

CENTRO PAULA SOUZA
FACULDADE DE TECNOLOGIA DE MOCOCA
CURSO SUPERIOR DE TECNOLOGIA EM INFORMÁTICA COM
ÊNFASE EM BANCO DE DADOS

Altiéris Marcelino Peixoto

INDEXAÇÃO DE IMAGENS MÉDICAS VIA PLATAFORMA WEB –
UMA ABORDAGEM PRÁTICA UTILIZANDO O POSTGRESQL

Mococa, SP
2º semestre / 2010

ALTIÉRIS MARCELINO PEIXOTO

**INDEXAÇÃO DE IMAGENS MÉDICAS VIA PLATAFORMA WEB –
UMA ABORDAGEM PRÁTICA UTILIZANDO O POSTGRESQL**

Monografia apresentada à Faculdade de
Tecnologia de Mococa, como parte dos
requisitos para a obtenção do título de
Tecnólogo em Informática com Ênfase em
Banco de Dados.

Orientador: **Prof. Ms. Geraldo Henrique
Neto**

**Mococa, SP
2º Semestre / 2010**

Dedico este trabalho aos meus pais, que tanto fizeram e ainda fazem pela minha realização pessoal e profissional.

AGRADECIMENTOS

Ao Centro Paula Souza, pela oportunidade.

À Faculdade de Tecnologia de Mococa, pela formação acadêmica.

Ao professor Geraldo Henrique Neto, pela orientação, apoio e motivação em todos os momentos de desenvolvimento deste trabalho.

Ao professor Fernando Vieira Duarte, pelo apoio e auxílio no desenvolvimento deste trabalho.

Aos professores do curso de Informática com Ênfase em Banco de Dados, por terem contribuído tanto na minha formação acadêmica e profissional.

A minha família, especialmente meus pais, pelo ensino dos valores morais que uma pessoa de bem deve ter, pela educação que me passaram, pelo apoio financeiro, pelo auxílio nas horas difíceis e pela vibração conjunta nas conquistas. Muito obrigado a essas pessoas que iluminam a cada dia mais e mais a minha vida.

Aos amigos, que mesmo com a distancia e a falta de tempo para visitá-los, eu sei que torcem pela minha vitória, assim como eu torço pela vitória em suas vidas.

Aos colegas de classe, por termos compartilhado juntos três anos de nossa vida em pró a um futuro mais próspero.

A Deus.

“As idéias simples só estão ao alcance de
espíritos complexos.”

(Rémy de Gourmont)

RESUMO

Atualmente, a atividade médica lida com significantes volumes de informações. Entre essa gama de informações, as imagens médicas possuem uma importância expressiva para a promulgação de diagnósticos fidedignos. O uso de aplicativos os quais implementam segurança, armazenamento e recuperação em tempo hábil destas promovem grande ajuda para o cotidiano médico. Através da problemática do armazenamento de imagens em arquivos no sistema operacional, teve-se a idéia de indexá-las e organizá-las em um SGBDs (Sistema Gerenciadores de Banco de Dados), utilizando um modelo de banco de dados gerado pelo estudo da ontologia e taxonomia da medicina. Para a viabilidade deste trabalho foram utilizadas tecnologias *free*, de forte documentação e apoio como o PostgreSQL e PHP. O aplicativo desenvolvido proporciona um método de indexação de imagens baseado em recursos *Web*, podendo vir a colaborar na atividade da prática ou na docência.

Palavras-Chave: Diagnóstico. Imagens. Taxonomia. Ontologia. PostgreSQL. PHP. *Web*.

ABSTRACT

Currently, the medical practice handles significant volumes of information. Within this range of information, medical images have a significant importance to the enactment of reliable diagnoses. The use of applications which implement security, storage and timely retrieval of such great help to promote the medical practice. Through the issue of storing images in files in the operating system, had the idea of indexing them and organize them in a DBMS (System Managers Database) using a model database generated by the study of ontology and taxonomy of medicine. To study the feasibility of technologies were used free of documentation and strong support as the PostgreSQL and PHP. The application developed provides a method of indexing images based on web resources, and may collaborate in the activity of the practice or teaching.

