguagens.

Todos os tipos derivam de uma classe base chamada *System.Object*. Esta classe base é dividida em 2 categorias, são elas:

- a) tipos por valor: estes tipos são armazenados na pilha e seus valores são acessados diretamente. Exemplos de tipos: inteiros, reais, verdadeiro ou falso, registros e enumerados;
- b) tipos por referência: são tipos armazenados na memória *heap*. Estes tipos armazenam o endereço onde o tipo está instanciado.

A Figura 3 mostra a divisão dos tipos.



Fonte: Pacheco (2004, p. 17)

Figura 3 – Tipos na CTS

2.1.2.3 Common Language Specification (CLS)

Define um conjunto de regras que as linguagens que implementam o .NET devem seguir para que a CLR possa gerar a MSIL. Em outras palavras, se um determinado compilador de linguagem segue as especificações da CLS, ela é compatível com .NET, e gera códigos MSIL compatíveis com a CLR.

Como todas as linguagens .NET devem respeitar esse conjunto de regras, pode-se dizer que um programa escrito em C#, VB.NET, Delphi.NET dentre outras linguagens .NET, ao ser compilado, tem o mesmo código MSIL (D' ANGELO, 2003b).

2.1.2.4 .NET Framework Class Library (FCL)

O *.NET Framework Class Library* (FCL) é a biblioteca da plataforma .NET de classes, interfaces, tipos dentre outros elementos utilizados para desenvolver aplicações .NET (PACHECO, 2004, p. 18).

A FCL é similar à VCL no Delphi e à MFC no Visual C++, sendo que a principal diferença é que a FCL está disponível para todas as linguagens .NET.

2.2 BORLAND DELPHI 8.0 FOR THE MICROSOFT .NET FRAMEWORK

A primeira versão do Delphi era Win16 e compilava código para o Windows 3.1. O Delphi 2 introduziu um compilador Win32 (na época, Windows 95), que continuou evoluindo até o Delphi 7. Já no Delphi 8, o compilador gera código para .NET.

O Delphi 8 dispõe de 3 edições: *Architect*, *Enterprise* e *Professional*. Em sua edição mais completa, a *Architect*, o Delphi 8 oferece soluções completas para o desenvolvimento de sistemas, com um ambiente totalmente integrado com outras ferramentas de apoio ao desenvolvimento como, por exemplo, *CaliberRM* (sistema de gerenciamento de requisitos), *StarTeam* (sistema automatizado de configuração e gerenciamento de alterações do projeto), *Optimizeit Profiler for the Microsoft® .NET Framework* (solução de performance para Microsoft .NET Framework), *Together technologies* (solução para modelagem e *design* de projetos), e ainda inclui o *Enterprise Core Objects* (ECO) para desenvolvimento orientado pelo *design* (BORLAND, 2003a).

A versão 8 do Delphi teve mudanças com relação às suas versões anteriores, tanto na IDE como nas características da linguagem. A Figura 4 apresenta a nova IDE do Delphi 8.

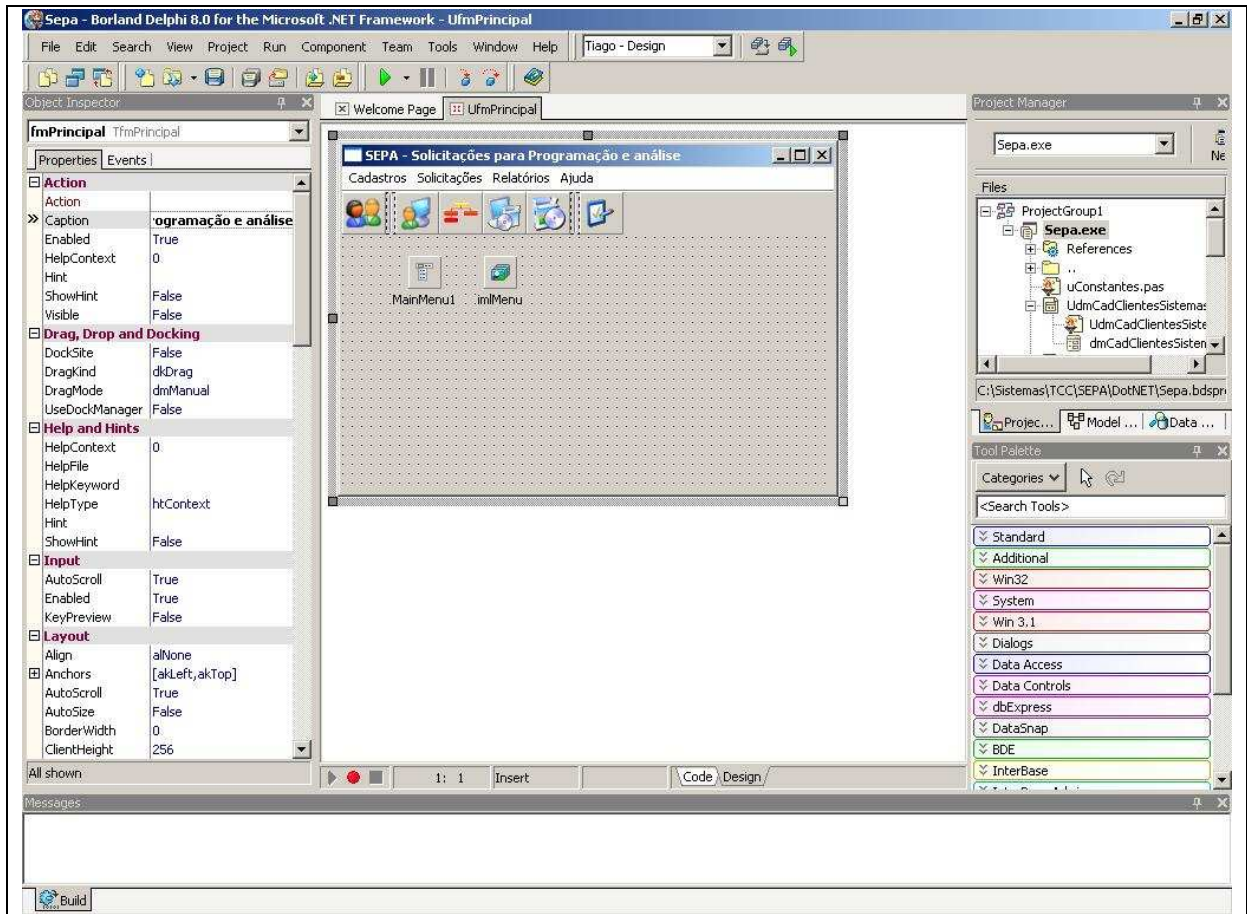
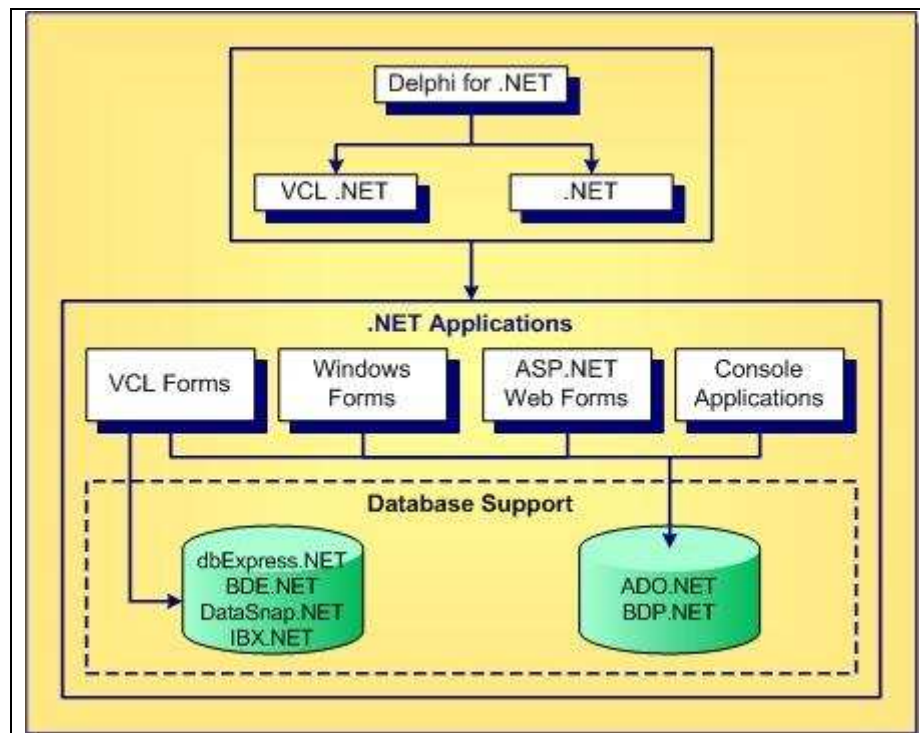


Figura 4 – IDE do Delphi 8

Pode-se observar na Figura 4, que no Delphi 8 optou-se pelo uso de janelas (“*Object Inspector*”, “*Project Manager*”, “*Tool Palette*”, “*Messages*”, etc.) do tipo *dockable* e também se restringiu o formulário do projeto a uma área fixa, não permitindo a “flutuação” do mesmo assim como era feito nas versões anteriores do Delphi.

Outra mudança está nos diferentes tipos de aplicações possíveis de serem desenvolvidas no Delphi 8 como, por exemplo, “*VCL Forms Application*”, “*Windows Forms Application*”, “*ASP.NET Forms Web Application*” e “*Console Application*”, ilustrados pela Figura 5.



Fonte: Borland Software Corporation

Figura 5 – Aplicações .NET no Delphi 8

Conforme exibido na Figura 5, VCL.NET e a estrutura .NET coexistem dentro de Delphi 8. Ambas as estruturas (VCL.NET e NET) fornecem os componentes e as funcionalidades que permitem a construção de aplicações .NET. A VCL.NET permite construir aplicações VCL Forms, enquanto o .NET permite construir aplicações *Windows Forms*, *Web Forms* e *console*, usando os componentes .NET com código Delphi encapsulado.

A VCL.NET e .NET são funcionalmente equivalentes (ver Anexo A). Assim como o .NET, a VCL.NET fornece bibliotecas de componentes, controles, classes, e funcionalidades de baixo nível que ajudam a construir aplicações que funcionam no Microsoft .NET Framework. Entretanto, segundo Swart (2004, p. 5), devido ao fato da VCL.NET ser desenvolvida baseada na VCL para Win32, ela permite mais facilmente a migração de aplicações Win32 para .NET.

O objetivo do Delphi 8 em conjunto com a VCL.NET é permitir que os desenvolvedores de versões anteriores ao Delphi 8 migrem para .NET aproveitando todo o seu conhecimento na linguagem e reutilizem boa parte do seu código fonte atual.

Outra característica do Delphi 8 é o uso de *namespace*. Para o Delphi 8, uma *unit* ainda é o recipiente básico para *types*. A CLR introduziu uma outra camada de organização chamada de *namespace*. No .NET, um *namespace* é um recipiente contextual de *types*. No Delphi 8, uma *namespace* é um recipiente de *units*. Todas as classes da VCL.NET são encontradas dentro da *namespace* “*Borland.Vcl*”.

Embora a migração da VCL para VCL para o .NET seja razoavelmente fácil, existem algumas alterações que devem ser feitas devido às diferenças entre as plataformas Win32 e .NET. Estas alterações estão relacionadas ao fato que o código do .NET será executado pelo CLR de maneira controlada, assim todo código considerado inseguro para o Delphi .NET, deve ser trocado por código seguro.

Swart (2004, p. 3-4) escreveu um artigo sobre a migração de aplicações Delphi para Win32 para Microsoft .NET usando o Delphi 8. Neste artigo são feitas analogias entre as funcionalidades do Delphi para .NET e suas versões anteriores. Como exemplos de migrações, foram usados programas que vêm junto com a instalação do Delphi (SWART, 2004, p. 10). Entretanto, o autor não fez qualquer menção relativa a comparação entre as diferentes plataformas.

Com esses novos recursos, o Delphi 8 mantém-se como uma ferramenta importante, sobretudo para os usuários de suas versões anteriores, justamente pela facilidade de migração de aplicações Win32 para .NET, tornando-se uma ferramenta competitiva no desenvolvimento RAD de aplicações para .NET.

3 DESENVOLVIMENTO DO TRABALHO

Este capítulo tem por objetivo apresentar as etapas envolvidas no desenvolvimento deste trabalho. São abordadas as seguintes etapas: requisitos, especificação, implementação, migração, comparativo de performance e os resultados obtidos.

É apresentado o sistema SEPA que foi desenvolvido como um estudo de caso com o propósito único de demonstrar a migração e exibir os resultados do comparativo de performance.

3.1 REQUISITOS PRINCIPAIS DO PROBLEMA A SER TRABALHADO

O problema a ser abordado nesse trabalho é a migração de um sistema, para isto foi utilizada a aplicação SEPA. Após o desenvolvimento do sistema SEPA, deve-se descrever passo-a-passo todos os processos realizados para efetuar a migração. Depois de migrado, deve-se fazer uma comparação de performance entre os sistemas. Para isso, antes é necessário definir os critérios a serem utilizados no comparativo que serão demonstrados mais adiante.

