Centro Universitário de Brasília – UniCEUB
Faculdade de Ciências Exatas e de Tecnologia – FAET
Curso de Engenharia da Computação
Disciplina: Projeto Final
Professor Orientador: M.Sc Antonio José Gonçalves Pinto

Mecanismo de Avaliação de Qualidade de Serviço

Diogo Henrique da Rocha Bastos Cardoso Curto
Matrícula: 2023796-3

Brasília/DF, 1º semestre de 2007.
UniCEUB – Centro Universitário de Brasília

FAET – Faculdade de Ciências Exatas e Tecnologia

Curso de Engenharia da Computação

MECANISMO DE AVALIAÇÃO DE QUALIDADE DE SERVIÇO

DIOGO HENRIQUE DA ROCHA BASTOS CARDOSO CURTO

Monografia apresentada ao Curso Engenharia da Computação, como requisito parcial a obtenção do grau de Engenheiro de Computação.

Uniceub – Centro Universitário de Brasília

Orientador: Prof. Msc. Antonio José Gonçalves Pinto

Brasília/DF, 1º Semestre de 2007

RESUMO

Neste trabalho é apresentada a construção de um software acadêmico de avaliação quantitativa de pacotes de rede. O software grava históricos de pacotes de redes durante uma conexão entre dois computadores que estão utilizando um mesmo serviço de rede. Estes pacotes capturados são analisados com base em comparações de normas dos parâmetros de qualidade de serviço QoS, como largura de banda, flutuação, atraso e perdas de pacotes. Após analisar os pacotes de redes capturados, o software apresenta um resumo da análise e os gráficos dos parâmetros de QoS calculados.

Palavras chave: redes, QoS, monitoração de rede, VoIP, TCP/IP, UDP, análise de serviço de redes, cálculo de parâmetros de QoS.

ABSTRACT

In this work is presented the construction of a network package quantitative evaluation academic software. The software records history of network packages during a conection between two computers that are utilizing the same network service. These captured packages are analised on the basis of comparisons with Quality of Service standards, such as bandwidth, jitter, delay e loses of packages. After the analysis of the captured network packages, the software presents a summary of its analysis and the graphics of the QoS parameters calculated.

Keywords: network, QoS, network monitoring, VoIP, TCP/IP, UDP, network service analysis, QoS parameters calculations

AGRADECIMENTO

Em um primeiro momento achei que não deveria fazer um agradecimento muito extenso. Porém, agradecer é reconhecer e eu não poderia terminar esta fase sem nomear cada pessoa que me ajudou nestes quase dez anos de faculdade.

Gostaria de começar agradecendo aos docentes com os quais tive o prazer de conviver nestes anos. Professores como meu orientador Antônio Gonçalves, o professor Francisco Javier e o professor Abiezer Amarila, estes de Brasília. Aos meus professores da Faculdade de Engenharia de Sorocaba, Youzo e Marcus Vinícius.

Não poderia deixar de agradecer aos meus amigos e companheiros de faculdade, tais como Alexandre "Mangaba", Rodrigo "Xamba", Renatinho e Renatão, de Sorocaba. Aos amigos e companheiros de Brasília: Michel "Todynho" Calheiros, Gustavo "Capitão", Wagner Bastos, João Paulo Barbosa e Alessandro Catão.

Aos meus pais, Ivana e Normando, por terem me educado. Aos meus avós Octávio, Odila e Cida pelo carinho e amor dedicados. A toda minha família por ter me apoiado em todos os momentos e me conduzirem até aqui.

Um agradecimento especial aos meus sócios Rodrigo Benin e Luiz Miamoto, por agüentarem meu mal-humor e minhas ausências.

A minha esposa Daniele. Eu te amo.

Gostaria também de agradecer aqueles que por algum lapso de memória tenha esquecido de citar, mas que de uma forma ou de outra me apoiaram até aqui.

SUMÁRIO

ÍNDICE DE FIGURAS

ÍNDICE DE TABELAS

ÍNDICE DE EQUAÇÕES

LISTA DE TERMOS E SIGLAS

API – Application Programming Interface – Interface de Programação de Aplicação

CAIS – Centro de Atendimento a Incidentes de Segurança

CODEC – acrônimo de Codificador/Decodificador, dispositivo de hardware ou software que codifica/decodifica sinais.

FLL – Frequency-Lock Loop – Loop de Ajuste de Freqüência

FTP – File Transfer Protocol – Protocolo de Transferência de Arquivos

GPS – Global Positioning System – Sistema de Posicionamento Global

HTTP – HyperText Transfer Protocol – Protocolo de Transferência de Hipertexto

IHL – Internet Header Length – Comprimento do Cabeçalho da Internet

IP – Internet Protocol – Protocolo da Internet

IPX/SPX – Internetwork Packet Exchange/Sequenced Packet Exchange – Troca de Pacotes de Redes / Seqüência de Troca de Pacotes.

ISO – International Organization for Standardization – Organização Internacional de Normatização.

ITU-T – International Telecomunication Union – União Internacional de Telecomunicações

MAQS – Mecanismo de Avaliação de Qualidade de Serviço

NetBIOS – Network Basic Input/Output System – Sistema Básico de Entrada e Saída de Redes.

NTP – Network Time Protocol – Protocolo de Tempo de Redes.

OSI – Open Systems Interconnection – Interconexão de Sistemas Abertos

PLL – Phase-Lock Loop – Loop de Ajuste de Fase

QoS – Quality of Service – Qualidade de Serviço

RTP – Real Time Protocol – Protocolo de aplicações de Tempo Real

SQL – (Structured Query Language – Linguagem de Consulta Estruturada)

SSH – Secure Shell – Shell Seguro

TCP – Transmission Control Protocol – Protocolo de controle de Transmissão

TTL – Time To Live – Tempo Para Viver

UDP – User Datagrama Protocol – Protocolo de Datagramas de Usuário

UTC – Universal Time Coordinated – Coordenada de Tempo Universal

VOIP – Voice Internet Protocol - Voz sobre o protocolo IP

1. INTRODUÇÃO

Até o início dos anos 80, a Internet estava limitada às suas raízes acadêmicas e as suas aplicações e tráfego eram restritos. Pode-se citar como exemplos de aplicações a transferência de arquivos via *FTP*, e-mail via *SMTP* e acesso remoto e comunicação de terminais via *TELNET*[1] (VEGESNA, 2001).

Como estas aplicações mais simples exigiam menos recursos de rede para funcionar, os requisitos de qualidade de serviço não eram tratados na construção dos caminhos de transferência de dados.

A importância da qualidade de serviço foi aumentando conforme a Internet evoluía das raízes acadêmicas para o atual estágio popular e comercial. O desenvolvimento de novas aplicações, como *VOIP* e Videoconferência, consomem mais recursos de rede e necessitam de um controle de tráfego maior do que o usado até a década de 80 (VEGESNA, 2001).

Hoje em dia, as novas aplicações trafegam em redes corporativas e na Internet, demandando cada vez mais largura de banda e disponibilidade de uso da banda (TANENBAUM, 2003).

Por outro lado, a velocidade com que se trafega estes dados em rede, não tem acompanhado esta evolução de consumo, e estas novas aplicações ainda concorrem com as aplicações mais antigas, como, por exemplo, tráfego de arquivos e aplicações corporativas, banco de dados, etc. O resultado de tudo isso é a degradação de desempenho e congestionamento da rede.

Neste contexto, faz-se necessário o acompanhamento, a monitoração e adequação dos parâmetros de *QoS,* tais como: flutuação, latência, largura de banda e perdas de uma rede.

[1] TELNET – Protocolo de acesso remoto e comunicação entre terminais. [TAN 89]

1.1. Motivação

Atualmente, as empresas buscam, cada vez mais, reduzir custos e agilizar seus serviços. Muitas delas tem visto na Internet uma oportunidade de se efetuar esta redução de custos e agilizar o relacionamento com seus clientes e fornecedores.

Soluções antigas como a telefonia convencional, e viagens a serviço, podem ser trocadas por soluções que tem custo menor do que os praticados anteriormente como sites com soluções de auto-atendimento, VoIP e videoconferência.

Com o custo de uma conexão simples de Internet, as empresas podem usufruir destas soluções, abrindo um novo leque de alternativas para comunicação e novos modelos de prestação de serviços.

Se compararmos a solução de VoIP com a telefonia convencional, pode-se verificar que ela tem um custo mais baixo. Mas para que a solução de VoIP funcione a contento, é preciso se tomar alguns cuidados, garantindo a qualidade do serviço ao usuário final. Para garantir tal qualidade, a rede corporativa da empresa necessita estar adequada para o funcionamento da solução.

A motivação deste trabalho é apresentar um mecanismo que possibilite a análise de uma rede de computadores baseado em parâmetros de qualidade de serviço. Utilizando estes parâmetros de qualidade de serviço de redes, podemos compará-los com uma norma técnica, ou parâmetros descritos pelo próprio operador do sistema, e identificar os possíveis desvios. Dessa forma, torna-se possível tomar ações visando à garantia de qualidade das soluções baseadas em redes, e conseqüentemente, a redução de custos operacionais.

1.2. Objetivo do Projeto

Este projeto concentra-se na implementação de um mecanismo de avaliação de qualidade de serviço (MAQS) para aplicações em redes de computadores. As aplicações demandam requisitos variados de qualidade de serviço, o que exige uma análise especifica para cada tipo de aplicação, visando a garantia de qualidade do mesmo.

A implementação deste mecanismo é feita através de um software que permite fazer uma estimativa dos parâmetros de QoS de uma determinada rede, através da avaliação do tráfego gerado por aplicações e serviços em execução, efetuando-se comparações para determinar se o desempenho da rede está compatível com o tipo de serviço que está sendo utilizado.

Para o teste do mecanismo desenvolvido por este trabalho escolheu-se uma aplicação de rede VoIP, e os parâmetros de qualidade de serviço obtidos no MAQS serão comparados com normas técnicas desenvolvidas pelo ITU-T(International Telecommunications Union – Telecomunications, ou Organização Internacional dos padrões de telecomunicações).

É desenvolvido neste projeto, um software com dois de apoio, um para captura de pacotes de rede e outro para análise destes pacotes capturados.

1.3. Estrutura do Trabalho

Além do capitulo de introdução, o trabalho foi desenvolvido nos seguintes capítulos:

O **Capitulo 2**, onde são apresentados os principais conceitos, em forma de revisão, que serão essenciais para o entendimento deste trabalho.

O **Capitulo 3** apresenta a construção dos dois módulos do software desenvolvido. São especificadas as tecnologias e as bibliotecas de programas utilizadas. Este capitulo também apresenta trechos comentados do código desenvolvido.

No **Capitulo 4** são mostrados três estudos de caso, onde se utiliza o software na resolução de um problema real, e se detalha o modo de funcionamento do MAQS. Neste capitulo também são descritos os principais resultados obtidos e as conclusões que foram observadas nos casos estudados.

Por fim, no **Capítulo 5** são apresentadas as considerações finais, as dificuldades, as contribuições, e as sugestões para trabalhos futuros.

2. REDE DE COMPUTADORES E QUALIDADE DE SERVIÇO

Uma rede de computadores é construída para um propósito definido. Esta rede de computadores pode disponibilizar diversos serviços a seus usuários. Em uma empresa que distribui produtos, a rede de computadores pode ter o propósito de ligar seus diversos usuários a um computador central, onde são executados os programas administrativos da empresa, compartilhar a conexão da Internet, ligando os funcionários da empresa à Internet, entre outros serviços.

Diversos serviços disponibilizados por uma rede de computadores necessitam de um mínimo de qualidade de serviço, para que possam ser executados de maneira satisfatória.

Para o entendimento dos benefícios proporcionados pela qualidade de serviço é apresentado neste capítulo um estudo teórico sobre modelos de referência e protocolos de rede, demonstrando o funcionamento da comunicação de uma rede IP (Internet Protocol – Protocolo de Internet).

Faz-se, também a introdução teórica sobre um protocolo de sincronização de relógios de computadores, o NTP (Network Time Protocol – Protocolo de Tempo de Rede).

Por fim, traz-se à luz também a fundamentação de conceitos de qualidade de serviço, que viabilizam as ações de melhora e monitoramento das comunicações entre computadores em rede, possibilitando a entrega de serviços de rede com qualidade.

2.1. Modelos de Referência

Neste trabalho se abordará duas das principais arquiteturas de redes: o modelo de referência OSI, e o modelo de referência TCP/IP. A função desses modelos é padronizar a forma de comunicação entre computadores. (TANENBAUM, 2003)

Não se estudará de forma detalhada nenhum dos dois modelos, pois o objetivo é explicar o funcionamento de uma rede de computadores de forma a

entender a importância da qualidade de serviço em uma rede, e não aprofundar o seu funcionamento.

2.1.1. O modelo de referência OSI

O modelo OSI (Open System Interconection – Sistema Aberto de Interconexão), trata de sistemas que estão abertos a comunicação com outros sistemas. Este modelo é baseado em uma proposta da ISO (International Standards Organization – Organização Internacional de Normas) [TAN03].

O modelo tem sete camadas[2], como se pode ver na figura a seguir:

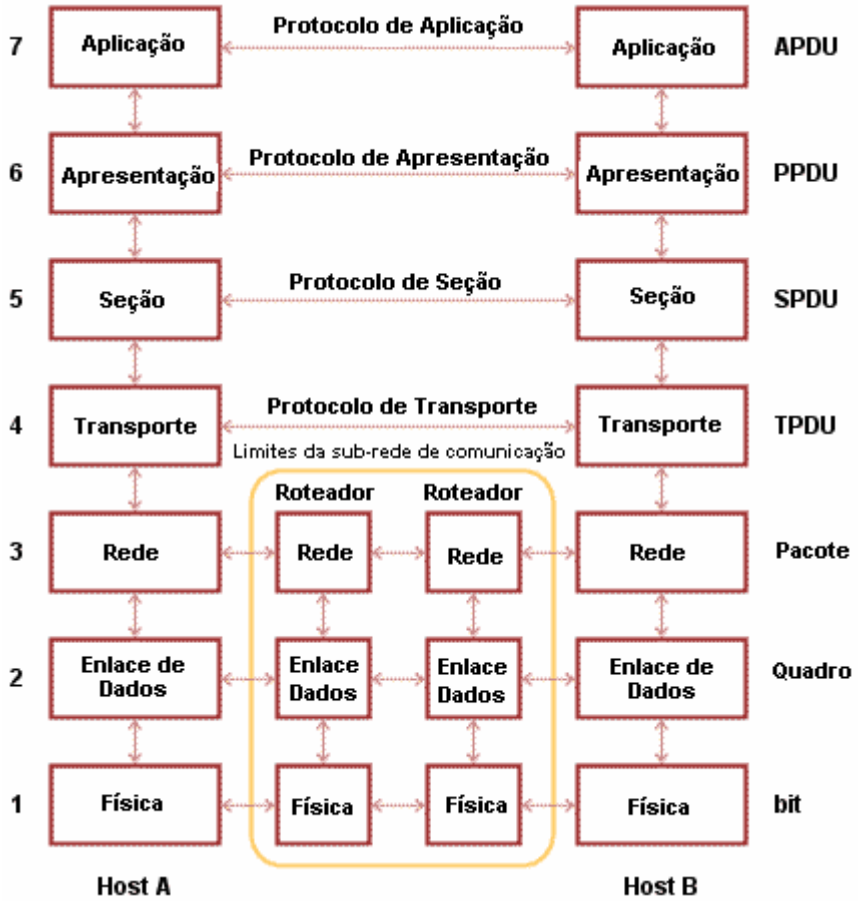


Figura 2-1 – O modelo de Referência OSI (TANENBAUM, 2003)

O modelo tem como princípios (TANENBAUM, 2003):

[2] Camada: uma camada, ou nível, de rede é uma espécie de máquina virtual, que oferece serviços a outras camadas da mesma rede.

1. Uma camada deve ser criada onde houver necessidade de um grau de abstração adicional.
2. Cada camada deve executar uma função bem definida.
3. A função de cada camada deve ser escolhida tendo em vista a definição de protocolos padronizados internacionalmente.
4. Os limites de camadas devem ser escolhidos para minimizar o fluxo de informações pelas interfaces[3].
5. O número de camadas deve ser grande o bastante para que funções distintas não precisem ser desnecessariamente colocadas na mesma camada, e pequeno o suficiente para que a arquitetura não se torne difícil de controlar.

As camadas do modelo OSI e suas principais funções (TANENBAUM, 2003):

1. Camada física: trata da transmissão de bits brutos por um canal de comunicação.
2. Camada de enlace de dados: transforma um canal de transmissão puro em uma linha que pareça livre de erros para a camada de rede. Esta camada faz com que o transmissor divida os dados em quadros de dados (com algumas centenas ou milhares de bytes) e os transmita seqüencialmente.
3. Camada de rede: controla a operação da sub-rede. Determina o caminho que o pacote irá percorrer da origem ao destino.
4. Camada de transporte: esta camada recebe os dados da camada de seção, e assegura que os dados chegarão corretamente ao destino. Tem a função, também, de dividir os dados recebidos e dividi-los em unidades menores ao repassa-los a camada de rede, se for o caso.
5. Camada de seção: permite que os usuários de diferentes máquinas de uma mesma rede estabeleçam sessões[4] entre eles.

[3] Interface: define as operações e os serviços que uma camada de um modelo oferece a outra.

6. Camada de apresentação: gerencia estruturas de dados abstratas e permite a definição e intercâmbio das mesmas (exemplo: registros bancários).
7. Camada de aplicação: contém os protocolos utilizados pelos usuários da rede (exemplo: HTTP, FTP).

2.1.2. O modelo de referência TCP/IP

O modelo TCP/IP tem esse nome por se basear em dois protocolos: o TCP (Transmission Control Protocol – Protocolo de Controle de Transmissão), e o IP (Internet Protocol – Protocolo da Internet). Este modelo tem sua origem na ARPANET[5] uma das primeiras redes de computadores de que se tem notícia, e necessitava que várias redes com protocolos diferentes se comunicassem.

Este é um modelo mais simples do que o modelo OSI, como se pode ver na figura a seguir:

Figura 2-2 – O modelo de Referência TCP/IP (TANENBAUM, 2003)

Vejamos as camadas do modelo TCP/IP e suas principais funções (TANENBAUM, 1989):

---

[4] Sessão: uma sessão de rede oferece diversos serviços, como o controle de quem deve transmitir em que momento (controle de diálogo), sincronização (verificação periódica de transmissões longas para permitir o reinício de uma transmissão interrompida, a partir do ponto onde ela parou) [TAN 03].

[5] ARPANET: rede de pesquisa constituída na década de 1960 pelo Departamento de Defesa dos EUA [TAN 89].

1. Camada aplicação: esta camada contém todos os protocolos de nível mais alto como TELNET (Protocolo de Terminal Virtual) e FTP (Protocolo de Transferência de Arquivos).
2. Camada de Transporte: tem como finalidade permitir que as entidades nos hosts de origem e destino mantenham uma conversação, idêntica a do modelo OSI.
3. Camada de inter-redes: tem como principal tarefa permitir que qualquer host injete pacotes em qualquer rede e garantir que eles trafegarão até o destino, integrando a arquitetura TCP/IP.
4. Camada host/rede: o modelo de referência não especifica o que ocorre nesta camada, exceto ao fato de que o host deve se conectar a rede utilizando algum protocolo para que seja possível enviar pacotes IP. Mas esse protocolo não é definido e varia de host para host e rede para rede.

2.2. Protocolos de rede

Nos primeiros projetos de redes de computadores, o software ficou em segundo plano, sendo a principal preocupação o hardware. Atualmente, esta estratégia foi abandonada, sendo o software de rede a principal preocupação.

Para reduzir a complexidade de um projeto de redes de computadores, a maioria das redes é organizada em camadas ou níveis. Estas camadas são colocadas umas sobre as outras. O número dessas camadas, seus nomes e funções são diferentes em cada rede.

Mas em todas as redes o objetivo de uma camada é oferecer serviços para outra camada. Este método de separação por camadas permite que os detalhes de implementação dos serviços sejam desconhecidos pelas camadas que utilizam o serviço, sendo conhecidas somente pela camada que implementa o serviço.

Quando os computadores se comunicam, eles se comunicam através das camadas. Em outras palavras, a camada ‘X’ do computador ‘A’ se

comunica com a camada ‘X’ do computador ‘B’. As regras e convenções desta comunicação são chamadas de protocolo. (TANENBAUM, 2003)

Neste trabalho estudaremos os protocolos IP (Internet Protocol), TCP (Transmission Control Protocol – Protocolo de Controle de Transferência), UDP (User Datagrama Protocol – Protocolo de Datagramas de Usuário) e NTP (Network Time Protocol – Protocolo de Tempo de Rede).

2.2.1. Protocolo IP

O IP (Internet Protocol – Protocolo de Internet) é um protocolo usado entre duas máquinas em rede para encaminhamento dos dados. Os dados numa rede IP são enviados em blocos referidos como pacotes ou datagramas.

O IP oferece um serviço de datagramas não confiável (também chamado de melhor esforço); ou seja, não apresenta garantias de entrega. Os pacotes podem chegar desordenados (comparado com outros pacotes enviados entre os mesmos hosts), podendo chegar duplicados, ou ainda serem perdidos por inteiro. Quando se precisa de confiabilidade em uma aplicação, esta é adicionada na camada de transporte (camada 5 do modelo OSI), através dos protocolos TCP ou UDP (TANENBAUM, 1989).

As principais funções da acamada IP são (CARVALHO, 1994):

- Transporte do bloco de dados da sub-rede de origem, até a sub-rede destino.
- Fragmentar o datagrama em datagramas menores, se a sub-rede por onde está passando os dados assim necessitar.
- Remontar o datagrama original a partir dos fragmentos.
- Mapeamento dos endereços IP em físicos e vice-versa.

2.2.2. Protocolo TCP

O TCP (Transmission Control Protocol – Protocolo de Controle de Transferência) é o principal protocolo da Internet nos dias de hoje. A versatilidade e robustez tornaram o protocolo adequado para redes globais, já

que este verifica se os dados são enviados de forma correta, na seqüência apropriada e sem erros pela rede (TANENBAUM, 1989).

O objetivo do TCP é oferecer aos usuários de uma rede de computadores um serviço de transferência confiável de dados, implementando mecanismos de recuperação de dados e minimizando o atraso de trânsito para a transmissão dos dados.

O TCP é um protocolo do nível da camada de transporte (camada 4) do Modelo OSI e é sobre o qual assentam a maioria das aplicações utilizadas na Internet, como o SSH (Secure Shell – Shell Seguro), FTP, HTTP.

Em resumo podemos destacar os seguintes serviços providos pelo protocolo TCP:

- Multiplexação[6]: suporta diversos usuários através da utilização de portas;
- Gerenciamento da conexão: estabelece, mantém e termina uma conexão entre os computadores de uma rede.
- Transferência de dados: entrega dos dados entre computadores conectados.
- Sinalização de dados urgentes: especifica e entrega os dados prioritários;
- Data Stream Push: obriga o envio de dados através de uma conexão.
- Relatório de erros.

Em uma conexão, o protocolo TCP especifica três fases: estabelecimento da ligação, manutenção e término de ligação.

O estabelecimento de conexão permite que dois usuários TCP ativem uma conexão lógica. Esta conexão lógica é caracterizada por características que persistem durante toda a conexão. As características mais importantes são: nível de segurança e precedência e parâmetros especificados pelos dois usuários TCP conectados.

[6] A multiplexação é uma técnica que possibilita a transmissão de mais de um sinal utilizando o mesmo meio físico. Ela pode ser: por divisão de freqüência (FDM); por divisão de comprimento de onda (WDM); por divisão de tempo (TDM); por divisão de código (CDM) [TAN 03]

O serviço de manutenção de conexão faz o intercâmbio e o transporte de dados entre os dois usuários conectados. Uma terminação de conexão pode acontecer de duas formas: abrupta (podem ocorrer perda de dados) ou normal (os usuários se desconectam após a transferência dos dados) (CARVALHO, 1994).

O TCP introduz o conceito de porta associado a um serviço. Assim, cada um dos intervenientes na conexão dispõe de uma porta associada (um valor de 16 bits) que dificilmente será o mesmo do interlocutor. Alguns serviços (que fazem uso de protocolos específicos) são tipicamente acessíveis em portas fixas, que são aqueles numerados do 1 ao 1023, como por exemplo o HTTP com a porta 80.

O TCP usa o IP para a entrega dos datagramas à rede, e os pontos de acesso à aplicação são identificados por portas acessadas por multiplexação, o que permite múltiplas ligações em cada host.

O IP trata o pacote TCP como dados e não interpreta qualquer conteúdo da mensagem do TCP, sendo que os dados TCP viajam pela rede em datagramas IP. Os roteadores que interligam as redes apenas verificam o cabeçalho IP, quando fazem o envio dos datagramas. O TCP no destino interpreta as mensagens do protocolo TCP. (TANENBAUM, 2003)

2.2.3. Protocolo UDP

O protocolo UDP (User Datagrama Protocol – Protocolo de Datagramas de Usuário) dá às aplicações acesso direto ao serviço de entrega de datagramas, como o serviço de entrega que o IP fornece. Este protocolo é pouco confiável, não sendo orientado à conexão. Isso significa que este protocolo não utiliza técnicas para confirmar que os dados chegaram ao destino corretamente, não provê meios para controlar a taxa de entrega dos pacotes de dados. Desta forma, pode-se duplicar, perder ou receber dados fora da ordem. A aplicação que utiliza o protocolo UDP é que deve se preocupar com estas características.

O protocolo UDP provê mecanismos simples em que os processos de cada unidade de transmissão de dados UDP (estação ou host) podem usar

para enviar mensagens a processos que estão sendo executados em outras estações. Tais processos utilizam pontos de acesso do UDP que são designados por "portas", que os identificam dentro da estação. Cada mensagem UDP contém tanto a porta de destino quanto a de origem.

Todas as estações contêm uma série de portas de serviço identificadas por números inteiros positivos. O sistema operacional de uma estação provê mecanismos para que os processos executados na própria estação possam especificar uma porta ou acessá-la. Para se comunicar com outro processo de outra estação o primeiro processo necessita conhecer o endereço IP da estação de destino e seu número de porta correspondente (CARVALHO, 1994).

O UDP, tal como o TCP, usa o IP para a entrega dos dados à rede, e os pontos de acesso à aplicação são identificados por portas acessadas por multiplexação, tal como acontece com o TCP, o que permite múltiplas ligações em cada host. (TANENBAUM, 2003)

2.3. O Protocolo NTP

Os servidores NTP (Network Time Protocol – Protocolo de Tempo de Rede) são fundamentais para administração de redes, pois permitem a sincronização dos relógios de equipamentos em rede a partir de uma referência aceita mundialmente. Estas referências são conhecidas como UTC (Universal Time Coordinated - Coordenada de Tempo Universal).

A extensão do alcance da Internet torna a sincronização do tempo crucial para a troca de informações entre milhares de computadores que operam na base “24x7”, ou seja, vinte e quatro horas por dia, sete dias por semana. Os benefícios da utilização do NTP atingem tanto usuários quanto administradores de rede.

Do ponto de vista da administração de redes, a utilização do NTP possibilita a sincronização automática de todos os equipamentos conectados em rede, ou seja, o administrador pode garantir através do servidor que todas as máquinas da rede local estarão no mesmo horário, de forma sincronizada, sem a necessidade de intervenção em cada máquina cliente (RNP, 2000).

Pelo lado dos usuários, a sincronização dos relógios de computadores pode ser vital em certas operações. Tomando como exemplo a mesa de operações interbancárias[7] de um banco. As operações devem ocorrer até um determinado horário, o horário em que o mercado interbancário fecha. Se alguma estação de trabalho não estiver sincronizada, poderá perder o fechamento de alguma operação e conseqüentemente o banco terá prejuízo.

Além disso, a questão da segurança é reforçada com a adoção da sincronização dos relógios dos equipamentos em rede, pois a investigação de eventos de ataques em computadores depende da verificação de históricos em diversos equipamentos. A inconsistência dos horários registrados dificulta muito esse trabalho.

O NTP implementa um modelo de sincronização hierárquico distribuído. No topo encontram-se os servidores de tempo *stratum* 1, computadores conectados diretamente a dispositivos conhecidos como "relógios de referência" (ou servidores *stratum* 0), de altíssima precisão. Tipicamente, estes dispositivos podem ser relógios atômicos, receptores GPS (Global Positioning System – Sistema de Posicionamento Global) ou receptores de rádio. Qualquer servidor NTP que tenha como referência de tempo um servidor *stratum* 1 passa a ser um *stratum* 2, qualquer servidor NTP que tenha como referência de tempo um servidor *stratum* 2 passa a ser um *stratum* 3, e assim por diante. (RNP, 2000)

2.4. Qualidade de Serviço – QoS

Uma forma de ver a camada de transporte é considerar que sua principal função seja melhorar a Qualidade de Serviço (QoS – Quality of Service) oferecida pela camada de rede do modelo OSI. Se o serviço de rede for perfeito, o trabalho da camada de transporte será fácil. No entanto, se o serviço de rede não for perfeito, a camada de transporte terá que servir de ponte para cobrir a distância entre o que os usuários de transporte desejam e o que a

[7] Operações interbancárias: operações de crédito realizadas entre bancos em um mercado (bolsa) específico.

camada de rede oferece. (MURHAMMER; ATAKAN; BRETZ; PUGH; SUZUKI & WOOD, 2000)

Qualidade de Serviço pode ser definida por um número específico de parâmetros. O serviço de transporte pode permitir ao usuário determinar os valores preferenciais, os valores aceitáveis e os valores mínimos para vários parâmetros de serviço no momento em que uma conexão é estabelecida. (MURHAMMER; ATAKAN; BRETZ; PUGH; SUZUKI & WOOD, 2000)

Alguns parâmetros também podem ser usados no transporte sem conexão. É tarefa da camada de transporte examinar esses parâmetros e, dependendo do(s) tipo(s) de serviço(s) de rede disponível(eis), determinar se é possível realizar o serviço solicitado. (TANENBAUM, 2003)

Os requisitos de QoS comumente mais utilizados são: vazão(largura de banda), perdas de pacotes, flutuação e atraso (latência). (TANENBAUM, 2003)

2.4.1. Vazão (Largura de Banda)

Um dos principais atributos de um enlace de transmissão de dados é sua capacidade nominal de transportar bits, ou banda nominal.

A banda nominal é definida com uma propriedade física do meio de transmissão, e em geral depende da construção, da espessura e do comprimento do fio. Esta propriedade pode ser medida pelas faixas de freqüências transmitidas pelo meio, sem serem fortemente atenuadas. Em redes de computadores, a largura de banda, é percebida pela taxa de transferência de dados em um dado seguimento de rede. (TANENBAUM, 2003)

A vazão, ou a largura de banda de rede ocupada pela mídia, corresponde à taxa de bits que são efetivamente transportados, tendo como valor máximo a banda nominal (FONSECA & STANTON, 2004). No projeto a vazão poderá ser tratada como largura de banda.

2.4.2. Perdas de pacotes

O parâmetro de QoS de perdas de pacotes pode ser definido como o requisito de QoS que mede a quantidade de pacotes que não são entregues a

seu destino em um determinado espaço de tempo, seja este em decorrência de falhas na comunicação, ou congestionamento do tráfego de pacotes (FONSECA & STANTON, 2004).

A taxa de perdas de pacotes é calculada no lado do receptor como a razão entre as quantidades de pacotes perdidos e a quantidade de pacotes transmitidos, em cada intervalo de tempo considerado. As perdas de pacotes em aplicações de mídia contínua geralmente são devidas ao descompasso entre a taxa de transmissão (ou demanda) dos pacotes e a capacidade de transmissão do canal, o que leva ao congestionamento em um ou mais roteadores da rede, com o crescimento das filas de pacotes, o aumento de retardo e eventual descarte dos pacotes (FONSECA & STANTON, 2004).

2.4.3. Atraso (Latência ou Retardo)

Latência é o tempo que uma mensagem demora a atravessar um sistema (TANENBAUM, 2003). Latência está intrinsecamente ligada a outro conceito de engenharia, o débito. Embora de certa forma sejam ambos uma medida de velocidade, não são, de todo, a mesma coisa. Latência é a medida do tempo decorrido entre o início de uma atividade e a sua conclusão, enquanto que débito é o número total de tais atividades durante um determinado espaço de tempo.

Tanto latência como a perda têm efeitos dramáticos no desenho de sistemas de telecomunicações e de computação, já que geralmente melhorias num deles repercutem-se negativamente no outro. Para a maioria das operações, como transferência de arquivos, a perda é a medida mais importante, pois este tipo de operação não será concluído enquanto todos os dados tiverem sido transferidos. Em contrapartida, o sistema de travagem de um automóvel necessita de muito pouca informação (ativo ou inativo), mas esta informação deve ser enviada com o menor atraso possível (TANENBAUM, 2003).

2.4.4. Flutuação (Jitter)

As redes não conseguem garantir uma entrega constante de pacotes ao destino. Assim, os pacotes chegam de forma variável, como mostra a figura 2.3, ocasionando o *jitter*, que nada mais é do que uma flutuação na latência, ou variação estatística do retardo (BRUM; VOGT; GONÇALVES & MENDES, 2002).

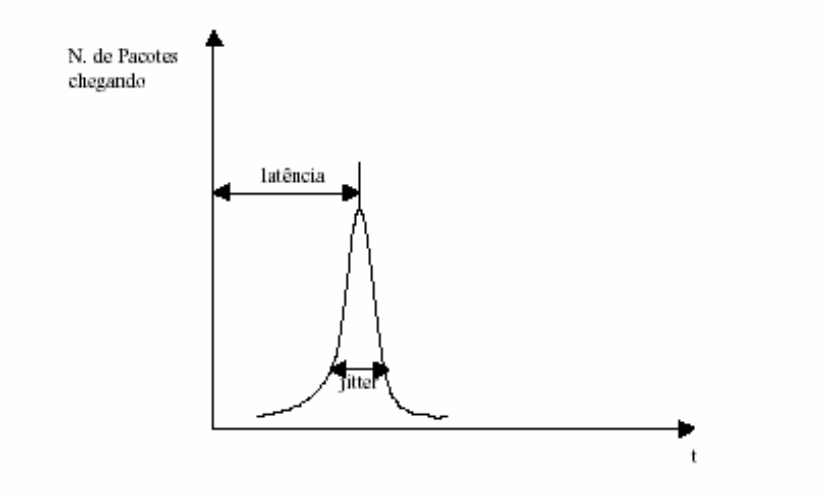


Figura 2-3 – Comparação entre latência e flutuação (BRUM; VOGT; GONÇALVES & MENDES, 2002).

A conseqüência da flutuação é que a aplicação no destino deve criar um *buffer* cujo tamanho vai depender da flutuação, gerando mais atraso na conversação. Esse buffer vai servir como uma reserva para manter a taxa de entrega constante no interlocutor. Daí a importância de latência e flutuação baixas em determinadas aplicações sensíveis a esses fatores, como videoconferência (BRUM; VOGT; GONÇALVES & MENDES, 2002).

A Figura 2.4 ilustra o efeito da flutuação entre a entrega de pacotes na origem e o seu processamento no destino. Observe que a flutuação causa não somente uma entrega com periodicidade variável como também a entrega de pacotes fora de ordem (LIMA; LAMARCA & OLIVEIRA, 2002).

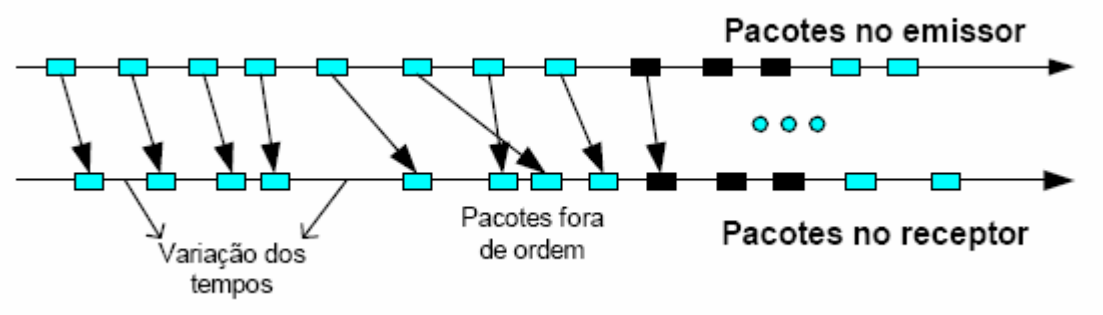


Figura 2-4 – Efeito da flutuação para as aplicações (LIMA; LAMARCA & OLIVEIRA, 2002)

2.5. Aplicações de Redes e Parâmetros de QoS

Podemos relacionar a sensibilidade a variações dos parâmetros de QoS com algumas aplicações que são utilizadas em redes conforme a tabela 2.1 (MARTINS, 1999).

Tabela 2-1: Aplicações x Sensibilidade ao Parâmetro de QoS

Aplicação	Vazão	Perdas	Latência	Jitter
Voz (Telefonia)	Muito baixa	Média	Alta	Alta
Comercio eletrônico	Baixa	Alta	Alta	Baixa
Transações	Baixa	Alta	Alta	Baixa
Correio eletrônico	Baixa	Alta	Baixa	Baixa
Acesso remoto (Telnet)	Baixa	Alta	Média	Baixa
Navegação web casual	Baixa	Média	Média	Baixa
Navegação web crítica	Média	Alta	Alta	Baixa
Transferência de arquivos	Alta	Média	Baixa	Baixa
Videoconferência	Alta	Média	Alta	Alta
Multicast	Alta	Alta	Alta	Alta

As aplicações de Rede se relacionam com Vazão de acordo com a tabela 2-2 (MARTINS, 1999):

Tabela 2-2 Vazões típicas de uma Aplicação de Rede

Aplicação	Vazão Típica
Aplicações Transacionais	1 kbps a 50 kbps
Voz (VoIP)	10 kbps a 120 kbps
Aplicações Web (WWW)	10 kbps a 500 kbps
Vídeo (Streaming)	100 kbps a 1 Mbps
Aplicação Conferência	500 kbps a 1 Mbps
Vídeo MPEG	1 Mbps a 10 Mbps
Aplicação Imagens Médicas	10 Mbps a 100 Mbps
Aplicação Realidade Virtual	80 Mbps a 150 Mbps

A tabela 2-3 apresenta os níveis toleráveis de um parâmetro de QoS para as aplicações de vídeo e áudio (LUNARDI, 2001):

Tabela 2-3 - Características de Multimídia

	Áudio	**Vídeo**
Tolerância a Perdas (%)	3 à 9%	1%
Tolerância a atrasos (ms)	250 – Telefonia 40 – Conferências sem cancelar eco 150 – Conferências com cancelamento de eco	200
Tolerância à flutuação (MS)	20	10

2.5.1. Tráfego VoIP

A seleção do tamanho dos pacotes de VoIP é um compromisso entre o tempo de processamento e transmissão na rede e o efeito que as perdas têm na performance. Pacotes de tamanhos grandes reduzem a largura de banda necessária à transmissão (o cabeçalho tem sempre o mesmo tamanho), mas aumentam o atraso devido ao tempo necessário para preencher cada pacote de dados. Pacotes de dados pequenos permitem que, em caso de perda de um

pacote, não se perca uma parte significativa da mensagem a transmitir (visto que os pacotes em falta não serão retransmitidos).

O tamanho dos pacotes varia de acordo com o CODEC que utilizarmos e com o tamanho das amostras de voz utilizadas. Numa compressão a 8 Kbps, com um envio de pacotes de dados a uma freqüência de 20 ms, temos que cada pacote de dados terá um tamanho de 20 bytes. No entanto, se a transmissão for feita numa rede IP usando o protocolo RTP, teremos de acrescentar ainda um cabeçalho Ethernet de 14 bytes, um cabeçalho IP de 20 bytes, um UDP de 8 bytes e ainda mais 12 bytes para o cabeçalho RTP. Ou seja, para enviarmos um pacote de 20 bytes teremos um overhead de 54 bytes (BOGER, s.n.).

2.5.2. A norma ITU-T

O ITU-T (International Telecomunication Union – União Internacional de Telecomunicações) normatiza vários aspectos de telecomunicações. A tabela 2-4 mostra a normatização da ITU-T para Audio e Vídeo (ITU-T, 1996):

Tabela 2-4 – Norma ITU-T

Áudio		**Video**	
Largura de banda	acima de 100kbps	Largura de banda	acima de 1000kbps
Atrasos	até 150ms	Atrasos	até 150ms
Variação de atraso *(Jitter)*	De 30 ms	Variação de atraso *(Jitter)*	De 30 ms
Taxas de perda	< 1% do total transferido	Taxas de perda	< 1% do total transferido

2.6. Cálculo dos Parâmetros de QoS

Nesta seção são elucidadas as metodologias matemáticas utilizadas no processo de cálculo de QoS que foram implementadas no software utilizado neste trabalho.

2.6.1. Cálculo de Vazão Utilizada

Para estimar o uso local de banda, simplesmente adiciona-se o tamanho dos pacotes enviados, recebidos e detectados em um período fixo de tempo. Se N é o número de pacotes enviados e recebidos por um nó no período de tempo T e S é o tamanho destes pacotes em bytes, a banda média usada no período T é:

$$BW(bps) = \frac{N * S * 8}{T} \qquad (1)$$

A precisão do cálculo de banda depende do intervalo T entre medidas consecutivas. Quanto maior T, mais preciso é o resultado. Entretanto, T deve ser pequeno o suficiente para ser transparente à dinâmica do canal. Portanto a escolha de T é um compromisso entre precisão e transparência. Usando este método, pacotes de qualquer tipo transitando através da área de detecção de portadora serão parte do cálculo.

Assume-se que o tamanho do pacote é conhecido, mesmo quando ele está fora da área de transmissão, mas pode-se transpor esta suposição através da medição da duração do sinal recebido.

Assumindo-se que dois nós estão dentro da área de transmissão do outro, a banda disponível no enlace entre estes dois nós será a menor banda disponível de todos os nós pertencentes a esta área de detecção.

Propõe-se o uso de mensagens de “hello” para se obter a banda disponível para os vizinhos de um salto somente. Cada nó envia a mensagem clássica de “hello” com uma extensão: Banda Disponível. Ela corresponde à banda disponível da fonte da mensagem de “hello”. Cada nó, recebendo esta

mensagem, armazena o valor de banda disponível na sua tabela de cachê de vizinhos.

A decisão que o nó toma de propagar as mensagens é baseada no mínimo da banda disponível do próprio nó e dos seus nós vizinhos para aquele salto. Desta forma, a banda adquirida é transparente para qualquer transmissão em andamento, dentro da área de detecção dos vizinhos de um salto (AUGUSTO; CERVEIRA & REZENDE, 2005).

2.6.2. Cálculo de Atraso e Flutuação

A variação do atraso em um sentido pode ser definida sobre dois pacotes i e k quaisquer, contidos no fluxo A=>B. A flutuação (dv_{ik}) é calculada pela diferença entre os atrasos unidirecionais d_k e d_i. Ao contrário do atraso, a flutuação, quando calculada para pacotes consecutivos (o que é o mais comum), independe da sincronização de fase dos relógios (i.e., se os relógios de ambos os hosts marcam a mesma hora). Ainda assim, o cálculo de flutuação depende da sincronização de freqüência dos relógios (i.e., se um relógio se atrasa ou adianta em relação a outro com o passar do tempo). De qualquer forma, o erro de sincronização na freqüência é de várias ordens de magnitude inferior ao valor medido e pode ser ignorado. Isso permite fazer cálculos de flutuação mesmo sem garantias de sincronização de relógios (BARBOSA, 2003).

Para explicar porque a sincronização não é necessária, sejam T_A^i e T_A^k o instante das saídas dos pacotes i e k, marcado pelo relógio do host A, e sejam T_B^i e T_B^k o instante de chegada dos mesmos pacotes, marcados pelo relógio do host B. Considere também que os relógios dos hosts A e B estão distantes, na fase, por unidades de tempo. Como já foi dito, a flutuação é calculada pela diferença entre os atrasos (BARBOSA, 2003):

$$dv^{ik} = d^k - d^i \qquad (2)$$

Cada atraso é calculado pela diferença entre o momento de chegada no host B e o momento de saída do host A, mais um erro dado pela diferença de fase dos relógios no instante da chegada do pacote (BARBOSA, 2003):

$$dv^{ik} = \left(T_B^k - T_A^k - \varepsilon^k\right) - \left(T_B^i - T_A^i - \varepsilon^i\right) \quad (3)$$

Admitindo-se que o cálculo é feito em pacotes adjacentes, tem-se que:

$$\varepsilon^k \cong \varepsilon^i \quad (4)$$

Portanto, a variação no atraso é dada pela equação (BARBOSA, 2003):

$$dv^{ik} = \left(T_B^k - T_A^k\right) - \left(T_B^i - T_A^i\right) \quad (5)$$

2.6.3. Cálculo de Perdas de Pacotes

O cálculo da perda de pacotes (P) pode ser definido como a percentagem de pacotes que chegam ao destino, sem que haja a necessidade de retransmissão. O cálculo pode ser feito através da equação (BARBOSA, 2003):

$$P = \left(\frac{\sum\left(E_{A\to B}\right) - \sum\left(R_B\right)}{\sum\left(E_{A\to B}\right)}\right) * 100\% \quad (6)$$

Onde $\sum\left(E_{A\to B}\right)$ é a quantidade de pacotes enviados do host A para o host B. O somatório de pacotes recebidos no host B é definido por $\sum\left(R_B\right)$.

3. MECANISMO DE AVALIAÇÃO DE QUALIDADE DE SERVIÇO (MAQS)

A avaliação do desempenho de uma aplicação que troca informações pela rede pode depender do serviço que está sendo utilizado pelo usuário da rede. Para alguns serviços, esse desempenho pode depender principalmente do sentido do tráfego de pacotes (de A para B, ou de B para A). Aplicações de tempo real, como vídeo sobre demanda e voz sobre IP são exemplos dessas aplicações.

O Mecanismo de Avaliação de Qualidade de Serviço (MAQS) em Redes de Computadores que foi desenvolvido é um software que faz uma análise dos parâmetros de QoS a fim de medir esses parâmetros, e permitir que o operador detecte possíveis falhas de desempenho que impliquem numa baixa qualidade das aplicações que estejam em operação na rede avaliada.

Para que possa fazer a avaliação, o MAQS foi dividido em dois módulos distintos:

- Módulo Farejador: módulo de captura de pacotes de rede;
- Módulo Analisador: módulo de análise de parâmetros de QoS.

A Figura 3.1 mostra o módulo Farejador com seu produto final, que é o histórico de pacotes capturados (Histórico de Pacotes). Também mostra o módulo Analisador e o seu insumo (o Histórico de Pacotes) e seus produtos (relatórios e gráficos).

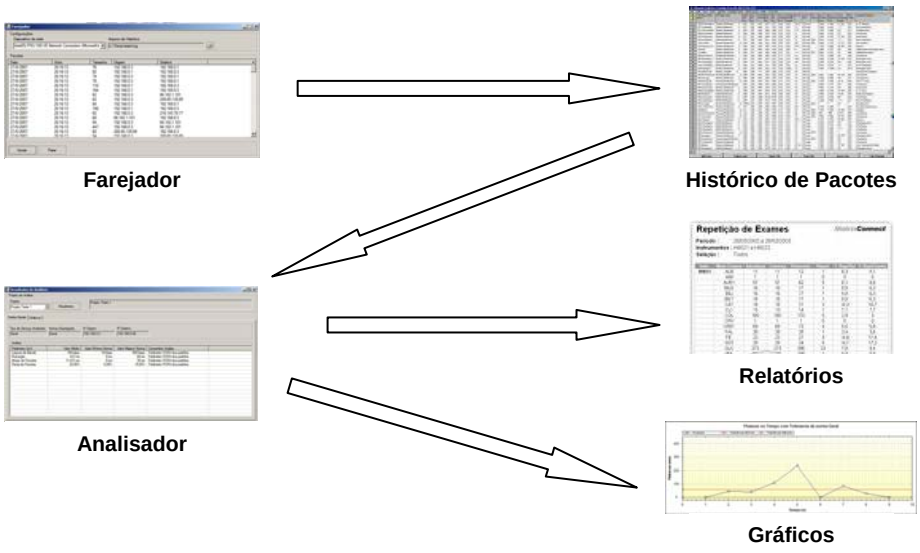


Figura 3-1- Interação entre os módulos do MAQS

A linguagem de programação utilizada para a construção é a C# do *framework*[8] *.Net* utilizando o *Visual Studio 2005* como ambiente de programação.

A construção do software utiliza também a API *SharpPcap*, que será apresentada na seção 3.1 – A API *SharpPcap*.

Para gerar os gráficos no software é utilizada a API Zedgraph, que será apresentada na seção 3.2 – A API Zedgraph.

3.1. A API SharpPcap

A API *libpcap* é uma biblioteca de funções para sistemas UNIX e sua versão Windows *winPcap* são as mais utilizadas bibliotecas para captura e análise de pacotes de rede. Vários softwares reconhecidos mundialmente utilizam a biblioteca *libpcap/winPcap* como base, como os softwares *TCPDump*[9]*, Ethereal*[10] e outros.

O propósito da API *SharpPcap* é prover um *framework* para capturar, injetar e analisar pacotes de rede para aplicações *.NET* baseado no *driver* de captura do *winPcap*.

Nas seções abaixo, são colocados exemplos de codificação das APIs utilizadas. O *Apêndice E - Código Fonte Completo do Software* detalha todo o código fonte do software e seus módulos.

3.1.1. Utilizando o SharpPcap

[8] Framework: pacotes de programas que capturam funcionalidades comuns a várias aplicações.

[9] TCPDump: Programa que captura pacotes de redes em uma interface de rede. Mais informações em http://www.tcpdump.org/.

[10] Ethereal: Programa de análise de redes. Mais informações em http://www.ethereal.com/.

Para que o MAQS possa capturar os pacotes de rede, é necessário, primeiramente, escolher um dispositivo de rede. A figura 3.2 demonstra como isto pode ser feito:

```
/* Lista todos dispositivos de rede do computador*/
foreach(PcapDevice dev in devices)
{
    /* mostra a descrição do disp. rede*/
    Console.WriteLine("{0}) {1}",i,dev.PcapDescription);
    Console.WriteLine();
    /* mostra o nome do disp. rede */
    Console.WriteLine("\tName:\t\t{0}",dev.PcapName);

    /*
    Se o dispositivo de rede é um dispositivo físico
    mostrar as informações avançadas
    */
    if(dev is NetworkDevice)
    {

        NetworkDevice netDev = (NetworkDevice)dev;
        /* Mostra as informações do disp.rede */
        Console.WriteLine("\tIP Address:\t\t{0}",
                                    netDev.IpAddress);
        Console.WriteLine("\tSubnet Mask:\t\t{0}",
                                    netDev.SubnetMask);
        Console.WriteLine("\tMAC Address:\t\t{0}",
                                    netDev.MacAddress);
        Console.WriteLine("\tDefault Gateway:\t{0}",
                                    netDev.DefaultGateway);
        Console.WriteLine("\tPrimary WINS:\t\t{0}",
                                    netDev.WinsServerPrimary);
        Console.WriteLine("\tSecondary WINS:\t\t{0}",
                                    netDev.WinsServerSecondary);
        Console.WriteLine("\tDHCP Enabled:\t\t{0}",
                                    netDev.DhcpEnabled);
    }
    Console.WriteLine();
    i++;
}
```

Figura 3-2 – Código Exemplo – Seleção de Dispositivo de Rede

Após selecionar o dispositivo de rede com o código da figura 3-2, é preciso ativar a captura dos pacotes e para que se possa gerar o histórico. A figura 3-3 demonstra como a captura de pacotes pode ser feita:

```
//Escolhe um dispositivo de rede da lista
PcapDevice device = devices[i];

//Registra a função onde serão feitos os procedimentos
//específicos com o pacote de rede capturado
device.PcapOnPacketArrival +=
  new SharpPcap.PacketArrivalEvent(device_PcapOnPacketArrival);

//Abre o dispositivo para iniciar a captura
//true – modo promíscuo da placa de rede
//1000 – captura um pacote de rede a cada 1000 ms
device.PcapOpen(true, 1000);

Console.WriteLine(
    "-- Listenning on {0}, hit 'Enter' to stop...",
    device.PcapDescription);

//Inicia o processo de captura
device.PcapStartCapture();

//Espera um ‘Enter’
Console.ReadLine();

//finaliza o processo de captura
device.PcapStopCapture();

//fecha o dispositivo de rede.
device.PcapClose();
```

Figura 3-3 – Código Exemplo – Captura de Pacotes

Estes procedimentos demonstrados acima foram utilizados na captura dos pacotes de rede pelo módulo Farejador.

Esta API foi escolhida por ser um software livre[11] e ser a única aplicação encontrada com as funcionalidades de captura de pacotes de rede disponível na linguagem de programação utilizada neste projeto.

3.2. A API Zedgraph

A API Zedgraph é um conjunto de funcionalidades para geração de gráficos 2D (em duas dimensões) em escrito em linguagem de programação C#. A figura 3-4 mostra o código para criar o gráfico apresentado na figura 3-5.

[11] Software Livre é o software disponível com a permissão para qualquer um usá-lo, copiá-lo, e distribuí-lo, seja na sua forma original ou com modificações, seja gratuitamente ou com custo.

```
   GraphPane myPane = zgc.GraphPane;

   // Entra com o titulo do gráfico e os rótulos dos eixos
   myPane.Title.Text = "Line Graph with Band Demo";
   myPane.XAxis.Title.Text = "Sequence";
   myPane.YAxis.Title.Text = "Temperature, C";

   // entra com dados randômicos
   double[] y = { 100, 115, 75, 22, 98, 40, 10 };
   double[] y2 = { 90, 100, 95, 35, 80, 35, 35 };
   double[] y3 = { 80, 110, 65, 15, 54, 67, 18 };
   double[] x = { 100, 200, 300, 400, 500, 600, 700 };

   // preenche o fundo dos eixos com uma cor em gradiente
   myPane.Chart.Fill = new Fill( Color.FromArgb( 255, 255, 245),
Color.FromArgb( 255, 255, 190), 90F );

   // Gera a curva vermelha “Curve 1”
   LineItem myCurve = myPane.AddCurve( "Curve 1", x, y, Color.Red );
   // Make the symbols opaque by filling them with white
   myCurve.Symbol.Fill = new Fill( Color.White );
   // Gera a curva azul “Curve 2”
   myCurve = myPane.AddCurve( "Curve 2", x, y2, Color.Blue );
   // Make the symbols opaque by filling them with white
   myCurve.Symbol.Fill = new Fill( Color.White );
   // Gera a curva verde “Curve 3”
   myCurve = myPane.AddCurve( "Curve 3", x, y3, Color.Green );
   // Make the symbols opaque by filling them with white
   myCurve.Symbol.Fill = new Fill( Color.White );

   // entra com a range de valores do eixo X
   myPane.XAxis.Scale.Min = 0;  myPane.XAxis.Scale.Max = 800;
   // mostra as linhas e grades do eixo Y
   myPane.YAxis.MajorGrid.IsVisible = true;
   myPane.YAxis.MinorGrid.IsVisible = true;

   // desenha um quadrado com uma range de valores
   BoxObj box = new BoxObj( 0, 100, 800, 30, Color.Empty,
         Color.FromArgb( 150, Color.LightGreen ) );
   box.Fill = new Fill( Color.White, Color.FromArgb( 200,
Color.LightGreen ), 45.0F );
   box.ZOrder = ZOrder.E_BehindAxis;
   myPane.GraphObjList.Add( box );

   // Adiciona um texto com o nome da range de valores
   TextObj text = new TextObj( "Optimal\nRange", 750, 85,
CoordType.AxisXYScale,
                     AlignH.Right, AlignV.Center );
   text.FontSpec.Fill.IsVisible = false;
   text.FontSpec.Border.IsVisible = false;
   myPane.GraphObjList.Add( text );

   // Calcula os dados do eixo
   zgc.AxisChange();
```

Figura 3-4 – Código Exemplo – Gerando Gráfico com Banda

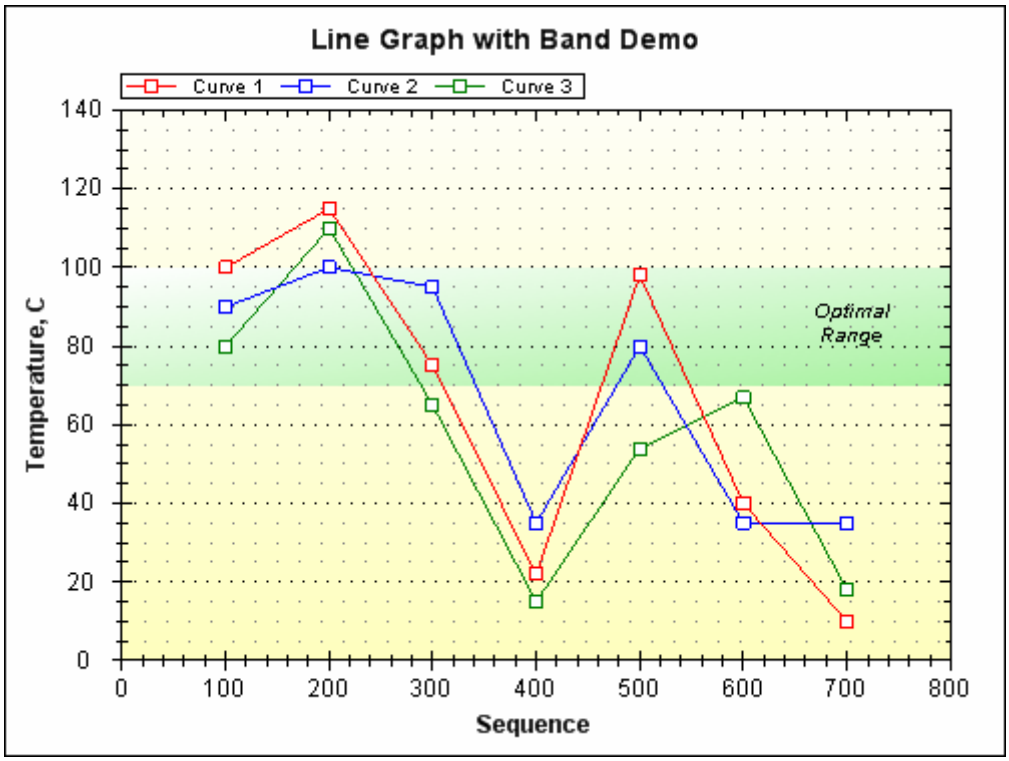


Figura 3-5 – Gráfico Exemplo – Gerando Gráfico com Banda

Esta API foi escolhida por ser um software livre e os conhecimentos de sua utilização já serem do conhecimento do autor deste projeto.

3.3. Módulo Farejador

Este módulo é responsável, única e exclusivamente, pela captura dos pacotes de rede e seu armazenamento.

O módulo ficará “escutando” o dispositivo de rede a espera da chegada de pacotes. Quando um pacote chega ao computador, o software captura uma cópia deste pacote e grava em um arquivo texto escolhido pelo usuário.

O Farejador é executado nos computadores onde há o interesse em se verificar a qualidade de serviço da rede analisada, durante a execução da aplicação a ser testada. O histórico gerado será utilizado para a análise dos parâmetros de QoS da rede de dados no módulo Analisador.

Este módulo deve ser usado concomitantemente nos dois microcomputadores onde estará sendo executada a aplicação a ser analisada.

Por exemplo: no caso de uma aplicação VoIP, o analisador deve ser executado nos dois microcomputadores onde está sendo executado o aplicativo VoIP, e durante a execução do mesmo. Este procedimento fará com que o Farejador gere um histórico dos pacotes de redes trocados entre os microcomputadores que estão se comunicando pelo serviço de VoIP.

3.4. Módulo de Análise de QoS

O Analisador é responsável por fazer os cálculos dos parâmetros de QoS. Após a importação dos históricos coletados dos microcomputadores em análise, escolhe-se uma norma técnica cadastrada, e o Analisador calcula os parâmetros de QoS dos pacotes capturados, e compara os resultados obtidos com a norma técnica escolhida pelo operador do sistema, e aponta os parâmetros de QoS que não estão dentro desta norma.

3.4.1. A importação dos arquivos de histórico

Foi escolhido transferir os dados dos arquivos gerados pelo Farejador para um arquivo de banco de dados Access, pois o Access implementa de forma simplificada o acesso aos dados de forma simultânea, e implementa facilidades de acesso aos dados como a programação em SQL[12] (Structured Query Language – Linguagem de Consulta Estruturada). O banco de dados escolhido poderia ter sido qualquer outro, como o IBM DB2, Microsoft SQL Server, Oracle 10g, Mysql, etc. O Access foi escolhido por não requerer instalação e estar bem integrado com o software de desenvolvimento utilizado no MAQS, o Microsoft Visual Studio, facilitando a implementação do software.

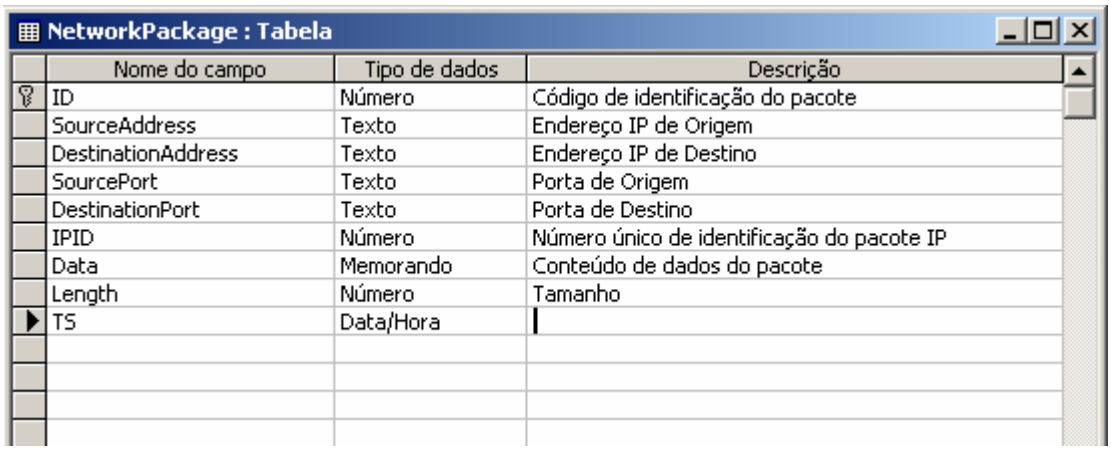
Os pacotes gravados no arquivo de histórico serão recuperados, utilizando a biblioteca *SharpPcap* conforme exemplificado na figura 3-6.

[12] O SQL é uma linguagem padronizada para a definição e manipulação de bancos de dados relacionais.

```
private static void device_PcapOnPacketArrival
        (object sender, Packet packet)
{
        // verifica se o pacote é Ethernet
        if( packet is EthernetPacket )
        {
        Console.WriteLine("Is Ethernet Packet");
        }
        // verifica se o pacote é TCP
        if (packet is TCPPacket)
        {
        Console.WriteLine("Is TCP Packet");
        }
        // verifica se o pacote é UDP
        if (packet is UDPPacket)
        {
        Console.WriteLine("Is UDP Packet");
        }
        // verifica se o pacote é IP
        if (packet is IPPacket)
        {
        Console.WriteLine("Is IP Packet");
        }
}
```

Figura 3-6 – Código Exemplo – Recuperação de Histórico de Pacotes de Rede

O pacote recuperado será gravado na tabela *NetworkPackage* representada pela figura 3-7 Os campos *SourcePort*, *DestinationPort*, *SourceAddress*, *DestinationAddress*, *IPID*, são necessários para identificação do pacote de rede, seu destino e origem. Já os campos *TS*, *Length* e *Data* são necessários para a geração de estatíscas da rede em estudo.