Keywords: Diagnosis. Pictures. Taxonomy. Ontology. PostgreSQL. PHP. Web

LISTA DE FIGURAS

| | |
|---|----|
| Figura 1: Ontologia do domínio de Átomo. | 19 |
| Figura 2: Utilização da Ontologia de Átomo em Base de Conhecimento | 19 |
| Figura 3: Estados e atividades no ciclo de vida da ontologia..... | 23 |
| Figura 4: Modelo conceitual da hierarquia taxonômica da Medicina. | 25 |
| Figura 5: Exemplo de hierarquia taxonômica da Ginecologia.(exames ginecológicos) .. | 26 |
| Figura 6: Metadados relacionados a imagem mamográfica..... | 29 |
| Figura 7:Tabela pg_largeobject. | 34 |
| Figura 8:Modelagem conceitual do banco de dados..... | 45 |
| Figura 9: Modelagem Lógica do banco de dados..... | 47 |
| Figura 10: Tela principal do sistema | 52 |
| Figura 11: Tela de <i>upload</i> de imagens..... | 53 |
| Figura 12: <i>Upload</i> realizado | 53 |
| Figura 13: Tela de busca de imagens..... | 54 |
| Figura 14: Painel de imagens recuperadas | 55 |
| Figura 15: Mamografia ampliada | 55 |
| Figura 16: Diagnóstico e Observações relacionados a imagem selecionada | 56 |
| Figura 17: Tela de Busca de imagens para desativação/ativação | 57 |
| Figura 18: Desativação de imagens | 57 |

LISTA DE TABELAS

| | |
|---|----|
| Tabela 1 - Funções de manipulação de objetos grandes com SQL..... | 35 |
| Tabela 2 - Octetos com seqüência de escape para literais <i>bytea</i> | 36 |

LISTA DE SIGLAS

| | |
|-------|--|
| ACID | Atomicidade, Consistência, Isolamento e Durabilidade |
| ARO | Army Research Office |
| BD | Banco de Dados |
| BLOB | Binary Large Object |
| CAD | Computer Aided Diagnose |
| CR | Radiografia Digital |
| CSS | Cascading Style Sheets |
| CT | Tomografia Computadorizada |
| DARPA | Defense Advanced Research Projects Agency |
| DER | Diagrama Entidade - Relacionamento |
| DICOM | Digital Imaging Communications in Medicine |
| DOM | Document Object Model |
| ER | Entidade - Relacionamento |
| GB | Gigabyte |
| GPL | GNU General Public License |
| HTML | Hypertext Markup Language |
| IEEE | Institute of Electrical and Electronic Engineer |
| JPEG | Join Pictures Expert Group |
| MB | Megabyte |
| MIT | Massachusetts Institute of Technology |
| MRI | Ressonância Magnética |
| NCSA | National Center for Super Computing Applications |
| NFS | National Science Foundation |
| OID | Object Identifier |
| PDF | Portable Document Format |
| PHP | PHP: Hypertext Preprocessor |
| PNG | Portable Network Graphics |
| SGBD | Sistema Gerenciador de Banco de Dados |
| SQL | Structured Query Language |
| XML | Extensible Markup Language |

SUMÁRIO

| | | |
|----------|--|-----------|
| 1 | INTRODUÇÃO | 14 |
| 2 | A ORGANIZAÇÃO DA INFORMAÇÃO | 16 |
| 2.1 | ONTOLOGIA..... | 16 |
| 2.1.1 | O emprego da Ontologia em Bases de Conhecimento | 18 |
| 2.1.2 | Abordagens para o desenvolvimento de Ontologias | 20 |
| 2.1.3 | Metodologia para o desenvolvimento de Ontologias | 21 |
| 2.2 | TAXONOMIA | 24 |
| 2.3 | PROPOSTA DE UM MODELO CONCEITUAL DE TAXONOMIA..... | 25 |
| 3 | ARMAZENAMENTO DE DADOS NÃO CONVENCIONAIS | 27 |
| 3.1 | DADOS NÃO CONVENCIONAIS | 27 |
| 3.2 | MANIPULAÇÃO DE DADOS NÃO CONVENCIONAIS EM SGBD | 27 |
| 3.3 | INDEXAÇÃO DE DADOS NÃO CONVENCIONAIS ATRAVÉS DE METADADOS29 | |
| 4 | SGBD POSTGRESQL | 32 |
| 4.1 | DESCRIÇÃO DO SGBD POSTGRESQL | 32 |
| 4.2 | MANIPULAÇÃO DE DADOS NÃO CONVENCIONAIS NO POSTGRESQL | 33 |
| 4.2.1 | Manipulação de Objetos Grandes no PostgreSQL..... | 33 |
| 4.2.1.1 | Funções utilizadas com SQL | 35 |
| 4.2.2 | BYTEA | 36 |
| 5 | FERRAMENTA DE INDEXAÇÃO E VISUALIZAÇÃO DE IMAGENS MÉDICAS.. | 38 |
| 5.1 | DESCRIÇÃO DO SISTEMA | 38 |
| 5.2 | REQUISITOS DO SISTEMA | 39 |

