**UNIVERSIDADE DE SÃO PAULO
ESCOLA DE ENGENHARIA DE SÃO CARLOS
Departamento de Engenharia Elétrica**

FABIO BRUSSOLO DE OLIVEIRA

**AVALIAÇÃO DO DESEMPENHO DE BUSCA DE IMAGENS POR CONTEÚDO EM
REDES DE COMPUTADORES: UMA PROPOSTA DE REENGENHARIA COM
APLICAÇÃO A IMAGENS MÉDICAS.**

**SÃO CARLOS
2012**

FABIO BRUSSOLO DE OLIVEIRA

**AVALIAÇÃO DO DESEMPENHO DE BUSCA DE IMAGENS POR CONTEÚDO EM
REDES DE COMPUTADORES: UMA PROPOSTA DE REENGENHARIA COM
APLICAÇÃO A IMAGENS MÉDICAS.**

Dissertação apresentada à Escola de Engenharia de São Carlos da Universidade de São Paulo, como parte dos requisitos para a obtenção do título de Mestre em Ciência, Programa de Engenharia Elétrica.

Área de Concentração: Processamento de Sinais e Instrumentação.

Orientador:
Prof. Dr. Adilson Gonzaga

SÃO CARLOS
2012

Ficha Catalográfica

Ficha catalográfica preparada pela Seção de Tratamento
da Informação do Serviço de Biblioteca – EESC/USP

048a	<p>Oliveira, Fabio Brussolo de</p> <p>Avaliação do desempenho de busca de imagens por conteúdo em redes de computadores: uma proposta de reengenharia com aplicação a imagens médicas. / Fabio Brussolo de Oliveira ; orientador Adilson Gonzaga. São Carlos, 2012.</p> <p>Dissertação (Mestrado - Programa de Pós-Graduação em Engenharia Elétrica e Área de Concentração em Processamento de Sinais e Instrumentação)-- Escola de Engenharia de São Carlos da Universidade de São Paulo, 2012.</p> <p>1. Sinais. 2. CBIR. 3. Rede de computadores. 4. Métricas de rede. 5. API. 6. Java. 7. Internet. Título.</p>
------	---

FOLHA DE JULGAMENTO

Candidato: Bacharel **FABIO BRUSSOLO DE OLIVEIRA.**

Título da dissertação: “Avaliação do desempenho de busca de imagens por conteúdo em redes de computadores: uma proposta de reengenharia com aplicação a imagens médicas”.

Data da defesa: 27/02/2012

Comissão Julgadora:

Resultado:

Prof. Associado **Adilson Gonzaga (Orientador)**
(Escola de Engenharia de São Carlos/EESC)

APROVADO

Prof. Associado **Evandro Luís Linhari Rodrigues**
(Escola de Engenharia de São Carlos/EESC)

APROVADO

Prof. Dr. **Walter Aloisio Santana**
(Faculdade de Tecnologia/ campus de Carapicuíba)

Aprovado

Coordenador do Programa de Pós-Graduação em Engenharia Elétrica:
Prof. Titular **Denis Vinicius Coury**

Presidente da Comissão de Pós-Graduação:
Prof. Associado **Paulo Cesar Lima Segantine**

*Dedico este trabalho aos meus pais, Antonio e Madali,
ao meu filho Caio e esposa Aldi,
ao Professor Adilson, pela orientação, pela oportunidade
oferecida para meu ingresso e também pela confiança,
a Magali pelas ajudas,
a todos que estiveram envolvidos neste trabalho,
e especialmente a Carol, pelo apoio intelectual,
pela paciência e apoio técnico, sem o qual estaria perdido...*

RESUMO

O objetivo deste trabalho é avaliar a recuperação de imagem por conteúdo (CBIR - *Content-based Image Retrieval*) em uma rede de computadores, estabelecendo-se métricas de controle que otimizem a utilização da rede e ao mesmo tempo garanta melhor qualidade na resposta. Prevê-se a recuperação de imagens baseada em consulta por similaridade, combinando-se o extrator de características com a função de distância. O desenvolvimento é realizado em uma linguagem de programação independente da plataforma e de âmbito de internet, utilizado Java, desenvolvendo-se uma API (*Application Programming Interface*), visando, em especial, a reutilização de código, o que implica na diminuição do tempo de desenvolvimento. O trabalho demonstra a utilização de estruturas de indexação sequencial, comparada com a estrutura de indexação de árvore “*Slim-tree*”. A principal contribuição deste trabalho é a análise e a utilização de métricas na rede de computadores em um projeto de CBIR.

Palavras-chave: CBIR. Rede de Computadores. Métricas de rede. API. Java. Internet.

ABSTRACT

The objective of this study is to evaluate the Content-based Image Retrieval (CBIR) in a computer network, establishing control metrics that optimize the use of network while ensuring better quality in the response. It is expected that images are going to be recovered based on queries that are similar, combining the features extractor with the distance function. The development environment is an independent language of the platform and of the Internet scope, using Java, that has developed an Application Programming Interfaces (API), aiming primarily the code reuse, which implies a decrease in the development time. This thesis demonstrates the use of sequential indexing structures compared to the tree index structure "*Slim-tree*". The main contribution of this thesis is the analysis and utilization of metrics in a computer network project in CBIR.

Keywords: CBIR. Network Computers. Metrics Network. API. Java. Internet.

LISTA DE FIGURAS

Figura 2. 1 - Imagens com diferenças na resolução de níveis de cinza.	19
Figura 2. 2 - Imagem Matricial.....	20
Figura 2. 3 - Imagem Vetorial	20
Figura 2. 4 - Fase de estabelecimento de Conexão.....	23
Figura 2. 5 - Sistema Cliente/Servidor.....	25
Figura 2. 6 - Arquitetura Cliente/Servidor como Servidor de Banco de Dados.....	26
Figura 2. 7 - Integração entre os processos clientes e servidores	27
Figura 2. 8 - O modelo OSI	28
Figura 2. 9 - Comparação entre as 7 camadas do OSI e as 4 camadas do TCP/IP ..	30
Figura 2. 10 - Encapsulamento passando pelas camadas do TCP/IP	32
Figura 3. 1 - Métrica NOC	39
Figura 3. 2 - Métrica WMC	39
Figura 3. 3 - Métrica DIT.....	40
Figura 3. 4 - Método CBO	41
Figura 3. 5 - Exemplo da métrica LCOM	41
Figura 3. 6 - Exemplo da métrica RFC	42
Figura 3. 7 - Exemplo de uma estrutura de Slim-tree.....	45
Figura 4. 1 – Recuperação de imagens baseada em conteúdo	48
Figura 4. 2 - Arquitetura de um sistema de recuperação de imagens por conteúdo ..	48
Figura 4. 3 - Representação da Imagem por Histograma de intensidade de pixels ..	50
Figura 4. 4 - Classificação das técnicas de formas	51
Figura 4. 5 – A) imagem microscópica de um supercondutor; B) colesterol humano; C) um microprocessador	52
Figura 4. 6 - Extração de Características de Imagens	53
Figura 4. 7 - Modelo Esquemático de Análise em multirresolução.....	56
Figura 4. 8 – Transformada Wavelet Haar com dois níveis de resolução	56
Figura 4. 9 - Imagem usando Transformada Wavelet Haar.....	57
Figura 5. 1 - Esquema da Extração do Vetor em Java com API	60
Figura 5. 2 - Classe de Coluna Vertebral.....	61
Figura 5. 3 - Classe de Colo Lombar.....	61
Figura 5. 4 - Classe de Crânio.....	61
Figura 5. 5 - Classe de Tórax	61
Figura 5. 6 - Classe de Abdômen	61
Figura 5. 7 - Imagem de utilizada no teste de transmissão	63
Figura 5. 8 - Parte do código que faz a transformada Wavelet Haar.....	65
Figura 5. 9 - Função e_da().....	65
Figura 5. 10 - Função mv_da(M)	65
Figura 5. 11 - SuperClasse da Reengenharia	66
Figura 5. 12 - Processo de multirresolução e a transformada	67

Figura 6. 1 - Tempo total de upload de imagens para cada usuário	69
Figura 6. 2- Variação no atraso do Download e Upload.....	71
Figura 6. 3 - Latência dos usuários durante os testes.....	72
Figura 6. 4 - Tempo em milissegundos de upload das imagens	73
Figura 6. 5 - Download da 1ª imagem do usuário em ms.....	74
Figura 6. 6 - Download da 2ª imagem do usuário em ms.....	74
Figura 6. 7 - Download da 3ª imagem do usuário em ms.....	75
Figura 6. 8 - Download da 4ª imagem do usuário em ms.....	75
Figura 6. 9 - Desempenho de cada base de dados na reengenharia.....	76
Figura 6. 10 - Comparação entre os tipos de Sistemas	77
Figura 6. 11 - Número de serviços na Classe ConnectionManager	77
Figura 6. 12 - Número de serviços na Classe na HaarDemo	78
Figura 6. 13 - Número de serviços na Classe na ImagemHash	78
Figura 6. 14 - número máximo de superclasses (DIT)	79
Figura 6. 15 - Número de subclasses abaixo da superclasse	79
Figura 6. 16 - Nível de acoplamento entre as classes da aplicação.....	80
Figura 6. 17 - Resultado da Métrica RFC.....	80
Figura 7. 1 - Comparação do Jitter de Upload e Download com a média de Jitter....	84
Figura 8. 1 - Acesso a página de HTML.....	115
Figura 8. 2 - Como funcionam o acesso a programas CGI	116
Figura 8. 3- Procedimentos de execução de instruções utilizando Java.....	119

LISTAS DE TABELAS

Tabela 2. 1 - Formatos de padrões de imagens.....	22
Tabela 2. 2 - As camadas OSI e suas funções	28
Tabela 2. 3 - As camadas TCP/IP suas funções e seus protocolos	30
Tabela 3. 1 - Tempo aceitável de latência.....	36
Tabela 3. 2 -Resumo das Métricas CK.....	42
Tabela 7. 1 - Tempo médio do download de 20 imagens.....	86

LISTA DE SIGLAS

API	<i>Application Programming Interface</i>
ARP	<i>Address Resolution Protocol</i>
CBIR	<i>Content-basedImage Retrieval</i>
DNS	<i>Domain Name System</i>
FTP	<i>File Transfer Protocol</i>
HTTP	<i>HyperText Transfer Protocol</i>
IETF	<i>Internet Engineering Task Force</i>
ISO/OSI	<i>International Organization for Standardization/ Open System Inter-connection</i>
ICMP	<i>Internet Control Message Protocol</i>
JPEG	<i>Joint Photographics Experts Group</i>
PACS	<i>Picture Archiving and Communication System</i>
RARP	<i>Reverse Address Resolution Protocol</i>
RTP	<i>Real-time Transport Protocol</i>
SGBD	<i>Sistema Gerencial de Banco de Dados</i>
SQL	<i>Structured Query Language</i>
SMTP	<i>Simple Mail Transfer Protocol</i>
TCP/IP	<i>Transmission Control Protocol/ Internet Protocol</i>
TFTP	<i>Trivial File Transfer Protocol</i>
UDP	<i>User Datagram Protocol</i>

SUMÁRIO

Capítulo 1 – Introdução	15
1.1 Considerações Gerais.....	15
1.2 Motivação e Objetivos.....	16
1.3 Organização do Trabalho e Apresentação.....	17
Capítulo 2 – Teoria de comunicação de imagens médicas em redes	19
2.1 Imagem Digital	19
2.2 Compressão de imagens	21
2.3 O Protocolo TCP/IP.....	22
2.4 Modelo Cliente-Servidor.....	24
2.5 Camadas da Arquitetura Cliente-Servidor.....	26
2.6 Modelo ISO/OSI	27
2.7 Modelo TCP/IP	29
2.8 Encapsulamento	31
2.9 Considerações sobre este capítulo	33
Capítulo 3 – Métricas de Desempenho	35
3.1 Introdução	35
3.2 Métricas de rede	35
3.2.1 – Atraso (latência RTT).....	35
3.2.2 – Perda de pacotes.....	36
3.2.3 – Download/Upload.....	37
3.2.4 – Disponibilidade.....	37
3.2.5 – Capacidade.....	37
3.3 Métricas de Softwares.....	38
3.3.1 – Métricas de Orientação a Objetos.....	38
3.4 Métricas de Banco de Dados	42
3.4.1 Definições	43
3.4.2 Estrutura de Indexação Ordenada.....	43
3.4.3 Estrutura de Indexação da Slim-tree.....	44
3.5 Considerações sobre este capítulo	45
Capítulo 4 – Recuperação Baseada em Conteúdos	47
4.1 Introdução	47

4.2 Recuperação de imagens por conteúdo	47
4.3 Arquitetura de um Sistema CBIR	48
4.4 Atributos da Imagem	49
4.4.1 Cor	49
4.4.2 Forma	50
4.4.3 Textura	51
4.5 Extração de Características	52
4.6 A Transformada Wavelet	54
4.6.1 Transformada Wavelet Haar	54
4.7 Análise em Multirresolução	55
4.8 Considerações sobre este capítulo	57
Capítulo 5 – Metodologia	59
5.1 Introdução	59
5.2 Metodologia e Materiais	59
5.3 Configuração do Setup de Testes	62
5.4 Reengenharia do projeto	64
5.5 Considerações sobre este capítulo	68
Capítulo 6 – Resultados	69
6.1 Introdução	69
6.2 Experimentos de Desempenho de Tipos de Imagens na Rede	69
6.3 JITTER	70
6.4 Latência (atraso)	71
6.5 Perda de Pacotes	72
6.6 Download e Upload	73
6.7 Métricas de Reengenharia	76
6.8 Métricas de Orientação a Objetos	77
6.9 Considerações sobre este capítulo	81
Capítulo 7 – Conclusões Gerais	83
7.1 Considerações Gerais e Contribuições	83
7.1.1 Estabelecimento de métricas para desempenho de recuperação de imagens médicas baseadas em conteúdo em redes de computadores.	83
7.1.1 Métricas de software, reengenharia do sistema, troca do MatLab pelo Java (desenvolvimento de API).	85

7.1.3 Alteração da estrutura de armazenamento dos vetores de características, e Benchmark entre as bases de dados.....	86
7.2 Trabalhos Futuros	87
Referências	89
Apêndice A – Código na Linguagem MatLab Server.....	92
Apêndice B – Código na Linguagem JAVA (Reengenharia)	95
Apêndice C – Código na Linguagem JAVA– Conversão de DICOM para JPEG	105
Apêndice D – Planilhas utilizadas para a criação das estatísticas	106
Apêndice E – Planilhas utilizadas para a criação das estatísticas	110
Anexo A – A forma de Desenvolvimento de Aplicativos para Internet.....	115
Anexo B – Particularidades da linguagem JAVA.....	118
Anexo C – SIMET de cada usuário	120

Capítulo 1 – Introdução

1.1 Considerações Gerais

A aquisição de imagens digitais cresce em ambientes hospitalares vertiginosamente, oferecendo panoramas para o desenvolvimento de técnicas de redução de espaço físico, de memória requerida e rápida recuperação em bases de dados descentralizados, com alta taxa de transmissão de imagens em redes de computadores. Por essa razão, o desenvolvimento de métodos que possibilitem melhor armazenamento, recuperação e especialmente a transmissão de imagens em redes de computadores tem se tornado de grande interesse.

O processo de integração de estações de trabalho com as visualizações distribuídas, base de dados em tempo real, sistemas que gerenciam prontuários e imagens permitem que as informações sejam compartilhadas por todos os profissionais de saúde, tanto no interior desses centros, como em locais remotos, podendo ser acessadas em diversos locais em um mesmo instante.

Os trabalhos com imagens médicas podem ser divididos em quatro categorias: geração de imagens, análise de imagens, gerenciamento de imagens e gerenciamento de informações.

A partir da década de 1980, o gerenciamento de imagens médicas segue uma padronização chamada PACS (*Picture Archiving and Communication System*) ou sistemas de arquivamento e comunicação de imagens (MARSH, 1997). Estes sistemas permitiram o armazenamento e recuperação das imagens médicas em uma rede de computadores que propiciaram uma grande coleção de imagens armazenadas em banco de dados. Em virtude do grande volume de imagens, foram propostos alguns mecanismos que auxiliam na recuperação de imagens baseado em conteúdo - CBIR (*Content-based Image Retrieval*), que buscam a imagem por similaridade usando características visuais como cor, textura e formas (TORRES, 2006).

A busca de imagens similares é um processo que retorna uma consulta rápida e eficaz ao usuário. Entretanto, se o sistema de recuperação não for otimizado, inviabiliza o processo inteiro. Nesse tratamento da informação em tempo real estão relacionados a velocidade de transmissão do equipamento, a abstração do usuário para o sistema e as métricas de desempenho.

1.2 Motivação e Objetivos

A proposta deste trabalho nasceu em decorrência de algumas questões que foram suscitadas na finalização do trabalho de mestrado de Ana Carolina Nicolosi da Rocha Gracioso (GRACIOSO, 2008), com o tema de “Avaliação do desempenho de recuperação de imagens médicas baseadas em conteúdo em redes de computadores”, que discorreu sobre a aceitabilidade dos tempos de resposta em uma rede (métricas), o armazenamento das informações e o uso do *software MatLab WebServer*.

O Projeto “Avaliação do desempenho de recuperação de imagens médicas baseadas em conteúdo em redes de computadores”, finalizou com a conclusão de um programa de recuperação de imagens baseada em conteúdo, aplicado na área das ciências médicas, sobre uma infraestrutura de rede de computadores. Nesse modelo, existe uma base contendo todos os vetores de características de cada imagem. Cada vetor possui 16 campos numéricos. Ao ser inserida uma imagem para consulta, o sistema extrai o vetor de características e executa uma comparação deste vetor com os vetores armazenados (usando distância Euclidiana). E será devolvido ao usuário, um grupo de imagens de menor distância.

No trabalho supracitado, a aceitabilidade dos tempos de resposta (Cliente/Servidor) e da interface foram medidas de maneira subjetiva, não existindo uma ferramenta que mensurasse o desempenho do sistema em uma rede de alta velocidade. Outro fator que é de extrema relevância é o armazenamento dos vetores de características, que neste caso, foram armazenados em uma planilha, decorrendo assim um grau de lentidão. A utilização do *MatLab Web Server* permitiu que fossem criadas aplicações *MatLab*, possibilitando o uso dos recursos da internet para enviar dados para *Mat-Lab*, para o cálculo e exibir os resultados em um navegador.

O principal objetivo deste trabalho é propor uma reengenharia no sistema adotado anteriormente de forma incremental, capturas de requisitos funcionais e não funcionais que auxiliará na utilização dos indicadores de qualidade deste modelo arquitetural para um modelo novo, ou seja, trocar o código desenvolvido na linguagem do *MatLab Web Server* para linguagem Java, e desenvolver uma API (*Application Programming Interface*) para ser integrada a qualquer linguagem de programação voltada a internet para o retorno do vetor de característica e imagens similares.

Além disso, reunir a fundamentação da teoria das métricas para o acompanhamento do usuário na rede (demonstrar métricas de CBIR na rede); alterar a estrutura do armazenamento dos vetores, armazenando-se as informações em um sistema gerencial de banco de dados (SGBD), utilizando uma indexação adequada e garantir a segurança e a integridade da informação; desenvolver uma ferramenta de busca por similaridades entre imagens, utilizando a rede, comparando com o projeto anterior (GRACIOSO, 2008) e quantificar as características que permitem a usabilidade do sistema com maior eficácia.

1.3 Organização do Trabalho e Apresentação

Este primeiro capítulo discorre a respeito de conceitos gerais relacionados ao tema deste trabalho e aos objetivos a serem alcançados.

O Capítulo 2 introduz os conceitos relacionados a imagens digitais, a compressão de imagem JPEG, a importância da imagem no formato JPEG em uma rede, as camadas de transmissão em uma rede, modelo ISO/ OSI, TCP/IP e encapsulamento de pacotes.

No Capítulo 3 há o detalhamento da teoria das métricas, sendo definidas as métricas de desempenho de um sistema em rede, as métricas de sistemas orientadas a objetos e finaliza nas métricas aplicadas ao banco de dados.

O Capítulo 4 descreve a recuperação baseada por conteúdo, explicitando-se a arquitetura CBIR e a extração de características. É apresentada também, a Transformada *Wavelet*, particularizando a transformada *Wavelet Haar*, o ambiente multirresolução e as similaridades semânticas das imagens.

No Capítulo 5 está descrita a metodologia utilizada neste trabalho, bem como a descrição dos ambientes envolvidos nos cenários propostos. Discute-se a recuperação de imagens médicas em rede utilizando API (*Application Programming Interface*), a base de dados gerencial indexada e a aplicação dos conceitos de métricas.

O Capítulo 6 apresenta os resultados dos experimentos realizados sobre a avaliação de desempenho na busca de imagens por conteúdo em redes de computadores.

As conclusões gerais e linhas de futuras pesquisas são delineadas no Capítulo 7 desta dissertação.

O material adicional de auxílio à leitura e compreensão deste estudo encontra-se nos Apêndices e Anexos no final desta dissertação. O Apêndice A mostra o código original desenvolvido na linguagem *MatLab* (GRACIOSO, 2008). O Apêndice B mostra o resultado da reengenharia do Código desenvolvido em Java. O Apêndice C complementa a reengenharia e acresce a conversão do formato DICOM (*Digital Imaging and Communications in Medicine*) para JPEG (*Joint Photographics Experts Group*). Os Apêndices D e E mostram as tabelas que serviram como base para a criação das estatísticas e dos gráficos deste estudo. O Anexo A evidencia a forma de desenvolvimento de sistemas para a Internet e o Anexo B particulariza o desenvolvimento na Linguagem Java. No Anexo C, são demonstradas as telas do Sismet dos usuários nos testes.

Capítulo 2 – Teoria de comunicação de imagens médicas em redes

2.1 Imagem Digital

Uma imagem digital é o resultado de discretização de uma função $I = (x, y)$, sendo armazenada como um arranjo de duas dimensões $I = (m, n)$, sendo $m = 0, 1, \dots, N_x - 1$ e $n = 0, 1, \dots, N_y - 1$, $N_x \times N_y$ (para escala de cinza). Cada elemento descrito na matriz (m, n) , corresponde a um pixel, cujo brilho ou intensidade é $I = (m, n)$, que é quantificada em níveis discretos de cinza (N_g), ou seja, a profundidade da imagem. Portanto, a sequência $I = (m, n)$ estará entre $0, 1, \dots, N_g - 1$, ao passo que a profundidade será sempre uma potência de dois (THEODORIDIS, 1999).

A Figura 2.1 ilustra quatro diferentes profundidades em uma imagem digital. Pode-se observar que quanto maior for a resolução dos níveis de cinza (N_g), mais detalhes poderão ser visíveis na imagem.

Figura 2. 1 - Imagens com diferenças na resolução de níveis de cinza.

A) $N_g=16$; B) $N_g=8$; C) $N_g=4$; D) $N_g=2$

Extraído de: THEODORIDIS, 1999

A imagem digital possui algumas características fundamentais como o número de cores e a resolução, e pode ser processada, representada e armazenada no formato de uma imagem matricial ou vetorial.

A imagem matricial é formada por uma série ordenada de pixels dispostos continuamente em linha e colunas, chamada de matriz de pontos, e cada pixel tem suas características definidas com cor e brilho (FONSECA, 1998). A Figura 2.2 ilustra uma imagem com uma reta e sua ampliação em uma imagem matricial. Na figura A, mostra a reta no tamanho original e na figura B, mostra a seleção da imagem ampliada.

Figura 2. 2 - Imagem Matricial

Extraído de: FONSECA, 1998

Já a imagem vetorial é formada por um conjunto de instruções matemáticas, cuja imagem é gerada após o processamento das instruções. Pode-se alterar o tamanho da imagem sem perder definição, pois serão calculadas as instruções novamente para o novo tamanho (FONSECA, 1998). A Figura 2.3 ilustra uma imagem com uma reta e sua ampliação. É possível perceber que não há perda de definição na reta, pois para realizar a ampliação foram recalculadas. Nesta Figura 2.3, a opção A mostra a reta do tamanho original e na opção B, mostra a seleção da imagem ampliada.

Figura 2. 3 - Imagem Vetorial

Extraído de: FONSECA, 1998

2.2 Compressão de imagens

Em razão da popularização das redes e o compartilhamento de informações e arquivos digitais, necessitou-se da utilização de um padrão de imagem que compreendesse uma alta taxa de compressão, com a maior fidelidade à imagem original. O formato *Joint Photographics Experts Group* (JPEG) foi aprovado em 1992 pela Organização Internacional para Padronização (SILVA, 2009).

Os padrões de compressão de imagens em tons de cinza ou coloridas, estáticas ou dinâmicas, em muitos casos, foram desenvolvidos e sancionados pela *International Standards Organization* (ISO) e pelo *Comité Consultatif International des Téléphonique et Télégraphique* (CCITT) atualmente denominado *International Telecommunication Union* (ITU-T) (TORRES, 2006).

O padrão JPEG foi desenvolvido para comprimir imagens paradas em tons contínuos de cinza ou de cor, de cenas do mundo real e imagens naturais. O JPEG utiliza um algoritmo de compactação que se baseia na capacidade do olho humano. No entanto, mesmo sabendo-se que arquivos em JPEG podem trabalhar com até 16,8 milhões de cores, o olho humano não é capaz de enxergar todas elas de uma vez. Assim, é possível tirar uma série de informações que representam cores em imagens e manter apenas aquelas visíveis ao olho humano. Em outras palavras, o formato JPEG “tira” da imagem aquilo que os humanos não conseguem ver. Esse processo é conhecido como compressão. Isso permite que imagens realistas sejam criadas e ao mesmo tempo esses arquivos ficam compactos. Portanto, os arquivos podem ter diferentes níveis de compressão nesse formato, permitindo *downloads* ou *uploads* mais curtos para arquivos de imagem grandes. Além disso, ele parece manter a qualidade da maioria das fotos quase intacta (WANG, 2001).