O sistema SEPA tem como principais requisitos, controlar as solicitações feitas para os programadores e analistas, sendo que, uma solicitação pode ser feita por mais de um solicitante e um solicitante pode ser um cliente ou um funcionário. Caso ele seja um funcionário, a solicitação deve ser cadastrada como interna, pois isso auxiliará o revisor na definição de qual versão dos sistemas a solicitação deve ser atendida. Depois de revisada, a solicitação deve ser implementada para somente então ser testada e classificada como “Aprovada” ou “Não aprovada”.

As tarefas de cadastrar, revisar, implementar e testar as solicitações, somente devem ser feitas por funcionários que possuem tais funções.

Também é dever do SEPA, controlar quais os sistemas envolvidos na solicitação. Cada cliente pode ter um ou mais sistemas e cada um desses sistemas deve possuir um funcionário responsável por ele, onde serão centralizadas as solicitações.

3.2 ESPECIFICAÇÃO

A especificação do aplicativo foi realizada utilizando a modelagem *Unified Modeling Language* (UML). Segundo Furlan (1998), a UML é a padronização da linguagem de

desenvolvimento orientado a objetos para visualização, especificação, construção e documentação de sistemas.

Para a modelagem do sistema foram utilizados o diagrama de casos de uso, o diagrama de classes e o diagrama de seqüência, onde os mesmos foram feitos utilizando a ferramenta *Rational Rose 2000* da empresa *Rational Software Corporation*. Já o modelo físico da base de dados foi desenvolvido utilizando a ferramenta *PowerDesigner 9* da empresa *Sybase*.

3.2.1 DIAGRAMA DE CASOS DE USO

Nesta aplicação existem os casos de uso apresentados na Figura 6.

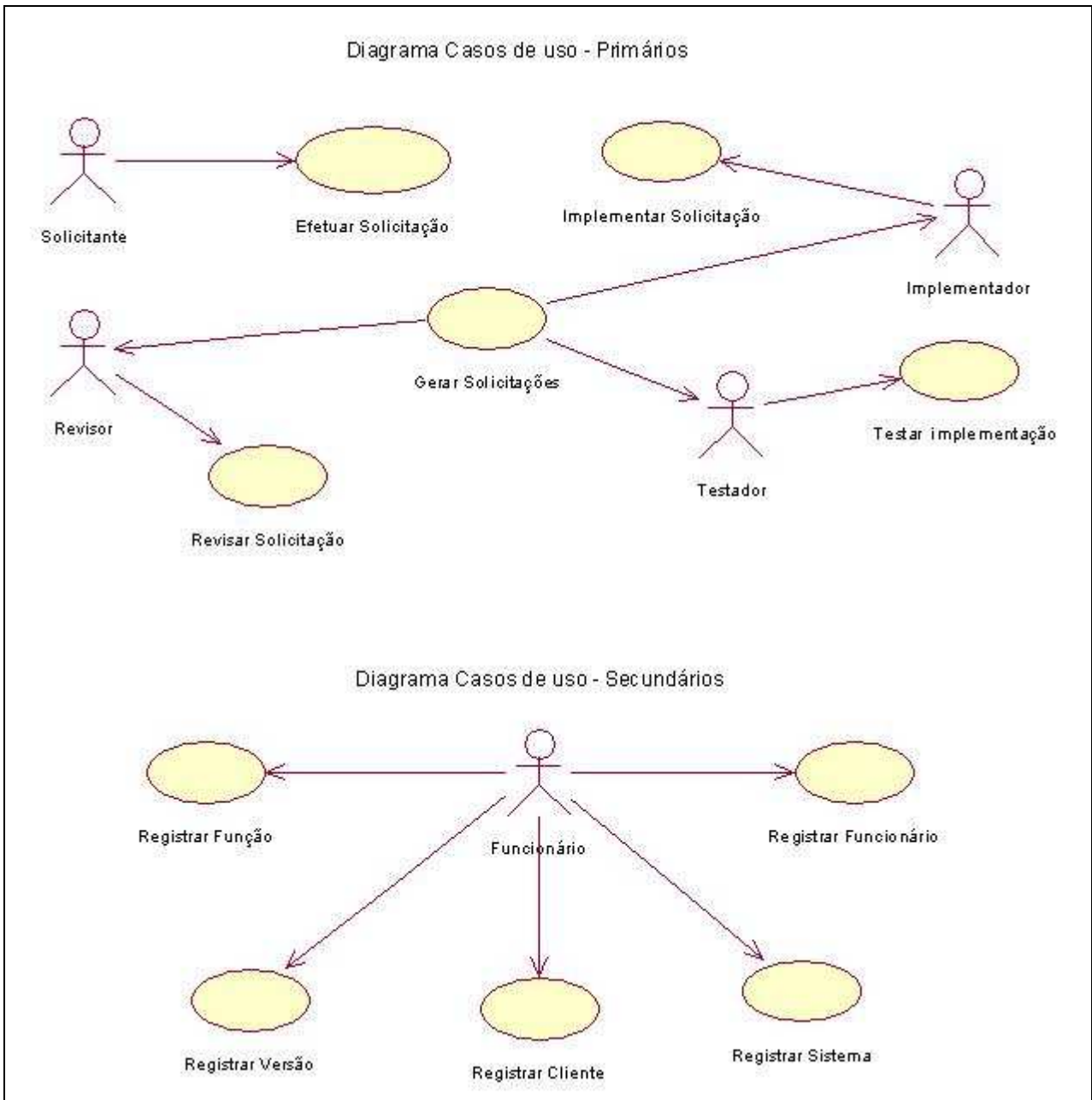


Figura 6 – Diagrama de casos de uso

O Quadro 1 descreve as funcionalidades de cada um dos casos de uso apresentados na Figura 6, agrupando por atores, seguidos de duas colunas, uma com o nome do caso de uso e a outra com um breve descritivo.

ATOR: SOLICITANTE	
Efetuar Solicitação	Uma solicitação é feita para programação e análise. Neste momento são informados os seguintes dados: o cadastrante, a data do cadastro, se a solicitação é interna ou externa, nome da opção e a descrição da

	solicitação, bem como todos os solicitantes e os sistemas atingidos. Estes solicitantes podem ser, tanto clientes quanto funcionários.
ATOR: REVISOR	
Revisar Solicitação	O Revisor verifica se a solicitação é viável e se ela já não é atendida de alguma forma pelo sistema. Ao validar a solicitação, o revisor informa a data da revisão, a versão que ela será implementada bem como quem será responsável pela análise e pela implementação.
Gerar Solicitações	O sistema permite visualizar as solicitações podendo filtrar por: número da solicitação, cadastrante, data do cadastro, revisor, data de revisão, versão, analista, data da análise, implementador, data da implementação, testador e pela data do teste.
ATOR: IMPLEMENTADOR	
Implementar Solicitação	O implementador informa a data da implementação e comentários para os testes, caso seja necessário.
Gerar Solicitações	Idem Gerar Solicitações do ator Revisor.
ATOR: TESTADOR	
Testar Implementação	A solicitação que foi implementada é testada sendo informado quem realizou o teste, o dia que foi testado e se foi aprovada ou não.
Gerar Solicitações	Idem Gerar Solicitações do ator Revisor.
ATOR: FUNCIONÁRIO	
Registrar Função	O funcionário registra a função.
Registrar Versão	O funcionário registra a versão.
Registrar Sistema	O funcionário registra o sistema, informando o responsável pelo mesmo.
Registrar Cliente	O funcionário registra o cliente e relaciona um responsável pelo sistema.
Registrar Funcionário	O funcionário registra um funcionário e informa as funções do mesmo.

Quadro 1 – Apresentação dos casos de uso

3.2.2 DIAGRAMA DE CLASSES

A Figura 7 exibe as classes da aplicação juntamente com os relacionamentos, atributos e métodos.

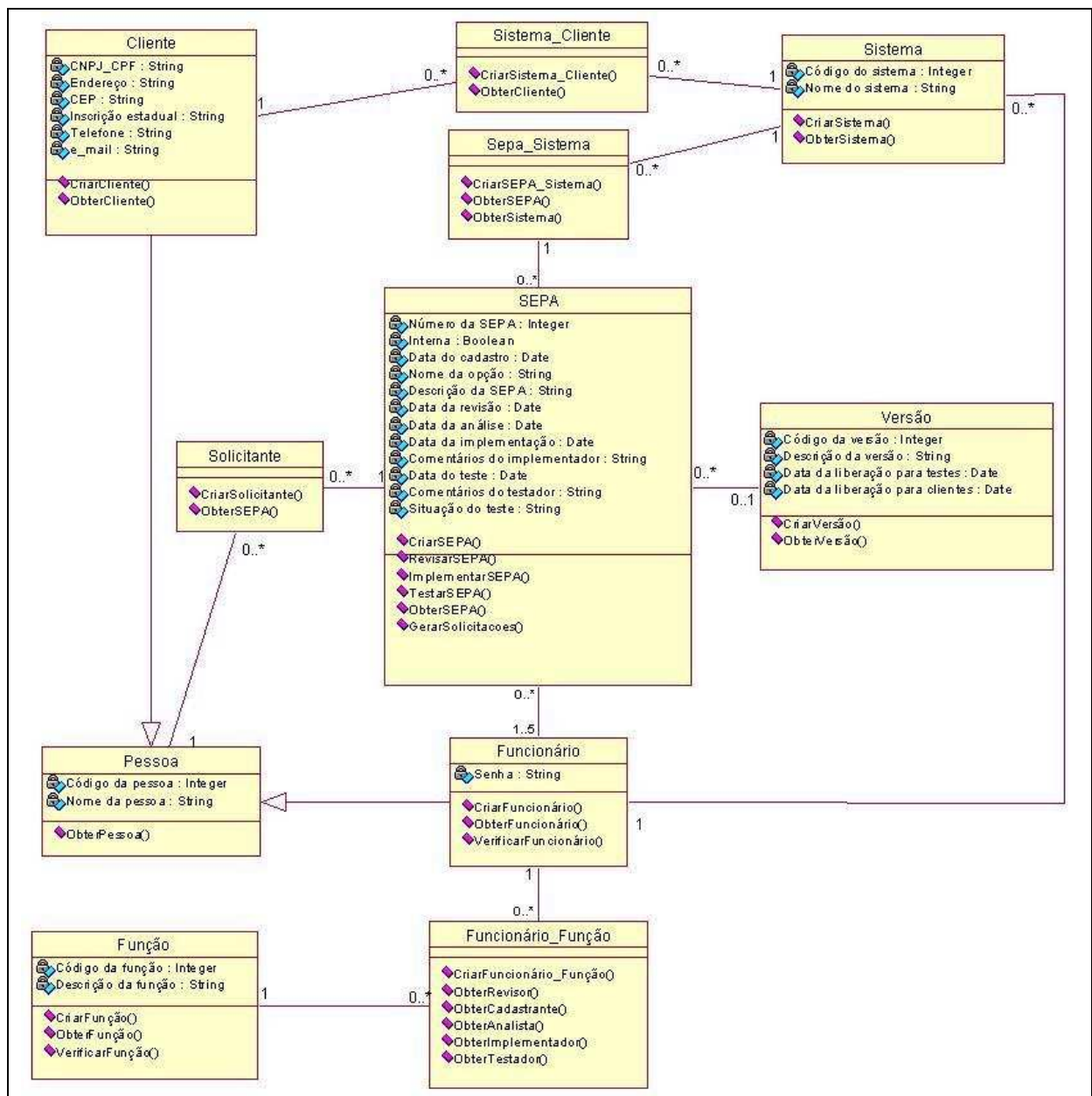


Figura 7 – Diagrama de classes

Abaixo estão relacionadas todas as classes mostradas no diagrama anterior, que são:

- Pessoa: contém as informações comuns entre os clientes e os funcionários;
- Cliente: contém as informações dos clientes;
- Funcionário: contém as informações dos funcionários;
- Função: contém as informações das funções;
- Funcionário_Função: contém as informações das funções do funcionário;
- Sistema: contém as informações dos sistemas;
- Solicitante: contém as informações das pessoas que fizeram a solicitação;
- Sistema_Cliente: contém as informações dos sistemas do cliente;

- i) SEPA: contém as informações das solicitações;
- j) Sepa_Sistema: contém as informações dos sistemas relacionados à solicitação;
- k) Versão: contém as informações da versão que a solicitação deve ser implementada.

3.2.3 DIAGRAMA DE SEQÜÊNCIA

A seguir são apresentados os diagramas de seqüência para os casos de uso primários apresentados na Figura 6.

O diagrama de seqüência para o caso de uso Efetuar Solicitação pode ser visualizado na Figura 8.