| | |
|---|-----------|
| 5.2.1 Requisitos Funcionais | 39 |
| 5.2.2 Requisitos não Funcionais | 39 |
| 5.2.3 Restrições do Sistema | 40 |
| 5.3 TECNOLOGIAS UTILIZADAS | 40 |
| 5.3.1 Servidor Apache | 41 |
| 5.3.2 PHP : Hipertext Preprocessor | 41 |
| 5.3.3 jQuery | 42 |

| | |
|---|-----------|
| 6 IMPLEMENTAÇÃO DO SISTEMA | 44 |
|---|-----------|

| | |
|--|-----------|
| 6.1 MODELAGEM DO BANCO DE DADOS | 44 |
| 6.1.1 Modelagem Conceitual | 44 |
| 6.1.2 Modelagem Lógica | 46 |
| 6.1.3 Modelagem Física | 48 |
| 6.2 MÉTODO DE UPLOAD DE IMAGENS..... | 48 |
| 6.3 MÉTODO DE VISUALIZAÇÃO DE IMAGENS | 49 |

| | |
|---|-----------|
| 7 DEMONSTRAÇÃO DA FERRAMENTA | 52 |
|---|-----------|

| | |
|--|-----------|
| 7.1 MÓDULO DE UPLOAD DE IMAGENS MÉDICAS | 52 |
| 7.2 MÓDULO DE RECUPERAÇÃO DE IMAGENS MÉDICAS | 54 |
| 7.4 MÓDULO DE ATIVAÇÃO E DESATIVAÇÃO DE IMAGENS | 56 |

| | |
|-------------------------------------|-----------|
| 8 CONSIDERAÇÕES FINAIS | 58 |
|-------------------------------------|-----------|

| | |
|-------------------------|-----------|
| REFERÊNCIAS..... | 59 |
|-------------------------|-----------|

| | |
|------------------------|-----------|
| APÊNDICES | 61 |
|------------------------|-----------|

| | |
|--|-----------|
| APENDICE A – MODELAGEM FÍSICA DO BANCO DE DADOS | 62 |
|--|-----------|

| | |
|---|-----------|
| ANEXOS | 65 |
| ANEXO A – FUNÇÕES DE MANIPULAÇÃO DE LARGE OBJECTS NO POSTGRESQL..... | 66 |

1 INTRODUÇÃO

Atualmente a computação tem aplicabilidade em diversas áreas, tendo como destaque a área médica. A utilização de imagens médicas para o auxílio no diagnóstico (CAD) *Computer Aided Diagnose* atualmente vem sendo utilizado de maneira intensa. Tecnologias como Tomografia computadorizada (CT), Ultrassonografia, Ressonância Magnética (MRI) e Radiografia Digital (CR) geram uma grande quantidade de imagens digitais, necessitando da implementação de algoritmos específicos para o armazenamento e indexação destes objetos.

A utilização de SGBDs (Sistema de Gerenciador de Banco de Dados) se torna sobretudo um recurso imprescindível para o armazenamento de imagens médicas, implementando tecnologias diferenciais como: a utilização de uma linguagem padrão SQL (*Structured Query Language*), suporte a transações ACID (Atomicidade, Consistência, Isolamento e Durabilidade), uso de *logs*¹ nativo, dentre outros. Tais tecnologias promovem facilitadores para a recuperação, classificação e indexação de patologias, constituindo assim uma taxonomia específica para área médica.

Este trabalho tem como objetivo o desenvolvimento de uma base de dados para o armazenamento e organização de imagens médicas, juntamente com uma ferramenta *Web* para a indexação e recuperação posterior dos objetos e suas respectivas descrições (anamnese e diagnóstico) vindo a corroborar na atividade prática/docência.