O padrão JPEG apresenta quatro modos de operação: JPEG com perdas ou *Joint Photographic Experts Group - Discrete Cosine Transform* (JPEG - DCT), JPEG sem perdas ou *Joint Photographic Experts Group - Differential Pulse Coding Modulation* (JPEG - DPCM), modo Progressivo e o modo Hierárquico. Em uma transmissão de rede, quando há referência ao padrão JPEG, trata-se do JPEG – DCT, pois os graus de qualidade e de velocidade de descompressão podem ser variados de acordo com os parâmetros da compressão.

A compressão do padrão JPEG (imagem do tipo vetorial) é ajustável, balanceando a qualidade da imagem com a compressão. Pode-se comprimir até dez vezes o tamanho original sem a perda da imagem. Esse processo pode ser realizado sem perdas (*loss/less*) ou com perdas (*lossy*).

Os Algoritmos sem perdas (*loss/less*) eliminam somente as informações redundantes, possibilitando, assim, a reconstrução total da imagem original após o processo de descompressão. Já os algoritmos de compressão com perdas eliminam, além das informações redundantes, as informações irrelevantes, possibilitando somente a reconstrução aproximada da imagem original após a descompressão (FONSECA, 1998).

Em compressão de imagens, dois fatores são de extrema importância: velocidade de processamento e a taxa de compressão. Um esquema típico de compressão sem perda para imagens em tons de cinza obtém taxa de compressão da ordem de 50%, possibilitando reaver a imagem original com base na versão comprimida.

A Tabela 2.1 compara um arquivo com dimensão de 1943 x 1702 pixels com 24 *bits* (16 milhões de cores) nos diversos formatos de padrões de imagens. O que deve ser ressaltado é o tamanho final do arquivo para a transmissão dessa imagem via rede de computadores, tornando-se demorada (FONSECA, 1998).

Tabela 2. 1 - Formatos de padrões de imagens

Formato do Arquivo	Tamanho do Arquivo	Observações
<i>Tag Image File Format</i> (TIFF)	9,9 MB	Sem compressão
<i>Lempel-Ziv-Welch</i> (TIFF – LZW)	8,4 MB	Com compressão LZW
<i>Portable Network Graphics</i> (PNG)	6,5 MB	Com compressão PNG
<i>Joint Photographic Experts Group</i> (JPEG)	1,0 MB	JPEG - DCT
<i>Bitmap</i> (BMP)	9,9 MB	Sem compressão

Extraído de: FONSECA, 1998

2.3 O Protocolo TCP/IP

Um protocolo confiável é aquele que possui recursos de correção de erros, de controle de fluxo e de retransmissão incorporada em si. Protocolos não confiáveis dependem de protocolos de camadas mais altas para fornecer confiabilidade. Um exemplo bem simples para a compreensão de protocolo confiável é o envio de uma

carta registrada, pois existe uma pessoa garantindo o seu recebimento, ao passo que o não confiável é como se a carta fosse colocada debaixo da porta e o seu recebimento não está garantido (CHAPPELL, 2002).

Um protocolo orientado à conexão possui alguns métodos para estabelecer sua conexão, como: controle de fluxo, controle de erros e término de sessão. Ainda garantirá que os pacotes estejam em ordem e fiscaliza os contadores de um *timeout*. Caso haja o erro, o protocolo solicita a retransmissão do pacote perdido. No protocolo orientado à conexão, é estabelecida uma associação lógica entre as entidades de protocolos (CHAPPELL, 2002). Na Figura 2.4 é mostrado um esquema, no qual a máquina receptora envia avisos de recepção da comunicação, assim a máquina emissora é fiadora da validade dos dados que envia. Os dados são assim enviados sob a forma de fluxo:

Figura 2. 4 - Fase de estabelecimento de Conexão

Extraído de: CHAPPELL, 2002

O TCP/IP (*Transmission Control Protocol/Internet Protocol*) é um protocolo orientado à conexão. A sequência para o estabelecimento de conexão TCP é chamada de *Three-way handshake* (*Handshake de três vias*):

- O *host* (qualquer máquina ou computador conectado a uma rede) que quer iniciar a sessão envia um pacote de sincronização (SYN) TCP;
- O *host* receptor confirma o SYN e, no mesmo pacote, envia o seu próprio SYN;
- O *host* original confirma o recebimento do SYN do outro *host*.

O TCP foi desenvolvido para a comunicação entre equipamentos da internet. Basicamente esse protocolo tem a função de controlar o que é transmitido de uma estação (*host*) para outra, bem como o tratamento do endereçamento dos pacotes, a fragmentação dos dados e a verificação de erros (CHAPPELL, 2002).

2.4 Modelo Cliente-Servidor

O modelo cliente-servidor é compreendido como o processamento de diversas máquinas submetidas por diversos nós a um único ponto central para que possam ser processadas essas requisições. Os servidores recebem as requisições, processam e retornam o resultado esperado. Tipicamente essas aplicações estão divididas entre um provedor de acesso, uma central de dados e numerosos clientes contendo uma interface gráfica para usuários acessarem e manipularem dados (MCKIE, 1997).

O modelo Cliente-servidor é definido como uma relação entre processos que estão ocorrendo em máquinas diferentes. O processo servidor é o fornecimento de serviços. O cliente é o consumidor de serviços. Fundamentalmente uma arquitetura cliente-servidor implementa uma separação lógica de funções baseada no conceito de serviço (HULQUIST, 1997). Este modelo é importante, pois tem sido a base das aplicações desenvolvidas para a internet.

Em uma definição para a arquitetura Cliente-Servidor é necessária a existência de uma plataforma base para que as aplicações, as quais um ou mais Clientes e um ou mais Servidores, juntamente com o Sistema Operacional e o Sistema Operacional de Rede, executem um processamento distribuído.

Algumas características do Cliente e Servidor são descritas a seguir (HULQUIST, 2007):

- Cliente:
 - Também denominado de “*front-end*” e “*WorkStation*”, é um processo que interage com o usuário por meio de uma interface gráfica ou não, permitindo consultas ou comandos para recuperação de dados e análise e representando o meio pela qual os resultados são apresentados.
 - Além disso, apresenta algumas características distintas:
 - É o processo ativo na relação Cliente-Servidor.
 - Inicia e termina as conversações com os servidores, solicitando serviços distribuídos.
 - Não se comunica com outros clientes.
 - Torna a rede transparente ao usuário.

- Servidor:
 - Denominado Servidor ou “*back-end*”, fornece um determinado serviço que fica disponível para todo cliente que o necessita. A natureza e escopo do serviço são definidos pelo objetivo da aplicação Cliente-Servidor.
 - É o processo reativo na relação Cliente-Servidor.
 - Possui uma execução contínua.
 - Recebe e responde às solicitações dos clientes.
 - Não se comunica com outros servidores enquanto estiver fazendo o papel de servidor.
 - Presta serviços distribuídos.
 - Atende a diversos clientes simultaneamente.

Pode-se encontrar dentro de uma arquitetura Cliente-Servidor a interoperabilidade das estações clientes e servidores entre as redes de computadores, a escalabilidade da arquitetura visando ao crescimento e à redução dos elementos constituintes, a adaptabilidade de novas tecnologias desenvolvidas, a *performance* do *hardware* envolvido na arquitetura, a portabilidade entre as diversas estações que compõem a arquitetura e a segurança dos dados e processos. Ainda pode-se destacar: confiabilidade, matriz de computadores agregando capacidade de processamento, tanto o cliente como o servidor possuem ambientes operacionais individuais (HULQUIST, 2007).

A Figura 2.5 mostra uma arquitetura Cliente-Servidor genérica em que divide uma aplicação em processos, e são executados em diferentes máquinas conectadas a uma Rede de Computadores formando um único sistema.

Figura 2. 5 - Sistema Cliente/Servidor

Extraído de: HULQUIST, 2007

2.5 Camadas da Arquitetura Cliente-Servidor

A arquitetura Cliente-Servidor é dividida em três camadas básicas (COULOURIS, 2007):

- A camada de Aplicação consiste dos processos da aplicação, entre eles, os processos cliente e servidor.
- A camada de Serviços de Sistemas compreende o Sistema Operacional (SO) e o Sistema Operacional de Rede (SOR), destinando-se ao controle do *hardware*.
- A camada de *hardware*, na qual estão localizados os periféricos ligados aos clientes e servidores.

Existem vários sistemas que podem ser baseados na estrutura Cliente-Servidor. O uso mais frequente são as aplicações de banco de dados usando processos *Structured Query Language* (SQL) de *front-end*, para acessar remotamente, as bases de dados. A Figura 2.6 mostra uma estrutura baseada num servidor de banco de dados. Esta estrutura ocasiona um maior fluxo de informações na rede, uma vez que todo o arquivo será transferido para o Cliente para então ser trabalhado.

Figura 2. 6 - Arquitetura Cliente/Servidor como Servidor de Banco de Dados

Extraído de: COULOURIS, 2007

A Figura 2.6 consiste na utilização de um servidor de banco de dados dedicado, podendo coexistir normalmente com o servidor de arquivos. Nesse momento, com um servidor de banco de dados exclusivo, o fluxo de informações trafegadas na rede diminui, já que, somente a resposta da consulta é retornada ao cliente, ao invés de transferir todo o arquivo como era feito anteriormente (COULOURIS, 2007).

O Cliente, nesse tipo de estrutura é o usuário, passando a ter uma visão da aplicação como se as partes, cliente e servidor, fossem algo único. A Figura 2.7 exemplifica essa visão.

Figura 2. 7 - Integração entre os processos clientes e servidores

Extraído de: COULOURIS, 2007

Este tipo de estrutura favorece o aumento da *performance* da rede de comunicação, possibilitando assim, um maior número de ligações simultâneas de diversos clientes com diversos servidores.

2.6 Modelo ISO/OSI

Para facilitar o desenvolvimento de comunicação entre equipamentos, estabeleceu-se como critério o modelo *International Organization for Standardization/ Open System Interconnection* (ISO/OSI) como um conjunto de padrões ou convenções para a conexão de computadores heterogêneos (CHAPPELL, 2002).

O modelo ISO/OSI é um conjunto de sete camadas sobrepostas, em que cada camada executa funções necessárias para se comunicar com a camada anterior ou posterior de acordo com o fluxo de informação. O modelo OSI é mostrado na Figura 2.8.

Figura 2. 8 - O modelo OSI

Extraído de: CHAPPELL, 2002

Na Tabela 2.2 são descritas as funções de cada camada do modelo OSI:

Tabela 2. 2 - As camadas OSI e suas funções

N.º	Camada	Especificação
7	Aplicação	Especifica a interface com o usuário e a interface com programas aplicativos e refere-se ao significado dos dados, gerenciamento de tarefas.
6	Apresentação	Aceita os dados da camada de aplicativos e negocia a sintaxe, reapresentação, compactação e criptografia dos dados com sistemas associados.
5	Sessão	Acrescenta mecanismos de controle como pontos de verificação, terminações e reinicializações para estabelecer, manter e sincronizar a comunicação entre aplicativos.
4	Transporte	Fornece a responsabilidade fim-a-fim e garante a remessa de dados confiáveis usando mecanismos de confirmação, números de sequência e de controle de fluxo.
3	Rede	Especifica o roteamento de redes e comunicação entre os segmentos de rede.
2	Enlace de dados	Formata a informação em quadros (<i>frames</i>), detecção e recuperação de erros de transmissão e pela definição dos endereços de dispositivos físicos.

continua

continuação da Tabela 2.2

N.º	Camada	Especificação
1	Física	Define a conexão física entre o sistema computacional e a rede. Especifica o nível de tensão, o conector, dimensões físicas.

Extraído de: CHAPPELL, 2002

2.7 Modelo TCP/IP

O TCP é um protocolo orientado a conexões. Ele mantém um diálogo entre a origem e o destino enquanto empacota informações da camada de aplicação em unidades chamadas segmentos. O termo “orientado” a conexões não quer dizer que existe um circuito entre os computadores que se comunicam. Significa que segmentos da Camada 4 trafegam entre dois *hosts* para confirmar que a conexão existe logicamente durante um certo período (CHAPPELL, 2002).

O IP é o principal protocolo da Camada 3 no conjunto da internet. Além de roteamento de redes, o IP fornece fragmentação e remontagem de datagramas e relatórios de erros.

O *Transmission Control Protocol/Internet Protocol* (TCP/IP) é um conjunto de protocolos ou regras desenvolvidas para a cooperação entre computadores para que compartilhem recursos por intermédio de uma rede. Para ativar o TCP/IP em uma estação de trabalho, esta precisa ser configurada por meio das ferramentas do sistema operacional (CHAPPELL, 2002).

Os modelos OSI e TCP/IP possuem camadas que explicam como os dados são comunicados desde um computador para outro. Os modelos diferem no número e função das camadas. Entretanto, cada modelo pode ser usado para ajudar na descrição e fornecimento de detalhes sobre o fluxo de informação desde uma fonte até um destino.

A Figura 2.9 mostra o compartilhamento dos modelos OSI e TCP/IP, considerando-se as suas camadas.

Figura 2. 9 - Comparação entre as 7 camadas do OSI e as 4 camadas do TCP/IP

Extraído de: CHAPPELL, 2002

Na Tabela 2.3 são descritas as quatro camadas do Sistema TCP/IP

Tabela 2. 3 - As camadas TCP/IP suas funções e seus protocolos

N.º	Camada	Especificação	Protocolos
4	Aplicação	A camada de aplicação trata de questões de representação, codificação e controle de diálogo. Estabelece os protocolos de aplicativos TCP/IP.	HTTP, Telnet, FTP, TFTP, SNMP, DNS, SMTP.
3	Transporte	Trabalha com questões de qualidade de serviços de confiabilidade, controle de fluxo e correção de erros. Um de seus protocolos, o TCP fornece formas excelentes e flexíveis de se desenvolver comunicações de rede confiáveis com baixa taxa de erros e bom fluxo.	TCP, UDP, RTP.
2	Internet	O propósito da camada de Internet é dividir os segmentos TCP em pacotes e enviá-los de qualquer rede. Os pacotes chegam à rede de destino independente do caminho. O protocolo específico que governa essa camada é chamado	IP, ICMP, ARP, RARP.

continua

continuação da Tabela 2.3

N.º	Camada	Especificação	Protocolos
1	Interface de rede	<p><i>Internet Protocol (IP)</i>. A determinação do melhor caminho e a comutação de pacotes ocorrem nesta camada. Também ocorre o encapsulamento de dados em datagramas IP, que contém informações de endereço de origem e destino. Executa o roteamento da datagrama IP.</p> <p>É conhecida como a camada <i>host-para-rede</i>. Esta camada trata todos os componentes, tanto físico como lógico, que são necessários para fazer um <i>link</i> físico. Isso inclui os detalhes da tecnologia de redes, inclusive todos os detalhes nas camadas física e de enlace do OSI.</p>	Ethernet, Token Ring, FDDI, RS-232, V.35., X.25, retransmissão de quadros.

Extraído de: CHAPPELL, 2002

2.8 Encapsulamento

As comunicações entre os nós de uma rede começam em uma origem e são enviadas a um destino (*host* origem para *host* destino). As informações enviadas através da rede são conhecidas como dados ou pacotes de dados. Se um *host* A quer enviar dados para outro *host* B, então os dados devem ser empacotados por meio de um processo chamado encapsulamento (MCKIE, 1997).

O encapsulamento coloca as informações de protocolo necessárias antes que trafeguem pela rede. E, à medida que o pacote de dados desce pelas camadas do modelo OSI ou TCP/IP, ele recebe cabeçalhos, *trailers* e outras informações.

As redes efetuam cinco etapas de conversão para encapsular os dados (MCKIE, 1997):

1. Gerar os dados.
2. Empacotar os dados para transporte fim-a-fim.
3. Adicionar o endereço IP da rede ao cabeçalho.
4. Adicionar o cabeçalho e o *trailer* da camada de enlace de dados.
5. Converter em *bits* para transmissão.

Na Figura 2.10 pode-se observar um diagrama sobre o encapsulamento realizado pelo TCP/IP, exemplificando as camadas que são utilizadas pelo *host* quando o usuário acessa o navegador e o que é segmentado (MCKIE, 1997).

Figura 2. 10 - Encapsulamento passando pelas camadas do TCP/IP

Extraído de: MCKIE, 1997

Para exemplificar a Figura 2.10, quando o usuário digita um endereço no seu programa de internet, no navegador, o *software* monta uma instrução do tipo "Protocolo de aplicação HTTP". Esta instrução é referente a um pedido de arquivo feito a algum servidor *Web*. Então, o navegador passa essa instrução aos seus *drivers* de rede, mais precisamente à camada de Transporte. A camada de Transporte aloja essa instrução HTTP em sua área de dados e monta um cabeçalho com informações relevantes sobre o tipo de conexão que será mantida entre os dois computadores para a transmissão desse arquivo. Após especificar de que forma os computadores irão trocar essa informação, sua área de dados (instrução HTTP) acrescida destas informações sobre a conexão, são repassadas para a camada inferior, *Internet*, que aloca essa informação em sua área de dados. Essa camada acrescentará informações referentes ao endereço de destino (o servidor *Web*) que receberá esses dados, assim como preferências de rotas e dados de controle do pacote enquanto ele estiver a caminho do seu destino. Estando este cabeçalho pronto e anexado à área de dados, essa informação é agora passada ao *driver* da placa de rede, que montará um quadro de envio compatível com o meio físico ao qual está ligado.

Chegando ao destino, a placa de rede instalada nesse servidor abrirá esse pacote e repassará sua área de dados ao nível superior. A área de dados do nível físico será repassada justamente ao nível internet. Este nível, por sua vez, checa todos os dados de controle contidos no cabeçalho, como endereço destino e origem, e repassa sua área de dados (o pacote do nível transporte) ao nível superior, o Nível de Transporte. Este nível, ao receber seu pacote, checa os dados do cabeçalho quanto aos tipos de conexão. E, finalmente chega à área de dados (o comando HTTP) ao servidor *Web*, que responde pelo Nível de Aplicação. Ao receber o comando, o servidor *Web* busca o arquivo pedido pelo navegador e o envia, fazendo novamente todo o processo de empacotamento e transmissão até chegar ao seu computador.

2.9 Considerações sobre este capítulo

Para transmissão de imagens médicas em redes de computadores é fundamental que se utilize compactação de dados com baixa perda. A compactação garante maior velocidade de transmissão de uma imagem e baixa perda, e não interfere na qualidade visual da imagem.

Dentre as propostas de imagens existentes, optou-se por utilizar, neste trabalho, o formato JPEG, que permite compressão sem perda utilizando predição espacial para a recuperação integral da imagem.

A rede de computadores interligada à internet opera no modelo TCP/IP que encapsula os dados e transmite entre clientes e servidores. Esses conceitos foram avaliados neste capítulo, pois são a base para o desenvolvimento deste trabalho.

Capítulo 3 – Métricas de Desempenho

3.1 Introdução

Para avaliar o desempenho de sistemas, tem-se como primeiro desafio a utilização de métricas mais adequadas. O conhecimento da aplicação é um dos fatores críticos da escolha da técnica. Os estudos do desempenho desses ambientes são tarefas complexas, visto que em algumas fases o desempenho não depende da aplicação e sim de terceiros. Entretanto, neste capítulo serão conceituadas as principais métricas aplicadas no trabalho e estarão divididas em métricas de rede, métricas de *softwares* e métricas de banco de dados.

3.2 Métricas de rede

Em sistemas distribuídos, o desempenho da rede é essencial para a eficácia do funcionamento da aplicação. Para que se obtenha uma garantia de que as redes funcionem corretamente, faz-se necessária uma aplicação de tecnologia que permita um nível de tráfego ideal e confiável para dados e aplicações utilizadas nas transações. E um grande desafio de se medir esta eficácia é escolher a métrica ideal (podendo ser mais de uma), escolher o método e a ferramenta que garantam a coleta dos dados (COULOURIS, 2007).

A *Internet Engineering Task Force* (IETF), entidade internacional, cujo objetivo é a definição de métricas de desempenho relacionadas à qualidade e confiabilidade de aplicativos em rede, estabelece como padrão as seguintes métricas para o desempenho:

- Atraso (latência RTT).
- Perda de pacotes.
- *Download/Upload*.
- Disponibilidade.
- Capacidade.

3.2.1 – Atraso (latência RTT)

A latência em uma rede pode ser compreendida com o tempo decorrido após uma operação de envio e antes que os dados comecem a chegar ao seu destino

(COULOURIS, 2007). Ainda pode-se conceituar como o tempo necessário para transferir uma mensagem vazia. A latência inclui:

- O tempo em milissegundos (ms) da transmissão do primeiro *bit* de um conjunto de *bits* transmitidos para chegar ao seu destino.
- O atraso no acesso a rede, que aumentará consideravelmente quando a rede estiver carregada de acessos.
- O tempo do processamento gasto pelos serviços de comunicação entre o Sistema Operacional nos processos de envio e recepção do pacote, variando com a carga do processador.

As três medidas de um atraso são:

- a) Atraso em um único sentido: que se entende como o tempo que um pacote percorre um caminho da origem ao destino.
- b) Atraso ida e volta (RTT - *round-triptime*): tempo que leva para o pacote percorrer um caminho até o nó destino e retornar ao nó origem;
- c) Variação do atraso (*Jitter*), que é a diferença entre o intervalo da recepção de dois pacotes consecutivos e o intervalo das transmissões.
- d) Existência de uma estatística do aceitável da latência em uma rede.

A Tabela 3.1 mostra os tempos no caso da latência máxima, de acordo com a tecnologia utilizada.

Tabela 3.1 - Tempo aceitável de latência

Tecnologia	Latência em milissegundos
<i>Ethernet</i>	3ms
<i>Analog Modem</i>	100 – 200ms
<i>ISDN</i>	15 – 30ms
<i>DSL/ Cable</i>	10 – 60ms
<i>Stationary Satellite</i>	> 500ms

Extraído de: COULOURIS, 2007

3.2.2 – Perda de pacotes

Duas abordagens são relevantes quando é tratada a perda do pacote ou simplesmente o descarte (TANENBAUM, 2011):

- a) Taxa da perda: representa o número de pacotes perdidos em um total de pacotes enviados em um intervalo de tempo.
- b) Distribuição de perdas consecutivas, que indica o total de pacotes perdidos em sequência.

3.2.3 – *Download/Upload*

Download significa copiar arquivos de um computador qualquer que esteja conectado à rede para o computador local. O tempo do *download* está inteiramente associado à métrica vazão. Para se estimar um tempo de *download*, utiliza-se a função $D = \frac{S}{T}$, sendo S o Tamanho do Arquivo e T significa a vazão da conexão TCP. E *upload* significa a transferência de um arquivo de um computador local para um computador remoto (TANENBAUM, 2011).

3.2.4 – Disponibilidade

Pode-se definir como disponibilidade a quantidade de tempo que o serviço fica disponível em relação ao tempo total da observação (TANENBAUM, 2011). Essa métrica afeta diretamente no acesso e testes do sistema proposto neste trabalho.

3.2.5 – Capacidade

A capacidade está vinculada diretamente à habilidade do sistema para transmitir dados pela rede. Nesta instância, podem-se utilizar as seguintes métricas (TANENBAUM, 2011):

- a) Largura de banda disponível: é a quantidade de espaço não utilizada de uma rede ou de todas as redes ao longo do caminho do pacote.
- b) Vazão: muito conhecido pelo seu termo em inglês (*Throughput*), ao qual significa o total dos pacotes enviados a rede em um tempo específico.

3.3 Métricas de Softwares

De acordo com o SEI (*Software Engineering Institute*), Instituto de Engenharia de *Software*, define-se as métricas de *software* como um conjunto de medidas de um processo ou produto de *software*, em que um produto abstrato evolui para uma instrução inicial e finaliza em um *software* completo. São consideradas boas métricas para *software* aquelas que facilitam os modelos capazes de estimar um processo ou parâmetros de sistema. MILLS (1999) estabelece algumas métricas básicas:

- Simplicidade: o quanto a métrica é compreensível e aplicável.
- Validade: a métrica dimensiona realmente o que foi proposto.
- Objetividade: as métricas podem ser classificadas conforme a sua objetividade, ou seja, a aplicabilidade.

No trabalho de GRACIOSO (2008), não é desenvolvida uma métrica objetiva e sim subjetiva. Entende-se por métrica objetiva aquelas que são obtidas por meio de regras muito bem definidas, tendo os mesmos resultados independentes do tempo a ser aplicado.

3.3.1 – Métricas de Orientação a Objetos

Embora exista farta literatura sobre métricas de *softwares*, a maioria não apresenta detalhes relativos à qualidade, no quesito de orientação a objeto (OO). A razão é simples: o desenvolvimento de *software* utilizando esse enfoque ainda não dispõe de métricas precisas e bem entendidas que possam ser utilizadas para avaliar produtos e processos de desenvolvimento nessa abordagem. As várias propostas para métricas OO levam em consideração as características básicas e interações do sistema como: número de classes, número de métodos, linhas de código por método, profundidade máxima da hierarquia de classes, a relação existente entre métodos públicos e privados, entre outros. Tais métricas baseiam-se na análise detalhada do projeto.

As métricas escolhidas neste trabalho são as métricas de orientação a objetos (XENOS et al., 2000). E está dividida em cinco categorias: métricas de classes, métricas de métodos, métricas de herança, métricas de acoplamento e métricas do sistema.