Nome do campo	Tipo de dados	Descrição
ID	Número	Código de identificação do pacote
SourceAddress	Texto	Endereço IP de Origem
DestinationAddress	Texto	Endereço IP de Destino
SourcePort	Texto	Porta de Origem
DestinationPort	Texto	Porta de Destino
IPID	Número	Número único de identificação do pacote IP
Data	Memorando	Conteúdo de dados do pacote
Length	Número	Tamanho
TS	Data/Hora	

Figura 3-7 – Tabela NetworkPackage

O código fonte da classe[13] que implementa o objeto *NetworkPackage*, pode ser encontrado no anexo *Apêndice E - Código Fonte Completo do Software*.

Para que se possa filtrar um determinado tipo de fluxo de pacotes relacionado a um serviço, o mesmo deverá ser feito utilizando os campos: porta de destino e origem, e endereço IP de origem e destino.

Após gravado na tabela *NetworkPackage*, o pacote já está pronto para ser analisado.

3.4.2. Análise de Pacotes

Os pacotes não podem ser analisados da forma como foram gravados no arquivo de histórico. Como existe interação manual dos operadores, tanto o início como o termino da gravação dos históricos não são sincronizadas. Assim haverão pacotes que não serão capturados pelo histórico neste período.

A solução encontrada para este problema foi analisar os pacotes dos históricos na faixa de tempo onde se podem observar atividades de envio e recepção de pacotes nas duas pontas da conexão, ou seja, tanto no host A quanto no host B.

Um exemplo desta situação é pode ser vista na figura 3-8. A linha azul e a linha vermelha demonstram o início de comunicação entre o host A e o host B respectivamente.

Podemos observar que a gravação dos históricos do host A (linha verde) e do host B (linha roxa) não tem o início e o final sincronizados. Para efeito de análise os pacotes gravados no histórico que serão analisados, são os pacotes que pertencem a ficha de tempo em que ambos os históricos foram gravados.

[13] Classes: Modelos de um objeto real (casa, cadeira, cliente, etc.), escritos em uma linguagem de programação que suporta a tecnologia de Orientação a Objetos, que definem as características ou ações que este objeto poderá executar.

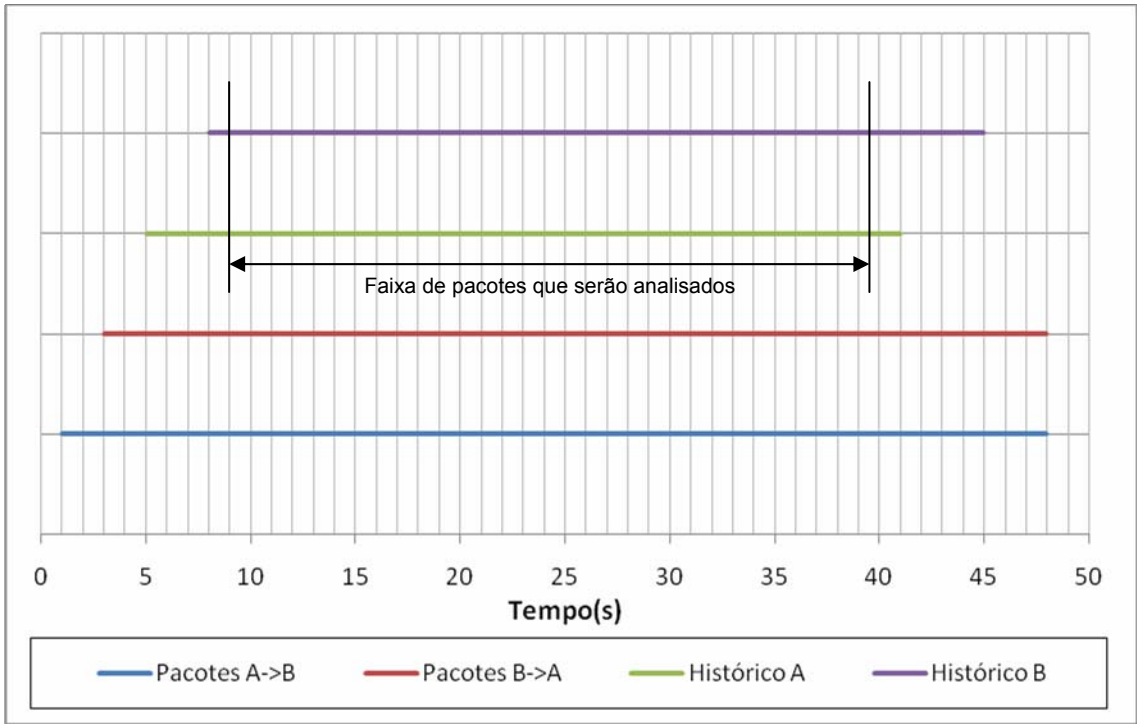


Figura 3-8 – Envio e gravação de pacotes em uma conexão de rede em função do tempo

3.4.3. Análise de Vazão

O módulo Analisador disponibiliza em gráfico onde poderão ser visualizados os seguintes parâmetros: Vazão total do fluxo A-> B, Vazão de um serviço, e vazão típica (mínimo e máximo) para o tipo de serviço.

O cálculo de vazão foi definido na seção 2.6.1 Cálculo de Vazão Utilizada. A faixa de valores de vazão típica para aplicações foi definida na seção 2.5 Aplicações de Redes e Parâmetros de QoS.

3.4.4. Análise de Perdas

O cálculo de Perdas de pacotes, assim como o de atraso e flutuação depende da análise dos pacotes nos dois computadores envolvidos na aplicação. Para tanto, é necessário que os computadores estejam com seus relógios sincronizados. O funcionamento desta sincronização será detalhado na seção 3.5 Sincronização dos relógios do fluxo.

No software é disponibilizado um gráfico onde poderão ser visualizados os parâmetros: Perdas de pacotes do Fluxo A->B, e perdas de pacotes típicas aceitáveis para o tipo de serviço especificado.

O cálculo de vazão foi definido na seção 2.6.3 Cálculo de Perdas de Pacotes. A facha de valores típica para perdas de pacotes aceitáveis para aplicações foi definida na seção 2.5 Aplicações de Redes e Parâmetros de QoS.

3.4.5. Análise de Atraso e Flutuação

Como definido na seção anterior, para o cálculo do atraso e flutuação de um determinado fluxo de pacotes, necessitamos da análise da chegada de pacotes nos dois computadores do fluxo.

Para ambos os cálculos são necessários que ambos os computadores estejam com seus relógios sincronizados. Esta sincronização é necessária, pois o cálculo do atraso e da flutuação leva em consideração a hora em que o pacote chegou ao computador. A sincronização dos relógios de ambos os computadores é tratada na seção 3.5 Sincronização dos relógios dos computadores do fluxo.

Os cálculos de atraso e flutuação já foram definidos na seção 2.6.2 Cálculo de Atraso e Flutuação. A facha de valores típica para atraso e flutuação de aplicações foi definida na seção 2.5 Aplicações de Redes e Parâmetros de QoS.

3.5. Sincronização dos relógios dos computadores do fluxo

Para o cálculo das métricas que são em função do tempo, como atraso e flutuação, é necessário ater-se à qualidade dos mecanismos de operação do relógio nas máquinas usadas para capturar o tráfego. A princípio, dado que os atrasos da rede são imprevisíveis, supõe-se que é absolutamente necessário garantir que os relógios das máquinas que irão capturar o tráfego de rede (os pacotes) estejam sincronizados.

3.5.1. Sincronização de relógios com o Network Time Protocol

O NTP (MILLS, 1996) faz os ajustes de fase e freqüência através de um disciplinador de relógio, que usa um loop[14] de ajuste de fase (PLL – *Phase-Lock Loop*) e um loop de ajuste de freqüência (FLL – *Frequency-Lock Loop*), que fazem ajustes mínimos no relógio o tempo todo (com o objetivo de melhorar a qualidade da hora fornecida pelo relógio de sistema em qualquer instante). Ambos são alimentados por dados obtidos através de requisições a relógios de referência na Internet e por dados obtidos do próprio relógio de sistema local, para comparação. Tendo-se uma boa conexão de rede até um relógio de referência, o NTP pode ser configurado para garantir um erro inferior a 1ms.

Mesmo com relógios locais de baixa qualidade, e sem nenhum mecanismo de sincronização, é possível ter garantias de erro máximo no cálculo de flutuação entre pacotes adjacentes, já que o erro na sincronização da freqüência (desvio) de um relógio local típico em relação a outro, para um intervalo de tempo suficientemente pequeno, é desprezível.

Um relógio comum tem um desvio inferior a 50 PPM, o que dá 50μs a cada segundo. Supondo-se que estamos avaliando pacotes adjacentes cujos intervalos de saída são inferiores a 100ms, e supondo que não teremos perda de pacotes adjacentes, teremos que o intervalo máximo entre dois pacotes analisados é de 200ms e assim o erro máximo possível na medição da flutuação é de 0,2 x 0,00005 = 0, 00001, ou seja, 10μs.

Se tivermos a certeza de que os relógios das máquinas que capturam o tráfego têm desvio inferior a 50 PPM e admitirmos que 10μs de erro, este erro no cálculo da flutuação é desprezível, em uma aplicação que envia pacotes em intervalos máximos de 100ms, não há necessidade de acrescentar nenhum mecanismo de sincronização de relógios para fazer o cálculo da flutuação no experimento.

Para o cálculo do atraso, não há como escapar da necessidade de sincronização dos relógios. Porém, em alguns casos o uso do NTP é suficiente para garantir um erro máximo aceitável. Obviamente, o uso de alguns relógios

[14] Loop (programação): processo executado várias vezes, até que se atinja um objetivo.

de alta confiabilidade para fazer experimentos de atraso é desejável, mas seu custo é muitas vezes proibitivo.

3.5.2. Instalando um Cliente NTP no Windows

Segundo o CAIS (Centro de Atendimento a Incidentes de Segurança), órgão pertencente à RNP (Rede Nacional de Ensino e Pesquisa), a melhor ferramenta para cliente de NTP para Windows é a ferramenta Dimension 4[15], pois é a de configuração mais intuitiva e completa, bem como de tamanho razoável para ser instalada nas máquinas clientes.

A instalação efetuada nos computadores de rede pode ser vista no apêndice F – Procedimento para instalação do cliente NTP.

3.6. Funcionamento dos módulos

Como visto anteriormente, os parâmetros de QoS flutuação e atraso são calculados com base no momento em que os pacotes deixam a máquina de origem, e chegam na máquina de destino. Por este motivo, teremos que instalar o Farejador em ambas as máquinas, e repassar os dados para um computador que servirá de servidor, onde funcionará o Analisador.

A figura 3-10 mostra o esquema de funcionamento do software com todos os elementos de rede que podem ser utilizados sendo mostrados. Os elementos **a** e **d** são os computadores onde serão instalados os módulos Farejadores.

15 Dimension 4: Software de sincronização de relógios de sistemas operacionais produzido pela empresa Thinking Man Software. Mais informações em http://www.thinkman.com/dimension4/.

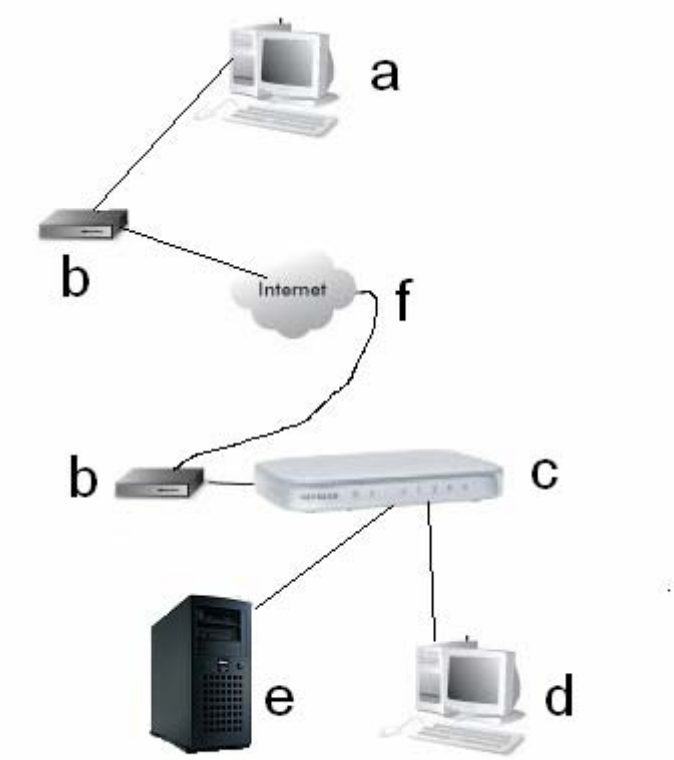


Figura 3-9 - Esquema de funcionamento do software em rede/internet

O elemento **e** é o servidor, onde estará instalado o módulo Analisador. O elemento **c** representa um roteador interno de uma rede. Os elementos **b** são modens ADSL.

Neste esquema está representado um computador com acesso a internet se comunicando com outro que esta inserida em uma rede corporativa e também tem acesso a internet.

O esquema representado aqui é somente uma suposição. Os dois computadores que estão se comunicando podendo estar perfeitamente inseridos cada um em uma rede corporativa acessando a internet, ou estarem até mesmo dentro da mesma rede. Já o módulo Analisador não necessita estar em um servidor, podendo ser instalado em uma das maquinas onde estará o Farejador.

3.7. Modelo de Dados

A figura 3-10 abaixo representa o modelo de dados do software Analisador.

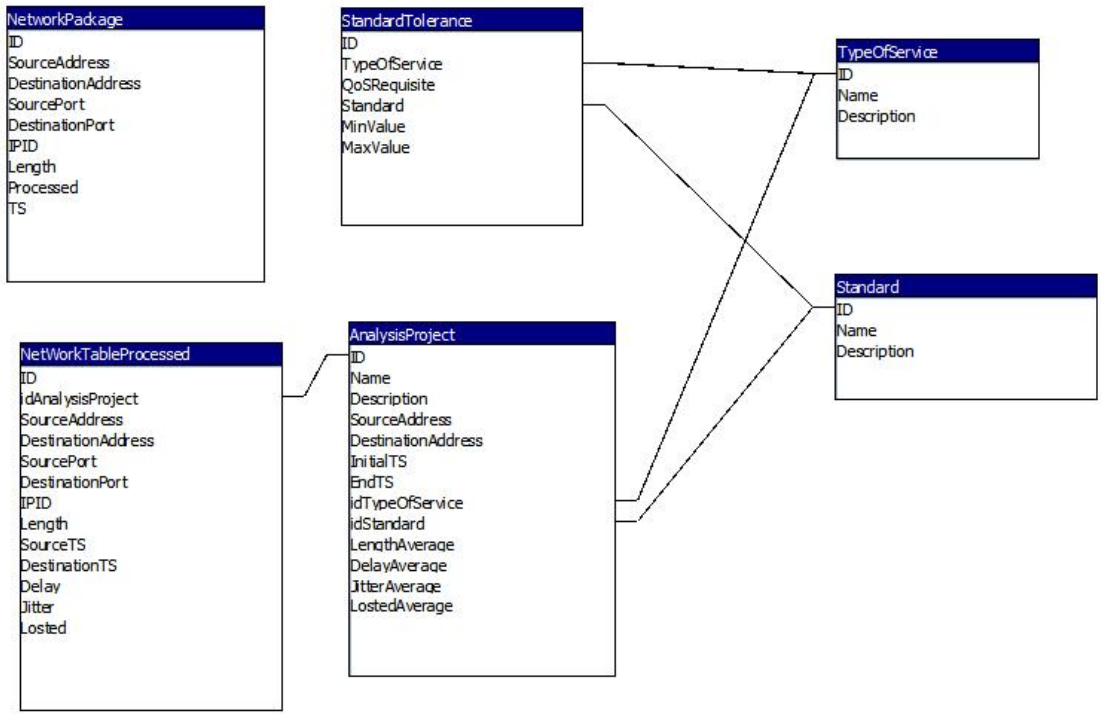


Figura 3-10 – Modelo de Dados do Analisador

Na tabela *NetworkPackage* são gravados os históricos de pacotes quando são importados para o Analisador. Ao se processar a análise, os pacotes processados e analisados são gravados na tabela *NetworkPackageProcessed*, e as médias e normas utlizadas na tabela *AnalysisProject*.

A tabela *TypeofService*, guarda os tipos de serviços cadastrados no software. A tabela *Standard* mantém o histórico de normas técnicas gravados no sitema, e os limites desta norma são armazenados na tabela *StandardTolerance*.

4. ESTUDOS DE CASO

Neste capitulo serão apresentados alguns estudos de caso que ajudaram a avaliar o funcionamento do MAQS.

Serão apresentados dois estudos de caso. O primeiro é o estudo de uma conversa através do serviço VoIP com dois computadores conectados pela internet. O segundo é o estudo de uma mesma conversa VoIP, mas com um dos computadores da conexão estará com sua largura de banda limitada.

4.1. Estudo de Caso 1: Comunicação VoIP

Neste estudo de caso é analisada uma conversa de aproximadamente 20 segundos feita por duas estações conectadas via internet pelo serviço de rede VoIP. Esta conversa é feita sem nenhuma interferência, ou seja, não existe nenhum outro processo acessando os serviços de rede de nenhuma das duas estações.

O serviço de rede VoIP é fornecido pelo programa *Skype* na sua versão 2.0 em ambas as estações que é produzido pela empresa *Skype Limited*.

A conexão de internet da estação A (IP 200.193.229.70) é uma conexão de 300 kbps e a conexão da estação B (IP 189.6.84.236) de 600 kbps. Com esta configuração podemos considerar a conexão entre as duas estações de 300 kbps.

A qualidade da conversa percebida pelos operadores do MAQS foi um pouco ruim, mas transcorreu de forma audível. Esta dificuldade percebida pode ser justificada pela conexão de Internet comum muito baixa, pois as operadoras de conexão de Internet não garantem a largura de banda nominal de contrato, e sim uma parte dela.

O esquema de conexão de rede deste estudo pode ser observado na figura 4-1.

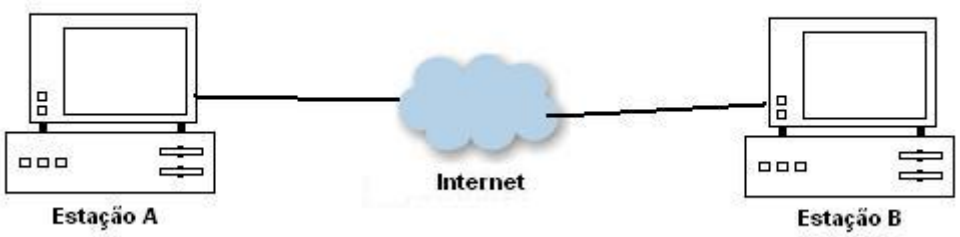


Figura 4-1 – Esquema de rede do estudo de caso 1

O objetivo deste estudo de caso é demonstrar uma conversa via VoIP que esteja com seus parâmetros de QoS dentro dos limites estabelecidos pela norma ITU-T apresentada na seção *2.5.2 A norma ITU-T*.

4.1.1. Análise do estudo de caso 1

Na Figura 4-2 podemos observar o resumo da análise feita pelo MAQS para o estudo de caso 1. É Informado IP de origem e destino, e os valores calculados largura de banda, flutuação, atraso de pacotes e a perda de pacotes.

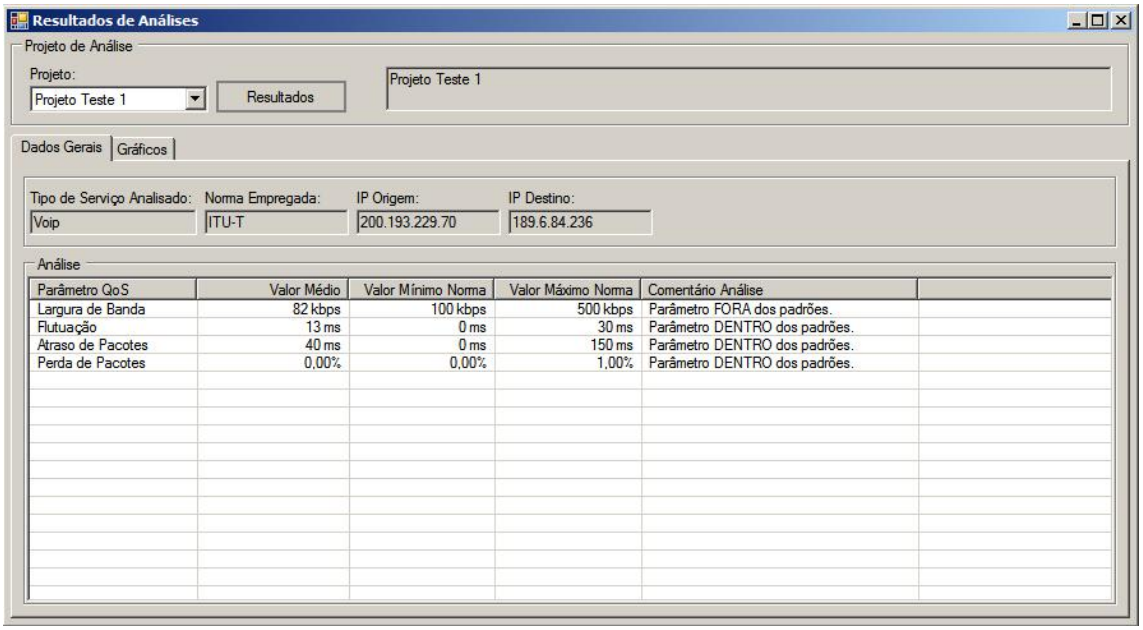


Figura 4-2 – Resultado da análise do estudo de caso 1

Pelo resultado da análise do estudo de caso 1, podemos observar que quase todos os parâmetros analisados se encontram dentro dos padrões estabelecidos pela norma ITU-T.

O único parâmetro que se encontra fora do padrão é a largura de banda. Este desvio pode ser explicado pela baixa qualidade da conexão de Internet fornecida pelas operadoras, como dito anteriormente.

Observamos que este mesmo parâmetro (largura de banda) fica relativamente próximo ao nível mais baixo da norma (indicado como uma conversa de boa qualidade). Desta forma, podemos justificar a qualidade percebida no transcorrer da conversa.

No gráfico da figura 4-3 podemos observar que em todo o período de análise da conversa, a largura de banda fica muito próxima do limite definido pela norma ITU-T como necessário para uma conversa com boa qualidade por VoIP, o que explica a qualidade de conversa percebida pelos operadores do MAQS.

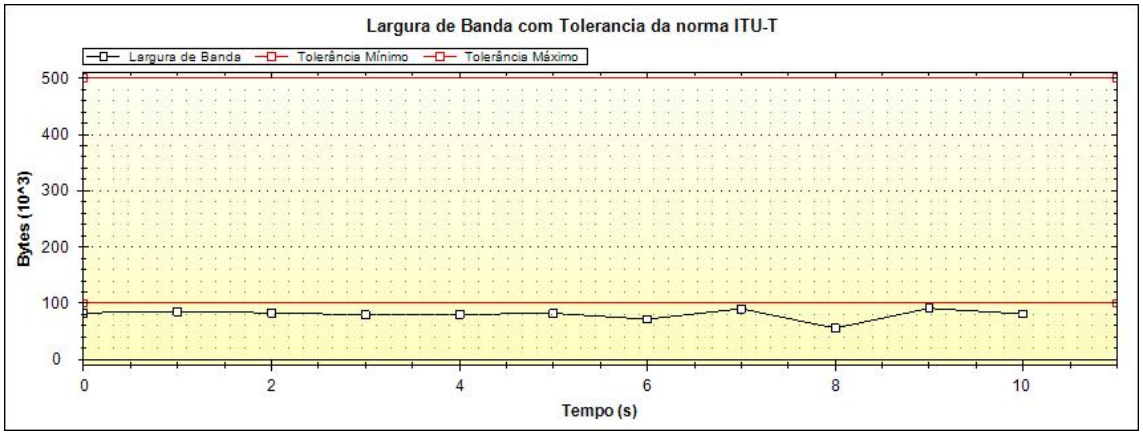


Figura 4-3 – Gráfico de vazão (largura de banda) do estudo de caso 1

Na figura 4-4, podemos observar que em todo o período de análise a flutuação se mantém dentro dos níveis definidos pela norma ITU-T.

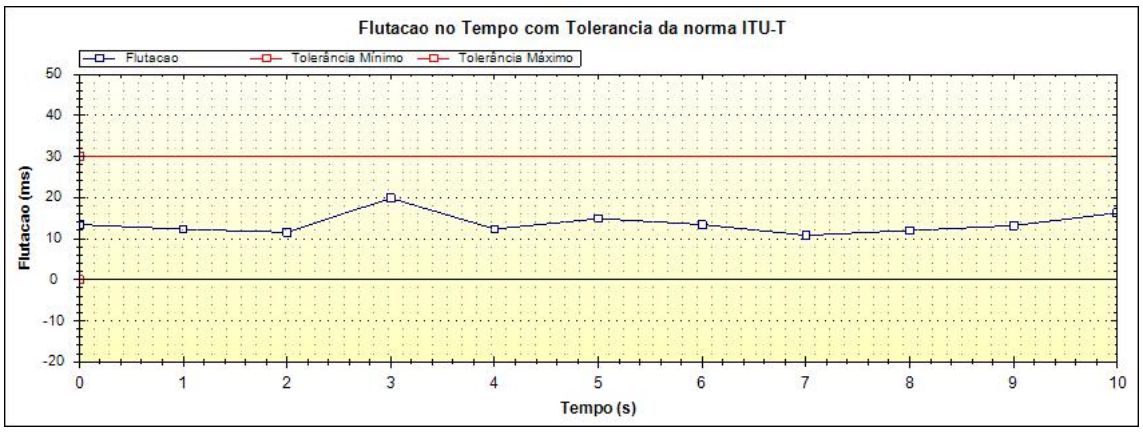


Figura 4-4 – Gráfico de flutuação do estudo de caso 1

No gráfico de atraso mostrado na figura 4-5 podemos observar que todo o período da análise, o atraso fica dentro dos limites definidos na norma ITU-T.

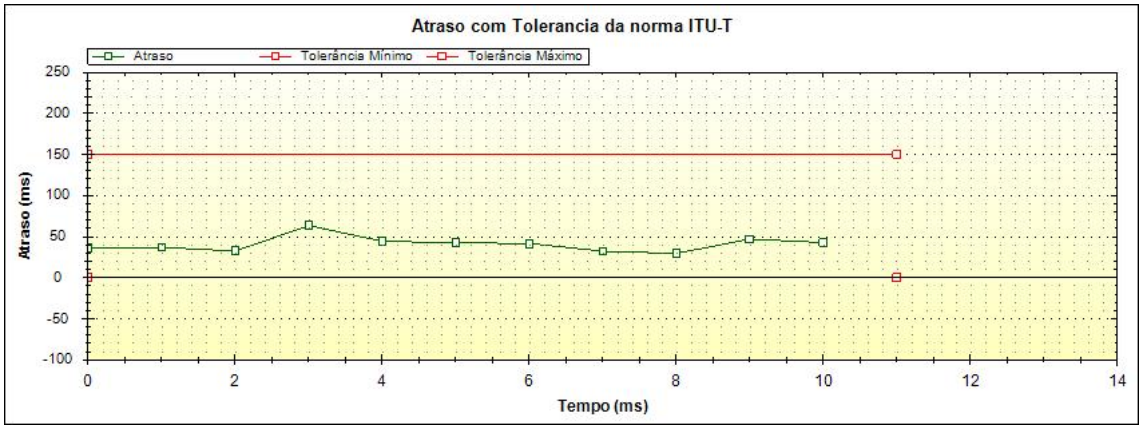


Figura 4-5 – Gráfico de atraso do estudo de caso 1

Podemos perceber tanto com o parâmetro flutuação quanto atraso, que os pacotes enviados chegam ao seu destino de forma rápida, facilitando a conversa via VoIP.

No gráfico de perdas mostrado na figura 4-6 podemos observar que quase não se vê perdas de pacotes de rede significativas em todo o período da análise. O nível de perdas apresentado está dentro dos limites definidos na norma ITU-T.

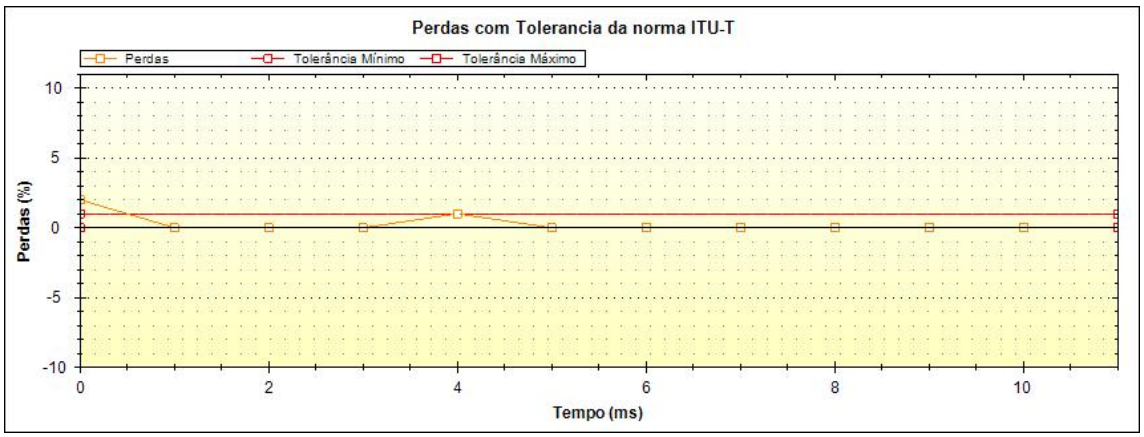


Figura 4-6 – Gráfico de perdas do estudo de caso 1

4.1.2. Conclusões do estudo de caso 1

Neste estudo de caso é analisada uma conversa VoIP transcorrida de forma normal com uma percepção dos operadores do MAQS de uma pequena de dificuldade para se manter a conversação.

Estas características do serviço analisado podem ser observadas pela análise feita pelo MAQS. A pequena dificuldade é justificada pela vazão que está um pouco abaixo do que a norma ITU-T considera como ideal para uma conversa VoIP ser considerada boa.

Pelo estudo de caso 1 pudemos observar que o MAQS pode ser bastante útil na análise de um serviço em uma aplicação prática.

4.2. Estudo de Caso 2: Comunicação VoIP com limitação de vazão

Neste estudo de caso é feita uma análise de uma comunicação VoIP com dificuldades. A configuração do esquema de rede é a mesma do estudo de caso 1.

A dificuldade na comunicação VoIP foi simulada com um download efetuado na estação A, diminuindo muito a vazão da conexão e tornando a conversa bem ruim.

O objetivo deste estudo é verificar se o MAQS reflete em sua análise a dificuldade verificada na conversa.

4.2.1. Análise do estudo de caso 2

Na Figura 4-7 podemos observar o resumo da análise feita pelo MAQS para o estudo de caso 2. É Informado IP de origem e destino, e os valores calculados largura de banda, flutuação, atraso de pacotes e a perda de pacotes.

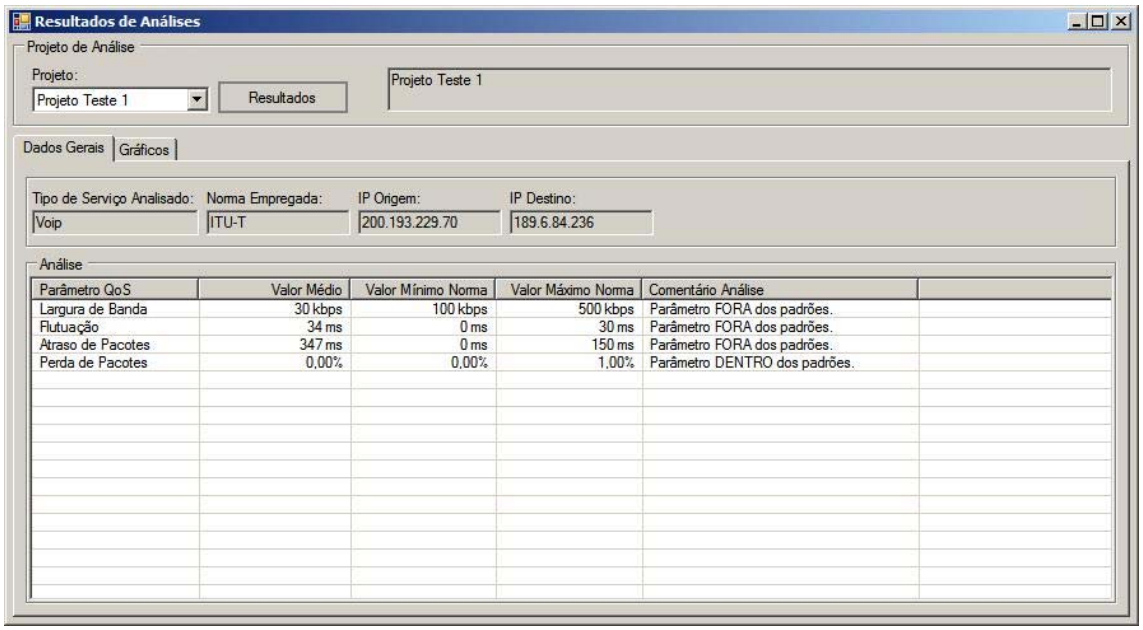


Figura 4-7 – Resultado da análise do estudo de caso 2

Podemos observar que a dificuldade gerada na simulação força quase todos os parâmetros de QoS ficarem fora dos padrões estabelecidos pela norma ITU-T.

Como a largura de banda é muito baixa (aproximadamente 30 kbps em média) os pacotes necessários para a comunicação demoram a chegar gerando um nível de atraso e flutuação muito alto.

No gráfico da figura 4-8 podemos observar que em todo o período de análise da conversa, a largura de banda fica muito baixa do limite definido pela norma ITU-T como necessário para uma conversa com boa qualidade por VoIP, configurando uma conversa muito ruim.

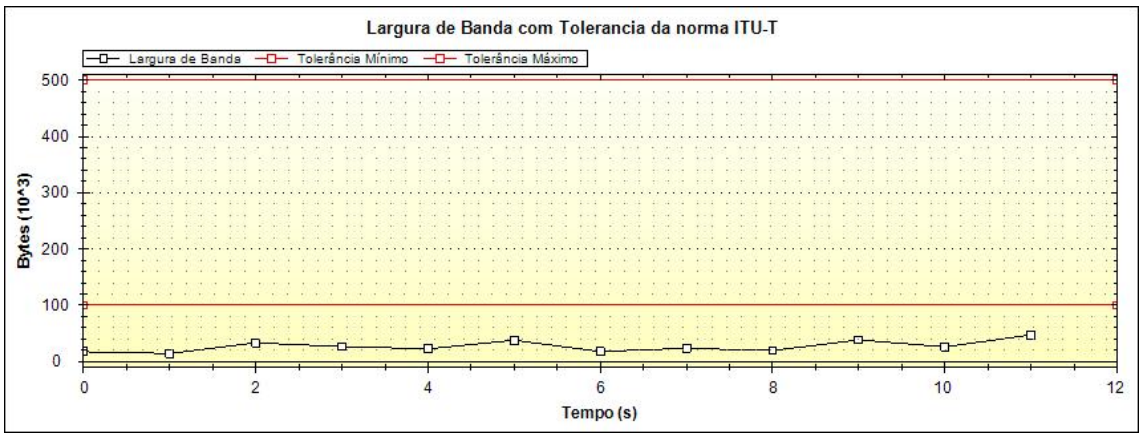


Figura 4-8 – Gráfico de vazão (largura de banda) do estudo de caso 2

Na figura 4-9, podemos observar que em todo o período de análise a flutuação se mantém muito próxima dos níveis definidos pela norma ITU-T, ultrapassando o limite em dois picos, graças ao gargalo gerado pelo reenvio de pacotes.

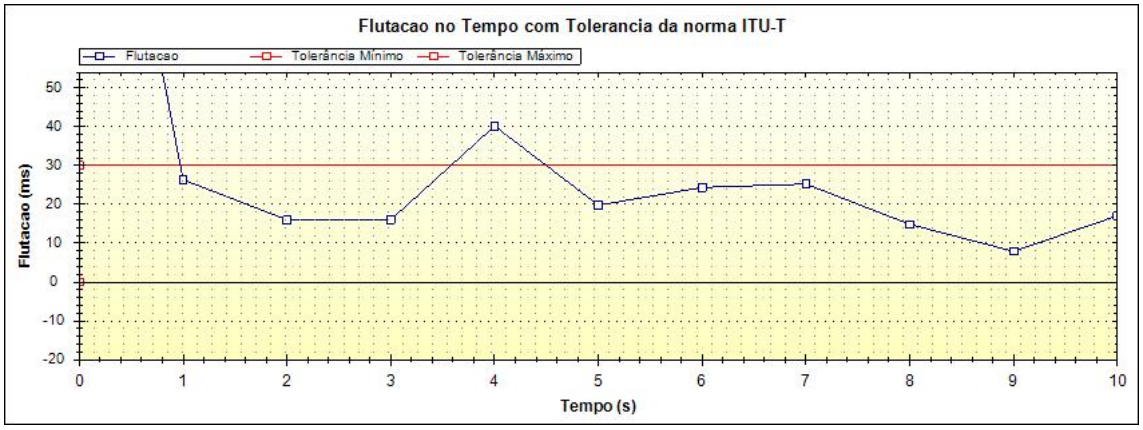


Figura 4-9 – Gráfico de flutuação do estudo de caso 2

No gráfico de atraso mostrado na figura 4-10 podemos observar que em quase todo o período da análise, o atraso fica fora dos limites definidos na norma ITU-T. Este atraso limita ainda mais a possibilidade da conversa se tornar audível.

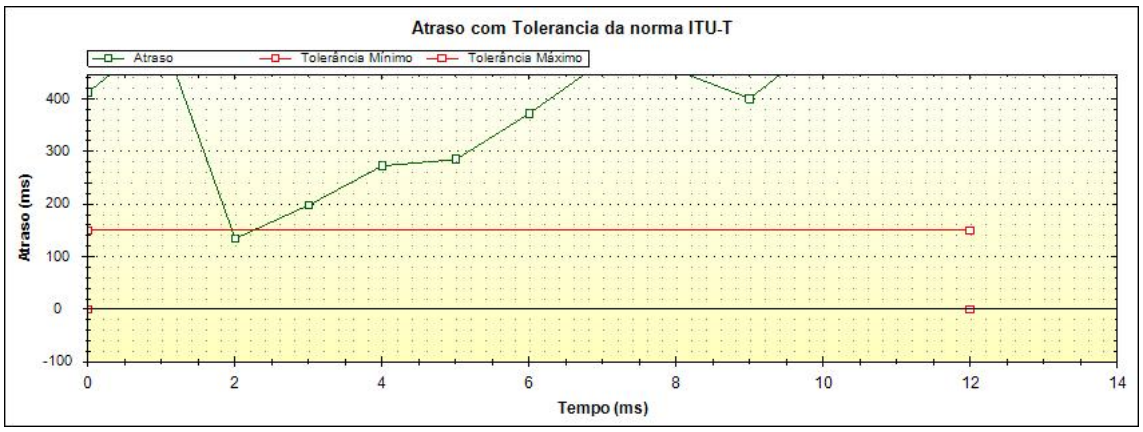


Figura 4-10 – Gráfico de atraso do estudo de caso 2

Podemos perceber tanto com o parâmetro flutuação quanto atraso, que os pacotes enviados chegam de forma muito lenta e desordenada ao seu destino, dificultando a conversa via VoIP.

No gráfico de perdas mostrado na figura 4-11 podemos observar que, excetuando-se o período inicial, todo o período da análise quase não se vê perdas de pacotes de rede. O nível de perdas apresentado está dentro dos limites definidos na norma ITU-T.

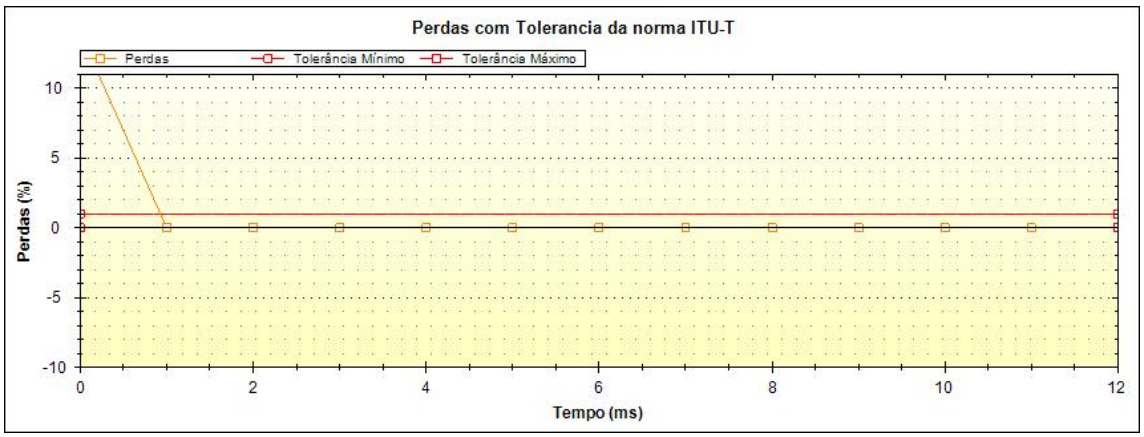


Figura 4-11 – Gráfico de perdas do estudo de caso 2

4.2.2. Conclusões do estudo de caso 2

Neste estudo de caso é analisada uma conversa VoIP com grandes dificuldades de comunicação, tornando-se inaudível.

Estas características do serviço analisado podem ser observadas pela análise feita pelo MAQS. A grande dificuldade observada na conversa é refletida nos parâmetros largura de banda, flutuação e atraso que estão bem fora do nível que a norma ITU-T considera como ideal para uma conversa VoIP ser considerada boa.

4.3. Conclusões dos Estudos de Caso

Com os estudos de caso apresentados pudemos notar que o MAQS reflete a forma como percebemos um serviço de rede de forma mensurável.

Desta forma, o MAQS pode ser utilizado como ferramenta de análise de serviço de redes, tornando a análise de problemas, que um serviço de rede possa vir a apresentar, menos empírica e baseadas em cálculos e comparações com normas.

5. CONCLUSÃO

5.1. Dificuldades Encontradas

As dificuldades enfrentadas na execução deste projeto serão descritas neste tópico, visando mostrar a experiência de implementação do projeto, contribuindo para execução de futuros projetos.

A primeira grande dificuldade, e mais relevante, neste trabalho foi encontrar material adequado para realização do mesmo. Existe muito material publicado sobre redes e qualidade de serviço. Mas material sobre o cálculo dos parâmetros de qualidade de serviço, que é o propósito básico deste trabalho, é muito escasso.

Esta dificuldade acabou atrasando em muito a finalização da idéia do projeto. Como exemplo pode-se citar que inicialmente o módulo Farejador iria ficar em uma única estação de trabalho com o driver de rede trabalhando em modo promíscuo, capturando os pacotes de rede. Mas ao se aprofundar o estudo verificou-se a necessidade de se ter o módulo nas duas pontas da conexão de rede, para que ao se fazer a análise o calculo dos parâmetros de qualidade de serviço esteja correto.

Nos primeiros testes, também houve um problema nos cálculos dos parâmetros de qualidade de serviço decorrente da natural falta de sincronismo do inicio e finalização da captura dos pacotes de rede. Para resolver este problema o módulo Analisador espera a estabilização da conexão (descarta uma quantidade no início e no fim da gravação dos históricos das duas estações de trabalho) para processar os cálculos, como descrito na seção *3.4.2 Análise de Pacotes*.

5.2. Resultados Obtidos

O principal resultado obtido é o Mecanismo de Avaliação de Qualidade de Serviço (MAQS). Este mecanismo é um software que se mostrou capaz de fazer cálculos de parâmetros de qualidade de QoS e pode ser útil para

avaliação de redes computacionais, conforme visto nos estudos de caso apresentados

Ao avaliar uma conexão entre dois computadores em rede o MAQS, mostrou-se capaz de analisar uma serviço de rede, e através desta análise pode-se chegar à conclusão de que uma rede não está adequada para suportar o serviço avaliado (seja Voip, conferência ou qualquer outro).

5.3. Considerações Finais

Uma análise de qualidade de serviço precisa que uma conexão de rede seja analisada em ambos os pontos da conexão, conforme pode ser visto no módulo de captura Farejador.

Para que se possa extrair dados confiáveis para a análise, é preciso que se trate somente a faixa de histórico em que as estações de trabalho estavam se comunicando, descartando-se as faixas de tempo onde o histórico não grava pacotes de ambas as estações.

O MAQS mostrou-se um instrumento útil para avaliar o desempenho de uma conexão de redes com relação a um determinado serviço utilizado nesta conexão.

Quando o serviço apresenta algum problema ou degradação de sua qualidade, o MAQS pode ajudar na verificação do problema, tornando o trabalho de descoberta do problema menos empírico, através dos cálculos de parâmetros de qualidade de serviço.

5.4. Sugestões para Trabalhos Futuros

Algumas evoluções do presente trabalho podem ser consideradas para execução de trabalhos futuros. Estas evoluções podem ser divididas em dois grupos: o primeiro trata de mudança de modelos do MAQS e a outra considera algumas melhorias sob a ótica do usuário final:

As sugestões de mudança no modelo do MAQS são:

- Alteração do MAQS para que trabalhe numa estação dedicada, capturando pacotes de uma rede e fazendo a análise desta rede de forma automática.
- Alteração do MAQS para que funcione em um servidor de internet, verificando a qualidade do serviço disponibilizado pelo servidor.

As melhorias sob a ótica do usuário final são:

- Melhora nas interfaces de usuários, deixando os usuários cadastrar seus parâmetros nas interfaces onde estes parâmetros serão necessários. Ex.: cadastro de projeto poderia ser feito na interface de análise do histórico.
- Possibilidade de exportação dos dados da análise em diversos formatos de arquivo como pdf, doc, txt, xls, etc...
- A forma como os dados analisados e os cálculos feitos pelo MAQS são armazenados pelo sistema não é confortável para usuário (base de dados Access). A gravação destes dados em um arquivo tornaria a análise mais amigável, e os dados desta análise poderiam ser recuperados e transportados de maneira mais prática.

REFERÊNCIAS BIBLIOGRÁFICAS

ALMQUIST P.. Type of Service in the Internet Protocol Suite. *Request for Comments.* [S.l.: s.n.], 1992.Disponível em:<http://www.ietf.org/rfc/rfc1349.txt>. Acesso em 25 de março de 2006.

AUGUSTO, Carlos Henrique Pereira; CERVEIRA, Carlos Rodrigo & REZENDE, José Ferreira de. *Controle de Admissão Adaptativo para Redes Ad Hoc 802.11*. Anais eletrônicos do XXII Simpósio Brasileiro de Telecomunicações, Campinas, Brasil. [s.n.], Setembro de 2006. Disponível em:< http://www.gta.ufrj.br/ftp/gta/TechReports/ACR05.pdf >. Acesso em 13 de fevereiro de 2006.

BARBOSA, Rodrigo dos Santos Bacelar Gouveia *Calculando Métricas Unidirecionais na Internet.* Universidade Federal de Pernambuco. [S.l.: s.n.], 2003. Disponível em:< http://www.cin.ufpe.br/~tg/2004-2/rsbgb.pdf >. Acesso em 12 de abril de 2006.

BOGER, Y., *"Fine-tuning Voice over Packet services", White Paper*, RADCOM Ltd. [S.l.: s.n.]. Disponível em: <http://www.protocols.com/papers/VoIP2.htm>. Acesso em 15 de Abril de 2006.

BRUM, Altamir, VOGT, Eide; GONÇALVES, Marta & MENDES, Alessandra da Silveira (2002). *QoS – Qualidade de Serviço em TCP/IP*. Disponível em: < http://celepar7cta.pr.gov.br/Celepar/SiteCel.nsf/47e8a350e40324b203256d430041e706/e8aeca1f4146cc8803256b9d00658d8f?OpenDocument >. Acesso em 8 de abril 2006.

CAIS - Centro de Atendimento a Incidentes de Segurança. *Implementando o serviço NTP na sua rede local.* São Paulo, Brasil. Agosto de 2000. Disponível em: < http://www.rnp.br/_arquivo/cais/manual_ntp_v1b.pdf >. Acesso em 27 de Abril de 2007.

CASTRO, Maria Cristina F. *Planejamento de Redes Comutadas*. Puc-RS – Pontifícia Universidade Católica do Rio Grande do Sul, 2001. Disponível em: < http://www.ee.pucrs.br/~decastro/pdf/Redes_Comutadas_Cap2_2.pdf>. Acesso em 12 de março de 2006.

CARVALHO, Tereza Cristina de Melo. *Arquitetura de Redes de Computadores OSI e TCP/IP*. Editora Makron Books, [S.I.], 1994. Co-edição: Embratel.

FONSECA, José Luiz A. da & STANTON, Michael. *Estudo experimental de videoconferência pessoal em inter-redes IP com QoS*. Universidade Federal Fluminense, 2004. Disponível em: <http://www.rnp.br/newsgen/0111/ jl_wtr.html>.Acesso em 15 de março de 2006.

ITU-T - International Telecomunications Union. *ITU-T Recommendation H.323*: Packet based multimedia communications systems. [S.I.] Fevereiro de 1996.

KAMIENSKI, Carlos Alberto. *Qualidade de Serviço na Internet*. [S.I., s.n.], 2000. Disponível em: < http://www.cin.ufpe.br/~cak/publications/apostila-minicurso-sbrc2000.pdf >. Acesso em 12 de março de 2006.

LIMA, Rosely dos Anjos; LAMARCA, Ézyo & OLIVEIRA, Affonso Henderson Guedes de. *Aplicações Multimídia: Desenvolvimento, Uso e Análise de Desempenho*. Universidade Federal do Pará [S.I.], Março de 2002. Disponível em: < http://www2.ufpa.br/rcientifica/pdf/ed_03_ral.pdf >. Acesso em 10 de Fevereiro de 2006.

LUNARDI, Sediane Carmem. *Uma Camada de Suporte à Qualidade de Serviço para Aplicações Multimídia na Internet*. Pontifícia Universidade Católica do Rio Grande do Sul [S.I.], 2001. Disponível em: < http://www.pucrs.br/inf/pos/ dissertacoes/arquivos/sediane.pdf >. Acessado em 25 de fevereiro de 2006.

MURHAMMER, Martin W.; ATAKAN, Orcun; BRETZ, Stefan; PUGH, Larry R.; SUZUKI, Kazunari & WOOD, David H.; *TCP/IP Tutorial e Técnico*. São Paulo: Makron Books do Brasil Editora Ltda. 2000.

MARTINS, J. *Qualidade de Serviço (QoS) em Redes IP – Princípios Básicos, Parâmetros e Mecanismos.* [S.l., s.n.]1999. Disponível em: < http://www.jsmnet.com/info_tecnica4.htm>. Acessado em 23 de Março de 2006.

MILLS, D. L., *"Simple Network Time Protocol (SNTP) Version 4 for IPv4, IPv6 and OSI"*, RFC 2030, Outubro de 1996.

PASSITO, Alexandre; MOTA, Edjair; QUEIROZ, Saulo; BEZERRA, Eduardo & GALVÃO, Eduardo. *Análise de desempenho de tráfego VoIP utilizando o Protocolo IP Security*. Universidade Federal do Amazonas. Disponível em: <http://inf.unisul.br/~ines/workcomp/cd/pdfs/2407.pdf>. Acessado em 5 de maio de 2006.

RNP – Rede Nacional de Ensino e Pesquisa. *Sobre o NTP.* [S.l., s.n.], 2000. Disponível em: <http://www.rnp.br/ntp/ntp-sobre.html>. Acessado em 05 de maio de 2006.

SILVA, Dinailton José da. *Análise de Qualidade de Serviço em Redes Corporativas*. [S.l., s.n.], 2004. Disponível em: <http://libdigi.unicamp.br/document/ ?code=vtls000359191>. Acesso em 13 de Junho de 2006.

STEVENS, W. Network Working Group. Request for Comments 2001. *TCP Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery Algorithms.* Janeiro 1997. Disponível em: <http://www.ietf.org/rfc/rfc2001.txt>. Acessado em 3 de Abril de 2006.

TANENBAUM, Andrew S. Computer Networks – 2ª Edition. Editora Prentice-Hall International Editions. [S.l.], 1989.

TANENBAUM, Andrew S. Redes de Computadores – 4ª Edição. Editora Elsevier, [S.l.], 2003.

VEGESNA, Srinivas. *IP Quality of Service*. Editora Cisco Press, [S.l.], 2001.

Apêndice A – Formato do pacote TCP

Cada mensagem TCP pode ser chamada de datagrama e consiste em duas partes: um cabeçalho e uma área de dados conforme pode ser vista na figura A-1.

<table>
<tr><td colspan="3">Porta de Origem</td><td>Porta de Destino</td></tr>
<tr><td colspan="4">Número de Seqüência</td></tr>
<tr><td colspan="4">Número de Reconhecimento</td></tr>
<tr><td>Offset</td><td>RES</td><td>CODE</td><td>Window</td></tr>
<tr><td colspan="3">Checksum</td><td>Urgent Pointer</td></tr>
<tr><td colspan="3">Opções</td><td>Padding</td></tr>
<tr><td colspan="4">Dados</td></tr>
</table>

Figura A-1 – Formato de um datagrama TCP [TAN03]

Os campos do cabeçalho podem são definidos da seguinte maneira:

1. Porta Fonte e Destino: estes campos no cabeçalho TCP contêm os números de portas TCP que identificam os programas de aplicação dos extremos de uma conexão.
2. Número de sequência (32 bits): identifica a posição no fluxo de bytes do segmento enviado pelo transmissor. O número de sequência refere-se ao fluxo de dados que vai na mesma direção do segmento.
3. Número de Reconhecimento(32 bits): este campo identifica a posição do byte mais alto (ou último byte) que o fonte recebeu. O número de reconhecimento refere-se ao fluxo de dados na direção contrária ao segmento. Os reconhecimentos sempre especificam o número do próximo byte que o receptor espera receber.
4. Offset: contém um inteiro que especifica o início da porção de dados do segmento. Este campo é necessário já que o campo Options varia em comprimento dependendo de quais opções tenham sido incluídas. De modo que o tamanho do cabeçalho TCP varia dependendo das opções selecionadas.
5. RES: reservado para uso futuro.

6. CODE(6 bits): determina o propósito e conteúdo do segmento, codificado assim:
    i. URG - Campo de ponteiro Urgente é válido
    ii. ACK - Campo de Reconhecimento é válido
    iii. PSH - Este segmento solicita um PUSH
    iv. RST - Reset da conexão
    v. SYN - Sincroniza numeros de sequências
    vi. FIN - O transmissor chega ao fim do fluxo de bytes.
7. WINDOW: através deste campo o software TCP indica quantos dados ele tem capacidade de receber em seu buffer.
8. URGENT POINTER: TCP através deste campo permite que o transmissor especifique que alguns dados são urgentes, isto significa que os dados serão expedidos tão rápido quanto seja possível.
9. OPTIONS: o software TCP usa este campo para se comunicar com o software do outro extremo da conexão.
10. CHECKSUM: é usado para verificar a integridade tanto do cabeçalho como dos dados do segmento TCP. (TANENBAUM, 2003)

Apêndice B - Formato do Cabeçalho do IPv4

Cada mensagem TCP pode ser chamada de datagrama e consiste em duas partes: um cabeçalho e uma área de dados conforme pode ser vista na figura B-1.

+	0-3	4-7	8-15	16-18	19-31
0	Versão	Tamanho cabeçalho	Type of Service (Tos)	Comprimento (pacote)	
32	Identificador			Flags	Offset
64	Time to Live (TTL)		Protocolo	Checksum	
96	Endereço origem				
128	Endereço destino				
160	Opções				
192	Dados				

Figura B-1 – Formato de um cabeçalho IPv4 (CARVALHO, 1994)

O primeiro campo do header (ou cabeçalho) de um datagrama IPv4 é o campo de version (ou versão) de 4 bits.

O segundo campo, de 4 bits, é o IHL (acrónimo para Internet Header Length, ou seja, Comprimento do Cabeçalho da Internet) com o número de words de 32 bits no cabeçalho IPv4. Como o cabeçalho IPv4 pode conter um número variável de opções, este campo essencialmente especifica o offset para a porção de dados de um datagrama IPv4. Um cabeçalho mínimo tem 20 bytes de comprimento, logo o valor mínimo em decimal no campo IHL seria 5.

No RFC 791, os 8 bits seguintes são alocados para um campo tipo de Serviço (ToS) agora DiffServ e ECN. A intenção original era para um host especificar uma preferência para como os datagramas poderiam ser manuseados assim que circulariam pela rede

O campo de 16 bits seguinte do IPv4 define todo o tamanho do datagrama, incluindo cabeçalho e dados, em bytes de 8 bits. O datagrama de tamanho mínimo é de 20 bytes e o máximo é 65535.

O campo seguinte de 16 bits é um campo de identificação. Este campo é usado principalmente para identificar fragmentos identificativos do datagrama IP original.

O campo de 3 bits que segue é usado para controlar ou identificar fragmentos.

O campo offset do fragmento tem 13 bits, e permite que um receptor determine o sítio de um fragmento em particular no datagrama IP original.

O campo TTL (time to live, ou seja, tempo para viver) ajuda a prevenir que os datagramas persistam (andem em círculos) numa rede. Ou seja, o campo TTL limita a vida de um datagrama em segundos.

Um campo de Protocolo de 8 bits segue-se. Este campo define o protocolo seguinte usado numa porção de dados de um datagrama, um exemplo é oTCP.

O campo seguinte é um campo de verificação (checksum) para o cabeçalho do datagrama IPv4.

A seguir ao campo de verificação, seguem-se os endereço de origem e de destino, de 32 bits cada um. Note que os endereços IPv6 de origem e destino são de 128 bits cada.(TANENBAUM, 1989)

Apêndice C – Formato do Cabeçalho UDP

Cada mensagem UDP pode ser chamada de datagrama e consiste em duas partes: um cabeçalho e uma área de dados conforme pode ser vista na figura C-1.

porta de origem	porta de destino
tamanho da mensagen	checksun
dados	
...	

Figura C-1 – Formato de um datagrama UDP (TANENBAUM, 1989)

Os campos porta de origem e porta de destino identificam os números das portas de serviço UDP. A porta de destino é obrigatória para que se conheça o serviço a ser entregue os dados. A porta de origem só é necessária quando respostas forem enviadas.

O campo tamanho da mensagem contém o número total de octetos do datagrama UDP, incluindo o cabeçalho e os dados. O campo checksun é opcional e contém os dados utilizados para o cálculo de detecção de erros. (TANENBAUM, 1989)

Apêndice D – Tabelas de Tarifas de Telecomunicações

O propósito deste apêndice é demonstrar que as tarifas de telefonia VoIP são mais baratas do que as de telefonia convencionais.

1. Tarifas de telefonia convencional

1.1. Brasil Telecom

A tabela D-1 mostra os preços praticados pela empresa Brasil Telecom no Distrito Federal em ligações locais praticadas até 15/07/2007.

Tabela D-1 – Tabela de tarifas para ligações locais - Brasil Telecom

Tipo de Ligação	**Tarifa com imposto**
Fixo-Fixo	R$ 0,15454
Fixo-Móvel VIVO	R$ 0,67463
Fixo-Móvel TIM	R$ 0,70248
Fixo-Móvel CLARO-BCP	R$ 0,73486
Fixo-Móvel NEXTEL	R$ 0,66297

A tabela D-2 mostra os preços praticados pela empresa Brasil Telecom em ligações interurbanas praticadas até 15/07/2007.

Tabela D-2 – Tabela de tarifas normal para ligações interurbanas - Brasil Telecom

Degraus - Km (Distância Geodésica)	***Tarifa com imposto***
D1 - Até 50	R$ 0,12796
D2 - Acima de 50 até 100	R$ 0,21330
D3 - Acima de 100 até 300	R$ 0,23816
D4 - Acima de 300	R$ 0,29376

1.2. Telefônica

A tabela D-3 mostra os preços praticados pela empresa Telefônica no estado de São Paulo em ligações locais praticadas até 15/07/2007, pela filial Telesp.

Tabela D-3 – Tabela de tarifas para ligações locais - Telefônica

Tipo de Ligação	**Tarifa com imposto**
Fixo-Fixo	R$ 0,14672
Fixo-Móvel VIVO	R$ 0,73227
Fixo-Móvel TIM	R$ 0,71332
Fixo-Móvel CLARO	R$ 0,72369
Fixo-Móvel NEXTEL	R$ 0,66297

A tabela D-4 mostra os preços praticados pela empresa Telefônica em ligações interurbanas praticadas até 15/07/2007 pela filial Telesp.

Tabela D-4 – Tabela de tarifas normal para ligações interurbanas - Telefônica

Degraus - Km (Distância Geodésica)	***Tarifa com imposto***
D1 - Até 50	R$ 0,10231
D2 - Acima de 50 até 100	R$ 0,15748
D3 - Acima de 100 até 300	R$ 0,22146
D4 - Acima de 300	R$ 0,29547

2. Tarifas de telefonia VoIP

2.1. Skype

A tabela D-5 mostra os preços praticados pela empresa Skype Limited em ligações locais e interurbanas praticadas até 15/05/2007.

Tabela D-5 – Tabela com tarifas locais e interurbanas Skype

Localidade	**Tarifa**	*Tarifa com imposto*
Brasil – Outras	R$	R$ 0.152

Localidades	0.132	
Brasil – Celular	R$ 0.519	R$ 0.597
Brasil - Rio de Janeiro	R$ 0.081	R$ 0.093
Brasil - São Paulo	R$ 0.063	R$ 0.072

2.2. LIG

A tabela D-6 mostra os preços praticados pela empresa LIG, filiada ao portal de Internet IG em ligações locais e interurbanas praticadas até 15/05/2007.