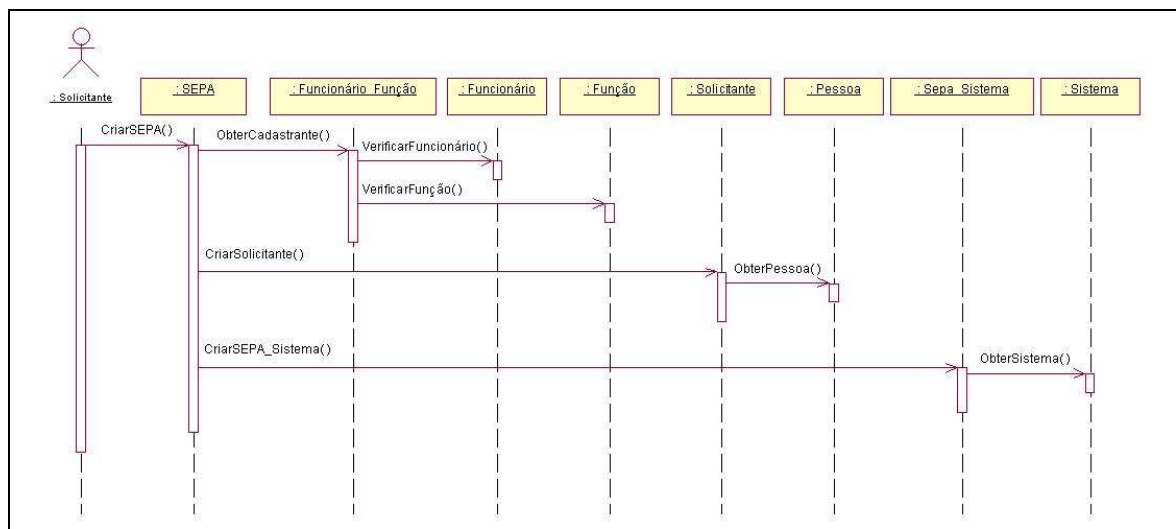


Figura 8 – Diagrama de seqüência – Efetuar Solicitação

Conforme apresentado na Figura 8, a solicitação é criada e é informado o responsável pelo cadastro, sendo, em seguida, relacionados os solicitantes e os sistemas à solicitação.

A Figura 9 mostra o diagrama de seqüência para o caso de uso Revisar Solicitação.

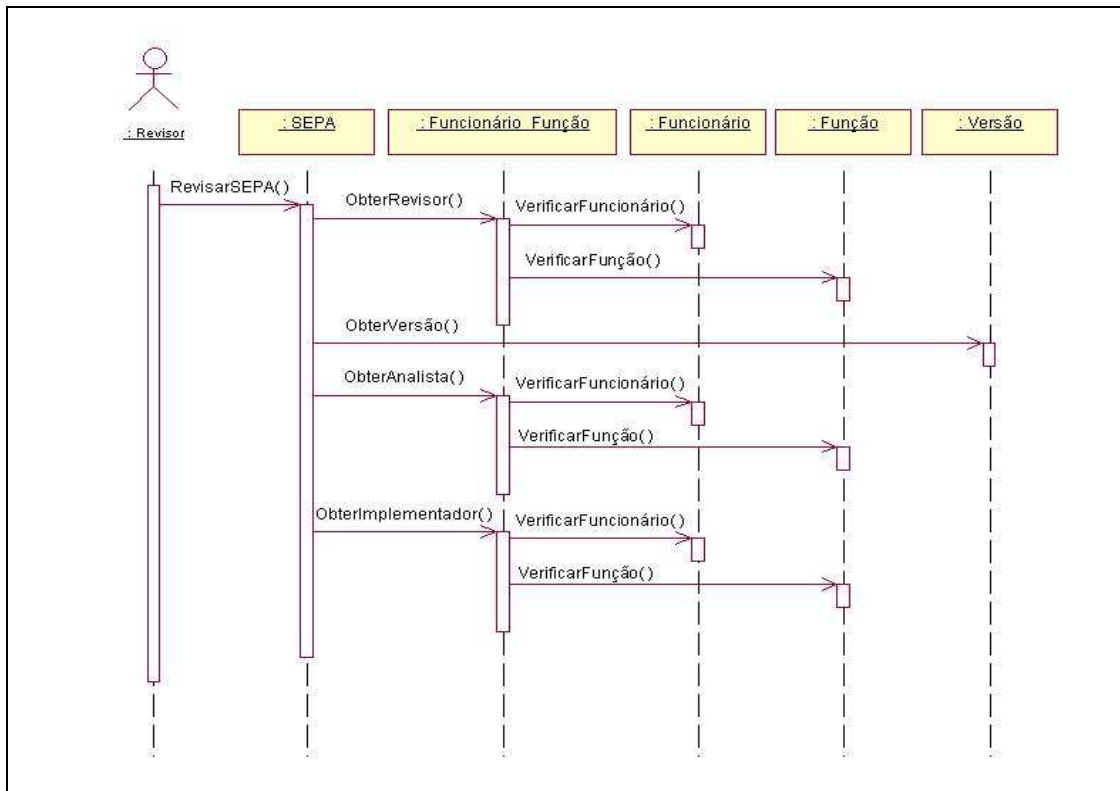


Figura 9 – Diagrama de seqüência – Revisar Solicitação

No diagrama de seqüência exibido na Figura 9, ao revisar a solicitação são informados o revisor, a versão na qual ela deve ser implementada, o analista e o implementador.

A Figura 10 mostra o diagrama de seqüência para o caso de uso Implementar Solicitação.

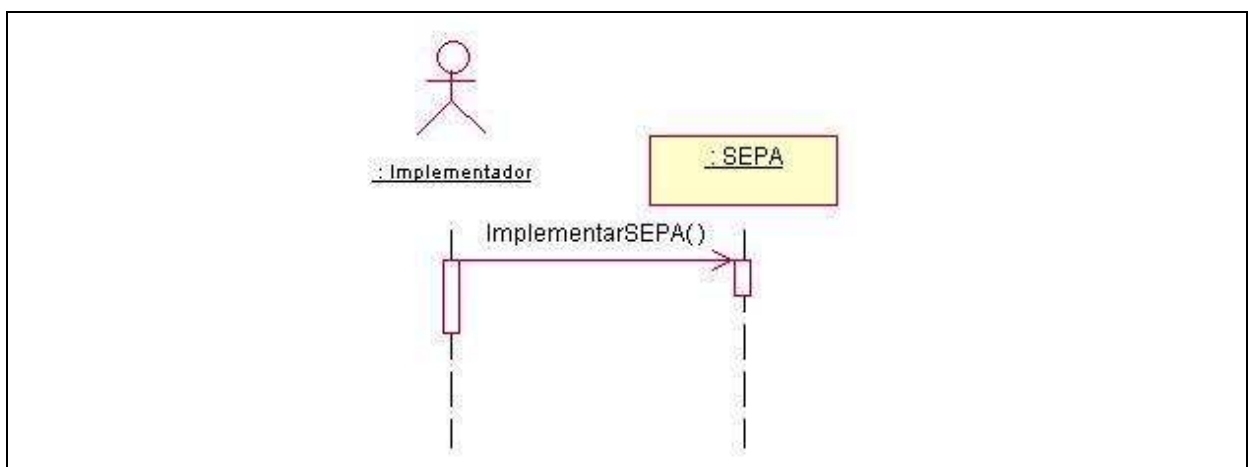


Figura 10 – Diagrama de seqüência – Implementar Solicitação

A Figura 11 mostra o diagrama de seqüência para o caso de uso Testar Implementação.

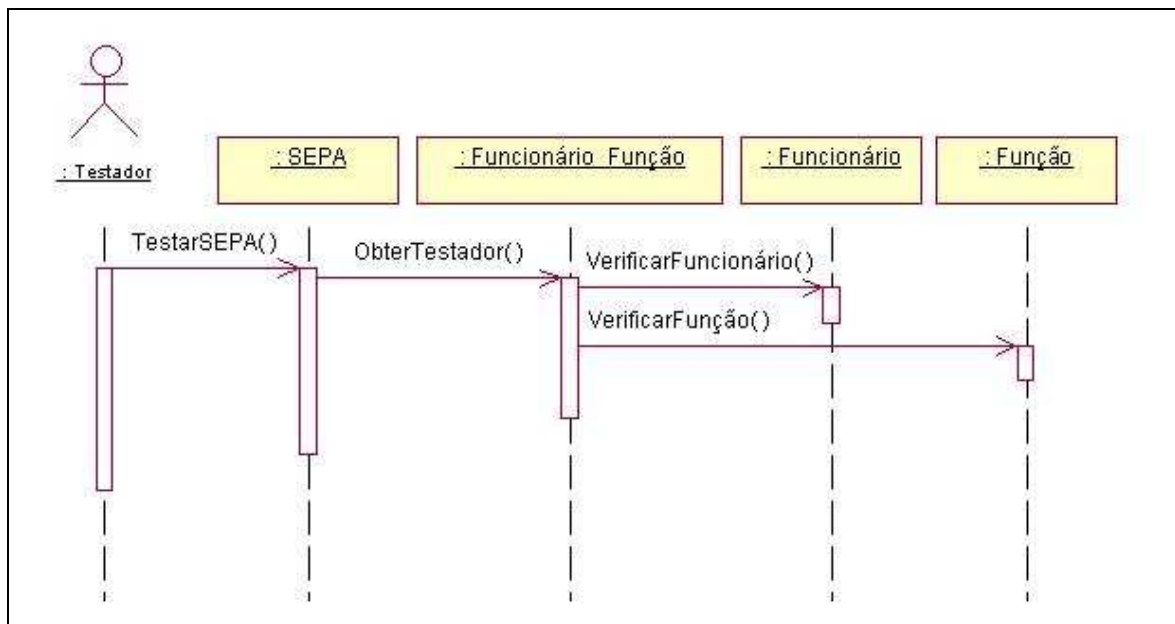


Figura 11 – Diagrama de seqüência – Testar Implementação

Como apresentando na Figura 11 no diagrama de seqüência Testar Implementação é informado o responsável pelo teste.

A Figura 12 mostra o diagrama de seqüência para o caso de uso Gerar Solicitações.

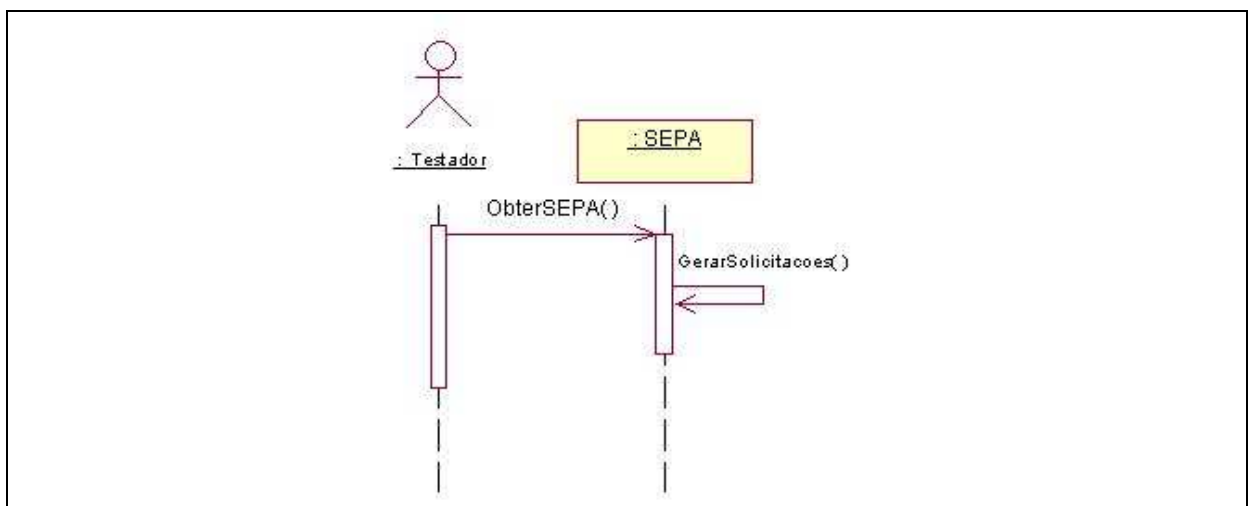


Figura 12 – Diagrama de seqüência – Gerar Solicitações

Para simplificar o diagrama de seqüência Gerar Solicitações, exibido na Figura 12, foi criado o método ObterSEPA(), que resulta em buscar todos os dados relativos à Solicitação. Este mesmo diagrama de seqüência serve para os atores Revisor e Implementador.

3.2.4 MODELO FÍSICO DA BASE DE DADOS

A Figura 13 apresenta o modelo físico da estrutura da base de dados usada na aplicação SEPA.