CHIDAMBER e KEMERER (1994, p. 482-486) sugerem os seguintes conceitos como métricas:

- **NOC (Number of Children)**: é a quantidade de subclasses imediatamente subordinadas à classe na hierarquia, ou seja, quantas subclasses estão herdando os métodos de uma superclasse. Quanto menor o seu valor, menor é o potencial de reuso das subclasses. A Figura 3.1 ilustra o número de subclasses posicionadas imediatamente abaixo da superclasse em questão, na qual o A representa a superclasse e B, C, ... F são as subclasses. A opção A, demonstra que a Superclasse A tem apenas uma subclasse, ou seja, $NOC(A) = 1$. Na opção B, a superclasse B possui duas subclasses, ou seja, $NOC(B) = 2$.

Figura 3.1 - Métrica NOC

Extraído de: CHIDAMBER; KEMERER, 1994

- **WMC (Weighted Methods Per Class)**: é a quantidade de métodos de uma determinada classe, levando-se em conta as complexidades dos métodos iguais. Quanto maior for o WMC, mais específica torna-se a classe e melhor os seus serviços atendem às necessidades individuais. A Figura 3.2 exemplifica o conceito, em que os métodos ponderados por classe podem ser observados no cálculo do número de serviços por classe, que neste caso são três serviços ofertados em uma classe X.

Figura 3.2 - Métrica WMC

$$WMC(X) = 3$$

Extraído de: CHIDAMBER; KEMERER, 1994

- **DIT (Depth of Inheritance Tree):** é a profundidade da árvore de herança, ou seja, quantas classes ancestrais podem afetar potencialmente uma determinada classe. O DIT demonstra que muitas características foram adquiridas por herança e são comuns a outras classes, possibilitando a reutilização. No outro extremo, aumenta a complexidade da herança, como pode ser observado na Figura 3.3, na opção A, as letras A e B são superclasses em relação à classe C, por isso, mostra $DIT(C) = 2$. Na opção B, as superclasses estão com as letras A e D em relação da classe E, classificando $DIT(E) = 2$

Figura 3. 3 - Métrica DIT

Extraído de: CHIDAMBER; KEMERER, 1994

- **CBO (Coupling between object classes):** é a quantidade de outras classes a que uma determinada classe está acoplada. Quanto maior a ligação entre elas, menor a possibilidade de reutilização. Na Figura 3.4 mostra qual é o nível de acoplamento entre as classes da aplicação. Quanto maior a ligação entre elas, menor a possibilidade de reutilização, pois a classe torna-se dependente de outras classes para cumprir suas obrigações. Portanto, o CBO está diretamente ligado ao nível de reaproveitamento. Assim, um alto acoplamento indica uma baixa independência de classe, o que aumenta consideravelmente a complexidade e, em consequência, o esforço de teste. E no exemplo o $CBO(X) = 2$ é considerado como baixo.

Figura 3.4 - Método CBO

Extraído de: CHIDAMBER; KEMERER, 1994

- **LCOM (Lack of cohesion in methods)**: número de acesso a um ou mais atributos em comum pelos serviços da própria classe. Dessa forma, os serviços tornam-se ligados pelos atributos, podendo indicar que foram mal projetados, pois apresentam baixa coesão. A Figura 3.5 exemplifica quatro métodos de uma classe *C* e compartilham das variáveis da instância *C*.

Seja *C* uma classe com 4 métodos e com os seguintes conjuntos de atributos utilizados por método:

- $I_1 = \{a, b, x\}$
- $I_2 = \{a, e, f, g\}$
- $I_3 = \{c, x\}$
- $I_4 = \{v, u, w\}$

Para o cálculo de LCOM é necessário obter as intersecções entre os pares

Figura 3.5 - Exemplo da métrica LCOM

Extraído de: CHIDAMBER; KEMERER, 1994

Conforme a equação 3.1, pode-se definir melhor o LCOM: Seja I_j o conjunto de variáveis de instância *C* utilizado pelo método M_j . Considerando-se que *C* contém *n* métodos, podem-se definir os conjuntos:

$$P = \{(I_j, I_k) \mid I_j \cap I_k = \emptyset\} \text{ e } Q = \{(I_j, I_k) \mid I_j \cap I_k \neq \emptyset\}$$

$$LCOM = \begin{cases} |P| - |Q| & \text{se } |P| > |Q| \\ 0 & \text{em caso contrário} \end{cases} \quad 3.1$$

- **RFC (Response for a class)**: indica a capacidade de resposta que a classe tem ao receber mensagens de seus objetos. Uma maior capacidade de resposta requer

uma estrutura de classe projetada para atender a essa particularidade gerando uma maior complexidade, tornando necessário um maior esforço de teste. Na Figura 3.6 pode ser vista a capacidade de resposta da classe *ScannerCS* com relação à Superclasse *Scanner*.

Figura 3. 6 - Exemplo da métrica RFC

Extraído de: CHIDAMBER; KEMERER, 1994

Os limites para as métricas de CHIDAMBER e KEMERER (CK) são mostrados na Tabela 3.2, como padrão de utilização no trabalho proposto.

Tabela 3. 2 -Resumo das Métricas CK

Métrica CK	Descrição
WMC	≤ 25 (ideal) ≤ 40 (aceitável)
DIT	5 predomina → herança muito utilizada, mas complexidade elevada. Para classes com DIT > 5 atenção às demais métricas, pois elas subestimam sua complexidade.
NOC	Não foi encontrado valor bom ou ruim, pois depende de outras métricas.
CBO	≤ 5
LCOM	(Inclui construtores e destrutores): ≤ 20 (ideal) ≤ 40 (aceitável)
RFC	≤ 50

Extraído de: CHIDAMBER; KEMERER, 1994

3.4 Métricas de Banco de Dados

Ferramentas básicas aos sistemas gerenciadores de bancos de dados, as estruturas de indexação possibilitam o correto armazenamento e sobretudo a facilidade na sua recuperação. Para suportar a busca por imagens similares foram desenvolvidas estruturas métricas (Métodos de acessos métricos – MAM), que são estruturas de dados espaciais com dimensões definidas. Portanto, a busca por similaridade entre os dados garante o quão similar são os dados buscados em uma base (CHINO, 2004). Normalmente esta busca é dada por uma função distância (função

de dissimilaridade), que retornará a zero se a imagem for idêntica, ou um valor positivo quando for encontrada uma semelhança. Quanto maior for a diferença entre as distâncias, mais diferentes serão os dados. Na literatura existem diversos métodos de estruturas, por exemplo, *Multi-vantage point tree* (MVP-TREE), GNAT, M-tree e *Slim-tree* (TRAINA, 2000).

3.4.1 Definições

a) Espaço métrico

O espaço métrico é um par $M = (D, d)$, sendo D o domínio do vetor de características (chave de indexação) e $d()$ a função distância que satisfaz as seguintes propriedades (TRAINA, 2000):

$$\text{Simétrico} = d(x, y) = d(y, x)$$

$$\text{Não negatividade: } 0 < d(x, y) < \infty, \text{ para } x \neq y$$

$$\text{Desigualdade triangular: } d(x, y) \leq d(x, z) + d(z, y)$$

sendo x, y e z objetos pertencentes a D .

b) Consulta aos k vizinhos mais próximos - $k - NN$ (*k-nearest neighbor*).

Dado um conjunto de dados $O = \{O_1, O_2, \dots, O_n\}$, de um domínio D , uma função de distância métrica $d()$, um objeto de consulta $Q(D)$ e um número inteiro $K = 1$, a consulta $K - NN$, selecionará os K objetos do conjunto de dados que estão mais próximos de Q .

Utilizando um exemplo, a consulta “Selecione as cinco estrelas mais próximas do Sol”, busca os $k = 5$ objetos “estrelas” mais próxima ao objeto “Sol”, considerando o mesmo domínio de objetos (TRAINA, 2000).

3.4.2 Estrutura de Indexação Ordenada

Em uma base de índices ordenada, cada estrutura de um índice é associada a uma chave de procura em particular. Os registros em um arquivo indexado podem, eles próprios, serem armazenados em alguma ordem. Um arquivo pode ter diversos índices com diferentes chaves de procura. Se o arquivo que contém os registros está ordenado sequencialmente, o índice cuja chave de procura especifica a ordem

sequencial do arquivo é o índice primário. Os índices primários também podem ser chamados de índice *clustering* (TRAINA, 2000).

3.4.3 Estrutura de Indexação da *Slim-tree*

A *Slim-tree* (TRAINA, 2000) é uma árvore multivias balanceada e dinâmica desenvolvida para indexar objetos contidos em um espaço métrico, possibilitando a execução de buscas por similaridade diminuindo o número de cálculos de distância quanto ao número de acessos ao disco.

A *Slim-tree* possui nós internos chamados *indexnodes*, e as folhas que são chamados de *leafnodes*. Cada nó possui um objeto representante, um raio de cobertura e objetos de conjunto de dados da sua área de cobertura.

A organização é feita em páginas de disco de tamanho fixo, cada uma contendo um nó da árvore. O balanceamento da estrutura é obtido mediante o crescimento de baixo para cima, das folhas para a raiz e os objetos são armazenados apenas no último nível da árvore.

Os *nodes* podem ser divididos em: *IndexNode* – Nó do índice, *LeafNode* – Nó da Folha.

Os Nós-índices são compostos por C entradas, cada uma referente a uma subárvore. Uma entrada é composta pelo objeto representativo S , o raio de cobertura R e o número de entradas da subárvore $\#Ent$. Ela ainda possui o ponteiro para a cabeça da subárvore Ptr e a distância entre o objeto S e o objeto representativo do nó S_{Rep} que está armazenado no nível superior da árvore. A estrutura de um nó-índice é definida por (CHINO, 2004):

$$IndexNode = [\text{vetor } [1..C] \text{ de } \langle S, R, \#Ent, Ptr, d(S_{rep}, S) \rangle]$$

Os Nós-folhas são responsáveis por armazenar todos os objetos, não possuindo subárvores associadas.

A Figura 3.7 (CHINO, 2004) ilustra um exemplo da estrutura de uma *Slim-tree*, com 12 objetos e a função de distância euclidiana.

Figura 3. 7 - Exemplo de uma estrutura de *Slim-tree*

Extraído de: CHINO, 2004

O algoritmo de consulta da *Slim-tree* aproveita-se da estrutura de árvore e da desigualdade triangular da função de distância para dar mais precisão ao número de acessos e do cálculo da distância.

Nessa consulta são necessários o objeto da consulta e o parâmetro à busca. Após percorrer os nós, será feita somente uma busca recursiva na qual as subárvores podem conter o resultado, dando assim um maior dinamismo da busca. O algoritmo terminará quando não houver mais ramos que possa ter uma similaridade do objeto procurado. Sempre que um objeto mais próximo ao raio da busca é encontrado, ele é inserido na resposta. Nessa situação, pode ser encontrado dois ou mais objetos no mesmo raio, e nessa situação a *Slim-tree* tem um critério de desempate, ou seja, ela pode trazer todas as imagens que estão nessa situação ou apenas o primeiro objeto da lista de empate encontrado (CHINO, 2004).

3.5 Considerações sobre este capítulo

Neste capítulo foram apresentadas as métricas de desempenho de sistemas em rede de computadores. As métricas de rede avaliam o desempenho da rede para um bom funcionamento de aplicação. As métricas de *software* são utilizadas para estimar em processo ou parâmetros do sistema. As métricas de banco de dados utilizam estruturas de indexação para avaliar especialmente a facilidade de recuperação de imagem.

As métricas a serem utilizadas neste projeto são:

- Métricas de rede: Atraso (latência RTT), Perda de Pacotes, *Download/Upload*, Disponibilidade e Capacidade.
- Métricas de *Software*: Métricas de Orientação a Objetos (NOC, WMC, DIT, CBO, LCOM e RFC).
- Métricas de Banco de Dados (*SELECT/UPDATE*).

Neste projeto, como uma opção de métrica, será demonstrada a eficiência da busca por similaridade em um banco com indexação ordenada (utilizando o MYSQL) e fará a comparação com a estrutura de arquivo de índice árvore *Slim-tree* em um segundo momento. A *Slim-tree* é um Método de Acesso Métrico (MAM), que se manifesta como uma árvore balanceada, na qual cada nó será armazenado em uma página em disco de tamanho fixo.

Capítulo 4 – Recuperação Baseada em Conteúdos

4.1 Introdução

Com a grande quantidade de imagens médicas utilizadas na *Web*, a necessidade da recuperação dessas imagens de forma rápida e com a redução de custo de armazenamento, cria-se um ambiente favorável à busca de grande quantidade de imagens de forma segura e compacta, visando consulta e recuperação de itens relevantes em banco de dados.

4.2 Recuperação de imagens por conteúdo

Um sistema de recuperação de imagens por conteúdo (CBIR - *Content Based Image Retrieval*) é específico na noção de similaridade de imagens. Existindo um banco de dados populado com imagens, dá-se uma imagem para a recuperação das mais similares de acordo com um padrão de consulta. Normalmente esse processo de recuperação é baseado na comparação de imagens com base em um vetor de características (GRACIOSO, 2008).

Diferente da busca feita em um banco de dados convencional, no CBIR esta busca é contextualizada em características como cor, forma ou textura, que é conhecido também como características de baixo nível.

Existem diversas formas de se construir um sistema CBIR (MÜLLER et al., 2004). Na Figura 4.1 é apresentada uma aplicação, em que dada uma imagem de consulta, pesquisa-se em um banco de dados que estão armazenadas as imagens e encontram-se as mais semelhantes, de acordo com uma métrica preestabelecida. Ao se conceituar que a recuperação é feita por imagem, isso significa que essa recuperação é feita pela característica extraída dela.

Figura 4.1 – Recuperação de imagens baseada em conteúdo

Extraído de: GRACIOSO, 2008

4.3 Arquitetura de um Sistema CBIR

A arquitetura de um sistema CBIR típico pode ser visto na Figura 4.2 (TORRES, 2006).

Figura 4.2 - Arquitetura de um sistema de recuperação de imagens por conteúdo

Extraído de: TORRES, 2006

O sistema inicial de inserção de dados, na Figura 4.2 está representado pelas setas tracejadas e por módulos, é responsável pela extração do vetor de características das imagens e pelo armazenamento em uma base de dados. Normalmente esse procedimento é executado uma única vez com cada imagem.

A interface permitirá que o usuário entre com uma imagem específica e após o processamento obtenha as imagens mais similares a inicial. O módulo de processamento de consulta é responsável por extrair o vetor de características do padrão de consulta e de acordo com uma métrica de distância preestabelecida, avaliar a similaridade das imagens, com o vetor de entrada. Para a exibição final ao usuário, devem-se classificar as imagens resultantes, retornando as mais similares.

4.4 Atributos da Imagem

Os sistemas CBIR trabalham com o conteúdo visual da imagem, como cor, forma e textura para indexar e representar imagens. Normalmente a recuperação de imagem baseia-se na similaridade das características derivadas de um atributo (SILVA, 2009).

4.4.1 Cor

Dentre todos os atributos, a cor é a característica mais usada para recuperação e indexação de imagens. A cor é o melhor descritor que simplifica a extração e identificação de um objeto em cena. Os olhos humanos podem diferenciar milhares de tons de cores e intensidades, comparando-se com duas dezenas de tonalidades de cinza.

Muitos trabalhos de extração de características são baseados na distribuição de cores concentrados no histograma de cor. Este histograma é uma aproximação probabilística da distribuição das cores na imagem. É associado cada intensidade de cor da imagem ao número de pixels da imagem. No histograma não há variação de cor independente da rotação ou translação da imagem e também com o histograma normalizado têm-se a invariância à escala. Na Figura 4.3, é demonstrada graficamente uma representação da imagem feita por seu histograma (SILVA, 2009).

Figura 4.3 - Representação da Imagem por Histograma de intensidade de pixels

Extraído de: SILVA, 2009

A distribuição de cores pode ser utilizada para montar o vetor de características, baseado nos coeficientes de *Wavelets*, com diminuição de tempo na busca das imagens similares e um ganho de eficiência de 35% (CASTAÑÓN, 2003).

4.4.2 Forma

É possível analisar ou identificar diversos objetos pelas formas destes. Entretanto, representar ou definir formas contidas em uma imagem não é uma tarefa fácil, pois normalmente os algoritmos de caracterização processa a imagem com algoritmo de segmentação, isolando o objeto desejado e aplicando-se posteriormente um extrator de características (CASTAÑÓN, 2003).

Os tumores podem ser citados como um exemplo bem clássico em imagens médicas, esses são classificados mediante a forma e o tamanho como maligno ou benigno. Existe uma grande probabilidade de um tumor maligno ter as suas bordas irregulares, ao passo que as bordas mais regulares são benignos.

Normalmente para a extração de forma utilizam-se dois tipos de descritores: baseados em contorno e baseado em região. Nos baseados em contorno é localizado apenas a borda ou fronteira da área desejada, ignorando totalmente a região interna. Já nos baseados em região, considera-se não somente a borda, mas os detalhes internos da imagem. Portanto, a similaridade da forma baseia-se em contorno ou região.

As técnicas mais utilizadas para a segmentação da forma de objetos podem ser vistas no diagrama da Figura 4.4.

Figura 4. 4 - Classificação das técnicas de formas

Extraído de: CASTAÑÓN, 2003

Está fora do escopo deste trabalho a discussão de cada um desses métodos, que podem ser encontrados na literatura sobre processamento de imagem ou visão computacional.

4.4.3 Textura

Entendem-se como textura as alterações na intensidade da imagem que formam determinados padrões repetitivos. Para o sistema visual humano, ela é reconhecida como “fina”, “grossa”, “lisa”, “áspera”, entre outras (THEODORIDIS, 1999). A imagem tem sua textura definida pela maneira como os níveis de cinza estão distribuídos nos seus pixels.

Simplificadamente pode-se entender a textura como um conjunto de propriedades intrínsecas da imagem como brilho, cor, tamanho etc. Sobre as características da textura da imagem podem-se analisar alguns aspectos como: ondulação, suavidade, aspereza, regularidade ou linearidade. Na Figura 4.5.A, a imagem mostra textura suave, a 4.5.B mostra a textura áspera e a 4.5.C. é um exemplo de textura regular (GONZALES; WOOD, 2007).

Figura 4.5 – A) imagem microscópica de um supercondutor; B) colesterol humano; C) um microprocessador

Extraído de: GONZALES; WOOD, 2007

As características de textura são parâmetros matemáticos computados sobre uma distribuição de pixels, no qual caracterizará o tipo de textura e a estrutura dos objetos na imagem. A divisão mais usada classifica os métodos em estruturais, baseados em modelos, baseados em transformadas e estatísticos (MATERKA; STRZELECKI, 1998).

A abordagem Estrutural representa a estrutura por um conjunto de primitivas¹ bem definidas e de um arranjo hierárquico dessa primitiva. A abordagem baseada em transformada utiliza transformadas, por exemplo, Gabor e *Wavelet* representando imagem em um espaço com sistemas de coordenadas relacionado à textura. Já a abordagem estatística está dividida em dois tipos: Estatística de primeira ordem, baseada em histograma, uma abordagem que mede a probabilidade de um valor de cinza num local escolhido da imagem, e estatística de segunda ordem, que usa relações de pares de pixels da imagem analisadas.

4.5 Extração de Características

A extração de características é um processo de criação de um conjunto de vetores que representam uma imagem, representado em um vetor v , chamado de vetor de características d (*feature vector*), no qual $v \in R^d$ (LEHMANN et al., 2005). Essas características devem introduzir uma alta descrição entre as possíveis classes de dados de entrada, usando assim técnicas de análise de imagens, combinadas ou não. O termo característica é qualquer aspecto, quantitativo ou qualitativo, ou seja, a característica pode ser simbólica, por exemplo, a cor; ou numérica, por exemplo, a

¹ Uma primitiva é utilizada para modelar objetos complexos de forma simplificada.

altura. O espaço de d-dimensões formado pelo vetor de características é chamado de espaço de características (GONZALES; WOODS, 2007).

Após a extração de características, chega-se a um novo conjunto de dados (vetor) com uma dimensão inferior ao da imagem inicial, ou seja, esse vetor deverá ser o mais descritivo possível, não comprometendo o desempenho do sistema de classificação, barateando sobretudo o custo computacional.

Um exemplo de extração e discriminação entre as imagens é apresentado na Figura 4.6.

Figura 4. 6 - Extração de Características de Imagens

Extraído de: FONSECA, 1998

Em recuperação de imagens médicas, as características extraídas das imagens devem ser equivalentes às características que os especialistas examinam. Se houver diferença entre as características extraídas pelo sistema computacional e as características do especialista (propriedades de imagens consideradas semanticamente semelhantes) – imagens recuperadas por um sistema CBIR – existe o que se chama de “gap” semântico² (FONSECA, 1998).

Quando se trabalha com imagens médicas é comum haver algumas com os seguintes problemas:

- **Contraste:** a calibração do equipamento pode resultar em imagens diferentes.
- **Escala:** para cada caso, os pacientes de diferentes tamanhos são analisados.
- **Rotação:** em alguns momentos não é possível manter o paciente em uma mesma posição para comparar as imagens.

² Diferença entre características de baixo nível extraídas da imagem, e o alto nível da informação necessária para o usuário.

- **Translação:** varia sempre em relação ao paciente e ao equipamento na forma da aquisição da imagem.

Uma das formas para extração de características em imagem médica é a utilização da Transformada *Wavelet* (CASTAÑÓN, 2003). A Transformada *Wavelet* recuperará componentes de baixa e alta frequência da imagem, em diversos níveis de decomposição. A imagem original é decomposta em um conjunto de coeficientes de tamanho igual. Esta transformada é derivada da Transformada de *Fourier*, na qual utilizava funções senoidais definidas pelo parâmetro frequência (MALLAT, 1989). Em comparação com a Transformada de *Fourier*, a Transformada *Wavelet* usa funções de base que são definidas por dois parâmetros: posição e escala, permitindo que as características de um sinal sejam extraídas em razão do tamanho e a posição da imagem garantindo assim a invariância à translação e escala.

4.6 A Transformada *Wavelet*

A Transformada *Wavelet* (TW) surgiu para a análise de sinais e tem sido aplicada amplamente em processamento digital de imagens. Um destaque importante é conseguir representar uma imagem com uma quantidade mínima de valores, em uma escala mínima, criando um vetor de características pequeno para a sua representação (GRACIOSO, 2008). Em se tratando de baixo custo computacional se dará maior ênfase a *Wavelet Haar*. Para este trabalho não será contemplada a reconstrução da imagem, visto que o interessante é o processo de decomposição da imagem.

4.6.1 Transformada *Wavelet Haar*

A primeira menção da Transformada *Wavelet Haar* apareceu em um apêndice da tese de Alfred Haar em 1909. Ele descreveu uma base ortonormal de *wavelet* sobre o domínio $[0,1]$, sendo $h_0(x), h_1(x), \dots, h_n(x)$, além das bases de *Fourier*, para qualquer função contínua $f(x)$ no intervalo $[0,1]$, conforme mostram as equações 4.1 e 4.2 (MÜLLER et al, 2004).

$$h(x) := \begin{cases} 1 & , \quad x \in [0, 0.5] \\ -1 & , \quad x \in [0.5, 1] \\ 0 & , \quad \text{caso contrário} \end{cases} \quad (4.1)$$

$$h_n(x) = 2^{\frac{j}{2}} h(2^j x - k) \quad (4.2)$$

sendo $n = 2^j + k, k \in [0, 2^j], x \in [k 2^{-j}, (k + 1)2^{-1}]$

4.7 Análise em Multirresolução

A análise em Multirresolução (AMR) é adequada para a extração de características, porque com poucos coeficientes é possível obter uma representação da imagem. A análise em Multirresolução decompõe a imagem em níveis (n), em um conjunto de coeficientes, sendo eles: cA_n são as baixas frequências, cD_n^h como coeficientes verticais e cD_n^d como coeficientes diagonais, representando as altas frequências. Em um próximo nível de decomposição, ou seja, $n + 1$ o cA_n é usado como base para o próximo nível (GONZALES; WOODS, 2007).

Uma decomposição Multirresolução possibilita uma interpretação invariante de escala da imagem. Esta escala varia em razão da distância entre a cena e o centro ótico da câmera. Uma representação Multirresolução será parcialmente invariante de escala, se a sequência de parâmetros da resolução variar exponencialmente (MALLAT, 1989).

A Figura 4.7 mostra um modelo esquemático da análise em Multirresolução e a Figura 4.8 apresenta a aplicação da Transformada *Wavelet Haar* com dois níveis de resolução (GRACIOSO, 2008).

Figura 4.9 - Imagem usando Transformada *Wavelet Haar*

Extraído de: GRACIOSO, 2008

A aplicação desta distância em busca por conteúdos entende-se que uma imagem A está próxima a um conjunto de imagens B. Quando as condições não são satisfatórias, o resultado será demonstrado com uma disparidade muito alta.

4.8 Considerações sobre este capítulo

A recuperação de imagens baseada em conteúdos (CBIR) é uma técnica que extrai características das imagens utilizando cor, textura e forma para busca em uma base de dados de imagens cujos vetores de características sejam similares.

Dentre as diversas metodologias existentes, optou-se para este trabalho, utilizar a Transformada *Wavelet Haar*, em razão da sua simplicidade de implementação e ser relativamente invariante à escala, rotação e translação das imagens médicas. Além disso, a possibilidade de se empregar a análise em multirresolução permite a redução do vetor de características melhorando o desempenho da busca, que é um objetivo deste trabalho.

Neste trabalho será utilizado imagens médicas com o tamanho de 512x512 e codificação de 8 *bits* de níveis de cinza, subdivididas em cinco classes distintas: crânio, tórax, abdômen, colo lombar e espinha, provenientes de exames de Tomografia Computadorizada.