Tabela D-6 – Tabela com tarifas locais e interurbanas LIG

Localidade	*Tarifa com imposto*
Brasil – Outras Localidades	R$ 0.23
Brasil – Celular	R$ 0.92
Brasil - Rio de Janeiro	R$ 0.16
Brasil - São Paulo	R$ 0.16
Brasil – Brasília	R$ 0.27

Apêndice E – Código Fonte Completo do Software

1. A API ObjectLibrary

1.1. A Classe LogImporter

```
        using System;
        using System.Collections.Generic;
        using System.Text;
        using System.Windows.Forms;

        using Tamir.IPLib;
        using Tamir.IPLib.Packets;

        namespace ObjectLibrary
        {
            public class LogImporter
            {
                private PcapDevice _onLineDevice;
                private string _logFile;

                public PcapDevice OnLineDevice
                {
                    get { return _onLineDevice; }
                    set { _onLineDevice = value; }
                }

                public string LogFile
                {
                    get { return _logFile; }
                    set { _logFile = value; }
                }

                public LogImporter()
                {
                    try
                    {
                    }
                    catch (Exception ex)
                    {
                        throw new Exception("Erro ao instanciar objeto.", ex);
                    }
                }

                public void Close()
                {
                    try
                    {
                        if (_onLineDevice.PcapOpened)
                            _onLineDevice.PcapClose();

                    }
                    catch (Exception ex)
                    {
                        throw new Exception("Erro ao fechar farejador.", ex);
                    }
                }

                public void Open(int PackageCount)
                {
                    try
                    {
                        //if (!_onLineDevice.PcapOpened)
                        {
                            _onLineDevice = SharpPcap.GetPcapOfflineDevice(_logFile);
                            _onLineDevice.PcapOpen();

                            _onLineDevice.PcapOnPacketArrival += new
SharpPcap.PacketArrivalEvent(device_PcapOnPacketArrival);
                            _onLineDevice.PcapCapture(PackageCount);
```

```
                //_onLineDevice.PcapClose();
            }
        }
        catch (Exception ex)
        {
            throw new Exception("Erro ao fechar farejador.", ex);
        }
    }

    /// <summary>
    /// Dumps each received packet to a pcap file
    /// </summary>
    private void device_PcapOnPacketArrival(object sender, Packet packet)
    {
        try
        {
            NetworkPackage netPack = new NetworkPackage();

            if (packet is UDPPacket)
            {
                UDPPacket UDP = (UDPPacket)packet;
                netPack.SourceAddress = UDP.SourceAddress;
                netPack.SourcePort = UDP.SourcePort.ToString();
                netPack.DestinationPort = UDP.DestinationPort.ToString();
                netPack.DestinationAddress = UDP.DestinationAddress;
                netPack.IPID = UDP.Id;
                //netPack.TS = UDP.PcapHeader.Date;
                netPack.TS = UDP.Timeval.Date;
                netPack.Length = UDP.PcapHeader.PacketLength;
            }
            if (packet is TCPPacket)
            {
                TCPPacket TCP = (TCPPacket)packet;
                netPack.SourceAddress = TCP.SourceAddress;
                netPack.DestinationAddress = TCP.DestinationAddress;
                netPack.SourcePort = TCP.SourcePort.ToString();
                netPack.DestinationPort = TCP.DestinationPort.ToString();
                netPack.IPID = TCP.Id;
                netPack.TS = TCP.Timeval.Date;
                netPack.Length = TCP.PcapHeader.PacketLength;
            }
            if ((netPack.SourceAddress == "") &&
                (netPack.DestinationAddress == "") &&
                (netPack.SourcePort == "") &&
                (netPack.DestinationPort == ""))
            {
                return;
            }

            netPack.TS = packet.PcapHeader.Date;
            netPack.Length = Convert.ToDouble(packet.Bytes.Length);

            if (netPack.IPID != 0)
            {
                netPack.save();

                if (PackageCaptured != null)
                {
                    PackageCaptured(this, new PackageEventArgs(netPack));
                }
            }
        }
        catch (Exception ex)
        {
            if (ex.InnerException == null)
                MessageBox.Show(ex.Message);
            else
                MessageBox.Show(ex.InnerException.Message);
        }
    }

    /// <summary>
    /// Occurs when the body heigth overflow.
    /// </summary>
    public event PackageEventHandler PackageCaptured;
```

```
            }
        }
```

1.2. A Classe LogImporterDialog

```
using System;
using System.IO;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;

using Tamir.IPLib;
using Tamir.IPLib.Packets;

using ObjectLibrary.ErrorHandler;

namespace ObjectLibrary
{
    public partial class LogImporterDialog : Form
    {
        bool _importerRuns = false;
        private LogImporter _importer;
        delegate void SetTextCallback(ObjectLibrary.NetworkPackage netPack);

        public LogImporterDialog()
        {
            try
            {
                InitializeComponent();
                _importer = new LogImporter();
                _importer.PackageCaptured += new
PackageEventHandler(_importer_PackageCaptured);
                backgroundWorkerWriter.RunWorkerCompleted += new
RunWorkerCompletedEventHandler(backgroundWorkerWriter_RunWorkerCompleted);
            }
            catch (Exception ex)
            {
                throw new Exception("Error during instance object", ex);
            }
        }

        private void LogImporterDialog_Load(object sender, EventArgs e)
        {
            try
            {
            }
            catch (Exception ex)
            {
                ErrorHandlerDialog.ShowErroHandler(ref ex);
            }
        }

         private void btnStop_Click(object sender, EventArgs e)
        {
            try
            {
                //_importer.Close();
            }
            catch (Exception ex)
            {
                ErrorHandlerDialog.ShowErroHandler(ref ex);
            }
        }

        private void btnStart_Click(object sender, EventArgs e)
        {
            try
            {
```

```
                verifyLogFile();

                backgroundWorkerWriter.RunWorkerAsync();
            }
            catch (Exception ex)
            {
                ErrorHandlerDialog.ShowErroHandler(ref ex);
            }
        }

        void verifyLogFile()
        {
            try
            {
                _importer.LogFile = txtLogFile.Text;

                if ((_importer.LogFile == "") || (_importer.LogFile == null) )
                    throw new Exception("Selecione um Arquivo de Histórico!");

                if (_importerRuns)
                    throw new Exception("O importador já está executando. Por favor
Aguarde!");
            }
            catch (Exception ex)
            {
                throw new Exception("", ex);
            }
        }

        void _importer_PackageCaptured(object sender, PackageEventArgs e)
        {
            try
            {
                NetworkPackage netPack = e.NetPack;
                updateWinListPacket(netPack);
            }
            catch (Exception ex)
            {
                ErrorHandlerDialog.ShowErroHandler(ref ex);
            }
        }

        void updateWinListPacket(NetworkPackage netPack)
        {
            try
            {
                //this.Cursor = Cursors.WaitCursor;

                ListViewItem item = new ListViewItem(netPack.TS.ToShortDateString());
                item.SubItems.Add(netPack.TS.ToLongTimeString());
                item.SubItems.Add(netPack.IPID.ToString("#,##0"));
                item.SubItems.Add(netPack.Length.ToString());
                item.SubItems.Add(netPack.SourceAddress);
                item.SubItems.Add(netPack.DestinationAddress);

                if (this.lstPacket.InvokeRequired)
                {
                    SetTextCallback d = new SetTextCallback(updateWinListPacket);
                    this.Invoke(d, new object[] { netPack });
                }
                else
                {
                    lstPacket.Items.Add(item);
                }

                //this.Cursor = Cursors.Arrow;
            }
            catch (Exception ex)
            {
                //this.Cursor = Cursors.Arrow;
                throw new Exception("", ex);
            }
        }

        private void backgroundWorkerWriter_DoWork(object sender, DoWorkEventArgs e)
        {
```

```
            try
            {
                _importer.Open(SharpPcap.INFINITE);
                _importerRuns = true;
            }
            catch (Exception ex)
            {
                ErrorHandlerDialog.ShowErroHandler(ref ex);
            }
        }

        void backgroundWorkerWriter_RunWorkerCompleted(object sender,
RunWorkerCompletedEventArgs e)
        {
            _importerRuns = false;
        }

        private void btnOpenLogFile_Click(object sender, EventArgs e)
        {
            try
            {
                openFileDialog.Filter = "Log File|*.log";
                openFileDialog.Title = "Selecione um nome para o arquivo de histórico
para importar";
                openFileDialog.ShowDialog();

                if (openFileDialog.FileName != "")
                {
                    FileStream fs = (FileStream)openFileDialog.OpenFile();
                    txtLogFile.Text = fs.Name;
                    _importer.LogFile = fs.Name;
                    fs.Close();
                }
            }
            catch (Exception ex)
            {
                ErrorHandlerDialog.ShowErroHandler(ref ex);
            }
        }
    }
        }

        namespace ObjectLibrary
        {
            partial class LogImporterDialog
            {
                /// <summary>
                /// Required designer variable.
                /// </summary>
                private System.ComponentModel.IContainer components = null;

                /// <summary>
                /// Clean up any resources being used.
                /// </summary>
                /// <param name="disposing">true if managed resources should be disposed;
otherwise, false.</param>
                protected override void Dispose(bool disposing)
                {
                    if (disposing && (components != null))
                    {
                        components.Dispose();
                    }
                    base.Dispose(disposing);
                }

                #region Windows Form Designer generated code

                /// <summary>
                /// Required method for Designer support - do not modify
                /// the contents of this method with the code editor.
                /// </summary>
                private void InitializeComponent()
                {
                    this.groupBox1 = new System.Windows.Forms.GroupBox();
                    this.btnOpenLogFile = new System.Windows.Forms.Button();
                    this.label1 = new System.Windows.Forms.Label();
```

```
                    this.txtLogFile = new System.Windows.Forms.TextBox();
                    this.groupBox2 = new System.Windows.Forms.GroupBox();
                    this.lstPacket = new System.Windows.Forms.ListView();
                    this.columnHeader1 = new System.Windows.Forms.ColumnHeader();
                    this.columnHeader3 = new System.Windows.Forms.ColumnHeader();
                    this.columnHeader2 = new System.Windows.Forms.ColumnHeader();
                    this.columnHeader4 = new System.Windows.Forms.ColumnHeader();
                    this.columnHeader5 = new System.Windows.Forms.ColumnHeader();
                    this.btnStart = new System.Windows.Forms.Button();
                    this.backgroundWorkerWriter = new
System.ComponentModel.BackgroundWorker();
                    this.openFileDialog = new System.Windows.Forms.OpenFileDialog();
                    this.tableLayoutPanel1 = new System.Windows.Forms.TableLayoutPanel();
                    this.columnHeader6 = new System.Windows.Forms.ColumnHeader();
                    this.groupBox1.SuspendLayout();
                    this.groupBox2.SuspendLayout();
                    this.tableLayoutPanel1.SuspendLayout();
                    this.SuspendLayout();
                    //
                    // groupBox1
                    //
                    this.groupBox1.Controls.Add(this.btnOpenLogFile);
                    this.groupBox1.Controls.Add(this.label1);
                    this.groupBox1.Controls.Add(this.txtLogFile);
                    this.groupBox1.Dock = System.Windows.Forms.DockStyle.Fill;
                    this.groupBox1.Location = new System.Drawing.Point(3, 3);
                    this.groupBox1.Name = "groupBox1";
                    this.groupBox1.Size = new System.Drawing.Size(703, 61);
                    this.groupBox1.TabIndex = 2;
                    this.groupBox1.TabStop = false;
                    this.groupBox1.Text = "Configurações";
                    //
                    // btnOpenLogFile
                    //
                    this.btnOpenLogFile.FlatStyle = System.Windows.Forms.FlatStyle.Popup;
                    this.btnOpenLogFile.Image =
global::ObjectLibrary.Properties.Resources.open;
                    this.btnOpenLogFile.Location = new System.Drawing.Point(305, 31);
                    this.btnOpenLogFile.Name = "btnOpenLogFile";
                    this.btnOpenLogFile.Size = new System.Drawing.Size(23, 23);
                    this.btnOpenLogFile.TabIndex = 6;
                    this.btnOpenLogFile.UseVisualStyleBackColor = true;
                    this.btnOpenLogFile.Click += new
System.EventHandler(this.btnOpenLogFile_Click);
                    //
                    // label1
                    //
                    this.label1.AutoSize = true;
                    this.label1.Location = new System.Drawing.Point(6, 16);
                    this.label1.Name = "label1";
                    this.label1.Size = new System.Drawing.Size(102, 13);
                    this.label1.TabIndex = 8;
                    this.label1.Text = "Arquivo de Histórico";
                    //
                    // txtLogFile
                    //
                    this.txtLogFile.Location = new System.Drawing.Point(9, 33);
                    this.txtLogFile.Name = "txtLogFile";
                    this.txtLogFile.ReadOnly = true;
                    this.txtLogFile.Size = new System.Drawing.Size(290, 20);
                    this.txtLogFile.TabIndex = 7;
                    this.txtLogFile.Text = "c:\\Temp\\teste.log";
                    //
                    // groupBox2
                    //
                    this.groupBox2.Controls.Add(this.lstPacket);
                    this.groupBox2.Dock = System.Windows.Forms.DockStyle.Fill;
                    this.groupBox2.Location = new System.Drawing.Point(3, 70);
                    this.groupBox2.Name = "groupBox2";
                    this.groupBox2.Size = new System.Drawing.Size(703, 274);
                    this.groupBox2.TabIndex = 3;
                    this.groupBox2.TabStop = false;
                    this.groupBox2.Text = "Pacotes";
                    //
                    // lstPacket
                    //
```

```
                    this.lstPacket.Columns.AddRange(new
System.Windows.Forms.ColumnHeader[] {
                    this.columnHeader1,
                    this.columnHeader3,
                    this.columnHeader6,
                    this.columnHeader2,
                    this.columnHeader4,
                    this.columnHeader5});
                    this.lstPacket.Dock = System.Windows.Forms.DockStyle.Fill;
                    this.lstPacket.Location = new System.Drawing.Point(3, 16);
                    this.lstPacket.Name = "lstPacket";
                    this.lstPacket.Size = new System.Drawing.Size(697, 255);
                    this.lstPacket.TabIndex = 0;
                    this.lstPacket.UseCompatibleStateImageBehavior = false;
                    this.lstPacket.View = System.Windows.Forms.View.Details;
                    //
                    // columnHeader1
                    //
                    this.columnHeader1.Text = "Data";
                    this.columnHeader1.Width = 133;
                    //
                    // columnHeader3
                    //
                    this.columnHeader3.Text = "Hora";
                    this.columnHeader3.Width = 115;
                    //
                    // columnHeader2
                    //
                    this.columnHeader2.Text = "Tamanho";
                    this.columnHeader2.Width = 65;
                    //
                    // columnHeader4
                    //
                    this.columnHeader4.Text = "Origem";
                    this.columnHeader4.Width = 139;
                    //
                    // columnHeader5
                    //
                    this.columnHeader5.Text = "Destino";
                    this.columnHeader5.Width = 151;
                    //
                    // btnStart
                    //
                    this.btnStart.Dock = System.Windows.Forms.DockStyle.Right;
                    this.btnStart.FlatStyle = System.Windows.Forms.FlatStyle.Popup;
                    this.btnStart.Location = new System.Drawing.Point(622, 350);
                    this.btnStart.Name = "btnStart";
                    this.btnStart.Size = new System.Drawing.Size(84, 26);
                    this.btnStart.TabIndex = 0;
                    this.btnStart.Text = "Importar";
                    this.btnStart.UseVisualStyleBackColor = true;
                    this.btnStart.Click += new System.EventHandler(this.btnStart_Click);
                    //
                    // backgroundWorkerWriter
                    //
                    this.backgroundWorkerWriter.DoWork += new
System.ComponentModel.DoWorkEventHandler(this.backgroundWorkerWriter_DoWork);
                    //
                    // tableLayoutPanel1
                    //
                    this.tableLayoutPanel1.ColumnCount = 1;
                    this.tableLayoutPanel1.ColumnStyles.Add(new
System.Windows.Forms.ColumnStyle(System.Windows.Forms.SizeType.Percent, 100F));
                    this.tableLayoutPanel1.Controls.Add(this.groupBox2, 0, 1);
                    this.tableLayoutPanel1.Controls.Add(this.btnStart, 0, 2);
                    this.tableLayoutPanel1.Controls.Add(this.groupBox1, 0, 0);
                    this.tableLayoutPanel1.Dock = System.Windows.Forms.DockStyle.Fill;
                    this.tableLayoutPanel1.Location = new System.Drawing.Point(0, 0);
                    this.tableLayoutPanel1.Name = "tableLayoutPanel1";
                    this.tableLayoutPanel1.RowCount = 3;
                    this.tableLayoutPanel1.RowStyles.Add(new
System.Windows.Forms.RowStyle(System.Windows.Forms.SizeType.Absolute, 67F));
                    this.tableLayoutPanel1.RowStyles.Add(new
System.Windows.Forms.RowStyle(System.Windows.Forms.SizeType.Percent, 100F));
                    this.tableLayoutPanel1.RowStyles.Add(new
System.Windows.Forms.RowStyle(System.Windows.Forms.SizeType.Absolute, 32F));
```

```
                    this.tableLayoutPanel1.Size = new System.Drawing.Size(709, 379);
                    this.tableLayoutPanel1.TabIndex = 4;
                    //
                    // columnHeader6
                    //
                    this.columnHeader6.Text = "IP ID";
                    this.columnHeader6.Width = 82;
                    //
                    // LogImporterDialog
                    //
                    this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);
                    this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
                    this.ClientSize = new System.Drawing.Size(709, 379);
                    this.Controls.Add(this.tableLayoutPanel1);
                    this.FormBorderStyle = System.Windows.Forms.FormBorderStyle.Fixed3D;
                    this.Name = "LogImporterDialog";
                    this.StartPosition =
System.Windows.Forms.FormStartPosition.CenterScreen;
                    this.Text = "Importador de Histórico";
                    this.Load += new System.EventHandler(this.LogImporterDialog_Load);
                    this.groupBox1.ResumeLayout(false);
                    this.groupBox1.PerformLayout();
                    this.groupBox2.ResumeLayout(false);
                    this.tableLayoutPanel1.ResumeLayout(false);
                    this.ResumeLayout(false);

                }

                #endregion

                private System.Windows.Forms.GroupBox groupBox1;
                private System.Windows.Forms.GroupBox groupBox2;
                private System.Windows.Forms.Button btnStart;
                private System.Windows.Forms.ListView lstPacket;
                private System.Windows.Forms.ColumnHeader columnHeader1;
                private System.ComponentModel.BackgroundWorker backgroundWorkerWriter;
                private System.Windows.Forms.ColumnHeader columnHeader3;
                private System.Windows.Forms.ColumnHeader columnHeader4;
                private System.Windows.Forms.ColumnHeader columnHeader5;
                private System.Windows.Forms.ColumnHeader columnHeader2;
                private System.Windows.Forms.Button btnOpenLogFile;
                private System.Windows.Forms.Label label1;
                private System.Windows.Forms.TextBox txtLogFile;
                private System.Windows.Forms.OpenFileDialog openFileDialog;
                private System.Windows.Forms.TableLayoutPanel tableLayoutPanel1;
                private System.Windows.Forms.ColumnHeader columnHeader6;

            }
        }
```

1.3. A Classe NetworkPackage

```
using System;
using System.Data;
using System.Data.OleDb;
using System.Collections.Generic;
using ObjectLibrary.Communication;

namespace ObjectLibrary
{
        public class NetworkPackage
    {

        #region Fields

        private DateTime m_TS;
               private double m_IPID;
               private string m_DestinationPort;
               private string m_SourcePort;
               private string m_DestinationAddress;
               private string m_SourceAddress;
```

```
        private double m_Length;
               private double m_ID;
        private bool m_Processed;

               //Estados do objeto
               private bool m_IsHardObject;  //Existe no BD
               private bool m_IsSavedObject; //Atualizado no BD

        #endregion

        #region Constructors

        public NetworkPackage()
               {
                      try
                      {
                             clear();
                      }
                      catch (Exception ex)
                      {
                             Exception tmpEx = new Exception("Erro ao carregar objeto",
ex);
                             throw tmpEx;
                      }
               }

               #endregion

               #region Properties

               //Todos objetos devem conter esta propriedade. Ela diz se o objeto existe
no banco de dados
               internal bool IsHardObject
               {
                      get
                      {
                             return m_IsHardObject;
                      }
               }

               //Todos objetos devem conter esta propriedade. Ela diz se o objeto está
salvo no banco de dados
               internal bool IsSavedObject
               {
                      get
                      {
                             return m_IsSavedObject;
                      }
               }

               public DateTime TS
               {
                      get
                      {
                             return m_TS;
                      }
                      set
                      {
                             m_TS = value;
                             m_IsSavedObject = false;
                      }
               }

               public double IPID
               {
                      get
                      {
                             return m_IPID;
                      }
                      set
                      {
                             m_IPID = value;
                             m_IsSavedObject = false;
                      }
               }
```

```
            public string DestinationPort
            {
                    get
                    {
                            return m_DestinationPort;
                    }
                    set
                    {
                            m_DestinationPort = value;
                            m_IsSavedObject = false;
                    }
            }

            public string SourcePort
            {
                    get
                    {
                            return m_SourcePort;
                    }
                    set
                    {
                            m_SourcePort = value;
                            m_IsSavedObject = false;
                    }
            }

            public string DestinationAddress
            {
                    get
                    {
                            return m_DestinationAddress;
                    }
                    set
                    {
                            m_DestinationAddress = value;
                            m_IsSavedObject = false;
                    }
            }

            public string SourceAddress
            {
                    get
                    {
                            return m_SourceAddress;
                    }
                    set
                    {
                            m_SourceAddress = value;
                            m_IsSavedObject = false;
                    }
            }

        public double Length
        {
            get
            {
                return m_Length;
            }
            set
            {
                m_Length = value;
                m_IsSavedObject = false;
            }
        }

            public double ID
            {
                    get
                    {
                            return m_ID;
                    }
                    set
                    {
                            m_ID = value;
                            m_IsSavedObject = false;
                    }
```

```
            }

        public bool Processed
        {
            get
            {
                return m_Processed;
            }
            set
            {
                m_Processed = value;
                m_IsSavedObject = false;
            }
        }

            #endregion

            #region Static Methods

            public static NetworkPackage findNetworkPackage(double ID)
            {
                    try
                    {
                            NetworkPackage obj = new NetworkPackage();

                            List<NetworkPackage> alNetworkPackages = new
List<NetworkPackage>();

                            string clausulaWhere="";
                            clausulaWhere =
                                    "   ID = " +  ID ;


                            alNetworkPackages = findAll(clausulaWhere);

                            if(alNetworkPackages.Count > 1)
                            {
                                    Exception tmpEx = new Exception("A procura retornou
mais de um objeto");
                                    throw tmpEx;
                            }
                            if(alNetworkPackages.Count == 1)
                                    obj = (NetworkPackage) alNetworkPackages[0];
                            return obj;
                    }
                    catch (Exception ex)
                    {
                            Exception tmpEx = new Exception("Erro ao procurar objeto",
ex);
                            throw tmpEx;
                    }
            }

        public static NetworkPackage findLastNetworkPackage(string sourceAddress)
        {
            try
            {
                NetworkPackage obj = new NetworkPackage();

                DataTable dt = new DataTable();

                string clausulaWhere = "";
                clausulaWhere =
                        "   sourceAddress = '" + sourceAddress + "' ";

                            string sql = " SELECT " +
                                         "  LAST(TS) as LastTS "+
                                         " FROM " +
                                         "  NetworkPackage" +
                                    " WHERE " + clausulaWhere;

                            OleDbCommand cmm = new OleDbCommand(sql,
DataBase.Connection, DataBase.Transaction);
                            OleDbDataAdapter da = new OleDbDataAdapter(cmm);
                            da.Fill(dt);
```

```
                string LastTS = "";
                DataRow dr = dt.Rows[0];
                if (dr["LastTS"].ToString() != "")
                {
                    LastTS = Convert.ToString(dr["LastTS"]);
                }

                List<NetworkPackage> alNetworkPackages = new List<NetworkPackage>();

                clausulaWhere = "   TS = '" + LastTS + "' ";

                alNetworkPackages = findAll(clausulaWhere);

                obj = (NetworkPackage)alNetworkPackages[0];

                return obj;
            }
            catch (Exception ex)
            {
                Exception tmpEx = new Exception("Erro ao procurar objeto", ex);
                throw tmpEx;
            }
        }

        public static NetworkPackage findFirstNetworkPackage(string sourceAddress)
        {
            try
            {
                NetworkPackage obj = new NetworkPackage();

                DataTable dt = new DataTable();

                string clausulaWhere = "";
                clausulaWhere =
                        "   sourceAddress = '" + sourceAddress + "' ";

                string sql = " SELECT " +
                        "  FIRST(TS) as FirstTS " +
                        " FROM " +
                        "  NetworkPackage" +
                        " WHERE " + clausulaWhere;

                OleDbCommand cmm = new OleDbCommand(sql, DataBase.Connection,
DataBase.Transaction);
                OleDbDataAdapter da = new OleDbDataAdapter(cmm);
                da.Fill(dt);

                string FirstTS = "";
                DataRow dr = dt.Rows[0];
                if (dr["FirstTS"].ToString() != "")
                {
                    FirstTS = Convert.ToString(dr["FirstTS"]);
                }

                List<NetworkPackage> alNetworkPackages = new List<NetworkPackage>();

                clausulaWhere = "   TS = '" + FirstTS + "' ";

                alNetworkPackages = findAll(clausulaWhere);

                obj = (NetworkPackage)alNetworkPackages[0];

                return obj;
            }
            catch (Exception ex)
            {
                Exception tmpEx = new Exception("Erro ao procurar objeto", ex);
                throw tmpEx;
            }
        }

        public static List<NetworkPackage> findNetworkPackages(string sourceAddress,
            string destinationAddress, double IPID, double IDOrigem)
        {
            try
            {
```

```
                NetworkPackage obj = null;

                List<NetworkPackage> alNetworkPackages = new List<NetworkPackage>();

                string clausulaWhere = "";
                clausulaWhere =
                        "     IPID = " + IPID + " " +
                        " AND sourceAddress = '" + sourceAddress + "' " +
                        //" AND sourcePort = '" + sourcePort + "' " +
                        " AND destinationAddress = '" + destinationAddress + "' " +
                        //" AND destinationPort = '" + destinationPort + "' " +
                        " AND ID <> " + IDOrigem + " ";


                alNetworkPackages = findAll(clausulaWhere, "IPID, TS");

                return alNetworkPackages;
            }
            catch (Exception ex)
            {
                Exception tmpEx = new Exception("Erro ao procurar objeto", ex);
                throw tmpEx;
            }
        }

               public static List<NetworkPackage> findNetworkPackages()
               {
                      try
                      {
                             string clausulaWhere = "";
                             List<NetworkPackage> alNetworkPackages = new
List<NetworkPackage>();

                             alNetworkPackages = findAll(clausulaWhere);
                             return alNetworkPackages;
                      }
                      catch (Exception ex)
                      {
                             Exception tmpEx = new Exception("Erro ao procurar objeto",
ex);
                             throw tmpEx;
                      }
               }

        public static List<NetworkPackage> findNetworkPackages_OrderByIPID(DateTime
initialTS, DateTime endTS,
            string sourceAddress, string destinationAddress)
        {
            try
            {
                string clausulaWhere = "";
                List<NetworkPackage> alNetworkPackages = new List<NetworkPackage>();

                endTS = endTS.AddSeconds(-5);
                initialTS = initialTS.AddSeconds(5);

                clausulaWhere = //" TS BETWEEN '" + initialTS.ToString() + "' " +
                    //" AND '" + endTS.ToString() + "' " +
                    //" AND sourceAddress = '" + sourceAddress + "' " +
                    " sourceAddress = '" + sourceAddress + "' " +
                    " AND destinationAddress = '" + destinationAddress + "' ";

                alNetworkPackages = findAll(clausulaWhere, " IPID, TS ");
                return alNetworkPackages;
            }
            catch (Exception ex)
            {
                Exception tmpEx = new Exception("Erro ao procurar objeto", ex);
                throw tmpEx;
            }
        }

        public static List<NetworkPackage> findNetworkPackages_OrderByTS(DateTime
initialTS, DateTime endTS,
            string sourceAddress, string destinationAddress)
        {
```

```
            try
            {
                string clausulaWhere = "";
                List<NetworkPackage> alNetworkPackages = new List<NetworkPackage>();

                endTS = endTS.AddSeconds(-5);
                initialTS = initialTS.AddSeconds(5);

                clausulaWhere = //" TS BETWEEN '" + initialTS.ToString() + "' " +
                    //" AND '" + endTS.ToString() + "' " +
                    //" AND sourceAddress = '" + sourceAddress + "' " +
                    " sourceAddress = '" + sourceAddress + "' " +
                    " AND destinationAddress = '" + destinationAddress + "' ";

                alNetworkPackages = findAll(clausulaWhere, " TS, IPID ");
                return alNetworkPackages;
            }
            catch (Exception ex)
            {
                Exception tmpEx = new Exception("Erro ao procurar objeto", ex);
                throw tmpEx;
            }
        }


        public static List<NetworkPackage> findNetworkPackagesExceptDestinationAddress(
            string sourceAddress, string destinationAddress)
        {
            try
            {
                string clausulaWhere = "";
                List<NetworkPackage> alNetworkPackages = new List<NetworkPackage>();

                clausulaWhere = //" TS like '" + date.ToString("dd/MM/yyyy") + "*' " +
                    " sourceAddress = '" + sourceAddress + "' " +
                    " AND destinationAddress <> '" + destinationAddress + "' ";

                alNetworkPackages = findAll(clausulaWhere);
                return alNetworkPackages;
            }
            catch (Exception ex)
            {
                Exception tmpEx = new Exception("Erro ao procurar objeto", ex);
                throw tmpEx;
            }
        }

        public static List<NetworkPackage> findNetworkPackagesExceptSourceAddress(
                string sourceAddress, string destinationAddress)
        {
            try
            {
                string clausulaWhere = "";
                List<NetworkPackage> alNetworkPackages = new List<NetworkPackage>();

                clausulaWhere = //" TS like '" + date.ToString("dd/MM/yyyy") + "*' " +
                    " sourceAddress <> '" + sourceAddress + "' " +
                    " AND destinationAddress = '" + destinationAddress + "' ";

                alNetworkPackages = findAll(clausulaWhere);
                return alNetworkPackages;
            }
            catch (Exception ex)
            {
                Exception tmpEx = new Exception("Erro ao procurar objeto", ex);
                throw tmpEx;
            }
        }

        public static List<string> findSourceAddresses()
        {
            try
            {
                string clausulaWhere = "";
                List<string> colString = new List<string>();
```

```
                colString = findGroup(clausulaWhere, "SourceAddress");
                return colString;
            }
            catch (Exception ex)
            {
                Exception tmpEx = new Exception("Erro ao procurar objeto", ex);
                throw tmpEx;
            }
        }

        public static List<string> findDestinationAddresses()
        {
            try
            {
                string clausulaWhere = "";
                List<string> colString = new List<string>();

                colString = findGroup(clausulaWhere, "DestinationAddress");
                return colString;
            }
            catch (Exception ex)
            {
                Exception tmpEx = new Exception("Erro ao procurar objeto", ex);
                throw tmpEx;
            }
        }


        public static void updateSourceAddress(string oldSourceAddress, string
newSourceAddress)
        {
            try
            {
                string sql = " UPDATE NetworkPackage SET" +
                      " SourceAddress = " + "'" + newSourceAddress + "'" + " " +
                      " WHERE SourceAddress = " + "'" + oldSourceAddress + "'";

                OleDbCommand cmm = new OleDbCommand(sql, DataBase.Connection,
DataBase.Transaction);
                int recordsAffected = cmm.ExecuteNonQuery();
            }
            catch (Exception ex)
            {
                Exception tmpEx = new Exception("Erro ao procurar objeto", ex);
                throw tmpEx;
            }
        }

        public static void updateDestinationAddress(string oldDestinationAddress, string
newDestinationAddress)
        {
            try
            {
                string  sql = " UPDATE NetworkPackage SET" +
                      " DestinationAddress = " + "'" + newDestinationAddress + "'" + " "
+
                      " WHERE DestinationAddress = " + "'" + oldDestinationAddress +
"'";

                OleDbCommand cmm = new OleDbCommand(sql, DataBase.Connection,
DataBase.Transaction);
                int recordsAffected = cmm.ExecuteNonQuery();
            }
            catch (Exception ex)
            {
                Exception tmpEx = new Exception("Erro ao procurar objeto", ex);
                throw tmpEx;
            }
        }

           #endregion

           #region Instance Methods

           protected void loadNetworkPackage(double ID)
           {
```

```
                        try
                        {
                                NetworkPackage obj = NetworkPackage.findNetworkPackage(ID);

                                this.load(
                                                obj.TS,
                          obj.Length,
                                                obj.IPID,
                                                obj.DestinationPort,
                                                obj.SourcePort,
                                                obj.DestinationAddress,
                                                obj.SourceAddress,
                                                obj.ID,
                          obj.Processed,
                                                obj.IsHardObject,
                                                obj.IsSavedObject);

                        }
                        catch (Exception ex)
                        {
                                Exception tmpEx = new Exception("Erro ao carregar objeto",
ex);
                                throw tmpEx;
                        }
                }

                public virtual void save()
                {
                        try
                        {
                                validateProperties();

                                if (m_IsHardObject == false)
                                {
                                        insert();
                                        m_IsSavedObject = true;
                                }
                                else
                                {
                                        if (m_IsSavedObject == false)
                                        {
                                                update();
                                        }
                                }
                        }
                        catch (Exception ex)
                        {
                                Exception tmpEx = new Exception("Erro ao salvar objeto",
ex);
                                throw tmpEx;
                        }
                }

                internal virtual void remove()
                {
                        try
                        {
                                int recordsAffected;
                                string sql="";

                                if (m_IsHardObject == false)
                                {
                                        Exception tmpEx = new Exception("O objeto não
existe no banco de dados!");
                                        throw tmpEx;
                                }

                                sql = " DELETE FROM " +
                                                "  NetworkPackage" +
                                                " WHERE " +
                                                "   ID = " +  ID ;

                                sql = sql +" AND TS = '" + m_TS.ToString("yyyyMMddHHmmss")
+ "' ";
                                OleDbCommand cmm = new OleDbCommand(sql,
DataBase.Connection);
```

```
                                recordsAffected = cmm.ExecuteNonQuery();

                                if (recordsAffected > 1)
                                {
                                        Exception tmpEx = new Exception("A atualização do
objeto afetou mais de um registro no banco de dados.");
                                        throw tmpEx;
                                }

                                if (recordsAffected == 0)
                                {
                                        Exception tmpEx = new Exception("Registro alterado
anteriormente. Consulte novamente para atualizar e repita a operação.");
                                        throw tmpEx;
                                }

                                clear();
                                m_IsHardObject = false;
                                m_IsSavedObject = false;
                        }
                        catch (Exception ex)
                        {
                                Exception tmpEx = new Exception("Objeto não foi criado no
banco de dados, portanto não pode ser excluído fisicamente.", ex);
                                throw tmpEx;
                        }
                }

                #endregion

                #region Private Methods

                private void validateProperties()
                {
                        try
                        {
                 if (m_DestinationPort == "")
                 {
                     Exception tmpEx = new Exception("DestinationPort não preenchido!");
                     throw tmpEx;
                 }
                 if (m_Length == 0)
                 {
                     Exception tmpEx = new Exception("Length não preenchido!");
                     throw tmpEx;
                 }
                 if (m_SourcePort == "")
                 {
                     Exception tmpEx = new Exception("SourcePort não preenchido!");
                     throw tmpEx;
                 }
                 if (m_DestinationAddress == "")
                 {
                     Exception tmpEx = new Exception("DestinationAddress não
preenchido!");
                     throw tmpEx;
                 }
                 if (m_SourceAddress == "")
                 {
                     Exception tmpEx = new Exception("SourceAddress não preenchido!");
                     throw tmpEx;
                 }

                 if (m_IPID == 0)
                 {
                     Exception tmpEx = new Exception("IP ID não preenchido!");
                     throw tmpEx;
                 }
                        }
                        catch (Exception ex)
                        {
                                Exception tmpEx = new Exception("Erro ao validar objeto",
ex);
                                throw tmpEx;
                        }
                }
```

```
            private void insert()
            {
                    try
                    {
                            string sql="";
                            long NextID = 0;

                            NextID = getNextID();

                            sql = " INSERT INTO" +
                                    "       NetworkPackage" +
                                    " (TS, Length, IPID, DestinationPort, SourcePort,
DestinationAddress, " +
                    "  SourceAddress, ID) " +
                                    " VALUES ( " +
                                     "'" + m_TS.ToString("dd/MM/yyyy hh:mm:ss:fff") +
"'" + "," +
                     m_Length.ToString() + "," +
                                      + m_IPID + "," +
                                     "'" + m_DestinationPort + "'" + "," +
                                     "'" + m_SourcePort + "'" + "," +
                                     "'" + m_DestinationAddress + "'" + "," +
                                     "'" + m_SourceAddress + "'" + "," +
                                     NextID + ")";

                     OleDbCommand cmm = new OleDbCommand(sql, DataBase.Connection,
DataBase.Transaction);
                            cmm.ExecuteNonQuery();

                            m_ID = NextID;

                            m_IsHardObject = true;
                            m_IsSavedObject = true;
                    }
                    catch (Exception ex)
                    {
                            Exception tmpEx = new Exception("Objeto não foi criado no
banco de dados", ex);
                            throw tmpEx;
                    }
            }

            private long getNextID()
            {
                    try
                    {
                            string sql="";
                            long NextID = 0;

                     DataTable dt = new DataTable();
                     sql = " SELECT MAX(NetworkPackage.ID) as LastID FROM NetworkPackage ";

                            OleDbCommand cmm = new OleDbCommand(sql,
DataBase.Connection);
                     OleDbDataAdapter da = new OleDbDataAdapter(cmm);
                            da.Fill(dt);

                            DataRow dr = dt.Rows[0];
                     if (dr["LastID"].ToString() != "")
                            {
                                    NextID = Convert.ToUInt32(dr["LastID"]);
                            }

                            NextID ++;
                            return NextID;

                    }
                    catch (Exception ex)
                    {
                            Exception tmpEx = new Exception("Erro ao procurar próximo
id", ex);
                            throw tmpEx;
                    }
            }
```

```
		private void update()
		{
			try
			{
				int recordsAffected;
				string sql="";

				sql = " UPDATE NetworkPackage SET" +
			" TS = " + "'" + m_TS.ToString("dd/MM/yyyy hh:mm:ss:fff") + "'" +
", " +
			" Length = " + m_Length + ", " +
					" IPID = " + m_IPID + ", " +
					" DestinationPort = " + "'" + m_DestinationPort +
"'" + ", " +
					" SourcePort = " + "'" + m_SourcePort + "'" + ", "
+
					" DestinationAddress = " + "'" +
m_DestinationAddress + "'" + ", " +
					" SourceAddress = " + "'" + m_SourceAddress + "'"
+ ", " +
			" Processed = " + Convert.ToInt16(m_Processed) + " " +
					" WHERE " +
						"   ID = " +  ID ;

				OleDbCommand cmm = new OleDbCommand(sql,
DataBase.Connection, DataBase.Transaction);
				recordsAffected = cmm.ExecuteNonQuery();

				if (recordsAffected > 1)
				{
					Exception tmpEx = new Exception("A atualização do
objeto afetou mais de um registro no banco de dados.");
					throw tmpEx;
				}

				if (recordsAffected == 0)
				{
					Exception tmpEx = new Exception("Registro alterado
anteriormente. Consulte novamente para atualizar e repita a operação.");
					throw tmpEx;
				}

				m_IsHardObject = true;
				m_IsSavedObject = true;

			}
			catch (Exception ex)
			{
				Exception tmpEx = new Exception("Objeto não foi atualizado
no banco de dados", ex);
				throw tmpEx;
			}
		}

		private void load(DateTime TS, double Length, double IPID, string
DestinationPort, string SourcePort,
		string DestinationAddress, string SourceAddress, double ID, bool Processed,
		bool isHardObject, bool isSavedObject)
		{
			try
			{
				m_TS = TS;
		 m_Length = Length;
				m_IPID = IPID;
				m_DestinationPort = DestinationPort;
				m_SourcePort = SourcePort;
				m_DestinationAddress = DestinationAddress;
				m_SourceAddress = SourceAddress;
				m_ID = ID;
		 m_Processed = Processed;
				m_IsHardObject = isHardObject;
				m_IsSavedObject = isSavedObject;

			}
			catch (Exception ex)
			{
```

```
                                Exception tmpEx = new Exception("Erro de conversão de tipos
de dados", ex);
                                throw tmpEx;
                        }
                }

                private void clear()
                {
                        try
                        {
                                m_TS = DateTime.Now;
                 m_Length = 0;
                                m_IPID = 0;
                                m_DestinationPort = "";
                                m_SourcePort = "";
                                m_DestinationAddress = "";
                                m_SourceAddress = "";
                                m_ID = 0;
                                m_IsHardObject = false;
                                m_IsSavedObject = false;
                        }
                        catch (Exception ex)
                        {
                                Exception tmpEx = new Exception("Erro ao limpar objeto",
ex);
                                throw tmpEx;
                        }
                }


                private static List<NetworkPackage> findAll(string clausulaWhere)
                {
                        try
                        {
                                return findAll(clausulaWhere, "ID");
                        }
                        catch (Exception ex)
                        {
                                Exception tmpEx = new Exception("Erro ao procurar objeto",
ex);
                                throw tmpEx;
                        }
                }

                private static List<NetworkPackage> findAll(string clausulaWhere, string
orderby)
                {
                        try
                        {
                                List<NetworkPackage> alNetworkPackages = new
List<NetworkPackage>();
                                string sql="";
                                DataTable dt = new DataTable();

                                sql = " SELECT " +
                                                "  TS, Length, IPID, DestinationPort,
SourcePort, " +
                          "  DestinationAddress, SourceAddress, ID, Processed " +
                                                " FROM " +
                                                "  NetworkPackage";

                                if(clausulaWhere != "")
                                {
                                        sql += " WHERE " + clausulaWhere;
                                }


                 if (orderby != "")
                 {
                     sql += " ORDER BY " + orderby;
                 }

                                OleDbCommand cmm = new OleDbCommand(sql,
DataBase.Connection, DataBase.Transaction);
                                OleDbDataAdapter da = new OleDbDataAdapter(cmm);
                                da.Fill(dt);
```

```
                            foreach (DataRow dr in dt.Rows)
                            {

                    int year = Convert.ToInt32(dr["TS"].ToString().Substring(6, 4));
                    int mounth = Convert.ToInt32(dr["TS"].ToString().Substring(3, 2));
                    int day = Convert.ToInt32(dr["TS"].ToString().Substring(0, 2));
                    int hour = Convert.ToInt32(dr["TS"].ToString().Substring(11, 2));
                    int minutes = Convert.ToInt32(dr["TS"].ToString().Substring(14, 2));
                    int seconds = Convert.ToInt32(dr["TS"].ToString().Substring(17, 2));
                    int miliseconds = Convert.ToInt32(dr["TS"].ToString().Substring(20,
3));

                    DateTime TS = new DateTime(year, mounth, day,
                        hour, minutes, seconds, miliseconds);

                    bool processed = false;
                    if (dr["Processed"] != null)
                        processed = Convert.ToBoolean(dr["Processed"]);

                                    NetworkPackage obj = new NetworkPackage();
                                    obj.load(
                                            TS,
                                            Convert.ToDouble(dr["Length"]),
                        Convert.ToDouble(dr["IPID"]),
                                            Convert.ToString(dr["DestinationPort"]),
                                            Convert.ToString(dr["SourcePort"]),
                                            Convert.ToString(dr["DestinationAddress"]),
                                            Convert.ToString(dr["SourceAddress"]),
                                            Convert.ToDouble(dr["ID"]),
                        processed,
                                            true, true);

                                    alNetworkPackages.Add(obj);
                            }
                            return alNetworkPackages;
                      }
                      catch (Exception ex)
                      {
                            Exception tmpEx = new Exception("Erro ao procurar objeto",
ex);
                            throw tmpEx;
                      }
              }


        private static List<string> findGroup(string clausulaWhere, string GroupBy)
        {
            try
            {
                List<string> colString = new List<string>();
                string sql = "";
                DataTable dt = new DataTable();

                sql = " SELECT " +
                        GroupBy +
                        " FROM " +
                        "  NetworkPackage ";

                if (clausulaWhere != "")
                {
                    sql = sql + " WHERE " + clausulaWhere;
                }

                sql += "  GROUP BY " + GroupBy;
                sql += "  ORDER BY " + GroupBy;

                OleDbCommand cmm = new OleDbCommand(sql, DataBase.Connection);
                OleDbDataAdapter da = new OleDbDataAdapter(cmm);
                da.AcceptChangesDuringUpdate = true;
                da.Fill(dt);

                foreach (DataRow dr in dt.Rows)
                {

                    colString.Add(Convert.ToString(dr[GroupBy]));
```

```
                }
                return colString;
            }
            catch (Exception ex)
            {
                Exception tmpEx = new Exception("Erro ao procurar objeto", ex);
                throw tmpEx;
            }
        }

           #endregion

       }
}
```

1.4. A Classe NetworkPackageDialog

```
       using System;
       using System.Collections;
       using System.Collections.Generic;
       using System.ComponentModel;
       using System.Data;
       using System.Drawing;
       using System.Text;
       using System.Windows.Forms;

       namespace ObjectLibrary
       {
           public partial class NetworkPackageDialog : Form
           {
               public NetworkPackageDialog()
               {
                   InitializeComponent();
               }

               private void dataGridView1_RowValidated(object sender,
DataGridViewCellEventArgs e)
               {
                   try
                   {
                       DataGridView datagrid = (DataGridView)sender;

                       NetworkPackage package =
(NetworkPackage)datagrid.Rows[e.RowIndex].DataBoundItem;
                       if(package!=null)
                           package.save();
                   }
                   catch (Exception ex)
                   {
                       MessageBox.Show(ex.InnerException.Message);
                   }
               }

               private void NetworkPackageDialog_Load(object sender, EventArgs e)
               {
                   try
                   {
                       List<NetworkPackage> col = NetworkPackage.findNetworkPackages();
                       networkPackageBindingSource.DataSource = col;
                       dataGridView.DataSource = networkPackageBindingSource;
                   }
                   catch (Exception ex)
                   {
                       MessageBox.Show(ex.InnerException.Message);
                   }
               }
           }
       }
       namespace ObjectLibrary
       {
           partial class NetworkPackageDialog
```

```
            {
                /// <summary>
                /// Required designer variable.
                /// </summary>
                private System.ComponentModel.IContainer components = null;

                /// <summary>
                /// Clean up any resources being used.
                /// </summary>
                /// <param name="disposing">true if managed resources should be disposed;
otherwise, false.</param>
                protected override void Dispose(bool disposing)
                {
                    if (disposing && (components != null))
                    {
                        components.Dispose();
                    }
                    base.Dispose(disposing);
                }

                #region Windows Form Designer generated code

                /// <summary>
                /// Required method for Designer support - do not modify
                /// the contents of this method with the code editor.
                /// </summary>
                private void InitializeComponent()
                {
                    this.components = new System.ComponentModel.Container();
                    this.dataGridView = new System.Windows.Forms.DataGridView();
                    this.sourceAddressDataGridViewTextBoxColumn = new
System.Windows.Forms.DataGridViewTextBoxColumn();
                    this.destinationAddressDataGridViewTextBoxColumn = new
System.Windows.Forms.DataGridViewTextBoxColumn();
                    this.destinationPortDataGridViewTextBoxColumn = new
System.Windows.Forms.DataGridViewTextBoxColumn();
                    this.sourcePortDataGridViewTextBoxColumn = new
System.Windows.Forms.DataGridViewTextBoxColumn();
                    this.dataDataGridViewTextBoxColumn = new
System.Windows.Forms.DataGridViewTextBoxColumn();
                    this.networkPackageBindingSource = new
System.Windows.Forms.BindingSource(this.components);

((System.ComponentModel.ISupportInitialize)(this.dataGridView)).BeginInit();

((System.ComponentModel.ISupportInitialize)(this.networkPackageBindingSource)).BeginInit
();
                    this.SuspendLayout();
                    //
                    // dataGridView
                    //
                    this.dataGridView.AutoGenerateColumns = false;
                    this.dataGridView.ColumnHeadersHeightSizeMode =
System.Windows.Forms.DataGridViewColumnHeadersHeightSizeMode.AutoSize;
                    this.dataGridView.Columns.AddRange(new
System.Windows.Forms.DataGridViewColumn[] {
                    this.sourceAddressDataGridViewTextBoxColumn,
                    this.destinationAddressDataGridViewTextBoxColumn,
                    this.destinationPortDataGridViewTextBoxColumn,
                    this.sourcePortDataGridViewTextBoxColumn,
                    this.dataDataGridViewTextBoxColumn});
                    this.dataGridView.DataSource = this.networkPackageBindingSource;
                    this.dataGridView.Dock = System.Windows.Forms.DockStyle.Fill;
                    this.dataGridView.Location = new System.Drawing.Point(0, 0);
                    this.dataGridView.Name = "dataGridView";
                    this.dataGridView.Size = new System.Drawing.Size(607, 394);
                    this.dataGridView.TabIndex = 0;
                    this.dataGridView.RowValidated += new
System.Windows.Forms.DataGridViewCellEventHandler(this.dataGridView1_RowValidated);
                    //
                    // sourceAddressDataGridViewTextBoxColumn
                    //
                    this.sourceAddressDataGridViewTextBoxColumn.DataPropertyName =
"SourceAddress";
                    this.sourceAddressDataGridViewTextBoxColumn.HeaderText =
"SourceAddress";
```

```
                    this.sourceAddressDataGridViewTextBoxColumn.Name =
"sourceAddressDataGridViewTextBoxColumn";
                    //
                    // destinationAddressDataGridViewTextBoxColumn
                    //
                    this.destinationAddressDataGridViewTextBoxColumn.DataPropertyName =
"DestinationAddress";
                    this.destinationAddressDataGridViewTextBoxColumn.HeaderText =
"DestinationAddress";
                    this.destinationAddressDataGridViewTextBoxColumn.Name =
"destinationAddressDataGridViewTextBoxColumn";
                    //
                    // destinationPortDataGridViewTextBoxColumn
                    //
                    this.destinationPortDataGridViewTextBoxColumn.DataPropertyName =
"DestinationPort";
                    this.destinationPortDataGridViewTextBoxColumn.HeaderText =
"DestinationPort";
                    this.destinationPortDataGridViewTextBoxColumn.Name =
"destinationPortDataGridViewTextBoxColumn";
                    //
                    // sourcePortDataGridViewTextBoxColumn
                    //
                    this.sourcePortDataGridViewTextBoxColumn.DataPropertyName =
"SourcePort";
                    this.sourcePortDataGridViewTextBoxColumn.HeaderText = "SourcePort";
                    this.sourcePortDataGridViewTextBoxColumn.Name =
"sourcePortDataGridViewTextBoxColumn";
                    //
                    // dataDataGridViewTextBoxColumn
                    //
                    this.dataDataGridViewTextBoxColumn.DataPropertyName = "Data";
                    this.dataDataGridViewTextBoxColumn.HeaderText = "Data";
                    this.dataDataGridViewTextBoxColumn.Name =
"dataDataGridViewTextBoxColumn";
                    //
                    // networkPackageBindingSource
                    //
                    this.networkPackageBindingSource.DataSource =
typeof(ObjectLibrary.NetworkPackage);
                    //
                    // NetworkPackageDialog
                    //
                    this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);
                    this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
                    this.ClientSize = new System.Drawing.Size(607, 394);
                    this.Controls.Add(this.dataGridView);
                    this.Name = "NetworkPackageDialog";
                    this.Text = "NetworkPackageDialog";
                    this.Load += new System.EventHandler(this.NetworkPackageDialog_Load);

((System.ComponentModel.ISupportInitialize)(this.dataGridView)).EndInit();

((System.ComponentModel.ISupportInitialize)(this.networkPackageBindingSource)).EndInit()
;
                    this.ResumeLayout(false);

                }

                #endregion

                private System.Windows.Forms.DataGridView dataGridView;
                private System.Windows.Forms.DataGridViewTextBoxColumn
sourceAddressDataGridViewTextBoxColumn;
                private System.Windows.Forms.DataGridViewTextBoxColumn
destinationAddressDataGridViewTextBoxColumn;
                private System.Windows.Forms.DataGridViewTextBoxColumn
destinationPortDataGridViewTextBoxColumn;
                private System.Windows.Forms.DataGridViewTextBoxColumn
sourcePortDataGridViewTextBoxColumn;
                private System.Windows.Forms.DataGridViewTextBoxColumn
dataDataGridViewTextBoxColumn;
                private System.Windows.Forms.BindingSource networkPackageBindingSource;

            }
        }
```

1.5. A Classe PackageEventArgs

```
using System;

namespace ObjectLibrary
{

        #region Delegate

        /// <summary>
        /// Represents the method that will handle the Add event of a ReportControl.
        /// </summary>
        /// <remarks>
        /// </remarks>
        public delegate void PackageEventHandler(object sender, PackageEventArgs e);

        #endregion

        public class PackageEventArgs : EventArgs
        {

                #region Fields

         private NetworkPackage _netPack;

                #endregion

                #region Constructors

                /// <summary>
                /// Initializes a new instance of the PackageEventArgs class with the
                /// specified report control child and parent.
                /// </summary>
                /// <param name="childPage">
                /// </param>
         public PackageEventArgs(NetworkPackage netPack)
                {
             _netPack = netPack;
                }

                #endregion

                #region Properties

                /// <summary>
                /// Gets the network package.
                /// </summary>
         public NetworkPackage NetPack
                {
                        get
                        {
                                return _netPack;
                        }
                }

                #endregion

        }
}
```

1.6. A Classe PackageSniffer

```
using System;
using System.Collections.Generic;
using System.Text;
using System.Windows.Forms;

using Tamir.IPLib;
```

```
using Tamir.IPLib.Packets;

namespace ObjectLibrary
{
    public class PackageSniffer
    {
        private string _logFile;
        private PcapDevice _onLineDevice;

        public PcapDevice OnLineDevice
        {
            get { return _onLineDevice; }
            set { _onLineDevice = value; }
        }

        public string LogFile
        {
            get { return _logFile; }
            set { _logFile = value; }
        }

        public PackageSniffer()
        {
            try
            {
            }
            catch (Exception ex)
            {
                throw new Exception("Erro ao instanciar objeto.", ex);
            }
        }

        public void Close()
        {
            try
            {
                if (_onLineDevice == null)
                    return;

                if (_onLineDevice.PcapOpened)
                    _onLineDevice.PcapClose();

            }
            catch (Exception ex)
            {
                throw new Exception("Erro ao fechar farejador.", ex);
            }
        }

        public void Open(int PackageCount, bool promiscuous_mode, int read_timeout)
        {
            try
            {
                _onLineDevice.PcapOnPacketArrival                 +=                 new
SharpPcap.PacketArrivalEvent(device_PcapOnPacketArrival);
                if (!_onLineDevice.PcapOpened)
                {
                    _onLineDevice.PcapOpen(promiscuous_mode, read_timeout);
                    _onLineDevice.PcapDumpOpen(_logFile);
                    _onLineDevice.PcapCapture(PackageCount);
                }
            }
            catch (Exception ex)
            {
                throw new Exception("Erro ao fechar farejador.", ex);
            }
        }

        /// <summary>
        /// Dumps each received packet to a pcap file
        /// </summary>
        private void device_PcapOnPacketArrival(object sender, Packet packet)
        {
            try
            {
                NetworkPackage netPack = new NetworkPackage();
```

```
                if (packet is UDPPacket)
                {
                    UDPPacket UDP = (UDPPacket)packet;
                    netPack.SourceAddress = UDP.SourceAddress;
                    netPack.SourcePort = UDP.SourcePort.ToString();
                    netPack.DestinationPort = UDP.DestinationPort.ToString();
                    netPack.DestinationAddress = UDP.DestinationAddress;
                }
                if (packet is TCPPacket)
                {
                    TCPPacket TCP = (TCPPacket)packet;
                    netPack.SourceAddress = TCP.SourceAddress;
                    netPack.DestinationAddress = TCP.DestinationAddress;
                    netPack.SourcePort = TCP.SourcePort.ToString();
                    netPack.DestinationPort = TCP.DestinationPort.ToString();
                }
                if (packet is IPPacket)
                {
                    IPPacket TCP = (IPPacket)packet;
                    netPack.IPID = TCP.Id;
                }
                if ((netPack.SourceAddress == "") &&
                    (netPack.DestinationAddress == "") &&
                    (netPack.SourcePort == "") &&
                    (netPack.DestinationPort == ""))
                {
                    return;
                }

                netPack.TS = DateTime.Now;
                netPack.Length = Convert.ToDouble(packet.Bytes.Length);

                if (_onLineDevice.PcapDumpOpened)
                {
                    _onLineDevice.PcapDump(packet);
                }

                if (PackageCaptured != null)
                {
                    PackageCaptured(this, new PackageEventArgs(netPack));
                }
            }
            catch (Exception ex)
            {
                if (ex.InnerException == null)
                    MessageBox.Show(ex.Message);
                else
                    MessageBox.Show(ex.InnerException.Message);
            }
        }

        /// <summary>
        /// Occurs when the body heigth overflow.
        /// </summary>
        public event PackageEventHandler PackageCaptured;

    }
}
```

1.7. A Classe PackageSnifferDialog

```
using System;
using System.IO;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;

using Tamir.IPLib;
```

```
using Tamir.IPLib.Packets;

using ObjectLibrary.ErrorHandler;

namespace ObjectLibrary
{
    public partial class PackageSnifferDialog : Form
    {
        private PackageSniffer _sniffer;
        delegate void SetTextCallback(ObjectLibrary.NetworkPackage netPack);

        public PackageSnifferDialog()
        {
            try
            {
                InitializeComponent();
                _sniffer = new PackageSniffer();
                _sniffer.PackageCaptured += new
PackageEventHandler(_sniffer_PackageCaptured);
            }
            catch (Exception ex)
            {
                throw new Exception("Error during instance object", ex);
            }
        }

        private void PackageSnifferDialog_Load(object sender, EventArgs e)
        {
            try
            {
                fillDevices();
            }
            catch (Exception ex)
            {
                ErrorHandlerDialog.ShowErroHandler(ref ex);
            }
        }

        private void fillDevices()
        {
            try
            {
                PcapDeviceList devices = SharpPcap.GetAllDevices();
                cboDevices.DataSource = devices;
                cboDevices.DisplayMember = "PcapDescription";
            }
            catch (Exception ex)
            {
                throw new Exception("", ex);
            }
        }

        private void loadSettings()
        {
            try
            {
                if (cboDevices.Items.Count > 0)
                {
                    PcapDevice device = (PcapDevice)cboDevices.SelectedItem;
                    _sniffer.OnLineDevice = device;
                }
            }
            catch (Exception ex)
            {
                ErrorHandlerDialog.ShowErroHandler(ref ex);
            }
        }

        private void btnStop_Click(object sender, EventArgs e)
        {
            try
            {
                _sniffer.Close();
            }
            catch (Exception ex)
            {
```

```
                ErrorHandlerDialog.ShowErroHandler(ref ex);
            }
        }

        private void btnStart_Click(object sender, EventArgs e)
        {
            try
            {
                loadSettings();
                verifyLogFile();

                if (_sniffer.OnLineDevice == null)
                {
                    MessageBox.Show("Selecione um dispositivo de rede.");
                    return;
                }

                backgroundWorkerWriter.RunWorkerAsync();
            }
            catch (Exception ex)
            {
                ErrorHandlerDialog.ShowErroHandler(ref ex);
            }
        }

        void verifyLogFile()
        {
            try
            {
                _sniffer.LogFile = txtLogFile.Text;

                if ((_sniffer.LogFile == "") || (_sniffer.LogFile == null) )
                    throw new Exception("Selecione um Arquivo de Histórico!");
            }
            catch (Exception ex)
            {
                throw new Exception("", ex);
            }
        }

        void _sniffer_PackageCaptured(object sender, PackageEventArgs e)
        {
            try
            {
                NetworkPackage netPack = e.NetPack;
                updateWinListPacket(netPack);
            }
            catch (Exception ex)
            {
                ErrorHandlerDialog.ShowErroHandler(ref ex);
            }
        }

        void updateWinListPacket(NetworkPackage netPack)
        {
            try
            {
                ListViewItem item = new ListViewItem(netPack.TS.ToShortDateString());
                item.SubItems.Add(netPack.TS.ToLongTimeString());
                item.SubItems.Add(netPack.Length.ToString());
                item.SubItems.Add(netPack.SourceAddress);
                item.SubItems.Add(netPack.DestinationAddress);

                if (this.lstPacket.InvokeRequired)
                {
                    SetTextCallback d = new SetTextCallback(updateWinListPacket);
                    this.Invoke(d, new object[] { netPack });
                }
                else
                {
                    lstPacket.Items.Add(item);
                }
            }
            catch (Exception ex)
            {
                throw new Exception("", ex);
```

```
            }
        }

        private void backgroundWorkerWriter_DoWork(object sender, DoWorkEventArgs e)
        {
            try
            {
                _sniffer.Open(SharpPcap.INFINITE, true, 0);
            }
            catch (Exception ex)
            {
                ErrorHandlerDialog.ShowErroHandler(ref ex);
            }
        }

        private void btnOpenLogFile_Click(object sender, EventArgs e)
        {
            try
            {
                saveFileDialog.Filter = "Log File|*.log";
                saveFileDialog.Title = "Selecione um nome para o arquivo de log";
                saveFileDialog.ShowDialog();

                if (saveFileDialog.FileName != "")
                {
                    FileStream fs = (FileStream)saveFileDialog.OpenFile();
                    txtLogFile.Text = fs.Name;
                    _sniffer.LogFile = fs.Name;
                    fs.Close();
                }
            }
            catch (Exception ex)
            {
                ErrorHandlerDialog.ShowErroHandler(ref ex);
            }
        }
    }
}
namespace ObjectLibrary
{
    partial class PackageSnifferDialog
    {
        /// <summary>
        /// Required designer variable.
        /// </summary>
        private System.ComponentModel.IContainer components = null;

        /// <summary>
        /// Clean up any resources being used.
        /// </summary>
        /// <param name="disposing">true if managed resources should be disposed;
otherwise, false.</param>
        protected override void Dispose(bool disposing)
        {
            if (disposing && (components != null))
            {
                components.Dispose();
            }
            base.Dispose(disposing);
        }

        #region Windows Form Designer generated code

        /// <summary>
        /// Required method for Designer support - do not modify
        /// the contents of this method with the code editor.
        /// </summary>
        private void InitializeComponent()
        {
            this.groupBox1 = new System.Windows.Forms.GroupBox();
            this.btnOpenLogFile = new System.Windows.Forms.Button();
            this.label1 = new System.Windows.Forms.Label();
            this.txtLogFile = new System.Windows.Forms.TextBox();
            this.cboDevices = new System.Windows.Forms.ComboBox();
            this.lblDevice = new System.Windows.Forms.Label();
```

```
            this.groupBox2 = new System.Windows.Forms.GroupBox();
            this.lstPacket = new System.Windows.Forms.ListView();
            this.columnHeader1 = new System.Windows.Forms.ColumnHeader();
            this.columnHeader3 = new System.Windows.Forms.ColumnHeader();
            this.columnHeader2 = new System.Windows.Forms.ColumnHeader();
            this.columnHeader4 = new System.Windows.Forms.ColumnHeader();
            this.columnHeader5 = new System.Windows.Forms.ColumnHeader();
            this.btnStop = new System.Windows.Forms.Button();
            this.btnStart = new System.Windows.Forms.Button();
            this.backgroundWorkerWriter = new System.ComponentModel.BackgroundWorker();
            this.saveFileDialog = new System.Windows.Forms.SaveFileDialog();
            this.tableLayoutPanel1 = new System.Windows.Forms.TableLayoutPanel();
            this.groupBox3 = new System.Windows.Forms.GroupBox();
            this.groupBox1.SuspendLayout();
            this.groupBox2.SuspendLayout();
            this.tableLayoutPanel1.SuspendLayout();
            this.groupBox3.SuspendLayout();
            this.SuspendLayout();
            //
            // groupBox1
            //
            this.groupBox1.Controls.Add(this.btnOpenLogFile);
            this.groupBox1.Controls.Add(this.label1);
            this.groupBox1.Controls.Add(this.txtLogFile);
            this.groupBox1.Controls.Add(this.cboDevices);
            this.groupBox1.Controls.Add(this.lblDevice);
            this.groupBox1.Dock = System.Windows.Forms.DockStyle.Fill;
            this.groupBox1.Location = new System.Drawing.Point(3, 3);
            this.groupBox1.Name = "groupBox1";
            this.groupBox1.Size = new System.Drawing.Size(758, 61);
            this.groupBox1.TabIndex = 2;
            this.groupBox1.TabStop = false;
            this.groupBox1.Text = "Configurações";
            //
            // btnOpenLogFile
            //
            this.btnOpenLogFile.FlatStyle = System.Windows.Forms.FlatStyle.Popup;
            this.btnOpenLogFile.Image = global::ObjectLibrary.Properties.Resources.open;
            this.btnOpenLogFile.Location = new System.Drawing.Point(603, 32);
            this.btnOpenLogFile.Name = "btnOpenLogFile";
            this.btnOpenLogFile.Size = new System.Drawing.Size(23, 23);
            this.btnOpenLogFile.TabIndex = 6;
            this.btnOpenLogFile.UseVisualStyleBackColor = true;
            this.btnOpenLogFile.Click += new
System.EventHandler(this.btnOpenLogFile_Click);
            //
            // label1
            //
            this.label1.AutoSize = true;
            this.label1.Location = new System.Drawing.Point(304, 16);
            this.label1.Name = "label1";
            this.label1.Size = new System.Drawing.Size(102, 13);
            this.label1.TabIndex = 8;
            this.label1.Text = "Arquivo de Histórico";
            //
            // txtLogFile
            //
            this.txtLogFile.Location = new System.Drawing.Point(307, 34);
            this.txtLogFile.Name = "txtLogFile";
            this.txtLogFile.ReadOnly = true;
            this.txtLogFile.Size = new System.Drawing.Size(290, 20);
            this.txtLogFile.TabIndex = 7;
            this.txtLogFile.Text = "c:\\Temp\\teste.log";
            //
            // cboDevices
            //
            this.cboDevices.FormattingEnabled = true;
            this.cboDevices.Location = new System.Drawing.Point(21, 33);
            this.cboDevices.Name = "cboDevices";
            this.cboDevices.Size = new System.Drawing.Size(280, 21);
            this.cboDevices.TabIndex = 1;
            //
            // lblDevice
            //
            this.lblDevice.AutoSize = true;
            this.lblDevice.Location = new System.Drawing.Point(18, 16);
```

```
            this.lblDevice.Name = "lblDevice";
            this.lblDevice.Size = new System.Drawing.Size(97, 13);
            this.lblDevice.TabIndex = 0;
            this.lblDevice.Text = "Dispositivo de rede";
            //
            // groupBox2
            //
            this.groupBox2.Controls.Add(this.lstPacket);
            this.groupBox2.Dock = System.Windows.Forms.DockStyle.Fill;
            this.groupBox2.Location = new System.Drawing.Point(3, 70);
            this.groupBox2.Name = "groupBox2";
            this.groupBox2.Size = new System.Drawing.Size(758, 259);
            this.groupBox2.TabIndex = 3;
            this.groupBox2.TabStop = false;
            this.groupBox2.Text = "Pacotes";
            //
            // lstPacket
            //
            this.lstPacket.Columns.AddRange(new System.Windows.Forms.ColumnHeader[] {
            this.columnHeader1,
            this.columnHeader3,
            this.columnHeader2,
            this.columnHeader4,
            this.columnHeader5});
            this.lstPacket.Dock = System.Windows.Forms.DockStyle.Fill;
            this.lstPacket.Location = new System.Drawing.Point(3, 16);
            this.lstPacket.Name = "lstPacket";
            this.lstPacket.Size = new System.Drawing.Size(752, 240);
            this.lstPacket.TabIndex = 0;
            this.lstPacket.UseCompatibleStateImageBehavior = false;
            this.lstPacket.View = System.Windows.Forms.View.Details;
            //
            // columnHeader1
            //
            this.columnHeader1.Text = "Data";
            this.columnHeader1.Width = 133;
            //
            // columnHeader3
            //
            this.columnHeader3.Text = "Hora";
            this.columnHeader3.Width = 115;
            //
            // columnHeader2
            //
            this.columnHeader2.Text = "Tamanho";
            this.columnHeader2.Width = 65;
            //
            // columnHeader4
            //
            this.columnHeader4.Text = "Origem";
            this.columnHeader4.Width = 139;
            //
            // columnHeader5
            //
            this.columnHeader5.Text = "Destino";
            this.columnHeader5.Width = 151;
            //
            // btnStop
            //
            this.btnStop.DialogResult = System.Windows.Forms.DialogResult.Cancel;
            this.btnStop.FlatStyle = System.Windows.Forms.FlatStyle.Popup;
            this.btnStop.Location = new System.Drawing.Point(96, 16);
            this.btnStop.Name = "btnStop";
            this.btnStop.Size = new System.Drawing.Size(84, 27);
            this.btnStop.TabIndex = 1;
            this.btnStop.Text = "Parar";
            this.btnStop.UseVisualStyleBackColor = true;
            this.btnStop.Click += new System.EventHandler(this.btnStop_Click);
            //
            // btnStart
            //
            this.btnStart.FlatStyle = System.Windows.Forms.FlatStyle.Popup;
            this.btnStart.Location = new System.Drawing.Point(6, 16);
            this.btnStart.Name = "btnStart";
            this.btnStart.Size = new System.Drawing.Size(84, 27);
            this.btnStart.TabIndex = 0;
```

```
            this.btnStart.Text = "Iniciar";
            this.btnStart.UseVisualStyleBackColor = true;
            this.btnStart.Click += new System.EventHandler(this.btnStart_Click);
            //
            // backgroundWorkerWriter
            //
            this.backgroundWorkerWriter.DoWork += new
System.ComponentModel.DoWorkEventHandler(this.backgroundWorkerWriter_DoWork);
            //
            // tableLayoutPanel1
            //
            this.tableLayoutPanel1.ColumnCount = 1;
            this.tableLayoutPanel1.ColumnStyles.Add(new
System.Windows.Forms.ColumnStyle(System.Windows.Forms.SizeType.Percent, 100F));
            this.tableLayoutPanel1.Controls.Add(this.groupBox3, 0, 2);
            this.tableLayoutPanel1.Controls.Add(this.groupBox1, 0, 0);
            this.tableLayoutPanel1.Controls.Add(this.groupBox2, 0, 1);
            this.tableLayoutPanel1.Dock = System.Windows.Forms.DockStyle.Fill;
            this.tableLayoutPanel1.Location = new System.Drawing.Point(0, 0);
            this.tableLayoutPanel1.Name = "tableLayoutPanel1";
            this.tableLayoutPanel1.RowCount = 3;
            this.tableLayoutPanel1.RowStyles.Add(new
System.Windows.Forms.RowStyle(System.Windows.Forms.SizeType.Absolute, 67F));
            this.tableLayoutPanel1.RowStyles.Add(new
System.Windows.Forms.RowStyle(System.Windows.Forms.SizeType.Percent, 100F));
            this.tableLayoutPanel1.RowStyles.Add(new
System.Windows.Forms.RowStyle(System.Windows.Forms.SizeType.Absolute, 55F));
            this.tableLayoutPanel1.Size = new System.Drawing.Size(764, 387);
            this.tableLayoutPanel1.TabIndex = 4;
            //
            // groupBox3
            //
            this.groupBox3.Controls.Add(this.btnStart);
            this.groupBox3.Controls.Add(this.btnStop);
            this.groupBox3.Dock = System.Windows.Forms.DockStyle.Fill;
            this.groupBox3.FlatStyle = System.Windows.Forms.FlatStyle.System;
            this.groupBox3.Location = new System.Drawing.Point(3, 335);
            this.groupBox3.Name = "groupBox3";
            this.groupBox3.Size = new System.Drawing.Size(758, 49);
            this.groupBox3.TabIndex = 5;
            this.groupBox3.TabStop = false;
            //
            // PackageSnifferDialog
            //
            this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);
            this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
            this.ClientSize = new System.Drawing.Size(764, 387);
            this.Controls.Add(this.tableLayoutPanel1);
            this.FormBorderStyle = System.Windows.Forms.FormBorderStyle.Fixed3D;
            this.MaximizeBox = false;
            this.Name = "PackageSnifferDialog";
            this.StartPosition = System.Windows.Forms.FormStartPosition.CenterScreen;
            this.Text = "Farejador";
            this.Load += new System.EventHandler(this.PackageSnifferDialog_Load);
            this.groupBox1.ResumeLayout(false);
            this.groupBox1.PerformLayout();
            this.groupBox2.ResumeLayout(false);
            this.tableLayoutPanel1.ResumeLayout(false);
            this.groupBox3.ResumeLayout(false);
            this.ResumeLayout(false);

        }

        #endregion

        private System.Windows.Forms.GroupBox groupBox1;
        private System.Windows.Forms.GroupBox groupBox2;
        private System.Windows.Forms.Button btnStop;
        private System.Windows.Forms.Button btnStart;
        private System.Windows.Forms.ListView lstPacket;
        private System.Windows.Forms.Label lblDevice;
        private System.Windows.Forms.ColumnHeader columnHeader1;
        private System.ComponentModel.BackgroundWorker backgroundWorkerWriter;
        private System.Windows.Forms.ColumnHeader columnHeader3;
        private System.Windows.Forms.ColumnHeader columnHeader4;
        private System.Windows.Forms.ColumnHeader columnHeader5;
```

```
        private System.Windows.Forms.ColumnHeader columnHeader2;
        private System.Windows.Forms.ComboBox cboDevices;
        private System.Windows.Forms.Button btnOpenLogFile;
        private System.Windows.Forms.Label label1;
        private System.Windows.Forms.TextBox txtLogFile;
        private System.Windows.Forms.SaveFileDialog saveFileDialog;
        private System.Windows.Forms.TableLayoutPanel tableLayoutPanel1;
        private System.Windows.Forms.GroupBox groupBox3;

    }
}
```