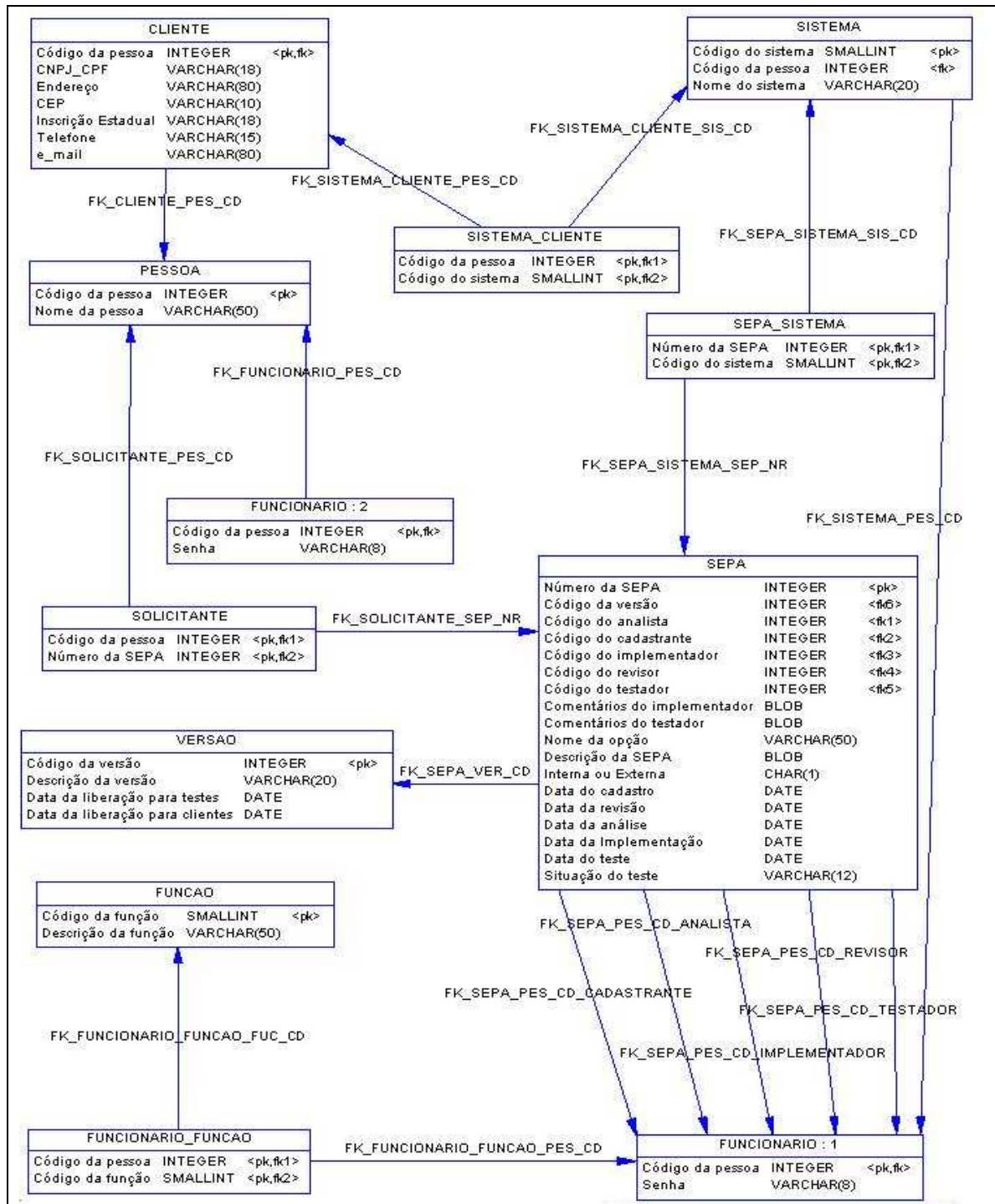


Figura 13 – Modelo Físico da Base de Dados

Conforme exibido na Figura 13 de modo geral, a estrutura do modelo físico é similar a do modelo lógico.

3.2.5 DICIONÁRIO DE DADOS

No Quadro 2 é apresentada a estrutura da base dados do sistema. Na primeira linha do Quadro 2 é exibido o nome físico da tabela seguido de uma breve descrição da mesma. Logo abaixo tem-se 5 colunas: Nome, Código, Tipo, P e M, que respectivamente significam Nome do campo, Código do campo, Tipo do campo, se o campo faz parte ou não da chave primária (P) e se o campo é obrigatório (M).

Tabela: CLIENTE - Clientes que possuem os Sistemas				
Nome	Código	Tipo	P	M
Código da pessoa	PES_CD	INTEGER	Sim	Sim
CEP	CLI_CEP	VARCHAR(10)	Não	Não
CNPJ_CPF	CLI_CNPJ_CPF	VARCHAR(18)	Não	Não
Endereço	CLI_END	VARCHAR(80)	Não	Não
Telefone	CLI_FON	VARCHAR(15)	Não	Não
Inscrição Estadual	CLI_INS_EST	VARCHAR(18)	Não	Não
e_mail	CLI_MAIL	VARCHAR(80)	Não	Não
Tabela: FUNCIONARIO - Funcionários que usam a aplicação				
Nome	Código	Tipo	P	M
Código da pessoa	PES_CD	INTEGER	Sim	Sim
Senha	FUN_PWD	VARCHAR(8)	Não	Sim
Tabela: FUNCIONARIO_FUNCAO - Relação entre Funcionários e Funções				
Nome	Código	Tipo	P	M
Código da pessoa	PES_CD	INTEGER	Sim	Sim
Código da função	FUC_CD	SMALLINT	Sim	Sim
Tabela: FUNCAO – Funções dos Funcionários				
Nome	Código	Tipo	P	M
Código da função	FUC_CD	SMALLINT	Sim	Sim
Descrição da função	FUC_DS	VARCHAR(50)	Não	Sim
Tabela: PESSOA - Dados comuns entre Clientes e Funcionários				
Nome	Código	Tipo	P	M
Código da pessoa	PES_CD	INTEGER	Sim	Sim
Nome da pessoa	PES_NM	VARCHAR(50)	Não	Sim
Tabela: SEPA – Solicitações				
Nome	Código	Tipo	P	M
Número da SEPA	SEP_NR	INTEGER	Sim	Sim
Código do analista	PES_CD_ANALISTA	INTEGER	Não	Não
Código do cadastrante	PES_CD_CADASTRANTE	INTEGER	Não	Sim
Código do implementador	PES_CD_IMPLEMENTADOR	INTEGER	Não	Não
Código do revisor	PES_CD_REVISOR	INTEGER	Não	Não
Código do testador	PES_CD_TESTADOR	INTEGER	Não	Não
Comentários do implementador	SEP_CMT_IMPLEMENTADOR	BLOB	Não	Não
Comentários do testador	SEP_CMT_TESTADOR	BLOB	Não	Não
Descrição da SEPA	SEP_DS	BLOB	Não	Não
Data da análise	SEP_DT_ANA	DATE	Não	Não

Data do cadastro	SEP_DT_CAD	DATE	Não	Não
Data da Implementação	SEP_DT_IMP	DATE	Não	Não
Data da revisão	SEP_DT_REV	DATE	Não	Não
Data do teste	SEP_DT_TST	DATE	Não	Não
Interna ou Externa	SEP_INT	CHAR(1)	Não	Sim
Nome da opção	SEP_NM_OPC	VARCHAR(50)	Não	Sim
Situação do teste	SEP_SIT_TST	VARCHAR(12)	Não	Sim
Código da versão	VER_CD	INTEGER	Não	Não
Tabela: SEPA_SISTEMA – Relação entre Solicitações e Sistemas				
Nome	Código	Tipo	P	M
Número da SEPA	SEP_NR	INTEGER	Sim	Sim
Código do sistema	SIS_CD	SMALLINT	Sim	Sim
Tabela: SISTEMA - Sistemas que são controlados as Solicitações				
Nome	Código	Tipo	P	M
Código do sistema	SIS_CD	SMALLINT	Sim	Sim
Código da pessoa	PES_CD	INTEGER	Não	Sim
Nome do sistema	SIS_NM	VARCHAR(20)	Não	Sim
Tabela: SISTEMA_CLIENTE - Relação entre os Clientes e Sistemas				
Nome	Código	Tipo	P	M
Código da pessoa	PES_CD	INTEGER	Sim	Sim
Código do sistema	SIS_CD	SMALLINT	Sim	Sim
Tabela: SOLICITANTE - Relação entre a Solicitação e os Solicitantes				
Nome	Código	Tipo	P	M
Número da SEPA	SEP_NR	INTEGER	Sim	Sim
Código da pessoa	PES_CD	INTEGER	Sim	Sim
Tabela: VERSAO - Versões dos Sistemas				
Nome	Código	Tipo	P	M
Código da versão	VER_CD	INTEGER	Sim	Sim
Descrição da versão	VER_DS	VARCHAR(20)	Não	Sim
Data da liberação para clientes	VER_DT_LIB_CLI	DATE	Não	Sim
Data da liberação para testes	VER_DT_LIB_TST	DATE	Não	Sim

Quadro 2 – Estrutura da base de dados

3.3 IMPLEMENTAÇÃO

Esta seção apresenta considerações sobre a implementação do protótipo, tais como as técnicas e ferramentas utilizadas e a operacionalidade da implementação.

3.3.1 TÉCNICAS E FERRAMENTAS UTILIZADAS

Para o desenvolvimento do sistema foi utilizada a linguagem *Object Pascal*, com o ambiente de desenvolvimento Borland Delphi 7 *Enterprise*. O sistema foi inteiramente desenvolvido utilizando os componentes e classes da VCL do Delphi. Para gerar relatório foi utilizada a ferramenta Rave Reports 5.0 da empresa Nevrona Designs, inclusa no CD de instalação do Delphi. O banco de dados utilizado foi o Interbase acessado por meio dos componentes InterBase Express (IBX).

As telas de cadastro foram feitas a partir de telas padrão, aumentando assim a produtividade no desenvolvimento e mantendo a padronização do sistema.

O sistema SEPA foi migrado para a plataforma .NET da Microsoft utilizando o ambiente Borland Delphi 8.0 versão *Architect*. A VCL, utilizada antes, foi substituída pela VCL.NET e a ferramenta de geração de relatórios utilizada foi a versão 5.5 do Rave Reports. O banco de dados e o acesso a ele continuaram os mesmos, só que na versão para .NET.

3.3.2 OPERACIONALIDADE DA IMPLEMENTAÇÃO

Para a validação deste trabalho, foi desenvolvida uma aplicação utilizando os conceitos e técnicas apresentadas. A seguir é apresentado um estudo de caso com a finalidade de apresentar as funcionalidades da aplicação.

As telas de cadastro da aplicação SEPA seguem um padrão, e estão separadas em dois tipos: tela de cadastro (simples) e tela de cadastro (mestre/detalhe).

Como exemplo de uma tela de cadastro simples pode-se citar a tela de Cadastro de sistemas, apresentada na Figura 144.

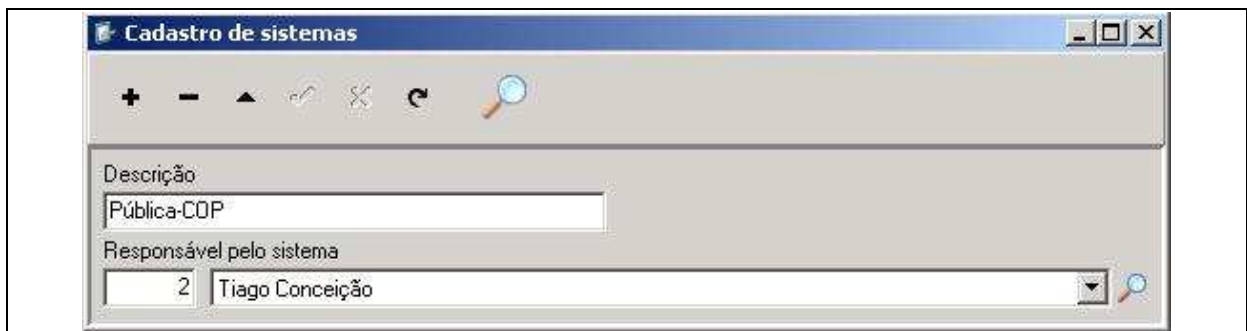









Figura 14 – Exemplo da tela de cadastro simples (Cadastro de sistemas)

A Tabela 1 explica as funcionalidades da tela de Cadastro de sistemas exibida na Figura 14.

Tabela 1 – Funcionalidades da tela de cadastro normal

Função	Descrição
	Inclui um registro.
	Apaga o registro atual.

	Altera o registro atual.
	Confirma a inclusão/alteração atual.
	Cancela a inclusão/alteração atual.
	Busca os dados da base de dados.
	Abre uma tela de pesquisa.

Ao abrir uma tela de cadastro, a mesma aparece sem nenhum registro, forçando o usuário a clicar no botão de pesquisa para selecionar um registro antes de alterá-lo ou apagá-lo. Só é possível alterar o registro exibido na tela, depois de clicar no botão “Alterar registro”.

Um exemplo de tela de cadastro mestre/detalhe é demonstrado através do “Cadastro de clientes/Sistemas do cliente” como é apresentado na Figura 15, tendo como principal diferença a divisão, feita por fichas, dos dados do cadastro mestre com os dados do cadastro detalhe, onde na ficha detalhe, os dados são exibidos em uma *Grid*, dispensando o uso do botão de pesquisa para selecionar um registro para alteração ou exclusão.