Portanto, a parte mais trabalhosa do processo de recuperação de imagem é a definição correta do espaço de característica para realizar a correlação entre os atributos similares, pois esse espaço de características será utilizado no processo de recuperação de imagens por conteúdo e definirá o que será comparado com a métrica estabelecida na busca da imagem.

Capítulo 5 – Metodologia

5.1 Introdução

Neste capítulo serão apresentados os principais experimentos realizados na utilização do desempenho de busca de imagens por conteúdos em rede de computadores. São descritos os conjuntos de dados utilizados, as formas de abordagens na recuperação da imagem na rede e os cenários utilizados.

Foi desenvolvido um código em uma arquitetura de programação, no qual garante a reutilização deste em outros eventos, finalizando uma ferramenta para auxiliar na usabilidade de extração de vetores de características usando a Transformada *Wavelet Haar*. Neste contexto, foram empregadas as métricas de rede, definindo-as e mensurando os resultados. Grande parte dos projetos de visão computacional está preocupada com a melhor técnica de extração ou o melhor tipo de busca de imagem, entretanto, não se dá a atenção com o desempenho do resultado em tela ou em uma rede, como a internet. Portanto, aqui se busca o desempenho de uma rede, utilizando a métrica com a extração mais adequada. Foram comparadas também as informações extraídas de uma planilha (base do projeto de GRACIOSO (2008)), com uma base em MYSQL (*Innobd* e *MYIsam*) e da *Slim-tree*, satisfazendo, assim, uma aplicação da metodologia em base de dados indexada, extraindo as métricas de acesso e recuperação de dados garantindo melhor desempenho do projeto proposto.

O Trabalho apresentou uma reengenharia de projeto por meio da alteração da linguagem de programação original (utilização do *Software MatLab Server*) e o *software* proposto (Java, implantando API para extração de características), utilizando as Métricas de Redes, verificando as métricas de velocidade e o aplicando uma estrutura de armazenamento e recuperação dos Vetores de Características em bases de dados.

5.2 Metodologia e Materiais

O sistema proposto foi desenvolvido em um equipamento de processador INTEL Centrino 2533 MHZ com 4GB de RAM, 500 GB de HD e Sistema Operacional *Windows Seven*. O servidor no qual foram feitos os testes possui um processador

Pentium 4, 2.8 MHz com HD de 200 GB e 1GB de memória RAM. O motivo de utilizar um equipamento mais antigo no servidor deve-se ao mecanismo de *Benchmark* com os dados colhidos em GRACIOSO (2008). O sistema foi desenvolvido em Java.

No desenvolvimento foi criada uma API (*Application Programming Interface*) *Java Servlet*. Chama-se de *Servlet* os módulos de códigos em Java que executam como aplicações servidoras do lado do servidor *Web*.

A tarefa de busca por similaridade é dividida em duas fases. A etapa 1 tem os objetivos de criar e armazenar nos bancos de dados (*Mysql - Engine InnoDB*, e na *Slim-tree*) os vetores de características das imagens de exames de tomografia computadorizada obtidas pela Transformada *Haar*.

Na etapa 2, faz-se o *upload* da imagem utilizando o sistema concebido, cria-se o vetor dessa imagem e aplica-se a distância euclidiana retornando as 20 imagens mais similares. A Figura 5.1 ilustra o processo.

Figura 5. 1 - Esquema da Extração do Vetor em Java com API

Extraído de: GRACIOSO, 2008

A base final do projeto possui 9527 imagens no formato JPEG, cedidas pelo GBdi (Grupo de Bases de Dados e de Imagens) do Instituto de Ciências Matemáticas e de Computação da USP de São Carlos, estas imagens são resultados de exames de tomografia computadorizada, possuindo uma padronização de 512x512 pixels, codificada em 8 níveis de cinza. O universo de comparação foi separado em

cinco classes distintas, ou seja, coluna vertebral, colo lombar, crânio, tórax e abdômen. Estas classes podem ser vistas nas Figuras de 5.2 a 5.6.

Figura 5. 2 - Classe de Coluna Vertebral

Extraído de: GRACIOSO, 2008

Figura 5. 3 - Classe de Colo Lombar

Extraído de: GRACIOSO, 2008

Figura 5. 4 - Classe de Crânio

Extraído de: GRACIOSO, 2008

Figura 5. 5 - Classe de Tórax

Extraído de: GRACIOSO, 2008

Figura 5. 6 - Classe de Abdômen

Extraído de: GRACIOSO, 2008

Com o intuito de preservar os valores referenciais da pesquisa e possível distorção, foram disponibilizadas as mesmas cargas de tráfego e os mesmos equipamentos durante os testes realizados.

Para garantir a eficiência do sistema, usou-se a Transformada *Wavelet Haar* para a criação dos vetores de características. Esse vetor com 16 posições foi utilizado para procurar as 20 imagens mais similares, com a métrica de distância euclidiana. Com 20 imagens por consulta consegue-se observar algumas particularidades da rede em teste, como por exemplo, o fluxo de tráfego, a carga de tráfego (para poder estabelecer capacidade de enlaces) e o comportamento do tráfego (considerações de broadcast, eficiência).

A técnica de aferição entre os dois sistemas que será empregada é *Benchmarks*, ou seja, a mesma tarefa será submetida a diferentes sistemas e seus resultados serão comparados entre si. Esse resultado será utilizado para avaliar o desempenho do Sistema.

5.3 Configuração do Setup de Testes

O Sistema foi instalado em um domínio público para que o usuário acessasse remotamente. Nesta situação, o usuário faz o *upload* da imagem para a realização do primeiro teste.

Com base no navegador do usuário, foi gerado tráfego de dados para o IP do servidor e este tráfego foi analisado com o auxílio do *software* IPERF (comando) e JPERF (Interface Gráfica). Esse tráfego de dados é realizado por intermédio de uma conexão cliente/servidor entre as estações envolvidas. Durante todo o período de medição, esse tráfego esteve presente em *download* e *upload* das imagens. Tanto o IPERF, como o JPERF são softwares cliente/servidor adequados para medições ativas. Além das medições, esses softwares podem ser usados com geradores simples de carga na rede, quando não há preocupação com o perfil do tráfego que está sendo gerado. Também é possível medir o Jitter (variação do atraso) e a perda, bem como é capaz de usar tanto o protocolo UDP, quanto TCP e pode lidar com múltiplas conexões simultâneas. No caso do protocolo TCP há um mecanismo de controle de

congestionamento que procurará usufruir de toda a banda disponível no caminho, se o mecanismo de controle de fluxo permitir.

Como parte do teste, foi solicitado que o usuário enviasse uma imagem no formato BMP, uma JPEG e uma TIFF para a demonstração da eficiência das transmissões das imagens JPEG em uma rede. As imagens têm os respectivos tamanhos: no formato BMP 769 KB, no formato TIFF 192 KB e no formato JPEG tem 18 KB. A Figura 5.7 mostra um exemplo das imagens utilizadas para esta simulação.

Figura 5. 7 - Imagem de utilizada no teste de transmissão

Extraído de: GRACIOSO, 2008

Na segunda parte do teste o usuário executa o procedimento: escolhe inicialmente uma imagem (das 9527 disponíveis), e a plataforma de teste, na qual pode ser a utilização de *MatLab* com a Planilha como base de dados, ou o Sistema em Java utilizando a base de dados do *Mysql*, ou o Sistema em Java utilizando a base do *Slim-tree*, finalizando o processo com o *upload* da imagem.

Ao enviar a imagem, o Sistema gera um *Log* de *Upload*. Existe o processamento da imagem, criando-se o vetor de características. O sistema faz a busca pela distância euclidiana e traz as 20 imagens mais similares à original. O usuário repete o mesmo procedimento utilizando outra imagem, ou seja, envia a imagem 2, 3 e 4. A imagem selecionada pelo usuário é comparada nas outras bases. Com esta estratégia, podemos fazer um *Benchmark* entre todas as bases de dados.

Utilizando as métricas e os indicadores de desempenho relacionados no Capítulo 3, foram aferidos: a eficiência de rede; *throughput*, banda consumida; entre outros indicadores que refletem o desempenho da rede em diversos aspectos. Estes resultados estão mostrados no Capítulo 6.

Neste cenário, participaram 150 usuários, dentre os quais os resultados de 105 usuários foram considerados como completos para a execução dos testes. Estes usuários possuem *links* ADSL variando de 128 Kbit/s, 512 Kbit/s, 1 Mbit/s, 2Mbit/s a 10 Mbit/s (de acordo com o *link* disponível no seu trabalho ou residência). A escolha da velocidade caracteriza um padrão disponibilizado a um usuário comum para navegação na internet.

Para se ter a lisura do processo de medição, pediu-se para que o usuário acessasse o link <http://simet.nic.br/> para testar a velocidade da conexão. A imagem de cada usuário está disponível no Anexo C.

5.4 Reengenharia do projeto

Uma das maneiras de aplicar reengenharia em um sistema procedimental, para amenizar seus custos e esforços de manutenção, sem alterar a funcionalidade do sistema, é mudar a linguagem de programação, de procedimental para a orientada a objetos. Serão demonstradas as principais alterações no código e as métricas de *softwares* aplicadas à orientação de Objetos.

No estudo de GRACIOSO (2008), foram efetuadas diversas comparações dentre as Transformadas *Wavelet* (Db2, Db3, Db4 e Transformada *Haar*), obteve-se um êxito de 100% de acerto (com o retorno de 20 imagens) em todas as classes quando efetuada a busca por similaridade com a Transformada *Wavelet de Haar*, aplicada em uma base com 4728 imagens, retornando assim as 30 imagens da classe desejada. Em comparação à Transformada *Wavelets* Db2, por exemplo, houve um acerto de 84,05% (GRACIOSO, 2008, p. 93). Portanto, este projeto utilizará a Transformada *Haar* para classificação e recuperação das imagens.

Na Figura 5.8 é mostrada o código no *MatLab* original para a Transformada *Wavelet Haar* desenvolvido. Nesta figura observa-se a entrada da imagem. O código inteiro está disponível no Apêndice A.

```

%le a imagem
img = imread(nome);
beta(i,:)=e_da(img); %energia
alfa(i,:)=mv_da(img); % média e variância

```

Figura 5. 8 - Parte do código que faz a transformada *Wavelet Haar*

Após a localização da imagem e o resultado desse processamento, o armazenado em uma variável “img”, existem duas funções que calcula a energia e na sequência, com o resultado, é calculado a variância. Na Figura 5.9 pode ser vista a função `e_da(img)`, que inicia o processo de multirresolução utilizando funções específicas do *MatLab*.

```

function v = e_da(M)

%primeira escala wavelet
[cA,cH,cV,cD] = dwt2((double(M)), 'Haar'); %aplica a base de wavelet Haar à imagem

%Segundo escala wavelet
[cAA,cHA,cVA,cDA] = dwt2(cA, 'Haar'); %aplica a base de wavelet Haar à imagem

%terceira escala wavelet
[cAAA,cHAA,cVAA,cDAA] = dwt2(cAA, 'Haar'); %aplica a base de wavelet Haar à imagem

%quarta escala wavelet
[cAAAA,cHAAA,cVAAA,cDAAA] = dwt2(cAAA, 'Haar'); %aplica a base de wavelet Haar à imagem

```

Figura 5. 9 - Função `e_da()`

A média e variância é mostrada na Figura 5.10 na função `mv_da(img)` e armazenados nas variáveis `V(1)`, `V(2)`..., `V(8)`.

```

function v = mv_da(M)

%primeira escala wavelet
[cA,cH,cV,cD] = dwt2((double(M)), 'Haar'); %aplica a base de wavelet Haar à imagem

%Segundo escala wavelet
[cAA,cHA,cVA,cDA] = dwt2(cA, 'Haar'); %aplica a base de wavelet Haar à imagem

%terceira escala wavelet
[cAAA,cHAA,cVAA,cDAA] = dwt2(cAA, 'Haar'); %aplica a base de wavelet Haar à imagem

%quarta escala wavelet
[cAAAA,cHAAA,cVAAA,cDAAA] = dwt2(cAAA, 'Haar'); %aplica a base de wavelet Haar à imagem

v(1)=mean(mean(cH));
v(2)=variancia(cH);
v(3)=mean(mean(cV));
v(4)=variancia(cV);
v(5)=mean(mean(cD));
v(6)=variancia(cD);
v(7)=mean(mean(cA));
v(8)=variancia(cA);

```

Figura 5. 10 - Função `mv_da(M)`

Na reengenharia, esses mesmos procedimentos mostrados nas Figuras 5.8, 5.9 e 5.10 são substituídos pela superclasse Teste (desenvolvida na linguagem Java). É nesta superclasse que se carrega a imagem e executa-se a função TransformadaHaar(), conforme Figura 5.11. O código completo está no Apêndice B (Reengenharia).

```
public static String teste(BufferedImage imagem) {
    Raster raster = null;
    /* try {
        //System.out.print("Imagem " + z + ": ");
        imagem = ImageIO.read(new File(z + ".jpg"));
    } catch (IOException ex) {
        ex.printStackTrace();
    }
    int x, y = 0;
    */
    try {
        raster = imagem.getRaster();
        double cores[] = new double[255];
        double matriz_main[][] = new double[512][512];
        for (int x = 0; x < imagem.getWidth(); x++) {
            for (int y = 0; y < imagem.getHeight(); y++) {
                raster.getPixel(x, y, cores);
                matriz_main[x][y] = (cores[0] + cores[1] + cores[2]) / 3;
            }
        }

        return transformadaHaar(matriz_main);
    } catch (Exception ex) {
        StackTraceElement[] ste = ex.getStackTrace();
        int i = 0;
        while (ste[i] != null) {
            System.out.println(ste[i].toString());
            i++;
        }
        return null;
    }
}
```

Figura 5.11 - SuperClasse da Reengenharia

E a chamada da Classe *HaarDemo* é que faz a Transformada *Wavelet de Haar* e devolve o vetor de característica com 16 campos, a Figura 5.12 mostra o processo de multirresolução e a transformada.


```

public static String transformadaHaar(double[][] mat) throws IOException {
    double[][] matr = new double[256][256];
    double[][] mat2 = new double[128][128];
    double[][] mat3 = new double[64][64];
    double[][] mat4 = new double[32][32];
    double[][] mat5 = new double[16][16];
    double[][] mat6 = new double[8][8];
    double[][] mat7 = new double[4][4];
    String retorno = "";
    for (int x = 0; x < 256; x++) {
        for (int y = 0; y < 256; y++) {
            matr[x][y] = (mat[2 * x][2 * y] + mat[2 * x][2 * y + 1] + mat[2 * x + 1][2 * y] + mat[2 * x + 1][2 * y + 1]) / Math.sqrt(2);
        }
        for (int x = 0; x < 128; x++) {
            for (int y = 0; y < 128; y++) {
                mat2[x][y] = (matr[2 * x][2 * y] + matr[2 * x][2 * y + 1] + matr[2 * x + 1][2 * y] + matr[2 * x + 1][2 * y + 1]) / Math.sqrt(2);
            }
        }
        for (int x = 0; x < 64; x++) {
            for (int y = 0; y < 64; y++) {
                mat3[x][y] = (mat2[2 * x][2 * y] + mat2[2 * x][2 * y + 1] + mat2[2 * x + 1][2 * y] + mat2[2 * x + 1][2 * y + 1]) / Math.sqrt(2);
            }
        }
        for (int x = 0; x < 32; x++) {
            for (int y = 0; y < 32; y++) {
                mat4[x][y] = (mat3[2 * x][2 * y] + mat3[2 * x][2 * y + 1] + mat3[2 * x + 1][2 * y] + mat3[2 * x + 1][2 * y + 1]) / Math.sqrt(2);
            }
        }
        for (int x = 0; x < 16; x++) {
            for (int y = 0; y < 16; y++) {
                mat5[x][y] = (mat4[2 * x][2 * y] + mat4[2 * x][2 * y + 1] + mat4[2 * x + 1][2 * y] + mat4[2 * x + 1][2 * y + 1]) / Math.sqrt(2);
            }
        }
        for (int x = 0; x < 8; x++) {
            for (int y = 0; y < 8; y++) {
                mat6[x][y] = (mat5[2 * x][2 * y] + mat5[2 * x][2 * y + 1] + mat5[2 * x + 1][2 * y] + mat5[2 * x + 1][2 * y + 1]) / Math.sqrt(2);
            }
        }
        for (int x = 0; x < 4; x++) {
            for (int y = 0; y < 4; y++) {
                mat7[x][y] = 1 / (mat6[2 * x][2 * y] + mat6[2 * x][2 * y + 1] + mat6[2 * x + 1][2 * y] + mat6[2 * x + 1][2 * y + 1]) / Math.sqrt(2);
                //System.out.printf("%11.9f;", mat7[x][y]);
                DecimalFormat df = new DecimalFormat("0.000000000");
                retorno+= df.format(mat7[x][y]) + ";";
            }
            retorno+= "\n";
        }

        //System.out.print("\n");
        return retorno;
    }
}

```

Figura 5. 12 - Processo de multirresolução e a transformada

5.5 Considerações sobre este capítulo

A metodologia apresentada neste Capítulo atendeu às necessidades de testes e é suficiente para atender aos objetivos do trabalho. Portanto, a qualidade de serviço em redes não é necessariamente resolvida com um único protocolo ou algoritmo ou programa. Na maioria dos casos e dependendo da necessidade da aplicação, um conjunto de novos recursos deve ser utilizado.

Capítulo 6 – Resultados

6.1 Introdução

Esta capítulo mostra os resultados dos experimentos realizados com base nos conceitos descritos no Capítulo 3 e complementados pela metodologia descrita no Capítulo 5. Com exceção das informações de média, desvio padrão, variância, mínimo e máximo, os resultados foram apresentados em forma de gráficos e complementados com tabelas, que encerram as métricas com informações de cada um dos parâmetros ou métrica, monitorado ou calculado. Podendo-se observar que as fronteiras que separam as informações dinâmicas de estatísticas podem ser misturadas. Neste contexto, cada métrica deverá ser analisada cuidadosa e individualmente, para não ser confundida em termos do seu valor analítico.

6.2 Experimentos de Desempenho de Tipos de Imagens na Rede

O primeiro estudo foi realizado para avaliar a qualidade de serviço percebida pelo tráfego de uma sessão interativa de transmissão de tipos diferentes de imagens (Arquivos .BMP, .TIF e .JPG) na internet. Este experimento envolveu todos os usuários e o servidor para a transmissão e recepção das imagens. Nessa situação, o usuário manda o arquivo para o servidor e é gerado um log de transmissão para calcular o tempo. Os usuários estavam conectados à rede de acesso ADSL. O tempo é dado em segundos, nesta situação. A seguir, a Figura 6.1 que mostra o comportamento regular durante todo o período de teste em torno da média.

Figura 6. 1 - Tempo total de *upload* de imagens para cada usuário

Tabela 6. 1 - Estatística de Comparação de Imagens do Tipo BMP com TIFF com JPEG

Estatísticas de <i>Upload</i> (Segundos)					
Imagem BMP		Imagem TIFF		Imagem JPEG	
Média	23,43	Média	5,65	Média	0,52
Desvio Padrão	12,63	Desvio Padrão	3,04	Desvio Padrão	0,28
Variância	159,71	Variância	9,29	Variância	0,08
Mínimo	8,40	Mínimo	2,02	Mínimo	0,18
Máximo	122,46	Máximo	29,53	Máximo	2,76

Conclui-se por meio das amostragens que a imagem BMP, apesar de perda de qualidade totalmente nula, nos testes apresentados, levou em média 23,43 segundos para ser transmitido ao Servidor. Já a imagem com o formato TIFF, apesar da sua compactação de altíssima qualidade e desenvolvimento para imagens de alta resolução, 300dpi, necessita de 5,65 segundos para a sua transmissão, entretanto, ainda assim mais vantajoso que Bitmap. Para finalizar, a JPEG se mostrou com mais eficiência, levando em média 0,52 segundos para ser colocada no servidor, a perda de qualidade pode ser controlada, sem impactos para a análise do Sistema proposto e a extração do Vetor de Características. Com o JPEG conseguiu-se a preservação de todas as informações que permitirão a reconstrução exata da imagem. Até escolhendo-se 99% de qualidade, o máximo, o tamanho do arquivo fica menor que o formato Bitmap. O que garantiu a sua eficiência na transmissão do arquivo para o servidor.

6.3 JITTER

O *Jitter* é uma métrica importante quando se discute desempenho em uma rede. O seu acompanhamento e controle são fundamentais para as aplicações de tempo real. Neste trabalho, o *Jitter* foi medido pelo resultado de tráfego de dados gerado para a realização dos testes na transmissão, como no recebimento das informações da página, ou seja, pode ser entendido como a variação do retardo na entrega de dados em uma rede. A Figura 6.2 mostra o comportamento desta métrica para o *Download* e *Upload*. Os valores médios aceitáveis desta métrica é de 5 milissegundos para *Download* e 3 milissegundos para *Upload* (COLOURIS, 2007).

Figura 6. 2- Variação no atraso do *Download* e *Upload*

Tabela 6. 2 - Estatística de *Jitter* em *Upload* e *Download*

Estatísticas de <i>Jitter</i> (ms)			
<i>Upload</i>		<i>Download</i>	
Média	2,31	Média	5,27
Desvio Padrão	4,51	Desvio Padrão	3,95
Variância	20,35	Variância	15,61
Mínimo	0,73	Mínimo	0,75
Máximo	27,00	Máximo	23,00

6.4 Latência (atraso)

A latência ou atraso é o tempo total que um pacote leva para trafegar de TX a RX, ou seja, a latência da rede pode ser entendida como o somatório dos atrasos impostos pela rede e equipamentos utilizados na comunicação. Esse tempo é dado milissegundos (ms). Conforme demonstrado no Capítulo 3, a latência aceitável em uma tecnologia DSL/*Cable* está entre 10ms a 60 ms (COLOURIS, 2007), a Figura 6.3 mostra a latência dos usuários nos testes.

Figura 6. 3 - Latência dos usuários durante os testes

Tabela 6. 3 - Estatística da Latência

Estatísticas da Latência dos Usuários	
Média	38,08
Desvio Padrão	12,60
Variância	158,78
Mínimo	0,89
Máximo	67,00

Conforme a Tabela 6.3, a média de Latência estabeleceu-se em 38,08ms, não ocasionando problemas na recuperação das imagens em Rede.

6.5 Perda de Pacotes

As redes IP não garantem a entrega dos pacotes. Este fato deve-se aos fortes requisitos de atraso impostos pelas aplicações interativas em tempo real, portanto, a perda de pacotes é inevitável e pode influenciar significativamente a qualidade do serviço da rede. Então, a perda de pacotes é definida como a porcentagem de pacotes transmitidos que não chegam ao destino. Nesta situação, não houve perda de pacote detectada entre o cliente e Servidor que comprometesse o experimento.

6.6 Download e Upload

Para o teste de *Upload* e *Download*, o usuário pôde escolher qualquer imagem dentre as 9527 disponibilizadas. As imagens podem variar de tamanho, sendo a menor com 14KB e a maior com 85KB, no formato JPEG. Cada usuário poderia realizar quatro testes nas três bases, utilizando todas as bases, totalizando 1272 testes. A Figura 6.4 mostra o tempo de *upload* (em milissegundos) de cada imagem escolhida pelo usuário para iniciar o processo. A análise de suas estatísticas informa que a média varia em torno de um valor aceitável para o usuário final, admitindo-se que cada usuário possui uma conexão com uma velocidade específica. A Tabela 6.4 mostra o comportamento desta métrica nas quatro imagens envolvidas.

Figura 6.4 - Tempo em milissegundos de *upload* das imagens

Tabela 6.4 - Estatísticas de *Upload* de imagens

Upload das Imagens (ms)							
Imagem 1		Imagem 2		Imagem 3		Imagem 4	
Média	1.369	Média	1.461	Média	1.418	Média	1.322
Desvio Padrão	668	Desvio Padrão	603	Desvio Padrão	645	Desvio Padrão	613
Variância	446.138	Variância	364.107	Variância	415.846	Variância	376.270
Mínimo	255	Mínimo	226	Mínimo	369	Mínimo	202
Máximo	4.000	Máximo	2.553	Máximo	2.712	Máximo	2.765

As figuras 6.5, 6.6, 6.7 e 6.8 traduzem a eficiência do *Download* das imagens 1, 2, 3 e 4 seleccionadas pelos usuários.