2. O Programa PackageAnalizator (Analisador)

2.1. A Classe AnalysisProject

```
using System;
using System.Data;
using System.Data.OleDb;
using System.Collections.Generic;
using ObjectLibrary.Communication;

namespace PackageAnalyzator
{
       public class AnalysisProject
    {

        #region Fields

        //Dados Báscios
               private string m_Description;
               private string m_Name;
               private long m_ID;

        //Dados da Análise
        private long m_idStandard;
        private long m_idTypeOfService;
        private double m_DelayAverage;
        private double m_LengthAverage;
        private double m_JitterAverage;
        private double m_LostedAverage;
        private DateTime m_InitialTS;
        private DateTime m_EndTS;
        private string m_DestinationAddress;
        private string m_SourceAddress;

        private Standard m_Standard;
        private TypeOfService m_TypeOfService;

        private List<NetWorkPackageProcessed> m_Packages;

               //Estados do objeto
               private bool m_IsHardObject;  //Existe no BD
               private bool m_IsSavedObject; //Atualizado no BD

        #endregion

        #region Constructors

        public AnalysisProject()
               {
                      try
                      {
                             clear();
                      }
                      catch (Exception ex)
                      {
                             Exception tmpEx = new Exception("Erro ao carregar objeto",
ex);
```

```
                                throw tmpEx;
                        }
                }

                #endregion

                #region Properties

                //Todos objetos devem conter esta propriedade. Ela diz se o objeto existe
no banco de dados
                internal bool IsHardObject
                {
                        get
                        {
                                return m_IsHardObject;
                        }
                }

                //Todos objetos devem conter esta propriedade. Ela diz se o objeto está
salvo no banco de dados
                internal bool IsSavedObject
                {
                        get
                        {
                                return m_IsSavedObject;
                        }
                }

                public string Description
                {
                        get
                        {
                                return m_Description;
                        }
                        set
                        {
                                m_Description = value;
                                m_IsSavedObject = false;
                        }
                }

                public string Name
                {
                        get
                        {
                                return m_Name;
                        }
                        set
                        {
                                m_Name = value;
                                m_IsSavedObject = false;
                        }
                }

                public long ID
                {
                        get
                        {
                                return m_ID;
                        }
                        set
                        {
                                m_ID = value;
                                m_IsSavedObject = false;
                        }
                }

        internal long idStandard
        {
            get
            {
                return m_idStandard;
            }
            set
            {
                m_idStandard = value;
```

```
                m_IsSavedObject = false;
            }
        }

        public Standard Standard
        {
            get
            {
                if (m_Standard == null)
                    m_Standard = Standard.findStandard(m_idStandard);

                return m_Standard;
            }
            set
            {
                m_Standard = value;
                m_idStandard = m_Standard.ID;
                m_IsSavedObject = false;
            }
        }

        internal long idTypeOfService
        {
            get
            {
                return m_idTypeOfService;
            }
            set
            {
                m_idTypeOfService = value;
                m_IsSavedObject = false;
            }
        }

        public TypeOfService TypeOfService
        {
            get
            {
                if (m_TypeOfService == null)
                    m_TypeOfService =
TypeOfService.findTypeOfService(m_idTypeOfService);

                return m_TypeOfService;
            }
            set
            {
                m_TypeOfService = value;
                m_idTypeOfService = m_TypeOfService.ID;
                m_IsSavedObject = false;
            }
        }


        public List<NetWorkPackageProcessed> Packages
        {
            get
            {
                if (m_Packages == null)
                    m_Packages =
NetWorkPackageProcessed.findNetWorkTableProcesseds_Discard(m_ID);

                return m_Packages;
            }
            set
            {
                m_Packages = value;
            }
        }

        public double DelayAverage
        {
            get
            {
                return m_DelayAverage;
            }
            set
```

```
            {
                m_DelayAverage = value;
                m_IsSavedObject = false;
            }
        }

        public double LengthAverage
        {
            get
            {
                return m_LengthAverage;
            }
            set
            {
                m_LengthAverage = value;
                m_IsSavedObject = false;
            }
        }

        public double JitterAverage
        {
            get
            {
                return m_JitterAverage;
            }
            set
            {
                m_JitterAverage = value;
                m_IsSavedObject = false;
            }
        }

        public double LostedAverage
        {
            get
            {
                return m_LostedAverage;
            }
            set
            {
                m_LostedAverage = value;
                m_IsSavedObject = false;
            }
        }

        public DateTime InitialTS
        {
            get
            {
                return m_InitialTS;
            }
            set
            {
                m_InitialTS = value;
                m_IsSavedObject = false;
            }
        }

        public DateTime EndTS
        {
            get
            {
                return m_EndTS;
            }
            set
            {
                m_EndTS = value;
                m_IsSavedObject = false;
            }
        }

        public string DestinationAddress
        {
            get
            {
                return m_DestinationAddress;
```

```
            }
            set
            {
                m_DestinationAddress = value;
                m_IsSavedObject = false;
            }
        }

        public string SourceAddress
        {
            get
            {
                return m_SourceAddress;
            }
            set
            {
                m_SourceAddress = value;
                m_IsSavedObject = false;
            }
        }

               #endregion

               #region Static Methods

               public static AnalysisProject findAnalysisProject(long ID)
               {
                       try
                       {
                               AnalysisProject obj = new AnalysisProject();

                               List<AnalysisProject> alAnalysisProjects = new
List<AnalysisProject>();

                               string clausulaWhere="";
                               clausulaWhere =
                                               "   ID = " +  ID ;


                               alAnalysisProjects = findAll(clausulaWhere);

                               if(alAnalysisProjects.Count > 1)
                               {
                                       Exception tmpEx = new Exception("A procura retornou
mais de um objeto");
                                       throw tmpEx;
                               }
                               if(alAnalysisProjects.Count == 1)
                                       obj = (AnalysisProject) alAnalysisProjects[0];
                               return obj;
                       }
                       catch (Exception ex)
                       {
                               Exception tmpEx = new Exception("Erro ao procurar objeto",
ex);
                               throw tmpEx;
                       }
               }

        public static List<AnalysisProject> findAnalysisProjects()
               {
                       try
                       {
                               string clausulaWhere = "";
                               List<AnalysisProject> alAnalysisProjects = new
List<AnalysisProject>();

                               alAnalysisProjects = findAll(clausulaWhere);
                               return alAnalysisProjects;
                       }
                       catch (Exception ex)
                       {
                               Exception tmpEx = new Exception("Erro ao procurar objeto",
ex);
                               throw tmpEx;
                       }
```

```
            }

            #endregion

            #region Instance Methods


        public virtual void save()
            {
                try
                {
                    validateProperties();

                    if (m_IsHardObject == false)
                    {
                        insert();
                        m_IsSavedObject = true;
                    }
                    else
                    {
                        if (m_IsSavedObject == false)
                        {
                            update();
                        }
                    }
                }
                catch (Exception ex)
                {
                    Exception tmpEx = new Exception("Erro ao salvar objeto",
ex);
                    throw tmpEx;
                }
            }

        public virtual void saveAnalysis()
        {
            try
            {
                int recordsAffected;
                string sql = "";

                sql = " UPDATE AnalysisProject SET" +
                    " idStandard = " + m_idStandard + ", " +
                    " idTypeOfService = " + m_idTypeOfService + ", " +
                    " EndTS = " + "'" + m_EndTS + "'" + ", " +
                    " InitialTS = " + "'" + m_InitialTS + "'" + ", " +
                    " SourceAddress = " + "'" + m_SourceAddress + "'" + ", " +
                    " DestinationAddress = " + "'" + m_DestinationAddress + "'" + ", " +
                    " DelayAverage = " + m_DelayAverage.ToString().Replace(",", ".") +",
" +
                    " LengthAverage = " + m_LengthAverage.ToString().Replace(",", ".") +
", " +
                    " JitterAverage = " + m_JitterAverage.ToString().Replace(",", ".") +
", " +
                    " LostedAverage = " + m_LostedAverage.ToString().Replace(",", ".") +
                    " WHERE " +
                       "   ID = " + ID;

                OleDbCommand cmm = new OleDbCommand(sql, DataBase.Connection);
                recordsAffected = cmm.ExecuteNonQuery();

                if (recordsAffected > 1)
                {
                    Exception tmpEx = new Exception("A atualização do objeto afetou mais
de um registro no banco de dados.");
                    throw tmpEx;
                }

                if (recordsAffected == 0)
                {
                    Exception tmpEx = new Exception("Registro alterado anteriormente.
Consulte novamente para atualizar e repita a operação.");
                    throw tmpEx;
                }

                m_IsHardObject = true;
```

```
                m_IsSavedObject = true;
            }
            catch (Exception ex)
            {
                Exception tmpEx = new Exception("Erro ao salvar objeto", ex);
                throw tmpEx;
            }
        }

        public virtual void remove()
              {
                     try
                     {
                            int recordsAffected;
                            string sql="";

                            if (m_IsHardObject == false)
                            {
                                   Exception tmpEx = new Exception("O objeto não
existe no banco de dados!");
                                   throw tmpEx;
                            }

                            sql = " DELETE FROM " +
                                          "  AnalysisProject" +
                                          " WHERE " +
                                          "   ID = " +  ID ;

                            OleDbCommand cmm = new OleDbCommand(sql,
DataBase.Connection);
                            recordsAffected = cmm.ExecuteNonQuery();

                            if (recordsAffected > 1)
                            {
                     throw new Exception("A atualização do objeto afetou mais de um
registro no banco de dados.");
                            }

                            if (recordsAffected == 0)
                            {
                     throw new Exception("Registro alterado anteriormente. Consulte
novamente para atualizar e repita a operação.");
                            }

                            clear();
                            m_IsHardObject = false;
                            m_IsSavedObject = false;
                     }
                     catch (Exception ex)
                     {
                            Exception tmpEx = new Exception("Objeto não foi criado no
banco de dados, portanto não pode ser excluído fisicamente.", ex);
                            throw tmpEx;
                     }
              }

              #endregion

              #region Private Methods

              private void validateProperties()
              {
                     try
                     {
                            if (m_Description == "")
                            {
                                   Exception tmpEx = new Exception("Description não
preenchido!");

                                   throw tmpEx;
                            }
                            if (m_Name == "")
                            {
                                   Exception tmpEx = new Exception("Name não
preenchido!");
                                   throw tmpEx;
                            }
```

```
                        }
                        catch (Exception ex)
                        {
                                Exception tmpEx = new Exception("Erro ao validar objeto",
ex);
                                throw tmpEx;
                        }
                }

                private void insert()
                {
                        try
                        {
                                string sql="";
                                long NextID = 0;

                                NextID = getNextID();

                                sql = " INSERT INTO" +
                                       "      AnalysisProject" +
                                       " (Description, Name, ID) " +
                                       " VALUES ( " +
                                        "'" + m_Description + "'" + "," +
                                        "'" + m_Name + "'" + "," +
                                         NextID + ")";

                                OleDbCommand cmm = new OleDbCommand(sql,
DataBase.Connection);
                                cmm.ExecuteNonQuery();

                                m_ID = NextID;

                                m_IsHardObject = true;
                                m_IsSavedObject = true;
                        }
                        catch (Exception ex)
                        {
                                Exception tmpEx = new Exception("Objeto não foi criado no
banco de dados", ex);
                                throw tmpEx;
                        }
                }

                private long getNextID()
                {
                        try
                        {
                                string sql="";
                                long NextID = 0;

                                DataTable dt = new DataTable();
                 sql = " SELECT MAX(AnalysisProject.ID) as LastAnalysisProject" +
                                             " FROM AnalysisProject ";

                                OleDbCommand cmm = new OleDbCommand(sql,
DataBase.Connection);
                                OleDbDataAdapter da = new OleDbDataAdapter(cmm);
                                da.Fill(dt);

                                DataRow dr = dt.Rows[0];
                 if (dr["LastAnalysisProject"].ToString() != "")
                                {
                      NextID = Convert.ToUInt32(dr["LastAnalysisProject"]);
                                }

                                NextID ++;
                                return NextID;

                        }
                        catch (Exception ex)
                        {
                                Exception tmpEx = new Exception("Erro ao procurar próximo
id", ex);
                                throw tmpEx;
                        }
                }
```

```
              private void update()
              {
                     try
                     {
                            int recordsAffected;
                            string sql="";

                            sql = " UPDATE AnalysisProject SET" +
                                    " Description = " + "'" + m_Description + "'" + ",
" +
                                    " Name = " + "'" + m_Name + "'" +
                                   " WHERE " +
                                          "   ID = " +  ID ;

                            OleDbCommand cmm = new
OleDbCommand(sql,DataBase.Connection);
                            recordsAffected = cmm.ExecuteNonQuery();

                            if (recordsAffected > 1)
                            {
                                   Exception tmpEx = new Exception("A atualização do
objeto afetou mais de um registro no banco de dados.");
                                   throw tmpEx;
                            }

                            if (recordsAffected == 0)
                            {
                                   Exception tmpEx = new Exception("Registro alterado
anteriormente. Consulte novamente para atualizar e repita a operação.");
                                   throw tmpEx;
                            }

                            m_IsHardObject = true;
                            m_IsSavedObject = true;

                     }
                     catch (Exception ex)
                     {
                            Exception tmpEx = new Exception("Objeto não foi atualizado
no banco de dados", ex);
                            throw tmpEx;
                     }
              }

        private void load(long ID, string Name, string Description, string
sourceAddress,
             string destinationAddress, DateTime initialTS, DateTime endTS, long
idStandard,
             long idTypeOfService, double delayAverage, double jitterAverage, double
lengthAverage,
             double lostedAvarage, bool isHardObject, bool isSavedObject)
              {
                     try
                     {
                            m_Description = Description;
                            m_Name = Name;
                            m_ID = ID;

                m_DelayAverage = delayAverage;
                m_idStandard = idStandard;
                m_idTypeOfService = idTypeOfService;
                m_JitterAverage = jitterAverage;
                m_LengthAverage = lengthAverage;
                m_LostedAverage = lostedAvarage;
                m_DestinationAddress = destinationAddress;
                m_EndTS = endTS;
                m_InitialTS = initialTS;
                m_SourceAddress = sourceAddress;

                            m_IsHardObject = isHardObject;
                            m_IsSavedObject = isSavedObject;

                     }
                     catch (Exception ex)
                     {
```

```
                                Exception tmpEx = new Exception("Erro de conversão de tipos
de dados", ex);
                                throw tmpEx;
                        }
                }

                private void clear()
                {
                        try
                        {
                                m_Description = "";
                                m_Name = "";
                                m_ID = 0;

                 m_DelayAverage = 0;
                 m_idStandard = 0;
                 m_idTypeOfService = 0;
                 m_JitterAverage = 0;
                 m_LengthAverage = 0;
                 m_LostedAverage = 0;
                 m_DestinationAddress = "";
                 m_EndTS = DateTime.MinValue;
                 m_InitialTS = DateTime.MinValue;
                 m_SourceAddress = "";

                 m_Standard = null;
                 m_TypeOfService = null;

                                m_IsHardObject = false;
                                m_IsSavedObject = false;
                        }
                        catch (Exception ex)
                        {
                                Exception tmpEx = new Exception("Erro ao limpar objeto",
ex);
                                throw tmpEx;
                        }
                }

                private static List<AnalysisProject> findAll(string clausulaWhere)
                {
                        try
                        {
                                List<AnalysisProject> alAnalysisProjects = new
List<AnalysisProject>();
                                string sql="";
                                DataTable dt = new DataTable();

                 sql = " SELECT " +
                           "  Description, " +
                           "  Name, " +
                           "  ID, " +
                           "  SourceAddress, " +
                           "  DestinationAddress, " +
                           "  InitialTS, " +
                           "  EndTS, " +
                           "  idTypeOfService, " +
                           "  idStandard, " +
                           "  LengthAverage, " +
                           "  DelayAverage, " +
                           "  JitterAverage, " +
                           "  LostedAverage " +
                           " FROM " +
                           "  AnalysisProject";

                                if(clausulaWhere != "")
                                {
                                        sql = sql + " WHERE " + clausulaWhere;
                                }

                                OleDbCommand cmm = new OleDbCommand(sql,
DataBase.Connection);
                                OleDbDataAdapter da = new OleDbDataAdapter(cmm);
                                da.Fill(dt);

                 foreach (DataRow dr in dt.Rows)
```

```
                                {
                    string sourceAddress = "";
                    if (dr["SourceAddress"] != DBNull.Value)
                        sourceAddress = Convert.ToString(dr["SourceAddress"]);

                    string destinationAddress = "";
                    if (dr["DestinationAddress"] != DBNull.Value)
                        destinationAddress = Convert.ToString(dr["DestinationAddress"]);

                    DateTime initialTS = DateTime.MinValue;
                    if (dr["InitialTS"] != DBNull.Value)
                        initialTS = Convert.ToDateTime(dr["InitialTS"]);

                    DateTime endTS = DateTime.MinValue;
                    if (dr["EndTS"] != DBNull.Value)
                        endTS = Convert.ToDateTime(dr["EndTS"]);

                    long idStandard = 0;
                    if (dr["idStandard"] != DBNull.Value)
                        idStandard = Convert.ToInt64(dr["idStandard"]);

                    long idTypeOfService = 0;
                    if (dr["idTypeOfService"] != DBNull.Value)
                        idTypeOfService = Convert.ToInt64(dr["idTypeOfService"]);

                    double delayAverage = 0;
                    if (dr["DelayAverage"] != DBNull.Value)
                        delayAverage = Convert.ToDouble(dr["DelayAverage"]);

                    double jitterAverage = 0;
                    if (dr["JitterAverage"] != DBNull.Value)
                        jitterAverage = Convert.ToDouble(dr["JitterAverage"]);

                    double lengthAverage = 0;
                    if (dr["LengthAverage"] != DBNull.Value)
                        lengthAverage = Convert.ToDouble(dr["LengthAverage"]);

                    double lostedAverage = 0;
                    if (dr["LostedAverage"] != DBNull.Value)
                        lostedAverage = Convert.ToDouble(dr["LostedAverage"]);

                    AnalysisProject obj = new AnalysisProject();

                    obj.load(
                                            Convert.ToUInt32(dr["ID"]),
                        Convert.ToString(dr["Name"]),
                        Convert.ToString(dr["Description"]),
                        sourceAddress,
                        destinationAddress,
                        initialTS,
                        endTS,
                        idStandard,
                        idTypeOfService,
                        delayAverage,
                        jitterAverage,
                        lengthAverage,
                        lostedAverage,
                        true, true);

                                alAnalysisProjects.Add(obj);
                            }
                            return alAnalysisProjects;
                        }
                        catch (Exception ex)
                        {
                            Exception tmpEx = new Exception("Erro ao procurar objeto",
ex);
                            throw tmpEx;
                        }
                }

                #endregion

        }
}
```

2.2. A Classe AnalysisProjectDialog

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;

using ObjectLibrary.ErrorHandler;
using ObjectLibrary.Communication;

namespace PackageAnalyzator
{
    public partial class AnalysisProjectDialog : Form
    {

        private AnalysisProject m_AnalysisProject;

        public AnalysisProjectDialog()
        {
            try
            {
                InitializeComponent();
            }
            catch (Exception ex)
            {
                throw new Exception("Erro ao instanciar objeto.", ex);
            }
        }

        private void AnalysisProjectDialog_Load(object sender, EventArgs e)
        {
            try
            {
                clear();
                fillListAnalysisProjects();
            }
            catch (Exception ex)
            {
                ErrorHandlerDialog.ShowErroHandler(ref ex);
            }
        }

        private void clear()
        {
            try
            {
                m_AnalysisProject = null;

                errorProvider.SetError(txtDescription, "");
                errorProvider.SetError(txtName, "");

                txtDescription.Text = "";
                txtName.Text = "";
            }
            catch (Exception ex)
            {
                throw new Exception("Erro ao limpar Projeto de Análise", ex);
            }
        }

        private void fillListAnalysisProjects()
        {
            try
            {
                lstAnalysisProjects.Items.Clear();

                lstAnalysisProjects.DisplayMember = "Name";
```

```
                List<AnalysisProject> alAnalysisProjects =
AnalysisProject.findAnalysisProjects();
                foreach (AnalysisProject obj in alAnalysisProjects)
                {
                    lstAnalysisProjects.Items.Add(obj);
                }
            }
            catch (Exception ex)
            {
                throw new Exception("Erro ao listar Tipos de Serviço", ex);
            }
        }

        private void btnNew_Click(object sender, EventArgs e)
        {
            try
            {
                clear();
            }
            catch (Exception ex)
            {
                ErrorHandlerDialog.ShowErroHandler(ref ex);
            }
        }

        private void btnRefresh_Click(object sender, EventArgs e)
        {
            try
            {
                fillListAnalysisProjects();
            }
            catch (Exception ex)
            {
                ErrorHandlerDialog.ShowErroHandler(ref ex);
            }
        }

        private void btnSave_Click(object sender, EventArgs e)
        {
            try
            {
                if (fieldsValidated("") == false)
                    return;

                DialogResult dlgr =
                    MessageBox.Show(this, "Tem certeza que deseja salvar?",
                    "Atenção!",
                    MessageBoxButtons.YesNo,
                    MessageBoxIcon.Question);

                if (dlgr == DialogResult.No)
                    return;

                save();
                MessageBox.Show(this,
                    "Projeto de Análise salvo com sucesso!",
                    "Analizador",
                    MessageBoxButtons.OK,
                    MessageBoxIcon.Information);

                clear();
                fillListAnalysisProjects();
            }
            catch (Exception ex)
            {
                ErrorHandlerDialog.ShowErroHandler(ref ex);
            }
        }

        private bool fieldsValidated(string nameField)
        {
            try
            {
                bool errorControl = true;
                bool result = true;
                return errorControl;
```

```
            }
            catch (Exception ex)
            {
                throw new Exception("Erro ao validar os dados do Projeto de Análise.",
ex);
            }
        }

        private void save()
        {
            try
            {
                if (m_AnalysisProject == null)
                    m_AnalysisProject = new AnalysisProject();

                m_AnalysisProject.Name = txtName.Text;
                m_AnalysisProject.Description = txtDescription.Text;

                //IDbTransaction tran = DataBase.Connection.BeginTransaction();
                try
                {
                    m_AnalysisProject.save();
                    //tran.Commit();
                }
                catch (Exception ex)
                {
                    //tran.Rollback();
                    throw new Exception("Erro ao salvar Projeto de Análise", ex);
                }
            }
            catch (Exception ex)
            {
                throw new Exception("Erro ao salvar Projeto de Análise", ex);
            }
        }

        private void lstAnalysisProjects_SelectedIndexChanged(object sender, EventArgs
e)
        {
            try
            {
                if (lstAnalysisProjects.SelectedItem != null)
                {
                    m_AnalysisProject =
(AnalysisProject)lstAnalysisProjects.SelectedItem;
                    loadAnalysisProject();
                }
            }
            catch (Exception ex)
            {
                ErrorHandlerDialog.ShowErroHandler(ref ex);
            }
        }

        private void loadAnalysisProject()
        {
            try
            {
                txtName.Text = m_AnalysisProject.Name;
                txtDescription.Text = m_AnalysisProject.Description;
            }
            catch (Exception ex)
            {
                throw new Exception("Erro ao selecionar Projeto de Análise", ex);
            }
        }

        private void btnDel_Click(object sender, EventArgs e)
        {
            try
            {
                if (m_AnalysisProject == null)
                {
                    throw new Exception("Por favor selecione um item primeiro.");
                }
```

```
                DialogResult dlgr =
                    MessageBox.Show(this, "Tem certeza que deseja excluir?",
                    "Atenção!",
                    MessageBoxButtons.YesNo,
                    MessageBoxIcon.Question);
                if (dlgr == DialogResult.No)
                    return;

                delete();

                MessageBox.Show(this,
                    "Projeto de Análise excluído com sucesso!",
                    "Analizador",
                    MessageBoxButtons.OK,
                    MessageBoxIcon.Information);

                fillListAnalysisProjects();
                clear();
            }
            catch (Exception ex)
            {
                ErrorHandlerDialog.ShowErroHandler(ref ex);
            }
        }

        private void delete()
        {
            try
            {
                //IDbTransaction tran = DataBase.Connection.BeginTransaction();
                try
                {
                    m_AnalysisProject.remove();
                    //tran.Commit();
                }
                catch (Exception ex)
                {
                    //tran.Rollback();
                    throw new Exception("Erro ao deletar Projeto de Análise", ex);
                }

            }
            catch (Exception ex)
            {
                throw new Exception("Erro ao deletar Projeto de Análise", ex);
            }
        }

        private void Fields_Validating(object sender, CancelEventArgs e)
        {
            try
            {
                Control c = (Control)sender;
                fieldsValidated(c.Name);
            }
            catch (Exception ex)
            {
                ErrorHandlerDialog.ShowErroHandler(ref ex);
            }
        }

        private void Fields_Enter(object sender, EventArgs e)
        {
            try
            {
                Control c = (Control)sender;
                errorProvider.SetError(c, "");
            }
            catch (Exception ex)
            {
                ErrorHandlerDialog.ShowErroHandler(ref ex);
            }
        }

    }
}
```

```
        namespace PackageAnalyzator
        {
            partial class AnalysisProjectDialog
            {
                /// <summary>
                /// Required designer variable.
                /// </summary>
                private System.ComponentModel.IContainer components = null;

                /// <summary>
                /// Clean up any resources being used.
                /// </summary>
                /// <param name="disposing">true if managed resources should be disposed;
otherwise, false.</param>
                protected override void Dispose(bool disposing)
                {
                    if (disposing && (components != null))
                    {
                        components.Dispose();
                    }
                    base.Dispose(disposing);
                }

                #region Windows Form Designer generated code

                /// <summary>
                /// Required method for Designer support - do not modify
                /// the contents of this method with the code editor.
                /// </summary>
                private void InitializeComponent()
                {
                    this.components = new System.ComponentModel.Container();
                    this.gpbEdit = new System.Windows.Forms.GroupBox();
                    this.lblDescription = new System.Windows.Forms.Label();
                    this.txtDescription = new System.Windows.Forms.TextBox();
                    this.lblName = new System.Windows.Forms.Label();
                    this.txtName = new System.Windows.Forms.TextBox();
                    this.btnSave = new System.Windows.Forms.Button();
                    this.btnDel = new System.Windows.Forms.Button();
                    this.btnNew = new System.Windows.Forms.Button();
                    this.gpbList = new System.Windows.Forms.GroupBox();
                    this.lstAnalysisProjects = new System.Windows.Forms.ListBox();
                    this.toolTip = new System.Windows.Forms.ToolTip(this.components);
                    this.errorProvider = new
System.Windows.Forms.ErrorProvider(this.components);
                    this.gpbEdit.SuspendLayout();
                    this.gpbList.SuspendLayout();

((System.ComponentModel.ISupportInitialize)(this.errorProvider)).BeginInit();
                    this.SuspendLayout();
                    //
                    // gpbEdit
                    //
                    this.gpbEdit.Controls.Add(this.lblDescription);
                    this.gpbEdit.Controls.Add(this.txtDescription);
                    this.gpbEdit.Controls.Add(this.lblName);
                    this.gpbEdit.Controls.Add(this.txtName);
                    this.gpbEdit.Font = new System.Drawing.Font("Microsoft Sans Serif",
8.25F, System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
                    this.gpbEdit.Location = new System.Drawing.Point(4, 199);
                    this.gpbEdit.Name = "gpbEdit";
                    this.gpbEdit.Size = new System.Drawing.Size(336, 126);
                    this.gpbEdit.TabIndex = 1;
                    this.gpbEdit.TabStop = false;
                    this.gpbEdit.Text = "Novo / Editar";
                    //
                    // lblDescription
                    //
                    this.lblDescription.AutoSize = true;
                    this.lblDescription.Font = new System.Drawing.Font("Microsoft Sans
Serif", 8.25F, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point,
((byte)(0)));
                    this.lblDescription.Location = new System.Drawing.Point(8, 66);
                    this.lblDescription.Name = "lblDescription";
                    this.lblDescription.Size = new System.Drawing.Size(58, 13);
                    this.lblDescription.TabIndex = 2;
```

```
                    this.lblDescription.Text = "Descrição:";
                    //
                    // txtDescription
                    //
                    this.txtDescription.Font = new System.Drawing.Font("Microsoft Sans
Serif", 8.25F, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point,
((byte)(0)));
                    this.txtDescription.Location = new System.Drawing.Point(11, 82);
                    this.txtDescription.Multiline = true;
                    this.txtDescription.Name = "txtDescription";
                    this.txtDescription.Size = new System.Drawing.Size(317, 38);
                    this.txtDescription.TabIndex = 3;
                    this.toolTip.SetToolTip(this.txtDescription, "Código do Banco na
compensação");
                    this.txtDescription.Enter += new
System.EventHandler(this.Fields_Enter);
                    this.txtDescription.Validating += new
System.ComponentModel.CancelEventHandler(this.Fields_Validating);
                    //
                    // lblName
                    //
                    this.lblName.AutoSize = true;
                    this.lblName.Font = new System.Drawing.Font("Microsoft Sans Serif",
8.25F, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point,
((byte)(0)));
                    this.lblName.Location = new System.Drawing.Point(8, 19);
                    this.lblName.Name = "lblName";
                    this.lblName.Size = new System.Drawing.Size(38, 13);
                    this.lblName.TabIndex = 0;
                    this.lblName.Text = "Nome:";
                    //
                    // txtName
                    //
                    this.txtName.Font = new System.Drawing.Font("Microsoft Sans Serif",
8.25F, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point,
((byte)(0)));
                    this.txtName.Location = new System.Drawing.Point(9, 35);
                    this.txtName.MaxLength = 255;
                    this.txtName.Name = "txtName";
                    this.txtName.Size = new System.Drawing.Size(319, 20);
                    this.txtName.TabIndex = 1;
                    this.txtName.Enter += new System.EventHandler(this.Fields_Enter);
                    this.txtName.Validating += new
System.ComponentModel.CancelEventHandler(this.Fields_Validating);
                    //
                    // btnSave
                    //
                    this.btnSave.FlatStyle = System.Windows.Forms.FlatStyle.Popup;
                    this.btnSave.Font = new System.Drawing.Font("Microsoft Sans Serif",
8.25F, System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
                    this.btnSave.ForeColor = System.Drawing.Color.MediumBlue;
                    this.btnSave.ImageAlign = System.Drawing.ContentAlignment.MiddleLeft;
                    this.btnSave.Location = new System.Drawing.Point(270, 331);
                    this.btnSave.Name = "btnSave";
                    this.btnSave.Size = new System.Drawing.Size(70, 24);
                    this.btnSave.TabIndex = 4;
                    this.btnSave.Text = "Salvar";
                    this.btnSave.TextAlign = System.Drawing.ContentAlignment.MiddleRight;
                    this.btnSave.Click += new System.EventHandler(this.btnSave_Click);
                    //
                    // btnDel
                    //
                    this.btnDel.FlatStyle = System.Windows.Forms.FlatStyle.Popup;
                    this.btnDel.Font = new System.Drawing.Font("Microsoft Sans Serif",
8.25F, System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
                    this.btnDel.ForeColor = System.Drawing.Color.Red;
                    this.btnDel.ImageAlign = System.Drawing.ContentAlignment.MiddleLeft;
                    this.btnDel.Location = new System.Drawing.Point(194, 331);
                    this.btnDel.Name = "btnDel";
                    this.btnDel.Size = new System.Drawing.Size(70, 24);
                    this.btnDel.TabIndex = 3;
                    this.btnDel.Text = "Excluir";
                    this.btnDel.TextAlign = System.Drawing.ContentAlignment.MiddleRight;
                    this.btnDel.Click += new System.EventHandler(this.btnDel_Click);
                    //
                    // btnNew
```

```
                    //
                    this.btnNew.FlatStyle = System.Windows.Forms.FlatStyle.Popup;
                    this.btnNew.Font = new System.Drawing.Font("Microsoft Sans Serif",
8.25F, System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
                    this.btnNew.ForeColor = System.Drawing.Color.MediumBlue;
                    this.btnNew.ImageAlign = System.Drawing.ContentAlignment.MiddleLeft;
                    this.btnNew.Location = new System.Drawing.Point(119, 331);
                    this.btnNew.Name = "btnNew";
                    this.btnNew.Size = new System.Drawing.Size(69, 24);
                    this.btnNew.TabIndex = 2;
                    this.btnNew.Text = "Novo";
                    this.btnNew.TextAlign = System.Drawing.ContentAlignment.MiddleRight;
                    this.btnNew.Click += new System.EventHandler(this.btnNew_Click);
                    //
                    // gpbList
                    //
                    this.gpbList.Controls.Add(this.lstAnalysisProjects);
                    this.gpbList.Font = new System.Drawing.Font("Microsoft Sans Serif",
8.25F, System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
                    this.gpbList.Location = new System.Drawing.Point(4, 7);
                    this.gpbList.Name = "gpbList";
                    this.gpbList.Size = new System.Drawing.Size(336, 189);
                    this.gpbList.TabIndex = 0;
                    this.gpbList.TabStop = false;
                    this.gpbList.Text = "Projetos Cadastrados:";
                    //
                    // lstAnalysisProjects
                    //
                    this.lstAnalysisProjects.Font = new System.Drawing.Font("Microsoft
Sans Serif", 8.25F, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point,
((byte)(0)));
                    this.lstAnalysisProjects.Location = new System.Drawing.Point(9, 21);
                    this.lstAnalysisProjects.Name = "lstAnalysisProjects";
                    this.lstAnalysisProjects.Size = new System.Drawing.Size(319, 160);
                    this.lstAnalysisProjects.Sorted = true;
                    this.lstAnalysisProjects.TabIndex = 0;
                    this.lstAnalysisProjects.SelectedIndexChanged += new
System.EventHandler(this.lstAnalysisProjects_SelectedIndexChanged);
                    //
                    // errorProvider
                    //
                    this.errorProvider.ContainerControl = this;
                    //
                    // AnalysisProjectDialog
                    //
                    this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);
                    this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
                    this.ClientSize = new System.Drawing.Size(345, 367);
                    this.Controls.Add(this.btnSave);
                    this.Controls.Add(this.gpbEdit);
                    this.Controls.Add(this.gpbList);
                    this.Controls.Add(this.btnDel);
                    this.Controls.Add(this.btnNew);
                    this.FormBorderStyle = System.Windows.Forms.FormBorderStyle.Fixed3D;
                    this.MaximizeBox = false;
                    this.MinimizeBox = false;
                    this.Name = "AnalysisProjectDialog";
                    this.StartPosition =
System.Windows.Forms.FormStartPosition.CenterScreen;
                    this.Text = "Cadastro de Projetos de Análise";
                    this.Load += new
System.EventHandler(this.AnalysisProjectDialog_Load);
                    this.gpbEdit.ResumeLayout(false);
                    this.gpbEdit.PerformLayout();
                    this.gpbList.ResumeLayout(false);

((System.ComponentModel.ISupportInitialize)(this.errorProvider)).EndInit();
                    this.ResumeLayout(false);

                }

                #endregion

                private System.Windows.Forms.GroupBox gpbEdit;
                private System.Windows.Forms.Button btnSave;
                private System.Windows.Forms.Label lblDescription;
```