Figura 15 – Exemplo da tela de cadastro mestre/detalhe (Cadastro de clientes/Sistemas do cliente)

A Figura 16 apresenta a tela principal do sistema, onde suas principais funções podem ser acessadas através do menu ou através da barra de ferramentas localizada na parte superior esquerda da tela.

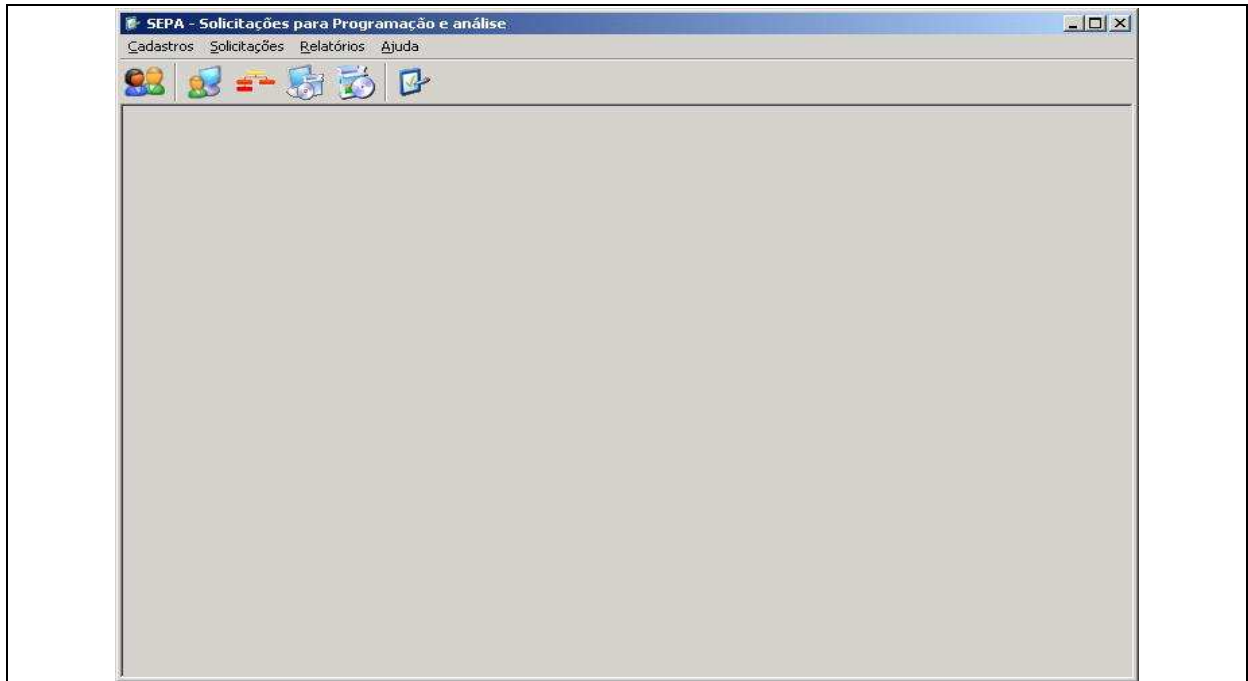


Figura 16 – Tela principal da aplicação

Com base nos padrões de telas de cadastros apresentados anteriormente, têm-se os controles das solicitações (feito em uma tela do tipo mestre/detalhe), dividida em etapas na seguinte ordem: cadastro, revisão, implementação e, por último, os testes.

Cada uma dessas etapas pelas quais a Solicitação passa, será mostrada separadamente, começando pela etapa de cadastro da solicitação apresentada pela Figura 17.

The screenshot shows a window titled 'Solicitações' with a toolbar containing icons for adding, deleting, undo, redo, and search. Below the toolbar are three tabs: 'Solicitação', 'Solicitantes', and 'Sistemas'. The 'Solicitação' tab is active and contains the following fields:

- NÚMERO:** A text box containing the value '41'.
- Interna:** A checked checkbox.
- Cadastrante:** A dropdown menu showing '9' and a text box containing 'Mônica'.
- Data cadastro:** A date field showing '11/11/2004' and a small calendar icon.

Below these fields are three more tabs: 'Conteúdo', 'Revisão', 'Implementador', and 'Testes'. The 'Conteúdo' tab is active and contains:

- Nome da opção:** A text box containing 'Rotinas de final de ano'.
- Descrição:** A large text area containing 'Fazer a inicialização do próximo exercício.'

Figura 17 – Tela de cadastro de solicitações

A Figura 17 mostra que ao incluir uma solicitação, o sistema irá gerar um número usado como identificador da solicitação. Para concluir a inclusão, é necessário informar se a solicitação é interna ou não, quem cadastrou e quando ela foi cadastrada, o nome da opção e a descrição da solicitação, bem como as pessoas que solicitaram e para quais sistemas a solicitação foi feita.

Na Figura 18 é possível visualizar a tela para informar as pessoas que fizeram a solicitação. Estas pessoas podem ser tanto um cliente como um funcionário.

The screenshot shows the same 'Solicitações' window, but with the 'Solicitantes' tab selected. The 'Solicitação' tab is still visible at the top. The 'Solicitantes' tab displays a list of names in a table-like structure:

Solicitante
Airison
Mônica
Rodrigo
Tiago Conceição

The 'Mônica' row is currently selected, highlighted in blue. A search icon is visible to the right of the list.

Figura 18 – Tela para relacionar os solicitantes com a solicitação

Para completar a inclusão da solicitação, é necessário informar quais sistemas estão relacionados à solicitação, conforme é mostrado na Figura 19.

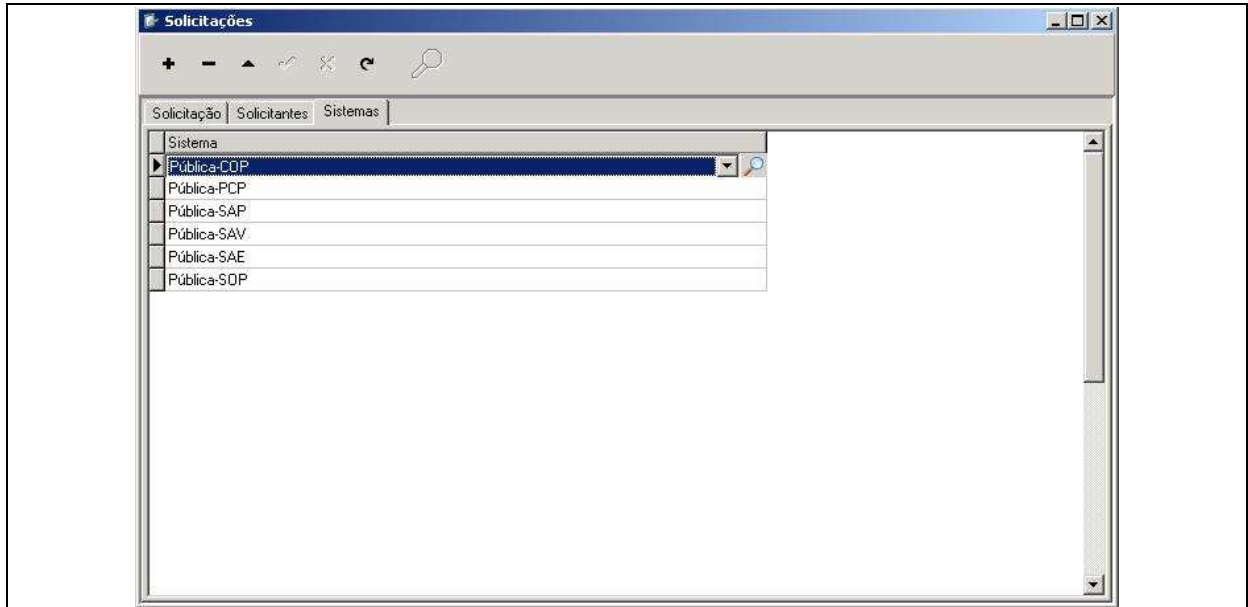


Figura 19 – Tela para relacionar os sistemas à solicitação

Depois de concluído o cadastro, a solicitação passa para a fase de revisão, conforme mostra a Figura 20.

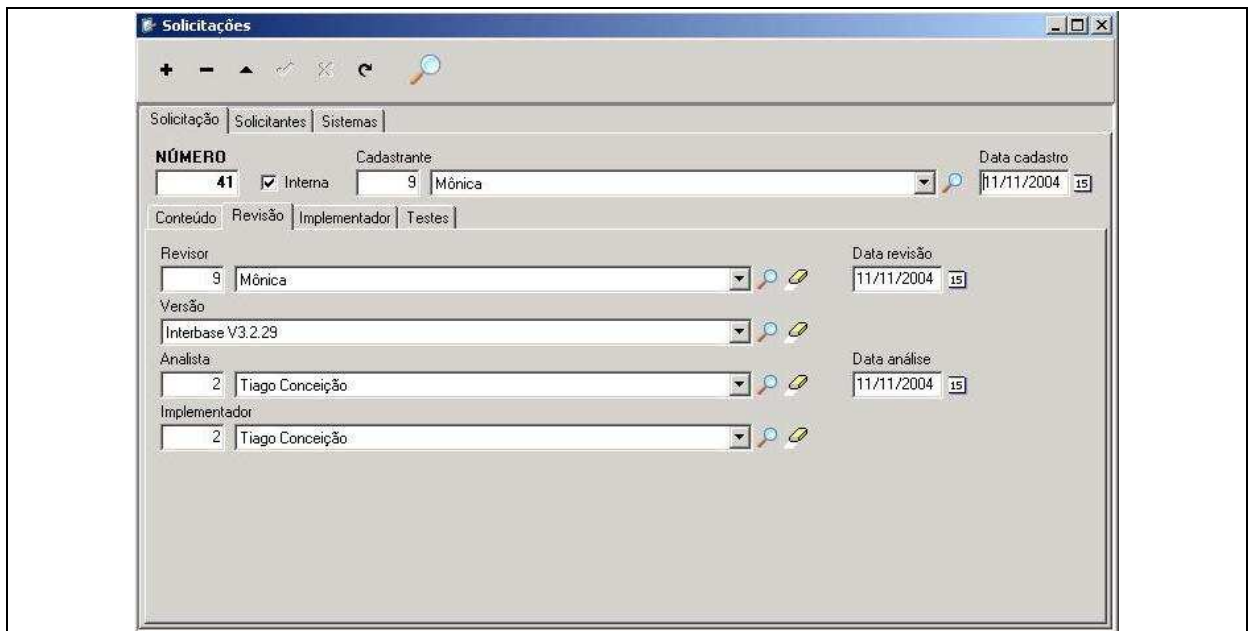
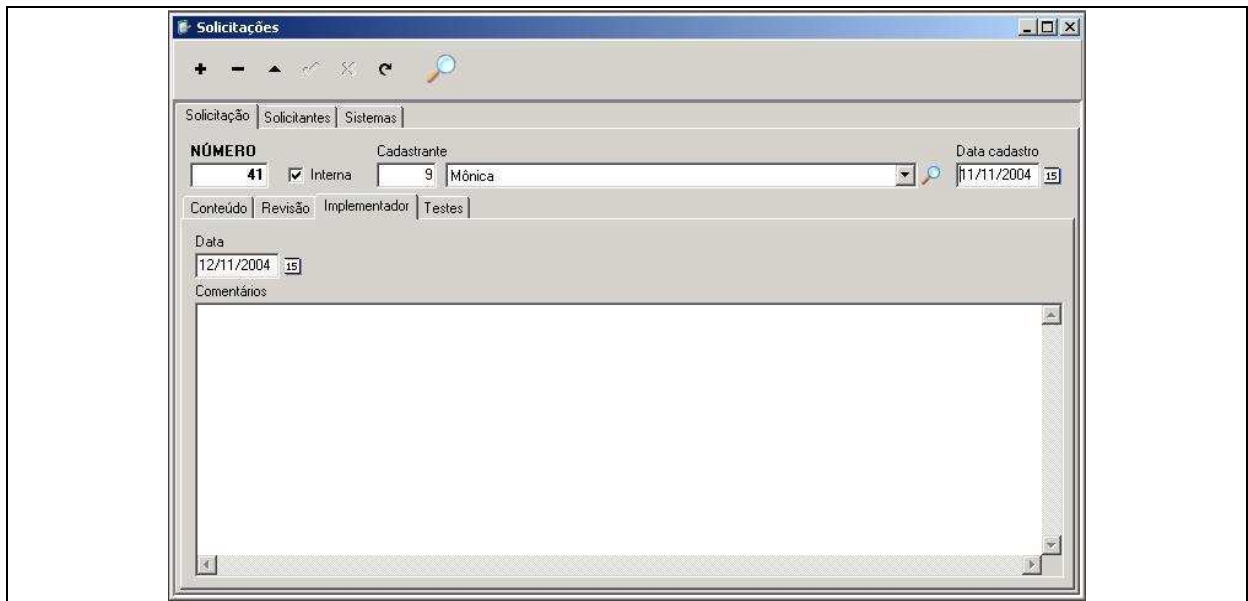


Figura 20 – Tela de revisão da solicitação

Nesta etapa de revisão mostrada na Figura 20, deve-se informar quando e quem revisou e analisou a solicitação, bem como quem irá implementar, e em qual versão será atendida a solicitação. Somente depois desta etapa a solicitação pode ser implementada.