Figura 6. 5 - *Download* da 1ª imagem do usuário em ms

Tabela 6. 5 - Estatísticas de *Download* da 1ª imagem do usuário

Imagem 1					
Base <i>MatLab</i>		Base MYSQL		Base <i>Slim-tree</i>	
Média	13.754	Média	12.185	Média	9.087
Desvio Padrão	6.968	Desvio Padrão	6.191	Desvio Padrão	5.066
Variância	48.552.336	Variância	38.330.051	Variância	25.664.865
Mínimo	607	Mínimo	513	Mínimo	395
Máximo	36.870	Máximo	32.826	Máximo	34.100

Figura 6. 6 - *Download* da 2ª imagem do usuário em ms

Tabela 6. 6 - Estatísticas de *Download* da 2ª imagem do usuário

Imagem 2					
Base <i>MatLab</i>		Base <i>MYSQL</i>		Base <i>Slim-tree</i>	
Média	1.658	Média	1.491	Média	1.069
Desvio Padrão	629	Desvio Padrão	590	Desvio Padrão	413
Variância	395.851	Variância	347.798	Variância	170.560
Mínimo	420	Mínimo	370	Mínimo	283
Máximo	2.720	Máximo	2.567	Máximo	1.764

Figura 6. 7 - *Download* da 3ª imagem do usuário em msTabela 6. 7 - Estatísticas de *Download* da 3ª imagem do usuário

Imagem 3					
Base <i>MatLab</i>		Base <i>MYSQL</i>		Base <i>Slim-tree</i>	
Média	1.616	Média	1.449	Média	1.045
Desvio Padrão	696	Desvio Padrão	622	Desvio Padrão	444
Variância	484.872	Variância	386.366	Variância	196.881
Mínimo	420	Mínimo	364	Mínimo	283
Máximo	2.738	Máximo	2.567	Máximo	1.734

Figura 6. 8 - *Download* da 4ª imagem do usuário em ms

Tabela 6. 8 - Estatísticas de *Download* da 4ª imagem do usuário

Imagem 4					
Base <i>MatLab</i>		Base MYSQL		Base <i>Slim-tree</i>	
Média	1.496	Média	1.337	Média	961
Desvio Padrão	646	Desvio Padrão	583	Desvio Padrão	417
Variância	416.721	Variância	339.615	Variância	173.747
Mínimo	423	Mínimo	400	Mínimo	283
Máximo	2.703	Máximo	2.516	Máximo	1.768

As métricas acima apresentadas deixam em evidência a eficiência de transmissão do *download* utilizando o Java com a base *Slim-tree*.

6.7 Métricas de Reengenharia

Quando se pretende avaliar o desempenho de um sistema, primeiro desafio está em definir a técnica mais adequada para essa avaliação.

Com base neste estudo, foi realizado um *Benchmark* com as três propostas de sistemas, ou seja, a utilização do *MatLab* utilizando uma planilha e ordenação (projeto inicial de GRACIOSO (2008), comparando-se com uma reengenharia de projeto e o desenvolvimento de um sistema mais atualizado, usando a Linguagem Java com a base desenvolvida em Mysql e o mesmo código desenvolvido em Java com a Base *Slim-tree*. Com estes três modelos chegou-se ao melhor desempenho para o usuário. A Figura 6.9 mostra essas divergências.

Figura 6. 9 - Desempenho de cada base de dados na reengenharia

A Figura 6.10 mostra o grau de eficácia na utilização da Linguagem Java com a base *Slim-tree* nas imagens selecionadas pelos usuários. Salientando-se que o maior desafio nesse desempenho está na tecnologia de transmissão.

Figura 6. 10 - Comparação entre os tipos de Sistemas

6.8 Métricas de Orientação a Objetos

No Capítulo 3 foram descritas as métricas de *software* e ficou definido que seria utilizada a Métrica de Orientação a Objetos. Portanto, aplicando os conceitos, encontram-se os dados apurados no Código desenvolvido em Java.

a) WMC (*Weighted Methods Per Class*): Cálculo do número de serviços por classe. Na Figura 6.11 mostra a Classe “**ConnectionManager**” e a direita o número de serviços dessa classe, representada pela subclasse “**getConnection()**” e “**CloseAll()**”.

Figura 6. 11 - Número de serviços na Classe ConnectionManager

Na classe *HaarDemo*, Figura 6.12, os serviços são: “teste()”, “vetorizar()”, “calculoEuclidiano()”, “updateimagem()”.

Figura 6.12 - Número de serviços na Classe na HaarDemo

Na Classe *ImagemHash*, Figura 6.13, os serviços são: “getDescrImagem()”, “setDescrImagem()”, “SetDistancia()”, “GetIdImagem()”, “SetIdImagem()”, “getVet()”, “SetVet()”

Figura 6.13 - Número de serviços na Classe na ImagemHash

b) DIT (*Depth of Inheritance Tree*): é o número máximo de superclasses posicionadas hierarquicamente acima da classe em questão. Na Figura 6.14 demonstra que a classe *ConnectionManager* tem apenas uma Superclasse na herança, ou seja, a *HaarDemo*.

Figura 6. 14 - número máximo de superclasses (DIT)

c) NOC (Number of Children): número de subclasses posicionadas imediatamente abaixo da superclasse em questão ou filhos diretos. Na Figura 6.15 mostra a *Haar-Demo* como superclasse e como subclasse a *ConnectionManager*, então o **NOC(A)** = 1.

Figura 6. 15 - Número de subclasses abaixo da superclasse

d) CBO (Coupling between object classes): mostra qual é o nível de acoplamento entre as classes da aplicação. No caso da Figura 6.16, a mensagem é disparada quando o usuário acessa a página e faz o *upload* e neste caso o nível de acoplamento entre as classe ocorre pela classe “Start()” que chama as de “Teste()”, “getDescrImagem()”, que na situação abaixo o **CBO(Upload)** = 3.

Figura 6. 16 - Nível de acoplamento entre as classes da aplicação.

e) LCOM (Lack of Cohesion of Methods): número de acesso a um ou mais atributos em comum pelos serviços da própria classe.

Nesta situação, não está havendo intersecção de atributos comuns na mesma classe.

f) RFC (Response for a class): indica a capacidade de resposta que a classe tem ao receber mensagens de seus objetos. Na Figura 6.17, pode-se observar:

CBO (Upload) = 3;

RFC (Upload) = 15.

Figura 6. 17 - Resultado da Métrica RFC

Em suma, nesse projeto os itens aferidos pelo CK estão dentro dos padrões esperados de métricas para orientação a Objetos.

6.9 Considerações sobre este capítulo

A metodologia proposta neste trabalho demonstrou que a troca da ferramenta de desenvolvimento para uma tecnologia (Java) apresenta resultados promissores.

A utilização das métricas de rede está evidenciada nos cenários desenvolvidos com os 105 usuários testados. Podem-se aplicar as métricas de rede para avaliar o desempenho de busca de imagens na rede.

Os dados para criação dos gráficos e tabelas estão disponíveis nos apêndices e anexos deste trabalho.

Capítulo 7 – Conclusões Gerais

7.1 Considerações Gerais e Contribuições

Este trabalho foi focado em três áreas principais para avaliar o desempenho de busca de imagens por conteúdo em redes de computadores, ou seja:

- Estabelecimento de métricas para desempenho de recuperação de imagens médicas baseadas em conteúdo em redes de computadores.
- Métricas de *software*, reengenharia do sistema, troca da linguagem de programação do MatLab para Java (desenvolvimento de API).
- Alteração da estrutura de armazenamento dos vetores de características e *Benchmark* entre as bases de dados.

As conclusões de cada uma destas áreas, pode-se ver a seguir.

7.1.1 Estabelecimento de métricas para desempenho de recuperação de imagens médicas baseadas em conteúdo em redes de computadores.

Em uma rede onde o desempenho é prioridade (como é o caso do cenário utilizado demonstrado no capítulo 6), obter baixas taxas de atraso é essencial para fazer o uso do sistema desenvolvido no *upload* ou *Download* das imagens para o usuário.

A primeira métrica mensurada foi o *Jitter* (figura 6.2). Não se observou variação de atraso de pacotes. COULOURIS (2007) destaca que o *Jitter* introduzido na rede deve ser menor do que 50 milissegundos. A comparação entre os tempos em milissegundos ocorre conforme a figura 7.1. No cenário medido, houve a média de 2,31 milissegundos no *upload* e 5,27 milissegundos no *Download*. No caso do usuário 189.46.181.3 e 201.93.221.215 (figura 6.2), observou-se que o valor da métrica se deu por uma mudança de rota causada por um congestionamento no provedor de serviços.

Figura 7. 1 - Comparação do Jitter de Upload e Download com a média de Jitter

A segunda métrica foi a latência da rede, que pode ser entendida como o somatório dos atrasos impostos pela rede e equipamentos utilizados na comunicação. Do ponto de vista da aplicação, a latência (atrasos) resulta em um tempo de resposta (tempo de entrega da informação - pacotes) para a aplicação. De acordo com a tabela 3.1, a média de latência aceitável em uma conexão DSL/ Cable, está fixada entre 10 milissegundos até 60 milissegundos. A média foi de 38,08%. Observa-se que os *hosts* são um fator importante para a qualidade do serviço entregue ao usuário, que neste cenário, mesmo que controlado, não impactaram no tempo de resposta do sistema.

O terceiro item na métrica de rede é a perda de pacote, que ocorre especialmente no descarte de pacotes nos roteadores ou durante a transmissão dos mesmos. De maneira geral, as perdas de pacotes em redes IP são um problema sério para determinadas aplicações (como neste estudo, consulta de imagens). Neste caso específico, a perda de pacotes implica numa perda de qualidade eventualmente não aceitável para a aplicação. Entretanto, no cenário estabelecido no capítulo 5, a perda de pacote não impactou no desempenho do sistema.

Os resultados obtidos foram mais baixos que as médias recomendadas (COULOURIS, 2007), dando um grau de confiabilidade ao método utilizado nos teste que podem ser facilmente reproduzidos, já que o cenário de teste foi baseado em um *software* desenvolvido na linguagem Java. Assim também, as respostas que o sistema desenvolvido teve diante do cenário de rede, com probabilidades de perdas de pacotes e atrasos foi satisfatória no padrão estabelecido por COULOURIS (2007), pois o sistema conseguiu manter um controle dinâmico e otimizado na utili-

zação dos canais de comunicações. Os resultados provam essa viabilidade através de um cenário real e com possível aplicação prática.

7.1.1 Métricas de software, reengenharia do sistema, troca do MatLab pelo Java (desenvolvimento de API).

Fundamentado nos conceitos de objetos e classes, a programação orientada a objetos representa uma evolução no desenvolvimento de *software* (comparada ao desenvolvimento estruturado), trazendo inúmeros benefícios à criação de programas, como reutilização de código, que reduz significativamente os tempos de desenvolvimento, manutenção e execução. O uso das métricas de orientação a objetos (como apresentado no Capítulo 3) mostrou a melhoria do processo de modo global, mostrando e dando evidências objetivas a respeito do modo de desenvolvimento de *software* e propiciando uma melhoria significativa na qualidade do sistema desenvolvido. Este trabalho utilizou métricas e análises que podem ser implantadas em sistemas orientados a objetos. A experiência prática mostra que a utilização de uma metodologia de qualidade de desenvolvimento é essencial para alcançar objetivos das métricas. Diante deste cenário, considera-se, assim atingido o objetivo da fundamentação das métricas, além de citar e executar os pontos mais importantes desta teoria.

No desenvolvimento do sistema foi criada uma API para a execução do sistema. A escolha de uma API se dá pelo fato de garantir o desenvolvimento mais ágil da aplicação, já que ela oferecerá inúmeros serviços como acesso aos bancos de dados, interface com o usuário, conexões, visualização de informações e principalmente processamento. Já a definição da utilização da linguagem Java, ocorreu pelas suas inúmeras bibliotecas e conjunto de ferramentas *open source*.

Na área de desenvolvimento de um sistema, o processo padrão de implementação de melhorias de um *software*, ocorreu neste trabalho, por meio de *Benchmarking*. Ou seja, comparou-se o sistema desenvolvido por GRACIOSO (2008), que se encontra disponível no apêndice A, com o sistema desenvolvido na linguagem Java, que está disponível nos apêndices B e C. A metodologia da comparação pode ser observada no Capítulo 5.

No desenvolvimento do sistema de GRACISO (2008) foram utilizadas 4728 imagens. Na reengenharia, o sistema foi implementado com tecnologia para Internet, utilizando-se de programas livres e linguagem de programação Java, e uma base contendo 9527 imagens para a execução da comparação. A Tabela 7.1 apresenta as médias dos resultados obtidos nas consultas realizadas (dados retirados do apêndice D).

Tabela 7.1 - Tempo médio do *download* de 20 imagens

Tempo médio de <i>download</i> de 20 imagens (milissegundos).			
	Matlab com Planilha	Java com Mysql	Java com <i>Slim-tree</i>
Imagem 1	13754	12185	9.087
Imagem 2	1658	1491	1.069
Imagem 3	1616	1449	1.045
Imagem 4	1496	1337	961
Média	4631	4115	3040

Na tabela 7.1, fica evidente a eficiência do sistema implantado em Java com a base *Slim-tree*, sendo 66% mais rápido no tempo de resposta com relação ao sistema desenvolvido por GRACIOSO (2008).

7.1.3 Alteração da estrutura de armazenamento dos vetores de características, e Benchmark entre as bases de dados.

Quando são feitos pedidos para os servidores em conexões lentas, geralmente existe uma demora na recuperação de objetos remotos. Além deste fator, a alta taxa de transferência de objetos pela rede leva um aumento de tráfego que reduz a largura de banda disponível e atrasos perceptíveis aos usuários.

Neste trabalho foram apresentadas três formas de armazenamento para os vetores de características utilizados nas recuperações de imagens em rede. Com base nessas formas, desenvolveu-se uma análise, com a intenção de descobrir o impacto que cada uma proporcionava no desempenho da recuperação das imagens na rede.

Mediante os testes, foi possível comprovar a grande influência do armazenamento dos vetores de características dentro das bases de dados (Mysql, *Slim-tree*)

no desempenho das consultas, se comparada com a busca sequencial em uma planilha.

Em relação ao armazenamento e consulta de dados, o sistema desenvolvido em Java e a base *Slim-tree* foi o método que apresentou o melhor desempenho dentre os três tratados neste trabalho. Esse método chegou a ser dois minutos e meio mais rápido que o método de armazenamento por planilha estudada em GRACIOSO (2008). O método por desempenho mediante planilha foi o método que, dentre os três utilizados na análise, apresentou o pior desempenho na busca e recuperação dos vetores de características. A média de busca e recuperação na base *Slim-tree* foi de 508 milissegundos, enquanto no Mysql foi de 1255 milissegundos e na planilha levou 27802 milissegundos. Esta diferença é perceptível para o usuário final.

Por fim, a implantação e uso de um sistema de recuperação de imagens em rede apresentou um desempenho muito superior ao ser comparado com o sistema em Matlab (original). Ao usar uma imagem de tamanho mediano (de 512 x 512 codificada em 8 níveis de cinza), os resultados são rápidos com o sistema desenvolvido em Java e a base *Slim-tree*. O desempenho avaliado demonstra a eficiência do sistema e o tempo computacional permanece sempre em limites aceitáveis (COULOURIS, 2007). Portanto, o uso de uma conexão de Internet (*link* acima de 1 Mbit), demonstra viável para um sistema de recuperação de imagens por conteúdo.

7.2 Trabalhos Futuros

O estudo da avaliação do desempenho de busca de imagens por conteúdo em redes de computadores visando a alternativa para melhorar o seu desempenho é complexo e com vários componentes que interferem diretamente nos resultados e objetivos que se deseja atingir. A seguir, são comentadas algumas sugestões de trabalhos futuros, as quais servem para demonstrar as possibilidades de continuação deste assunto.

- Rede
 - Utilização de *cache* no servidor de Web
 - Análise da utilização de um *cache* no Servidor para armazenamento de páginas dinâmicas pré-processadas, causando assim a diminuição do tempo de resposta ao cliente;
 - Priorização das políticas das filas de roteadores

- Os roteadores são responsáveis pela análise e classificação dos pacotes de rede, de acordo com sua prioridade. Entretanto, isto é feito dinamicamente. Portanto, se existir um trabalho no enfileiramento por prioridade, colocando o pacote que sai deste sistema como prioridade, teríamos o tempo de resposta também diminuído.
- Desenvolvimento de Sistemas
 - Neste trabalho foi desenvolvida uma API para fazer a Transformada *Wavelet* de Haar, entretanto, poderia se aprimorar esta API para outras saídas, onde o usuário escolhesse o tipo de transformada desejada, como por exemplo Transformada *Wavelet* Db2, Db3, Db4.
- Base de dados
 - Neste trabalho não foi possível fazer a conversão da *Slim-tree* para a linguagem Java. Contudo, a codificação deste Sistema Gerenciado de Banco de Dados em Java acarretará a melhor performance do sistema para a internet.

Referências

- [CASTAÑÓN, 2003] CASTANÓN, C. A. B. Extração de características em imagens médicas utilizando Wavelets. Dissertação de mestrado em Ciência de Computação e Matemática Computacional, Universidade de São Paulo – USP, São Carlos, 2003.
- [CHAPPELL, 2002] CHAPPELL, D.; JEWELL, T. Java Web Services. O'Reilly. ISBN: 0-596-00269-6; Março, 2002.
- [CHIDAMBER; KEMERER, 1994] CHIDAMBER, S., KEMERER, C.: A metrics suite for object oriented design. IEEE Transactions on Software Engineering 20 (1994) 476–493
- [CHINO, 2004] CHINO, Fabio Jun Takada. Visualizando a organização e o comportamento de estruturas métricas: Aplicações em consultas por similaridade. Dissertação de mestrado em Ciência de Computação e Matemática Computacional, Universidade de São Paulo – USP, São Carlos, 2004.
- [CONTE, 1988] CONTE, S. D.; Software Engineering Metrics and Models. California: Benjamin-Cummings, 1988.
- [COULOURIS, 2007] COULOURIS, George. Sistemas Distribuidos - Conceitos E Projeto. Bookman Companhia Ed, 2007.
- [FONSECA, 1998] FONSECA, J. Compressão de Imagens Médicas Utilizando a Técnica JPEG-DPCM , IV Fórum Nacional de Ciência e Tecnologia em Saúde - IV FNCTS, Curitiba, BR, 1998.
- [GONZALES; WOODS, 2007] GONZALEZ, R. C.; WOODS, R. E. Digital image processing. Reading, MA: Prentice Hall, 2007. 716p
- [GRACIOSO, 2008] GRACIOSO, Ana Carolina Nicolosi da Rocha. Avaliação do desempenho de recuperação de imagens médicas baseada em conteúdo em redes de computadores. 2008. 200 f. Dissertação de mestrado em Engenharia Elétrica, Universidade de São Paulo – USP, São Carlos, 2008.
- [HULQUIST,1997] Hultquist, Steve <ssh@vnet.ibm.com> “FAQ about Client/Server” <http://non.com/news.answers/client-server-faq.html>, 1997.

[ISO; ITU-T, 1992]. Recommendation T.81: Information technology – digital compression and coding of continuous-tone still images – requirements and guidelines. Zurich: ISO|ITU-T, 1992.

[LEHMANN, et al., 2005] LEHMANN, T. M.; GÜLDA, M. O.; DESELAERSB, T.; KEYSERSB, D.; SCHUBERTC, H.; SPITZERA, K.; NEYB, H.; WEINC, B. B. Automatic categorization of medical images for content-based retrieval and data mining. *Computerized Medical Imaging and Graphics*, v. 29, p.143-155, 2005.

[MALLAT, 1989] MALLAT, S. Multifrequency channel decompositions of image and wavelets models. *IEEE Transaction on Acoustics, Speech and Signal Processing*, v. 37, p. 12, p. 2091-2110, December 1989.

[MARSH, 1997] MARSH, A., "EUROMED – The Creation of a Telemedical Information Society", 10' IEE Symposium on Computer Based Medical Systems, Maribor, Slovenia, 1997, pp. 86-91

[MATERKA; STRZELECKI, 1998] MATERKA, A.; STRZELECKI, M. 1998. Texture analysis methods – a review. Cost b11 report, 1-33.

[MCKIE,1997] McKie, Stewart "Everything you wanted to know about Client/Server computing but were afraid to ask" *Eye on Technology*, <http://www.duke.com/controller/Issues/DecJan/Clientse.htm>

[MILLS, 1999] MILLS, Everaldo E. *Software Metrics*. Software Engineering Institute/SEI - Carnegie Mellon University, 1999.

[MÜLLER et al, 2004] MÜLLER, H.; MICHOUX, N.; BANDON, D.; GEISSBUHLER, A. A review of content-based image retrieval system in medical applications-clinical benefits and future direction. *International Journal of Medical Informatics*, v. 73, p. 1-23, 2004.

[ROSENBERG, 2001] ROSENBERG, D. *Use Case Driven Object Modeling with UML: a practical approach*. Massachusetts: Addison Wesley Longman, 1999.

[SIEGEL, 1999b] SIEGEL, E. L., "Current State of the Art and Future Trends", in *Filmless Radiology*, E. L. Siegel and R. M. Kolodner, Eds. New York City, NY: Springer Verlag, 1999a. pp. 3-20.

[SILVA, 2009] SILVA, Marcelo Ponciano. Processamento de consultas por similaridade em imagens médicas visando a recuperação percentual guiada pelo usuário. 117 f. Dissertação (Mestrado em Engenharia Elétrica) – Universidade de São Paulo, São Carlos, 2009.

[TANENBAUM, 2011]. TANENBAUM, A. S. “Redes de Computadores” quinta edição (2011), editora PRENTICE HALL BRASIL.

[TORRES, 2006] TORRES, Ricardo da Silva. Content-based image retrieval: Theory and applications. *Revista de Informatica Teorica e Aplicada*, 2(13):161–185, 2006.

[TRAINA, 2000] C. TRAINA Jr., A. Traina, B. Seeger, and C. Faloutsos, “Slim-Trees: High performance metric trees minimizing overlap between nodes,” *Lecture Notes in Computer Science*, vol. 1777, pp. 51–65, 2000.

[THEODORIDIS, 1999] THEODORIDIS, S. Pattern Recognition, in Academy Press. New York.

[XENOS et al., 2000] XENOS, Th.D., P. SPALLA and C. MITCHELL (2000): Neural network based TEC forecasting models, Paper 0904 presented at AP-2000 Meeting at Davos, Switzerland.

[YOURDON,1996] E. Yourdon. Java, the Web and software development. *IEEE Computer*, 29(8):25-30, Agosto, 1996.

[WANG, 2001] WANG, J. Z. Methodological Review – wavelets and imaging informatics: A Review of the Literature. *Journal of Biomedical Informatics*, page 129-141.

Apêndice A – Código na Linguagem *MatLab* Server

Principal.m

```
clc;
clear all;
tic;

%%%%% Parâmetros a serem configurados
%Diretório das imagens
dir = 'C:\sistema\imagens\banco\';
%Número de Imagens no diretório
NUM=100;
%Níveis do Histograma
%nh = 256;
classes=5;
busca=5;
%%%%% Fim Parâmetros

alfa=zeros(2,32); %vetor com mv
%beta=zeros(NUM,nh); %vetor com histograma
pesquisa=input('Entre com a Imagem a ser Pesquisada:') %imagem de pesquisa

%calcula os atributos de cada imagem
for i=1:NUM,
    nome=[dir sprintf('%d.jpg',i)];
    % le a imagem e coloca em escala de cinza
    %img=rgb2gray(imread(nome));
    %le a imagem
    img = imread(nome);
    beta(i,:)=e_da(img); %energia
    alfa(i,:)=mv_da(img); % média e variância
    %alfa(i,:)=e_da(img); %energia

    %beta(i,:)=histograma(img); %histograma
end
gama = horzcat(alfa,beta); %concatena os vetores alfa e beta

p=alfa(pesquisa,:);
%calcula a distancia euclidiana das imagens
for i=1:NUM,
    dist(i)=euclidiana(p,gama(i,:));
end
%ordena as imagens pela proximidade do atributo de interesse
[valor ordem]=sort(dist);

time = toc;
%
% %Plota as imagens
subplot(4,5,1);imshow([dir sprintf('%d.jpg',ordem(1))]);title(['Imagem: ',
sprintf('%d',ordem(1))]);
subplot(4,5,2);imshow([dir sprintf('%d.jpg',ordem(2))]);title(sprintf('%d',ordem(2)));
subplot(4,5,3);imshow([dir sprintf('%d.jpg',ordem(3))]);title(sprintf('%d',ordem(3)));
subplot(4,5,4);imshow([dir sprintf('%d.jpg',ordem(4))]);title(sprintf('%d',ordem(4)));
subplot(4,5,5);imshow([dir sprintf('%d.jpg',ordem(5))]);title(sprintf('%d',ordem(5)));

subplot(4,5,6);imshow([dir sprintf('%d.jpg',ordem(6))]);title(sprintf('%d',ordem(6)));
subplot(4,5,7);imshow([dir sprintf('%d.jpg',ordem(7))]);title(sprintf('%d',ordem(7)));
subplot(4,5,8);imshow([dir sprintf('%d.jpg',ordem(8))]);title(sprintf('%d',ordem(8)));
subplot(4,5,9);imshow([dir sprintf('%d.jpg',ordem(9))]);title(sprintf('%d',ordem(9)));
subplot(4,5,10);imshow([dir sprintf('%d.jpg',ordem(10))]);title(sprintf('%d',ordem(10)));

subplot(4,5,11);imshow([dir sprintf('%d.jpg',ordem(11))]);title(sprintf('%d',ordem(11)));
subplot(4,5,12);imshow([dir sprintf('%d.jpg',ordem(12))]);title(sprintf('%d',ordem(12)));
subplot(4,5,13);imshow([dir sprintf('%d.jpg',ordem(13))]);title(sprintf('%d',ordem(13)));
subplot(4,5,14);imshow([dir sprintf('%d.jpg',ordem(14))]);title(sprintf('%d',ordem(14)));
subplot(4,5,15);imshow([dir sprintf('%d.jpg',ordem(15))]);title(sprintf('%d',ordem(15)));

subplot(4,5,16);imshow([dir sprintf('%d.jpg',ordem(16))]);title(sprintf('%d',ordem(16)));
subplot(4,5,17);imshow([dir sprintf('%d.jpg',ordem(17))]);title(sprintf('%d',ordem(17)));
subplot(4,5,18);imshow([dir sprintf('%d.jpg',ordem(18))]);title(sprintf('%d',ordem(18)));
subplot(4,5,19);imshow([dir sprintf('%d.jpg',ordem(19))]);title(sprintf('%d',ordem(19)));
subplot(4,5,20);imshow([dir sprintf('%d.jpg',ordem(20))]);title(sprintf('%d',ordem(20)));
```

eda.m

```

%calcula a energia da imagem em três escalas wavelets (base de Haar)
% e retorna um vetor de dimensão 6
function v = e_da(M)
%primeira escala wavelet
[cA,cH,cV,cD] = dwt2(double(M),'Haar'); %aplica a base de wavelet Haar à imagem
%Segundo escala wavelet
[cAA,cHA,cVA,cDA] = dwt2(cA,'Haar'); %aplica a base de wavelet Haar à imagem
%terceira escala wavelet
[cAAA,cHAA,cVAA,cDAA] = dwt2(cAA,'Haar'); %aplica a base de wavelet Haar à imagem
%quarta escala wavelet
[cAAAA,cHAAA,cVAAA,cDAAA] = dwt2(cAAA,'Haar'); %aplica a base de wavelet Haar à imagem
v(1)=energia(cH);
v(2)=energia(cV);
v(3)=energia(cD);
v(4)=energia(cA);

v(5)=energia(cHA);
v(6)=energia(cVA);
v(7)=energia(cDA);
v(8)=energia(cAA);

v(9)=energia(cHAA);
v(10)=energia(cVAA);
v(11)=energia(cDAA);
v(12)=energia(cAAA);

v(13)=energia(cHAAA);
v(14)=energia(cVAAA);
v(15)=energia(cDAAA);
v(16)=energia(cAAAA);