```
            private System.Windows.Forms.TextBox txtDescription;
            private System.Windows.Forms.Label lblName;
            private System.Windows.Forms.TextBox txtName;
            private System.Windows.Forms.Button btnDel;
            private System.Windows.Forms.Button btnNew;
            private System.Windows.Forms.GroupBox gpbList;
            private System.Windows.Forms.ListBox lstAnalysisProjects;
            private System.Windows.Forms.ToolTip toolTip;
            private System.Windows.Forms.ErrorProvider errorProvider;
        }
    }
```

2.3. A Classe AnalysisProjectResultDialog

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;

using ObjectLibrary.ErrorHandler;
using ObjectLibrary.Communication;

using ObjectLibrary;
using ZedGraph;

namespace PackageAnalyzator
{
    public partial class AnalysisProjectResultDialog : Form
    {

        private AnalysisProject m_AnalysisProject;

        public AnalysisProjectResultDialog()
        {
            try
            {
                InitializeComponent();
            }
            catch (Exception ex)
            {
                throw new Exception("Erro ao instanciar objeto.", ex);
            }
        }

        public AnalysisProjectResultDialog(AnalysisProject analysisProject)
        {
            try
            {
                InitializeComponent();
                m_AnalysisProject = analysisProject;

                btnResults.Enabled = true;
                cboAnalysisProject.Enabled = true;
                if (m_AnalysisProject != null)
                {
                    btnResults.Enabled = false;
                    cboAnalysisProject.Enabled = false;
                }
            }
            catch (Exception ex)
            {
                throw new Exception("Erro ao instanciar objeto.", ex);
            }
        }

        private void AnalysisProjectProcessedDialog_Load(object sender, EventArgs e)
        {
            try
```

```
            {
                fillComboAnalysisProject();

                AnalysisProject ap = m_AnalysisProject;
                clear();
                m_AnalysisProject = ap;

                if (m_AnalysisProject != null)
                    cboAnalysisProject.SelectedValue = m_AnalysisProject.ID;

                loadAnalisysProjectResults();
            }
            catch (Exception ex)
            {
                ErrorHandlerDialog.ShowErroHandler(ref ex);
            }
        }

        private void fillComboAnalysisProject()
        {
            try
            {
                List<AnalysisProject> alAnalysisProject =
                    AnalysisProject.findAnalysisProjects();
                AnalysisProject tos = new AnalysisProject();
                tos.Name = "(Selecione)";
                alAnalysisProject.Insert(0, tos);

                cboAnalysisProject.DisplayMember = "Name";
                cboAnalysisProject.ValueMember = "ID";
                cboAnalysisProject.DataSource = alAnalysisProject;
            }
            catch (Exception ex)
            {
                throw new Exception("Erro ao listar Tipos de Serviço", ex);
            }
        }

        private void clear()
        {
            try
            {
                m_AnalysisProject = null;

                txtDestinationAddress.Text = "";
                txtSourceAddress.Text = "";
                txtStandard.Text = "";
                txtTypeOfService.Text = "";
                txtAnalysisProjectDescription.Text = "";

                lstParametersResults.Items.Clear();

                if (!btnResults.Focused)
                {
                    if (cboAnalysisProject.Items.Count > 0)
                        cboAnalysisProject.SelectedIndex = 0;
                }

                zedGraphJitter.GraphPane.XAxis.Scale.Min = 0;
                zedGraphJitter.GraphPane.XAxis.Scale.Max = 10;
                zedGraphJitter.GraphPane.GraphObjList.Clear();
                zedGraphJitter.GraphPane.CurveList.Clear();

            }
            catch (Exception ex)
            {
                throw new Exception("Erro ao limpar Projeto de Análise", ex);
            }
        }

        private void loadAnalisysProjectResults()
        {
            try
            {
                if (m_AnalysisProject == null)
                    return;
```

```
                txtAnalysisProjectDescription.Text = m_AnalysisProject.Description;
                txtDestinationAddress.Text = m_AnalysisProject.DestinationAddress;
                txtSourceAddress.Text = m_AnalysisProject.SourceAddress;
                txtStandard.Text = m_AnalysisProject.Standard.Name;
                txtTypeOfService.Text = m_AnalysisProject.TypeOfService.Name;

                fillListParameterResults();

                fillGraph();
            }
            catch (Exception ex)
            {
                throw new Exception("Erro ao listar Tipos de Serviço", ex);
            }
        }

        private void fillListParameterResults()
        {
            try
            {
                lstParametersResults.Items.Clear();

                if (m_AnalysisProject == null)
                    return;

                lstParametersResults.BeginUpdate();

                ListViewItem liOutFlow = fillListParameterResultsOutflow();
                lstParametersResults.Items.Add(liOutFlow);

                ListViewItem liJitter = fillListParameterResultsJitter();
                lstParametersResults.Items.Add(liJitter);

                ListViewItem liPackageDelay = fillListParameterResultsPackageDelay();
                lstParametersResults.Items.Add(liPackageDelay);

                ListViewItem liPackageLoses = fillListParameterResultsPackageLoses();
                lstParametersResults.Items.Add(liPackageLoses);

                lstParametersResults.EndUpdate();
            }
            catch (Exception ex)
            {
                throw new Exception("Erro ao carregar análise.", ex);
            }
        }

        private ListViewItem fillListParameterResultsOutflow()
        {
            try
            {
                QoSRequisite requisite;
                StandardTolerance st;
                ListViewItem li;

                requisite = new QoSRequisite(EnumQoSRequisite.Outflow);
                st = StandardTolerance.findStandardTolerance(
                    m_AnalysisProject.TypeOfService.ID,    m_AnalysisProject.Standard.ID,
requisite.ID);
                li = new ListViewItem(requisite.Name);
                li.SubItems.Add((m_AnalysisProject.LengthAverage).ToString("#,##0")  +  "
kbps");

                if (st == null)
                {
                    li.SubItems.Add("");
                    li.SubItems.Add("");
                    li.SubItems.Add("Parâmetro não cadastrado para a norma escolhida.");
                }
                else
                {
                    li.SubItems.Add((st.MinValue / 1000).ToString("#,##0") + " kbps");
                    li.SubItems.Add((st.MaxValue / 1000).ToString("#,##0") + " kbps");

                    if ((m_AnalysisProject.LengthAverage < st.MinValue) ||
```

```
                        (m_AnalysisProject.LengthAverage > st.MaxValue))
                    {
                        li.SubItems.Add("Parâmetro FORA dos padrões.");
                    }
                    else
                    {
                        li.SubItems.Add("Parâmetro DENTRO dos padrões.");
                    }
                }

                return li;
            }
            catch (Exception ex)
            {
                throw new Exception("Erro ao carregar análise.", ex);
            }
        }

        private ListViewItem fillListParameterResultsJitter()
        {
            try
            {
                QoSRequisite requisite;
                StandardTolerance st;
                ListViewItem li;

                //Jitter
                requisite = new QoSRequisite(EnumQoSRequisite.Jitter);
                st = StandardTolerance.findStandardTolerance(
                    m_AnalysisProject.TypeOfService.ID,    m_AnalysisProject.Standard.ID,
requisite.ID);
                li = new ListViewItem(requisite.Name);
                li.SubItems.Add(m_AnalysisProject.JitterAverage.ToString("#,##0")   +   "
ms");

                if (st == null)
                {
                    li.SubItems.Add("");
                    li.SubItems.Add("");
                    li.SubItems.Add("Parâmetro não cadastrado para a norma escolhida.");
                }
                else
                {
                    li.SubItems.Add(st.MinValue.ToString("#,##0") + " ms");
                    li.SubItems.Add(st.MaxValue.ToString("#,##0") + " ms");

                    if ((m_AnalysisProject.JitterAverage < st.MinValue) ||
                        (m_AnalysisProject.JitterAverage > st.MaxValue))
                    {
                        li.SubItems.Add("Parâmetro FORA dos padrões.");
                    }
                    else
                    {
                        li.SubItems.Add("Parâmetro DENTRO dos padrões.");
                    }
                }

                return li;
            }
            catch (Exception ex)
            {
                throw new Exception("Erro ao carregar análise.", ex);
            }
        }

        private ListViewItem fillListParameterResultsPackageDelay()
        {
            try
            {
                QoSRequisite requisite;
                StandardTolerance st;
                ListViewItem li;

                //Atraso
                requisite = new QoSRequisite(EnumQoSRequisite.PackageDelay);
                st = StandardTolerance.findStandardTolerance(
```

```
                    m_AnalysisProject.TypeOfService.ID,    m_AnalysisProject.Standard.ID,
requisite.ID);
                li = new ListViewItem(requisite.Name);
                li.SubItems.Add(m_AnalysisProject.DelayAverage.ToString("#,##0")    +    "
ms");

                if (st == null)
                {
                    li.SubItems.Add("");
                    li.SubItems.Add("");
                    li.SubItems.Add("Parâmetro não cadastrado para a norma escolhida.");
                }
                else
                {
                    li.SubItems.Add(st.MinValue.ToString("#,##0") + " ms");
                    li.SubItems.Add(st.MaxValue.ToString("#,##0") + " ms");

                    if ((m_AnalysisProject.DelayAverage < st.MinValue) ||
                        (m_AnalysisProject.DelayAverage > st.MaxValue))
                    {
                        li.SubItems.Add("Parâmetro FORA dos padrões.");
                    }
                    else
                    {
                        li.SubItems.Add("Parâmetro DENTRO dos padrões.");
                    }
                }

                return li;
            }
            catch (Exception ex)
            {
                throw new Exception("Erro ao carregar análise.", ex);
            }
        }

        private ListViewItem fillListParameterResultsPackageLoses()
        {
            try
            {
                QoSRequisite requisite;
                StandardTolerance st;
                ListViewItem li;

                //Perdas
                requisite = new QoSRequisite(EnumQoSRequisite.PackageLoses);
                st = StandardTolerance.findStandardTolerance(
                    m_AnalysisProject.TypeOfService.ID,    m_AnalysisProject.Standard.ID,
requisite.ID);
                li = new ListViewItem(requisite.Name);
                li.SubItems.Add(m_AnalysisProject.LostedAverage.ToString("#,##0.00")    +
"%");

                if (st == null)
                {
                    li.SubItems.Add("");
                    li.SubItems.Add("");
                    li.SubItems.Add("Parâmetro não cadastrado para a norma escolhida.");
                }
                else
                {
                    li.SubItems.Add(st.MinValue.ToString("#,##0.00") + "%");
                    li.SubItems.Add(st.MaxValue.ToString("#,##0.00") + "%");

                    if ((m_AnalysisProject.LostedAverage < st.MinValue) ||
                        (m_AnalysisProject.LostedAverage > st.MaxValue))
                    {
                        li.SubItems.Add("Parâmetro FORA dos padrões.");
                    }
                    else
                    {
                        li.SubItems.Add("Parâmetro DENTRO dos padrões.");
                    }
                }

                return li;
```

```
            }
            catch (Exception ex)
            {
                throw new Exception("Erro ao carregar análise.", ex);
            }
        }

        private void fillGraph()
        {
            try
            {
                if (m_AnalysisProject == null)
                    return;

                fillGraphJitter();
                fillGraphOutFlow();
                fillGraphPackageDelay();
                fillGraphPackageLoses();

            }
            catch (Exception ex)
            {
                throw new Exception("Erro ao listar Tipos de Serviço", ex);
            }
        }

        private void fillGraphJitter()
        {
            try
            {
                if (m_AnalysisProject == null)
                    return;

                NetWorkPackageProcessed firstPack = m_AnalysisProject.Packages[0];
                NetWorkPackageProcessed                    lastPack                    =
m_AnalysisProject.Packages[m_AnalysisProject.Packages.Count - 1];

                TimeSpan time1 = lastPack.SourceTS - firstPack.SourceTS;

                List<double> x = new List<double>();
                List<double> y = new List<double>();

                double actualJitterTotal = 0;
                double actualJitterAverage = 0;
                double maxJitter = 0;
                long actualSecond = 0;
                long actualPackagesCount = 0;
                long totalPackages = 0;

                m_AnalysisProject.Packages = null;
                foreach (NetWorkPackageProcessed pack in m_AnalysisProject.Packages)
                {
                    TimeSpan temp2 = pack.SourceTS - firstPack.SourceTS;

                    if (temp2.Seconds <= actualSecond)
                    {
                        if (!pack.Losted)
                            actualJitterTotal += pack.Jitter;

                        actualPackagesCount++;
                    }
                    else
                    {
                        actualJitterAverage = actualJitterTotal / actualPackagesCount;

                        if (actualJitterAverage > maxJitter)
                            maxJitter = actualJitterAverage;

                        x.Add(actualSecond);
                        y.Add(actualJitterAverage);

                        actualSecond = temp2.Seconds;
                        actualPackagesCount = 0;
                        actualJitterTotal = 0;
                    }
```

```
                    totalPackages++;
                }

                GraphPane pane = zedGraphJitter.GraphPane;
                pane.Title.Text = "Flutacao no Tempo com Tolerancia da norma " +
m_AnalysisProject.Standard.Name;
                pane.XAxis.Title.Text = "Tempo (s)";
                pane.YAxis.Title.Text = "Flutacao (ms)";

                pane.Chart.Fill = new Fill(Color.FromArgb(255, 255, 245),
Color.FromArgb(255, 255, 190), 90F);

                pane.CurveList.Clear();
                LineItem curve = pane.AddCurve("Flutacao", x.ToArray(), y.ToArray(),
Color.DarkBlue);
                curve.Symbol.Fill = new Fill(Color.White);

                StandardTolerance tolerance = StandardTolerance.findStandardTolerance(
                    m_AnalysisProject.TypeOfService.ID, m_AnalysisProject.Standard.ID,
(int)EnumQoSRequisite.Jitter);

                //// Manually set the x axis range
                if (m_AnalysisProject.JitterAverage < tolerance.MinValue)
                    pane.YAxis.Scale.Min = m_AnalysisProject.JitterAverage - 20;
                else
                    pane.YAxis.Scale.Min = tolerance.MinValue - 20;

                if (m_AnalysisProject.JitterAverage > tolerance.MaxValue)
                    pane.YAxis.Scale.Max = m_AnalysisProject.JitterAverage + 20;
                else
                    pane.YAxis.Scale.Max = tolerance.MaxValue + 20;

                // Display the Y axis grid lines
                pane.YAxis.MajorGrid.IsVisible = true;
                pane.YAxis.MinorGrid.IsVisible = true;

                // Calculate the Axis Scale Ranges
                zedGraphJitter.AxisChange();

                // Calculate the Axis Scale Ranges
                zedGraphOutFlow.AxisChange();

                if (tolerance != null)
                {
                    double[] xt = new double[2];
                    double[] yt = new double[2];

                    xt[0] = 0;
                    xt[1] = time1.Seconds;

                    yt[0] = tolerance.MinValue;
                    yt[1] = tolerance.MinValue;

                    LineItem curveMin = pane.AddCurve("Tolerância Mínimo", xt, yt,
Color.Red);
                    curveMin.Symbol.Fill = new Fill(Color.White);

                    xt[0] = 0;
                    xt[1] = time1.Seconds;

                    yt[0] = tolerance.MaxValue;
                    yt[1] = tolerance.MaxValue;

                    if (tolerance.MaxValue < pane.YAxis.Scale.Max)
                    {
                        LineItem curveMax = pane.AddCurve("Tolerância Máximo", xt, yt,
Color.Red);
                        curveMax.Symbol.Fill = new Fill(Color.White);
                    }
                }

                // Calculate the Axis Scale Ranges
                zedGraphJitter.AxisChange();
            }
            catch (Exception ex)
            {
```

```
                throw new Exception("Erro ao listar Tipos de Serviço", ex);
            }
        }

        private void fillGraphOutFlow()
        {
            try
            {
                if (m_AnalysisProject == null)
                    return;

                if (m_AnalysisProject.Packages.Count == 0)
                    return;

                NetWorkPackageProcessed firstPack = m_AnalysisProject.Packages[0];
                NetWorkPackageProcessed                    lastPack                    =
m_AnalysisProject.Packages[m_AnalysisProject.Packages.Count - 1];

                TimeSpan time1 = lastPack.SourceTS - firstPack.SourceTS;

                List<double> x = new List<double>();
                List<double> y = new List<double>();

                long actualSecond = 0;
                double actualLength = 0;
                foreach (NetWorkPackageProcessed pack in m_AnalysisProject.Packages)
                {
                    TimeSpan temp2 = pack.SourceTS - firstPack.SourceTS;

                    if (temp2.Seconds <= actualSecond)
                    {
                        actualLength += pack.Length;
                    }
                    else
                    {
                        x.Add(actualSecond);
                        y.Add(actualLength);

                        actualSecond = temp2.Seconds;
                        actualLength = 0;
                    }
                }

                GraphPane pane = zedGraphOutFlow.GraphPane;
                pane.Title.Text  =  "Largura  de  Banda  com  Tolerancia  da  norma  "  +
m_AnalysisProject.Standard.Name;
                pane.XAxis.Title.Text = "Tempo (s)";
                pane.YAxis.Title.Text = "Bytes";

                pane.Chart.Fill    =    new    Fill(Color.FromArgb(255,    255,    245),
Color.FromArgb(255, 255, 190), 90F);

                pane.CurveList.Clear();
                LineItem   curve   =   pane.AddCurve("Largura   de   Banda",   x.ToArray(),
y.ToArray(), Color.Black);
                curve.Symbol.Fill = new Fill(Color.White);

                StandardTolerance tolerance = StandardTolerance.findStandardTolerance(
                    m_AnalysisProject.TypeOfService.ID,    m_AnalysisProject.Standard.ID,
(int)EnumQoSRequisite.Outflow);

                // Manually set the x axis range
                pane.XAxis.Scale.Min = 0;
                pane.XAxis.Scale.Max = actualSecond;

                if (m_AnalysisProject.LengthAverage < (tolerance.MinValue / 1000))
                    pane.YAxis.Scale.Min = m_AnalysisProject.LengthAverage - 10000;
                else
                    pane.YAxis.Scale.Min = tolerance.MinValue - 10000;

                if (m_AnalysisProject.LengthAverage > (tolerance.MaxValue / 1000))
                    pane.YAxis.Scale.Max = m_AnalysisProject.LengthAverage + 10000;
                else
                    pane.YAxis.Scale.Max = tolerance.MaxValue + 10000;

                // Display the Y axis grid lines
```

```
                pane.YAxis.MajorGrid.IsVisible = true;
                pane.YAxis.MinorGrid.IsVisible = true;

                // Calculate the Axis Scale Ranges
                zedGraphOutFlow.AxisChange();

                if (tolerance != null)
                {
                    double[] xt = new double[2];
                    double[] yt = new double[2];

                    xt[0] = 0;
                    xt[1] = time1.Seconds;

                    yt[0] = tolerance.MinValue;
                    yt[1] = tolerance.MinValue;

                    LineItem curveMin = pane.AddCurve("Tolerância Mínimo", xt, yt,
Color.Red);
                    curveMin.Symbol.Fill = new Fill(Color.White);

                    xt[0] = 0;
                    xt[1] = time1.Seconds;

                    yt[0] = tolerance.MaxValue;
                    yt[1] = tolerance.MaxValue;

                    if (tolerance.MaxValue < pane.YAxis.Scale.Max)
                    {
                        LineItem curveMax = pane.AddCurve("Tolerância Máximo", xt, yt,
Color.Red);
                        curveMax.Symbol.Fill = new Fill(Color.White);
                    }
                }

                // Calculate the Axis Scale Ranges
                zedGraphOutFlow.AxisChange();

            }
            catch (Exception ex)
            {
                throw new Exception("Erro ao listar Tipos de Serviço", ex);
            }
        }

        private void fillGraphPackageDelay()
        {
            try
            {
                if (m_AnalysisProject == null)
                    return;

                NetWorkPackageProcessed firstPack = m_AnalysisProject.Packages[0];
                NetWorkPackageProcessed                    lastPack                    =
m_AnalysisProject.Packages[m_AnalysisProject.Packages.Count - 1];

                TimeSpan time1 = lastPack.SourceTS - firstPack.SourceTS;

                List<double> x = new List<double>();
                List<double> y = new List<double>();

                double actualDelayTotal = 0;
                double actualDelayAverage = 0;
                double maxDelay = 0;
                long actualSecond = 0;
                long actualPackagesCount = 0;
                long totalPackages = 0;

                m_AnalysisProject.Packages = null;
                foreach (NetWorkPackageProcessed pack in m_AnalysisProject.Packages)
                {
                    TimeSpan temp2 = pack.SourceTS - firstPack.SourceTS;

                    if (temp2.Seconds <= actualSecond)
                    {
                        if (!pack.Losted)
```

```
                            actualDelayTotal += pack.Delay;

                        actualPackagesCount++;
                    }
                    else
                    {
                        actualDelayAverage = actualDelayTotal / actualPackagesCount;

                        if (actualDelayAverage > maxDelay)
                            maxDelay = actualDelayAverage;

                        x.Add(actualSecond);
                        y.Add(actualDelayAverage);

                        actualSecond = temp2.Seconds;
                        actualPackagesCount = 0;
                        actualDelayTotal = 0;
                    }

                    totalPackages++;
                }

                GraphPane pane = zedGraphPackageDelay.GraphPane;
                pane.Title.Text = "Atraso com Tolerancia da norma " +
m_AnalysisProject.Standard.Name;
                pane.XAxis.Title.Text = "Tempo (ms)";
                pane.YAxis.Title.Text = "Atraso (ms)";

                pane.Chart.Fill = new Fill(Color.FromArgb(255, 255, 245),
Color.FromArgb(255, 255, 190), 90F);

                pane.CurveList.Clear();
                LineItem curve = pane.AddCurve("Atraso", x.ToArray(), y.ToArray(),
Color.DarkGreen);
                curve.Symbol.Fill = new Fill(Color.White);

                StandardTolerance tolerance = StandardTolerance.findStandardTolerance(
                    m_AnalysisProject.TypeOfService.ID, m_AnalysisProject.Standard.ID,
(int)EnumQoSRequisite.PackageDelay);

                // Manually set the x axis range
                if (m_AnalysisProject.DelayAverage < tolerance.MinValue)
                    pane.YAxis.Scale.Min = m_AnalysisProject.DelayAverage - 100;
                else
                    pane.YAxis.Scale.Min = tolerance.MinValue - 100;

                if (m_AnalysisProject.DelayAverage > tolerance.MaxValue)
                    pane.YAxis.Scale.Max = m_AnalysisProject.DelayAverage + 100;
                else
                    pane.YAxis.Scale.Max = tolerance.MaxValue + 100;

                // Display the Y axis grid lines
                pane.YAxis.MajorGrid.IsVisible = true;
                pane.YAxis.MinorGrid.IsVisible = true;

                // Calculate the Axis Scale Ranges
                zedGraphOutFlow.AxisChange();

                if (tolerance != null)
                {
                    double[] xt = new double[2];
                    double[] yt = new double[2];

                    xt[0] = 0;
                    xt[1] = time1.Seconds;

                    yt[0] = tolerance.MinValue;
                    yt[1] = tolerance.MinValue;

                    LineItem curveMin = pane.AddCurve("Tolerância Mínimo", xt, yt,
Color.Red);
                    curveMin.Symbol.Fill = new Fill(Color.White);

                    xt[0] = 0;
                    xt[1] = time1.Seconds;
```

```
                    yt[0] = tolerance.MaxValue;
                    yt[1] = tolerance.MaxValue;

                    if (tolerance.MaxValue < pane.YAxis.Scale.Max)
                    {
                        LineItem curveMax = pane.AddCurve("Tolerância Máximo", xt, yt,
Color.Red);
                        curveMax.Symbol.Fill = new Fill(Color.White);
                    }
                }

                // Calculate the Axis Scale Ranges
                zedGraphPackageDelay.AxisChange();

            }
            catch (Exception ex)
            {
                throw new Exception("Erro ao listar Tipos de Serviço", ex);
            }
        }

        private void fillGraphPackageLoses()
        {
            try
            {
                if (m_AnalysisProject == null)
                    return;

                NetWorkPackageProcessed firstPack = m_AnalysisProject.Packages[0];
                NetWorkPackageProcessed                    lastPack                    =
m_AnalysisProject.Packages[m_AnalysisProject.Packages.Count - 1];

                TimeSpan time1 = lastPack.SourceTS - firstPack.SourceTS;

                List<double> x = new List<double>();
                List<double> y = new List<double>();

                long actualLoses = 0;
                long actualSecond = 0;
                long actualPackagesCount = 0;
                foreach (NetWorkPackageProcessed pack in m_AnalysisProject.Packages)
                {
                    TimeSpan temp2 = pack.SourceTS - firstPack.SourceTS;

                    if (temp2.Seconds <= actualSecond)
                    {
                        if(pack.Losted)
                            actualLoses++;

                        actualPackagesCount++;
                    }
                    else
                    {
                        x.Add(actualSecond);
                        y.Add(actualLoses * 100 / actualPackagesCount);

                        actualSecond = temp2.Seconds;
                        actualPackagesCount = 0;
                        actualLoses = 0;
                    }
                }

                GraphPane pane = zedGraphPackageLoses.GraphPane;
                pane.Title.Text    =    "Perdas    com    Tolerancia    da    norma    "    +
m_AnalysisProject.Standard.Name;
                pane.XAxis.Title.Text = "Tempo (ms)";
                pane.YAxis.Title.Text = "Perdas (%)";

                pane.Chart.Fill    =    new    Fill(Color.FromArgb(255,    255,    245),
Color.FromArgb(255, 255, 190), 90F);

                pane.CurveList.Clear();
                LineItem   curve   =   pane.AddCurve("Perdas",   x.ToArray(),   y.ToArray(),
Color.DarkOrange);
                curve.Symbol.Fill = new Fill(Color.White);
```

```
                StandardTolerance tolerance = StandardTolerance.findStandardTolerance(
                    m_AnalysisProject.TypeOfService.ID,    m_AnalysisProject.Standard.ID,
(int)EnumQoSRequisite.PackageLoses);

                // Manually set the x axis range
                pane.XAxis.Scale.Min = 0;
                pane.YAxis.Scale.Min = tolerance.MinValue - 10;
                pane.YAxis.Scale.Max = tolerance.MaxValue + 10;
                pane.XAxis.Scale.Max = actualSecond;

                // Display the Y axis grid lines
                pane.YAxis.MajorGrid.IsVisible = true;
                pane.YAxis.MinorGrid.IsVisible = true;

                // Calculate the Axis Scale Ranges
                zedGraphOutFlow.AxisChange();

                if (tolerance != null)
                {
                    double[] xt = new double[2];
                    double[] yt = new double[2];

                    xt[0] = 0;
                    xt[1] = time1.Seconds;

                    yt[0] = tolerance.MinValue;
                    yt[1] = tolerance.MinValue;

                    LineItem  curveMin  =  pane.AddCurve("Tolerância  Mínimo",  xt,  yt,
Color.Red);
                    curveMin.Symbol.Fill = new Fill(Color.White);

                    xt[0] = 0;
                    xt[1] = time1.Seconds;

                    yt[0] = tolerance.MaxValue;
                    yt[1] = tolerance.MaxValue;

                    if (tolerance.MaxValue < pane.YAxis.Scale.Max)
                    {
                        LineItem  curveMax  =  pane.AddCurve("Tolerância  Máximo",  xt,  yt,
Color.Red);
                        curveMax.Symbol.Fill = new Fill(Color.White);
                    }
                }

                // Calculate the Axis Scale Ranges
                zedGraphPackageLoses.AxisChange();

            }
            catch (Exception ex)
            {
                throw new Exception("Erro ao listar Tipos de Serviço", ex);
            }
        }

        private void btnResults_Click(object sender, EventArgs e)
        {
            try
            {
                m_AnalysisProject = (AnalysisProject)cboAnalysisProject.SelectedItem; ;

                string messsageErr = "";

                if(m_AnalysisProject == null)
                    messsageErr = "Selecione um Projeto Primeiro!";

                if (m_AnalysisProject.ID == 0)
                    messsageErr = "Selecione um Projeto Primeiro!";

                if (messsageErr != "")
                    throw new Exception(messsageErr);

                AnalysisProject ap = m_AnalysisProject;
                clear();
                m_AnalysisProject = ap;
```

```
                loadAnalisysProjectResults();
            }
            catch (Exception ex)
            {
                ErrorHandlerDialog.ShowErroHandler(ref ex);
            }
        }

        private void cboAnalysisProject_SelectedIndexChanged(object sender, EventArgs e)
        {
            try
            {
                AnalysisProject                    analysisProject                    =
(AnalysisProject)cboAnalysisProject.SelectedItem; ;

                txtAnalysisProjectDescription.Text = analysisProject.Description;

            }
            catch (Exception ex)
            {
                ErrorHandlerDialog.ShowErroHandler(ref ex);
            }
        }

    }
}
        namespace PackageAnalyzator
        {
            partial class AnalysisProjectResultDialog
            {
                /// <summary>
                /// Required designer variable.
                /// </summary>
                private System.ComponentModel.IContainer components = null;

                /// <summary>
                /// Clean up any resources being used.
                /// </summary>
                /// <param name="disposing">true if managed resources should be disposed;
otherwise, false.</param>
                protected override void Dispose(bool disposing)
                {
                    if (disposing && (components != null))
                    {
                        components.Dispose();
                    }
                    base.Dispose(disposing);
                }

                #region Windows Form Designer generated code

                /// <summary>
                /// Required method for Designer support - do not modify
                /// the contents of this method with the code editor.
                /// </summary>
                private void InitializeComponent()
                {
                    this.components = new System.ComponentModel.Container();
                    System.Windows.Forms.ListViewItem      listViewItem1      =      new
System.Windows.Forms.ListViewItem("Vazão");
                    System.Windows.Forms.ListViewItem      listViewItem2      =      new
System.Windows.Forms.ListViewItem("Jitter");
                    System.Windows.Forms.ListViewItem      listViewItem3      =      new
System.Windows.Forms.ListViewItem("Atraso");
                    System.Windows.Forms.ListViewItem      listViewItem4      =      new
System.Windows.Forms.ListViewItem("Perda");
                    this.toolTip = new System.Windows.Forms.ToolTip(this.components);
                    this.errorProvider                          =                          new
System.Windows.Forms.ErrorProvider(this.components);
                    this.tabControl = new System.Windows.Forms.TabControl();
                    this.tabPageAnalysis = new System.Windows.Forms.TabPage();
                    this.tableLayoutPanel2 = new System.Windows.Forms.TableLayoutPanel();
                    this.groupBox3 = new System.Windows.Forms.GroupBox();
                    this.txtTypeOfService = new System.Windows.Forms.TextBox();
                    this.lblDestinationAddress = new System.Windows.Forms.Label();
```

```
                    this.txtSourceAddress = new System.Windows.Forms.TextBox();
                    this.lblSourceAddress = new System.Windows.Forms.Label();
                    this.lblStandard = new System.Windows.Forms.Label();
                    this.txtDestinationAddress = new System.Windows.Forms.TextBox();
                    this.txtStandard = new System.Windows.Forms.TextBox();
                    this.lblTypeOfService = new System.Windows.Forms.Label();
                    this.groupBox2 = new System.Windows.Forms.GroupBox();
                    this.lstParametersResults = new System.Windows.Forms.ListView();
                    this.columnHeader1 = new System.Windows.Forms.ColumnHeader();
                    this.columnHeader2 = new System.Windows.Forms.ColumnHeader();
                    this.columnHeader3 = new System.Windows.Forms.ColumnHeader();
                    this.columnHeader4 = new System.Windows.Forms.ColumnHeader();
                    this.columnHeader5 = new System.Windows.Forms.ColumnHeader();
                    this.tabPageGraphs = new System.Windows.Forms.TabPage();
                    this.tabControlGraph = new System.Windows.Forms.TabControl();
                    this.tabPageOutFlow = new System.Windows.Forms.TabPage();
                    this.zedGraphOutFlow = new ZedGraph.ZedGraphControl();
                    this.tabPageJitter = new System.Windows.Forms.TabPage();
                    this.zedGraphJitter = new ZedGraph.ZedGraphControl();
                    this.tabPagePackageDelay = new System.Windows.Forms.TabPage();
                    this.zedGraphPackageDelay = new ZedGraph.ZedGraphControl();
                    this.tabPagePackageLoses = new System.Windows.Forms.TabPage();
                    this.zedGraphPackageLoses = new ZedGraph.ZedGraphControl();
                    this.groupBox1 = new System.Windows.Forms.GroupBox();
                    this.btnResults = new System.Windows.Forms.Button();
                    this.lblAnalisysProject = new System.Windows.Forms.Label();
                    this.cboAnalysisProject = new System.Windows.Forms.ComboBox();
                    this.netWorkPackageProcessedBindingSource            =            new
System.Windows.Forms.BindingSource(this.components);
                    this.tableLayoutPanel1 = new System.Windows.Forms.TableLayoutPanel();
                    this.txtAnalysisProjectDescription               =               new
System.Windows.Forms.TextBox();

((System.ComponentModel.ISupportInitialize)(this.errorProvider)).BeginInit();
                    this.tabControl.SuspendLayout();
                    this.tabPageAnalysis.SuspendLayout();
                    this.tableLayoutPanel2.SuspendLayout();
                    this.groupBox3.SuspendLayout();
                    this.groupBox2.SuspendLayout();
                    this.tabPageGraphs.SuspendLayout();
                    this.tabControlGraph.SuspendLayout();
                    this.tabPageOutFlow.SuspendLayout();
                    this.tabPageJitter.SuspendLayout();
                    this.tabPagePackageDelay.SuspendLayout();
                    this.tabPagePackageLoses.SuspendLayout();
                    this.groupBox1.SuspendLayout();

((System.ComponentModel.ISupportInitialize)(this.netWorkPackageProcessedBindingSource)).
BeginInit();
                    this.tableLayoutPanel1.SuspendLayout();
                    this.SuspendLayout();
                    //
                    // errorProvider
                    //
                    this.errorProvider.ContainerControl = this;
                    //
                    // tabControl
                    //
                    this.tabControl.Controls.Add(this.tabPageAnalysis);
                    this.tabControl.Controls.Add(this.tabPageGraphs);
                    this.tabControl.Dock = System.Windows.Forms.DockStyle.Fill;
                    this.tabControl.Location = new System.Drawing.Point(3, 79);
                    this.tabControl.Name = "tabControl";
                    this.tabControl.SelectedIndex = 0;
                    this.tabControl.Size = new System.Drawing.Size(877, 379);
                    this.tabControl.TabIndex = 0;
                    //
                    // tabPageAnalysis
                    //
                    this.tabPageAnalysis.Controls.Add(this.tableLayoutPanel2);
                    this.tabPageAnalysis.Location = new System.Drawing.Point(4, 22);
                    this.tabPageAnalysis.Name = "tabPageAnalysis";
                    this.tabPageAnalysis.Padding = new System.Windows.Forms.Padding(3);
                    this.tabPageAnalysis.Size = new System.Drawing.Size(869, 353);
                    this.tabPageAnalysis.TabIndex = 0;
                    this.tabPageAnalysis.Text = "Dados Gerais";
```

```
                    this.tabPageAnalysis.UseVisualStyleBackColor = true;
                    //
                    // tableLayoutPanel2
                    //
                    this.tableLayoutPanel2.ColumnCount = 1;
                    this.tableLayoutPanel2.ColumnStyles.Add(new
System.Windows.Forms.ColumnStyle(System.Windows.Forms.SizeType.Percent, 100F));
                    this.tableLayoutPanel2.Controls.Add(this.groupBox3, 0, 0);
                    this.tableLayoutPanel2.Controls.Add(this.groupBox2, 0, 1);
                    this.tableLayoutPanel2.Dock = System.Windows.Forms.DockStyle.Fill;
                    this.tableLayoutPanel2.Location = new System.Drawing.Point(3, 3);
                    this.tableLayoutPanel2.Name = "tableLayoutPanel2";
                    this.tableLayoutPanel2.RowCount = 2;
                    this.tableLayoutPanel2.RowStyles.Add(new
System.Windows.Forms.RowStyle(System.Windows.Forms.SizeType.Absolute, 67F));
                    this.tableLayoutPanel2.RowStyles.Add(new
System.Windows.Forms.RowStyle(System.Windows.Forms.SizeType.Percent, 100F));
                    this.tableLayoutPanel2.Size = new System.Drawing.Size(863, 347);
                    this.tableLayoutPanel2.TabIndex = 25;
                    //
                    // groupBox3
                    //
                    this.groupBox3.Controls.Add(this.txtTypeOfService);
                    this.groupBox3.Controls.Add(this.lblDestinationAddress);
                    this.groupBox3.Controls.Add(this.txtSourceAddress);
                    this.groupBox3.Controls.Add(this.lblSourceAddress);
                    this.groupBox3.Controls.Add(this.lblStandard);
                    this.groupBox3.Controls.Add(this.txtDestinationAddress);
                    this.groupBox3.Controls.Add(this.txtStandard);
                    this.groupBox3.Controls.Add(this.lblTypeOfService);
                    this.groupBox3.Dock = System.Windows.Forms.DockStyle.Fill;
                    this.groupBox3.Location = new System.Drawing.Point(3, 3);
                    this.groupBox3.Name = "groupBox3";
                    this.groupBox3.Size = new System.Drawing.Size(857, 61);
                    this.groupBox3.TabIndex = 0;
                    this.groupBox3.TabStop = false;
                    //
                    // txtTypeOfService
                    //
                    this.txtTypeOfService.Location = new System.Drawing.Point(5, 31);
                    this.txtTypeOfService.Name = "txtTypeOfService";
                    this.txtTypeOfService.ReadOnly = true;
                    this.txtTypeOfService.Size = new System.Drawing.Size(131, 20);
                    this.txtTypeOfService.TabIndex = 5;
                    //
                    // lblDestinationAddress
                    //
                    this.lblDestinationAddress.AutoSize = true;
                    this.lblDestinationAddress.Location  =  new  System.Drawing.Point(377,
15);
                    this.lblDestinationAddress.Name = "lblDestinationAddress";
                    this.lblDestinationAddress.Size = new System.Drawing.Size(59, 13);
                    this.lblDestinationAddress.TabIndex = 21;
                    this.lblDestinationAddress.Text = "IP Destino:";
                    //
                    // txtSourceAddress
                    //
                    this.txtSourceAddress.Location = new System.Drawing.Point(261, 31);
                    this.txtSourceAddress.Name = "txtSourceAddress";
                    this.txtSourceAddress.ReadOnly = true;
                    this.txtSourceAddress.Size = new System.Drawing.Size(113, 20);
                    this.txtSourceAddress.TabIndex = 20;
                    //
                    // lblSourceAddress
                    //
                    this.lblSourceAddress.AutoSize = true;
                    this.lblSourceAddress.Location = new System.Drawing.Point(258, 15);
                    this.lblSourceAddress.Name = "lblSourceAddress";
                    this.lblSourceAddress.Size = new System.Drawing.Size(56, 13);
                    this.lblSourceAddress.TabIndex = 19;
                    this.lblSourceAddress.Text = "IP Origem:";
                    //
                    // lblStandard
                    //
                    this.lblStandard.AutoSize = true;
                    this.lblStandard.Location = new System.Drawing.Point(139, 15);
```

```
                    this.lblStandard.Name = "lblStandard";
                    this.lblStandard.Size = new System.Drawing.Size(98, 13);
                    this.lblStandard.TabIndex = 6;
                    this.lblStandard.Text = "Norma Empregada:";
                    //
                    // txtDestinationAddress
                    //
                    this.txtDestinationAddress.Location  =  new  System.Drawing.Point(380,
31);
                    this.txtDestinationAddress.Name = "txtDestinationAddress";
                    this.txtDestinationAddress.ReadOnly = true;
                    this.txtDestinationAddress.Size = new System.Drawing.Size(113, 20);
                    this.txtDestinationAddress.TabIndex = 22;
                    //
                    // txtStandard
                    //
                    this.txtStandard.Location = new System.Drawing.Point(142, 31);
                    this.txtStandard.Name = "txtStandard";
                    this.txtStandard.ReadOnly = true;
                    this.txtStandard.Size = new System.Drawing.Size(113, 20);
                    this.txtStandard.TabIndex = 7;
                    //
                    // lblTypeOfService
                    //
                    this.lblTypeOfService.AutoSize = true;
                    this.lblTypeOfService.Location = new System.Drawing.Point(2, 15);
                    this.lblTypeOfService.Name = "lblTypeOfService";
                    this.lblTypeOfService.Size = new System.Drawing.Size(134, 13);
                    this.lblTypeOfService.TabIndex = 4;
                    this.lblTypeOfService.Text = "Tipo de Serviço Analisado:";
                    //
                    // groupBox2
                    //
                    this.groupBox2.Controls.Add(this.lstParametersResults);
                    this.groupBox2.Dock = System.Windows.Forms.DockStyle.Fill;
                    this.groupBox2.Location = new System.Drawing.Point(3, 70);
                    this.groupBox2.Name = "groupBox2";
                    this.groupBox2.Size = new System.Drawing.Size(857, 274);
                    this.groupBox2.TabIndex = 24;
                    this.groupBox2.TabStop = false;
                    this.groupBox2.Text = "Análise";
                    //
                    // lstParametersResults
                    //
                    this.lstParametersResults.Columns.AddRange(new
System.Windows.Forms.ColumnHeader[] {
                    this.columnHeader1,
                    this.columnHeader2,
                    this.columnHeader3,
                    this.columnHeader4,
                    this.columnHeader5});
                    this.lstParametersResults.Dock = System.Windows.Forms.DockStyle.Fill;
                    this.lstParametersResults.GridLines = true;
                    this.lstParametersResults.Items.AddRange(new
System.Windows.Forms.ListViewItem[] {
                    listViewItem1,
                    listViewItem2,
                    listViewItem3,
                    listViewItem4});
                    this.lstParametersResults.Location = new System.Drawing.Point(3, 16);
                    this.lstParametersResults.Name = "lstParametersResults";
                    this.lstParametersResults.Size = new System.Drawing.Size(851, 255);
                    this.lstParametersResults.TabIndex = 23;
                    this.lstParametersResults.UseCompatibleStateImageBehavior = false;
                    this.lstParametersResults.View = System.Windows.Forms.View.Details;
                    //
                    // columnHeader1
                    //
                    this.columnHeader1.Text = "Parâmetro QoS";
                    this.columnHeader1.Width = 131;
                    //
                    // columnHeader2
                    //
                    this.columnHeader2.Text = "Valor Médio";
                    this.columnHeader2.TextAlign                                          =
System.Windows.Forms.HorizontalAlignment.Right;
```

```
                    this.columnHeader2.Width = 119;
                    //
                    // columnHeader3
                    //
                    this.columnHeader3.Text = "Valor Mínimo Norma";
                    this.columnHeader3.TextAlign                                    =
System.Windows.Forms.HorizontalAlignment.Right;
                    this.columnHeader3.Width = 114;
                    //
                    // columnHeader4
                    //
                    this.columnHeader4.Text = "Valor Máximo Norma";
                    this.columnHeader4.TextAlign                                    =
System.Windows.Forms.HorizontalAlignment.Right;
                    this.columnHeader4.Width = 115;
                    //
                    // columnHeader5
                    //
                    this.columnHeader5.Text = "Comentário Análise";
                    this.columnHeader5.Width = 217;
                    //
                    // tabPageGraphs
                    //
                    this.tabPageGraphs.Controls.Add(this.tabControlGraph);
                    this.tabPageGraphs.Location = new System.Drawing.Point(4, 22);
                    this.tabPageGraphs.Name = "tabPageGraphs";
                    this.tabPageGraphs.Padding = new System.Windows.Forms.Padding(3);
                    this.tabPageGraphs.Size = new System.Drawing.Size(869, 353);
                    this.tabPageGraphs.TabIndex = 1;
                    this.tabPageGraphs.Text = "Gráficos";
                    this.tabPageGraphs.UseVisualStyleBackColor = true;
                    //
                    // tabControlGraph
                    //
                    this.tabControlGraph.Controls.Add(this.tabPageOutFlow);
                    this.tabControlGraph.Controls.Add(this.tabPageJitter);
                    this.tabControlGraph.Controls.Add(this.tabPagePackageDelay);
                    this.tabControlGraph.Controls.Add(this.tabPagePackageLoses);
                    this.tabControlGraph.Dock = System.Windows.Forms.DockStyle.Fill;
                    this.tabControlGraph.Location = new System.Drawing.Point(3, 3);
                    this.tabControlGraph.Name = "tabControlGraph";
                    this.tabControlGraph.SelectedIndex = 0;
                    this.tabControlGraph.Size = new System.Drawing.Size(863, 347);
                    this.tabControlGraph.TabIndex = 1;
                    //
                    // tabPageOutFlow
                    //
                    this.tabPageOutFlow.Controls.Add(this.zedGraphOutFlow);
                    this.tabPageOutFlow.Location = new System.Drawing.Point(4, 22);
                    this.tabPageOutFlow.Name = "tabPageOutFlow";
                    this.tabPageOutFlow.Size = new System.Drawing.Size(855, 321);
                    this.tabPageOutFlow.TabIndex = 2;
                    this.tabPageOutFlow.Text = "Largura de Banda";
                    this.tabPageOutFlow.UseVisualStyleBackColor = true;
                    //
                    // zedGraphOutFlow
                    //
                    this.zedGraphOutFlow.Dock = System.Windows.Forms.DockStyle.Fill;
                    this.zedGraphOutFlow.EditButtons                                =
System.Windows.Forms.MouseButtons.Left;
                    this.zedGraphOutFlow.Location = new System.Drawing.Point(0, 0);
                    this.zedGraphOutFlow.Name = "zedGraphOutFlow";
                    this.zedGraphOutFlow.PanModifierKeys                            =
((System.Windows.Forms.Keys)((System.Windows.Forms.Keys.Shift                       |
System.Windows.Forms.Keys.None)));
                    this.zedGraphOutFlow.ScrollGrace = 0;
                    this.zedGraphOutFlow.ScrollMaxX = 0;
                    this.zedGraphOutFlow.ScrollMaxY = 0;
                    this.zedGraphOutFlow.ScrollMaxY2 = 0;
                    this.zedGraphOutFlow.ScrollMinX = 0;
                    this.zedGraphOutFlow.ScrollMinY = 0;
                    this.zedGraphOutFlow.ScrollMinY2 = 0;
                    this.zedGraphOutFlow.Size = new System.Drawing.Size(855, 321);
                    this.zedGraphOutFlow.TabIndex = 1;
                    //
                    // tabPageJitter
```

```
                    //
                    this.tabPageJitter.Controls.Add(this.zedGraphJitter);
                    this.tabPageJitter.Location = new System.Drawing.Point(4, 22);
                    this.tabPageJitter.Name = "tabPageJitter";
                    this.tabPageJitter.Padding = new System.Windows.Forms.Padding(3);
                    this.tabPageJitter.Size = new System.Drawing.Size(855, 321);
                    this.tabPageJitter.TabIndex = 0;
                    this.tabPageJitter.Text = "Flutuacao";
                    this.tabPageJitter.UseVisualStyleBackColor = true;
                    //
                    // zedGraphJitter
                    //
                    this.zedGraphJitter.Dock = System.Windows.Forms.DockStyle.Fill;
                    this.zedGraphJitter.EditButtons                                    =
System.Windows.Forms.MouseButtons.Left;
                    this.zedGraphJitter.Location = new System.Drawing.Point(3, 3);
                    this.zedGraphJitter.Name = "zedGraphJitter";
                    this.zedGraphJitter.PanModifierKeys                                =
((System.Windows.Forms.Keys)((System.Windows.Forms.Keys.Shift                          |
System.Windows.Forms.Keys.None)));
                    this.zedGraphJitter.ScrollGrace = 0;
                    this.zedGraphJitter.ScrollMaxX = 0;
                    this.zedGraphJitter.ScrollMaxY = 0;
                    this.zedGraphJitter.ScrollMaxY2 = 0;
                    this.zedGraphJitter.ScrollMinX = 0;
                    this.zedGraphJitter.ScrollMinY = 0;
                    this.zedGraphJitter.ScrollMinY2 = 0;
                    this.zedGraphJitter.Size = new System.Drawing.Size(849, 315);
                    this.zedGraphJitter.TabIndex = 0;
                    //
                    // tabPagePackageDelay
                    //
                    this.tabPagePackageDelay.Controls.Add(this.zedGraphPackageDelay);
                    this.tabPagePackageDelay.Location = new System.Drawing.Point(4, 22);
                    this.tabPagePackageDelay.Name = "tabPagePackageDelay";
                    this.tabPagePackageDelay.Padding                =                 new
System.Windows.Forms.Padding(3);
                    this.tabPagePackageDelay.Size = new System.Drawing.Size(855, 321);
                    this.tabPagePackageDelay.TabIndex = 1;
                    this.tabPagePackageDelay.Text = "Atraso";
                    this.tabPagePackageDelay.UseVisualStyleBackColor = true;
                    //
                    // zedGraphPackageDelay
                    //
                    this.zedGraphPackageDelay.Dock = System.Windows.Forms.DockStyle.Fill;
                    this.zedGraphPackageDelay.EditButtons                              =
System.Windows.Forms.MouseButtons.Left;
                    this.zedGraphPackageDelay.Location = new System.Drawing.Point(3, 3);
                    this.zedGraphPackageDelay.Name = "zedGraphPackageDelay";
                    this.zedGraphPackageDelay.PanModifierKeys                          =
((System.Windows.Forms.Keys)((System.Windows.Forms.Keys.Shift                          |
System.Windows.Forms.Keys.None)));
                    this.zedGraphPackageDelay.ScrollGrace = 0;
                    this.zedGraphPackageDelay.ScrollMaxX = 0;
                    this.zedGraphPackageDelay.ScrollMaxY = 0;
                    this.zedGraphPackageDelay.ScrollMaxY2 = 0;
                    this.zedGraphPackageDelay.ScrollMinX = 0;
                    this.zedGraphPackageDelay.ScrollMinY = 0;
                    this.zedGraphPackageDelay.ScrollMinY2 = 0;
                    this.zedGraphPackageDelay.Size = new System.Drawing.Size(849, 315);
                    this.zedGraphPackageDelay.TabIndex = 1;
                    //
                    // tabPagePackageLoses
                    //
                    this.tabPagePackageLoses.Controls.Add(this.zedGraphPackageLoses);
                    this.tabPagePackageLoses.Location = new System.Drawing.Point(4, 22);
                    this.tabPagePackageLoses.Name = "tabPagePackageLoses";
                    this.tabPagePackageLoses.Size = new System.Drawing.Size(855, 321);
                    this.tabPagePackageLoses.TabIndex = 3;
                    this.tabPagePackageLoses.Text = "Perdas";
                    this.tabPagePackageLoses.UseVisualStyleBackColor = true;
                    //
                    // zedGraphPackageLoses
                    //
                    this.zedGraphPackageLoses.Dock = System.Windows.Forms.DockStyle.Fill;
```

```
                    this.zedGraphPackageLoses.EditButtons                                   =
System.Windows.Forms.MouseButtons.Left;
                    this.zedGraphPackageLoses.Location = new System.Drawing.Point(0, 0);
                    this.zedGraphPackageLoses.Name = "zedGraphPackageLoses";
                    this.zedGraphPackageLoses.PanModifierKeys                               =
((System.Windows.Forms.Keys)((System.Windows.Forms.Keys.Shift                               |
System.Windows.Forms.Keys.None)));
                    this.zedGraphPackageLoses.ScrollGrace = 0;
                    this.zedGraphPackageLoses.ScrollMaxX = 0;
                    this.zedGraphPackageLoses.ScrollMaxY = 0;
                    this.zedGraphPackageLoses.ScrollMaxY2 = 0;
                    this.zedGraphPackageLoses.ScrollMinX = 0;
                    this.zedGraphPackageLoses.ScrollMinY = 0;
                    this.zedGraphPackageLoses.ScrollMinY2 = 0;
                    this.zedGraphPackageLoses.Size = new System.Drawing.Size(855, 321);
                    this.zedGraphPackageLoses.TabIndex = 1;
                    //
                    // groupBox1
                    //
                    this.groupBox1.Controls.Add(this.txtAnalysisProjectDescription);
                    this.groupBox1.Controls.Add(this.btnResults);
                    this.groupBox1.Controls.Add(this.lblAnalisysProject);
                    this.groupBox1.Controls.Add(this.cboAnalysisProject);
                    this.groupBox1.Dock = System.Windows.Forms.DockStyle.Fill;
                    this.groupBox1.Location = new System.Drawing.Point(3, 3);
                    this.groupBox1.Name = "groupBox1";
                    this.groupBox1.Size = new System.Drawing.Size(877, 70);
                    this.groupBox1.TabIndex = 1;
                    this.groupBox1.TabStop = false;
                    this.groupBox1.Text = "Projeto de Análise";
                    //
                    // btnResults
                    //
                    this.btnResults.FlatStyle = System.Windows.Forms.FlatStyle.Popup;
                    this.btnResults.Location = new System.Drawing.Point(161, 34);
                    this.btnResults.Name = "btnResults";
                    this.btnResults.Size = new System.Drawing.Size(102, 25);
                    this.btnResults.TabIndex = 16;
                    this.btnResults.Text = "Resultados";
                    this.btnResults.UseVisualStyleBackColor = true;
                    this.btnResults.Click                       +=                      new
System.EventHandler(this.btnResults_Click);
                    //
                    // lblAnalisysProject
                    //
                    this.lblAnalisysProject.AutoSize = true;
                    this.lblAnalisysProject.Location = new System.Drawing.Point(12, 22);
                    this.lblAnalisysProject.Name = "lblAnalisysProject";
                    this.lblAnalisysProject.Size = new System.Drawing.Size(43, 13);
                    this.lblAnalisysProject.TabIndex = 18;
                    this.lblAnalisysProject.Text = "Projeto:";
                    //
                    // cboAnalysisProject
                    //
                    this.cboAnalysisProject.DropDownStyle                                   =
System.Windows.Forms.ComboBoxStyle.DropDownList;
                    this.cboAnalysisProject.FormattingEnabled = true;
                    this.cboAnalysisProject.Location = new System.Drawing.Point(15, 38);
                    this.cboAnalysisProject.Name = "cboAnalysisProject";
                    this.cboAnalysisProject.Size = new System.Drawing.Size(140, 21);
                    this.cboAnalysisProject.TabIndex = 17;
                    this.cboAnalysisProject.SelectedIndexChanged          +=          new
System.EventHandler(this.cboAnalysisProject_SelectedIndexChanged);
                    //
                    // netWorkPackageProcessedBindingSource
                    //
                    this.netWorkPackageProcessedBindingSource.DataSource                    =
typeof(PackageAnalyzator.NetWorkPackageProcessed);
                    //
                    // tableLayoutPanel1
                    //
                    this.tableLayoutPanel1.ColumnCount = 1;
                    this.tableLayoutPanel1.ColumnStyles.Add(new
System.Windows.Forms.ColumnStyle(System.Windows.Forms.SizeType.Percent, 100F));
                    this.tableLayoutPanel1.Controls.Add(this.groupBox1, 0, 0);
                    this.tableLayoutPanel1.Controls.Add(this.tabControl, 0, 1);
```

```
                    this.tableLayoutPanel1.Dock = System.Windows.Forms.DockStyle.Fill;
                    this.tableLayoutPanel1.Location = new System.Drawing.Point(0, 0);
                    this.tableLayoutPanel1.Name = "tableLayoutPanel1";
                    this.tableLayoutPanel1.RowCount = 2;
                    this.tableLayoutPanel1.RowStyles.Add(new
System.Windows.Forms.RowStyle(System.Windows.Forms.SizeType.Absolute, 76F));
                    this.tableLayoutPanel1.RowStyles.Add(new
System.Windows.Forms.RowStyle(System.Windows.Forms.SizeType.Percent, 100F));
                    this.tableLayoutPanel1.Size = new System.Drawing.Size(883, 461);
                    this.tableLayoutPanel1.TabIndex = 2;
                    //
                    // txtAnalysisProjectDescription
                    //
                    this.txtAnalysisProjectDescription.Location           =           new
System.Drawing.Point(294, 23);
                    this.txtAnalysisProjectDescription.Multiline = true;
                    this.txtAnalysisProjectDescription.Name                                =
"txtAnalysisProjectDescription";
                    this.txtAnalysisProjectDescription.ReadOnly = true;
                    this.txtAnalysisProjectDescription.Size             =             new
System.Drawing.Size(569, 35);
                    this.txtAnalysisProjectDescription.TabIndex = 19;
                    //
                    // AnalysisProjectResultDialog
                    //
                    this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);
                    this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
                    this.ClientSize = new System.Drawing.Size(883, 461);
                    this.Controls.Add(this.tableLayoutPanel1);
                    this.Name = "AnalysisProjectResultDialog";
                    this.StartPosition                                                     =
System.Windows.Forms.FormStartPosition.CenterScreen;
                    this.Text = "Resultados de Análises";
                    this.Load                              +=                              new
System.EventHandler(this.AnalysisProjectProcessedDialog_Load);

((System.ComponentModel.ISupportInitialize)(this.errorProvider)).EndInit();
                    this.tabControl.ResumeLayout(false);
                    this.tabPageAnalysis.ResumeLayout(false);
                    this.tableLayoutPanel2.ResumeLayout(false);
                    this.groupBox3.ResumeLayout(false);
                    this.groupBox3.PerformLayout();
                    this.groupBox2.ResumeLayout(false);
                    this.tabPageGraphs.ResumeLayout(false);
                    this.tabControlGraph.ResumeLayout(false);
                    this.tabPageOutFlow.ResumeLayout(false);
                    this.tabPageJitter.ResumeLayout(false);
                    this.tabPagePackageDelay.ResumeLayout(false);
                    this.tabPagePackageLoses.ResumeLayout(false);
                    this.groupBox1.ResumeLayout(false);
                    this.groupBox1.PerformLayout();

((System.ComponentModel.ISupportInitialize)(this.netWorkPackageProcessedBindingSource)).
EndInit();
                    this.tableLayoutPanel1.ResumeLayout(false);
                    this.ResumeLayout(false);

                }

                #endregion

                private System.Windows.Forms.ToolTip toolTip;
                private System.Windows.Forms.ErrorProvider errorProvider;
                private System.Windows.Forms.GroupBox groupBox1;
                private System.Windows.Forms.TabControl tabControl;
                private System.Windows.Forms.TabPage tabPageAnalysis;
                private System.Windows.Forms.TabPage tabPageGraphs;
                private System.Windows.Forms.Button btnResults;
                private System.Windows.Forms.Label lblAnalisysProject;
                private System.Windows.Forms.ComboBox cboAnalysisProject;
                private System.Windows.Forms.TextBox txtStandard;
                private System.Windows.Forms.Label lblStandard;
                private System.Windows.Forms.TextBox txtTypeOfService;
                private System.Windows.Forms.Label lblTypeOfService;
                private System.Windows.Forms.TextBox txtSourceAddress;
                private System.Windows.Forms.TextBox txtDestinationAddress;
```

```
                private System.Windows.Forms.Label lblSourceAddress;
                private System.Windows.Forms.Label lblDestinationAddress;
                private System.Windows.Forms.GroupBox groupBox2;
                private System.Windows.Forms.ListView lstParametersResults;
                private System.Windows.Forms.ColumnHeader columnHeader1;
                private System.Windows.Forms.ColumnHeader columnHeader2;
                private System.Windows.Forms.ColumnHeader columnHeader3;
                private System.Windows.Forms.ColumnHeader columnHeader4;
                private System.Windows.Forms.ColumnHeader columnHeader5;
                private ZedGraph.ZedGraphControl zedGraphJitter;
                private System.Windows.Forms.TabControl tabControlGraph;
                private System.Windows.Forms.TabPage tabPageJitter;
                private System.Windows.Forms.TabPage tabPagePackageDelay;
                private                                 System.Windows.Forms.BindingSource
netWorkPackageProcessedBindingSource;
                private System.Windows.Forms.TabPage tabPageOutFlow;
                private System.Windows.Forms.TabPage tabPagePackageLoses;
                private ZedGraph.ZedGraphControl zedGraphOutFlow;
                private ZedGraph.ZedGraphControl zedGraphPackageDelay;
                private ZedGraph.ZedGraphControl zedGraphPackageLoses;
                private System.Windows.Forms.TableLayoutPanel tableLayoutPanel1;
                private System.Windows.Forms.TableLayoutPanel tableLayoutPanel2;
                private System.Windows.Forms.GroupBox groupBox3;
                private System.Windows.Forms.TextBox txtAnalysisProjectDescription;
            }
}
```

2.4. A Classe MainForm

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;

using ObjectLibrary;
using ObjectLibrary.Communication;

namespace PackageAnalyzator
{
    public partial class MainForm : Form
    {

        public MainForm()
        {
            InitializeComponent();
            CommunicationArea.AppPath = Application.StartupPath;
        }


        private void ExitToolsStripMenuItem_Click(object sender, EventArgs e)
        {
            Application.Exit();
        }

        private void snifferToolStripMenuItem_Click(object sender, EventArgs e)
        {
            PackageSnifferDialog frm = new PackageSnifferDialog();
            frm.MdiParent = this;
            frm.Show();
        }

        private void logImporterToolStripMenuItem_Click(object sender, EventArgs e)
        {
            LogImporterDialog frm = new LogImporterDialog();
            frm.MdiParent = this;
            frm.Show();
        }

        private void MainForm_FormClosing(object sender, FormClosingEventArgs e)
```

```
        {
            try
            {
                if (ObjectLibrary.Communication.DataBase.Connection.State !=
ConnectionState.Closed)
                    ObjectLibrary.Communication.DataBase.Connection.Close();
            }
            catch (Exception ex)
            {
                if (ex.InnerException == null)
                    MessageBox.Show(ex.Message);
                else
                    MessageBox.Show(ex.InnerException.Message);
            }
        }

        private void TypeofServiceToolStripMenuItem_Click(object sender, EventArgs e)
        {
            TypeOfServiceDialog frm = new TypeOfServiceDialog();
            frm.MdiParent = this;
            frm.Show();
        }

        private void StandardToleranceToolStripMenuItem_Click(object sender, EventArgs
e)
        {
            StandardToleranceDialog frm = new StandardToleranceDialog();
            frm.MdiParent = this;
            frm.Show();
        }

        private void PackageProcessToolStripMenuItem_Click(object sender, EventArgs e)
        {
            PackageProcessDialog frm = new PackageProcessDialog();
            frm.MdiParent = this;
            frm.Show();
        }

        private void normasToolStripMenuItem_Click(object sender, EventArgs e)
        {
            StandardDialog frm = new StandardDialog();
            frm.MdiParent = this;
            frm.Show();
        }

        private void cadastroDeProjetosDeAnáliseToolStripMenuItem_Click(object sender,
EventArgs e)
        {
            AnalysisProjectDialog frm = new AnalysisProjectDialog();
            frm.MdiParent = this;
            frm.Show();
        }

        private void históricoDeAnalisesToolStripMenuItem_Click(object sender, EventArgs
e)
        {
            AnalysisProjectResultDialog frm = new AnalysisProjectResultDialog();
            frm.MdiParent = this;
            frm.Show();
        }
    }
}
namespace PackageAnalyzator
{
    partial class MainForm
    {
        /// <summary>
        /// Required designer variable.
        /// </summary>
        private System.ComponentModel.IContainer components = null;

        /// <summary>
        /// Clean up any resources being used.
        /// </summary>
        /// <param name="disposing">true if managed resources should be disposed;
otherwise, false.</param>
```

```
        protected override void Dispose(bool disposing)
        {
            if (disposing && (components != null))
            {
                components.Dispose();
            }
            base.Dispose(disposing);
        }

        #region Windows Form Designer generated code

        /// <summary>
        /// Required method for Designer support - do not modify
        /// the contents of this method with the code editor.
        /// </summary>
        private void InitializeComponent()
        {
            this.components = new System.ComponentModel.Container();
            this.menuStrip = new System.Windows.Forms.MenuStrip();
            this.arquivoToolStripMenuItem = new
System.Windows.Forms.ToolStripMenuItem();
            this.farejadorToolStripMenuItem = new
System.Windows.Forms.ToolStripMenuItem();
            this.toolStripMenuItem1 = new System.Windows.Forms.ToolStripSeparator();
            this.importadorDeHistoricoToolStripMenuItem = new
System.Windows.Forms.ToolStripMenuItem();
            this.toolStripMenuItem3 = new System.Windows.Forms.ToolStripSeparator();
            this.sairToolStripMenuItem = new System.Windows.Forms.ToolStripMenuItem();
            this.analizadorToolStripMenuItem = new
System.Windows.Forms.ToolStripMenuItem();
            this.cadastroDeProjetosDeAnáliseToolStripMenuItem = new
System.Windows.Forms.ToolStripMenuItem();
            this.toolStripMenuItem2 = new System.Windows.Forms.ToolStripSeparator();
            this.PackageProcessToolStripMenuItem = new
System.Windows.Forms.ToolStripMenuItem();
            this.históricoDeAnalisesToolStripMenuItem = new
System.Windows.Forms.ToolStripMenuItem();
            this.cadastrosToolStripMenuItem = new
System.Windows.Forms.ToolStripMenuItem();
            this.TypeofServiceToolStripMenuItem = new
System.Windows.Forms.ToolStripMenuItem();
            this.normasToolStripMenuItem = new System.Windows.Forms.ToolStripMenuItem();
            this.StandardToleranceToolStripMenuItem = new
System.Windows.Forms.ToolStripMenuItem();
            this.ToolTip = new System.Windows.Forms.ToolTip(this.components);
            this.menuStrip.SuspendLayout();
            this.SuspendLayout();
            //
            // menuStrip
            //
            this.menuStrip.Items.AddRange(new System.Windows.Forms.ToolStripItem[] {
            this.arquivoToolStripMenuItem,
            this.analizadorToolStripMenuItem,
            this.cadastrosToolStripMenuItem});
            this.menuStrip.Location = new System.Drawing.Point(0, 0);
            this.menuStrip.Name = "menuStrip";
            this.menuStrip.Size = new System.Drawing.Size(632, 24);
            this.menuStrip.TabIndex = 0;
            this.menuStrip.Text = "MenuStrip";
            //
            // arquivoToolStripMenuItem
            //
            this.arquivoToolStripMenuItem.DropDownItems.AddRange(new
System.Windows.Forms.ToolStripItem[] {
            this.farejadorToolStripMenuItem,
            this.toolStripMenuItem1,
            this.importadorDeHistoricoToolStripMenuItem,
            this.toolStripMenuItem3,
            this.sairToolStripMenuItem});
            this.arquivoToolStripMenuItem.Name = "arquivoToolStripMenuItem";
            this.arquivoToolStripMenuItem.Size = new System.Drawing.Size(56, 20);
            this.arquivoToolStripMenuItem.Text = "Arquivo";
            //
            // farejadorToolStripMenuItem
            //
            this.farejadorToolStripMenuItem.Name = "farejadorToolStripMenuItem";
```

```
            this.farejadorToolStripMenuItem.Size = new System.Drawing.Size(198, 22);
            this.farejadorToolStripMenuItem.Text = "Farejador";
            this.farejadorToolStripMenuItem.Click += new
System.EventHandler(this.snifferToolStripMenuItem_Click);
            //
            // toolStripMenuItem1
            //
            this.toolStripMenuItem1.Name = "toolStripMenuItem1";
            this.toolStripMenuItem1.Size = new System.Drawing.Size(195, 6);
            //
            // importadorDeHistoricoToolStripMenuItem
            //
            this.importadorDeHistoricoToolStripMenuItem.Name =
"importadorDeHistoricoToolStripMenuItem";
            this.importadorDeHistoricoToolStripMenuItem.Size = new
System.Drawing.Size(198, 22);
            this.importadorDeHistoricoToolStripMenuItem.Text = "Importador de
Historico";
            this.importadorDeHistoricoToolStripMenuItem.Click += new
System.EventHandler(this.logImporterToolStripMenuItem_Click);
            //
            // toolStripMenuItem3
            //
            this.toolStripMenuItem3.Name = "toolStripMenuItem3";
            this.toolStripMenuItem3.Size = new System.Drawing.Size(195, 6);
            //
            // sairToolStripMenuItem
            //
            this.sairToolStripMenuItem.Name = "sairToolStripMenuItem";
            this.sairToolStripMenuItem.Size = new System.Drawing.Size(198, 22);
            this.sairToolStripMenuItem.Text = "Sair";
            this.sairToolStripMenuItem.Click += new
System.EventHandler(this.ExitToolsStripMenuItem_Click);
            //
            // analizadorToolStripMenuItem
            //
            this.analizadorToolStripMenuItem.DropDownItems.AddRange(new
System.Windows.Forms.ToolStripItem[] {
            this.cadastroDeProjetosDeAnáliseToolStripMenuItem,
            this.toolStripMenuItem2,
            this.PackageProcessToolStripMenuItem,
            this.históricoDeAnalisesToolStripMenuItem});
            this.analizadorToolStripMenuItem.Name = "analizadorToolStripMenuItem";
            this.analizadorToolStripMenuItem.Size = new System.Drawing.Size(69, 20);
            this.analizadorToolStripMenuItem.Text = "Analizador";
            //
            // cadastroDeProjetosDeAnáliseToolStripMenuItem
            //
            this.cadastroDeProjetosDeAnáliseToolStripMenuItem.Name =
"cadastroDeProjetosDeAnáliseToolStripMenuItem";
            this.cadastroDeProjetosDeAnáliseToolStripMenuItem.Size = new
System.Drawing.Size(239, 22);
            this.cadastroDeProjetosDeAnáliseToolStripMenuItem.Text = "Cadastro de
Projetos de Análise";
            this.cadastroDeProjetosDeAnáliseToolStripMenuItem.Click += new
System.EventHandler(this.cadastroDeProjetosDeAnáliseToolStripMenuItem_Click);
            //
            // toolStripMenuItem2
            //
            this.toolStripMenuItem2.Name = "toolStripMenuItem2";
            this.toolStripMenuItem2.Size = new System.Drawing.Size(236, 6);
            //
            // PackageProcessToolStripMenuItem
            //
            this.PackageProcessToolStripMenuItem.Name =
"PackageProcessToolStripMenuItem";
            this.PackageProcessToolStripMenuItem.Size = new System.Drawing.Size(239,
22);
            this.PackageProcessToolStripMenuItem.Text = "Processar Pacotes de Rede";
            this.PackageProcessToolStripMenuItem.Click += new
System.EventHandler(this.PackageProcessToolStripMenuItem_Click);
            //
            // históricoDeAnalisesToolStripMenuItem
            //
            this.históricoDeAnalisesToolStripMenuItem.Name =
"históricoDeAnalisesToolStripMenuItem";
```

```
            this.históricoDeAnalisesToolStripMenuItem.Size = new
System.Drawing.Size(239, 22);
            this.históricoDeAnalisesToolStripMenuItem.Text = "Histórico de Analises";
            this.históricoDeAnalisesToolStripMenuItem.Click += new
System.EventHandler(this.históricoDeAnalisesToolStripMenuItem_Click);
            //
            // cadastrosToolStripMenuItem
            //
            this.cadastrosToolStripMenuItem.DropDownItems.AddRange(new
System.Windows.Forms.ToolStripItem[] {
            this.TypeofServiceToolStripMenuItem,
            this.normasToolStripMenuItem,
            this.StandardToleranceToolStripMenuItem});
            this.cadastrosToolStripMenuItem.Name = "cadastrosToolStripMenuItem";
            this.cadastrosToolStripMenuItem.Size = new System.Drawing.Size(68, 20);
            this.cadastrosToolStripMenuItem.Text = "Cadastros";
            //
            // TypeofServiceToolStripMenuItem
            //
            this.TypeofServiceToolStripMenuItem.Name = "TypeofServiceToolStripMenuItem";
            this.TypeofServiceToolStripMenuItem.Size = new System.Drawing.Size(225, 22);
            this.TypeofServiceToolStripMenuItem.Text = "Tipo de Serviço";
            this.TypeofServiceToolStripMenuItem.Click += new
System.EventHandler(this.TypeofServiceToolStripMenuItem_Click);
            //
            // normasToolStripMenuItem
            //
            this.normasToolStripMenuItem.Name = "normasToolStripMenuItem";
            this.normasToolStripMenuItem.Size = new System.Drawing.Size(225, 22);
            this.normasToolStripMenuItem.Text = "Normas";
            this.normasToolStripMenuItem.Click += new
System.EventHandler(this.normasToolStripMenuItem_Click);
            //
            // StandardToleranceToolStripMenuItem
            //
            this.StandardToleranceToolStripMenuItem.Name =
"StandardToleranceToolStripMenuItem";
            this.StandardToleranceToolStripMenuItem.Size = new System.Drawing.Size(225,
22);
            this.StandardToleranceToolStripMenuItem.Text = "Tolerância de Tipo de
Serviço";
            this.StandardToleranceToolStripMenuItem.Click += new
System.EventHandler(this.StandardToleranceToolStripMenuItem_Click);
            //
            // MainForm
            //
            this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);
            this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
            this.ClientSize = new System.Drawing.Size(632, 453);
            this.Controls.Add(this.menuStrip);
            this.IsMdiContainer = true;
            this.MainMenuStrip = this.menuStrip;
            this.Name = "MainForm";
            this.StartPosition = System.Windows.Forms.FormStartPosition.CenterScreen;
            this.Text = "Analizador de Pacotes";
            this.WindowState = System.Windows.Forms.FormWindowState.Maximized;
            this.FormClosing += new
System.Windows.Forms.FormClosingEventHandler(this.MainForm_FormClosing);
            this.menuStrip.ResumeLayout(false);
            this.menuStrip.PerformLayout();
            this.ResumeLayout(false);
            this.PerformLayout();

        }
        #endregion


        private System.Windows.Forms.MenuStrip menuStrip;
        private System.Windows.Forms.ToolTip ToolTip;
        private System.Windows.Forms.ToolStripMenuItem analizadorToolStripMenuItem;
        private System.Windows.Forms.ToolStripMenuItem cadastrosToolStripMenuItem;
        private System.Windows.Forms.ToolStripMenuItem TypeofServiceToolStripMenuItem;
        private System.Windows.Forms.ToolStripMenuItem
StandardToleranceToolStripMenuItem;
        private System.Windows.Forms.ToolStripMenuItem PackageProcessToolStripMenuItem;
        private System.Windows.Forms.ToolStripMenuItem normasToolStripMenuItem;
```

```
        private System.Windows.Forms.ToolStripMenuItem
cadastroDeProjetosDeAnáliseToolStripMenuItem;
        private System.Windows.Forms.ToolStripSeparator toolStripMenuItem2;
        private System.Windows.Forms.ToolStripMenuItem
históricoDeAnalisesToolStripMenuItem;
        private System.Windows.Forms.ToolStripMenuItem arquivoToolStripMenuItem;
        private System.Windows.Forms.ToolStripMenuItem farejadorToolStripMenuItem;
        private System.Windows.Forms.ToolStripSeparator toolStripMenuItem1;
        private System.Windows.Forms.ToolStripMenuItem
importadorDeHistoricoToolStripMenuItem;
        private System.Windows.Forms.ToolStripSeparator toolStripMenuItem3;
        private System.Windows.Forms.ToolStripMenuItem sairToolStripMenuItem;
    }
}
```

2.5. A Classe NetWorkPackageProcessed

```
using System;
using System.Data;
using System.Data.OleDb;
using System.Collections.Generic;
using ObjectLibrary.Communication;

using ObjectLibrary;

namespace PackageAnalyzator
{
        public class NetWorkPackageProcessed
    {

        #region Fields

        private long m_idAnalysisProject;
        private AnalysisProject m_AnalysisProject;
                private DateTime m_DestinationTS;
                private DateTime m_SourceTS;
                private double m_Length;
                private double m_IPID;
                private string m_DestinationPort;
                private string m_SourcePort;
                private string m_DestinationAddress;
                private string m_SourceAddress;
        private double m_ID;
        private double m_Delay;
        private double m_Jitter;
        private bool m_Losted;

                //Estados do objeto
                private bool m_IsHardObject;  //Existe no BD
                private bool m_IsSavedObject; //Atualizado no BD

        #endregion

        #region Constructors

        public NetWorkPackageProcessed()
                {
                        try
                        {
                                clear();
                        }
                        catch (Exception ex)
                        {
                                Exception tmpEx = new Exception("Erro ao carregar objeto",
ex);
                                throw tmpEx;
                        }
                }

                #endregion
```

```
                #region Properties

                //Todos objetos devem conter esta propriedade. Ela diz se o objeto existe
no banco de dados
                internal bool IsHardObject
                {
                        get
                        {
                                return m_IsHardObject;
                        }
                }

                //Todos objetos devem conter esta propriedade. Ela diz se o objeto está
salvo no banco de dados
                internal bool IsSavedObject
                {
                        get
                        {
                                return m_IsSavedObject;
                        }
                }

                public DateTime DestinationTS
                {
                        get
                        {
                                return m_DestinationTS;
                        }
                        set
                        {
                                m_DestinationTS = value;
                                m_IsSavedObject = false;
                        }
                }

                public DateTime SourceTS
                {
                        get
                        {
                                return m_SourceTS;
                        }
                        set
                        {
                                m_SourceTS = value;
                                m_IsSavedObject = false;
                        }
                }

                public double Length
                {
                        get
                        {
                                return m_Length;
                        }
                        set
                        {
                                m_Length = value;
                                m_IsSavedObject = false;
                        }
                }

                public double IPID
                {
                        get
                        {
                                return m_IPID;
                        }
                        set
                        {
                                m_IPID = value;
                                m_IsSavedObject = false;
                        }
                }

                public string DestinationPort
                {
```

```
                get
                {
                        return m_DestinationPort;
                }
                set
                {
                        m_DestinationPort = value;
                        m_IsSavedObject = false;
                }
        }

        public string SourcePort
        {
                get
                {
                        return m_SourcePort;
                }
                set
                {
                        m_SourcePort = value;
                        m_IsSavedObject = false;
                }
        }

        public string DestinationAddress
        {
                get
                {
                        return m_DestinationAddress;
                }
                set
                {
                        m_DestinationAddress = value;
                        m_IsSavedObject = false;
                }
        }

        public string SourceAddress
        {
                get
                {
                        return m_SourceAddress;
                }
                set
                {
                        m_SourceAddress = value;
                        m_IsSavedObject = false;
                }
        }

        public double Delay
        {
                get
                {
                        return m_Delay;
                }
                set
                {
                        m_Delay = value;
                        m_IsSavedObject = false;
                }
        }

    public double Jitter
    {
        get
        {
            return m_Jitter;
        }
        set
        {
            m_Jitter = value;
            m_IsSavedObject = false;
        }
    }
```

```
        public bool Losted
        {
            get
            {
                return m_Losted;
            }
            set
            {
                m_Losted = value;
                m_IsSavedObject = false;
            }
        }

        private long idAnalysisProject
        {
            get
            {
                return m_idAnalysisProject;
            }
            set
            {
                m_idAnalysisProject = value;
                m_IsSavedObject = false;
            }
        }

        public AnalysisProject AnalysisProject
        {
            get
            {
                if (m_AnalysisProject == null)
                    m_AnalysisProject =
AnalysisProject.findAnalysisProject(m_idAnalysisProject);

                return m_AnalysisProject;
            }
            set
            {
                m_AnalysisProject = value;
                m_idAnalysisProject = m_AnalysisProject.ID;
                m_IsSavedObject = false;
            }
        }

        public double ID
        {
            get
            {
                return m_ID;
            }
            set
            {
                m_ID = value;
                m_IsSavedObject = false;
            }
        }

               #endregion

               #region Static Methods

               public static NetWorkPackageProcessed findNetWorkTableProcessed(long ID)
               {
                      try
                      {
                                  NetWorkPackageProcessed obj = new
NetWorkPackageProcessed();

                                  List<NetWorkPackageProcessed> alNetWorkTableProcesseds =
new List<NetWorkPackageProcessed>();

                                  string clausulaWhere="";
                                  clausulaWhere =
                                                "   ID = " +  ID ;
```

```
                                alNetWorkTableProcesseds = findAll(clausulaWhere);

                                if(alNetWorkTableProcesseds.Count > 1)
                                {
                                        Exception tmpEx = new Exception("A procura retornou
mais de um objeto");
                                        throw tmpEx;
                                }
                                if(alNetWorkTableProcesseds.Count == 1)
                                        obj = (NetWorkPackageProcessed)
alNetWorkTableProcesseds[0];
                                return obj;
                        }
                        catch (Exception ex)
                        {
                                Exception tmpEx = new Exception("Erro ao procurar objeto",
ex);
                                throw tmpEx;
                        }
                }

        public static List<NetWorkPackageProcessed> findNetWorkTableProcesseds()
                {
                        try
                        {
                                string clausulaWhere = "";
                  List<NetWorkPackageProcessed> alNetWorkTableProcesseds = new
List<NetWorkPackageProcessed>();

                                alNetWorkTableProcesseds = findAll(clausulaWhere);
                                return alNetWorkTableProcesseds;
                        }
                        catch (Exception ex)
                        {
                                Exception tmpEx = new Exception("Erro ao procurar objeto",
ex);
                                throw tmpEx;
                        }
                }

        public static List<NetWorkPackageProcessed> findNetWorkTableProcesseds(long
idAnalysisProject)
        {
            try
            {
                string clausulaWhere = "";
                List<NetWorkPackageProcessed> alNetWorkTableProcesseds = new
List<NetWorkPackageProcessed>();

                clausulaWhere = " idAnalysisProject = " + idAnalysisProject.ToString();

                alNetWorkTableProcesseds = findAll(clausulaWhere, " SourceTS ");
                return alNetWorkTableProcesseds;
            }
            catch (Exception ex)
            {
                Exception tmpEx = new Exception("Erro ao procurar objeto", ex);
                throw tmpEx;
            }
        }

        public static List<NetWorkPackageProcessed>
findNetWorkTableProcesseds_Discard(long idAnalysisProject)
        {
            try
            {
                string clausulaWhere = "";
                List<NetWorkPackageProcessed> alNetWorkTableProcesseds = new
List<NetWorkPackageProcessed>();
                alNetWorkTableProcesseds =
findNetWorkTableProcesseds(idAnalysisProject);

                List<NetWorkPackageProcessed> colNotDiscard = new
List<NetWorkPackageProcessed>();

                NetWorkPackageProcessed firstPack = alNetWorkTableProcesseds[0];
```

```
                NetWorkPackageProcessed lastPack =
alNetWorkTableProcesseds[alNetWorkTableProcesseds.Count - 1];

                foreach (NetWorkPackageProcessed pack in alNetWorkTableProcesseds)
                {

                    TimeSpan initialTime = pack.SourceTS - firstPack.SourceTS;
                    TimeSpan endTime = lastPack.SourceTS - pack.SourceTS;

                    if (initialTime.Seconds <= 3)
                        continue;

                    if (endTime.Seconds <= 3)
                        continue;

                    colNotDiscard.Add(pack);
                }

                return colNotDiscard;
            }
            catch (Exception ex)
            {
                Exception tmpEx = new Exception("Erro ao procurar objeto", ex);
                throw tmpEx;
            }
        }

        #endregion

               #region Instance Methods

               public virtual void save()
               {
                      try
                      {
                             validateProperties();

                             if (m_IsHardObject == false)
                             {
                                    insert();
                                    m_IsSavedObject = true;
                             }
                             else
                             {
                                    if (m_IsSavedObject == false)
                                    {
                                           update();
                                    }
                             }
                      }
                      catch (Exception ex)
                      {
                             Exception tmpEx = new Exception("Erro ao salvar objeto",
ex);
                             throw tmpEx;
                      }
               }

               public virtual void remove()
               {
                      try
                      {
                             int recordsAffected;
                             string sql="";

                             if (m_IsHardObject == false)
                             {
                                    Exception tmpEx = new Exception("O objeto não
existe no banco de dados!");
                                    throw tmpEx;
                             }

                             sql = " DELETE FROM " +
                                           "  NetWorkTableProcessed" +
                        " WHERE " +
                        "   ID = " + ID +
```

```
                        " AND " +
                        "   idAnalysisProject = " + m_idAnalysisProject;

                                OleDbCommand cmm = new OleDbCommand(sql,
DataBase.Connection);
                                recordsAffected = cmm.ExecuteNonQuery();

                                if (recordsAffected > 1)
                                {
                                        Exception tmpEx = new Exception("A atualização do
objeto afetou mais de um registro no banco de dados.");
                                        throw tmpEx;
                                }

                                if (recordsAffected == 0)
                                {
                                        Exception tmpEx = new Exception("Registro alterado
anteriormente. Consulte novamente para atualizar e repita a operação.");
                                        throw tmpEx;
                                }