A Figura 21 exibe a tela da etapa de implementação.



A imagem mostra uma janela de software intitulada "Solicitações". No topo, há uma barra de ferramentas com ícones para adicionar, remover, expandir, fechar, atualizar e pesquisar. Abaixo, há uma barra de abas com "Solicitação", "Solicitantes" e "Sistemas". O formulário principal contém os seguintes campos:

- NÚMERO:** Campo de texto com o valor "41".
- Cadastrante:** Campo de texto com o valor "9" e "Mônica".
- Data cadastro:** Campo de data com o valor "11/11/2004".
- Conteúdo:** Aba selecionada.
- Revisão:** Aba não selecionada.
- Implementador:** Aba não selecionada.
- Testes:** Aba não selecionada.
- Data:** Campo de data com o valor "12/11/2004".
- Comentários:** Área de texto grande e vazia para inserir comentários.

Figura 21 – Tela de implementação da solicitação

Nesta etapa de implementação, exibida na Figura 21, o implementador deve informar a data em que a solicitação foi implementada e um comentário caso seja necessário.

Como última etapa do controle de solicitações para programadores e analistas, têm-se as informações dos testes, como exibido na Figura 22.

Figura 22 – Tela de informações dos testes

Esta última etapa, apresentada na Figura 22, consiste em informar quem foi responsável pelo teste, a data em que foi testada, a situação do teste (Aprovada ou Não aprovada) e algum comentário sobre o teste (caso necessário).

3.4 RESULTADOS E DISCUSSÃO

Os resultados e discussões acerca do trabalho desenvolvido estão divididos em duas partes: uma que define e avalia o processo de migração e outra que segue a mesma abordagem para a comparação da performance do sistema nas duas plataformas.

3.4.1 MIGRAÇÃO

Para facilitar o processo de migração, o Delphi 7 dispõe de um recurso que auxilia na sua compatibilidade com o Delphi 8. Este recurso é disponibilizado através da ativação de avisos nas mensagens de compilação, que podem ser ativadas na ficha de "Compiler Messages" na tela de "Project Options" do projeto, conforme exibido na Figura 23.

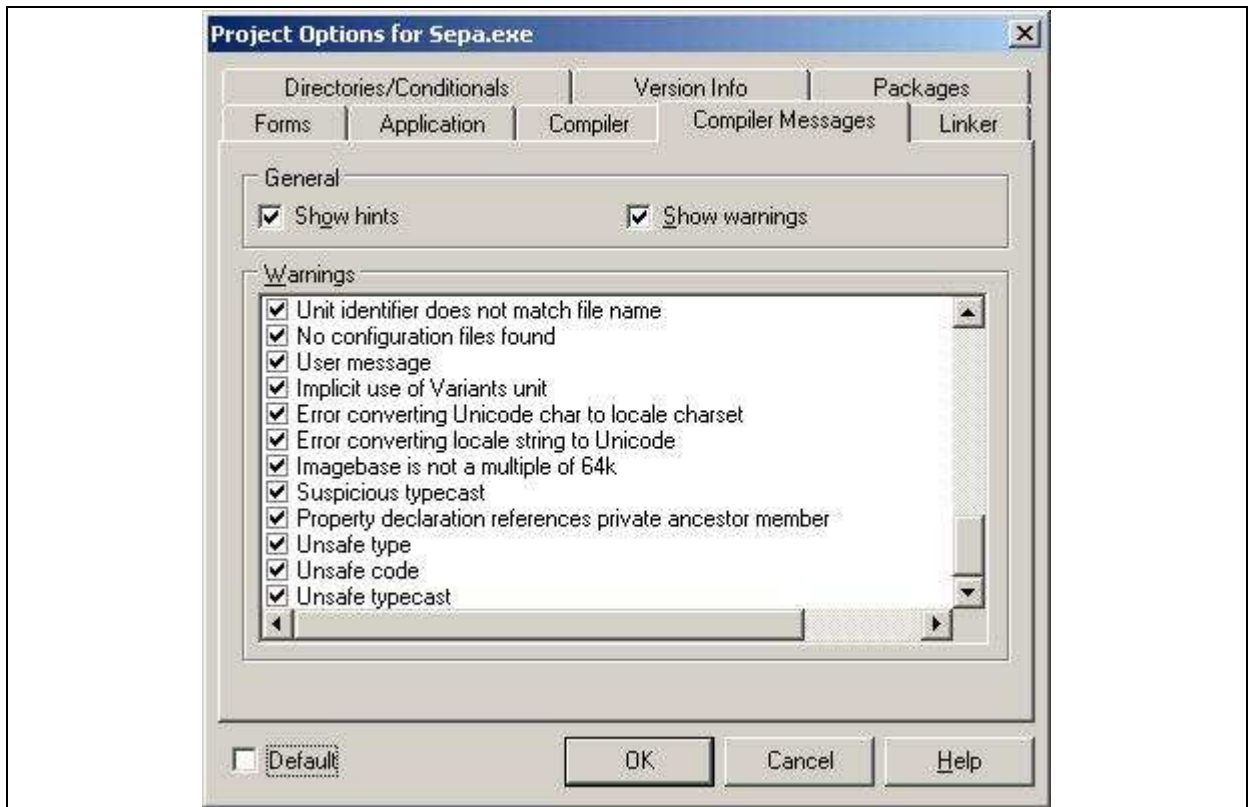


Figura 23 – Tela opções do projeto (*Project Options*)

A Figura 24 exibe quatro mensagens de aviso na compilação do sistema SEPA no Delphi 7.

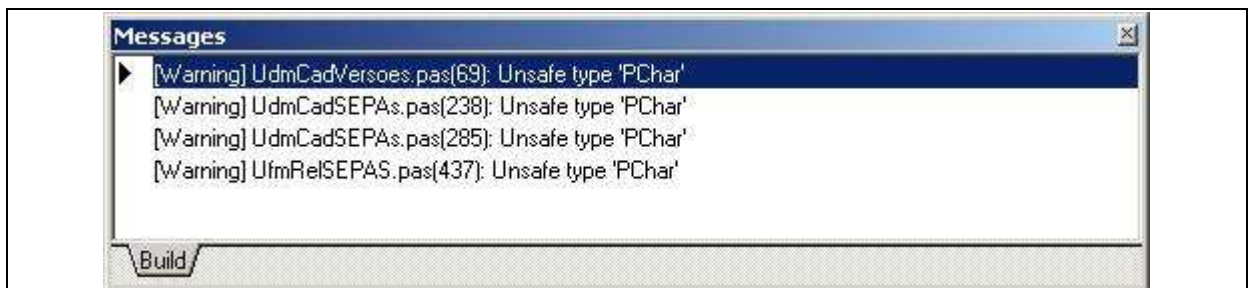


Figura 24 – Tela de mensagens de avisos

O código fonte que gerou esses avisos exibidos na Figura 24 deve ser revisto no Delphi.NET, pois no Delphi.NET estes avisos tornam-se erros de compilação, devido ao fato da CLR não compilar código inseguro. Tendo o conhecimento desses avisos, o Sistema SEPA foi desenvolvido com o máximo de cuidado para evitá-los. Desta forma, a aplicação passou a ser desenvolvida havendo a portabilidade entre as duas versões do Delphi.

Para manter exatamente o mesmo código fonte compilando tanto no Delphi 7 como no Delphi 8, foi feito o uso de diretivas de compilação. O Quadro 3 apresenta um trecho do código onde houve a necessidade do uso das diretivas.

```
//verifica se a data de implementação é menor q a data de revisão
else if (dstSepasSEP_DT_IMP.AsDateTime < dstSepasSEP_DT_REV.AsDateTime) then
begin
  {$IFDEF D7}
  Application.MessageBox(PChar('A data da implementação (' +
    dstSepasSEP_DT_IMP.AsString + ') não pode ser ' +
    'menor que a data de revisão (' +
    dstSepasSEP_DT_REV.AsString + ') da SEPA.'),
    'Informação', MB_ICONINFORMATION);
  {$ENDIF}
  {$IFDEF D8}
  Application.MessageBox('A data da implementação (' +
    dstSepasSEP_DT_IMP.AsString + ') não pode ser ' +
    'menor que a data de revisão (' +
    dstSepasSEP_DT_REV.AsString + ') da SEPA.',
    'Informação', MB_ICONINFORMATION);
  {$ENDIF}

  fmCadSEPAs.pcOpcoes.ActivePage := fmCadSEPAs.tsImplementador;
  fmCadSEPAs.dbedDataImplementacao.SetFocus;
  SysUtils.Abort;
end;
```

Quadro 3 – Diretivas de compilação

Fazendo o uso das diretivas de compilação, exibidas no Quadro 3, é possível manter o código fonte compilando 100% nas duas versões do Delphi. Deve-se observar no código do Delphi 7 (diretiva {\$IFDEF D7}) é feito um *typecast* utilizando o tipo PChar. No Delphi 8 (diretiva {\$IFDEF D8}) este código deve ser suprimido pois o tipo PChar é considerado um tipo inseguro pela CLR.

Após ter concluído o desenvolvimento do Sistema SEPA no Delphi 7, tem-se início a fase de migração deste trabalho. Com os cuidados tomados na implementação da aplicação Win32 este processo tornou-se bem simples, pois basta abrir a aplicação desenvolvida no Delphi 7, dentro do ambiente do Delphi 8. Para realizar essa tarefa, clica-se no menu “File” e em seguida “Open Project” e deve ser selecionado o arquivo “Sepa.dpr” no diretório onde estão os fontes do sistema.

Depois de aberto o projeto, o mesmo deve ser compilado através da opção do menu “Project | Compile Sepa”. Ao compilar o projeto, surgiram algumas mensagens de erros de compilação, conforme exibido na Figura 25.

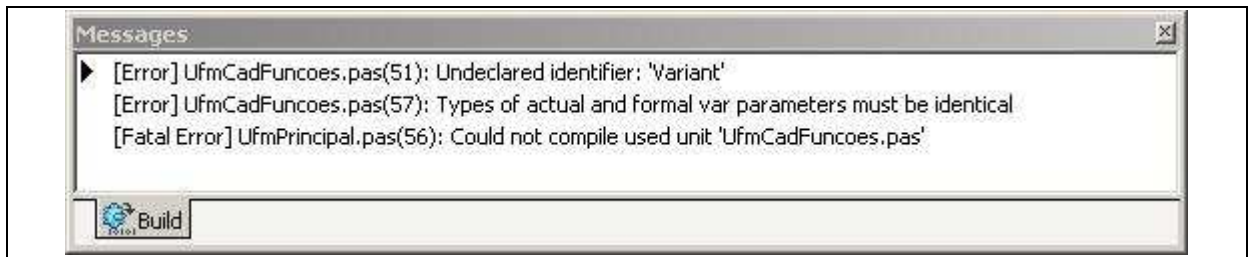


Figura 25 – Mensagens de erro da compilação

Para solucionar o erro de compilação “*Undeclared identifier: Variant*” exibido na Figura 25, basta acrescentar a *unit* “*Variants*” na clausula *uses* da *unit* onde ocorre o erro. Essa alteração deve ser feita em todas as *units* onde ocorrer essa mensagem de erro, uma vez que o tipo *variant* foi inserido nesta nova *unit* no Delphi 8.

Depois de feita a alteração anterior, o projeto deve ser compilado novamente. Nesse momento, o Delphi exibirá um erro de arquivo não encontrado, conforme mostra a Figura 26.

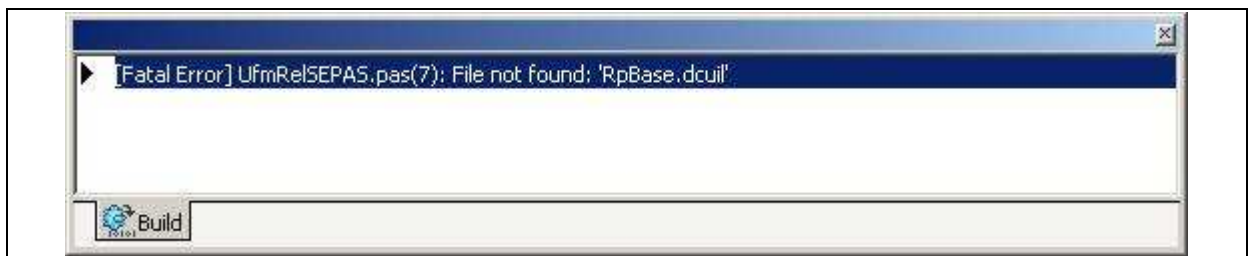


Figura 26 – Erro de arquivo não encontrado

O erro exibido na Figura 26, ocorre devido ao fato do Delphi 8 trabalhar com *namespace* e não ter sido informado o prefixo do *namespace* da biblioteca Rave. Como solução para esse erro, deve-se acessar o menu “*Project/Options*” e em seguida “*Directories/Conditionals*” e informar o prefixo “*Nevrona.Rave*” conforme mostra a Figura 27.