```

Mv_da.m

```

%calcula a média e variancia nas escalas wavelet de um sinal

function v = mv_da(M)

%primeira escala wavelet
[cA,cH,cV,cD] = dwt2(double(M),'Haar'); %aplica a base de wavelet Haar à imagem

%Segundo escala wavelet
[cAA,cHA,cVA,cDA] = dwt2(cA,'Haar'); %aplica a base de wavelet Haar à imagem

%terceira escala wavelet
[cAAA,cHAA,cVAA,cDAA] = dwt2(cAA,'Haar'); %aplica a base de wavelet Haar à imagem

%quarta escala wavelet
[cAAAA,cHAAA,cVAAA,cDAAA] = dwt2(cAAA,'Haar'); %aplica a base de wavelet Haar à imagem

v(1)=mean(mean(cH));
v(2)=variancia(cH);
v(3)=mean(mean(cV));
v(4)=variancia(cV);
v(5)=mean(mean(cD));
v(6)=variancia(cD);
v(7)=mean(mean(cA));
v(8)=variancia(cA);

v(9)=mean(mean(cHA));
v(10)=variancia(cHA);
v(11)=mean(mean(cVA));

```

```
v(12)=variancia(cVA);  
v(13)=mean(mean(cDA));  
v(14)=variancia(cDA);  
v(15)=mean(mean(cAA));  
v(16)=variancia(cAA);  
  
v(17)=mean(mean(chAA));  
v(18)=variancia(chAA);  
v(19)=mean(mean(cVAA));  
v(20)=variancia(cVAA);  
v(21)=mean(mean(cDAA));  
v(22)=variancia(cDAA);  
v(23)=mean(mean(cAAA));  
v(24)=variancia(cAAA);  
  
v(25)=mean(mean(chAAA));  
v(26)=variancia(chAAA);  
v(27)=mean(mean(cVAAA));  
v(28)=variancia(cVAAA);  
v(29)=mean(mean(cDAAA));  
v(30)=variancia(cDAAA);  
v(31)=mean(mean(cAAAA));  
v(32)=variancia(cAAAA);
```

Apêndice B – Código na Linguagem JAVA (Reengenharia)

HaarDemo.Java

```
package fbo;
```

```
import java.sql.PreparedStatement;
import java.awt.image.BufferedImage;
import java.awt.image.Raster;
import java.io.*;
import java.sql.Connection;
import java.sql.ResultSet;
import java.text.DecimalFormat;
import java.util.ArrayList;
```

```
public class HaarDemo {
```

```
    public static String transformadaHaar(double[][] mat) throws IOException {
        double[][] matr = new double[256][256];
        double[][] mat2 = new double[128][128];
        double[][] mat3 = new double[64][64];
        double[][] mat4 = new double[32][32];
        double[][] mat5 = new double[16][16];
        double[][] mat6 = new double[8][8];
        double[][] mat7 = new double[4][4];
        String retorno = "";
        for (int x = 0; x < 256; x++) {
            for (int y = 0; y < 256; y++) {
                matr[x][y] = (mat[2 * x][2 * y] + mat[2 * x][2 * y + 1] + mat[2 * x + 1][2 *
y] + mat[2 * x + 1][2 * y + 1]) / Math.sqrt(2);
            }
        }
        for (int x = 0; x < 128; x++) {
            for (int y = 0; y < 128; y++) {
                mat2[x][y] = (matr[2 * x][2 * y] + matr[2 * x][2 * y + 1] + matr[2 * x + 1][2
* y] + matr[2 * x + 1][2 * y + 1]) / Math.sqrt(2);
            }
        }
        for (int x = 0; x < 64; x++) {
            for (int y = 0; y < 64; y++) {
                mat3[x][y] = (mat2[2 * x][2 * y] + mat2[2 * x][2 * y + 1] + mat2[2 * x + 1][2
* y] + mat2[2 * x + 1][2 * y + 1]) / Math.sqrt(2);
            }
        }
        for (int x = 0; x < 32; x++) {
            for (int y = 0; y < 32; y++) {
                mat4[x][y] = (mat3[2 * x][2 * y] + mat3[2 * x][2 * y + 1] + mat3[2 * x + 1][2
* y] + mat3[2 * x + 1][2 * y + 1]) / Math.sqrt(2);
            }
        }
        for (int x = 0; x < 16; x++) {
            for (int y = 0; y < 16; y++) {
                mat5[x][y] = (mat4[2 * x][2 * y] + mat4[2 * x][2 * y + 1] + mat4[2 * x + 1][2
* y] + mat4[2 * x + 1][2 * y + 1]) / Math.sqrt(2);
            }
        }
        for (int x = 0; x < 8; x++) {
            for (int y = 0; y < 8; y++) {
                mat6[x][y] = (mat5[2 * x][2 * y] + mat5[2 * x][2 * y + 1] + mat5[2 * x + 1][2
* y] + mat5[2 * x + 1][2 * y + 1]) / Math.sqrt(2);
            }
        }
        for (int x = 0; x < 4; x++) {
            for (int y = 0; y < 4; y++) {
```

```

        mat7[x][y] = 1 / (mat6[2 * x][2 * y] + mat6[2 * x][2 * y + 1] + mat6[2 * x +
1][2 * y] + mat6[2 * x + 1][2 * y + 1]) / Math.sqrt(2);
        //System.out.printf("%11.9f;", mat7[x][y]);
        DecimalFormat df = new DecimalFormat("0.000000000");
        retorno+= df.format(mat7[x][y]) + ";";
    }
    retorno+= "\n";
}

//System.out.print("\n");
return retorno;
}

public static String teste(BufferedImage imagem) {
    Raster raster = null;
    /* try {
        //System.out.print("Imagem " + z + ": ");
        imagem = ImageIO.read(new File(z + ".jpg"));
    } catch (IOException ex) {
        ex.printStackTrace();
    }
    int x, y = 0;
    */
    try {
        raster = imagem.getRaster();
        double cores[] = new double[255];
        double matriz_main[][] = new double[512][512];
        for (int x = 0; x < imagem.getWidth(); x++) {
            for (int y = 0; y < imagem.getHeight(); y++) {
                raster.getPixel(x, y, cores);
                matriz_main[x][y] = (cores[0] + cores[1] + cores[2]) / 3;
            }
        }

        return transformadaHaar(matriz_main);
    } catch (Exception ex) {
        StackTraceElement[] ste = ex.getStackTrace();
        int i = 0;
        while (ste[i] != null) {
            System.out.println(ste[i].toString());
            i++;
        }
        return null;
    }
}
/*
public static void main(String argv[]) throws IOException {

    int z = 0;

    for (z = 1; z <= 4728; z++) {
        BufferedImage imagem = null;
        Raster raster = null;
        try {
            System.out.print("Imagem " + z + ": ");
            imagem = ImageIO.read(new File(z + ".jpg"));
        } catch (IOException ex) {
            ex.printStackTrace();
        }
        int x, y = 0;

        raster = imagem.getRaster();
        double cores[] = new double[255];
        double matriz_main[][] = new double[512][512];
        for (x = 0; x < imagem.getWidth(); x++) {
            for (y = 0; y < imagem.getHeight(); y++) {
                raster.getPixel(x, y, cores);

```

```

matriz_main[x][y] = (cores[0] + cores[1] + cores[2]) / 3;
}
}

transformadaHaar(matriz_main);
}

} // main*/

public static String vetorizar(BufferedImage imagem) throws IOException {
    int z = 0;
    String retorno = "";
    Raster raster = null;
    for (z = 1; z <= 1; z++) {

        int x, y = 0;

        raster = imagem.getRaster();
        double cores[] = new double[255];
        double matriz_main[][] = new double[512][512];
        for (x = 0; x < imagem.getWidth(); x++) {
            for (y = 0; y < imagem.getHeight(); y++) {
                raster.getPixel(x, y, cores);
                matriz_main[x][y] = (cores[0] + cores[1] + cores[2]) / 3;
            }
        }

        retorno+= transformadaHaar(matriz_main);
    }
    return retorno;
}

public static Double calculoEuclidiano(double[] vetBD, double[] vet){
    double retorno = 0;
    int i = 0;
    //BigDecimal vet = new BigDecimal(0);
    double acumulado = 0;
    while(vetBD[i]!=0){
        double aux = Math.pow((vet[i] - vetBD[i]), 2);
        acumulado+= aux;
        i++;
    }
    retorno = Math.sqrt(acumulado);

    return retorno;
}

public static ArrayList updateImagem(double[] valor, int limit) throws Exception{
    ArrayList retorno = null;
    String leitura = "select * from imagem";
    String sqlUpdate = "update imagem set distancia = ? where idImagem = ?";
    Connection conn = null;
    PreparedStatement pstmt = null;
    ResultSet result = null;

    conn = ConnectionManager.getConnection();
    pstmt = conn.prepareStatement(leitura);
    result = pstmt.executeQuery();
    ImagemHash ih = new ImagemHash();
    int i = 0;
    while(result.next()){
        ih.setIdImagem(result.getInt("idImagem"));
        ih.setVet(result.getDouble("vt"+(i+1)), 0);
        ih.setVet(result.getDouble("vt"+(i+1)), 1);
        ih.setVet(result.getDouble("vt"+(i+1)), 2);
        ih.setVet(result.getDouble("vt"+(i+1)), 3);
        ih.setVet(result.getDouble("vt"+(i+1)), 4);
    }
}

```

```

        ih.setVet(result.getDouble("vt" + (i + 1)), 5);
        ih.setVet(result.getDouble("vt" + (i + 1)), 6);
        ih.setVet(result.getDouble("vt" + (i + 1)), 7);
        ih.setVet(result.getDouble("vt" + (i + 1)), 8);
        ih.setVet(result.getDouble("vt" + (i + 1)), 9);
        ih.setVet(result.getDouble("vt" + (i + 1)), 10);
        ih.setVet(result.getDouble("vt" + (i + 1)), 11);
        ih.setVet(result.getDouble("vt" + (i + 1)), 12);
        ih.setVet(result.getDouble("vt" + (i + 1)), 13);
        ih.setVet(result.getDouble("vt" + (i + 1)), 14);
        ih.setVet(result.getDouble("vt" + (i + 1)), 15);

        double distancia = HaarDemo.calculoEuclidiano(ih.getVet(), valor);
        pstmt = conn.prepareStatement(sqlUpdate);
        pstmt.setDouble(1, distancia);
        pstmt.setLong(2, ih.getIdImagem());
        pstmt.executeUpdate();

    }
    String sql = "select * from imagem order by distancia limit ?";
    pstmt = conn.prepareStatement(sql);
    pstmt.setInt(1, limit);
    result = pstmt.executeQuery();
    if(result != null){
        retorno = new ArrayList();
    }
    i = 0;
    while(result.next()){
        ImagemHash ih2 = new ImagemHash();
        ih2.setIdImagem(result.getLong("idImagem"));

        retorno.add(ih2);
    }
    return retorno;
}
} // haarDemo

```

Upload.java

```

package fbo;

import java.io.File;
import java.io.IOException;
import java.io.PrintWriter;
import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;
import javax.imageio.ImageIO;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import org.apache.commons.fileupload.FileItem;
import org.apache.commons.fileupload.FileItemFactory;
import org.apache.commons.fileupload.disk.DiskFileItemFactory;
import org.apache.commons.fileupload.servlet.ServletFileUpload;

/**
 *
 * @author Fabio Brussolo de Oliveira
 */
public class Upload extends HttpServlet {

```



```

/**
 * Processes requests for both HTTP <code>GET</code> and <code>POST</code> methods.
 * @param request servlet request
 * @param response servlet response
 * @throws ServletException if a servlet-specific error occurs
 * @throws IOException if an I/O error occurs
 */
protected void processRequest(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
    response.setContentType("text/html;charset=UTF-8");
    PrintWriter out = response.getWriter();
    try {
        /* TODO output your page here
        out.println("<html>");
        out.println("<head>");
        out.println("<title>Servlet Upload</title>");
        out.println("</head>");
        out.println("<body>");
        out.println("<h1>Servlet Upload at " + request.getContextPath () + "</h1>");
        out.println("</body>");
        out.println("</html>");
        */
    } finally {
        out.close();
    }
}

// <editor-fold defaultstate="collapsed" desc="HttpServletRequest methods. Click on the + sign on
the left to edit the code.">
/**
 * Handles the HTTP <code>GET</code> method.
 * @param request servlet request
 * @param response servlet response
 * @throws ServletException if a servlet-specific error occurs
 * @throws IOException if an I/O error occurs
 */
@Override
protected void doGet(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
    processRequest(request, response);
}

/**
 * Handles the HTTP <code>POST</code> method.
 * @param request servlet request
 * @param response servlet response
 * @throws ServletException if a servlet-specific error occurs
 * @throws IOException if an I/O error occurs
 */
@Override
protected void doPost(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
    doAction(request, response);
}

protected void doAction(HttpServletRequest request, HttpServletResponse response) throws
IOException {
    long init;
    long end;
    long diff;

    init = System.currentTimeMillis();

    try {

        String cod = "";

```

```

String teste = "";
boolean isMultipart = ServletFileUpload.isMultipartContent(request);
//limite de resultados
int limit = 10;
if (isMultipart) {
    FileItemFactory factory = new DiskFileItemFactory();
    ServletFileUpload upload = new ServletFileUpload(factory);
    List items = upload.parseRequest(request);
    Iterator itr = items.iterator();
    while (itr.hasNext()) {
        FileItem item = (FileItem) itr.next();
        if (item.isFormField()) {
            String name = item.getFieldName();
            String value = item.getString();
            if (name.equals("cod")) {
                cod = value;
            }
        } else {
            String itemName = item.getName().toString();
            String nomeArquivo = itemName.substring(itemName.lastIndexOf("\\") +
1);

            String dirName = "C:\\FBO_v1\\web\\file_uploads\\";
            File savedFile = new File(dirName + cod + "_" + nomeArquivo);

            item.write(savedFile);
            teste = HaarDemo.vetorizar(ImageIO.read(new File(dirName + cod + "_" +
nomeArquivo)));
        }
    }

    teste = teste.replace(",", ".");
    teste = teste.replace("\n", "");
    String[] vet = teste.split(";");

    int i = 0;
    double[] vetDouble = new double[vet.length];
    while (i < vet.length) {
        vetDouble[i] = (double) Double.parseDouble(vet[i]);
        System.out.println(vet[i]);
        i++;
    }
    //request.getSession().setAttribute("teste", teste);
    //response.sendRedirect("teste.jsp");
    ArrayList imgs = HaarDemo.updateImagem(vetDouble, limit);
    end = System.currentTimeMillis();
    diff = end - init;
    String tmp = "Página processada em " + diff + " segundos";
    if (imgs != null) {
        request.getSession().setAttribute("imgs", imgs);
        request.getSession().setAttribute("tmp", tmp);
        response.sendRedirect("imagensProximas.jsp");
    } else {
        request.getSession().setAttribute("imgs", imgs);
        response.sendRedirect("imagensProximas.jsp");
    }
} catch (Exception ex) {
    String except = ex.toString();
    StackTraceElement[] ste = ex.getStackTrace();
    /*int i = 0;
    while(ste[i] != null){
        except+= ste[i].toString()+"<br>";
        i++;
    }*/
    request.getSession().setAttribute("except", except);
    response.sendRedirect("msgError.jsp");
}

```

```

    }
}

```

ConnectionManager.Java

```

package fbo;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;

public class ConnectionManager {

    /**
     * Os valores das constantes abaixo devem estar de acordo com o banco de
     * dados utilizado.
     */
    private static final String STR_DRIVER = "com.mysql.jdbc.Driver";
    private static final String DATABASE = "test";
    private static final String IP = "localhost";
    private static final String STR_CON = "jdbc:mysql://" + IP + ":3306/" + DATABASE;
    private static final String USER = "root";
    private static final String PASSWORD = "euc7656";

    /**
     * Metodo responsavel por obter conexao com o BD
     * @return objeto de conexao com o BD
     * @throws Exception
     */
    public static Connection getConnection() throws Exception {
        Connection conn = null;
        try {
            Class.forName(STR_DRIVER);
            conn = DriverManager.getConnection(STR_CON, USER, PASSWORD);
            System.out.println("[ConnectionManager]: Obtendo conexao");
            return conn;
        } catch (ClassNotFoundException e) {
            throw new Exception("Driver nao encontrado", e);
        } catch (SQLException e) {
            throw new Exception("Erro ao obter a conexao", e);
        }
    }

    /**
     * Metodo responsavel por fechar a conexao com o BD
     * @param conn objeto de conexao
     */
    public static void closeAll(Connection conn) {
        try {
            if (conn != null) {
                conn.close();
            }
        } catch (Exception e) {
            System.out.println("----> Nao foi possivel fechar a conexao com o banco");
            System.out.println("----> Detalhamento do erro: ");
            e.printStackTrace();
        }
    }

    /**
     * Metodo responsavel por fechar a conexao com o BD e Statement

```

```

    * @param conn objeto de conexao
    * @param stmt interface Statement
    */
    public static void closeAll(Connection conn, Statement stmt) {
        try {
            if (stmt != null) {
                stmt.close();
            }
            closeAll(conn);
        } catch (Exception e) {
            System.out.println("---> Nao foi possivel fechar a conexao com o banco");
            System.out.println("---> Detalhamento do erro: ");
            e.printStackTrace();
        }
    }

    /**
     * Metodo responsavel por fechar a conexao com o BD, Statement e ResultSet
     * @param conn objeto de conexao
     * @param stmt interface Statement
     * @param rs ResultSet
     */
    public static void closeAll(Connection conn, Statement stmt, ResultSet rs) {
        try {

            if (rs != null) {
                rs.close();
            }
            closeAll(conn, stmt);
        } catch (SQLException ex) {
            StackTraceElement[] ste = ex.getStackTrace();
            int i = 0;
            while(ste[i] != null){
                System.out.println(ste[i].toString());
                i++;
            }
        }
    }

}

```

ImagensProxima.Java

```

<!--
    Document    : imagensProximas
    Created on  : 06/06/2011, 20:53:01
    Author      : Fabio
-->
<%@page import="fbo.ImagemHash"%>
<%@page import="java.util.Iterator"%>
<%@page import="java.util.ArrayList" %>
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
    "http://www.w3.org/TR/html4/loose.dtd">

<%
    ArrayList imgs = (ArrayList)request.getSession().getAttribute("imgs");
    request.getSession().removeAttribute("imgs");
    String tmpExec = (String)request.getSession().getAttribute("tmp");
    request.getSession().removeAttribute("tmp");
%>

```

```

<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>Imagens Semelhantes</title>
  </head>
  <body>
    <%
      if(imgs != null){
        %>
        <table>
          <tr>
            <td>Imagens ordenada por maior semelhança</td>
          </tr>
          <%
            for(Iterator it = imgs.iterator(); it.hasNext();){
              Object object = it.next();
              ImagemHash ih = (ImagemHash)object;
              %>
              <tr>
                <td>
                  Id da imagem = <%=ih.getIdImagem()%>
                </td>
                <td>
                  
                </td>
              </tr>
            <%
          }
          %>
        </table>
        <%
      }else{
        %>
        Ocorreu um erro e a comparação das imagens não pode ser processada com sucesso!!!
        <%
      }
    %>
    <%=tmpExec%>
  </body>
</html>

```

ImagemHash.Java

```

/**
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */

package fbo;

/**
 *
 * @author FBO
 */
public class ImagemHash {
    private long idImagem;
    private String descrImagem;
    private double vet[] = new double[17];
    private double distancia;

    public ImagemHash() {
    }
}

```

```

    public ImagemHash(long idImagem, String descrImagem, double distancia) {
        this.idImagem = idImagem;
        this.descrImagem = descrImagem;
        this.distancia = distancia;
    }

    public ImagemHash(String descrImagem, double distancia) {
        this.descrImagem = descrImagem;
        this.distancia = distancia;
    }

    public String getDescrImagem() {
        return descrImagem;
    }

    public void setDescrImagem(String descrImagem) {
        this.descrImagem = descrImagem;
    }

    public double getDistancia() {
        return distancia;
    }

    public void setDistancia(double distancia) {
        this.distancia = distancia;
    }

    public long getIdImagem() {
        return idImagem;
    }

    public void setIdImagem(long idImagem) {
        this.idImagem = idImagem;
    }

    public double getVet(int posicao) {
        return vet[posicao];
    }

    public void setVet(double vet, int posicao) {
        this.vet[posicao] = vet;
    }

    public double[] getVet() {
        return vet;
    }
}

```

Apêndice C – Código na Linguagem JAVA– Conversão de DICOM para JPEG

```
//
class CxImageDCM : public CxImage
{
public:
    CxImageDCM();
    virtual ~CxImageDCM();

    bool LoadDCM(const TCHAR* filename);
    bool SaveAsDCM(const TCHAR* filename);
    bool SaveAsJPG(const TCHAR* fileName);

};//

Rodando uma imagem DICOM
//
bool CxImageDCM::LoadDCM(const TCHAR* filename)
{
    DcmFileFormat *dfile = new DcmFileFormat();
    OFCondition cond = dfile->loadFile(filename, EXS_Unknown,
                                      EGL_withoutGL, DCM_MaxReadLength, OFFalse);

    if (cond.bad()) {
        AfxMessageBox(cond.text());
    }

    E_TransferSyntax xfer =
        dfile->getDataset()->getOriginalXfer();
    DicomImage *di = new DicomImage(dfile, xfer,
                                    CIF_AcrNemaCompatibility, 0, 1);

    if (di->getStatus() != EIS_Normal)
        AfxMessageBox(DicomImage::getString(di->getStatus()));

    di->writeBMP("c:\\from_dicom.bmp", 24);

    return CxImage::Load("c:\\from_dicom.bmp", CXIMAGE_FORMAT_BMP);

}//

Convertendo de DICOM para JPEG
//
bool CxImageDCM::SaveAsJPG(const TCHAR* fileName)
{
    return CxImage::Save(fileName, CXIMAGE_FORMAT_JPG);

}//
```

Apêndice D – Planilhas utilizadas para a criação das estatísticas

Este Apêndice tem como objetivo mostrar o tempo de *Download* e *Upload* das imagens com o tipo de arquivo BMP, TIFF e JPEG. Esta tabela foi utilizada como base para a criação das estatísticas e gráficos apresentados neste trabalho.

Imagem BMP tem o tamanho de 769 KB, no formato TIFF tem o tamanho de 192 KB e no formato JPEG tem 18 KB.