                                clear();
                                m_IsHardObject = false;
                                m_IsSavedObject = false;
                        }
                        catch (Exception ex)
                        {
                                Exception tmpEx = new Exception("Objeto não foi criado no
banco de dados, portanto não pode ser excluído fisicamente.", ex);
                                throw tmpEx;
                        }
                }

                #endregion

                #region Private Methods

                private void validateProperties()
                {
                        try
                        {
                 if (m_idAnalysisProject == 0)
                 {
                     Exception tmpEx = new Exception("AnalysisProject não preenchido!");
                     throw tmpEx;
                 }

                                if (m_Length == 0)
                                {
                                        Exception tmpEx = new Exception("Length não
preenchido!");
                                        throw tmpEx;
                                }
                                if (m_IPID == 0)
                                {
                                        Exception tmpEx = new Exception("IPID não
preenchido!");
                                        throw tmpEx;
                                }
                                if (m_DestinationPort == "")
                                {
                                        Exception tmpEx = new Exception("DestinationPort
não preenchido!");
                                        throw tmpEx;
                                }
                                if (m_SourcePort == "")
                                {
                                        Exception tmpEx = new Exception("SourcePort não
preenchido!");
                                        throw tmpEx;
                                }
                                if (m_DestinationAddress == "")
                                {
                                        Exception tmpEx = new Exception("DestinationAddress
não preenchido!");
                                        throw tmpEx;
```

```
                        }
                        if (m_SourceAddress == "")
                        {
                                Exception tmpEx = new Exception("SourceAddress não
preenchido!");
                                throw tmpEx;
                        }
                }
                catch (Exception ex)
                {
                        Exception tmpEx = new Exception("Erro ao validar objeto",
ex);
                        throw tmpEx;
                }
        }

        private void insert()
        {
                try
                {
                        string sql="";
                        long NextID = 0;

                        NextID = getNextID();

                        sql = " INSERT INTO" +
                                "       NetWorkTableProcessed" +
                                " (DestinationTS, SourceTS, Length, IPID,
DestinationPort, SourcePort, " +
                "  DestinationAddress, SourceAddress, Delay, Jitter, Losted,
idAnalysisProject, ID) " +
                                " VALUES ( " +
                " '" + m_DestinationTS.ToString("dd/MM/yyyy hh:mm:ss:fff") + "'," +
                " '" + m_SourceTS.ToString("dd/MM/yyyy hh:mm:ss:fff") + "'," +
                                 m_Length.ToString() + "," +
                                 m_IPID.ToString() + "," +
                                 "'" + m_DestinationPort + "'" + "," +
                                 "'" + m_SourcePort + "'" + "," +
                                 "'" + m_DestinationAddress + "'" + "," +
                                 "'" + m_SourceAddress + "'" + "," +
                m_Delay.ToString() + ", " +
                m_Jitter.ToString() + ", " +
                Convert.ToInt16(m_Losted) + ", " +
                m_idAnalysisProject.ToString() + ", " +
                                 NextID + ")";

            OleDbCommand cmm = new OleDbCommand(sql, DataBase.Connection,
DataBase.Transaction);
                        cmm.ExecuteNonQuery();

                        m_ID = NextID;

                        m_IsHardObject = true;
                        m_IsSavedObject = true;
                }
                catch (Exception ex)
                {
                        Exception tmpEx = new Exception("Objeto não foi criado no
banco de dados", ex);
                        throw tmpEx;
                }
        }

        private long getNextID()
        {
                try
                {
                        string sql="";
                        long NextID = 0;

                        DataTable dt = new DataTable();
             sql = " SELECT MAX(NetWorkTableProcessed.ID) as LastID FROM
NetWorkTableProcessed " +
                " WHERE idAnalysisProject = " + m_idAnalysisProject;
```

```
                OleDbCommand cmm = new OleDbCommand(sql, DataBase.Connection,
DataBase.Transaction);
                                OleDbDataAdapter da = new OleDbDataAdapter(cmm);
                                da.Fill(dt);

                                DataRow dr = dt.Rows[0];
                if (dr["LastID"].ToString() != "")
                                {
                    NextID = Convert.ToUInt32(dr["LastID"]);
                                }

                                NextID ++;
                                return NextID;

                        }
                        catch (Exception ex)
                        {
                                Exception tmpEx = new Exception("Erro ao procurar próximo
id", ex);
                                throw tmpEx;
                        }
                }

                private void update()
                {
                        try
                        {
                                int recordsAffected;
                                string sql="";

                 sql = " UPDATE NetWorkTableProcessed SET" +
                       " DestinationTS = " + " '" + m_DestinationTS.ToString("dd/MM/yyyy
hh:mm:ss:fff") + "'," +
                       " SourceTS = " + " '" + m_SourceTS.ToString("dd/MM/yyyy
hh:mm:ss:fff") + "'," +
                       " Length = " + m_Length.ToString() + ", " +
                       " IPID = " + m_IPID.ToString() + ", " +
                       " DestinationPort = " + "'" + m_DestinationPort + "'" + ", " +
                       " SourcePort = " + "'" + m_SourcePort + "'" + ", " +
                       " DestinationAddress = " + "'" + m_DestinationAddress + "'" + ", "
+
                       " SourceAddress = " + "'" + m_SourceAddress + "'" + ", " +
                       " Delay = " + m_Delay.ToString() + ", " +
                       " Jitter = " + m_Jitter.ToString() + ", " +
                       " Losted = " + Convert.ToInt16(m_Losted) + ", " +
                       " idAnalysisProject = " + m_idAnalysisProject.ToString() + " " +
                       " WHERE " +
                       "   ID = " + ID +
                       " AND " +
                       "   idAnalysisProject = " + m_idAnalysisProject;

                OleDbCommand cmm = new OleDbCommand(sql, DataBase.Connection,
DataBase.Transaction);
                                recordsAffected = cmm.ExecuteNonQuery();

                                if (recordsAffected > 1)
                                {
                                        Exception tmpEx = new Exception("A atualização do
objeto afetou mais de um registro no banco de dados.");
                                        throw tmpEx;
                                }

                                if (recordsAffected == 0)
                                {
                                        Exception tmpEx = new Exception("Registro alterado
anteriormente. Consulte novamente para atualizar e repita a operação.");
                                        throw tmpEx;
                                }

                                m_IsHardObject = true;
                                m_IsSavedObject = true;

                        }
                        catch (Exception ex)
                        {
```

```
                                Exception tmpEx = new Exception("Objeto não foi atualizado
no banco de dados", ex);
                                throw tmpEx;
                        }
                }

        private void load(DateTime DestinationTS, DateTime SourceTS, double Length,
double IPID,
                string DestinationPort, string SourcePort, string DestinationAddress, string
SourceAddress,
                double Delay, double Jitter, bool Losted, long idAnalysisProject, double ID,
                bool isHardObject, bool isSavedObject)
                   {
                        try
                        {
                                m_DestinationTS = DestinationTS;
                                m_SourceTS = SourceTS;
                                m_Length = Length;
                                m_IPID = IPID;
                                m_DestinationPort = DestinationPort;
                                m_SourcePort = SourcePort;
                                m_DestinationAddress = DestinationAddress;
                                m_SourceAddress = SourceAddress;
                    m_idAnalysisProject = idAnalysisProject;
                                m_ID = ID;

                    m_Delay = Delay;
                    m_Jitter = Jitter;
                    m_Losted = Losted;

                                m_IsHardObject = isHardObject;
                                m_IsSavedObject = isSavedObject;
                        }
                        catch (Exception ex)
                        {
                                Exception tmpEx = new Exception("Erro de conversão de tipos
de dados", ex);
                                throw tmpEx;
                        }
                }

                private void clear()
                {
                        try
                        {
                                m_DestinationTS = DateTime.Now;
                                m_SourceTS = DateTime.Now;
                                m_Length = 0;
                                m_IPID = 0;
                                m_DestinationPort = "";
                                m_SourcePort = "";
                                m_DestinationAddress = "";
                                m_SourceAddress = "";
                                m_ID = 0;
                    m_idAnalysisProject = 0;
                    m_Losted = false;
                    m_Jitter = 0;
                    m_Delay = 0;

                                m_IsHardObject = false;
                                m_IsSavedObject = false;
                        }
                        catch (Exception ex)
                        {
                                Exception tmpEx = new Exception("Erro ao limpar objeto",
ex);
                                throw tmpEx;
                        }
                }

        private static List<NetWorkPackageProcessed> findAll(string clausulaWhere)
        {
            try
                        {
                    return findAll(clausulaWhere, "");
            }
```

```
            catch (Exception ex)
            {
                Exception tmpEx = new Exception("Erro ao limpar objeto", ex);
                throw tmpEx;
            }
        }

               private static List<NetWorkPackageProcessed> findAll(string
clausulaWhere, string orderby)
               {
                      try
                      {
                             List<NetWorkPackageProcessed> alNetWorkTableProcesseds =
new List<NetWorkPackageProcessed>();
                             string sql="";
                             DataTable dt = new DataTable();

                             sql = " SELECT " +
                                          "  DestinationTS, SourceTS, Length, IPID,
DestinationPort, SourcePort, " +
                                    "  DestinationAddress, SourceAddress, Delay, Jitter, Losted, ID,
idAnalysisProject " +
                                          " FROM " +
                                          "  NetWorkTableProcessed";

                             if(clausulaWhere != "")
                             {
                                    sql = sql + " WHERE " + clausulaWhere;
                             }

                if (orderby != "")
                {
                    sql += " ORDER BY " + orderby;
                }
                else
                {
                    sql += " ORDER BY ID ";
                }

                             OleDbCommand cmm = new OleDbCommand(sql,
DataBase.Connection);
                             OleDbDataAdapter da = new OleDbDataAdapter(cmm);
                da.AcceptChangesDuringUpdate = true;
                             da.Fill(dt);

                             foreach (DataRow dr in dt.Rows)
                             {

                    bool losted = false;
                    int year = 0;
                    int mounth = 0;
                    int day = 0;
                    int hour = 0;
                    int minutes = 0;
                    int seconds = 0;
                    int miliseconds = 0;

                    if (dr["Losted"] != null)
                        losted = Convert.ToBoolean(dr["Losted"]);

                    year = Convert.ToInt32(dr["DestinationTS"].ToString().Substring(6,
4));
                    mounth = Convert.ToInt32(dr["DestinationTS"].ToString().Substring(3,
2));
                    day = Convert.ToInt32(dr["DestinationTS"].ToString().Substring(0,
2));
                    hour = Convert.ToInt32(dr["DestinationTS"].ToString().Substring(11,
2));
                    minutes =
Convert.ToInt32(dr["DestinationTS"].ToString().Substring(14, 2));
                    seconds =
Convert.ToInt32(dr["DestinationTS"].ToString().Substring(17, 2));
                    miliseconds =
Convert.ToInt32(dr["DestinationTS"].ToString().Substring(20, 3));

                    DateTime destinationTS = new DateTime(year, mounth, day,
```

```
                        hour, minutes, seconds, miliseconds);

                    year = Convert.ToInt32(dr["sourceTS"].ToString().Substring(6, 4));
                    mounth = Convert.ToInt32(dr["sourceTS"].ToString().Substring(3, 2));
                    day = Convert.ToInt32(dr["sourceTS"].ToString().Substring(0, 2));
                    hour = Convert.ToInt32(dr["sourceTS"].ToString().Substring(11, 2));
                    minutes = Convert.ToInt32(dr["sourceTS"].ToString().Substring(14,
2));
                    seconds = Convert.ToInt32(dr["sourceTS"].ToString().Substring(17,
2));
                    miliseconds =
Convert.ToInt32(dr["sourceTS"].ToString().Substring(20, 3));

                    DateTime sourceTS = new DateTime(year, mounth, day,
                        hour, minutes, seconds, miliseconds);

                                        NetWorkPackageProcessed obj = new
NetWorkPackageProcessed();
                                        obj.load(
                                                destinationTS,
                                                sourceTS,
                                                Convert.ToDouble(dr["Length"]),
                        Convert.ToDouble(dr["IPID"]),
                                                Convert.ToString(dr["DestinationPort"]),
                                                Convert.ToString(dr["SourcePort"]),
                                                Convert.ToString(dr["DestinationAddress"]),
                                                Convert.ToString(dr["SourceAddress"]),
                        Convert.ToDouble(dr["Delay"]),
                        Convert.ToDouble(dr["Jitter"]),
                        losted,
                        Convert.ToInt64(dr["idAnalysisProject"]),
                                                Convert.ToDouble(dr["ID"]),
                                                true, true);

                                        alNetWorkTableProcesseds.Add(obj);
                                }
                                return alNetWorkTableProcesseds;
                        }
                        catch (Exception ex)
                        {
                                Exception tmpEx = new Exception("Erro ao procurar objeto",
ex);
                                throw tmpEx;
                        }
                }

                #endregion

        }
}
```

2.6. A Classe NetWorkPackageProcessedControl

```
using System;
using System.Collections.Generic;
using System.Text;

using ObjectLibrary;

namespace PackageAnalyzator
{
    public partial class NetWorkPackageProcessedControl
    {

        public      static      void      calculateJitter(List<NetWorkPackageProcessed>
colNetWorkPackageProcessed)
        {
            try
            {
                NetWorkPackageProcessed previousPackage = null;
```

```
                foreach (NetWorkPackageProcessed nextPackage in
colNetWorkPackageProcessed)
                {
                    if (previousPackage != null)
                    {
                        double jitter = nextPackage.Delay - previousPackage.Delay;
                        nextPackage.Jitter = Math.Abs(jitter);
                        nextPackage.save();
                    }
                    previousPackage = nextPackage;
                }

            }
            catch (Exception ex)
            {
                throw new Exception("", ex);
            }
        }

        public static void calculateAverages(List<NetWorkPackageProcessed>
colNetWorkPackageProcessed,
            TypeOfService typeOfService, Standard standard, AnalysisProject
analysisProject)
        {
            try
            {
                long totalPacks = 0;
                long totalLosted = 0;
                long totalDelay = 0;
                double totalLength = 0;
                double totalJitter = 0;
                DateTime startTime = DateTime.MinValue;
                DateTime endTime = DateTime.MinValue;

                if (colNetWorkPackageProcessed.Count == 0)
                    throw new Exception("Nenhum Pacote Processado!");

                NetWorkPackageProcessed firstPack = colNetWorkPackageProcessed[0];
                NetWorkPackageProcessed lastPack =
colNetWorkPackageProcessed[colNetWorkPackageProcessed.Count -1];

                foreach (NetWorkPackageProcessed pack in colNetWorkPackageProcessed)
                {

                    TimeSpan initialSpan = pack.SourceTS - firstPack.SourceTS;
                    TimeSpan endSpan = lastPack.SourceTS - pack.SourceTS;

                    //if (initialSpan.Seconds <= 3)
                    //    continue;

                    //if (endSpan.Seconds <= 3)
                    //    continue;

                    if (pack.Losted)
                    {
                        totalLosted++;
                    }
                    else
                    {
                        totalDelay += Convert.ToInt64(pack.Delay);
                        totalLength += pack.Length;
                        totalJitter += Math.Abs(pack.Jitter);
                    }

                    if (startTime > pack.SourceTS)
                        startTime = pack.SourceTS;

                    if (endTime < pack.SourceTS)
                        endTime = pack.SourceTS;
                }

                totalPacks = colNetWorkPackageProcessed.Count - totalLosted;

                TimeSpan time = endTime.Subtract(startTime);

                analysisProject.idStandard = standard.ID;
```

```
                analysisProject.idTypeOfService = typeOfService.ID;

                analysisProject.DelayAverage = totalDelay / totalPacks;
                analysisProject.LengthAverage  =  totalLength  /  1000  /  time.Seconds;
//transforma em kb
                analysisProject.JitterAverage = totalJitter / totalPacks;
                analysisProject.LostedAverage = totalLosted / totalPacks;

                analysisProject.saveAnalysis();
            }
            catch (Exception ex)
            {
                throw new Exception("", ex);
            }
        }

    }
}
using System;
using System.Collections.Generic;
using System.Text;

using ObjectLibrary;

namespace PackageAnalyzator
{
    public partial class NetWorkPackageProcessedControl
    {

        public static List<NetWorkPackageProcessed> processNetworkPackage(
            List<NetworkPackage>          colPackageSource,          List<NetworkPackage>
colPackageDestination, AnalysisProject project)
        {
            try
            {
                List<NetWorkPackageProcessed>       colPackageProcessed       =       new
List<NetWorkPackageProcessed>();

                if ((colPackageSource.Count == 0) &&
                    (colPackageDestination.Count == 0))
                    return colPackageProcessed;

                NetworkPackage     endPackage     =     getLastPackage(colPackageSource,
colPackageDestination);
                NetworkPackage    firstPackage    =    getFirstPackage(colPackageSource,
colPackageDestination);

               //foreach (NetworkPackage package in colPackageSource)
                for(int iCount = 0; iCount < colPackageSource.Count; iCount ++)
                {

                    NetworkPackage firstPair = new NetworkPackage();
                    NetworkPackage secondPair = new NetworkPackage();

                    firstPair = colPackageSource[iCount];

                    if(iCount + 1 < colPackageSource.Count)
                        secondPair = colPackageSource[iCount + 1];

                    TimeSpan initialTime = firstPair.TS - firstPackage.TS;
                    TimeSpan endTime = endPackage.TS - firstPair.TS;

                    //if (initialTime.Seconds <= 3)
                    //    continue;

                    //if (endTime.Seconds <= 3)
                    //    continue;

                    firstPair = NetworkPackage.findNetworkPackage(firstPair.ID);
                    secondPair = NetworkPackage.findNetworkPackage(secondPair.ID);

                    if (firstPair.Processed == true)
                        continue;

                    NetWorkPackageProcessed          packProcessed          =          new
NetWorkPackageProcessed();
```

```
                    if ((secondPair.IPID == firstPair.IPID) &&
                        (secondPair.Processed == false))
                    {
                        packProcessed      =      saveNetWorkPackageProcessed(firstPair,
secondPair, project);
                        iCount++;
                    }
                    else
                    {
                        packProcessed   =   saveNetWorkPackageProcessed(firstPair,   null,
project);
                    }

                    colPackageProcessed.Add(packProcessed);
                }

                for (int iCount = 0; iCount < colPackageDestination.Count; iCount++)
                {

                    NetworkPackage firstPair = new NetworkPackage();
                    NetworkPackage secondPair = new NetworkPackage();

                    firstPair = colPackageDestination[iCount];

                    if (iCount + 1 < colPackageDestination.Count)
                        secondPair = colPackageDestination[iCount + 1];

                    TimeSpan initialTime = firstPair.TS - firstPackage.TS;
                    TimeSpan endTime = endPackage.TS - firstPair.TS;

                    //if (initialTime.Seconds <= 3)
                    //    continue;

                    //if (endTime.Seconds <= 3)
                    //    continue;

                    firstPair = NetworkPackage.findNetworkPackage(firstPair.ID);
                    secondPair = NetworkPackage.findNetworkPackage(secondPair.ID);

                    if (firstPair.Processed == true)
                        continue;

                    NetWorkPackageProcessed          packProcessed          =          new
NetWorkPackageProcessed();

                    if ((secondPair.IPID == firstPair.IPID) &&
                        (secondPair.Processed == false))
                    {
                        packProcessed      =      saveNetWorkPackageProcessed(firstPair,
secondPair, project);
                        iCount++;
                    }
                    else
                    {
                        packProcessed   =   saveNetWorkPackageProcessed(firstPair,   null,
project);
                    }

                    colPackageProcessed.Add(packProcessed);
                }

                return colPackageProcessed;

            }
            catch (Exception ex)
            {
                throw new Exception("", ex);
            }
        }

        private      static      NetworkPackage      getLastPackage(List<NetworkPackage>
colPackageSource, List<NetworkPackage> colPackageDestination)
        {
            try
            {
```

```
                NetworkPackage endPackage = null;

                NetworkPackage endPackSource = colPackageSource[0];
                endPackSource                                                   =
NetworkPackage.findLastNetworkPackage(endPackSource.SourceAddress);

                NetworkPackage                    endPackDestination            =
NetworkPackage.findLastNetworkPackage(endPackSource.DestinationAddress);

                TimeSpan spanLast = endPackSource.TS - endPackDestination.TS;
                if (spanLast.Seconds < 0)
                    endPackage = endPackSource;
                else
                    endPackage = endPackDestination;

                //NetworkPackage endPackSource = colPackageSource[colPackageSource.Count
- 1];
                //NetworkPackage                   endPackDestination           =
colPackageDestination[colPackageDestination.Count - 1];

                //TimeSpan spanLast = endPackSource.TS - endPackDestination.TS;
                //if (spanLast.Seconds < 0)
                //    endPackage = endPackSource;
                //else
                //    endPackage = endPackDestination;

                return endPackage;
            }
            catch (Exception ex)
            {
                throw new Exception("", ex);
            }
        }

        private      static      NetworkPackage      getFirstPackage(List<NetworkPackage>
colPackageSource, List<NetworkPackage> colPackageDestination)
        {
            try
            {
                NetworkPackage firstPackage = null;

                NetworkPackage firstPackSource = colPackageSource[0];
                firstPackSource                                                 =
NetworkPackage.findFirstNetworkPackage(firstPackSource.SourceAddress);

                NetworkPackage                   firstPackDestination           =
NetworkPackage.findFirstNetworkPackage(firstPackSource.DestinationAddress);

                TimeSpan spanfirst = firstPackSource.TS - firstPackDestination.TS;
                if (spanfirst.Seconds < 0)
                    firstPackage = firstPackSource;
                else
                    firstPackage = firstPackDestination;

                //NetworkPackage firstPackSource = colPackageSource[0];
                //NetworkPackage firstPackDestination = colPackageDestination[0];

                //TimeSpan spanFirst = firstPackSource.TS - firstPackDestination.TS;
                //if (spanFirst.Seconds < 0)
                //    firstPackage = firstPackSource;
                //else
                //    firstPackage = firstPackDestination;

                return firstPackage;
            }
            catch (Exception ex)
            {
                throw new Exception("", ex);
            }
        }

        private static NetWorkPackageProcessed saveNetWorkPackageProcessed(
            NetworkPackage   firstPair,   NetworkPackage   secondPair,   AnalysisProject
project)
        {
            try
```

```
            {
                NetWorkPackageProcessed pack = new NetWorkPackageProcessed();
                pack.DestinationAddress = firstPair.DestinationAddress;
                pack.DestinationPort = firstPair.DestinationPort;
                pack.IPID = firstPair.IPID;
                pack.Length = firstPair.Length *8;
                pack.SourceAddress = firstPair.SourceAddress;
                pack.SourcePort = firstPair.SourcePort;
                pack.SourceTS = firstPair.TS;
                pack.AnalysisProject = project;

                if (secondPair != null)
                {
                    pack.DestinationTS = secondPair.TS;

                    TimeSpan delay = pack.DestinationTS - pack.SourceTS;
                    pack.Delay = delay.TotalMilliseconds;

                    secondPair.Processed = true;
                    secondPair.save();
                }
                else
                {
                    pack.Losted = true;
                    pack.DestinationTS = DateTime.MinValue;
                }

                firstPair.Processed = true;
                firstPair.save();

                pack.save();

                return pack;
            }
            catch (Exception ex)
            {
                throw new Exception("", ex);
            }
        }

        public   static   List<NetWorkPackageProcessed>   processesOthersPackages(string
sourceAddress, string destinationAddress,
            AnalysisProject project)
        {
            try
            {

                List<NetWorkPackageProcessed>      colPackagesProcessed      =      new
List<NetWorkPackageProcessed>();

                List<NetworkPackage> colSourcesPackages =
                    NetworkPackage.findNetworkPackagesExceptDestinationAddress(
                        sourceAddress, destinationAddress);

                foreach (NetworkPackage sourcePackage in colSourcesPackages)
                {
                    NetWorkPackageProcessed pack = new NetWorkPackageProcessed();
                    pack.DestinationAddress = sourcePackage.DestinationAddress;
                    pack.DestinationPort = sourcePackage.DestinationPort;
                    pack.IPID = sourcePackage.IPID;
                    pack.Length = sourcePackage.Length;
                    pack.SourceAddress = sourcePackage.SourceAddress;
                    pack.SourcePort = sourcePackage.SourcePort;
                    pack.SourceTS = sourcePackage.TS;
                    pack.AnalysisProject = project;
                    pack.save();

                    colPackagesProcessed.Add(pack);
                }

                List<NetworkPackage> colDestinationPackages =
                    NetworkPackage.findNetworkPackagesExceptSourceAddress(
                    sourceAddress, destinationAddress);

                foreach (NetworkPackage DestinationPackage in colDestinationPackages)
                {
```

```
                    NetWorkPackageProcessed pack = new NetWorkPackageProcessed();
                    pack.DestinationAddress = DestinationPackage.DestinationAddress;
                    pack.DestinationPort = DestinationPackage.DestinationPort;
                    pack.IPID = DestinationPackage.IPID;
                    pack.Length = DestinationPackage.Length;
                    pack.SourceAddress = DestinationPackage.SourceAddress;
                    pack.SourcePort = DestinationPackage.SourceAddress;
                    pack.SourceTS = DestinationPackage.TS;
                    pack.AnalysisProject = project;
                    pack.save();

                    colPackagesProcessed.Add(pack);
                }

                return colPackagesProcessed;
            }
            catch (Exception ex)
            {
                throw new Exception("", ex);
            }
        }
    }
}
```

2.7. A Classe PackageDialog

```
using System;
using System.IO;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;

using Tamir.IPLib;
using Tamir.IPLib.Packets;

namespace PackageAnalyzator
{
    public partial class PackageDialog : Form
    {
        Settings _settings;
        private PcapDevice _offLineDevice;

        public PackageDialog()
        {
            InitializeComponent();
            _settings = new Settings();
        }

        private void PackagesDialog_Load(object sender, EventArgs e)
        {
        }

        /// <summary>
        /// Prints the source and dest MAC addresses of each received Ethernet frame
        /// </summary>
        private void offLineDevice_PcapOnPacketArrival(object sender, Packet packet)
        {
            if (packet is EthernetPacket)
            {
                EthernetPacket etherFrame = (EthernetPacket)packet;
                //Console.WriteLine("At: {0}:{1}: MAC:{2} -> MAC:{3}",
                //    etherFrame.PcapHeader.Date.ToString(),
                //    etherFrame.PcapHeader.Date.Millisecond,
                //    etherFrame.SourceHwAddress,
                //    etherFrame.DestinationHwAddress);

                ListViewItem                item                =                new
ListViewItem(etherFrame.PcapHeader.Date.ToString());
                item.Tag = etherFrame;
```

```
                item.SubItems.Add(etherFrame.PcapHeader.Date.Millisecond.ToString());
                item.SubItems.Add(etherFrame.SourceHwAddress);
                item.SubItems.Add(etherFrame.DestinationHwAddress);

                lstPacket.Items.Add(item);
            }
        }

        private void backgroundWorkerReader_DoWork(object sender, DoWorkEventArgs e)
        {
            //Start capture 'INFINTE' number of packets
            //This method will return when EOF reached.
            _offLineDevice.PcapCapture(SharpPcap.INFINITE);
        }

        private     void     backgroundWorkerReader_RunWorkerCompleted(object     sender,
RunWorkerCompletedEventArgs e)
        {
            _offLineDevice.PcapClose();
        }

        private void btnStart_Click(object sender, EventArgs e)
        {
            if (txtLogFile.Text == "")
            {
                MessageBox.Show("Selecione um arquivo de log primeiro!");
                return;
            }
            try
            {
                _offLineDevice = SharpPcap.GetPcapOfflineDevice(txtLogFile.Text);
                _offLineDevice.PcapOpen();
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message);
                return;
            }
            _offLineDevice.PcapOnPacketArrival +=
                new SharpPcap.PacketArrivalEvent(offLineDevice_PcapOnPacketArrival);

            //_offLineDevice.PcapCapture(SharpPcap.INFINITE);
            backgroundWorkerReader.RunWorkerAsync();
        }

        private void btnOpenLogFile_Click(object sender, EventArgs e)
        {
            openFileDialog.Filter = "Log File|*.log";
            openFileDialog.Title = "Selecione um arquivo de log";
            openFileDialog.ShowDialog();

            if (openFileDialog.FileName != "")
            {
                FileStream fs = (FileStream)openFileDialog.OpenFile();
                txtLogFile.Text = fs.Name;
                fs.Close();
            }
        }

    }
}
        namespace PackageAnalyzator
        {
            partial class PackageDialog
            {
                /// <summary>
                /// Required designer variable.
                /// </summary>
                private System.ComponentModel.IContainer components = null;

                /// <summary>
                /// Clean up any resources being used.
                /// </summary>
                /// <param name="disposing">true if managed resources should be disposed;
otherwise, false.</param>
```

```
                protected override void Dispose(bool disposing)
                {
                    if (disposing && (components != null))
                    {
                        components.Dispose();
                    }
                    base.Dispose(disposing);
                }

                #region Windows Form Designer generated code

                /// <summary>
                /// Required method for Designer support - do not modify
                /// the contents of this method with the code editor.
                /// </summary>
                private void InitializeComponent()
                {
                    this.groupBox2 = new System.Windows.Forms.GroupBox();
                    this.lstPacket = new System.Windows.Forms.ListView();
                    this.columnHeader1 = new System.Windows.Forms.ColumnHeader();
                    this.columnHeader2 = new System.Windows.Forms.ColumnHeader();
                    this.columnHeader3 = new System.Windows.Forms.ColumnHeader();
                    this.columnHeader4 = new System.Windows.Forms.ColumnHeader();
                    this.columnHeader5 = new System.Windows.Forms.ColumnHeader();
                    this.backgroundWorkerReader                    =                    new
System.ComponentModel.BackgroundWorker();
                    this.groupBox1 = new System.Windows.Forms.GroupBox();
                    this.btnStart = new System.Windows.Forms.Button();
                    this.btnOpenLogFile = new System.Windows.Forms.Button();
                    this.label1 = new System.Windows.Forms.Label();
                    this.txtLogFile = new System.Windows.Forms.TextBox();
                    this.openFileDialog = new System.Windows.Forms.OpenFileDialog();
                    this.tableLayoutPanel1 = new System.Windows.Forms.TableLayoutPanel();
                    this.groupBox2.SuspendLayout();
                    this.groupBox1.SuspendLayout();
                    this.tableLayoutPanel1.SuspendLayout();
                    this.SuspendLayout();
                    //
                    // groupBox2
                    //
                    this.groupBox2.Controls.Add(this.lstPacket);
                    this.groupBox2.Dock = System.Windows.Forms.DockStyle.Fill;
                    this.groupBox2.Location = new System.Drawing.Point(3, 80);
                    this.groupBox2.Name = "groupBox2";
                    this.groupBox2.Size = new System.Drawing.Size(652, 277);
                    this.groupBox2.TabIndex = 4;
                    this.groupBox2.TabStop = false;
                    this.groupBox2.Text = "Pacotes";
                    //
                    // lstPacket
                    //
                    this.lstPacket.Columns.AddRange(new
System.Windows.Forms.ColumnHeader[] {
                    this.columnHeader1,
                    this.columnHeader2,
                    this.columnHeader3,
                    this.columnHeader4,
                    this.columnHeader5});
                    this.lstPacket.Dock = System.Windows.Forms.DockStyle.Fill;
                    this.lstPacket.Location = new System.Drawing.Point(3, 16);
                    this.lstPacket.Name = "lstPacket";
                    this.lstPacket.Size = new System.Drawing.Size(646, 258);
                    this.lstPacket.TabIndex = 0;
                    this.lstPacket.UseCompatibleStateImageBehavior = false;
                    this.lstPacket.View = System.Windows.Forms.View.Details;
                    //
                    // columnHeader1
                    //
                    this.columnHeader1.Text = "Data";
                    this.columnHeader1.Width = 187;
                    //
                    // columnHeader2
                    //
                    this.columnHeader2.DisplayIndex = 2;
                    this.columnHeader2.Text = "Tamanho";
                    this.columnHeader2.Width = 84;
```

```
                    //
                    // columnHeader3
                    //
                    this.columnHeader3.DisplayIndex = 1;
                    this.columnHeader3.Text = "Hora";
                    this.columnHeader3.Width = 115;
                    //
                    // columnHeader4
                    //
                    this.columnHeader4.Text = "Origem";
                    this.columnHeader4.Width = 139;
                    //
                    // columnHeader5
                    //
                    this.columnHeader5.Text = "Destino";
                    this.columnHeader5.Width = 151;
                    //
                    // backgroundWorkerReader
                    //
                    this.backgroundWorkerReader.DoWork                += new
System.ComponentModel.DoWorkEventHandler(this.backgroundWorkerReader_DoWork);
                    this.backgroundWorkerReader.RunWorkerCompleted          += new
System.ComponentModel.RunWorkerCompletedEventHandler(this.backgroundWorkerReader_RunWork
erCompleted);
                    //
                    // groupBox1
                    //
                    this.groupBox1.Controls.Add(this.btnStart);
                    this.groupBox1.Controls.Add(this.btnOpenLogFile);
                    this.groupBox1.Controls.Add(this.label1);
                    this.groupBox1.Controls.Add(this.txtLogFile);
                    this.groupBox1.Dock = System.Windows.Forms.DockStyle.Fill;
                    this.groupBox1.Location = new System.Drawing.Point(3, 3);
                    this.groupBox1.Name = "groupBox1";
                    this.groupBox1.Size = new System.Drawing.Size(652, 71);
                    this.groupBox1.TabIndex = 5;
                    this.groupBox1.TabStop = false;
                    this.groupBox1.Text = "Configurações";
                    //
                    // btnStart
                    //
                    this.btnStart.FlatStyle = System.Windows.Forms.FlatStyle.Popup;
                    this.btnStart.Location = new System.Drawing.Point(458, 28);
                    this.btnStart.Name = "btnStart";
                    this.btnStart.Size = new System.Drawing.Size(84, 27);
                    this.btnStart.TabIndex = 6;
                    this.btnStart.Text = "Iniciar";
                    this.btnStart.UseVisualStyleBackColor = true;
                    this.btnStart.Click += new System.EventHandler(this.btnStart_Click);
                    //
                    // btnOpenLogFile
                    //
                    this.btnOpenLogFile.FlatStyle = System.Windows.Forms.FlatStyle.Popup;
                    this.btnOpenLogFile.Image                                    =
global::PackageAnalyzator.Properties.Resources.open;
                    this.btnOpenLogFile.Location = new System.Drawing.Point(389, 32);
                    this.btnOpenLogFile.Name = "btnOpenLogFile";
                    this.btnOpenLogFile.Size = new System.Drawing.Size(23, 23);
                    this.btnOpenLogFile.TabIndex = 2;
                    this.btnOpenLogFile.UseVisualStyleBackColor = true;
                    this.btnOpenLogFile.Click                    +=                    new
System.EventHandler(this.btnOpenLogFile_Click);
                    //
                    // label1
                    //
                    this.label1.AutoSize = true;
                    this.label1.Location = new System.Drawing.Point(8, 18);
                    this.label1.Name = "label1";
                    this.label1.Size = new System.Drawing.Size(79, 13);
                    this.label1.TabIndex = 5;
                    this.label1.Text = "Arquivo de Log";
                    //
                    // txtLogFile
                    //
                    this.txtLogFile.Location = new System.Drawing.Point(11, 35);
                    this.txtLogFile.Name = "txtLogFile";
```

```
                    this.txtLogFile.Size = new System.Drawing.Size(372, 20);
                    this.txtLogFile.TabIndex = 4;
                    //
                    // openFileDialog
                    //
                    this.openFileDialog.FileName = "openFileDialog1";
                    //
                    // tableLayoutPanel1
                    //
                    this.tableLayoutPanel1.ColumnCount = 1;
                    this.tableLayoutPanel1.ColumnStyles.Add(new
System.Windows.Forms.ColumnStyle(System.Windows.Forms.SizeType.Percent, 100F));
                    this.tableLayoutPanel1.Controls.Add(this.groupBox1, 0, 0);
                    this.tableLayoutPanel1.Controls.Add(this.groupBox2, 0, 1);
                    this.tableLayoutPanel1.Dock = System.Windows.Forms.DockStyle.Fill;
                    this.tableLayoutPanel1.Location = new System.Drawing.Point(0, 0);
                    this.tableLayoutPanel1.Name = "tableLayoutPanel1";
                    this.tableLayoutPanel1.RowCount = 2;
                    this.tableLayoutPanel1.RowStyles.Add(new
System.Windows.Forms.RowStyle(System.Windows.Forms.SizeType.Absolute, 77F));
                    this.tableLayoutPanel1.RowStyles.Add(new
System.Windows.Forms.RowStyle(System.Windows.Forms.SizeType.Percent, 100F));
                    this.tableLayoutPanel1.Size = new System.Drawing.Size(658, 360);
                    this.tableLayoutPanel1.TabIndex = 6;
                    //
                    // PackagesDialog
                    //
                    this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);
                    this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
                    this.ClientSize = new System.Drawing.Size(658, 360);
                    this.Controls.Add(this.tableLayoutPanel1);
                    this.Name = "PackagesDialog";
                    this.StartPosition                                              =
System.Windows.Forms.FormStartPosition.CenterScreen;
                    this.Text = "Log de Pacotes";
                    this.Load += new System.EventHandler(this.PackagesDialog_Load);
                    this.groupBox2.ResumeLayout(false);
                    this.groupBox1.ResumeLayout(false);
                    this.groupBox1.PerformLayout();
                    this.tableLayoutPanel1.ResumeLayout(false);
                    this.ResumeLayout(false);

                }

                #endregion

                private System.Windows.Forms.GroupBox groupBox2;
                private System.Windows.Forms.ListView lstPacket;
                private System.Windows.Forms.ColumnHeader columnHeader1;
                private System.Windows.Forms.ColumnHeader columnHeader2;
                private System.Windows.Forms.ColumnHeader columnHeader3;
                private System.Windows.Forms.ColumnHeader columnHeader4;
                private System.Windows.Forms.ColumnHeader columnHeader5;
                private System.ComponentModel.BackgroundWorker backgroundWorkerReader;
                private System.Windows.Forms.GroupBox groupBox1;
                private System.Windows.Forms.Button btnOpenLogFile;
                private System.Windows.Forms.Label label1;
                private System.Windows.Forms.TextBox txtLogFile;
                private System.Windows.Forms.Button btnStart;
                private System.Windows.Forms.OpenFileDialog openFileDialog;
                private System.Windows.Forms.TableLayoutPanel tableLayoutPanel1;
            }
        }
```

2.8. A Classe PackageProcessDialog

```
using System;
using System.IO;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Data.OleDb;
```

```
using System.Drawing;
using System.Text;
using System.Windows.Forms;

using Tamir.IPLib;
using Tamir.IPLib.Packets;

using ObjectLibrary.ErrorHandler;
using ObjectLibrary.Communication;
using ObjectLibrary;

namespace PackageAnalyzator
{
    public partial class PackageProcessDialog : Form
    {

        private List<NetworkPackage> _colPackageSource;
        private List<NetworkPackage> _colPackageDestination;

        public PackageProcessDialog()
        {
            try
            {
                InitializeComponent();
            }
            catch (Exception ex)
            {
                throw new Exception("Error during instance object", ex);
            }
        }

        private void LogImporterDialog_Load(object sender, EventArgs e)
        {
            try
            {
                fillCombosAddress();
                fillComboStandard();
                fillComboTypeOfService();
                fillComboAnalysisProject();

                clear();
            }
            catch (Exception ex)
            {
                ErrorHandlerDialog.ShowErroHandler(ref ex);
            }
        }

        private void fillCombosAddress()
        {
            try
            {
                List<string> colSourceAddress = NetworkPackage.findSourceAddresses();
                colSourceAddress.Insert(0, "(Selecione)");
                cboSourceAddress.DataSource = colSourceAddress;
                //cboSourceAddressSubs.DataSource = colSourceAddress;

                List<string>                    colDestinationAddress                    =
NetworkPackage.findDestinationAddresses();
                colDestinationAddress.Insert(0, "(Selecione)");
                cboDestinationAddress.DataSource = colDestinationAddress;
                //cboDestinationAddressSubs.DataSource = colDestinationAddress;
            }
            catch (Exception ex)
            {
                throw new Exception("", ex);
            }
        }

        private void fillComboStandard()
        {
            try
            {
                List<Standard> alStandard =
                    Standard.findStandards();
                Standard tos = new Standard("", "(Selecione)", 0);
```

```
                alStandard.Insert(0, tos);

                cboStandard.DisplayMember = "Name";
                cboStandard.ValueMember = "ID";
                cboStandard.DataSource = alStandard;
            }
            catch (Exception ex)
            {
                throw new Exception("Erro ao listar Tipos de Serviço", ex);
            }
        }

        private void fillComboAnalysisProject()
        {
            try
            {
                List<AnalysisProject> alAnalysisProject =
                    AnalysisProject.findAnalysisProjects();
                AnalysisProject tos = new AnalysisProject();
                tos.Name = "(Selecione)";
                alAnalysisProject.Insert(0, tos);

                cboAnalysisProject.DisplayMember = "Name";
                cboAnalysisProject.ValueMember = "ID";
                cboAnalysisProject.DataSource = alAnalysisProject;
            }
            catch (Exception ex)
            {
                throw new Exception("Erro ao listar Tipos de Serviço", ex);
            }
        }

        private void fillComboTypeOfService()
        {
            try
            {
                List<TypeOfService> alTypeOfService =
                    TypeOfService.findTypeOfServices();
                TypeOfService tos = new TypeOfService("", "(Selecione)", 0);
                alTypeOfService.Insert(0, tos);

                cboTypeOfService.DisplayMember = "Name";
                cboTypeOfService.ValueMember = "ID";
                cboTypeOfService.DataSource = alTypeOfService;
            }
            catch (Exception ex)
            {
                throw new Exception("Erro ao listar Tipos de Serviço", ex);
            }
        }

        private void clear()
        {
            try
            {
                errorProvider.SetError(cboAnalysisProject, "");
                errorProvider.SetError(cboDestinationAddress, "");
                errorProvider.SetError(cboSourceAddress, "");
                errorProvider.SetError(cboStandard, "");
                errorProvider.SetError(cboTypeOfService, "");
                errorProvider.SetError(lstPacket, "");

                lstPacket.Items.Clear();
                txtNewIP.Text = "";
                txtOldIP.Text = "";

                cboStandard.SelectedIndex = 0;
                cboSourceAddress.SelectedIndex = 0;
                cboTypeOfService.SelectedIndex = 0;
                cboAnalysisProject.SelectedIndex = 0;
                cboDestinationAddress.SelectedIndex = 0;
            }
            catch (Exception ex)
            {
                throw new Exception("Erro ao limpar Tipo de Serviço", ex);
            }
```

```
        }

        private void btnStart_Click(object sender, EventArgs e)
        {
            try
            {
                if (fieldsValidated("") == false)
                    return;


                this.Cursor = Cursors.WaitCursor;

                processPackage();

                fillListNetWorkPackage();

                this.Cursor = Cursors.Arrow;

                AnalysisProjectResultDialog frm = new AnalysisProjectResultDialog(
                    (AnalysisProject)cboAnalysisProject.SelectedItem);
                frm.MdiParent = this.MdiParent;
                frm.Show();

            }
            catch (Exception ex)
            {
                this.Cursor = Cursors.Arrow;
                ErrorHandlerDialog.ShowErroHandler(ref ex);
            }
        }

        private bool fieldsValidated(string nameField)
        {
            try
            {
                bool errorControl = true;
                bool result = true;

                if ((nameField == cboTypeOfService.Name) || (nameField == ""))
                {
                    if       (cboTypeOfService.SelectedItem       ==       null       ||
Convert.ToInt32(cboTypeOfService.SelectedValue) == 0)
                    {
                        errorControl = false;
                        errorProvider.SetIconPadding(cboTypeOfService,
(cboTypeOfService.Width / 2 * -1));
                        errorProvider.SetError(cboTypeOfService,      "Selecione      uma
cidade");
                    }
                    else
                        errorProvider.SetError(cboTypeOfService, "");
                }

                if ((nameField == cboStandard.Name) || (nameField == ""))
                {
                    if        (cboStandard.SelectedItem        ==        null        ||
Convert.ToInt32(cboStandard.SelectedValue) == 0)
                    {
                        errorControl = false;
                        errorProvider.SetIconPadding(cboStandard, (cboStandard.Width / 2
* -1));
                        errorProvider.SetError(cboStandard,   "Selecione   um   item   do
combo.");
                    }
                    else
                        errorProvider.SetError(cboStandard, "");
                }

                if ((nameField == cboAnalysisProject.Name) || (nameField == ""))
                {
                    if      (cboAnalysisProject.SelectedItem      ==       null       ||
Convert.ToInt32(cboAnalysisProject.SelectedValue) == 0)
                    {
                        errorControl = false;
                        errorProvider.SetIconPadding(cboAnalysisProject,
(cboAnalysisProject.Width / 2 * -1));
```

```
                        errorProvider.SetError(cboAnalysisProject, "Selecione um item do
combo.");
                    }
                    else
                        errorProvider.SetError(cboAnalysisProject, "");
                }

                if ((nameField == lstPacket.Name) || (nameField == ""))
                {
                    errorProvider.SetError(lstPacket, "");

                    if (lstPacket.Items.Count <= 0)
                    {
                        errorControl = false;
                        errorProvider.SetIconPadding(lstPacket, (lstPacket.Width / 2 * -
1));
                        errorProvider.SetError(lstPacket, "Liste os pacotes primeiro!");
                    }
                }

                return errorControl;
            }
            catch (Exception ex)
            {
                throw new Exception("Erro ao validar os dados do Tipo de Serviço.", ex);
            }
        }

        private void processPackage()
        {
            try
            {
                Standard standard = (Standard)cboStandard.SelectedItem;
                TypeOfService                       typeOfService                       =
(TypeOfService)cboTypeOfService.SelectedItem;
                AnalysisProject                     analysisProject                     =
(AnalysisProject)cboAnalysisProject.SelectedItem;

                List<NetWorkPackageProcessed>     colNetWorkPackageProcessed     =     new
List<NetWorkPackageProcessed>();

                colNetWorkPackageProcessed                                              =
NetWorkPackageProcessedControl.processNetworkPackage(_colPackageSource,
                    _colPackageDestination,
(AnalysisProject)cboAnalysisProject.SelectedItem);


NetWorkPackageProcessedControl.calculateJitter(colNetWorkPackageProcessed);

                NetWorkPackageProcessedControl.calculateAverages(
                    colNetWorkPackageProcessed,         typeOfService,         standard,
analysisProject);

                analysisProject.SourceAddress = cboSourceAddress.Text;
                analysisProject.DestinationAddress = cboDestinationAddress.Text;
                analysisProject.saveAnalysis();

                //DataBase.Transaction.Commit();

            }
            catch (Exception ex)
            {
                //DataBase.Transaction.Rollback();
                throw ex;
            }
        }

        private void processPackageBidirecional()
        {
            try
            {

            }
            catch (Exception ex)
            {
                DataBase.Transaction.Rollback();
```

```
                throw ex;
            }
        }

        private void btnListPackage_Click(object sender, EventArgs e)
        {
            try
            {
                this.Cursor = Cursors.WaitCursor;

                if (fieldsValidatedList("") == false)
                    return;

                fillListNetWorkPackage();

                this.Cursor = Cursors.Arrow;
            }
            catch (Exception ex)
            {
                this.Cursor = Cursors.Arrow;
                ErrorHandlerDialog.ShowErroHandler(ref ex);
            }
        }

        private bool fieldsValidatedList(string nameField)
        {
            try
            {
                bool errorControl = true;

                if ((nameField == cboSourceAddress.Name) || (nameField == ""))
                {
                    if       (cboSourceAddress.SelectedItem       ==       null       ||
cboSourceAddress.SelectedValue.ToString() == "")
                    {
                        errorControl = false;
                        errorProvider.SetIconPadding(cboSourceAddress,
(cboSourceAddress.Width / 2 * -1));
                        errorProvider.SetError(cboSourceAddress,      "Selecione      uma
cidade");
                    }
                    else
                        errorProvider.SetError(cboSourceAddress, "");
                }

                if ((nameField == cboDestinationAddress.Name) || (nameField == ""))
                {
                    if      (cboDestinationAddress.SelectedItem      ==      null      ||
cboDestinationAddress.SelectedValue.ToString() == "")
                    {
                        errorControl = false;
                        errorProvider.SetIconPadding(cboDestinationAddress,
(cboDestinationAddress.Width / 2 * -1));
                        errorProvider.SetError(cboDestinationAddress, "Selecione um item
do combo.");
                    }
                    else
                        errorProvider.SetError(cboDestinationAddress, "");
                }

                return errorControl;
            }
            catch (Exception ex)
            {
                throw new Exception("Erro ao validar os dados do Tipo de Serviço.", ex);
            }
        }

        private void fillListNetWorkPackage()
        {
            try
            {
                _colPackageSource                                                    =
NetworkPackage.findNetworkPackages_OrderByIPID(dtpInitialTS.Value, dtpEndTS.Value,
                    cboSourceAddress.Text, cboDestinationAddress.Text);
```

```
                _colPackageDestination                                                  =
NetworkPackage.findNetworkPackages_OrderByIPID(dtpInitialTS.Value, dtpEndTS.Value,
                    cboDestinationAddress.Text, cboSourceAddress.Text);

                lstPacket.Items.Clear();

                lstPacket.BeginUpdate();
                foreach (NetworkPackage package in _colPackageSource)
                {
                    ListViewItem item = new ListViewItem(package.TS.ToString("dd/MM/yyyy
hh:mm:ss:fff"));
                    item.Tag = package;
                    item.SubItems.Add(package.Length.ToString());
                    item.SubItems.Add(package.SourceAddress);
                    item.SubItems.Add(package.DestinationAddress);
                    item.SubItems.Add(package.IPID.ToString());
                    item.SubItems.Add(package.Processed == true ? "Sim" : "Não" );

                    lstPacket.Items.Add(item);
                }

                foreach (NetworkPackage package in _colPackageDestination)
                {
                    ListViewItem item = new ListViewItem(package.TS.ToString("dd/MM/yyyy
hh:mm:ss:fff"));
                    item.Tag = package;
                    item.SubItems.Add(package.Length.ToString());
                    item.SubItems.Add(package.SourceAddress);
                    item.SubItems.Add(package.DestinationAddress);
                    item.SubItems.Add(package.IPID.ToString());
                    item.SubItems.Add(package.Processed == true ? "Sim" : "Não");

                    lstPacket.Items.Add(item);
                }

                lstPacket.EndUpdate();
            }
            catch (Exception ex)
            {
                throw new Exception("", ex);
            }
        }

        private void btnProcessSubs_Click(object sender, EventArgs e)
        {
            try
            {
                this.Cursor = Cursors.WaitCursor;

                processSub();

                fillCombosAddress();

                MessageBox.Show("Substituição processada com sucesso!");

                this.Cursor = Cursors.Arrow;
            }
            catch (Exception ex)
            {
                this.Cursor = Cursors.Arrow;
                ErrorHandlerDialog.ShowErroHandler(ref ex);
            }
        }

        private void processSub()
        {
            try
            {

                DataBase.Transaction = DataBase.Connection.BeginTransaction();

                NetworkPackage.updateSourceAddress(txtOldIP.Text, txtNewIP.Text);
                NetworkPackage.updateDestinationAddress(txtOldIP.Text, txtNewIP.Text);

                DataBase.Transaction.Commit();
```

```
            }
            catch (Exception ex)
            {
                DataBase.Transaction.Rollback();
                throw new Exception("", ex);
            }
        }

        private void Fields_Validating(object sender, CancelEventArgs e)
        {
            try
            {
                Control c = (Control)sender;
                fieldsValidated(c.Name);
            }
            catch (Exception ex)
            {
                ErrorHandlerDialog.ShowErroHandler(ref ex);
            }
        }

        private void Fields_Enter(object sender, EventArgs e)
        {
            try
            {
                Control c = (Control)sender;
                errorProvider.SetError(c, "");
            }
            catch (Exception ex)
            {
                ErrorHandlerDialog.ShowErroHandler(ref ex);
            }
        }

    }
}
namespace PackageAnalyzator
{
    partial class PackageProcessDialog
    {
        /// <summary>
        /// Required designer variable.
        /// </summary>
        private System.ComponentModel.IContainer components = null;

        /// <summary>
        /// Clean up any resources being used.
        /// </summary>
        ///  <param  name="disposing">true  if  managed  resources  should  be  disposed;
otherwise, false.</param>
        protected override void Dispose(bool disposing)
        {
            if (disposing && (components != null))
            {
                components.Dispose();
            }
            base.Dispose(disposing);
        }

        #region Windows Form Designer generated code

        /// <summary>
        /// Required method for Designer support - do not modify
        /// the contents of this method with the code editor.
        /// </summary>
        private void InitializeComponent()
        {
            this.components = new System.ComponentModel.Container();
            this.tableLayoutPanel1 = new System.Windows.Forms.TableLayoutPanel();
            this.groupBox3 = new System.Windows.Forms.GroupBox();
            this.lblAnalisysProject = new System.Windows.Forms.Label();
            this.lblStandard = new System.Windows.Forms.Label();
            this.cboStandard = new System.Windows.Forms.ComboBox();
            this.cboAnalysisProject = new System.Windows.Forms.ComboBox();
            this.lblTypeOfService = new System.Windows.Forms.Label();
```

```
            this.cboTypeOfService = new System.Windows.Forms.ComboBox();
            this.btnStart = new System.Windows.Forms.Button();
            this.groupBox2 = new System.Windows.Forms.GroupBox();
            this.lstPacket = new System.Windows.Forms.ListView();
            this.columnHeader1 = new System.Windows.Forms.ColumnHeader();
            this.columnHeader2 = new System.Windows.Forms.ColumnHeader();
            this.columnHeader4 = new System.Windows.Forms.ColumnHeader();
            this.columnHeader5 = new System.Windows.Forms.ColumnHeader();
            this.columnHeader3 = new System.Windows.Forms.ColumnHeader();
            this.columnHeader6 = new System.Windows.Forms.ColumnHeader();
            this.groupBox1 = new System.Windows.Forms.GroupBox();
            this.dtpEndTS = new System.Windows.Forms.DateTimePicker();
            this.label2 = new System.Windows.Forms.Label();
            this.label1 = new System.Windows.Forms.Label();
            this.cboDestinationAddress = new System.Windows.Forms.ComboBox();
            this.cboSourceAddress = new System.Windows.Forms.ComboBox();
            this.btnListPackage = new System.Windows.Forms.Button();
            this.dtpInitialTS = new System.Windows.Forms.DateTimePicker();
            this.lblDate = new System.Windows.Forms.Label();
            this.lblInitialHour = new System.Windows.Forms.Label();
            this.gpbSubsIP = new System.Windows.Forms.GroupBox();
            this.txtOldIP = new System.Windows.Forms.TextBox();
            this.btnProcessSubs = new System.Windows.Forms.Button();
            this.txtNewIP = new System.Windows.Forms.TextBox();
            this.label5 = new System.Windows.Forms.Label();
            this.label3 = new System.Windows.Forms.Label();
            this.errorProvider                          =                          new
System.Windows.Forms.ErrorProvider(this.components);
            this.tableLayoutPanel1.SuspendLayout();
            this.groupBox3.SuspendLayout();
            this.groupBox2.SuspendLayout();
            this.groupBox1.SuspendLayout();
            this.gpbSubsIP.SuspendLayout();

((System.ComponentModel.ISupportInitialize)(this.errorProvider)).BeginInit();
            this.SuspendLayout();
            //
            // tableLayoutPanel1
            //
            this.tableLayoutPanel1.ColumnCount = 1;
            this.tableLayoutPanel1.ColumnStyles.Add(new
System.Windows.Forms.ColumnStyle(System.Windows.Forms.SizeType.Percent, 100F));
            this.tableLayoutPanel1.Controls.Add(this.groupBox3, 0, 3);
            this.tableLayoutPanel1.Controls.Add(this.groupBox2, 0, 2);
            this.tableLayoutPanel1.Controls.Add(this.groupBox1, 0, 1);
            this.tableLayoutPanel1.Controls.Add(this.gpbSubsIP, 0, 0);
            this.tableLayoutPanel1.Dock = System.Windows.Forms.DockStyle.Fill;
            this.tableLayoutPanel1.Location = new System.Drawing.Point(0, 0);
            this.tableLayoutPanel1.Name = "tableLayoutPanel1";
            this.tableLayoutPanel1.RowCount = 4;
            this.tableLayoutPanel1.RowStyles.Add(new
System.Windows.Forms.RowStyle(System.Windows.Forms.SizeType.Absolute, 70F));
            this.tableLayoutPanel1.RowStyles.Add(new
System.Windows.Forms.RowStyle(System.Windows.Forms.SizeType.Absolute, 70F));
            this.tableLayoutPanel1.RowStyles.Add(new
System.Windows.Forms.RowStyle(System.Windows.Forms.SizeType.Percent, 100F));
            this.tableLayoutPanel1.RowStyles.Add(new
System.Windows.Forms.RowStyle(System.Windows.Forms.SizeType.Absolute, 70F));
            this.tableLayoutPanel1.RowStyles.Add(new
System.Windows.Forms.RowStyle(System.Windows.Forms.SizeType.Absolute, 20F));
            this.tableLayoutPanel1.Size = new System.Drawing.Size(785, 442);
            this.tableLayoutPanel1.TabIndex = 4;
            //
            // groupBox3
            //
            this.groupBox3.Controls.Add(this.lblAnalisysProject);
            this.groupBox3.Controls.Add(this.lblStandard);
            this.groupBox3.Controls.Add(this.cboStandard);
            this.groupBox3.Controls.Add(this.cboAnalysisProject);
            this.groupBox3.Controls.Add(this.lblTypeOfService);
            this.groupBox3.Controls.Add(this.cboTypeOfService);
            this.groupBox3.Controls.Add(this.btnStart);
            this.groupBox3.Dock = System.Windows.Forms.DockStyle.Fill;
            this.groupBox3.Location = new System.Drawing.Point(3, 375);
            this.groupBox3.Name = "groupBox3";
            this.groupBox3.Size = new System.Drawing.Size(779, 64);
```

```
            this.groupBox3.TabIndex = 3;
            this.groupBox3.TabStop = false;
            this.groupBox3.Text = "Análise";
            //
            // lblAnalisysProject
            //
            this.lblAnalisysProject.AutoSize = true;
            this.lblAnalisysProject.Location = new System.Drawing.Point(9, 18);
            this.lblAnalisysProject.Name = "lblAnalisysProject";
            this.lblAnalisysProject.Size = new System.Drawing.Size(43, 13);
            this.lblAnalisysProject.TabIndex = 0;
            this.lblAnalisysProject.Text = "Projeto:";
            //
            // lblStandard
            //
            this.lblStandard.AutoSize = true;
            this.lblStandard.Location = new System.Drawing.Point(322, 18);
            this.lblStandard.Name = "lblStandard";
            this.lblStandard.Size = new System.Drawing.Size(41, 13);
            this.lblStandard.TabIndex = 4;
            this.lblStandard.Text = "Norma:";
            //
            // cboStandard
            //
            this.cboStandard.DropDownStyle                                          =
System.Windows.Forms.ComboBoxStyle.DropDownList;
            this.cboStandard.FormattingEnabled = true;
            this.cboStandard.Location = new System.Drawing.Point(325, 34);
            this.cboStandard.Name = "cboStandard";
            this.cboStandard.Size = new System.Drawing.Size(140, 21);
            this.cboStandard.TabIndex = 5;
            this.cboStandard.Validating                         +=                      new
System.ComponentModel.CancelEventHandler(this.Fields_Validating);
            this.cboStandard.Enter += new System.EventHandler(this.Fields_Enter);
            //
            // cboAnalysisProject
            //
            this.cboAnalysisProject.DropDownStyle                                   =
System.Windows.Forms.ComboBoxStyle.DropDownList;
            this.cboAnalysisProject.FormattingEnabled = true;
            this.cboAnalysisProject.Location = new System.Drawing.Point(12, 34);
            this.cboAnalysisProject.Name = "cboAnalysisProject";
            this.cboAnalysisProject.Size = new System.Drawing.Size(161, 21);
            this.cboAnalysisProject.TabIndex = 1;
            this.cboAnalysisProject.Validating                    +=                    new
System.ComponentModel.CancelEventHandler(this.Fields_Validating);
            this.cboAnalysisProject.Enter += new System.EventHandler(this.Fields_Enter);
            //
            // lblTypeOfService
            //
            this.lblTypeOfService.AutoSize = true;
            this.lblTypeOfService.Location = new System.Drawing.Point(176, 18);
            this.lblTypeOfService.Name = "lblTypeOfService";
            this.lblTypeOfService.Size = new System.Drawing.Size(85, 13);
            this.lblTypeOfService.TabIndex = 2;
            this.lblTypeOfService.Text = "Tipo de Serviço:";
            //
            // cboTypeOfService
            //
            this.cboTypeOfService.DropDownStyle                                     =
System.Windows.Forms.ComboBoxStyle.DropDownList;
            this.cboTypeOfService.FormattingEnabled = true;
            this.cboTypeOfService.Location = new System.Drawing.Point(179, 34);
            this.cboTypeOfService.Name = "cboTypeOfService";
            this.cboTypeOfService.Size = new System.Drawing.Size(140, 21);
            this.cboTypeOfService.TabIndex = 3;
            this.cboTypeOfService.Validating                      +=                    new
System.ComponentModel.CancelEventHandler(this.Fields_Validating);
            this.cboTypeOfService.Enter += new System.EventHandler(this.Fields_Enter);
            //
            // btnStart
            //
            this.btnStart.FlatStyle = System.Windows.Forms.FlatStyle.Popup;
            this.btnStart.Location = new System.Drawing.Point(627, 30);
            this.btnStart.Name = "btnStart";
            this.btnStart.Size = new System.Drawing.Size(143, 25);
```

```
            this.btnStart.TabIndex = 7;
            this.btnStart.Text = "Processar";
            this.btnStart.UseVisualStyleBackColor = true;
            this.btnStart.Click += new System.EventHandler(this.btnStart_Click);
            //
            // groupBox2
            //
            this.groupBox2.Controls.Add(this.lstPacket);
            this.groupBox2.Dock = System.Windows.Forms.DockStyle.Fill;
            this.groupBox2.Location = new System.Drawing.Point(3, 143);
            this.groupBox2.Name = "groupBox2";
            this.groupBox2.Size = new System.Drawing.Size(779, 226);
            this.groupBox2.TabIndex = 2;
            this.groupBox2.TabStop = false;
            this.groupBox2.Text = "Pacotes";
            //
            // lstPacket
            //
            this.lstPacket.Columns.AddRange(new System.Windows.Forms.ColumnHeader[] {
            this.columnHeader1,
            this.columnHeader2,
            this.columnHeader4,
            this.columnHeader5,
            this.columnHeader3,
            this.columnHeader6});
            this.lstPacket.Dock = System.Windows.Forms.DockStyle.Fill;
            this.lstPacket.FullRowSelect = true;
            this.lstPacket.GridLines = true;
            this.lstPacket.Location = new System.Drawing.Point(3, 16);
            this.lstPacket.Name = "lstPacket";
            this.lstPacket.Size = new System.Drawing.Size(773, 207);
            this.lstPacket.TabIndex = 0;
            this.lstPacket.UseCompatibleStateImageBehavior = false;
            this.lstPacket.View = System.Windows.Forms.View.Details;
            //
            // columnHeader1
            //
            this.columnHeader1.Text = "Data/Hora";
            this.columnHeader1.Width = 147;
            //
            // columnHeader2
            //
            this.columnHeader2.Text = "Tamanho";
            this.columnHeader2.Width = 65;
            //
            // columnHeader4
            //
            this.columnHeader4.Text = "Origem";
            this.columnHeader4.Width = 139;
            //
            // columnHeader5
            //
            this.columnHeader5.Text = "Destino";
            this.columnHeader5.Width = 151;
            //
            // columnHeader3
            //
            this.columnHeader3.Text = "IP ID";
            this.columnHeader3.Width = 100;
            //
            // columnHeader6
            //
            this.columnHeader6.Text = "Processado";
            this.columnHeader6.Width = 80;
            //
            // groupBox1
            //
            this.groupBox1.Controls.Add(this.dtpEndTS);
            this.groupBox1.Controls.Add(this.label2);
            this.groupBox1.Controls.Add(this.label1);
            this.groupBox1.Controls.Add(this.cboDestinationAddress);
            this.groupBox1.Controls.Add(this.cboSourceAddress);
            this.groupBox1.Controls.Add(this.btnListPackage);
            this.groupBox1.Controls.Add(this.dtpInitialTS);
            this.groupBox1.Controls.Add(this.lblDate);
            this.groupBox1.Controls.Add(this.lblInitialHour);
```