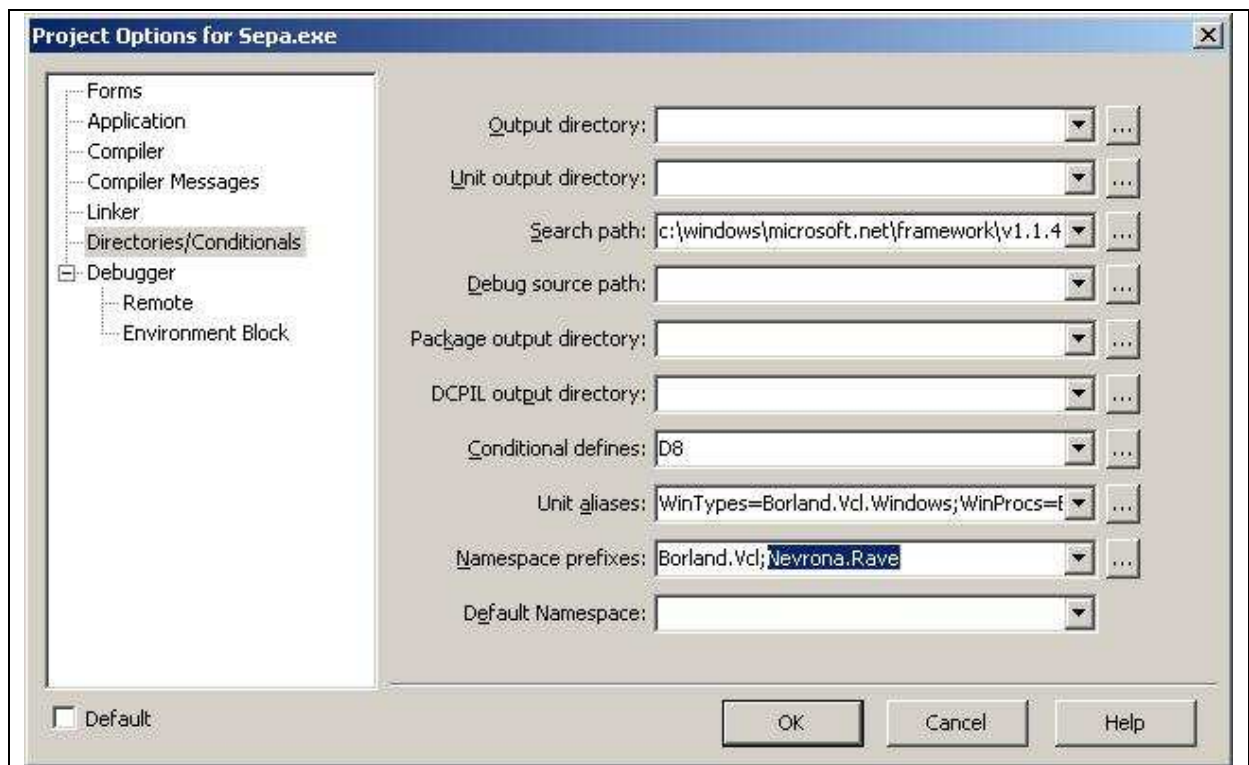


Figura 27 – Informando o *Namespace* do *Rave*

Depois de informado o *namespace*, o projeto deve ser compilado novamente.

A etapa de compilação foi concluída de forma relativamente simples. É importante destacar, entretanto, que alguns procedimentos exigem que alguns erros sejam estudados, ou seja, o processo não se dá por simplesmente abrir o sistema no Delphi 8, compilar e executar diretamente.

Assim, após a solução de todos os erros de compilação pôde-se executar o programa na nova plataforma. Para realizar essa tarefa, basta selecionar o menu “*Run/Run*” ou pressionar a tecla F9.

Ao executar a aplicação foram encontrados vários erros de exceção no sistema. Entretanto esses erros foram solucionados no *General Update 2* do Delphi 8, ou seja não foram conseqüências do processo de migração. Esses erros trouxeram muitas dificuldades na fase de validação do sistema no Delphi 8, implicando em um estudo mais detalhado sobre a ocorrência dos mesmos.

Como resumo do processo de migração foram seguidos os passos:

- a) compilar a aplicação SEPA no Delphi 7 com o uso de diretivas de compilação evitando os avisos (*warnings*) nas mensagens de compilação ativos e evitando a existência de ;
- b) abrir o projeto no Delphi 8;
- c) compilar o projeto;
- d) corrigir os erros de compilação;
- e) repetir os passos “c” e “d” até que não existam mais erros de compilação.

Esta estratégia não é a única para processos de migração de aplicações Win32 para .NET, mas demonstrou-se muito adequada pois permitiu, sobretudo manter a portabilidade do código das duas aplicações.

Depois de verificados e estudados todos os problemas de execução e realizadas as devidas correções, passou-se para a etapa de comparação de performance entre as aplicações, a qual será descrita na próxima seção.

3.4.2 COMPARATIVO DE PERFORMANCE

Para o processo de avaliação foi consultada a norma NBR13596 (ABNT, 1996), a qual define características para avaliação de softwares em relação à funcionalidade, confiabilidade, usabilidade, performance, manutenibilidade e portabilidade. Entretanto, como o foco não é a avaliação do sistema desenvolvido e sim sua comparação em duas plataformas de desenvolvimento diferentes, optou-se por utilizar apenas o critério relacionado à performance do mesmo, destacando a sub-característica denominada de tempo de resposta, a qual avalia a velocidade de execução do sistema.

As opções do Sistema SEPA selecionadas para serem avaliadas neste comparativo de performance foram definidas de forma que a ação do avaliador não interferisse nos resultados, ou seja, o roteiro de testes está ligado ao tempo de resposta dado pelo computador para a tarefa solicitada pelo usuário.

Assim, foram definidos 8 pontos críticos para serem usados no comparativo. A Tabela 2 exibe o roteiro seguido no processo de avaliação. Este roteiro é composto por 2 colunas, sendo que a primeira significa a ordem em que deve-se executar as ações e a segunda coluna refere-se ao nome da ação que será gravada no arquivo de log.

Tabela 2 – Roteiro da avaliação

Ordem	Nome da ação
01	Solicitações – Abrir tela
02	Solicitações – Localizar 200.000
03	Cientes – Abrir tela
04	Cientes – Localizar 100.000
05	Funcionários – Abrir tela
06	Funcionários – Localizar 200.000
07	Relatórios – Abrir tela
08	Relatórios – Montagem do SQL

Devido ao fato de se ter acesso ao código-fonte dos dois sistemas, a avaliação foi feita através da inclusão de uma função antes e depois da opção que se desejava avaliar o tempo de resposta. O nome da função utilizada é “*GetTickCount*”, que retorna o número de milissegundos transcorridos desde a inicialização do Windows. Para cada opção avaliada, foi dado um nome para identificá-la. No arquivo de performance foi gravado o nome da opção seguido do tempo de resposta (em milissegundos) que a opção demorou a executar. A base de dados utilizada foi a mesma para os dois sistemas.

A Tabela 3 apresenta a quantidade de registros por tabela do banco de dados utilizado no comparativo.

Tabela 3 – Quantidade de registros das tabelas

Nome da tabela	Quantidade de registros
Cientes	100.000
Funcionários	100.000
Funcionarios_Funcoes	200.000
Funções	5
Pessoas	200.000
Sepas	200.000
Sepas_Sistemas	200.000
Sistemas	6
Sistemas_Cientes	200.000
Solicitantes	200.000
Versões	201
Total de registros	1.400.212

As características do computador usado para executar essa avaliação são exibidas na Tabela 4.

Tabela 4 – Características do computador

Hardware/Software	Especificação
Placa mãe/Barramento interno	ECS-K7S5A / 266MHz
Processador	1.3 GHz
Memória RAM	512 MB DDR

HD	20 GB
Placa de vídeo	64 MB
Sistema operacional	Microsoft Windows XP Professional
Plataforma .NET	Microsoft .NET Framework 1.1

Os sistemas foram avaliados considerando a mesma quantidade de recursos disponíveis.

Para garantir uma maior igualdade nas condições das avaliações, foram seguidos alguns passos. Primeiramente foi inicializado o computador e o sistema operacional e após isso, foi inicializado o serviço de banco de dados do *Interbase* para só então executar o Sistema SEPA e seguir o roteiro de avaliação definido anteriormente.

A avaliação das aplicações foi realizada com base nos dados coletados na segunda execução das aplicações, para que o tempo de compilação (no caso da aplicação .NET) não influenciasse no resultado do comparativo.

A Figura 28 apresenta um gráfico comparando os resultados das avaliações dos sistemas.

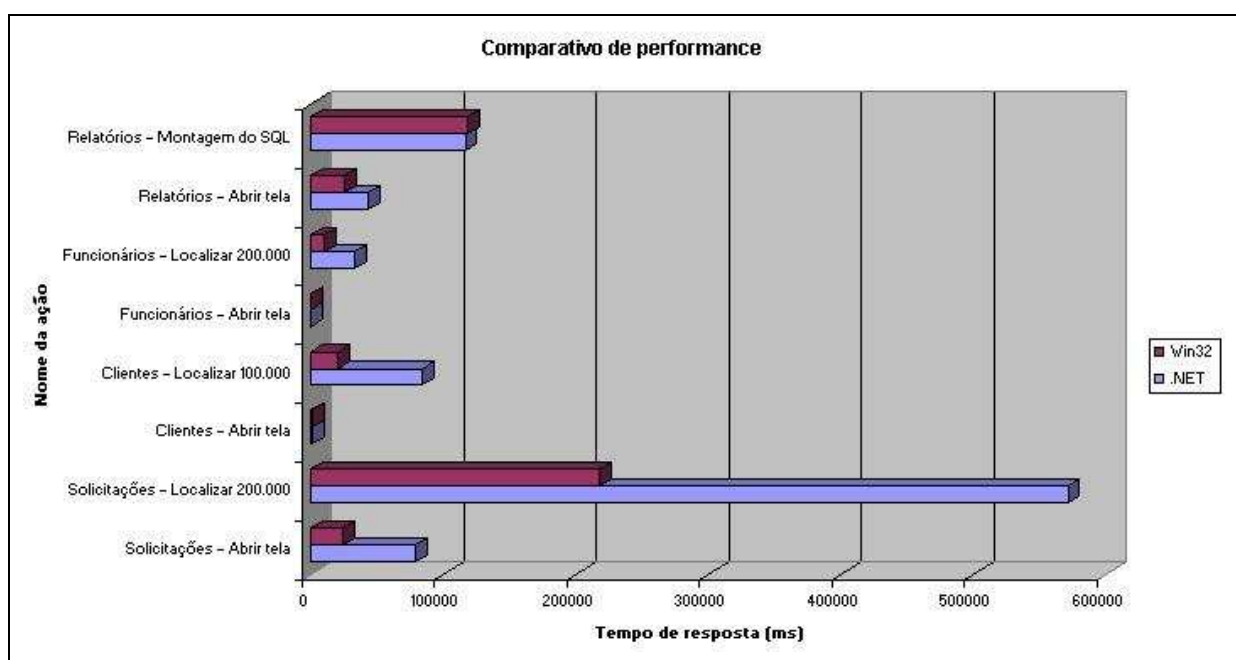


Figura 28 – Comparativo de performance

A Figura 28 deixa claro que a aplicação desenvolvida para Win32 é significativamente mais rápida que a versão migrada para .NET, chegando em alguns casos ser até 4 vezes menor o tempo de resposta.

Uma possível explicação para essa diferença de performance é devido ao fato da CLR executar várias tarefas importantes que não ocorrem na versão Win32, tais como carregar o código, alocar e desalocar memória e executar a verificação de tipo, entre outras tarefas.

4 CONCLUSÕES

Esse trabalho apresentou o processo de migração de uma aplicação de Win32 para .NET e o comparativo de performance do sistema nas diferentes plataformas.

O sistema SEPA desenvolvido para este trabalho mostrou-se apropriado para avaliar esse processo de migração, pois apresenta várias características presentes em sistemas mais complexos, tais como: operações básicas de gerenciamento dos dados em um banco de dados, uso de chaves primária e estrangeira contendo campos simples e compostos, relatórios, consultas entre outros. Com o sistema desenvolvido foi possível alcançar todos os objetivos inicialmente pré-estabelecidos.

Este trabalho permitiu verificar que de fato o processo de migração de Delphi Win32 para Delphi .NET não é uma tarefa complexa. Para compilar a aplicação no Delphi 8, por exemplo, foram necessárias poucas modificações no código fonte. Como maior desafio para esse processo encontrou-se a dificuldade de descobrir as causas dos erros ocorridos tanto na compilação quanto na execução e a carência de documentação relativa a estes erros.