Usuários				Velocidade (UDP) Kbit/s		Velocidade (TCP) Kbit/s		JITTER Kbit/s		Latência (Ping) ms	Velocidade Conexão ADSL	Upload e Download das imagens em segundos					
Numero	IP	DATA	HORA	Upload	Download	Upload	Download	Upload	Download			Imagem BMP	Download	Imagem TIFF	Download	Imagem JPG	Download
1	187.38.115.184	17/10/2011	02:00	754	961	758	9840,64	1	1	13	10 Mbit/s	8,401055409	0,647112383	2,026385224	0,156087409	0,189973615	0,014633195
2	187.38.115.184	17/10/2011	02:03	774	966	753	9871,36	1	1	12	10 Mbit/s	8,456839309	0,645098548	2,039840637	0,155601660	0,191235060	0,014587656
3	187.38.115.184	17/10/2011	02:05	754	958	754	9830,4	1	1	12	10 Mbit/s	8,445623342	0,647786458	2,037135279	0,156250000	0,190981432	0,014648438
4	177.12.208.100	21/10/2011	00:14	367	811	357	817	3	3	23	1 Mbit/s	21,88316151	7,123042506	5,278350515	1,718120805	0,494845361	0,161073826
5	177.9.232.191	19/10/2011	02:00	290	882	287	901	1	7	46	1 Mbit/s	22,18815331	7,067702553	5,351916376	1,704772475	0,501742160	0,159822420
6	177.9.235.50	19/10/2011	02:00	290	863	288	872	1	7	0,89	1 Mbit/s	22,11111111	7,302752294	5,333333333	1,761467890	0,500000000	0,165137615
7	177.9.238.134	19/10/2011	02:00	252	812	252	850	1	7	46	1 Mbit/s	25,26984127	7,491764706	6,095238095	1,807058824	0,571428571	0,169411765
8	201.95.51.8	19/10/2011	02:00	251	898	243	924	1	7	46	1 Mbit/s	26,20576132	6,891774892	6,320987654	1,662337662	0,592592593	0,155844156
9	201.95.51.95	19/10/2011	02:00	291	865	288	890	2	7	43	1 Mbit/s	22,11111111	7,155056180	5,333333333	1,725842697	0,500000000	0,161797753
10	187.34.166.2	19/10/2011	21:04	290	857	279	854	0,9	0,89	36	1 Mbit/s	22,82437276	7,456674473	5,505376344	1,798594848	0,516129032	0,168618267
11	189.62.122.72	20/10/2011	13:44	581	1960	559	1900	1	0,75	11	2 Mbit/s	11,39177102	3,351578947	2,747763864	0,808421053	0,257602862	0,075789474
12	189.62.122.72	20/10/2011	15:47	563	1770	562	1810	1	0,92	11	2 Mbit/s	11,33096085	3,518232044	2,733096085	0,848618785	0,256227758	0,079558011
13	201.92.208.122	20/10/2011	17:22	291	988	287	977	0,85	0,99	37	1 Mbit/s	22,18815331	6,517911975	5,351916376	1,572159672	0,501742160	0,147389969
14	187.101.99.139	20/10/2011	17:40	310	975	275	964	1	1	37	1 Mbit/s	23,15636364	6,605809129	5,585454545	1,593360996	0,523636364	0,149377593
15	187.101.93.32	20/10/2011	17:45	310	861	283	905	1	1	36	1 Mbit/s	22,50176678	7,036464088	5,427561837	1,697237569	0,508833922	0,159116022
16	201.92.211.108	20/10/2011	17:52	310	922	286	975	1	1	36	1 Mbit/s	22,26573427	6,531282051	5,370629371	1,575384615	0,503496503	0,147692308
17	201.13.131.180	20/10/2011	17:55	291	917	289	944	1	1	36	1 Mbit/s	22,03460208	6,745762712	5,314878893	1,627118644	0,498269896	0,152542373
18	201.13.129.132	20/10/2011	17:58	291	984	279	950	1	1	37	1 Mbit/s	22,82437276	6,703157895	5,505376344	1,616842105	0,516129032	0,151578947
19	189.19.151.284	20/10/2011	18:05	291	986	289	942	1	3	37	1 Mbit/s	22,03460208	6,760084926	5,314878893	1,630573248	0,498269896	0,152866242
20	187.101.101.148	20/10/2011	18:10	310	883	286	890	1	1	39	1 Mbit/s	22,26573427	7,155056180	5,370629371	1,725842697	0,503496503	0,161797753

Usuários				Velocidade (UDP) Kbit/s		Velocidade (TCP) Kbit/s		JITTER Kbit/s		Latência (Ping) ms	Velocidade Conexão ADSL	Upload e Download das imagens em segundos					
Numero	IP	DATA	HORA	Upload	Download	Upload	Download	Upload	Download			Imagem BMP	Download	Imagem TIFF	Download	Imagem JPG	Download
21	189.110.43.109	20/10/2011	18:14	310	973	289	981	1	1	37	1 Mbit/s	22,03460208	6,491335372	5,314878893	1,565749235	0,498269896	0,146788991
22	189.19.147.150	20/10/2011	18:16	310	913	284	973	1	1	37	1 Mbit/s	22,42253521	6,544707091	5,408450704	1,578622816	0,507042254	0,147995889
23	189.110.44.147	20/10/2011	18:19	310	871	279	957	1	12	36	1 Mbit/s	22,82437276	6,654127482	5,505376344	1,605015674	0,516129032	0,150470219
24	200.232.131.124	20/10/2011	18:23	310	898	284	915	1	1	39	1 Mbit/s	22,42253521	6,959562842	5,408450704	1,678688525	0,507042254	0,157377049
25	189.110.45.145	20/10/2011	18:27	310	901	281	954	1	1	36	1 Mbit/s	22,66192171	6,675052411	5,466192171	1,610062893	0,512455516	0,150943396
26	187.74.101.152	20/10/2011	18:30	310	900	273	905	1	1	36	1 Mbit/s	23,32600733	7,036464088	5,626373626	1,697237569	0,527472527	0,159116022
27	201.13.126.25	20/10/2011	18:33	290	886	278	903	1	1	37	1 Mbit/s	22,90647482	7,052048726	5,525179856	1,700996678	0,517985612	0,159468439
28	201.13.130.179	20/10/2011	18:33	310	976	284	986	1	1	33	1 Mbit/s	22,42253521	6,458417850	5,408450704	1,557809331	0,507042254	0,146044625
29	187.101.99.40	20/10/2011	18:36	310	958	284	978	1	1	36	1 Mbit/s	22,42253521	6,511247444	5,408450704	1,570552147	0,507042254	0,147239264
30	201.13.129.126	20/10/2011	18:42	291	984	292	917	1	1	37	1 Mbit/s	21,80821918	6,944383860	5,260273973	1,675027263	0,493150685	0,157033806
31	187.34.164.70	20/10/2011	18:45	310	900	285	964	1	1	37	1 Mbit/s	22,34385965	6,605809129	5,389473684	1,593360996	0,505263158	0,149377593
32	201.13.131.156	20/10/2011	18:56	310	991	295	986	1	1	37	1 Mbit/s	21,58644068	6,458417850	5,206779661	1,557809331	0,488135593	0,146044625
33	201.68.222.44	20/10/2011	22:16	0	965	277	1070	4	1	33	1 Mbit/s	22,98916968	5,951401869	5,545126354	1,435514019	0,519855596	0,134579439
34	201.13.190.160	20/10/2011	22:33	77	374	236	471	3	1	33	1 Mbit/s	26,98305085	13,52016985	6,508474576	3,261146497	0,610169492	0,305732484
35	201.13.190.170	20/10/2011	22:46	116	727	235	793	8	7	39	1 Mbit/s	27,09787234	8,030264817	6,536170213	1,936948298	0,612765957	0,181588903
36	177.9.232.122	21/10/2011	02:00	290	882	282	919	1	7	46	1 Mbit/s	22,58156028	6,929270947	5,446808511	1,671381937	0,510638298	0,156692057
37	177.9.232.122	21/10/2011	02:05	290	865	288	896	1	7	46	1 Mbit/s	22,11111111	7,107142857	5,333333333	1,714285714	0,500000000	0,160714286
38	177.9.232.162	21/10/2011	02:00	290	881	286	902	1	7	46	1 Mbit/s	22,26573427	7,059866962	5,370629371	1,702882483	0,503496503	0,159645233
39	177.9.232.162	21/10/2011	02:02	291	915	287	934	1	7	46	1 Mbit/s	22,18815331	6,817987152	5,351916376	1,644539615	0,501742160	0,154175589
40	177.9.232.170	21/10/2011	02:00	290	830	287	881	1	7	46	1 Mbit/s	22,18815331	7,228149830	5,351916376	1,743473326	0,501742160	0,163450624
41	177.9.232.170	21/10/2011	02:06	291	865	292	916	1	7	46	1 Mbit/s	21,80821918	6,951965066	5,260273973	1,676855895	0,493150685	0,15720524
42	177.9.232.78	21/10/2011	02:00	290	863	290	911	1	7	46	1 Mbit/s	21,95862069	6,990120746	5,296551724	1,686059276	0,496551724	0,158068057
43	177.9.232.78	21/10/2011	02:03	290	915	292	927	2	7	46	1 Mbit/s	21,80821918	6,869471413	5,260273973	1,656957929	0,493150685	0,155339806
44	177.9.236.128	21/10/2011	02:00	290	882	284	915	1	7	46	1 Mbit/s	22,42253521	6,959562842	5,408450704	1,678688525	0,507042254	0,157377049
45	177.9.236.128	21/10/2011	02:03	290	900	284	920	2	7	46	1 Mbit/s	22,42253521	6,921739130	5,408450704	1,669565217	0,507042254	0,156521739
46	177.9.236.9	21/10/2011	02:00	290	881	291	906	1	7	46	1 Mbit/s	21,88316151	7,028697572	5,278350515	1,695364238	0,494845361	0,158940397
47	177.9.236.9	21/10/2011	02:03	290	865	286	896	1	7	46	1 Mbit/s	22,26573427	7,107142857	5,370629371	1,714285714	0,503496503	0,160714286

Usuários				Velocidade (UDP) Kbit/s		Velocidade (TCP) Kbit/s		JITTER Kbit/s		Latência (Ping) ms	Velocidade Conexão ADSL	Upload e Download das imagens em segundos					
Numero	IP	DATA	HORA	Upload	Download	Upload	Download	Upload	Download			Imagem BMP	Download	Imagem TIFF	Download	Imagem JPG	Download
48	177.9.237.108	21/10/2011	02:00	290	881	284	905	1	7	46	1 Mbit/s	22,42253521	7,036464088	5,408450704	1,697237569	0,507042254	0,159116022
49	177.9.237.108	21/10/2011	02:02	290	848	292	896	1	7	46	1 Mbit/s	21,80821918	7,107142857	5,260273973	1,714285714	0,493150685	0,160714286
50	177.9.239.15	21/10/2011	02:00	291	882	295	902	1	7	46	1 Mbit/s	21,58644068	7,059866962	5,206779661	1,702882483	0,488135593	0,159645233
51	177.9.239.215	21/10/2011	02:00	290	825	290	857	1	7	46	1 Mbit/s	21,95862069	7,430571762	5,296551724	1,792298716	0,496551724	0,168028005
52	189.46.181.3	21/10/2011	02:00	38	37	52	45	2	23	47	128 Kbit/s	122,4615385	141,5111111	29,53846154	34,13333333	2,769230769	3,200000000
53	189.78.77.131	21/10/2011	02:00	291	900	288	937	1	7	46	1 Mbit/s	22,11111111	6,796157951	5,333333333	1,63927428	0,500000000	0,153681964
54	189.78.77.131	21/10/2011	02:03	290	900	291	930	1	7	46	1 Mbit/s	21,88316151	6,847311828	5,278350515	1,651612903	0,494845361	0,154838701
55	189.78.77.158	21/10/2011	02:00	290	917	284	928	1	7	46	1 Mbit/s	22,42253521	6,862068966	5,408450704	1,655172414	0,507042254	0,155172414
56	189.78.77.158	21/10/2011	02:02	271	882	283	872	1	8	46	1 Mbit/s	22,50176678	7,302752294	5,427561837	1,76146789	0,508833922	0,165137615
57	189.78.77.203	21/10/2011	02:00	290	881	291	923	1	7	46	1 Mbit/s	21,88316151	6,899241603	5,278350515	1,664138678	0,494845361	0,156013001
58	189.78.77.203	21/10/2011	02:02	290	746	292	791	4	14	67	1 Mbit/s	21,80821918	8,0505689	5,260273973	1,941845765	0,493150685	0,182048040
59	201.95.51.133	21/10/2011	02:00	290	773	286	865	1	7	44	1 Mbit/s	22,26573427	7,361849711	5,370629371	1,775722543	0,503496503	0,166473988
60	201.95.51.212	21/10/2011	02:00	290	915	287	925	1	7	46	1 Mbit/s	22,18815331	6,884324324	5,351916376	1,660540541	0,501742160	0,155675676
61	201.95.51.212	21/10/2011	02:03	308	898	292	922	1	7	46	1 Mbit/s	21,80821918	6,906724512	5,260273973	1,665943601	0,493150685	0,156182213
62	201.95.51.212	21/10/2011	02:05	290	830	286	883	1	7	46	1 Mbit/s	22,26573427	7,211778029	5,370629371	1,739524349	0,503496503	0,163080408
63	201.95.52.166	21/10/2011	02:00	290	863	292	897	1	7	46	1 Mbit/s	21,80821918	7,099219621	5,260273973	1,712374582	0,493150685	0,160535117
64	201.95.52.166	21/10/2011	02:06	309	842	287	867	2	1	34	1 Mbit/s	22,18815331	7,344867359	5,351916376	1,771626298	0,501742160	0,166089965
65	201.95.52.60	21/10/2011	02:00	290	881	286	914	2	7	46	1 Mbit/s	22,26573427	6,967177243	5,370629371	1,680525164	0,503496503	0,157549234
66	201.95.52.60	21/10/2011	02:03	291	865	290	939	1	7	46	1 Mbit/s	21,95862069	6,781682641	5,296551724	1,635782748	0,496551724	0,153354633
67	201.95.52.80	21/10/2011	02:00	290	934	292	954	1	7	46	1 Mbit/s	21,80821918	6,675052411	5,260273973	1,610062893	0,493150685	0,150943396
68	201.95.52.80	21/10/2011	02:06	309	933	296	951	1	7	46	1 Mbit/s	21,51351351	6,696109359	5,189189189	1,615141956	0,486486486	0,151419558
69	201.95.52.80	21/10/2011	02:08	309	900	294	925	1	7	46	1 Mbit/s	21,65986395	6,884324324	5,224489796	1,660540541	0,489795918	0,155675676
70	201.95.53.10	21/10/2011	02:00	290	848	283	877	1	7	46	1 Mbit/s	22,50176678	7,261117446	5,427561837	1,751425314	0,508833922	0,164196123
71	201.95.53.10	21/10/2011	02:02	291	881	289	916	1	7	46	1 Mbit/s	22,03460208	6,951965066	5,314878893	1,676855895	0,498269896	0,157205240
72	201.95.53.179	21/10/2011	02:00	291	865	292	871	1	7	46	1 Mbit/s	21,80821918	7,311136625	5,260273973	1,763490241	0,493150685	0,165327210
73	201.95.53.179	21/10/2011	02:03	291	915	294	924	2	7	46	1 Mbit/s	21,65986395	6,891774892	5,224489796	1,662337662	0,489795918	0,155844156
74	201.95.53.179	21/10/2011	02:05	291	846	287	874	1	7	46	1 Mbit/s	22,18815331	7,28604119	5,351916376	1,757437071	0,50174216	0,164759725

Usuários				Velocidade (UDP) Kbit/s		Velocidade (TCP) Kbit/s		JITTER Kbit/s		Latência (Ping) ms	Velocidade Conexão ADSL	Upload e Download das imagens em segundos					
Numero	IP	DATA	HORA	Upload	Download	Upload	Download	Upload	Download			Imagem BMP	Download	Imagem TIFF	Download	Imagem JPG	Download
75	201.95.53.209	21/10/2011	02:00	290	829	289	858	1	7	46	1 Mbit/s	22,03460208	7,421911422	5,314878893	1,79020979	0,498269896	0,167832168
76	201.95.53.209	21/10/2011	02:04	290	846	285	876	1	7	46	1 Mbit/s	22,34385965	7,269406393	5,389473684	1,753424658	0,505263158	0,164383562
77	201.95.53.218	21/10/2011	02:00	309	933	297	962	1	7	46	1 Mbit/s	21,44107744	6,61954262	5,171717172	1,596673597	0,484848485	0,149688150
78	201.95.53.218	21/10/2011	02:10	309	898	298	934	1	7	46	1 Mbit/s	21,36912752	6,817987152	5,154362416	1,644539615	0,483221477	0,154175589
79	201.95.53.40	21/10/2011	02:00	290	882	292	928	1	7	46	1 Mbit/s	21,80821918	6,862068966	5,260273973	1,655172414	0,493150685	0,155172414
80	201.95.53.40	21/10/2011	00:00	290	863	290	898	1	7	46	1 Mbit/s	21,95862069	7,091314031	5,296551724	1,710467706	0,496551724	0,160356347
81	201.95.53.40	21/10/2011	02:07	290	882	287	906	1	7	46	1 Mbit/s	22,18815331	7,028697572	5,351916376	1,695364238	0,50174216	0,158940397
82	201.95.53.40	21/10/2011	02:09	290	830	287	862	2	7	46	1 Mbit/s	22,18815331	7,387470998	5,351916376	1,781902552	0,501742160	0,167053364
83	201.95.53.79	21/10/2011	02:00	290	86	282	860	1	7	46	1 Mbit/s	22,58156028	7,404651163	5,446808511	1,786046512	0,510638298	0,167441860
84	201.95.53.79	21/10/2011	02:05	290	101	289	845	1	7	46	1 Mbit/s	22,03460208	7,536094675	5,314878893	1,817751479	0,498269896	0,170414201
85	201.95.83.40	21/10/2011	02:00	290	830	290	871	1	7	46	1 Mbit/s	21,95862069	7,311136625	5,296551724	1,763490241	0,496551724	0,165327210
86	201.27.108.27	21/10/2011	05:31	38	829	266	981	14	1	33	1 Mbit/s	23,93984962	6,491335372	5,774436090	1,565749235	0,541353383	0,146788991
87	201.27.107.92	21/10/2011	05:40	116	753	293	961	23	6	35	1 Mbit/s	21,73378840	6,626430801	5,242320819	1,598335068	0,491467577	0,149843913
88	201.52.117.94	21/10/2011	09:22	0	557	244	563	0,84	3	12	512 Kbit/s	26,09836066	11,31083481	6,295081967	2,728241563	0,590163934	0,255772647
89	201.52.117.94	21/10/2011	10:13	0	565	243	556	0,88	5	11	512 Kbit/s	26,20576132	11,45323741	6,320987654	2,762589928	0,592592593	0,258992806
90	201.52.117.94	21/10/2011	10:34	0	552	243	549	1	5	12	512 Kbit/s	26,20576132	11,59927140	6,320987654	2,797814208	0,592592593	0,262295082
91	201.52.117.94	21/10/2011	10:56	0	537	244	564	0,75	3	15	512 Kbit/s	26,09836066	11,29078014	6,295081967	2,723404255	0,590163934	0,255319149
92	201.52.117.94	21/10/2011	11:23	0	563	220	563	0,88	2	10	512 Kbit/s	28,94545455	11,31083481	6,981818182	2,728241563	0,654545455	0,255772647
93	201.52.117.94	21/10/2011	11:45	0	558	243	567	0,77	1	16	512 Kbit/s	26,20576132	11,23104056	6,320987654	2,708994709	0,592592593	0,253968254
94	189.62.122.72	21/10/2011	11:46	581	91	558	871	1	0,85	11	2 Mbit/s	11,41218638	7,311136625	2,752688172	1,763490241	0,258064516	0,165327210
95	201.52.117.94	21/10/2011	12:17	0	549	243	593	0,73	1	11	512 Kbit/s	26,20576132	10,73861720	6,320987654	2,590219224	0,592592593	0,242833052
96	189.62.122.72	21/10/2011	13:19	563	1930	559	1840	0,93	0,77	12	2 Mbit/s	11,39177102	3,460869565	2,747763864	0,834782609	0,257602862	0,078260870
97	189.62.122.72	21/10/2011	14:57	523	1930	496	1900	0,95	2	15	2 Mbit/s	12,83870968	3,351578947	3,096774194	0,808421053	0,290322581	0,075789474
98	201.27.101.222	21/10/2011	17:04	233	830	305	868	27	23	33	1 Mbit/s	20,87868852	7,336405530	5,036065574	1,769585253	0,472131148	0,165898618
99	201.27.107.44	21/10/2011	17:23	232	906	272	880	25	9	64	1 Mbit/s	23,41176471	7,236363636	5,647058824	1,745454545	0,529411765	0,163636364
100	201.93.221.215	21/10/2011	17:37	0	0	266	935	15	5	34	1 Mbit/s	23,93984962	6,810695187	5,774436090	1,642780749	0,541353383	0,154010695
101	201.68.223.246	21/10/2011	18:03	116	829	275	922	5	2	34	1 Mbit/s	23,15636364	6,906724512	5,585454545	1,665943601	0,523636364	0,156182213

Usuários				Velocidade (UDP) Kbit/s		Velocidade (TCP) Kbit/s		JITTER Kbit/s		Latência (Ping) ms	Velocidade Conexão ADSL	Upload e Download das imagens em segundos					
Numero	IP	DATA	HORA	Upload	Download	Upload	Download	Upload	Download			Imagem BMP	Download	Imagem TIFF	Download	Imagem JPG	Download
102	189.46.3.38	21/10/2011	18:15	19	803	278	890	12	12	33	1 Mbit/s	22,90647482	7,15505618	5,525179856	1,725842697	0,517985612	0,161797753
103	189.68.40.14	22/10/2011	02:00	77	103	66	120	2	0,85	47	128 Kbit/s	96,48484848	53,06666667	23,27272727	12,80000000	2,181818182	1,200000000
104	201.95.52.166	22/10/2011	02:00	290	848	286	875	2	7	46	1 Mbit/s	22,26573427	7,277714286	5,370629371	1,755428571	0,503496503	0,164571429
105	201.95.53.243	22/10/2011	02:00	290	915	295	934	1	7	46	1 Mbit/s	21,58644068	6,817987152	5,206779661	1,644539615	0,488135593	0,154175589

Apêndice E – Planilhas utilizadas para a criação das estatísticas

Este Apêndice mostra a tabela de *download* de imagens utilizando os sistemas de banco de dados.

Usuário	Imagem 1				Imagem 2				Imagem 3				Imagem 4			
	Upload Imagem 1 (KB)	Tempo Download com MatLab (ms)	Tempo Download com MySQL (ms)	Tempo Download com Slim-Tree (ms)	Upload Imagem 2 (KB)	Tempo Download com MatLab (ms)	Tempo Download com MySQL (ms)	Tempo Download com Slim-Tree (ms)	Upload Imagem 3 (KB)	Tempo Download com MatLab (ms)	Tempo Download com MySQL (ms)	Tempo Download com Slim-Tree (ms)	Upload Imagem 4 (KB)	Tempo Download com MatLab (ms)	Tempo Download com MySQL (ms)	Tempo Download com Slim-Tree (ms)
1	76	1931,102	1722,002	1266,901	49	1538,60	1440,60	1019,20	35	1099,00	1078,00	728,00	79	2.528,00	2.164,60	1.595,80
2	64	1604,999	1438,246	1094,317	53	1664,20	1632,40	1081,20	64	1945,60	1676,80	1344,00	19	638,40	577,60	383,80
3	24	606,7301	512,7848	395,3532	41	1230,00	1238,20	836,40	42	1293,60	1201,20	856,80	21	651,00	625,80	432,60
4	28	7731,007	7178,792	5070,336	71	2243,60	2201,00	1448,40	51	1713,60	1479,00	1030,20	35	1.085,00	1.008,00	728,00
5	66	18339,09	15752,81	12108,5	81	2705,40	2235,60	1636,20	48	1440,00	1344,00	988,80	59	1.923,40	1.805,40	1.227,20
6	62	17133,03	17018,81	11650,46	73	2321,40	1985,60	1533,00	31	936,20	874,20	644,80	62	1.872,40	1.748,40	1.264,80
7	56	17388,99	15712,94	10684,8	19	592,80	551,00	387,60	34	1094,80	1006,40	707,20	85	2.703,00	2.516,00	1.768,00
8	56	16226,32	14466,84	9872,641	34	1149,20	945,20	714,00	14	470,40	378,00	288,40	28	929,60	789,60	582,40
9	74	21920,72	18980,13	13499,96	59	1982,40	1805,40	1215,40	51	1683,00	1448,40	1030,20	15	498,00	441,00	309,00
10	50	14310,82	13939,11	9385,667	46	1554,80	1196,00	947,60	16	483,20	419,20	323,20	41	1.246,40	1.246,40	836,40
11	80	10825,98	9548,379	6791,453	23	708,40	657,80	483,00	71	2286,20	1846,00	1491,00	41	1.394,00	1.148,00	844,60
12	59	8285,569	7920,796	5367,381	36	1159,20	950,40	756,00	70	2366,00	2170,00	1470,00	51	1.642,20	1.560,60	1.040,40
13	21	5790,461	5267,226	3662,641	43	1393,20	1143,80	903,00	45	1359,00	1323,00	927,00	33	1.042,80	871,20	673,20
14	40	10460,41	9521,66	6906,556	18	586,80	475,20	374,40	14	456,40	378,00	282,80	50	1.630,00	1.530,00	1.020,00
15	30	8999,602	7124,685	5517,613	68	2230,40	2026,40	1428,00	41	1246,40	1180,80	828,20	21	709,80	571,20	432,60