```
            this.groupBox1.Dock = System.Windows.Forms.DockStyle.Fill;
            this.groupBox1.Location = new System.Drawing.Point(3, 73);
            this.groupBox1.Name = "groupBox1";
            this.groupBox1.Size = new System.Drawing.Size(779, 64);
            this.groupBox1.TabIndex = 1;
            this.groupBox1.TabStop = false;
            //
            // dtpEndTS
            //
            this.dtpEndTS.CustomFormat = "dd/MM/yyyy hh:mm:ss";
            this.dtpEndTS.Format = System.Windows.Forms.DateTimePickerFormat.Custom;
            this.dtpEndTS.Location = new System.Drawing.Point(442, 34);
            this.dtpEndTS.Name = "dtpEndTS";
            this.dtpEndTS.Size = new System.Drawing.Size(132, 20);
            this.dtpEndTS.TabIndex = 7;
            //
            // label2
            //
            this.label2.AutoSize = true;
            this.label2.Location = new System.Drawing.Point(155, 17);
            this.label2.Name = "label2";
            this.label2.Size = new System.Drawing.Size(46, 13);
            this.label2.TabIndex = 2;
            this.label2.Text = "Destino:";
            //
            // label1
            //
            this.label1.AutoSize = true;
            this.label1.Location = new System.Drawing.Point(9, 17);
            this.label1.Name = "label1";
            this.label1.Size = new System.Drawing.Size(43, 13);
            this.label1.TabIndex = 0;
            this.label1.Text = "Origem:";
            //
            // cboDestinationAddress
            //
            this.cboDestinationAddress.DropDownStyle                                        =
System.Windows.Forms.ComboBoxStyle.DropDownList;
            this.cboDestinationAddress.FormattingEnabled = true;
            this.cboDestinationAddress.Location = new System.Drawing.Point(158, 33);
            this.cboDestinationAddress.Name = "cboDestinationAddress";
            this.cboDestinationAddress.Size = new System.Drawing.Size(140, 21);
            this.cboDestinationAddress.TabIndex = 3;
            this.cboDestinationAddress.Validating                 +=                 new
System.ComponentModel.CancelEventHandler(this.Fields_Validating);
            this.cboDestinationAddress.Enter                    +=                    new
System.EventHandler(this.Fields_Enter);
            //
            // cboSourceAddress
            //
            this.cboSourceAddress.DropDownStyle                                             =
System.Windows.Forms.ComboBoxStyle.DropDownList;
            this.cboSourceAddress.FormattingEnabled = true;
            this.cboSourceAddress.Location = new System.Drawing.Point(12, 33);
            this.cboSourceAddress.Name = "cboSourceAddress";
            this.cboSourceAddress.Size = new System.Drawing.Size(140, 21);
            this.cboSourceAddress.TabIndex = 1;
            this.cboSourceAddress.Validating                  +=                    new
System.ComponentModel.CancelEventHandler(this.Fields_Validating);
            this.cboSourceAddress.Enter += new System.EventHandler(this.Fields_Enter);
            //
            // btnListPackage
            //
            this.btnListPackage.FlatStyle = System.Windows.Forms.FlatStyle.Popup;
            this.btnListPackage.Location = new System.Drawing.Point(627, 31);
            this.btnListPackage.Name = "btnListPackage";
            this.btnListPackage.Size = new System.Drawing.Size(143, 23);
            this.btnListPackage.TabIndex = 8;
            this.btnListPackage.Text = "Listar Pacotes";
            this.btnListPackage.UseVisualStyleBackColor = true;
            this.btnListPackage.Click                       +=                    new
System.EventHandler(this.btnListPackage_Click);
            //
            // dtpInitialTS
            //
            this.dtpInitialTS.CustomFormat = "dd/MM/yyyy hh:mm:ss";
```

```
            this.dtpInitialTS.Format = System.Windows.Forms.DateTimePickerFormat.Custom;
            this.dtpInitialTS.Location = new System.Drawing.Point(304, 34);
            this.dtpInitialTS.Name = "dtpInitialTS";
            this.dtpInitialTS.Size = new System.Drawing.Size(132, 20);
            this.dtpInitialTS.TabIndex = 5;
            //
            // lblDate
            //
            this.lblDate.AutoSize = true;
            this.lblDate.Location = new System.Drawing.Point(301, 18);
            this.lblDate.Name = "lblDate";
            this.lblDate.Size = new System.Drawing.Size(34, 13);
            this.lblDate.TabIndex = 4;
            this.lblDate.Text = "Início";
            //
            // lblInitialHour
            //
            this.lblInitialHour.AutoSize = true;
            this.lblInitialHour.Location = new System.Drawing.Point(439, 18);
            this.lblInitialHour.Name = "lblInitialHour";
            this.lblInitialHour.Size = new System.Drawing.Size(26, 13);
            this.lblInitialHour.TabIndex = 6;
            this.lblInitialHour.Text = "Fim:";
            //
            // gpbSubsIP
            //
            this.gpbSubsIP.Controls.Add(this.txtOldIP);
            this.gpbSubsIP.Controls.Add(this.btnProcessSubs);
            this.gpbSubsIP.Controls.Add(this.txtNewIP);
            this.gpbSubsIP.Controls.Add(this.label5);
            this.gpbSubsIP.Controls.Add(this.label3);
            this.gpbSubsIP.Dock = System.Windows.Forms.DockStyle.Fill;
            this.gpbSubsIP.Location = new System.Drawing.Point(3, 3);
            this.gpbSubsIP.Name = "gpbSubsIP";
            this.gpbSubsIP.Size = new System.Drawing.Size(779, 64);
            this.gpbSubsIP.TabIndex = 0;
            this.gpbSubsIP.TabStop = false;
            this.gpbSubsIP.Text = "Substituir Ips para análise";
            //
            // txtOldIP
            //
            this.txtOldIP.Location = new System.Drawing.Point(12, 30);
            this.txtOldIP.Name = "txtOldIP";
            this.txtOldIP.Size = new System.Drawing.Size(113, 20);
            this.txtOldIP.TabIndex = 1;
            //
            // btnProcessSubs
            //
            this.btnProcessSubs.FlatStyle = System.Windows.Forms.FlatStyle.Popup;
            this.btnProcessSubs.Location = new System.Drawing.Point(279, 26);
            this.btnProcessSubs.Name = "btnProcessSubs";
            this.btnProcessSubs.Size = new System.Drawing.Size(146, 25);
            this.btnProcessSubs.TabIndex = 4;
            this.btnProcessSubs.Text = "Processar Substituição";
            this.btnProcessSubs.UseVisualStyleBackColor = true;
            this.btnProcessSubs.Click                       +=                          new
System.EventHandler(this.btnProcessSubs_Click);
            //
            // txtNewIP
            //
            this.txtNewIP.Location = new System.Drawing.Point(158, 30);
            this.txtNewIP.Name = "txtNewIP";
            this.txtNewIP.Size = new System.Drawing.Size(113, 20);
            this.txtNewIP.TabIndex = 3;
            //
            // label5
            //
            this.label5.AutoSize = true;
            this.label5.Location = new System.Drawing.Point(155, 16);
            this.label5.Name = "label5";
            this.label5.Size = new System.Drawing.Size(59, 13);
            this.label5.TabIndex = 2;
            this.label5.Text = "IP Internet:";
            //
            // label3
            //
```

```
            this.label3.AutoSize = true;
            this.label3.Location = new System.Drawing.Point(9, 16);
            this.label3.Name = "label3";
            this.label3.Size = new System.Drawing.Size(58, 13);
            this.label3.TabIndex = 0;
            this.label3.Text = "IP Original:";
            //
            // errorProvider
            //
            this.errorProvider.BlinkStyle                                                    =
System.Windows.Forms.ErrorBlinkStyle.NeverBlink;
            this.errorProvider.ContainerControl = this;
            //
            // PackageProcessDialog
            //
            this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);
            this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
            this.ClientSize = new System.Drawing.Size(785, 442);
            this.Controls.Add(this.tableLayoutPanel1);
            this.Name = "PackageProcessDialog";
            this.StartPosition = System.Windows.Forms.FormStartPosition.CenterScreen;
            this.Text = "Processador de Pacotes";
            this.Load += new System.EventHandler(this.LogImporterDialog_Load);
            this.tableLayoutPanel1.ResumeLayout(false);
            this.groupBox3.ResumeLayout(false);
            this.groupBox3.PerformLayout();
            this.groupBox2.ResumeLayout(false);
            this.groupBox1.ResumeLayout(false);
            this.groupBox1.PerformLayout();
            this.gpbSubsIP.ResumeLayout(false);
            this.gpbSubsIP.PerformLayout();
            ((System.ComponentModel.ISupportInitialize)(this.errorProvider)).EndInit();
            this.ResumeLayout(false);

        }

        #endregion

        private System.Windows.Forms.TableLayoutPanel tableLayoutPanel1;
        private System.Windows.Forms.GroupBox gpbSubsIP;
        private System.Windows.Forms.Button btnProcessSubs;
        private System.Windows.Forms.TextBox txtNewIP;
        private System.Windows.Forms.Label label5;
        private System.Windows.Forms.Label label3;
        private System.Windows.Forms.GroupBox groupBox3;
        private System.Windows.Forms.Button btnStart;
        private System.Windows.Forms.GroupBox groupBox2;
        private System.Windows.Forms.ListView lstPacket;
        private System.Windows.Forms.ColumnHeader columnHeader1;
        private System.Windows.Forms.ColumnHeader columnHeader2;
        private System.Windows.Forms.ColumnHeader columnHeader4;
        private System.Windows.Forms.ColumnHeader columnHeader5;
        private System.Windows.Forms.ColumnHeader columnHeader3;
        private System.Windows.Forms.ColumnHeader columnHeader6;
        private System.Windows.Forms.GroupBox groupBox1;
        private System.Windows.Forms.Label label2;
        private System.Windows.Forms.Label label1;
        private System.Windows.Forms.ComboBox cboDestinationAddress;
        private System.Windows.Forms.ComboBox cboSourceAddress;
        private System.Windows.Forms.Label lblDate;
        private System.Windows.Forms.DateTimePicker dtpInitialTS;
        private System.Windows.Forms.Label lblInitialHour;
        private System.Windows.Forms.Button btnListPackage;
        private System.Windows.Forms.TextBox txtOldIP;
        private System.Windows.Forms.Label lblTypeOfService;
        private System.Windows.Forms.ComboBox cboTypeOfService;
        private System.Windows.Forms.Label lblAnalisysProject;
        private System.Windows.Forms.ComboBox cboAnalysisProject;
        private System.Windows.Forms.Label lblStandard;
        private System.Windows.Forms.ComboBox cboStandard;
        private System.Windows.Forms.ErrorProvider errorProvider;
        private System.Windows.Forms.DateTimePicker dtpEndTS;

    }
}
```

2.9. A Classe Program

```
using System;
using System.Collections.Generic;
using System.Windows.Forms;

namespace PackageAnalyzator
{
    static class Program
    {
        /// <summary>
        /// The main entry point for the application.
        /// </summary>
        [STAThread]
        static void Main()
        {
            Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);
            Application.Run(new MainForm());
        }
    }
}
```

2.10. A Classe QoSRequisite

```
using System;
using System.Collections.Generic;

namespace PackageAnalyzator
{
    #region Enumerators

    public enum EnumQoSRequisite
    {
        Outflow = 1,
        PackageLoses = 2,
        PackageDelay = 3,
        Jitter = 4
    }

    #endregion

    public class QoSRequisite
        {
                private int m_idQoSRequisite;
                private string m_name;

                #region Constructors

                public QoSRequisite()
                {
                        try
                        {
                 this.load(0);
                        }
                        catch (Exception ex)
                        {
                                Exception tmpEx = new Exception("Erro ao carregar objeto",
ex);
                                throw tmpEx;
                        }
                }

                public QoSRequisite(int idQoSRequisite)
                {
                        try
```

```
                        {
                                this.load(idQoSRequisite);
                        }
                        catch (Exception ex)
                        {
                                Exception tmpEx = new Exception("Erro ao carregar objeto",
ex);
                                throw tmpEx;
                        }
                }

                public QoSRequisite(EnumQoSRequisite value)
                {
                        try
                        {
                                this.load((int)value);
                        }
                        catch (Exception ex)
                        {
                                Exception tmpEx = new Exception("Erro ao carregar objeto",
ex);
                                throw tmpEx;
                        }
                }

                #endregion

                #region Properties

                public string Name
                {
                        get
                        {
                                return m_name;
                        }
                }

                public int ID
                {
                        get
                        {
                                return m_idQoSRequisite;
                        }
                }

                #endregion

                #region Static Methods

                public static List<QoSRequisite> FindQoSRequisites()
                {
                        try
                        {
                         List<QoSRequisite> alQoSRequisites = new List<QoSRequisite>();

                                QoSRequisite obj;

                                obj = new QoSRequisite((int)EnumQoSRequisite.Outflow);
                                alQoSRequisites.Add(obj);

                                obj = new QoSRequisite((int)EnumQoSRequisite.PackageLoses);
                                alQoSRequisites.Add(obj);

                                obj = new QoSRequisite((int)EnumQoSRequisite.PackageDelay);
                                alQoSRequisites.Add(obj);

                                obj = new QoSRequisite((int)EnumQoSRequisite.Jitter);
                                alQoSRequisites.Add(obj);

                                return alQoSRequisites;
                        }
                        catch(Exception ex)
                        {
                                Exception tmpEx = new Exception("Erro ao carregar objeto",
ex);
                                throw tmpEx;
```

```
                        }
                }

                #endregion

                #region Private Methods

                private void load(int IdQoSRequisite)
                {
                        try
                        {
                                switch((EnumQoSRequisite)IdQoSRequisite)
                                {

                                        case EnumQoSRequisite.Outflow:
                                                m_idQoSRequisite                                =
(int)EnumQoSRequisite.Outflow;
                                                m_name = "Largura de Banda";
                                                break;
                                        case EnumQoSRequisite.PackageLoses:
                                                m_idQoSRequisite                                =
(int)EnumQoSRequisite.PackageLoses;
                                                m_name = "Perda de Pacotes";
                                                break;
                                        case EnumQoSRequisite.PackageDelay:
                                                m_idQoSRequisite                                =
(int)EnumQoSRequisite.PackageDelay;
                                                m_name = "Atraso de Pacotes";
                                                break;
                                        case EnumQoSRequisite.Jitter:
                                                m_idQoSRequisite                                =
(int)EnumQoSRequisite.Jitter;
                                                m_name = "Flutuação";
                                                break;
                                        default:
                          m_idQoSRequisite = 0;
                          m_name = "(Selecione)";
                          break;
                                }
                        }
                        catch(Exception ex)
                        {
                                Exception tmpEx = new Exception("Erro ao carregar objeto",
ex);
                                throw tmpEx;
                        }
                }

                #endregion

        }
}
```

2.11. A Classe Settings

```
using System;
using System.Configuration;

namespace PackageAnalyzator
{
    //Application settings wrapper class
    sealed class Settings : ApplicationSettingsBase
    {

        [ApplicationScopedSettingAttribute()]
        public String LogFile
        {
            get { return (String)this["LogFile"]; }
            set { this["LogFile"] = value; }
        }

        [ApplicationScopedSettingAttribute()]
```

```
        public int Device
        {
            get { return (int)(this["Device"]); }
            set { this["Device"] = value; }
        }
    }
}
```

2.12. A Classe Standard

```
using System;
using System.Data;
using System.Data.OleDb;
using System.Collections.Generic;
using ObjectLibrary.Communication;

namespace PackageAnalyzator
{
        public class Standard
        {
                private string m_Description;
                private string m_Name;
                private long m_ID;
                //Estados do objeto
                private bool m_IsHardObject;  //Existe no BD
                private bool m_IsSavedObject; //Atualizado no BD

                #region Constructors

                public Standard()
                {
                        try
                        {
                                clear();
                        }
                        catch (Exception ex)
                        {
                                Exception tmpEx = new Exception("Erro ao carregar objeto",
ex);
                                throw tmpEx;
                        }
                }

                public Standard(string Description, string Name, long ID)
                {
                        try
                        {
                                clear();
                                this.load(Description, Name, ID, false, false);
                        }
                        catch (Exception ex)
                        {
                                Exception tmpEx = new Exception("Erro ao carregar objeto",
ex);
                                throw tmpEx;
                        }
                }

                #endregion

                #region Properties

                //Todos objetos devem conter esta propriedade. Ela diz se o objeto existe
no banco de dados
                internal bool IsHardObject
                {
                        get
                        {
                                return m_IsHardObject;
                        }
                }
```

```
		//Todos objetos devem conter esta propriedade. Ela diz se o objeto está salvo no banco de dados
		internal bool IsSavedObject
		{
			get
			{
				return m_IsSavedObject;
			}
		}

		public string Description
		{
			get
			{
				return m_Description;
			}
			set
			{
				m_Description = value;
				m_IsSavedObject = false;
			}
		}

		public string Name
		{
			get
			{
				return m_Name;
			}
			set
			{
				m_Name = value;
				m_IsSavedObject = false;
			}
		}

		public long ID
		{
			get
			{
				return m_ID;
			}
			set
			{
				m_ID = value;
				m_IsSavedObject = false;
			}
		}

		#endregion

		#region Static Methods

		public static Standard findStandard(long ID)
		{
			try
			{
				Standard obj = new Standard();

				List<Standard> alStandards = new List<Standard>();

				string clausulaWhere="";
				clausulaWhere =
						"   ID = " +  ID ;


				alStandards = findAll(clausulaWhere);

				if(alStandards.Count > 1)
				{
					Exception tmpEx = new Exception("A procura retornou mais de um objeto");
					throw tmpEx;
				}
				if(alStandards.Count == 1)
					obj = (Standard) alStandards[0];
```

```
                        return obj;
                }
                catch (Exception ex)
                {
                        Exception tmpEx = new Exception("Erro ao procurar objeto",
ex);
                        throw tmpEx;
                }
        }

        public static List<Standard> findStandards()
        {
                try
                {
                        string clausulaWhere = "";
                        List<Standard> alStandards = new List<Standard>();

                        alStandards = findAll(clausulaWhere);
                        return alStandards;
                }
                catch (Exception ex)
                {
                        Exception tmpEx = new Exception("Erro ao procurar objeto",
ex);
                        throw tmpEx;
                }
        }

        #endregion

        #region Instance Methods

        protected void loadStandard(long ID)
        {
                try
                {
                        Standard obj = Standard.findStandard(ID);

                        this.load(
                                        obj.Description,
                                        obj.Name,
                                        obj.ID,
                                        obj.IsHardObject,
                                        obj.IsSavedObject);

                }
                catch (Exception ex)
                {
                        Exception tmpEx = new Exception("Erro ao carregar objeto",
ex);
                        throw tmpEx;
                }
        }

        public virtual void save()
        {
                try
                {
                        validateProperties();

                        if (m_IsHardObject == false)
                        {
                                insert();
                                m_IsSavedObject = true;
                        }
                        else
                        {
                                if (m_IsSavedObject == false)
                                {
                                        update();
                                }
                        }
                }
                catch (Exception ex)
                {
```

```
                        Exception tmpEx = new Exception("Erro ao salvar objeto",
ex);
                        throw tmpEx;
                    }
                }

        public virtual void remove()
                {
                    try
                    {
                        int recordsAffected;
                        string sql="";

                        if (m_IsHardObject == false)
                        {
                            Exception tmpEx = new Exception("O objeto não
existe no banco de dados!");
                            throw tmpEx;
                        }

                        sql = " DELETE FROM " +
                                "  Standard" +
                                " WHERE " +
                                "   ID = " +  ID ;

                        OleDbCommand cmm = new OleDbCommand(sql,
DataBase.Connection);
                        recordsAffected = cmm.ExecuteNonQuery();

                        if (recordsAffected > 1)
                        {
                    throw new Exception("A atualização do objeto afetou mais de um
registro no banco de dados.");
                        }

                        if (recordsAffected == 0)
                        {
                    throw new Exception("Registro alterado anteriormente. Consulte
novamente para atualizar e repita a operação.");
                        }

                        clear();
                        m_IsHardObject = false;
                        m_IsSavedObject = false;
                    }
                    catch (Exception ex)
                    {
                        Exception tmpEx = new Exception("Objeto não foi criado no
banco de dados, portanto não pode ser excluído fisicamente.", ex);
                        throw tmpEx;
                    }
                }

                #endregion

                #region Private Methods

                private void validateProperties()
                {
                    try
                    {
                        if (m_Description == "")
                        {
                            Exception tmpEx = new Exception("Description não
preenchido!");
                            throw tmpEx;
                        }
                        if (m_Name == "")
                        {
                            Exception tmpEx = new Exception("Name não
preenchido!");
                            throw tmpEx;
                        }
                    }
                    catch (Exception ex)
                    {
```

```
                        Exception tmpEx = new Exception("Erro ao validar objeto",
ex);
                        throw tmpEx;
                }
        }

        private void insert()
        {
                try
                {
                        string sql="";
                        long NextID = 0;

                        NextID = getNextID();

                        sql = " INSERT INTO" +
                               "       Standard" +
                               " (Description, Name, ID) " +
                               " VALUES ( " +
                                "'" + m_Description + "'" + "," +
                                "'" + m_Name + "'" + "," +
                                 NextID + ")";

                        OleDbCommand      cmm      =      new      OleDbCommand(sql,
DataBase.Connection);
                        cmm.ExecuteNonQuery();

                        m_ID = NextID;

                        m_IsHardObject = true;
                        m_IsSavedObject = true;
                }
                catch (Exception ex)
                {
                        Exception tmpEx = new Exception("Objeto não foi criado no
banco de dados", ex);
                        throw tmpEx;
                }
        }

        private long getNextID()
        {
                try
                {
                        string sql="";
                        long NextID = 0;

                        DataTable dt = new DataTable();
         sql = " SELECT MAX(Standard.ID) as LastStandard" +
                                        " FROM Standard ";

                        OleDbCommand      cmm      =      new      OleDbCommand(sql,
DataBase.Connection);
                        OleDbDataAdapter da = new OleDbDataAdapter(cmm);
                        da.Fill(dt);

                        DataRow dr = dt.Rows[0];
         if (dr["LastStandard"].ToString() != "")
                        {
              NextID = Convert.ToUInt32(dr["LastStandard"]);
                        }

                        NextID ++;
                        return NextID;

                }
                catch (Exception ex)
                {
                        Exception tmpEx = new Exception("Erro ao procurar próximo
id", ex);
                        throw tmpEx;
                }
        }

        private void update()
        {
```

```
                        try
                        {
                                int recordsAffected;
                                string sql="";

                                sql = " UPDATE Standard SET" +
                                        " Description = " + "'" + m_Description + "'" + ",
" +
                                        " Name = " + "'" + m_Name + "'" +
                                       " WHERE " +
                                              "   ID = " +  ID ;

                                OleDbCommand             cmm             =             new
OleDbCommand(sql,DataBase.Connection);
                                recordsAffected = cmm.ExecuteNonQuery();

                                if (recordsAffected > 1)
                                {
                                        Exception  tmpEx  =  new  Exception("A  atualização  do
objeto afetou mais de um registro no banco de dados.");
                                        throw tmpEx;
                                }

                                if (recordsAffected == 0)
                                {
                                        Exception  tmpEx  =  new  Exception("Registro  alterado
anteriormente. Consulte novamente para atualizar e repita a operação.");
                                        throw tmpEx;
                                }

                                m_IsHardObject = true;
                                m_IsSavedObject = true;

                        }
                        catch (Exception ex)
                        {
                                Exception tmpEx = new Exception("Objeto não foi atualizado
no banco de dados", ex);
                                throw tmpEx;
                        }
                }

        private void load(string Description, string Name, long ID, bool isHardObject,
bool isSavedObject)
                {
                        try
                        {
                                m_Description = Description;
                                m_Name = Name;
                                m_ID = ID;
                                m_IsHardObject = isHardObject;
                                m_IsSavedObject = isSavedObject;

                        }
                        catch (Exception ex)
                        {
                                Exception tmpEx = new Exception("Erro de conversão de tipos
de dados", ex);
                                throw tmpEx;
                        }
                }

                private void clear()
                {
                        try
                        {
                                m_Description = "";
                                m_Name = "";
                                m_ID = 0;
                                m_IsHardObject = false;
                                m_IsSavedObject = false;
                        }
                        catch (Exception ex)
                        {
                                Exception  tmpEx  =  new  Exception("Erro  ao  limpar  objeto",
ex);
```

```
                                throw tmpEx;
                        }
                }

                private static List<Standard> findAll(string clausulaWhere)
                {
                        try
                        {
                                List<Standard> alStandards = new List<Standard>();
                                string sql="";
                                DataTable dt = new DataTable();

                                sql = " SELECT " +
                                  "  Description, Name, ID " +
                                                  " FROM " +
                                                  "  Standard";

                                if(clausulaWhere != "")
                                {
                                        sql = sql + " WHERE " + clausulaWhere;
                                }

                                OleDbCommand      cmm      =      new      OleDbCommand(sql,
DataBase.Connection);
                                OleDbDataAdapter da = new OleDbDataAdapter(cmm);
                                da.Fill(dt);

                                foreach (DataRow dr in dt.Rows)
                                {
                                        Standard obj = new Standard();
                                        obj.load(
                                                Convert.ToString(dr["Description"]),
                                                Convert.ToString(dr["Name"]),
                                                Convert.ToUInt32(dr["ID"]),
                                                true, true);

                                        alStandards.Add(obj);
                                }
                                return alStandards;
                        }
                        catch (Exception ex)
                        {
                                Exception tmpEx = new Exception("Erro ao procurar objeto",
ex);
                                throw tmpEx;
                        }
                }

                #endregion

        }
}
```

2.13. A Classe StandardDialog

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;

using ObjectLibrary.ErrorHandler;
using ObjectLibrary.Communication;

namespace PackageAnalyzator
{
    public partial class StandardDialog : Form
    {

        private Standard m_Standard;
```

```
        public StandardDialog()
        {
            try
            {
                InitializeComponent();
            }
            catch (Exception ex)
            {
                throw new Exception("Erro ao instanciar objeto.", ex);
            }
        }

        private void StandardDialog_Load(object sender, EventArgs e)
        {
            try
            {
                clear();
                fillListStandards();
            }
            catch (Exception ex)
            {
                ErrorHandlerDialog.ShowErroHandler(ref ex);
            }
        }

        private void clear()
        {
            try
            {
                m_Standard = null;

                errorProvider.SetError(txtDescription, "");
                errorProvider.SetError(txtName, "");

                txtDescription.Text = "";
                txtName.Text = "";
            }
            catch (Exception ex)
            {
                throw new Exception("Erro ao limpar Norma", ex);
            }
        }

        private void fillListStandards()
        {
            try
            {
                lstStandards.Items.Clear();

                lstStandards.DisplayMember = "Name";

                List<Standard> alStandards = Standard.findStandards();
                foreach (Standard obj in alStandards)
                {
                    lstStandards.Items.Add(obj);
                }
            }
            catch (Exception ex)
            {
                throw new Exception("Erro ao listar Tipos de Serviço", ex);
            }
        }

        private void btnNew_Click(object sender, EventArgs e)
        {
            try
            {
                clear();
            }
            catch (Exception ex)
            {
                ErrorHandlerDialog.ShowErroHandler(ref ex);
            }
        }
```

```
        private void btnRefresh_Click(object sender, EventArgs e)
        {
            try
            {
                fillListStandards();
            }
            catch (Exception ex)
            {
                ErrorHandlerDialog.ShowErroHandler(ref ex);
            }
        }

        private void btnSave_Click(object sender, EventArgs e)
        {
            try
            {
                if (fieldsValidated("") == false)
                    return;

                DialogResult dlgr =
                    MessageBox.Show(this, "Tem certeza que deseja salvar?",
                    "Atenção!",
                    MessageBoxButtons.YesNo,
                    MessageBoxIcon.Question);

                if (dlgr == DialogResult.No)
                    return;

                save();
                MessageBox.Show(this,
                    "Norma salvo com sucesso!",
                    "Analizador",
                    MessageBoxButtons.OK,
                    MessageBoxIcon.Information);

                clear();
                fillListStandards();
            }
            catch (Exception ex)
            {
                ErrorHandlerDialog.ShowErroHandler(ref ex);
            }
        }

        private bool fieldsValidated(string nameField)
        {
            try
            {
                bool errorControl = true;
                bool result = true;

                if ((nameField == txtDescription.Name) || (nameField == ""))
                {
                    if (txtDescription.Text.Trim() == "")
                    {
                        errorControl = false;
                        errorProvider.SetIconPadding(txtDescription,
(txtDescription.Width / 2 * -1));
                        errorProvider.SetError(txtDescription, "Preencha o campo");
                    }
                    else
                        errorProvider.SetError(txtDescription, "");
                }


                if ((nameField == txtName.Name) || (nameField == ""))
                {
                    if (txtName.Text.Trim() == "")
                    {
                        errorControl = false;
                        errorProvider.SetIconPadding(txtName, (txtName.Width / 2 * -1));
                        errorProvider.SetError(txtName, "Preencha o campo");
                    }
                    else
                        errorProvider.SetError(txtName, "");
                }
```

```
            return errorControl;
        }
        catch (Exception ex)
        {
            throw new Exception("Erro ao validar os dados do Norma.", ex);
        }
    }

    private void save()
    {
        try
        {
            if (m_Standard == null)
                m_Standard = new Standard();

            m_Standard.Name = txtName.Text;
            m_Standard.Description = txtDescription.Text;

            //IDbTransaction tran = DataBase.Connection.BeginTransaction();
            try
            {
                m_Standard.save();
                //tran.Commit();
            }
            catch (Exception ex)
            {
                //tran.Rollback();
                throw new Exception("Erro ao salvar Norma", ex);
            }
        }
        catch (Exception ex)
        {
            throw new Exception("Erro ao salvar Norma", ex);
        }
    }

    private void lstStandards_SelectedIndexChanged(object sender, EventArgs e)
    {
        try
        {
            if (lstStandards.SelectedItem != null)
            {
                m_Standard = (Standard)lstStandards.SelectedItem;
                loadStandard();
            }
        }
        catch (Exception ex)
        {
            ErrorHandlerDialog.ShowErroHandler(ref ex);
        }
    }

    private void loadStandard()
    {
        try
        {
            txtName.Text = m_Standard.Name;
            txtDescription.Text = m_Standard.Description;
        }
        catch (Exception ex)
        {
            throw new Exception("Erro ao selecionar Norma", ex);
        }
    }

    private void btnDel_Click(object sender, EventArgs e)
    {
        try
        {
            if (m_Standard == null)
            {
                throw new Exception("Por favor selecione um item primeiro.");
            }

            DialogResult dlgr =
```

```
                    MessageBox.Show(this, "Tem certeza que deseja excluir?",
                    "Atenção!",
                    MessageBoxButtons.YesNo,
                    MessageBoxIcon.Question);
                if (dlgr == DialogResult.No)
                    return;

                delete();

                MessageBox.Show(this,
                    "Norma excluído com sucesso!",
                    "Analizador",
                    MessageBoxButtons.OK,
                    MessageBoxIcon.Information);

                fillListStandards();
                clear();
            }
            catch (Exception ex)
            {
                ErrorHandlerDialog.ShowErroHandler(ref ex);
            }
        }

        private void delete()
        {
            try
            {
                //IDbTransaction tran = DataBase.Connection.BeginTransaction();
                try
                {
                    m_Standard.remove();
                    //tran.Commit();
                }
                catch (Exception ex)
                {
                    //tran.Rollback();
                    throw new Exception("Erro ao deletar Norma", ex);
                }

            }
            catch (Exception ex)
            {
                throw new Exception("Erro ao deletar Norma", ex);
            }
        }

        private void Fields_Validating(object sender, CancelEventArgs e)
        {
            try
            {
                Control c = (Control)sender;
                fieldsValidated(c.Name);
            }
            catch (Exception ex)
            {
                ErrorHandlerDialog.ShowErroHandler(ref ex);
            }
        }

        private void Fields_Enter(object sender, EventArgs e)
        {
            try
            {
                Control c = (Control)sender;
                errorProvider.SetError(c, "");
            }
            catch (Exception ex)
            {
                ErrorHandlerDialog.ShowErroHandler(ref ex);
            }
        }

    }
}
```

```
        namespace PackageAnalyzator
        {
            partial class StandardDialog
            {
                /// <summary>
                /// Required designer variable.
                /// </summary>
                private System.ComponentModel.IContainer components = null;

                /// <summary>
                /// Clean up any resources being used.
                /// </summary>
                /// <param name="disposing">true if managed resources should be disposed;
otherwise, false.</param>
                protected override void Dispose(bool disposing)
                {
                    if (disposing && (components != null))
                    {
                        components.Dispose();
                    }
                    base.Dispose(disposing);
                }

                #region Windows Form Designer generated code

                /// <summary>
                /// Required method for Designer support - do not modify
                /// the contents of this method with the code editor.
                /// </summary>
                private void InitializeComponent()
                {
                    this.components = new System.ComponentModel.Container();
                    this.gpbEdit = new System.Windows.Forms.GroupBox();
                    this.lblDescription = new System.Windows.Forms.Label();
                    this.txtDescription = new System.Windows.Forms.TextBox();
                    this.lblName = new System.Windows.Forms.Label();
                    this.txtName = new System.Windows.Forms.TextBox();
                    this.btnSave = new System.Windows.Forms.Button();
                    this.btnDel = new System.Windows.Forms.Button();
                    this.btnNew = new System.Windows.Forms.Button();
                    this.gpbList = new System.Windows.Forms.GroupBox();
                    this.lstStandards = new System.Windows.Forms.ListBox();
                    this.toolTip = new System.Windows.Forms.ToolTip(this.components);
                    this.errorProvider                    =                    new
System.Windows.Forms.ErrorProvider(this.components);
                    this.gpbEdit.SuspendLayout();
                    this.gpbList.SuspendLayout();

((System.ComponentModel.ISupportInitialize)(this.errorProvider)).BeginInit();
                    this.SuspendLayout();
                    //
                    // gpbEdit
                    //
                    this.gpbEdit.Controls.Add(this.lblDescription);
                    this.gpbEdit.Controls.Add(this.txtDescription);
                    this.gpbEdit.Controls.Add(this.lblName);
                    this.gpbEdit.Controls.Add(this.txtName);
                    this.gpbEdit.Font = new System.Drawing.Font("Microsoft Sans Serif",
8.25F, System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
                    this.gpbEdit.Location = new System.Drawing.Point(4, 199);
                    this.gpbEdit.Name = "gpbEdit";
                    this.gpbEdit.Size = new System.Drawing.Size(336, 135);
                    this.gpbEdit.TabIndex = 1;
                    this.gpbEdit.TabStop = false;
                    this.gpbEdit.Text = "Novo / Editar";
                    //
                    // lblDescription
                    //
                    this.lblDescription.AutoSize = true;
                    this.lblDescription.Font = new System.Drawing.Font("Microsoft Sans
Serif", 8.25F, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point,
((byte)(0)));
                    this.lblDescription.Location = new System.Drawing.Point(8, 66);
                    this.lblDescription.Name = "lblDescription";
                    this.lblDescription.Size = new System.Drawing.Size(58, 13);
                    this.lblDescription.TabIndex = 2;
```

```
                    this.lblDescription.Text = "Descrição:";
                    //
                    // txtDescription
                    //
                    this.txtDescription.Font = new System.Drawing.Font("Microsoft Sans
Serif", 8.25F, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point,
((byte)(0)));
                    this.txtDescription.Location = new System.Drawing.Point(11, 82);
                    this.txtDescription.Multiline = true;
                    this.txtDescription.Name = "txtDescription";
                    this.txtDescription.Size = new System.Drawing.Size(317, 38);
                    this.txtDescription.TabIndex = 3;
                    this.toolTip.SetToolTip(this.txtDescription, "Código do Banco na
compensação");
                    this.txtDescription.Enter += new
System.EventHandler(this.Fields_Enter);
                    this.txtDescription.Validating += new
System.ComponentModel.CancelEventHandler(this.Fields_Validating);
                    //
                    // lblName
                    //
                    this.lblName.AutoSize = true;
                    this.lblName.Font = new System.Drawing.Font("Microsoft Sans Serif",
8.25F, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point,
((byte)(0)));
                    this.lblName.Location = new System.Drawing.Point(8, 19);
                    this.lblName.Name = "lblName";
                    this.lblName.Size = new System.Drawing.Size(38, 13);
                    this.lblName.TabIndex = 0;
                    this.lblName.Text = "Nome:";
                    //
                    // txtName
                    //
                    this.txtName.Font = new System.Drawing.Font("Microsoft Sans Serif",
8.25F, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point,
((byte)(0)));
                    this.txtName.Location = new System.Drawing.Point(9, 35);
                    this.txtName.MaxLength = 255;
                    this.txtName.Name = "txtName";
                    this.txtName.Size = new System.Drawing.Size(319, 20);
                    this.txtName.TabIndex = 1;
                    this.txtName.Enter += new System.EventHandler(this.Fields_Enter);
                    this.txtName.Validating += new
System.ComponentModel.CancelEventHandler(this.Fields_Validating);
                    //
                    // btnSave
                    //
                    this.btnSave.FlatStyle = System.Windows.Forms.FlatStyle.Popup;
                    this.btnSave.Font = new System.Drawing.Font("Microsoft Sans Serif",
8.25F, System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
                    this.btnSave.ForeColor = System.Drawing.Color.MediumBlue;
                    this.btnSave.ImageAlign = System.Drawing.ContentAlignment.MiddleLeft;
                    this.btnSave.Location = new System.Drawing.Point(270, 340);
                    this.btnSave.Name = "btnSave";
                    this.btnSave.Size = new System.Drawing.Size(70, 24);
                    this.btnSave.TabIndex = 4;
                    this.btnSave.Text = "Salvar";
                    this.btnSave.TextAlign = System.Drawing.ContentAlignment.MiddleRight;
                    this.btnSave.Click += new System.EventHandler(this.btnSave_Click);
                    //
                    // btnDel
                    //
                    this.btnDel.FlatStyle = System.Windows.Forms.FlatStyle.Popup;
                    this.btnDel.Font = new System.Drawing.Font("Microsoft Sans Serif",
8.25F, System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
                    this.btnDel.ForeColor = System.Drawing.Color.Red;
                    this.btnDel.ImageAlign = System.Drawing.ContentAlignment.MiddleLeft;
                    this.btnDel.Location = new System.Drawing.Point(194, 340);
                    this.btnDel.Name = "btnDel";
                    this.btnDel.Size = new System.Drawing.Size(70, 24);
                    this.btnDel.TabIndex = 3;
                    this.btnDel.Text = "Excluir";
                    this.btnDel.TextAlign = System.Drawing.ContentAlignment.MiddleRight;
                    this.btnDel.Click += new System.EventHandler(this.btnDel_Click);
                    //
                    // btnNew
```

```
                    //
                    this.btnNew.FlatStyle = System.Windows.Forms.FlatStyle.Popup;
                    this.btnNew.Font = new System.Drawing.Font("Microsoft Sans Serif",
8.25F, System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
                    this.btnNew.ForeColor = System.Drawing.Color.MediumBlue;
                    this.btnNew.ImageAlign = System.Drawing.ContentAlignment.MiddleLeft;
                    this.btnNew.Location = new System.Drawing.Point(119, 340);
                    this.btnNew.Name = "btnNew";
                    this.btnNew.Size = new System.Drawing.Size(69, 24);
                    this.btnNew.TabIndex = 2;
                    this.btnNew.Text = "Novo";
                    this.btnNew.TextAlign = System.Drawing.ContentAlignment.MiddleRight;
                    this.btnNew.Click += new System.EventHandler(this.btnNew_Click);
                    //
                    // gpbList
                    //
                    this.gpbList.Controls.Add(this.lstStandards);
                    this.gpbList.Font = new System.Drawing.Font("Microsoft Sans Serif",
8.25F, System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
                    this.gpbList.Location = new System.Drawing.Point(4, 7);
                    this.gpbList.Name = "gpbList";
                    this.gpbList.Size = new System.Drawing.Size(336, 189);
                    this.gpbList.TabIndex = 0;
                    this.gpbList.TabStop = false;
                    this.gpbList.Text = "Normas Cadastradas:";
                    //
                    // lstStandards
                    //
                    this.lstStandards.Font = new System.Drawing.Font("Microsoft Sans
Serif", 8.25F, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point,
((byte)(0)));
                    this.lstStandards.Location = new System.Drawing.Point(9, 21);
                    this.lstStandards.Name = "lstStandards";
                    this.lstStandards.Size = new System.Drawing.Size(319, 160);
                    this.lstStandards.Sorted = true;
                    this.lstStandards.TabIndex = 0;
                    this.lstStandards.SelectedIndexChanged += new
System.EventHandler(this.lstStandards_SelectedIndexChanged);
                    //
                    // errorProvider
                    //
                    this.errorProvider.ContainerControl = this;
                    //
                    // StandardDialog
                    //
                    this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);
                    this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
                    this.ClientSize = new System.Drawing.Size(345, 372);
                    this.Controls.Add(this.btnSave);
                    this.Controls.Add(this.gpbEdit);
                    this.Controls.Add(this.gpbList);
                    this.Controls.Add(this.btnDel);
                    this.Controls.Add(this.btnNew);
                    this.FormBorderStyle = System.Windows.Forms.FormBorderStyle.Fixed3D;
                    this.MaximizeBox = false;
                    this.MinimizeBox = false;
                    this.Name = "StandardDialog";
                    this.StartPosition =
System.Windows.Forms.FormStartPosition.CenterScreen;
                    this.Text = "Cadastro de Normas";
                    this.Load += new System.EventHandler(this.StandardDialog_Load);
                    this.gpbEdit.ResumeLayout(false);
                    this.gpbEdit.PerformLayout();
                    this.gpbList.ResumeLayout(false);

((System.ComponentModel.ISupportInitialize)(this.errorProvider)).EndInit();
                    this.ResumeLayout(false);

                }

                #endregion

                private System.Windows.Forms.GroupBox gpbEdit;
                private System.Windows.Forms.Button btnSave;
                private System.Windows.Forms.Label lblDescription;
                private System.Windows.Forms.TextBox txtDescription;
```

```
                private System.Windows.Forms.Label lblName;
                private System.Windows.Forms.TextBox txtName;
                private System.Windows.Forms.Button btnDel;
                private System.Windows.Forms.Button btnNew;
                private System.Windows.Forms.GroupBox gpbList;
                private System.Windows.Forms.ListBox lstStandards;
                private System.Windows.Forms.ToolTip toolTip;
                private System.Windows.Forms.ErrorProvider errorProvider;
            }
        }
```

2.14. A Classe StandardTolerance

```
using System;
using System.Data;
using System.Data.OleDb;
using System.Collections.Generic;
using ObjectLibrary.Communication;

namespace PackageAnalyzator
{
        public class StandardTolerance
        {
         private long m_ID;
                private double m_MaxValue;
                private double m_MinValue;
                private long m_idTypeOfService;
         private TypeOfService m_TypeOfService;
         private int m_idQoSRequisite;
         private QoSRequisite m_QoSRequisite;
         private long m_idStandard;
         private Standard m_Standard;

                //Estados do objeto
                private bool m_IsHardObject;  //Existe no BD
                private bool m_IsSavedObject; //Atualizado no BD

                #region Constructors

                public StandardTolerance()
                {
                        try
                        {
                                clear();
                        }
                        catch (Exception ex)
                        {
                  throw new Exception("Erro ao carregar objeto", ex);
                        }
                }

                #endregion

                #region Properties

                //Todos objetos devem conter esta propriedade. Ela diz se o objeto existe
no banco de dados
                internal bool IsHardObject
                {
                        get
                        {
                                return m_IsHardObject;
                        }
                }

                //Todos objetos devem conter esta propriedade. Ela diz se o objeto está
salvo no banco de dados
                internal bool IsSavedObject
                {
                        get
                        {
                                return m_IsSavedObject;
                        }
                }
```

```
        public long ID
        {
            get
            {
                return m_ID;
            }
            set
            {
                m_ID = value;
                m_IsSavedObject = false;
            }
        }

               public double MaxValue
               {
                      get
                      {
                             return m_MaxValue;
                      }
                      set
                      {
                             m_MaxValue = value;
                             m_IsSavedObject = false;
                      }
               }

               public double MinValue
               {
                      get
                      {
                             return m_MinValue;
                      }
                      set
                      {
                             m_MinValue = value;
                             m_IsSavedObject = false;
                      }
               }

        private long idTypeOfService
               {
                      get
                      {
                             return m_idTypeOfService;
                      }
                      set
                      {
                             m_idTypeOfService = value;
                             m_IsSavedObject = false;
                      }
               }

        public TypeOfService TypeOfService
        {
            get
            {
                if (m_TypeOfService == null)
                    m_TypeOfService                                          =
TypeOfService.findTypeOfService(m_idTypeOfService);

                return m_TypeOfService;
            }
            set
            {
                m_TypeOfService = value;
                m_idTypeOfService = m_TypeOfService.ID;
                m_IsSavedObject = false;
            }
        }

        private int idQoSRequisite
        {
            get
            {
                return m_idQoSRequisite;
```

```
            }
            set
            {
                m_idQoSRequisite = value;
                m_IsSavedObject = false;
            }
        }

        public QoSRequisite QoSRequisite
        {
            get
            {
                if (m_QoSRequisite == null)
                    m_QoSRequisite = new QoSRequisite(m_idQoSRequisite);

                return m_QoSRequisite;
            }
            set
            {
                m_QoSRequisite = value;
                m_idQoSRequisite = m_QoSRequisite.ID;
                m_IsSavedObject = false;
            }
        }

        private long idStandard
        {
            get
            {
                return m_idStandard;
            }
            set
            {
                m_idStandard = value;
                m_IsSavedObject = false;
            }
        }

        public Standard Standard
        {
            get
            {
                if (m_Standard == null)
                    m_Standard = Standard.findStandard(m_idStandard);

                return m_Standard;
            }
            set
            {
                m_Standard = value;
                m_idStandard = m_Standard.ID;
                m_IsSavedObject = false;
            }
        }

           #endregion

           #region Static Methods

        public static StandardTolerance findStandardTolerance(
            long idTypeOfService, long idStandard, int idQoSRequisite)
              {
                     try
                     {
                            StandardTolerance obj = new StandardTolerance();

                            List<StandardTolerance>    alStandardTolerances    =    new
List<StandardTolerance>();

                            string clausulaWhere="";
                 clausulaWhere =
                          "   TypeOfService = " + idTypeOfService +
                          " AND QoSRequisite = " + idQoSRequisite +
                          " AND Standard = " + idStandard;

                            alStandardTolerances = findAll(clausulaWhere);
```

```
                                if(alStandardTolerances.Count > 1)
                                {
                        throw new Exception("A procura retornou mais de um objeto");
                                }
                                if(alStandardTolerances.Count == 1)
                                        obj = (StandardTolerance) alStandardTolerances[0];
                                return obj;
                        }
                        catch (Exception ex)
                        {
                 throw new Exception("Erro ao procurar objeto", ex);
                        }
                }

                public static List<StandardTolerance> findStandardTolerances()
                {
                        try
                        {
                                string clausulaWhere = "";
                                List<StandardTolerance>    alStandardTolerances    =    new
List<StandardTolerance>();

                                alStandardTolerances = findAll(clausulaWhere);
                                return alStandardTolerances;
                        }
                        catch (Exception ex)
                        {
                                Exception tmpEx = new Exception("Erro ao procurar objeto",
ex);
                                throw tmpEx;
                        }
                }

                #endregion

                #region Instance Methods

        public virtual void save()
                {
                        try
                        {
                                validateProperties();

                                if (m_IsHardObject == false)
                                {
                                        insert();
                                        m_IsSavedObject = true;
                                }
                                else
                                {
                                        if (m_IsSavedObject == false)
                                        {
                                                update();
                                        }
                                }
                        }
                        catch (Exception ex)
                        {
                                Exception  tmpEx  =  new  Exception("Erro  ao  salvar  objeto",
ex);
                                throw tmpEx;
                        }
                }

        public virtual void remove()
                {
                        try
                        {
                                int recordsAffected;
                                string sql="";

                                if (m_IsHardObject == false)
                                {
                                        Exception  tmpEx  =  new  Exception("O  objeto  não
existe no banco de dados!");
```

```
                                        throw tmpEx;
                                }

                                sql = " DELETE FROM " +
                                                "  StandardTolerance" +
                                                " WHERE " +
                           "   TypeOfService = " + idTypeOfService;

                                OleDbCommand      cmm      =      new      OleDbCommand(sql,
DataBase.Connection);
                                recordsAffected = cmm.ExecuteNonQuery();

                                if (recordsAffected > 1)
                                {
                                        Exception  tmpEx  =  new  Exception("A  atualização  do
objeto afetou mais de um registro no banco de dados.");
                                        throw tmpEx;
                                }

                                if (recordsAffected == 0)
                                {
                                        Exception  tmpEx  =  new  Exception("Registro  alterado
anteriormente. Consulte novamente para atualizar e repita a operação.");
                                        throw tmpEx;
                                }

                                clear();
                                m_IsHardObject = false;
                                m_IsSavedObject = false;
                        }
                        catch (Exception ex)
                        {
                                Exception  tmpEx  =  new  Exception("Objeto  não  foi  criado  no
banco de dados, portanto não pode ser excluído fisicamente.", ex);
                                throw tmpEx;
                        }
                }

                #endregion

                #region Private Methods

                private void validateProperties()
                {
                        try
                        {
                                if (m_idTypeOfService == 0)
                                {
                                        Exception  tmpEx  =  new  Exception("TypeOfService  não
preenchido!");
                                        throw tmpEx;
                                }

                 if (m_idQoSRequisite == 0)
                 {
                     Exception tmpEx = new Exception("QoSRequisite não preenchido!");
                     throw tmpEx;
                 }

                 if (m_idStandard == 0)
                 {
                     Exception tmpEx = new Exception("Standard não preenchido!");
                     throw tmpEx;
                 }
                        }
                        catch (Exception ex)
                        {
                                Exception  tmpEx  =  new  Exception("Erro  ao  validar  objeto",
ex);
                                throw tmpEx;
                        }
                }

                private void insert()
                {
                        try
```

```
                        {
                                string sql="";
                long NextID = 0;

                NextID = getNextID();

                                sql = " INSERT INTO" +
                                      "      StandardTolerance" +
                    " (ID, MaxValue, MinValue, TypeOfService, QoSRequisite, Standard) "
+
                                      " VALUES ( " +
                     NextID.ToString() + "," +
                                       m_MaxValue.ToString() + "," +
                     m_MinValue.ToString() + "," +
                                       m_idTypeOfService.ToString() + "," +
                     m_idQoSRequisite.ToString() + "," +
                     m_idStandard.ToString() + ")";

                                OleDbCommand      cmm      =      new      OleDbCommand(sql,
DataBase.Connection);
                                cmm.ExecuteNonQuery();

                                m_IsHardObject = true;
                                m_IsSavedObject = true;
                        }
                        catch (Exception ex)
                        {
                throw new Exception("Objeto não foi criado no banco de dados", ex);
                        }
              }

        private long getNextID()
        {
            try
            {
                string sql = "";
                long NextID = 0;

                DataTable dt = new DataTable();
                sql = " SELECT MAX(ID) as LastStandard" +
                        " FROM StandardTolerance ";

                OleDbCommand cmm = new OleDbCommand(sql, DataBase.Connection);
                OleDbDataAdapter da = new OleDbDataAdapter(cmm);
                da.Fill(dt);

                DataRow dr = dt.Rows[0];
                if (dr["LastStandard"].ToString() != "")
                {
                    NextID = Convert.ToUInt32(dr["LastStandard"]);
                }

                NextID++;
                return NextID;

            }
            catch (Exception ex)
            {
                throw new Exception("Erro ao procurar próximo id", ex);
            }
        }

              private void update()
              {
                     try
                     {
                                string sql="";

                 sql = " UPDATE StandardTolerance SET" +
                      " MaxValue = " + m_MaxValue.ToString() + ", " +
                      " MinValue = " + m_MinValue.ToString() + ", " +
                      " Standard = " + m_idStandard.ToString() + ", " +
                      " QoSRequisite = " + m_idQoSRequisite.ToString() + ", " +
                      " TypeOfService = " + m_idTypeOfService.ToString() + " " +
                    " WHERE " +
```

```
                        "   ID = " + m_ID;

                            OleDbCommand             cmm             =             new
OleDbCommand(sql,DataBase.Connection);
                            recordsAffected = cmm.ExecuteNonQuery();

                            if (recordsAffected > 1)
                            {
                    throw  new  Exception("A  atualização  do  objeto  afetou  mais  de  um
registro no banco de dados.");
                            }

                            if (recordsAffected == 0)
                            {
                    throw  new  Exception("Registro  alterado  anteriormente.  Consulte
novamente para atualizar e repita a operação.");
                            }

                            m_IsHardObject = true;
                            m_IsSavedObject = true;

                        }
                        catch (Exception ex)
                        {
                 throw new Exception("Objeto não foi atualizado no banco de dados", ex);
                        }
                }

                private void clear()
                {
                        try
                        {
                 m_ID = 0;
                            m_MaxValue = 0;
                            m_MinValue = 0;
                 m_idQoSRequisite = 0;
                            m_idTypeOfService = 0;
                 m_Standard = null;
                 m_QoSRequisite = null;
                 m_TypeOfService = null;

                            m_IsHardObject = false;
                            m_IsSavedObject = false;
                        }
                        catch (Exception ex)
                        {
                 throw new Exception("Erro ao limpar objeto", ex);
                        }
                }

        private   void   load(long   ID,   double   MaxValue,   double   MinValue,   long
idTypeOfService,
            int idQoSRequisite, long idStandard, bool isHardObject, bool isSavedObject)
        {
            try
            {
                m_ID = ID;
                m_MaxValue = MaxValue;
                m_MinValue = MinValue;
                m_idStandard = idStandard;
                m_idTypeOfService = idTypeOfService;
                m_idQoSRequisite = idQoSRequisite;
                m_IsHardObject = isHardObject;
                m_IsSavedObject = isSavedObject;
            }
            catch (Exception ex)
            {
                throw new Exception("Erro de conversão de tipos de dados", ex);
            }
        }

                private static List<StandardTolerance> findAll(string clausulaWhere)
                {
                        try
                        {
```

```
                        List<StandardTolerance>    alStandardTolerances    =    new
List<StandardTolerance>();
                        string sql="";
                        DataTable dt = new DataTable();

                        sql = " SELECT " +
                    "  ID, MaxValue, MinValue, TypeOfService, QoSRequisite, Standard
" +
                                        " FROM " +
                                        "  StandardTolerance";

                        if(clausulaWhere != "")
                        {
                                sql = sql + " WHERE " + clausulaWhere;
                        }

                        OleDbCommand      cmm      =      new      OleDbCommand(sql,
DataBase.Connection);
                        OleDbDataAdapter da = new OleDbDataAdapter(cmm);
                        da.Fill(dt);

                        foreach (DataRow dr in dt.Rows)
                        {
                                StandardTolerance obj = new StandardTolerance();
                                obj.load(
                    Convert.ToInt64(dr["ID"]),
                                        Convert.ToDouble(dr["MaxValue"]),
                                        Convert.ToDouble(dr["MinValue"]),
                                        Convert.ToInt64(dr["TypeOfService"]),
                    Convert.ToInt32(dr["QoSRequisite"]),
                    Convert.ToInt64(dr["Standard"]),
                                        true, true);

                                alStandardTolerances.Add(obj);
                        }
                        return alStandardTolerances;
                    }
                    catch (Exception ex)
                    {
                            throw new Exception("Erro ao procurar objeto", ex);
                    }
                }

                #endregion

        }
}
```

2.15. A Classe StandardToleranceDialog

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;

using ObjectLibrary.ErrorHandler;
using ObjectLibrary.Communication;

namespace PackageAnalyzator
{
    public partial class StandardToleranceDialog : Form
    {

        private StandardTolerance m_StandardTolerance;

        public StandardToleranceDialog()
        {
            try
```

```
            {
                InitializeComponent();
            }
            catch (Exception ex)
            {
                throw new Exception("Erro ao instanciar objeto.", ex);
            }
        }

        private void StandardToleranceDialog_Load(object sender, EventArgs e)
        {
            try
            {
                fillComboStandard();
                fillComboQoSRequisite();
                fillComboTypeOfService();
                fillListStandardTolerances();
                clear();
            }
            catch (Exception ex)
            {
                ErrorHandlerDialog.ShowErroHandler(ref ex);
            }
        }

        private void clear()
        {
            try
            {
                m_StandardTolerance = null;

                errorProvider.SetError(txtMaxValue, "");
                errorProvider.SetError(txtMinValue, "");
                errorProvider.SetError(cboStandard, "");
                errorProvider.SetError(cboQoSRequisite, "");
                errorProvider.SetError(cboTypeOfService, "");

                cboStandard.SelectedIndex = 0;
                cboQoSRequisite.SelectedIndex = 0;
                cboTypeOfService.SelectedIndex = 0;
                txtMaxValue.Text = "";
                txtMinValue.Text = "";
            }
            catch (Exception ex)
            {
                throw new Exception("Erro ao limpar Tipo de Serviço", ex);
            }
        }

        private void fillListStandardTolerances()
        {
            try
            {
                lstStandardTolerances.Items.Clear();

                List<StandardTolerance>              alStandardTolerances              =
StandardTolerance.findStandardTolerances();
                foreach (StandardTolerance obj in alStandardTolerances)
                {
                    ListViewItem item = new ListViewItem(obj.Standard.Name);
                    item.Tag = obj;
                    item.SubItems.Add(obj.TypeOfService.Name);
                    item.SubItems.Add(obj.QoSRequisite.Name);
                    item.SubItems.Add(obj.MinValue.ToString());
                    item.SubItems.Add(obj.MaxValue.ToString());

                    lstStandardTolerances.Items.Add(item);
                }
            }
            catch (Exception ex)
            {
                throw new Exception("Erro ao listar Tipos de Serviço", ex);
            }
        }

        private void fillComboStandard()
```

```
        {
            try
            {
                List<Standard> alStandard =
                    Standard.findStandards();
                Standard tos = new Standard("", "(Selecione)", 0);
                alStandard.Insert(0, tos);

                cboStandard.DisplayMember = "Name";
                cboStandard.ValueMember = "ID";
                cboStandard.DataSource = alStandard;
            }
            catch (Exception ex)
            {
                throw new Exception("Erro ao listar Tipos de Serviço", ex);
            }
        }

        private void fillComboTypeOfService()
        {
            try
            {
                List<TypeOfService> alTypeOfService =
                    TypeOfService.findTypeOfServices();
                TypeOfService tos = new TypeOfService("", "(Selecione)", 0);
                alTypeOfService.Insert(0, tos);

                cboTypeOfService.DisplayMember = "Name";
                cboTypeOfService.ValueMember = "ID";
                cboTypeOfService.DataSource = alTypeOfService;
            }
            catch (Exception ex)
            {
                throw new Exception("Erro ao listar Tipos de Serviço", ex);
            }
        }

        private void fillComboQoSRequisite()
        {
            try
            {
                List<QoSRequisite> alQoSRequisite =
                    QoSRequisite.FindQoSRequisites();

                QoSRequisite qos = new QoSRequisite();
                alQoSRequisite.Insert(0, qos);

                cboQoSRequisite.DisplayMember = "Name";
                cboQoSRequisite.ValueMember = "ID";
                cboQoSRequisite.DataSource = alQoSRequisite;
            }
            catch (Exception ex)
            {
                throw new Exception("Erro ao listar Tipos de Serviço", ex);
            }
        }

        private void btnNew_Click(object sender, EventArgs e)
        {
            try
            {
                clear();
            }
            catch (Exception ex)
            {
                ErrorHandlerDialog.ShowErroHandler(ref ex);
            }
        }

        private void btnRefresh_Click(object sender, EventArgs e)
        {
            try
            {
                fillListStandardTolerances();
            }
            catch (Exception ex)
```

```
            {
                ErrorHandlerDialog.ShowErroHandler(ref ex);
            }
        }

        private void btnSave_Click(object sender, EventArgs e)
        {
            try
            {
                if (fieldsValidated("") == false)
                    return;

                DialogResult dlgr =
                    MessageBox.Show(this, "Tem certeza que deseja salvar?",
                    "Atenção!",
                    MessageBoxButtons.YesNo,
                    MessageBoxIcon.Question);

                if (dlgr == DialogResult.No)
                    return;

                save();
                MessageBox.Show(this,
                    "Tolerancia salva com sucesso!",
                    "Analizador",
                    MessageBoxButtons.OK,
                    MessageBoxIcon.Information);

                clear();
                fillListStandardTolerances();
            }
            catch (Exception ex)
            {
                ErrorHandlerDialog.ShowErroHandler(ref ex);
            }
        }

        private bool fieldsValidated(string nameField)
        {
            try
            {
                bool errorControl = true;
                bool result = true;

                if ((nameField == cboTypeOfService.Name) || (nameField == ""))
                {
                    if       (cboTypeOfService.SelectedItem       ==       null       ||
Convert.ToInt32(cboTypeOfService.SelectedValue) == 0)
                    {
                        errorControl = false;
                        errorProvider.SetIconPadding(cboTypeOfService,
(cboTypeOfService.Width / 2 * -1));
                        errorProvider.SetError(cboTypeOfService,      "Selecione      uma
cidade");
                    }
                    else
                        errorProvider.SetError(cboTypeOfService, "");
                }

                if ((nameField == cboStandard.Name) || (nameField == ""))
                {
                    if        (cboStandard.SelectedItem        ==        null        ||
Convert.ToInt32(cboStandard.SelectedValue) == 0)
                    {
                        errorControl = false;
                        errorProvider.SetIconPadding(cboStandard, (cboStandard.Width / 2
* -1));
                        errorProvider.SetError(cboStandard,   "Selecione   um   item   do
combo.");
                    }
                    else
                        errorProvider.SetError(cboStandard, "");
                }

                if ((nameField == cboQoSRequisite.Name) || (nameField == ""))
                {
```

```
                    if       (cboQoSRequisite.SelectedItem       ==       null       ||
Convert.ToInt32(cboQoSRequisite.SelectedValue) == 0)
                    {
                        errorControl = false;
                        errorProvider.SetIconPadding(cboQoSRequisite,
(cboQoSRequisite.Width / 2 * -1));
                        errorProvider.SetError(cboQoSRequisite,  "Selecione  um  item  do
combo.");
                    }
                    else
                        errorProvider.SetError(cboQoSRequisite, "");
                }

                if ((nameField == txtMaxValue.Name) || (nameField == ""))
                {
                    try
                    {
                        double x = Double.Parse(txtMaxValue.Text);
                        errorProvider.SetError(txtMaxValue, "");
                        if (x < 0)
                        {
                            errorControl = false;
                            errorProvider.SetIconPadding(txtMaxValue,  (txtMaxValue.Width
/ 2 * -1));
                            errorProvider.SetError(txtMaxValue, "Preencha o campo com um
número válido!");
                        }
                    }
                    catch
                    {
                        errorControl = false;
                        errorProvider.SetIconPadding(txtMaxValue, (txtMaxValue.Width / 2
* -1));
                        errorProvider.SetError(txtMaxValue,  "Preencha  o  campo  com  um
número válido!");
                    }
                }

                if ((nameField == txtMinValue.Name) || (nameField == ""))
                {
                    try
                    {
                        double x = Double.Parse(txtMinValue.Text);
                        errorProvider.SetError(txtMinValue, "");
                        if (x < 0)
                        {
                            errorControl = false;
                            errorProvider.SetIconPadding(txtMinValue,  (txtMinValue.Width
/ 2 * -1));
                            errorProvider.SetError(txtMinValue, "Preencha o campo com um
número válido!");
                        }
                    }
                    catch
                    {
                        errorControl = false;
                        errorProvider.SetIconPadding(txtMinValue, (txtMinValue.Width / 2
* -1));
                        errorProvider.SetError(txtMinValue,  "Preencha  o  campo  com  um
número válido!");
                    }
                }

                return errorControl;
            }
            catch (Exception ex)
            {
                throw new Exception("Erro ao validar os dados do Tipo de Serviço.", ex);
            }
        }

        private void save()
        {
            try
            {
                if (m_StandardTolerance == null)
```