O comparativo permitiu verificar que, nas condições de teste apresentadas, a diferença de performance entre as aplicações foi significativa, chegando em alguns casos ser até 4 vezes mais lenta a versão .NET em relação a versão Win32, sendo este um aspecto relevante a ser avaliado na hora de migrar uma aplicação.

Entretanto, é importante considerar não apenas a performance como um fator determinante para a migração, mas também outros aspectos tais como: a portabilidade em diferentes plataformas, a atualização tecnológica e a conectividade com diferentes linguagens.

Neste sentido, o .NET contempla de modo mais eficiente esses aspectos do que as aplicações Win32 uma vez que ele já foi concebido como uma plataforma de desenvolvimento com essas características.

Assim, a migração dos sistemas Delphi Win32 para Delphi .NET pode ser uma boa opção para os desenvolvedores que desejam aproveitar tanto o seu conhecimento na linguagem Delphi como também o seu código fonte atual alcançando a um novo patamar de atualização tecnológica.

4.1 EXTENSÕES

Como sugestões para futuros trabalhos, destacam-se

- a) a possibilidade de se construir outros tipos de aplicações com grau de complexidade diverso da aplicação desenvolvida neste trabalho, para realizar o mesmo processo aqui descrito;
- b) a possibilidade de comparação de performance entre uma aplicação desenvolvida em Delphi 7 para uma aplicação *Windows Forms* utilizando Delphi 8, avaliando as vantagens e desvantagens da migração.

REFERÊNCIAS BIBLIOGRÁFICAS

ABNT - ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS. **NBR 13596**: tecnologia de informação - avaliação de produto de *software*: características de qualidade e diretrizes para o seu uso. Rio de Janeiro: ABNT, 1996. 10 p.

BORLAND. **Delphi™ 8 architect**, [s.l], [2003a]. Disponível em: <http://www.borland.com.br/delphi_net/architect/index.html>. Acesso em: 13 abr. 2004.

BORLAND. **Delphi™ 8 for the Microsoft® .NET framework**, [s.l], [2003b]. Disponível em: <http://www.borland.com.br/delphi_net/>. Acesso em: 13 abr. 2004.

BORLAND SOFTWARE CORPORATION. **Borland Help**, Borland, 2003.

D' ANGELO, Fernando. **.NET Framework e ASP.NET**, [s.l], dez. 2003a. Disponível em: <<http://www.aspbrasil.com.br/tutoriais/detalhes.aspx?codConteudo=1291>>. Acesso em: 10 jun. 2004.

D' ANGELO, Fernando. **Microsoft .NET a plataforma Java da Microsoft**, [s.l], nov. 2003b. Disponível em: <<http://www.aspbrasil.com.br/tutoriais/detalhes.aspx?codConteudo=1274>>. Acesso em: 29 mar. 2004.

DOSYUKOV, Serge. **D7 Ent/D8 Arch component migration list**, [s.l], jun. 2004. Disponível em: <<http://bdn.borland.com/article/0,1410,31984,00.html>>. Acesso em: 10 set. 2004.

FERREIRA, Ricardo. **ASP.NET no Delphi 8**, [s.l], [2004]. Disponível em: <<http://www.delphibr.com.br/artigos/aspdotnetd8.php/>>. Acesso em: 29 mar. 2004.

FURLAN, José David. **Modelagem de objetos através da UML – the unified modeling language**, São Paulo: Makron Books, 1998.

HADDAD, Renato. **Dez razões para migrar para ASP.NET**, [s.l], nov. 2002. Disponível em: <http://www.msdnbrasil.com.br/tecnologias/aspnet_migrar.aspx>. Acesso em: 13 abr. 2004.

MICROSOFT. **Introdução a .NET**, [s.l], [2003]. Disponível em: <<http://www.microsoft.com/brasil/dotnet/introducao/oquee.asp>>. Acesso em: 29 mar. 2004.

PACHECO, Xavier. **Delphi for .NET Developer's Guide**, Indianapolis: Sams, 2004.

SWART, Bob. **Migrating Borland® Delphi™ applications to the Microsoft® .NET Framework with Delphi**, Borland, fev. 2004. Disponível em:
<http://www.borland.com/products/white_papers/pdf/del_migrating_delphi_win32_to_dotnet.pdf>. Acesso em: 10 jun. 2004. 22 p.

ANEXO A – Relação de componentes do Delphi 7 disponíveis no Delphi 8

Componente	Delphi 7 Enterprise	Delphi 8 Architect	Delphi 8 WinForms
Standard			
TFrames	✓	✓	
TMainMenu	✓	✓	✓
TPopopMenu	✓	✓	✓
TLabel	✓	✓	✓
TLinkLabel			✓
TEdit	✓	✓	✓
TMemo	✓	✓	
TButton	✓	✓	✓
TCheckBox	✓	✓	✓
TRadioButton	✓	✓	✓
TListBox	✓	✓	✓
TComboBox	✓	✓	✓
TScrollBar	✓	✓	✓
TGroupBox	✓	✓	✓
TRadioGroup	✓	✓	
TPanel	✓	✓	✓
TPropertyGrid			✓
TActionList	✓	✓	
Additional			
TBitBtn	✓	✓	
TSpeedButton	✓	✓	
TMaskEdit	✓	✓	
TStringGrid	✓	✓	
TDrawGrid	✓	✓	
TImage	✓	✓	✓
TShape	✓	✓	
TBevel	✓	✓	
TScrollBar	✓	✓	
TCheckListBox	✓	✓	✓
TSplitter	✓	✓	✓
TStaticText	✓	✓	
TControlBar	✓	✓	
TApplicationEvents	✓	✓	
TValueListEditor	✓	✓	
TLabelEdit	✓	✓	
TColorBox	✓	✓	
TColorListBox	✓	✓	
TChart	✓	www.steema.com	
TActionManager	✓		
TActionMainMenuBar	✓		
TActionToolBar	✓		

TXPColorMap...	✓		
TCustomizedDlg	✓		
Win32			
TTabControl	✓	✓	✓
TPageControl	✓	✓	
TImageList	✓	✓	✓
TRichEdit	✓	✓	✓
TTrackBar	✓	✓	✓
TProgressBar	✓	✓	✓
TUpDown	✓	✓	✓
THotKey	✓	✓	
TAnimate	✓	✓	
TDateTimePicker	✓	✓	✓
TMonthCalendar	✓	✓	✓
TTreeView	✓	✓	✓
TListView	✓	✓	✓
THeaderControl	✓	✓	
TStatusBar	✓	✓	✓
TToolBar	✓	✓	✓
TCoolBar	✓	✓	
TPageScroller	✓	✓	
TComboBoxEx	✓	✓	
TXPManifest	✓		
HelpProvider			✓
ToolTip			✓
NotifyIcon			✓
ErrorProvider			✓
System			
TTimer	✓	✓	✓
TPaintBox	✓	✓	
FileSystemWatcher			
EventLog			
MessageQueue			
PerformanceCounter			
Process			
ServiceController			
ReportDocument			
TMediaPlayer	✓	✓	
TOleContainer	✓		
TDDEClientConv	✓		
TDDEClientItem	✓		
TDDEServerConv	✓		
TDDEServerItem	✓		
Win 3.1			
TDBLookupList	✓		
TDBLookupCombo	✓		

TTabSet	✓	✓	
TOutline	✓	✓	
TTabbedNotebook	✓	✓	
TNotebook	✓	✓	
THeader	✓	✓	
TFileListBox	✓	✓	
TDirectoryListBox	✓	✓	
TDirectorySearcher			
TDriveComboBox	✓	✓	
TFilterComboBox	✓	✓	
Data Access			
TDataSource	✓	✓	
TClientDataSet	✓	✓	
TDataSetProvider	✓	✓	
TXMLTransform	✓		
TXMLTransformProvider	✓		
TXMLTransformClient	✓		
TADONetConnector		Borland.VCL.Design.AdoNet.dll	
Data Controls			
TDBGrid	✓	✓	✓
TDBNavigator	✓	✓	
TDBText	✓	✓	✓
TDBEdit	✓	✓	✓
TDBMemo	✓	✓	
TDBImage	✓	✓	✓
TDBListBox	✓	✓	✓
TDBComboBox	✓	✓	✓
TDBCheckBox	✓	✓	✓
TDBRadioGroup	✓	✓	
TDBLookupListBox	✓	✓	
TDBLookupComboBox	✓	✓	
TDBRichEdit	✓	✓	✓
TDBCtrlGrid	✓	✓	
TDBChart	✓	www.steema.com	
DataSnap			
TDCOMConnection	✓	✓	
TSocketConnection	✓		
TSimpleObjectBroker	✓		
TWEBConnection	✓		
TConnectionBroker	✓	Borland.VCL.Design.Compat.dll	
TSharedConnection	✓		
TLocalConnection	✓	Borland.VCL.Design.Compat.dll	

ADO			
TADOConnection	✓		✓
TADOCommand	✓		✓
TADODataSet	✓		✓
TADOTable	✓		
TADOQuery	✓		
TADOStoredProc	✓		
TRDSCONNECTION	✓		
BDE			
TTable	✓	✓	
TQuery	✓	✓	
TStoredProc	✓	Borland.VCL.Design.Compat.dll	
TDatabase	✓	✓	
TSession	✓	✓	
TBatchMove	✓	✓	
TUpdateSQL	✓	Borland.VCL.Design.Compat.dll	
TNestedTable	✓	Borland.VCL.Design.Compat.dll	
Interbase			
TIBTable	✓	✓	
TIBQuery	✓	✓	
TIBStoredProc	✓	✓	
TIBDatabase	✓	✓	
TIBTransaction	✓	✓	
TIBUpdateSQL	✓	✓	
TIBDataset	✓	✓	
TIBSQL	✓	✓	
TIBDatabaseInfo	✓	✓	
TIBSQLMonitor	✓	✓	
TIBEvents	✓		
TIBExtract	✓	✓	
TIBClientDataset	✓		
TIBConnectionBroker	✓	✓	
TIBScript	✓	✓	
TIBConnectionBroker	✓	✓	
TIBSQLParser	✓	✓	
TIBDatabaseNI	✓	✓	
TIBFilterDialog	✓		
Interbase Admin			
TIBConfigService	✓	✓	
TIBBackupService	✓	✓	
TIBRestoreService	✓	✓	
TIBValidationService	✓	✓	
TIBStatisticalService	✓	✓	

TIBLogService	✓	✓	
TIBSecurityService	✓	✓	
TIBServerProperties	✓	✓	
TIBLicensingService	✓	✓	
TIBInstall	✓		
TIBUnInstall	✓		
dbExpress			
TSQLConnection	✓	✓	
TSQLDataset	✓	✓	
TSQLQuery	✓	✓	
TSQLStoredProc	✓	✓	
TSQLTable	✓	✓	
TSQLMonitor	✓	✓	
TSimpleDataset	✓		
WebServices			
THTTPRIO	✓		
THTTPReqResp	✓		
TOPToSoapDomConvert	✓		
TSOAPConnection	✓		
THTTPSoapDispatcher	✓		
TWSDLHTMLPublish	✓		
THTTPSoapPascalInvoker	✓		
InternetExpress			
TXMLBroker	✓		
TInetXPageProducer	✓		
Internet			
TWebDispatcher	✓		
TPageProducer	✓		
TDataSetTableProducer	✓		
TDataSetPageProducer	✓		
TQueryPageProducer	✓		
TSQLQueryPageProducer	✓		
TTCPCClient	✓		
TTCPServer	✓		
TUDPSocket	✓		
TXMLDocument	✓	Borland.VCL.Design.XML.dll	
TWebBrowser	✓		
WebSnap			
TAdapter	✓		
TPageAdapter	✓		
TDataSetAdapter	✓		
TLoginFormAdapter	✓		
TStringValuesList	✓		
TDataSetValuesList	✓		
TWEBAppComponents	✓		

TApplicationAdapter	✓		
TEndUserAdapter	✓		
TEndUserSessionAdapter	✓		
TPageDispatcher	✓		
TAdapterDispatcher	✓		
TLocateFileService	✓		
TSessionsService	✓		
TWebUserList	✓		
TXSLPageProducer	✓		
TAdapterPageProducer	✓		
Dialogs			
TOpenDialog	✓	✓	✓
TSaveDialog	✓	✓	✓
TOpenPictureDialog	✓	✓	
TSavePictureDialog	✓	✓	
TFontDialog	✓	✓	✓
TColorDialog	✓	✓	✓
TPrintDialog	✓	✓	✓
TPrinterSetupDialog	✓	✓	
TFindDialog	✓	✓	
TReplaceDialog	✓	✓	
TPageSetupDialog	✓	✓	✓
PrintPreviewControl			✓
PrintPreviewDialog			✓
PrintDocument			✓
CrystalReportViewer			
Decision Cube			
...	✓		
Samples			
...	✓	(\$Delphi)\Demos \VCL\Samples	
ActiveX			
...	✓		
Rave			
...	✓	Available from CD	
Indy			
...	✓	✓	
COM+			
...	✓		
IntraWeb			
...	✓	www.atozed.com	
Servers			
...	✓		

Fonte: Adaptado de Dosyukov (2004)