Usuário	Imagem 1				Imagem 2				Imagem 3				Imagem 4			
	Upload Imagem 1 (KB)	Tempo Download com MailLab (ms)	Tempo Download com MySQL (ms)	Tempo Download com Slim-Tree (ms)	Upload Imagem 2 (KB)	Tempo Download com MailLab (ms)	Tempo Download com MySQL (ms)	Tempo Download com Slim-Tree (ms)	Upload Imagem 3 (KB)	Tempo Download com MailLab (ms)	Tempo Download com MySQL (ms)	Tempo Download com Slim-Tree (ms)	Upload Imagem 4 (KB)	Tempo Download com MailLab (ms)	Tempo Download com MySQL (ms)	Tempo Download com Slim-Tree (ms)
16	70	18953,85	15967,18	11716,92	20	676,00	556,00	420,00	64	1920,00	1984,00	1305,60	32	966,40	979,20	665,60
17	63	17612,88	16445,76	10716,27	36	1166,40	1058,40	727,20	38	1162,80	1147,60	790,40	69	2.152,80	2.042,40	1.449,00
18	80	21450,11	18232,59	13674,44	58	1844,40	1566,00	1183,20	73	2438,20	2000,20	1489,20	62	1.971,60	1.909,60	1.264,80
19	58	16000,51	15210,36	9975,626	69	2277,00	2042,40	1435,20	78	2371,20	2324,40	1638,00	66	2.046,00	2.032,80	1.372,80
20	30	8048,809	8208,719	5596,854	53	1759,60	1473,40	1102,40	14	448,00	378,00	285,60	59	1.899,80	1.616,60	1.239,00
21	41	10260,39	8927,87	6862,467	21	676,20	634,20	432,60	64	2137,60	1856,00	1305,60	60	1.980,00	1.644,00	1.236,00
22	74	20481,48	15876,18	12361,6	66	2006,40	1768,80	1386,00	53	1664,20	1579,40	1113,00	82	2.542,00	2.328,80	1.656,40
23	65	17727,31	15878,45	10984,41	74	2427,20	2190,40	1509,60	48	1478,40	1459,20	979,20	36	1.116,00	972,00	741,60
24	69	19883,72	15691,37	12217,7	81	2478,60	2268,00	1668,60	73	2365,20	1898,00	1489,20	78	2.542,80	2.199,60	1.591,20
25	25	6420,964	6338,113	4183,983	23	726,80	667,00	473,80	69	2263,20	2056,20	1449,00	23	694,60	657,80	469,20
26	48	14189,79	13078,54	8804,508	55	1804,00	1485,00	1133,00	77	2387,00	2125,20	1555,40	41	1.361,20	1.189,00	852,80
27	26	6980,731	6478,848	4699,446	79	2528,00	2433,20	1659,00	14	445,20	364,00	294,00	54	1.792,80	1.620,00	1.134,00
28	34	9433,834	7535,903	5861,258	62	2083,20	1698,80	1302,00	73	2321,40	2233,80	1474,60	36	1.173,60	964,80	741,60
29	79	20910,92	17942,09	13295,21	29	968,60	829,40	585,80	54	1728,00	1566,00	1101,60	25	835,00	650,00	505,00
30	29	7680,087	6968,026	5137,012	46	1407,60	1380,00	947,60	74	2427,20	2279,20	1494,80	43	1.453,40	1.229,80	868,60
31	20	5310,373	4402,905	3461,826	50	1580,00	1490,00	1010,00	44	1408,00	1232,00	915,20	65	2.158,00	1.755,00	1.326,00
32	16	4217,606	3523,57	2749,452	51	1672,80	1377,00	1071,00	68	2257,60	2080,80	1387,20	35	1.099,00	1.015,00	728,00
33	21	4948,411	4150,28	3320,224	73	2336,00	2160,80	1489,20	35	1078,00	987,00	721,00	59	1.817,20	1.640,20	1.203,60
34	15	8056,561	6820,892	5189,809	74	2382,80	1968,40	1494,80	80	2656,00	2144,00	1616,00	60	1.992,00	1.584,00	1.224,00
35	31	9338,008	8410,391	6431,475	75	2265,00	2220,00	1545,00	29	974,40	829,40	609,00	69	2.290,80	1.904,40	1.449,00
36	57	16048,14	12996,04	9943,939	54	1717,20	1490,40	1090,80	67	2144,00	2023,40	1366,80	64	2.150,40	1.920,00	1.292,80
37	44	13290	12261,61	7989,821	77	2402,40	2340,80	1586,20	28	884,80	823,20	582,40	45	1.440,00	1.359,00	909,00
38	56	15396,9	14800,89	10430,16	55	1771,00	1617,00	1133,00	74	2353,20	2234,80	1539,20	49	1.597,40	1.332,80	999,60
39	75	20033,83	19904,58	13054,3	34	1074,40	952,00	707,20	65	2015,00	1846,00	1365,00	15	510,00	432,00	306,00
40	19	5768,445	4824,518	3530,988	58	1879,20	1728,40	1171,60	37	1110,00	1021,20	747,40	28	845,60	728,00	582,40
41	16	4526,987	4304,803	2916,157	68	2189,60	2108,00	1428,00	34	1135,60	945,20	714,00	57	1.710,00	1.493,40	1.174,20

Usuário	Imagem 1				Imagem 2				Imagem 3				Imagem 4			
	Upload Imagem 1 (KB)	Tempo Download com MatLab (ms)	Tempo Download com MySQL (ms)	Tempo Download com Slim-Tree (ms)	Upload Imagem 2 (KB)	Tempo Download com MatLab (ms)	Tempo Download com MySQL (ms)	Tempo Download com Slim-Tree (ms)	Upload Imagem 3 (KB)	Tempo Download com MatLab (ms)	Tempo Download com MySQL (ms)	Tempo Download com Slim-Tree (ms)	Upload Imagem 4 (KB)	Tempo Download com MatLab (ms)	Tempo Download com MySQL (ms)	Tempo Download com Slim-Tree (ms)
42	49	14575,98	12406,94	9023,227	51	1683,00	1581,00	1040,40	54	1771,20	1458,00	1123,20	74	2.501,20	2.086,80	1.524,40
43	33	9154,693	7667,055	5836,117	22	730,40	633,60	453,20	55	1738,00	1617,00	1144,00	14	428,40	411,60	294,00
44	45	13421,64	12158,43	8052,984	57	1801,20	1675,80	1151,40	71	2186,80	1860,20	1476,80	66	2.019,60	2.046,00	1.386,00
45	46	12640	11040	8320	56	1848,00	1489,60	1131,20	21	646,80	634,20	428,40	39	1.255,80	1.029,60	819,00
46	65	18349,93	16503,4	11771,66	51	1550,40	1428,00	1050,60	84	2587,20	2452,80	1696,80	27	826,20	729,00	550,80
47	83	24707,05	22487,86	14942,59	15	483,00	417,00	306,00	50	1550,00	1450,00	1030,00	30	930,00	906,00	606,00
48	24	6916,243	6267,845	4365,878	23	722,20	685,40	464,60	81	2737,80	2365,20	1684,80	73	2.263,00	2.160,80	1.489,20
49	54	16125,71	13600	10005,71	75	2310,00	1980,00	1545,00	49	1499,40	1519,00	1019,20	39	1.240,20	1.193,40	803,40
50	44	12465,9	10685,06	8052,506	66	2204,40	2046,00	1346,40	85	2686,00	2567,00	1734,00	49	1.577,80	1.411,20	1.009,40
51	24	7137,83	6344,737	4582,31	32	1036,80	934,40	652,80	66	2059,20	1887,60	1333,20	56	1.848,00	1.724,80	1.131,20
52	14	22513	20533	16200	14	448,00	369,60	291,20	14	420,00	414,40	282,80	15	495,00	402,00	303,00
53	85	24065,57	20441,24	15077,23	37	1235,80	1028,60	777,00	51	1601,40	1499,40	1040,40	18	608,40	536,40	378,00
54	77	21154,41	17849,03	13618,15	24	816,00	624,00	489,60	64	2137,60	1881,60	1292,80	57	1.732,80	1.618,80	1.151,40
55	67	18602,07	15616,55	11597,59	73	2204,60	2190,00	1489,20	25	790,00	775,00	525,00	59	1.817,20	1.545,80	1.227,20
56	32	9733,486	7727,798	5958,073	83	2490,00	2324,00	1709,80	68	2284,80	1849,60	1428,00	29	875,80	754,00	609,00
57	47	13347,43	11638,31	8220,065	72	2275,20	2073,60	1468,80	51	1591,20	1377,00	1030,20	24	782,40	628,80	489,60
58	35	10942,1	9883,186	7200,607	33	1049,40	917,40	673,20	26	821,60	691,60	535,60	26	821,60	780,00	540,80
59	75	23044,25	20961,94	14020,9	75	2265,00	2175,00	1530,00	61	2013,00	1659,20	1256,60	30	1.002,00	846,00	606,00
60	16	4192,086	4247,611	2915,027	51	1734,00	1530,00	1071,00	44	1390,40	1311,20	906,40	66	2.204,40	1.782,00	1.346,40
61	35	9712,451	9284,859	6352,798	61	2074,00	1683,60	1281,00	18	601,20	486,00	367,20	66	2.072,40	1.993,20	1.372,80
62	49	14128,74	13860,48	9389,354	62	1860,00	1624,40	1264,80	30	960,00	834,00	606,00	70	2.156,00	1.932,00	1.470,00
63	56	16310,9	14609,77	10406,96	77	2356,20	2233,00	1601,60	14	464,80	392,00	291,20	76	2.401,60	2.219,20	1.596,00
64	18	5152,941	5018,224	3536,332	62	2095,60	1872,40	1302,00	55	1650,00	1518,00	1122,00	40	1.216,00	1.232,00	832,00
65	20	5753,698	4713,873	3604,726	31	1035,40	917,60	638,60	73	2423,60	1898,00	1518,40	38	1.223,60	1.064,00	782,80
66	44	12612,91	10672,46	7761,789	34	1074,40	1006,40	693,60	64	2163,20	1868,80	1344,00	38	1.261,60	1.124,80	775,20
67	84	22197,23	21066,16	14703,9	73	2365,20	2190,00	1489,20	48	1459,20	1372,80	998,40	49	1.626,80	1.460,20	1.009,40

Usuário	Imagem 1				Imagem 2				Imagem 3				Imagem 4			
	Upload Imagem 1 (KB)	Tempo Download com MatLab (ms)	Tempo Download com MySQL (ms)	Tempo Download com Slim-Tree (ms)	Upload Imagem 2 (KB)	Tempo Download com MatLab (ms)	Tempo Download com MySQL (ms)	Tempo Download com Slim-Tree (ms)	Upload Imagem 3 (KB)	Tempo Download com MatLab (ms)	Tempo Download com MySQL (ms)	Tempo Download com Slim-Tree (ms)	Upload Imagem 4 (KB)	Tempo Download com MatLab (ms)	Tempo Download com MySQL (ms)	Tempo Download com Slim-Tree (ms)
68	54	13810,22	11824,44	9116,551	81	2462,40	2332,80	1701,00	73	2204,60	1956,40	1503,80	61	2.061,80	1.708,00	1.281,00
69	26	6880,865	6116,324	4587,243	63	2079,00	1801,80	1310,40	58	1925,60	1542,80	1171,60	51	1.540,20	1.519,80	1.030,20
70	29	9026,408	7038,449	5480,319	50	1670,00	1540,00	1030,00	58	1798,00	1647,20	1183,20	21	705,60	609,00	424,20
71	80	21425,5	18086,46	14469,17	23	713,00	699,20	464,60	74	2338,40	2086,80	1554,00	68	2.244,00	2.026,40	1.428,00
72	45	12385,12	11810,98	8612,17	49	1519,00	1391,60	1029,00	40	1200,00	1160,00	808,00	71	2.158,40	1.917,00	1.462,60
73	16	4713,593	4007,965	2907,186	84	2671,20	2284,80	1764,00	18	597,60	507,60	374,40	53	1.706,60	1.473,40	1.081,20
74	57	16800	15339,13	10747,83	80	2720,00	2224,00	1680,00	80	2496,00	2192,00	1680,00	56	1.814,40	1.691,20	1.153,60
75	51	15380,51	12832,82	9718,974	66	1993,20	1900,80	1346,40	24	768,00	696,00	489,60	17	510,00	459,00	346,80
76	18	5617,9	5152,511	3423,927	73	2350,60	2263,00	1474,60	17	574,60	459,00	346,80	58	1.763,20	1.740,00	1.194,80
77	84	22021,46	19930,81	14216,38	28	884,80	728,00	576,80	68	2176,00	2080,80	1400,80	64	2.150,40	1.843,20	1.318,40
78	72	20431,86	19077,94	12431,43	53	1770,20	1399,20	1081,20	29	899,00	835,20	609,00	55	1.870,00	1.441,00	1.122,00
79	31	8485,69	7357,845	5424,397	48	1488,00	1334,40	988,80	63	2066,40	1789,20	1323,00	14	467,60	406,00	291,20
80	76	22658,44	20487,57	13839,29	53	1643,00	1547,60	1081,20	16	505,60	489,60	326,40	81	2.592,00	2.268,00	1.636,20
81	84	24458,81	19835,5	15212,19	70	2114,00	2128,00	1428,00	35	1092,00	938,00	721,00	27	907,20	750,60	567,00
82	75	21936,71	19021,07	14022,83	59	1852,60	1829,00	1203,60	27	815,40	712,80	561,60	30	966,00	852,00	612,00
83	26	7748,186	7367,907	4943,628	78	2496,00	2106,00	1606,80	26	795,60	676,00	546,00	32	998,40	915,20	652,80
84	69	20627,88	20236,21	13316,73	29	951,20	777,20	591,60	17	513,40	503,20	350,20	32	1.088,00	902,40	659,20
85	16	4599,403	4363,536	3066,269	35	1113,00	931,00	707,00	85	2720,00	2312,00	1717,00	14	459,20	420,00	288,40
86	36	8979,409	7988,175	6064,016	28	929,60	812,00	576,80	67	2197,60	1902,80	1407,00	55	1.804,00	1.606,00	1.122,00
87	75	19854,32	17107,18	12611,86	85	2652,00	2567,00	1751,00	79	2370,00	2180,40	1659,00	81	2.511,00	2.268,00	1.652,40
88	84	36869,63	32825,86	24738,33	73	2204,60	2219,20	1489,20	77	2340,80	2356,20	1601,60	39	1.326,00	1.029,60	803,40
89	59	26093,53	23063,31	17676,26	52	1768,00	1549,60	1060,80	70	2100,00	1862,00	1470,00	38	1.162,80	1.102,00	790,40
90	14	6653,843	5643,133	4421,858	72	2232,00	1958,40	1454,40	75	2550,00	2025,00	1560,00	31	998,20	948,60	626,20
91	63	30049,36	27724,11	18601,99	35	1106,00	917,00	728,00	36	1130,40	950,40	756,00	69	2.097,60	1.973,40	1.421,40
92	30	14360,5	12830,98	8582,309	27	885,60	750,60	561,60	69	2235,60	2042,40	1407,60	70	2.212,00	2.170,00	1.470,00
93	22	9552,593	8389,136	6307,16	52	1768,00	1591,20	1060,80	68	2080,80	2067,20	1428,00	84	2.654,40	2.251,20	1.713,60

Usuário	Imagem 1				Imagem 2				Imagem 3				Imagem 4			
	Upload Imagem 1 (KB)	Tempo Download com MatLab (ms)	Tempo Download com MySQL (ms)	Tempo Download com Slim-Tree (ms)	Upload Imagem 2 (KB)	Tempo Download com MatLab (ms)	Tempo Download com MySQL (ms)	Tempo Download com Slim-Tree (ms)	Upload Imagem 3 (KB)	Tempo Download com MatLab (ms)	Tempo Download com MySQL (ms)	Tempo Download com Slim-Tree (ms)	Upload Imagem 4 (KB)	Tempo Download com MatLab (ms)	Tempo Download com MySQL (ms)	Tempo Download com Slim-Tree (ms)
94	41	12563,4	11352,47	7719,679	23	713,00	616,40	464,60	65	2145,00	2002,00	1326,00	68	2.189,60	1.836,00	1.373,60
95	69	29042,83	26250,25	19175,72	65	1976,00	1924,00	1365,00	61	2061,80	1842,20	1244,40	57	1.858,20	1.755,60	1.162,80
96	61	8979,043	7172,609	5578,696	66	2046,00	1821,60	1346,40	32	1043,20	972,80	652,80	21	667,80	596,40	441,00
97	76	10744,25	9071,495	6498,021	14	420,00	420,00	282,80	40	1256,00	1144,00	832,00	16	508,80	428,80	329,60
98	63	19757,6	15573,64	12203,23	51	1683,00	1438,20	1050,60	34	1108,40	924,80	693,60	70	2.310,00	1.960,00	1.456,00
99	55	15628,36	13724,91	10519,09	24	772,80	624,00	504,00	51	1611,60	1356,60	1060,80	24	816,00	739,20	484,80
100	66	17170,4	16146,99	11598,55	40	1336,00	1192,00	840,00	71	2257,80	2101,60	1476,80	49	1.499,40	1.362,20	1.029,00
101	78	22255,62	17913,06	13706,2	58	1890,80	1763,20	1218,00	72	2260,80	2016,00	1468,80	78	2.449,20	2.199,60	1.575,60
102	38	10598,38	9518,292	6818,067	55	1771,00	1650,00	1133,00	14	450,80	414,40	291,20	50	1.590,00	1.380,00	1.030,00
103	14	5161	4599	34100	14	473,20	417,20	288,40	14	425,60	408,80	282,80	14	459,20	417,20	282,80
104	60	17575,41	14480,37	11164,25	71	2343,00	1988,00	1476,80	67	2264,60	1876,00	1366,80	14	422,80	400,40	291,20
105	38	9803,512	9610,021	6578,672	53	1696,00	1462,80	1113,00	78	2605,20	2215,20	1591,20	53	1.770,20	1.611,20	1.113,00

Anexo A – A forma de Desenvolvimento de Aplicativos para Internet

Os navegadores comunicam-se através de uma rede, com os servidores de internet utilizando um protocolo de comunicação chamado HTTP (*HyperText Transfer Protocol*). Este protocolo é utilizado para trocas de pacotes de informações entre computadores conectados à internet por meio de protocolos TCP/IP (*Transfer Control Protocol/Internet Protocol*). Em qualquer transação do tipo HTTP, o sistema abre uma conexão entre um navegador cliente e um servidor de internet, submetendo uma solicitação ao servidor de internet, retornando a resposta do servidor ao cliente e então encerra a conexão. Nenhuma informação sobre a solicitação é mantida após o encerramento da conexão.

Ao acessar um site, um mecanismo conhecido por DNS (*Domain Name Service*) muda o endereço do *site* em um endereço IP, que é um identificador de máquina para a internet. Ao se conectar ao endereço IP, o navegador então passa a localização e o nome do documento para o servidor como uma solicitação. O servidor responde enviando o documento solicitado ao navegador, que o interpreta e o exibe na tela.

Um modelo simples de desenvolvimento para a internet é em termos de relações cliente/servidor. A maior parte das interações na internet pode ser feita mediante solicitações e respostas.

Uma das arquiteturas mais simples para aplicações voltadas para a internet envolve a simples recuperação de documentos HTML da WWW. A linguagem HTML (*Hypertext Markup Language*) permite que os documentos enviados pelo servidor sejam exibidos em um navegador no cliente (YOURDON, 1996). A Figura 8.1 mostra o acesso a uma página de HTML.

Figura 8. 1 - Acesso a página de HTML

Este formato HTML é um método limitado no controle das informações da internet, pois exibe informações pré-determinadas e estáticas, não recuperando nada de banco de dados ou solicitações de entrada de dados ao usuário. Para exibir dados atualizados, utilizando este formato, seria necessária a edição frequente do arquivo HTML.

A arquitetura mais dinâmica e comum na internet envolve formulários que solicitam entrada de dados, exibem informações e fornecem uma combinação familiar de componentes de interface tais como campos de texto, botões de opções, barras de rolagem e listas de escolhas. Nesta arquitetura, após o preenchimento dos dados requisitados estes são submetidos ao servidor por um botão na página HTML. Este tipo de procedimento leva à execução de um comando CGI (Common Gateway Interface) que chama um programa que processa os dados fornecidos (YOURDON, 1996). A Figura 8.2 ilustra a comunicação através de CGI.

Figura 8. 2 - Como funcionam o acesso a programas CGI

Extraído de: YOURDON, 1996

O CGI é um padrão de comunicação com regras específicas para a criação de aplicações cooperativas. Ao contrário do formato HTML, um programa ou *script* CGI exibe informações dinâmicas obtidas de recursos disponíveis no servidor de internet.

Os programas CGI podem ser escritos em diversas linguagens de programação como, C, C++, Java, *Visual Basic* ou outros, e também podem trabalhar com diferentes Sistemas Operacionais. A diferença da Linguagem Java com outras linguagens de programação é que Java é nativo para redes de computadores, como a internet. Programas CGI escritos em Java são independentes de plataforma.

Os programas CGI são processados, geralmente, no servidor. Existem outros tipos de *scripts* que podem ser integrados a páginas HTML e executar operações dinamicamente sem sobrecarregar o servidor, tais como *JavaScript* e *VBScript*. Neste tipo de arquitetura a maior parte do processamento é feito no cliente, liberando o servidor da sobrecarga de processamento.

Anexo B – Particularidades da linguagem JAVA

A linguagem Java é uma programação orientada a objetos com sintaxe semelhante ao C e C++. As aplicações em Java são mais robustas que aplicações correspondentes em C ou C++, pois um sistema na linguagem Java ou JVM (*Java Virtual Machine*) gerencia toda a memória de um equipamento.

Com as mesmas características de robustez, a linguagem Java também fornece segurança, mesmo quando as aplicações trafegam pela internet, porque o JVM possui mecanismos de segurança que protegem contra interferências indevidas. Aplicações que executam tarefas concorrentes são mais rápidas, pois o Java possui suporte próprio para tratamento de concorrência (YOURDON, 1996).

A principal vantagem em Java é que as aplicações são completamente portáteis. Uma vez que o código é escrito não há a necessidade de adaptá-lo ou mesmo recompilá-lo. Ao invés de produzir instruções específicas para máquinas distintas, o compilador Java produz *bytecode* independente de plataforma. O ambiente *runtime* Java, ou máquina virtual, irá traduzir este *bytecode* em instruções específicas para a máquina em uso. A máquina virtual Java é instalada na máquina do usuário, ou como parte de um navegador ou como parte do sistema operacional (HAMILTON, 1996).

A origem da linguagem foi em 1990 com James Gosling, um desenvolvedor da *Sun Microsystems*, que fazia parte de um grupo de pesquisa para investigação de técnicas avançadas para desenvolvimento de *software*, voltadas para uma grande variedade de dispositivos de rede e sistemas integrados.

Como a maioria dos programas orientados a objetos, todas as variáveis e funcionalidades em um programa Java estão contidas em objetos que por sua vez são definidos por meio de classes.

Antes da linguagem Java era difícil gerenciar as informações na internet, portanto, desenvolver e utilizar aplicações que acessassem informações e execução de aplicações de forma eficiente era considerado um “trauma” para os programadores (YOURDON, 1996). Hoje, com a linguagem JAVA, a integridade do programa é garantida em tempo de execução como mostrado na Figura 8.3.

Figura 8. 3- Procedimentos de execução de instruções utilizando Java

Extraído de: YOURDON, 1996

Anexo C – SIMET de cada usuário

Resumo dos resultados

Velocidade: 9.64 Mbit/s **Bom!**
Latência: 12 ms **Bom!**
Perda de Tráfego: 0.00% **Bom!**

[Histórico de Testes](#)
[Sobre o SIMET](#)
[Compartilhe](#)
[Compartilhe](#)

Forneça seu CEP

[Esqueci meu CEP](#)

Acessando de casa ou trabalho?

para ajudar a melhorar a Internet Brasileira

em casa

Forneça a velocidade contratada

para podermos avaliar o teste

IP: 187.38.115.184

Data: 17/10/GMT-0200 ::

1 Latência

2 Jitter

3 TCP

4 UDP

5 Concluído

[Escolha de servidor para teste](#)

Velocidade (UDP)

Upload: 774 kbit/s

Download: 9.66 Mbit/s **Bom!**

Velocidade (TCP)

Upload: 753 kbit/s

Download: 9.64 Mbit/s **Bom!**

Jitter

Upload: 1 ms **Bom!**

Download: < 1 ms **Bom!**

Latência (Ping) 12 ms **Bom!**

ceptro.br nic.br cgi.br

Resumo dos resultados

Velocidade: 9.60 Mbit/s **Bom!**
Latência: 12 ms **Bom!**
Perda de Tráfego: 0.00% **Bom!**

[Histórico de Testes](#)
[Sobre o SIMET](#)
[Compartilhe](#)
[Compartilhe](#)

Forneça seu CEP

[Esqueci meu CEP](#)

Acessando de casa ou trabalho?

para ajudar a melhorar a Internet Brasileira

em casa

Forneça a velocidade contratada

para podermos avaliar o teste

IP: 187.38.115.184

Data: 17/10/GMT-0200 ::

1 Latência 2 Jitter 3 TCP 4 UDP 5 Concluído

Escolha de servidor para teste

Velocidade (UDP)

Upload: 754 kbit/s

Download: 9.58 Mbit/s **Bom!**

Velocidade (TCP)

Upload: 754 kbit/s

Download: 9.60 Mbit/s **Bom!**

Jitter

Upload: 1 ms **Bom!**

Download: < 1 ms **Bom!**

Latência (Ping) 12 ms **Bom!**

ceptro.br nic.br cgl.br

Resumo dos resultados

Velocidade: 556 kbit/s Bom!
Latência: 18 ms Bom!
Perda de Tráfego: 0.00% Bom!
Banda contratada: 512 kbit/s

[Histórico de Testes](#)

[Sobre o SIMET](#)

[Compartilhe](#)

[Compartilhe](#)

Forneça seu CEP

[Esqueci meu CEP](#)

Acessando de casa ou trabalho?

para ajudar a melhorar a Internet Brasileira

em casa

Forneça a velocidade contratada

para podermos avaliar o teste

IP: 201.52.117.94

Data: 18/10/2011 16:53:40

Latência

Jitter

TCP

UDP

Concluído

[Escolha de servidor para teste](#)

Velocidade (UDP)

Upload: 0 kbit/s

Download: 553 kbit/s Bom!

1.00 Mbit/s

1.00 Mbit/s

500 kbit/s

0 kbit/s

Upload
Download

Upload
Download

Velocidade (TCP)

Upload: 243 kbit/s

Download: 556 kbit/s Bom!

1.00 Mbit/s

500 kbit/s

0 kbit/s

Upload
Download

Upload
Download

Jitter

Upload: 129 ms Ruim!

Download: 7 ms Bom!

1000 ms

500 ms

0 ms

Upload
Download

Upload
Download

Latência (Ping) 18 ms Bom!

75 ms

50 ms

Velocidade (TCP)

Upload: 243 kbit/s

Download: 556 kbit/s Bom!

1.00 Mbit/s

500 kbit/s

0 kbit/s

Upload
Download

Upload
Download

Jitter

Upload: 129 ms Ruim!

Download: 7 ms Bom!

1000 ms

Upload