```
                    m_StandardTolerance = new StandardTolerance();

                m_StandardTolerance.QoSRequisite                                    =
(QoSRequisite)cboQoSRequisite.SelectedItem;
                m_StandardTolerance.TypeOfService                                   =
(TypeOfService)cboTypeOfService.SelectedItem;
                m_StandardTolerance.Standard = (Standard)cboStandard.SelectedItem;
                m_StandardTolerance.MaxValue = Convert.ToDouble(txtMaxValue.Text);
                m_StandardTolerance.MinValue = Convert.ToDouble(txtMinValue.Text);

                //IDbTransaction tran = DataBase.Connection.BeginTransaction();
                try
                {
                    m_StandardTolerance.save();
                    //tran.Commit();
                }
                catch (Exception ex)
                {
                    //tran.Rollback();
                    throw new Exception("Erro ao salvar Tipo de Serviço", ex);
                }
            }
            catch (Exception ex)
            {
                throw new Exception("Erro ao salvar Tipo de Serviço", ex);
            }
        }

        private void lstStandardTolerances_SelectedIndexChanged(object sender, EventArgs
e)
        {
            try
            {
                if (lstStandardTolerances.SelectedItems.Count != 0)
                {
                    m_StandardTolerance                                             =
(StandardTolerance)lstStandardTolerances.SelectedItems[0].Tag;
                    loadStandardTolerance();
                }
            }
            catch (Exception ex)
            {
                ErrorHandlerDialog.ShowErroHandler(ref ex);
            }
        }

        private void loadStandardTolerance()
        {
            try
            {
                txtMinValue.Text = m_StandardTolerance.MinValue.ToString();
                txtMaxValue.Text = m_StandardTolerance.MaxValue.ToString();
                cboStandard.SelectedIndex                                           =
Convert.ToInt32(m_StandardTolerance.Standard.ID);
                cboQoSRequisite.SelectedIndex                                       =
Convert.ToInt32(m_StandardTolerance.QoSRequisite.ID);
                cboTypeOfService.SelectedIndex                                      =
Convert.ToInt32(m_StandardTolerance.TypeOfService.ID);
            }
            catch (Exception ex)
            {
                throw new Exception("Erro ao selecionar Tipo de Serviço", ex);
            }
        }

        private void btnDel_Click(object sender, EventArgs e)
        {
            try
            {
                if (m_StandardTolerance == null)
                {
                    throw new Exception("Por favor selecione um item primeiro.");
                }

                DialogResult dlgr =
                    MessageBox.Show(this, "Tem certeza que deseja excluir?",
```

```
                    "Atenção!",
                    MessageBoxButtons.YesNo,
                    MessageBoxIcon.Question);
                if (dlgr == DialogResult.No)
                    return;

                delete();

                MessageBox.Show(this,
                    "Tolerancia excluído com sucesso!",
                    "Analizador",
                    MessageBoxButtons.OK,
                    MessageBoxIcon.Information);

                fillListStandardTolerances();
                clear();
            }
            catch (Exception ex)
            {
                ErrorHandlerDialog.ShowErroHandler(ref ex);
            }
        }

        private void delete()
        {
            try
            {
                IDbTransaction tran = DataBase.Connection.BeginTransaction();
                try
                {
                    m_StandardTolerance.remove();
                    tran.Commit();
                }
                catch (Exception ex)
                {
                    tran.Rollback();
                    throw new Exception("Erro ao deletar Tipo de Serviço", ex);
                }

            }
            catch (Exception ex)
            {
                throw new Exception("Erro ao deletar Tipo de Serviço", ex);
            }
        }

        private void Fields_Validating(object sender, CancelEventArgs e)
        {
            try
            {
                Control c = (Control)sender;
                fieldsValidated(c.Name);
            }
            catch (Exception ex)
            {
                ErrorHandlerDialog.ShowErroHandler(ref ex);
            }
        }

        private void Fields_Enter(object sender, EventArgs e)
        {
            try
            {
                Control c = (Control)sender;
                errorProvider.SetError(c, "");
            }
            catch (Exception ex)
            {
                ErrorHandlerDialog.ShowErroHandler(ref ex);
            }
        }

    }
}
        namespace PackageAnalyzator
```

```
        {
            partial class StandardToleranceDialog
            {
                /// <summary>
                /// Required designer variable.
                /// </summary>
                private System.ComponentModel.IContainer components = null;

                /// <summary>
                /// Clean up any resources being used.
                /// </summary>
                /// <param name="disposing">true if managed resources should be disposed;
otherwise, false.</param>
                protected override void Dispose(bool disposing)
                {
                    if (disposing && (components != null))
                    {
                        components.Dispose();
                    }
                    base.Dispose(disposing);
                }

                #region Windows Form Designer generated code

                /// <summary>
                /// Required method for Designer support - do not modify
                /// the contents of this method with the code editor.
                /// </summary>
                private void InitializeComponent()
                {
                    this.components = new System.ComponentModel.Container();
                    this.gpbEdit = new System.Windows.Forms.GroupBox();
                    this.lblStandard = new System.Windows.Forms.Label();
                    this.cboStandard = new System.Windows.Forms.ComboBox();
                    this.lblQoSRequisite = new System.Windows.Forms.Label();
                    this.cboQoSRequisite = new System.Windows.Forms.ComboBox();
                    this.lblTypeOfService = new System.Windows.Forms.Label();
                    this.cboTypeOfService = new System.Windows.Forms.ComboBox();
                    this.lblMaxValue = new System.Windows.Forms.Label();
                    this.txtMaxValue = new System.Windows.Forms.TextBox();
                    this.lblMinValue = new System.Windows.Forms.Label();
                    this.txtMinValue = new System.Windows.Forms.TextBox();
                    this.btnSave = new System.Windows.Forms.Button();
                    this.btnDel = new System.Windows.Forms.Button();
                    this.btnNew = new System.Windows.Forms.Button();
                    this.gpbList = new System.Windows.Forms.GroupBox();
                    this.lstStandardTolerances = new System.Windows.Forms.ListView();
                    this.columnHeader1 = new System.Windows.Forms.ColumnHeader();
                    this.columnHeader3 = new System.Windows.Forms.ColumnHeader();
                    this.columnHeader5 = new System.Windows.Forms.ColumnHeader();
                    this.columnHeader2 = new System.Windows.Forms.ColumnHeader();
                    this.columnHeader4 = new System.Windows.Forms.ColumnHeader();
                    this.toolTip = new System.Windows.Forms.ToolTip(this.components);
                    this.errorProvider                          =                          new
System.Windows.Forms.ErrorProvider(this.components);
                    this.gpbEdit.SuspendLayout();
                    this.gpbList.SuspendLayout();

((System.ComponentModel.ISupportInitialize)(this.errorProvider)).BeginInit();
                    this.SuspendLayout();
                    //
                    // gpbEdit
                    //
                    this.gpbEdit.Controls.Add(this.lblStandard);
                    this.gpbEdit.Controls.Add(this.lblQoSRequisite);
                    this.gpbEdit.Controls.Add(this.cboQoSRequisite);
                    this.gpbEdit.Controls.Add(this.cboStandard);
                    this.gpbEdit.Controls.Add(this.lblTypeOfService);
                    this.gpbEdit.Controls.Add(this.cboTypeOfService);
                    this.gpbEdit.Controls.Add(this.lblMaxValue);
                    this.gpbEdit.Controls.Add(this.txtMaxValue);
                    this.gpbEdit.Controls.Add(this.lblMinValue);
                    this.gpbEdit.Controls.Add(this.txtMinValue);
                    this.gpbEdit.Font = new System.Drawing.Font("Microsoft Sans Serif",
8.25F, System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
                    this.gpbEdit.Location = new System.Drawing.Point(4, 242);
```

```
                    this.gpbEdit.Name = "gpbEdit";
                    this.gpbEdit.Size = new System.Drawing.Size(596, 68);
                    this.gpbEdit.TabIndex = 1;
                    this.gpbEdit.TabStop = false;
                    this.gpbEdit.Text = "Novo / Editar";
                    //
                    // lblStandard
                    //
                    this.lblStandard.AutoSize = true;
                    this.lblStandard.Font = new System.Drawing.Font("Microsoft Sans
Serif", 8.25F, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point,
((byte)(0)));
                    this.lblStandard.Location = new System.Drawing.Point(8, 18);
                    this.lblStandard.Name = "lblStandard";
                    this.lblStandard.Size = new System.Drawing.Size(41, 13);
                    this.lblStandard.TabIndex = 0;
                    this.lblStandard.Text = "Norma:";
                    //
                    // cboStandard
                    //
                    this.cboStandard.DropDownStyle =
System.Windows.Forms.ComboBoxStyle.DropDownList;
                    this.cboStandard.Font = new System.Drawing.Font("Microsoft Sans
Serif", 8.25F, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point,
((byte)(0)));
                    this.cboStandard.FormattingEnabled = true;
                    this.cboStandard.Location = new System.Drawing.Point(9, 34);
                    this.cboStandard.Name = "cboStandard";
                    this.cboStandard.Size = new System.Drawing.Size(121, 21);
                    this.cboStandard.TabIndex = 1;
                    this.cboStandard.Validating += new
System.ComponentModel.CancelEventHandler(this.Fields_Validating);
                    this.cboStandard.Enter += new System.EventHandler(this.Fields_Enter);
                    //
                    // lblQoSRequisite
                    //
                    this.lblQoSRequisite.AutoSize = true;
                    this.lblQoSRequisite.Font = new System.Drawing.Font("Microsoft Sans
Serif", 8.25F, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point,
((byte)(0)));
                    this.lblQoSRequisite.Location = new System.Drawing.Point(262, 19);
                    this.lblQoSRequisite.Name = "lblQoSRequisite";
                    this.lblQoSRequisite.Size = new System.Drawing.Size(93, 13);
                    this.lblQoSRequisite.TabIndex = 4;
                    this.lblQoSRequisite.Text = "Requisito de QoS:";
                    //
                    // cboQoSRequisite
                    //
                    this.cboQoSRequisite.DropDownStyle =
System.Windows.Forms.ComboBoxStyle.DropDownList;
                    this.cboQoSRequisite.Font = new System.Drawing.Font("Microsoft Sans
Serif", 8.25F, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point,
((byte)(0)));
                    this.cboQoSRequisite.FormattingEnabled = true;
                    this.cboQoSRequisite.Location = new System.Drawing.Point(263, 35);
                    this.cboQoSRequisite.Name = "cboQoSRequisite";
                    this.cboQoSRequisite.Size = new System.Drawing.Size(121, 21);
                    this.cboQoSRequisite.TabIndex = 5;
                    this.cboQoSRequisite.Validating += new
System.ComponentModel.CancelEventHandler(this.Fields_Validating);
                    this.cboQoSRequisite.Enter += new
System.EventHandler(this.Fields_Enter);
                    //
                    // lblTypeOfService
                    //
                    this.lblTypeOfService.AutoSize = true;
                    this.lblTypeOfService.Font = new System.Drawing.Font("Microsoft Sans
Serif", 8.25F, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point,
((byte)(0)));
                    this.lblTypeOfService.Location = new System.Drawing.Point(135, 19);
                    this.lblTypeOfService.Name = "lblTypeOfService";
                    this.lblTypeOfService.Size = new System.Drawing.Size(85, 13);
                    this.lblTypeOfService.TabIndex = 2;
                    this.lblTypeOfService.Text = "Tipo de Serviço:";
                    //
                    // cboTypeOfService
```

```
                    //
                    this.cboTypeOfService.DropDownStyle                                    =
System.Windows.Forms.ComboBoxStyle.DropDownList;
                    this.cboTypeOfService.Font = new System.Drawing.Font("Microsoft Sans
Serif",   8.25F,   System.Drawing.FontStyle.Regular,   System.Drawing.GraphicsUnit.Point,
((byte)(0)));
                    this.cboTypeOfService.FormattingEnabled = true;
                    this.cboTypeOfService.Location = new System.Drawing.Point(136, 35);
                    this.cboTypeOfService.Name = "cboTypeOfService";
                    this.cboTypeOfService.Size = new System.Drawing.Size(121, 21);
                    this.cboTypeOfService.TabIndex = 3;
                    this.cboTypeOfService.Validating                +=                new
System.ComponentModel.CancelEventHandler(this.Fields_Validating);
                    this.cboTypeOfService.Enter                    +=                    new
System.EventHandler(this.Fields_Enter);
                    //
                    // lblMaxValue
                    //
                    this.lblMaxValue.AutoSize = true;
                    this.lblMaxValue.Font   =   new   System.Drawing.Font("Microsoft   Sans
Serif",   8.25F,   System.Drawing.FontStyle.Regular,   System.Drawing.GraphicsUnit.Point,
((byte)(0)));
                    this.lblMaxValue.Location = new System.Drawing.Point(488, 19);
                    this.lblMaxValue.Name = "lblMaxValue";
                    this.lblMaxValue.Size = new System.Drawing.Size(70, 13);
                    this.lblMaxValue.TabIndex = 8;
                    this.lblMaxValue.Text = "Valor Máximo";
                    //
                    // txtMaxValue
                    //
                    this.txtMaxValue.Font   =   new   System.Drawing.Font("Microsoft   Sans
Serif",   8.25F,   System.Drawing.FontStyle.Regular,   System.Drawing.GraphicsUnit.Point,
((byte)(0)));
                    this.txtMaxValue.Location = new System.Drawing.Point(491, 35);
                    this.txtMaxValue.Multiline = true;
                    this.txtMaxValue.Name = "txtMaxValue";
                    this.txtMaxValue.Size = new System.Drawing.Size(93, 19);
                    this.txtMaxValue.TabIndex = 9;
                    this.toolTip.SetToolTip(this.txtMaxValue,   "Código   do   Banco   na
compensação");
                    this.txtMaxValue.Enter += new System.EventHandler(this.Fields_Enter);
                    this.txtMaxValue.Validating                    +=                    new
System.ComponentModel.CancelEventHandler(this.Fields_Validating);
                    //
                    // lblMinValue
                    //
                    this.lblMinValue.AutoSize = true;
                    this.lblMinValue.Font   =   new   System.Drawing.Font("Microsoft   Sans
Serif",   8.25F,   System.Drawing.FontStyle.Regular,   System.Drawing.GraphicsUnit.Point,
((byte)(0)));
                    this.lblMinValue.Location = new System.Drawing.Point(389, 19);
                    this.lblMinValue.Name = "lblMinValue";
                    this.lblMinValue.Size = new System.Drawing.Size(72, 13);
                    this.lblMinValue.TabIndex = 6;
                    this.lblMinValue.Text = "Valor Mínimo:";
                    //
                    // txtMinValue
                    //
                    this.txtMinValue.Font   =   new   System.Drawing.Font("Microsoft   Sans
Serif",   8.25F,   System.Drawing.FontStyle.Regular,   System.Drawing.GraphicsUnit.Point,
((byte)(0)));
                    this.txtMinValue.Location = new System.Drawing.Point(390, 35);
                    this.txtMinValue.MaxLength = 255;
                    this.txtMinValue.Name = "txtMinValue";
                    this.txtMinValue.Size = new System.Drawing.Size(95, 20);
                    this.txtMinValue.TabIndex = 7;
                    this.txtMinValue.Enter += new System.EventHandler(this.Fields_Enter);
                    this.txtMinValue.Validating                    +=                    new
System.ComponentModel.CancelEventHandler(this.Fields_Validating);
                    //
                    // btnSave
                    //
                    this.btnSave.FlatStyle = System.Windows.Forms.FlatStyle.Popup;
                    this.btnSave.Font = new System.Drawing.Font("Microsoft Sans Serif",
8.25F, System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
                    this.btnSave.ForeColor = System.Drawing.Color.MediumBlue;
```

```
                    this.btnSave.ImageAlign = System.Drawing.ContentAlignment.MiddleLeft;
                    this.btnSave.Location = new System.Drawing.Point(530, 316);
                    this.btnSave.Name = "btnSave";
                    this.btnSave.Size = new System.Drawing.Size(70, 24);
                    this.btnSave.TabIndex = 4;
                    this.btnSave.Text = "Salvar";
                    this.btnSave.TextAlign = System.Drawing.ContentAlignment.MiddleRight;
                    this.btnSave.Click += new System.EventHandler(this.btnSave_Click);
                    //
                    // btnDel
                    //
                    this.btnDel.FlatStyle = System.Windows.Forms.FlatStyle.Popup;
                    this.btnDel.Font = new System.Drawing.Font("Microsoft Sans Serif",
8.25F, System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
                    this.btnDel.ForeColor = System.Drawing.Color.Red;
                    this.btnDel.ImageAlign = System.Drawing.ContentAlignment.MiddleLeft;
                    this.btnDel.Location = new System.Drawing.Point(454, 316);
                    this.btnDel.Name = "btnDel";
                    this.btnDel.Size = new System.Drawing.Size(70, 24);
                    this.btnDel.TabIndex = 3;
                    this.btnDel.Text = "Excluir";
                    this.btnDel.TextAlign = System.Drawing.ContentAlignment.MiddleRight;
                    this.btnDel.Click += new System.EventHandler(this.btnDel_Click);
                    //
                    // btnNew
                    //
                    this.btnNew.FlatStyle = System.Windows.Forms.FlatStyle.Popup;
                    this.btnNew.Font = new System.Drawing.Font("Microsoft Sans Serif",
8.25F, System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
                    this.btnNew.ForeColor = System.Drawing.Color.MediumBlue;
                    this.btnNew.ImageAlign = System.Drawing.ContentAlignment.MiddleLeft;
                    this.btnNew.Location = new System.Drawing.Point(379, 316);
                    this.btnNew.Name = "btnNew";
                    this.btnNew.Size = new System.Drawing.Size(69, 24);
                    this.btnNew.TabIndex = 2;
                    this.btnNew.Text = "Novo";
                    this.btnNew.TextAlign = System.Drawing.ContentAlignment.MiddleRight;
                    this.btnNew.Click += new System.EventHandler(this.btnNew_Click);
                    //
                    // gpbList
                    //
                    this.gpbList.Controls.Add(this.lstStandardTolerances);
                    this.gpbList.Font = new System.Drawing.Font("Microsoft Sans Serif",
8.25F, System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
                    this.gpbList.Location = new System.Drawing.Point(4, 7);
                    this.gpbList.Name = "gpbList";
                    this.gpbList.Size = new System.Drawing.Size(596, 236);
                    this.gpbList.TabIndex = 0;
                    this.gpbList.TabStop = false;
                    this.gpbList.Text = "Tolerâncias Cadastradas:";
                    //
                    // lstStandardTolerances
                    //
                    this.lstStandardTolerances.Activation                                =
System.Windows.Forms.ItemActivation.TwoClick;
                    this.lstStandardTolerances.Columns.AddRange(new
System.Windows.Forms.ColumnHeader[] {
                    this.columnHeader5,
                    this.columnHeader1,
                    this.columnHeader3,
                    this.columnHeader2,
                    this.columnHeader4});
                    this.lstStandardTolerances.Dock                                      =
System.Windows.Forms.DockStyle.Fill;
                    this.lstStandardTolerances.Font = new System.Drawing.Font("Microsoft
Sans Serif", 8.25F, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point,
((byte)(0)));
                    this.lstStandardTolerances.FullRowSelect = true;
                    this.lstStandardTolerances.GridLines = true;
                    this.lstStandardTolerances.HoverSelection = true;
                    this.lstStandardTolerances.Location = new System.Drawing.Point(3,
16);
                    this.lstStandardTolerances.Margin                =                new
System.Windows.Forms.Padding(5);
                    this.lstStandardTolerances.MultiSelect = false;
                    this.lstStandardTolerances.Name = "lstStandardTolerances";
```

```
                    this.lstStandardTolerances.Size = new System.Drawing.Size(590, 217);
                    this.lstStandardTolerances.Sorting                                    =
System.Windows.Forms.SortOrder.Ascending;
                    this.lstStandardTolerances.TabIndex = 0;
                    this.lstStandardTolerances.UseCompatibleStateImageBehavior = false;
                    this.lstStandardTolerances.View = System.Windows.Forms.View.Details;
                    this.lstStandardTolerances.SelectedIndexChanged        +=        new
System.EventHandler(this.lstStandardTolerances_SelectedIndexChanged);
                    //
                    // columnHeader1
                    //
                    this.columnHeader1.Text = "Tipo de Serviço";
                    this.columnHeader1.Width = 140;
                    //
                    // columnHeader3
                    //
                    this.columnHeader3.Text = "Requisito de QoS";
                    this.columnHeader3.Width = 140;
                    //
                    // columnHeader5
                    //
                    this.columnHeader5.Text = "Norma";
                    this.columnHeader5.Width = 124;
                    //
                    // columnHeader2
                    //
                    this.columnHeader2.Text = "Mínimo";
                    this.columnHeader2.Width = 80;
                    //
                    // columnHeader4
                    //
                    this.columnHeader4.Text = "Máximo";
                    this.columnHeader4.Width = 80;
                    //
                    // errorProvider
                    //
                    this.errorProvider.ContainerControl = this;
                    //
                    // StandardToleranceDialog
                    //
                    this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);
                    this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
                    this.ClientSize = new System.Drawing.Size(605, 352);
                    this.Controls.Add(this.btnSave);
                    this.Controls.Add(this.gpbEdit);
                    this.Controls.Add(this.gpbList);
                    this.Controls.Add(this.btnDel);
                    this.Controls.Add(this.btnNew);
                    this.FormBorderStyle = System.Windows.Forms.FormBorderStyle.Fixed3D;
                    this.MaximizeBox = false;
                    this.MinimizeBox = false;
                    this.Name = "StandardToleranceDialog";
                    this.StartPosition                                                    =
System.Windows.Forms.FormStartPosition.CenterScreen;
                    this.Text = "Cadastro de Tolerância de Tipo de Serviço";
                    this.Load                              +=                             new
System.EventHandler(this.StandardToleranceDialog_Load);
                    this.gpbEdit.ResumeLayout(false);
                    this.gpbEdit.PerformLayout();
                    this.gpbList.ResumeLayout(false);

((System.ComponentModel.ISupportInitialize)(this.errorProvider)).EndInit();
                    this.ResumeLayout(false);

                }

                #endregion

                private System.Windows.Forms.GroupBox gpbEdit;
                private System.Windows.Forms.Button btnSave;
                private System.Windows.Forms.Label lblMaxValue;
                private System.Windows.Forms.TextBox txtMaxValue;
                private System.Windows.Forms.Label lblMinValue;
                private System.Windows.Forms.TextBox txtMinValue;
                private System.Windows.Forms.Button btnDel;
                private System.Windows.Forms.Button btnNew;
```

```
                private System.Windows.Forms.GroupBox gpbList;
                private System.Windows.Forms.ToolTip toolTip;
                private System.Windows.Forms.ErrorProvider errorProvider;
                private System.Windows.Forms.Label lblTypeOfService;
                private System.Windows.Forms.ComboBox cboTypeOfService;
                private System.Windows.Forms.Label lblQoSRequisite;
                private System.Windows.Forms.ComboBox cboQoSRequisite;
                private System.Windows.Forms.ListView lstStandardTolerances;
                private System.Windows.Forms.ColumnHeader columnHeader1;
                private System.Windows.Forms.ColumnHeader columnHeader3;
                private System.Windows.Forms.ColumnHeader columnHeader2;
                private System.Windows.Forms.ColumnHeader columnHeader4;
                private System.Windows.Forms.Label lblStandard;
                private System.Windows.Forms.ComboBox cboStandard;
                private System.Windows.Forms.ColumnHeader columnHeader5;
        }
}
```

2.16. A Classe TypeOfService

```
using System;
using System.Data;
using System.Data.OleDb;
using System.Collections.Generic;
using ObjectLibrary.Communication;

namespace PackageAnalyzator
{
        public class TypeOfService
        {
                private string m_Description;
                private string m_Name;
                private long m_ID;
                //Estados do objeto
                private bool m_IsHardObject;  //Existe no BD
                private bool m_IsSavedObject; //Atualizado no BD

                #region Constructors

                public TypeOfService()
                {
                        try
                        {
                                clear();
                        }
                        catch (Exception ex)
                        {
                                Exception tmpEx = new Exception("Erro ao carregar objeto",
ex);
                                throw tmpEx;
                        }
                }

                public TypeOfService(string Description, string Name, long ID)
                {
                        try
                        {
                                clear();
                                this.load(Description, Name, ID, false, false);
                        }
                        catch (Exception ex)
                        {
                                Exception tmpEx = new Exception("Erro ao carregar objeto",
ex);
                                throw tmpEx;
                        }
                }

                #endregion

                #region Properties
```

```
		//Todos objetos devem conter esta propriedade. Ela diz se o objeto existe
no banco de dados
		internal bool IsHardObject
		{
			get
			{
				return m_IsHardObject;
			}
		}

		//Todos objetos devem conter esta propriedade. Ela diz se o objeto está
salvo no banco de dados
		internal bool IsSavedObject
		{
			get
			{
				return m_IsSavedObject;
			}
		}

		public string Description
		{
			get
			{
				return m_Description;
			}
			set
			{
				m_Description = value;
				m_IsSavedObject = false;
			}
		}

		public string Name
		{
			get
			{
				return m_Name;
			}
			set
			{
				m_Name = value;
				m_IsSavedObject = false;
			}
		}

		public long ID
		{
			get
			{
				return m_ID;
			}
			set
			{
				m_ID = value;
				m_IsSavedObject = false;
			}
		}

		#endregion

		#region Static Methods

		public static TypeOfService findTypeOfService(long ID)
		{
			try
			{
				TypeOfService obj = new TypeOfService();

				List<TypeOfService>	alTypeOfServices	=	new
List<TypeOfService>();

				string clausulaWhere="";
				clausulaWhere =
					"	ID = " +  ID ;
```

```
                                alTypeOfServices = findAll(clausulaWhere);

                                if(alTypeOfServices.Count > 1)
                                {
                                        Exception tmpEx = new Exception("A procura retornou
mais de um objeto");
                                        throw tmpEx;
                                }
                                if(alTypeOfServices.Count == 1)
                                        obj = (TypeOfService) alTypeOfServices[0];
                                return obj;
                        }
                        catch (Exception ex)
                        {
                                Exception tmpEx = new Exception("Erro ao procurar objeto",
ex);
                                throw tmpEx;
                        }
                }

        public static List<TypeOfService> findTypeOfServices()
                {
                        try
                        {
                                string clausulaWhere = "";
                                List<TypeOfService>       alTypeOfServices       =       new
List<TypeOfService>();

                                alTypeOfServices = findAll(clausulaWhere);
                                return alTypeOfServices;
                        }
                        catch (Exception ex)
                        {
                                Exception tmpEx = new Exception("Erro ao procurar objeto",
ex);
                                throw tmpEx;
                        }
                }

                #endregion

                #region Instance Methods

                protected void loadTypeOfService(long ID)
                {
                        try
                        {
                                TypeOfService obj = TypeOfService.findTypeOfService(ID);

                                this.load(
                                                obj.Description,
                                                obj.Name,
                                                obj.ID,
                                                obj.IsHardObject,
                                                obj.IsSavedObject);

                        }
                        catch (Exception ex)
                        {
                                Exception tmpEx = new Exception("Erro ao carregar objeto",
ex);
                                throw tmpEx;
                        }
                }

        public virtual void save()
                {
                        try
                        {
                                validateProperties();

                                if (m_IsHardObject == false)
                                {
                                        insert();
```

```
                                        m_IsSavedObject = true;
                                }
                                else
                                {
                                        if (m_IsSavedObject == false)
                                        {
                                                update();
                                        }
                                }
                        }
                        catch (Exception ex)
                        {
                                Exception tmpEx = new Exception("Erro ao salvar objeto",
ex);
                                throw tmpEx;
                        }
                }

        public virtual void remove()
                {
                        try
                        {
                                int recordsAffected;
                                string sql="";

                                if (m_IsHardObject == false)
                                {
                                        Exception tmpEx = new Exception("O objeto não
existe no banco de dados!");
                                        throw tmpEx;
                                }

                                sql = " DELETE FROM " +
                                        "  TypeOfService" +
                                        " WHERE " +
                                        "   ID = " +  ID ;

                                OleDbCommand      cmm      =      new      OleDbCommand(sql,
DataBase.Connection);
                                recordsAffected = cmm.ExecuteNonQuery();

                                if (recordsAffected > 1)
                                {
                                        Exception tmpEx = new Exception("A atualização do
objeto afetou mais de um registro no banco de dados.");
                                        throw tmpEx;
                                }

                                if (recordsAffected == 0)
                                {
                                        Exception tmpEx = new Exception("Registro alterado
anteriormente. Consulte novamente para atualizar e repita a operação.");
                                        throw tmpEx;
                                }

                                clear();
                                m_IsHardObject = false;
                                m_IsSavedObject = false;
                        }
                        catch (Exception ex)
                        {
                                Exception tmpEx = new Exception("Objeto não foi criado no
banco de dados, portanto não pode ser excluído fisicamente.", ex);
                                throw tmpEx;
                        }
                }

                #endregion

                #region Private Methods

                private void validateProperties()
                {
                        try
                        {
                                if (m_Description == "")
```

```
                                {
                                        Exception  tmpEx  =  new  Exception("Description  não
preenchido!");
                                        throw tmpEx;
                                }
                                if (m_Name == "")
                                {
                                        Exception   tmpEx   =   new   Exception("Name   não
preenchido!");
                                        throw tmpEx;
                                }
                        }
                        catch (Exception ex)
                        {
                                Exception tmpEx = new Exception("Erro ao validar objeto",
ex);
                                throw tmpEx;
                        }
                }

                private void insert()
                {
                        try
                        {
                                string sql="";
                                long NextID = 0;

                                NextID = getNextID();

                                sql = " INSERT INTO" +
                                      "       TypeOfService" +
                                      " (Description, Name, ID) " +
                                      " VALUES ( " +
                                       "'" + m_Description + "'" + "," +
                                       "'" + m_Name + "'" + "," +
                                        NextID + ")";

                                OleDbCommand      cmm      =      new      OleDbCommand(sql,
DataBase.Connection);
                                cmm.ExecuteNonQuery();

                                m_ID = NextID;

                                m_IsHardObject = true;
                                m_IsSavedObject = true;
                        }
                        catch (Exception ex)
                        {
                                Exception tmpEx = new Exception("Objeto não foi criado no
banco de dados", ex);
                                throw tmpEx;
                        }
                }

                private long getNextID()
                {
                        try
                        {
                                string sql="";
                                long NextID = 0;

                                DataTable dt = new DataTable();
                 sql = " SELECT MAX(ID) as LastTypeOfService" +
                                              " FROM TypeOfService ";

                                OleDbCommand      cmm      =      new      OleDbCommand(sql,
DataBase.Connection);
                                OleDbDataAdapter da = new OleDbDataAdapter(cmm);
                                da.Fill(dt);

                                DataRow dr = dt.Rows[0];
                 if (dr["LastTypeOfService"].ToString() != "")
                                {
                     NextID = Convert.ToUInt32(dr["LastTypeOfService"]);
                                }
```

```
                                NextID ++;
                                return NextID;

                        }
                        catch (Exception ex)
                        {
                                Exception tmpEx = new Exception("Erro ao procurar próximo
id", ex);
                                throw tmpEx;
                        }
                }

                private void update()
                {
                        try
                        {
                                int recordsAffected;
                                string sql="";

                                sql = " UPDATE TypeOfService SET" +
                                        " Description = " + "'" + m_Description + "'" + ",
" +
                                        " Name = " + "'" + m_Name + "'" +
                                       " WHERE " +
                                              "   ID = " +  ID ;

                                OleDbCommand             cmm             =             new
OleDbCommand(sql,DataBase.Connection);
                                recordsAffected = cmm.ExecuteNonQuery();

                                if (recordsAffected > 1)
                                {
                                        Exception tmpEx = new Exception("A atualização do
objeto afetou mais de um registro no banco de dados.");
                                        throw tmpEx;
                                }

                                if (recordsAffected == 0)
                                {
                                        Exception tmpEx = new Exception("Registro alterado
anteriormente. Consulte novamente para atualizar e repita a operação.");
                                        throw tmpEx;
                                }

                                m_IsHardObject = true;
                                m_IsSavedObject = true;

                        }
                        catch (Exception ex)
                        {
                                Exception tmpEx = new Exception("Objeto não foi atualizado
no banco de dados", ex);
                                throw tmpEx;
                        }
                }

        private void load(string Description, string Name, long ID, bool isHardObject,
bool isSavedObject)
                {
                        try
                        {
                                m_Description = Description;
                                m_Name = Name;
                                m_ID = ID;
                                m_IsHardObject = isHardObject;
                                m_IsSavedObject = isSavedObject;

                        }
                        catch (Exception ex)
                        {
                                Exception tmpEx = new Exception("Erro de conversão de tipos
de dados", ex);
                                throw tmpEx;
                        }
                }
```

```
		private void clear()
		{
			try
			{
				m_Description = "";
				m_Name = "";
				m_ID = 0;
				m_IsHardObject = false;
				m_IsSavedObject = false;
			}
			catch (Exception ex)
			{
				Exception tmpEx = new Exception("Erro ao limpar objeto",
ex);
				throw tmpEx;
			}
		}

		private static List<TypeOfService> findAll(string clausulaWhere)
		{
			try
			{
				List<TypeOfService> alTypeOfServices = new
List<TypeOfService>();
				string sql="";
				DataTable dt = new DataTable();

				sql = " SELECT " +
				  "  Description, Name, ID " +
							" FROM " +
							"  TypeOfService";

				if(clausulaWhere != "")
				{
					sql = sql + " WHERE " + clausulaWhere;
				}

				OleDbCommand cmm = new OleDbCommand(sql,
DataBase.Connection);
				OleDbDataAdapter da = new OleDbDataAdapter(cmm);
				da.Fill(dt);

				foreach (DataRow dr in dt.Rows)
				{
					TypeOfService obj = new TypeOfService();
					obj.load(
						Convert.ToString(dr["Description"]),
						Convert.ToString(dr["Name"]),
						Convert.ToUInt32(dr["ID"]),
						true, true);

					alTypeOfServices.Add(obj);
				}
				return alTypeOfServices;
			}
			catch (Exception ex)
			{
				Exception tmpEx = new Exception("Erro ao procurar objeto",
ex);
				throw tmpEx;
			}
		}

		#endregion

	}
}
```

2.17. A Classe TypeOfServiceDialog

```
using System;
using System.Collections.Generic;
```

```
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;

using ObjectLibrary.ErrorHandler;
using ObjectLibrary.Communication;

namespace PackageAnalyzator
{
    public partial class TypeOfServiceDialog : Form
    {

        private TypeOfService m_TypeOfService;

        public TypeOfServiceDialog()
        {
            try
            {
                InitializeComponent();
            }
            catch (Exception ex)
            {
                throw new Exception("Erro ao instanciar objeto.", ex);
            }
        }

        private void TypeOfServiceDialog_Load(object sender, EventArgs e)
        {
            try
            {
                clear();
                fillListTypeOfServices();
            }
            catch (Exception ex)
            {
                ErrorHandlerDialog.ShowErroHandler(ref ex);
            }
        }

        private void clear()
        {
            try
            {
                m_TypeOfService = null;

                errorProvider.SetError(txtDescription, "");
                errorProvider.SetError(txtName, "");

                txtDescription.Text = "";
                txtName.Text = "";
            }
            catch (Exception ex)
            {
                throw new Exception("Erro ao limpar Tipo de Serviço", ex);
            }
        }

        private void fillListTypeOfServices()
        {
            try
            {
                lstTypeOfServices.Items.Clear();

                lstTypeOfServices.DisplayMember = "Name";

                List<TypeOfService>                  alTypeOfServices                  =
TypeOfService.findTypeOfServices();
                foreach (TypeOfService obj in alTypeOfServices)
                {
                    lstTypeOfServices.Items.Add(obj);
                }
            }
            catch (Exception ex)
            {
```

```
                throw new Exception("Erro ao listar Tipos de Serviço", ex);
            }
        }

        private void btnNew_Click(object sender, EventArgs e)
        {
            try
            {
                clear();
            }
            catch (Exception ex)
            {
                ErrorHandlerDialog.ShowErroHandler(ref ex);
            }
        }

        private void btnRefresh_Click(object sender, EventArgs e)
        {
            try
            {
                fillListTypeOfServices();
            }
            catch (Exception ex)
            {
                ErrorHandlerDialog.ShowErroHandler(ref ex);
            }
        }

        private void btnSave_Click(object sender, EventArgs e)
        {
            try
            {
                if (fieldsValidated("") == false)
                    return;

                DialogResult dlgr =
                    MessageBox.Show(this, "Tem certeza que deseja salvar?",
                    "Atenção!",
                    MessageBoxButtons.YesNo,
                    MessageBoxIcon.Question);

                if (dlgr == DialogResult.No)
                    return;

                save();
                MessageBox.Show(this,
                    "Tipo de Serviço salvo com sucesso!",
                    "Analizador",
                    MessageBoxButtons.OK,
                    MessageBoxIcon.Information);

                clear();
                fillListTypeOfServices();
            }
            catch (Exception ex)
            {
                ErrorHandlerDialog.ShowErroHandler(ref ex);
            }
        }

        private bool fieldsValidated(string nameField)
        {
            try
            {
                bool errorControl = true;
                bool result = true;
                return errorControl;
            }
            catch (Exception ex)
            {
                throw new Exception("Erro ao validar os dados do Tipo de Serviço.", ex);
            }
        }

        private void save()
        {
```

```
            try
            {
                if (m_TypeOfService == null)
                    m_TypeOfService = new TypeOfService();

                m_TypeOfService.Name = txtName.Text;
                m_TypeOfService.Description = txtDescription.Text;

                //IDbTransaction tran = DataBase.Connection.BeginTransaction();
                try
                {
                    m_TypeOfService.save();
                    //tran.Commit();
                }
                catch (Exception ex)
                {
                    //tran.Rollback();
                    throw new Exception("Erro ao salvar Tipo de Serviço", ex);
                }
            }
            catch (Exception ex)
            {
                throw new Exception("Erro ao salvar Tipo de Serviço", ex);
            }
        }

        private void lstTypeOfServices_SelectedIndexChanged(object sender, EventArgs e)
        {
            try
            {
                if (lstTypeOfServices.SelectedItem != null)
                {
                    m_TypeOfService = (TypeOfService)lstTypeOfServices.SelectedItem;
                    loadTypeOfService();
                }
            }
            catch (Exception ex)
            {
                ErrorHandlerDialog.ShowErroHandler(ref ex);
            }
        }

        private void loadTypeOfService()
        {
            try
            {
                txtName.Text = m_TypeOfService.Name;
                txtDescription.Text = m_TypeOfService.Description;
            }
            catch (Exception ex)
            {
                throw new Exception("Erro ao selecionar Tipo de Serviço", ex);
            }
        }

        private void btnDel_Click(object sender, EventArgs e)
        {
            try
            {
                if (m_TypeOfService == null)
                {
                    throw new Exception("Por favor selecione um item primeiro.");
                }

                DialogResult dlgr =
                    MessageBox.Show(this, "Tem certeza que deseja excluir?",
                    "Atenção!",
                    MessageBoxButtons.YesNo,
                    MessageBoxIcon.Question);
                if (dlgr == DialogResult.No)
                    return;

                delete();

                MessageBox.Show(this,
                    "Tipo de Serviço excluído com sucesso!",
```

```
                    "Analizador",
                    MessageBoxButtons.OK,
                    MessageBoxIcon.Information);

                fillListTypeOfServices();
                clear();
            }
            catch (Exception ex)
            {
                ErrorHandlerDialog.ShowErroHandler(ref ex);
            }
        }

        private void delete()
        {
            try
            {
                //IDbTransaction tran = DataBase.Connection.BeginTransaction();
                try
                {
                    m_TypeOfService.remove();
                    //tran.Commit();
                }
                catch (Exception ex)
                {
                    //tran.Rollback();
                    throw new Exception("Erro ao deletar Tipo de Serviço", ex);
                }

            }
            catch (Exception ex)
            {
                throw new Exception("Erro ao deletar Tipo de Serviço", ex);
            }
        }

        private void Fields_Validating(object sender, CancelEventArgs e)
        {
            try
            {
                Control c = (Control)sender;
                fieldsValidated(c.Name);
            }
            catch (Exception ex)
            {
                ErrorHandlerDialog.ShowErroHandler(ref ex);
            }
        }

        private void Fields_Enter(object sender, EventArgs e)
        {
            try
            {
                Control c = (Control)sender;
                errorProvider.SetError(c, "");
            }
            catch (Exception ex)
            {
                ErrorHandlerDialog.ShowErroHandler(ref ex);
            }
        }

    }
}

        namespace PackageAnalyzator
        {
            partial class TypeOfServiceDialog
            {
                /// <summary>
                /// Required designer variable.
                /// </summary>
                private System.ComponentModel.IContainer components = null;

                /// <summary>
                /// Clean up any resources being used.
```

```
                /// </summary>
                /// <param name="disposing">true if managed resources should be disposed;
otherwise, false.</param>
                protected override void Dispose(bool disposing)
                {
                    if (disposing && (components != null))
                    {
                        components.Dispose();
                    }
                    base.Dispose(disposing);
                }

                #region Windows Form Designer generated code

                /// <summary>
                /// Required method for Designer support - do not modify
                /// the contents of this method with the code editor.
                /// </summary>
                private void InitializeComponent()
                {
                    this.components = new System.ComponentModel.Container();
                    this.gpbEdit = new System.Windows.Forms.GroupBox();
                    this.lblDescription = new System.Windows.Forms.Label();
                    this.txtDescription = new System.Windows.Forms.TextBox();
                    this.lblName = new System.Windows.Forms.Label();
                    this.txtName = new System.Windows.Forms.TextBox();
                    this.btnSave = new System.Windows.Forms.Button();
                    this.btnDel = new System.Windows.Forms.Button();
                    this.btnNew = new System.Windows.Forms.Button();
                    this.gpbList = new System.Windows.Forms.GroupBox();
                    this.lstTypeOfServices = new System.Windows.Forms.ListBox();
                    this.toolTip = new System.Windows.Forms.ToolTip(this.components);
                    this.errorProvider                    =                    new
System.Windows.Forms.ErrorProvider(this.components);
                    this.gpbEdit.SuspendLayout();
                    this.gpbList.SuspendLayout();

((System.ComponentModel.ISupportInitialize)(this.errorProvider)).BeginInit();
                    this.SuspendLayout();
                    //
                    // gpbEdit
                    //
                    this.gpbEdit.Controls.Add(this.lblDescription);
                    this.gpbEdit.Controls.Add(this.txtDescription);
                    this.gpbEdit.Controls.Add(this.lblName);
                    this.gpbEdit.Controls.Add(this.txtName);
                    this.gpbEdit.Font = new System.Drawing.Font("Microsoft Sans Serif",
8.25F, System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
                    this.gpbEdit.Location = new System.Drawing.Point(4, 199);
                    this.gpbEdit.Name = "gpbEdit";
                    this.gpbEdit.Size = new System.Drawing.Size(336, 126);
                    this.gpbEdit.TabIndex = 1;
                    this.gpbEdit.TabStop = false;
                    this.gpbEdit.Text = "Novo / Editar";
                    //
                    // lblDescription
                    //
                    this.lblDescription.AutoSize = true;
                    this.lblDescription.Font = new System.Drawing.Font("Microsoft Sans
Serif", 8.25F, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point,
((byte)(0)));
                    this.lblDescription.Location = new System.Drawing.Point(8, 66);
                    this.lblDescription.Name = "lblDescription";
                    this.lblDescription.Size = new System.Drawing.Size(58, 13);
                    this.lblDescription.TabIndex = 2;
                    this.lblDescription.Text = "Descrição:";
                    //
                    // txtDescription
                    //
                    this.txtDescription.Font = new System.Drawing.Font("Microsoft Sans
Serif", 8.25F, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point,
((byte)(0)));
                    this.txtDescription.Location = new System.Drawing.Point(11, 82);
                    this.txtDescription.Multiline = true;
                    this.txtDescription.Name = "txtDescription";
                    this.txtDescription.Size = new System.Drawing.Size(317, 38);
```

```
                    this.txtDescription.TabIndex = 3;
                    this.toolTip.SetToolTip(this.txtDescription,  "Código  do  Banco  na
compensação");
                    this.txtDescription.Enter                    +=                    new
System.EventHandler(this.Fields_Enter);
                    this.txtDescription.Validating                  +=                 new
System.ComponentModel.CancelEventHandler(this.Fields_Validating);
                    //
                    // lblName
                    //
                    this.lblName.AutoSize = true;
                    this.lblName.Font  =  new  System.Drawing.Font("Microsoft  Sans  Serif",
8.25F,        System.Drawing.FontStyle.Regular,        System.Drawing.GraphicsUnit.Point,
((byte)(0)));
                    this.lblName.Location = new System.Drawing.Point(8, 19);
                    this.lblName.Name = "lblName";
                    this.lblName.Size = new System.Drawing.Size(38, 13);
                    this.lblName.TabIndex = 0;
                    this.lblName.Text = "Nome:";
                    //
                    // txtName
                    //
                    this.txtName.Font  =  new  System.Drawing.Font("Microsoft  Sans  Serif",
8.25F,        System.Drawing.FontStyle.Regular,        System.Drawing.GraphicsUnit.Point,
((byte)(0)));
                    this.txtName.Location = new System.Drawing.Point(9, 35);
                    this.txtName.MaxLength = 255;
                    this.txtName.Name = "txtName";
                    this.txtName.Size = new System.Drawing.Size(319, 20);
                    this.txtName.TabIndex = 1;
                    this.txtName.Enter += new System.EventHandler(this.Fields_Enter);
                    this.txtName.Validating                      +=                    new
System.ComponentModel.CancelEventHandler(this.Fields_Validating);
                    //
                    // btnSave
                    //
                    this.btnSave.FlatStyle = System.Windows.Forms.FlatStyle.Popup;
                    this.btnSave.Font  =  new  System.Drawing.Font("Microsoft  Sans  Serif",
8.25F, System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
                    this.btnSave.ForeColor = System.Drawing.Color.MediumBlue;
                    this.btnSave.ImageAlign = System.Drawing.ContentAlignment.MiddleLeft;
                    this.btnSave.Location = new System.Drawing.Point(270, 331);
                    this.btnSave.Name = "btnSave";
                    this.btnSave.Size = new System.Drawing.Size(70, 24);
                    this.btnSave.TabIndex = 4;
                    this.btnSave.Text = "Salvar";
                    this.btnSave.TextAlign = System.Drawing.ContentAlignment.MiddleRight;
                    this.btnSave.Click += new System.EventHandler(this.btnSave_Click);
                    //
                    // btnDel
                    //
                    this.btnDel.FlatStyle = System.Windows.Forms.FlatStyle.Popup;
                    this.btnDel.Font  =  new  System.Drawing.Font("Microsoft  Sans  Serif",
8.25F, System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
                    this.btnDel.ForeColor = System.Drawing.Color.Red;
                    this.btnDel.ImageAlign = System.Drawing.ContentAlignment.MiddleLeft;
                    this.btnDel.Location = new System.Drawing.Point(194, 331);
                    this.btnDel.Name = "btnDel";
                    this.btnDel.Size = new System.Drawing.Size(70, 24);
                    this.btnDel.TabIndex = 3;
                    this.btnDel.Text = "Excluir";
                    this.btnDel.TextAlign = System.Drawing.ContentAlignment.MiddleRight;
                    this.btnDel.Click += new System.EventHandler(this.btnDel_Click);
                    //
                    // btnNew
                    //
                    this.btnNew.FlatStyle = System.Windows.Forms.FlatStyle.Popup;
                    this.btnNew.Font  =  new  System.Drawing.Font("Microsoft  Sans  Serif",
8.25F, System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
                    this.btnNew.ForeColor = System.Drawing.Color.MediumBlue;
                    this.btnNew.ImageAlign = System.Drawing.ContentAlignment.MiddleLeft;
                    this.btnNew.Location = new System.Drawing.Point(119, 331);
                    this.btnNew.Name = "btnNew";
                    this.btnNew.Size = new System.Drawing.Size(69, 24);
                    this.btnNew.TabIndex = 2;
                    this.btnNew.Text = "Novo";
```

```
                    this.btnNew.TextAlign = System.Drawing.ContentAlignment.MiddleRight;
                    this.btnNew.Click += new System.EventHandler(this.btnNew_Click);
                    //
                    // gpbList
                    //
                    this.gpbList.Controls.Add(this.lstTypeOfServices);
                    this.gpbList.Font = new System.Drawing.Font("Microsoft Sans Serif",
8.25F, System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
                    this.gpbList.Location = new System.Drawing.Point(4, 7);
                    this.gpbList.Name = "gpbList";
                    this.gpbList.Size = new System.Drawing.Size(336, 189);
                    this.gpbList.TabIndex = 0;
                    this.gpbList.TabStop = false;
                    this.gpbList.Text = "Tipos de Serviço Cadastrados:";
                    //
                    // lstTypeOfServices
                    //
                    this.lstTypeOfServices.Font = new System.Drawing.Font("Microsoft Sans
Serif",   8.25F,   System.Drawing.FontStyle.Regular,   System.Drawing.GraphicsUnit.Point,
((byte)(0)));
                    this.lstTypeOfServices.Location = new System.Drawing.Point(9, 21);
                    this.lstTypeOfServices.Name = "lstTypeOfServices";
                    this.lstTypeOfServices.Size = new System.Drawing.Size(319, 160);
                    this.lstTypeOfServices.Sorted = true;
                    this.lstTypeOfServices.TabIndex = 0;
                    this.lstTypeOfServices.SelectedIndexChanged          +=          new
System.EventHandler(this.lstTypeOfServices_SelectedIndexChanged);
                    //
                    // errorProvider
                    //
                    this.errorProvider.ContainerControl = this;
                    //
                    // TypeOfServiceDialog
                    //
                    this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);
                    this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
                    this.ClientSize = new System.Drawing.Size(345, 367);
                    this.Controls.Add(this.btnSave);
                    this.Controls.Add(this.gpbEdit);
                    this.Controls.Add(this.gpbList);
                    this.Controls.Add(this.btnDel);
                    this.Controls.Add(this.btnNew);
                    this.FormBorderStyle = System.Windows.Forms.FormBorderStyle.Fixed3D;
                    this.MaximizeBox = false;
                    this.MinimizeBox = false;
                    this.Name = "TypeOfServiceDialog";
                    this.StartPosition                                                  =
System.Windows.Forms.FormStartPosition.CenterScreen;
                    this.Text = "Cadastro de Tipo de Serviço";
                    this.Load += new System.EventHandler(this.TypeOfServiceDialog_Load);
                    this.gpbEdit.ResumeLayout(false);
                    this.gpbEdit.PerformLayout();
                    this.gpbList.ResumeLayout(false);

((System.ComponentModel.ISupportInitialize)(this.errorProvider)).EndInit();
                    this.ResumeLayout(false);

                }

                #endregion

                private System.Windows.Forms.GroupBox gpbEdit;
                private System.Windows.Forms.Button btnSave;
                private System.Windows.Forms.Label lblDescription;
                private System.Windows.Forms.TextBox txtDescription;
                private System.Windows.Forms.Label lblName;
                private System.Windows.Forms.TextBox txtName;
                private System.Windows.Forms.Button btnDel;
                private System.Windows.Forms.Button btnNew;
                private System.Windows.Forms.GroupBox gpbList;
                private System.Windows.Forms.ListBox lstTypeOfServices;
                private System.Windows.Forms.ToolTip toolTip;
                private System.Windows.Forms.ErrorProvider errorProvider;
            }
}
```

3. O Programa Sniffer (Farejador)

3.1. A Classe Program

```
using System;
using System.Collections.Generic;
using System.Windows.Forms;

using ObjectLibrary;

namespace Sniffer
{
    static class Program
    {
        /// <summary>
        /// The main entry point for the application.
        /// </summary>
        [STAThread]
        static void Main()
        {
            Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);
            Application.Run(new PackageSnifferDialog());
        }
    }
}
```

Apêndice F – Procedimento para instalação do cliente NTP

Instalação

Para instalar o programa Dimension 4 – cliente de NTP, siga os seguintes passos:

1. Fazer o download da ferramenta a partir do seguinte site: http://www.thinkman.com/dimension4

2. Proceder com a instalação da mesma.

3. Após a instalação, será preciso realizar alguns ajustes na configuração.

Configuração

1. Execute o programa dimension4. O programa irá inicializar e ficará minimizado na barra inferior próximo ao relógio do Windows.

2. Posicionando o mouse em cima do ícone minimizado, clique o botão direito do mouse e escolha a opção Open. Imediatamente, uma janela de título Dimension 4 será mostrada na tela.

3. A configuração em si envolve os seguintes passos:
    a. Remover todas as opções de servidores NTP listados, alguns deles não disponibilizam mais o serviço NTP e outros não podem ser livremente acessados.

    b. Adicionar na lista o seu servidor NTP local, para tal basta:
        i. Clicar no botão Add
        ii. Preencher o campo Server de acordo com os Servidores listados no apêndice H.

iii. Preencher o campo Location com alguma informação que permita identificar o servidor, por exemplo: "Servidor NTP Local"
iv. Escolher o SNTP como protocolo a ser usado
v. Clicar no botão OK para efetivar a inclusão do servidor NTP

c. Clicar no botão “Advanced”. Uma janela de título Dimension 4 Advanced. A janela Settings será mostrada.
    i. Na caixa de diálogo “Message Boxes” desabilitar as opções “Display Errors” e “Display Syncronization”
    ii. Clicar no botão OK

d. Na caixa de diálogo “How Often”, configurar de modo que a cada 30 minutos seja feita a sincronização.
e. Por último, clicar no botão “OK” e minimizar o programa. A configuração default já considera a execução automática do programa após a reinicialização da máquina. (CAIS, 2000)

Apêndice G – Lista de Servidores NTP no Brasil

1. ntp.cais.rnp.br (200.144.121.33)
   Location: Brazilian Research Network/Rede Nacional de Pesquisa (RNP)
   Synchronization: NTP V4 Secondary (stratum 2), SunSparc10/Solaris
   Service Area: Brazil
   Access Policy: Open access to stratum 2 and stratum 3 NTP servers. Please, send an e-mail to notify.
   Contact: ntp-admin@cais.rnp.br

2. ntp1.pucpr.br (200.192.112.8)
   Location: Pontificia Universidade Catolica do Paraná
   Synchronization: NTP V4 Secondary (stratum 2), SunSparc4/Solaris7
   Service Area: Brazil
   Access Policy:open access to stratum 2 servers. Please, send an e-mail to notify.
   Contact: ntp1@pucpr.br

3. ntp.pop-sc.rnp.br (200.135.0.3)
   Location: PoP-SC/Brazilian Research Network (RNP)
   Synchronization: NTP V 3.4y (stratum 2), AIX 4.3.1.0
   Service Area: Brazil
   Access Policy: Open access to stratum 2 and stratum 3 NTP servers. Please, send an e-mail to notify.
   Contact: ntp-admin@pop-sc.rnp.br

4. ntp.pop-rs.rnp.br (200.132.0.132)
   Location: PoP-RS/Brazilian Research Network (RNP)
   Synchronization: NTP V4 Secondary (stratum 2), SunSparc10/Solaris
   Service Area: Brazil
   Access Policy: Open access to stratum 2 and stratum 3 NTP servers. Please, send an e-mail to notify.
   Contact: ntp-admin@pop-rs.rnp.br

5. ntp.cert-rs.tche.br (200.132.0.157)
   Location: CERT-RS/Rede Tche
   Synchronization: NTP V4 Secondary (stratum 2), IBM RS6000/Aix
   Service Area: Brazil
   Access Policy: Open access to stratum 2 and stratum 3 NTP servers. Please, send an e-mail to notify.
   Contact: ntp-admin@pop-rs.rnp.br

6. ntp.pop-se.rnp.br (200.17.118.129)
   Location: PoP-SE/Brazilian Research Network (RNP)
   Synchronization: NTP Secondary (stratum 2), Cisco 2500/IOS 11.0(22)
   Service Area: Brazil
   Access Policy: Open access to stratum 2 and stratum 3 NTP servers.

Please, send an e-mail to notify.
Contact: ntp-admin@pop-se.rnp.br

7. ntp.pop-pi.rnp.br (200.137.160.129)
   Location: PoP-PI/Brazilian Research Network (RNP)
   Synchronization: NTP Secondary (stratum2), Cisco 2501/IOS 11.0
   Service Area : Brazil
   Access Policy: Open access to stratum 2 and stratum 3 NTP servers.
   Please, send an e-mail to notify.
   Contact: ntp-admin@pop-pi.rnp.br

8. ntp.pop-am.rnp.br (200.129.156.2)
   Location:PoP-AM/Brazilian Research Network (RNP)
   Synchronization: NTP V4 secundario (stratum 2), Netfinity/Linux
   Service Area: Brazil
   Access Policy: Open access to stratum 2 and stratum 3 NTP servers.
   Please, send an e-mail to notify.
   Contact: ntp-admin@pop-am.rnp.br

9. ntp.puc-rio.br (139.82.34.11)
   Location: Pontifícia Universidade Católica do Rio de Janeiro
   Synchronization: NTP V3 Secondary (stratum 2), Linux
   Service Area: Brazil
   Access Policy: Open access to stratum 2 NTP servers. Please, send an e-mail to notify.
   Contact: ntp-admin@puc-rio.br

10. ntp.pop-pe.rnp.br (200.133.0.35)
    Location: PoP-PE/Brazilian Research Network (RNP)
    Synchronization: NTP V4 Secondary (stratum 2),
    AIX 4.3.2
    Service Area: Brazil
    Access Policy: Open access to stratum 2 and stratum 3 NTP servers.
    Please, send an e-mail to notify.
    Contact: ntp-admin@pop-pe.rnp.br

11. ntp.pop-ba.rnp.br (192.188.11.33)
    Location: POP-BA/Brazilian Research Network (RNP)
    Synchronization: NTP V4 Secondary (stratum 2), AIX 4.2.1
    Service Area: Brazil
    Access Policy: Open access to stratum 2 and stratum 3 NTP servers.
    Please, send an e-mail to notify.
    Contact: ntp-admin@pop-ba.rnp.br

12. ntp.pop-df.rnp.br (200.19.119.119)
    Location: PoP-DF/Brazilian Research Network (RNP)
    Syncronization: NTP V4 Secondary (stratum 2), FreeBSD/Unix
    Service Area: Brazil
    Access Policy: Open access to stratum 2 and stratum 3 NTP servers.
    Please, send an e-mail to notify.
    Contact: ntp-admin@pop-df.rnp.br

13. ntp.pop-rn.rnp.br (200.137.0.41)
    Location: PoP-RN/Brazilian Research Network (RNP)
    Syncronization: NTP Secondary (stratum 2), Cisco 2500/IOS 11.0(4)
    Service Area: Brazil
    Access Policy: Open access to stratum 2 and stratum 3 NTP servers. Please, send an e-mail to notify.
    Contact: ntp-admin@pop-rn.rnp.br

14. ntp.pop-pr.rnp.br (200.238.128.103)
    Location: PoP-PR/Brazilian Research Network (RNP)
    Synchronization: NTP V4 Secondary (stratum 2), Netfinity/Linux 2.2.14
    Service Area: Brazil
    Access Policy: Restrict access to stratum 2 and stratum 3 servers. Please, send an e-mail to get access.
    Contact: ntp-admin@pop-pr.rnp.br

15. ntp.pop-ce.rnp.br (200.129.0.35)
    Location:PoP-CE/Brazilian Research Network (RNP)
    Synchronization: NTP V4 Secondary (stratum 2), AIX 4.2.1.0
    Service Area: Brazil
    Access Policy: Open access to stratum 2 and stratum 3 NTP servers. Please, send an e-mail to notify.
    Contact: ntp-admin@pop-ce.rnp.br

16. ntp.pop-mg.rnp.br (200.131.1.21)
    Location: POP-MG / Brazilian Research Network (RNP)
    Synchronization: NTP Secondary (stratum 2), Cisco 7507/IOS 12.0(14)
    Service area: Brazil
    Access Policy: Open access to stratum 2 and stratum 3 NTP servers. Please, send an e-mail to notify.
    Contact: ntp-admin@pop-mg.rnp.br

17. ntp.pop-al.rnp.br (200.17.117.130)
    Location : POP-AL / Brazilian Research Network (RNP)
    Synchronization: NTP V4 Secondary (stratum 2), IBM RS6000/AIX
    Service Area : Brazil
    Access Policy : Open access to stratum 2 and stratum 3 NTP servers within Brazilian Research Network (RNP). Please, send an e-mail to notify.
    Contact : ntp-admin@pop-al.rnp.br

18. ntp.pop-es.rnp.br (200.137.65.132)
    Location: POP-ES / Brazilian Research Network (RNP)
    Synchronization: NTP V4 Secondary (stratum 2), AIX 4
    Service Area: Brazil
    Access Policy: Open access to stratum 2 and stratum 3 NTP servers. Please, send an e-mail to get access.
    Contact: ntp-admin@pop-es.rnp.br

19. ntp.pop-pa.rnp.br (200.129.132.130)
    Location: POP-PA / Brazilian Research Network (RNP)

Synchronization: NTP V4 Secondary (stratum 2), Intel-x86/Linux
Service Area: Brazil
Access Policy: Open access to stratum 2 and stratum 3 NTP servers. Please, send an e-mail to notify.
Contact: ntp-admin@pop-pa.rnp.br

20. ntp.pop-pb.rnp.br (200.129.64.130)
Location: POP-PB / Brazilian Research Network (RNP)
Synchronization: NTP V4 Secondary (stratum 2), IBM Netfinity/Linux 2.2.16
Service Area: Brazil
Access Policy: Open access to stratum 2 and stratum 3 NTP servers. Please, send an e-mail to notify.
Contact: ntp-admin@pop-pb.rnp.br

21. ntp.pop-go.rnp.br (200.18.160.5)
Location: POP-GO / Brazilian Research Network (RNP)
Synchronization: NTP Secondary (stratum 2), Linux 2.2.18
Service area: Brazil
Access Policy: Restrict access to stratum 2 and stratum 3 servers. Please, send an e-mail to get access.
Contact: ntp-admin@pop-go.rnp.br

22. ntp.pop-mt.rnp.br (200.17.60.221)
Location: POP-MT / Brazilian Research Network (RNP)
Synchronization: NTP V4 Secondary (stratum 2), Linux/Slackware
Service area: Brazil
Access Policy: Open access to stratum 2 and stratum 3 NTP servers. Please, send an e-mail to notify.
Contact: ntp-admin@pop-mt.rnp.br

23. ntp.nc-rj.rnp.br (200.17.63.121)
Location: Brazilian Research Network/Rede Nacional de Pesquisa (RNP)
Synchronization: NTP V4 Secondary (stratum 2), IBM PC Server704/FreeBSD 4.3
Service Area: Brazil
Access Policy: Open access to NTP clients from NC-RJ.
Contact: ntp-admin@nc-rj.rnp.br

24. ntp.na-df.rnp.br (200.130.77.40)
Location: Núcleo de Apoio de Brasília/Brazilian Research Network (RNP)
Synchronization: NTP V4 Secondary (stratum 2), FreeBsd 4.3.
Service Area: Brazil
Access Policy: Open access to NTP clients from NA-DF.
Contact: ntp-admin@na-df.rnp.br