Através de uma metodologia de pesquisa científica e tecnológica e utilizando pesquisas bibliográficas e da plataforma *Web*, procura-se constituir o conteúdo necessário para a realização deste trabalho.

A organização do trabalho apresentado foi segmentado da seguinte forma:

No capítulo 2 são descritos conceitos de ontologia e taxonomia para a organização das informações em base de dados.

¹ Registro de atividades gerado por programas de computador

O capítulo 3 discorre sobre o armazenamento de dados não convencionais em SGBD. Uma técnica emergente que esta sendo amplamente utilizada para grandes volumes de dados.

O capítulo 4 relata informações acerca do SGBD PostgreSQL, descrevendo sua história e seus recursos para o armazenamento e recuperação de dados não convencionais.

O capítulo 5 aborda a descrição da ferramenta de indexação e visualização de imagens médicas, assim como seus requisitos, restrições e tecnologias utilizadas para a implementação e viabilização desta ferramenta.

O capítulo 6 apresenta as etapas de modelagem do banco de dados, assim como os métodos desenvolvidos para a indexação e recuperação das imagens médicas no sistema.

O capítulo 7 aborda uma demonstração da ferramenta desenvolvida, apontando seus principais módulos desenvolvidos.

O capítulo 8 são as considerações finais e propostas para trabalhos futuros.

2 A ORGANIZAÇÃO DA INFORMAÇÃO

2.1 ONTOLOGIA

“Originário da filosofia, o termo ontologia, inicialmente introduzido por Aristóteles, vem do grego *Ontos* que tem o significado de ser e *Logos* que significa palavra. A ontologia é um ramo da filosofia que lida com a natureza e a organização do ser, sendo que a base da filosofia ontológica é tentar responder questões como “O que é um ser?” e “Quais são as características de todos os seres?””. (MAEDCHE, 2002 apud GUIMARÃES, 2002, p.49)

De acordo com Silberschatz, Korth e Sudarshan (2006), “Ontologias são estruturas hierárquicas que refletem relacionamentos entre conceitos”.

The word ontology comes from the Greek *ontos* for being and *logos* for word. It is a relatively new term in the long history of philosophy, introduced by the 19th century German philosophers to distinguish the study of being as such from the study of various kinds of beings in the natural sciences. The more traditional term is Aristotle's word category (*Kathgoria*), which he used for classifying anything that can be said or predicated about anything (SOWA, 2000 apud ROUSSEY, 2005, p.1)

O termo ontologia é atualmente utilizado de forma maciça por comunidades que estudam inteligência artificial com o foco em representação do conhecimento. De acordo com Russel e Norvig (2006, p.309), a ontologia procura organizar tudo no mundo em uma hierarquia de categorias.

A palavra ontologia representa uma teoria específica sobre a natureza de ser ou existir. A ontologia determina os tipos de itens que existem, mas não determina suas propriedades específicas e seus inter-relacionamentos (RUSSEL; NORVIG, 2004, p. 253).

A utilização de ontologias pode proporcionar um entendimento amplo das características dos conceitos do domínio estudado, além de ser extensível, isto é, novos termos podem ser agregados; compartilhado para utilização com outras ontologias ou ferramentas, ou pode ser reutilizado para outros fins dentro de seu escopo.

De acordo com Smith (1996 apud OLIVEIRA, 2006, p.13), uma ontologia é desenvolvida para atender uma das seguintes finalidades:

- Permitir o compartilhamento de conhecimento entre múltiplos agentes;
- Permitir a melhor compreensão de certa área do conhecimento e ajudar as pessoas envolvidas a buscar um consenso sobre o significado e importância da ontologia aplicada sobre o domínio estudado;
- Ampliar o conhecimento das pessoas envolvidas na construção da ontologia do domínio estudado;
- Reduzir o conflito de comunicação entre pessoas na organização, ao estabelecer ontologias que reduzem confusões terminológicas e conceituais dentro da organização;
- Corroborar a tradução entre diferentes linguagens e representações, podendo ajudar na interoperabilidade entre sistemas.

Segundo Maedche (2002 apud MATTOS; SIMÕES; FARIAS, 2003) existem quatro tipos de ontologias segundo sua classificação:

- a) Ontologias de nível superior ou genéricas:** Constituem ontologias de nível mais abrangente formadas por termos genéricos e tem como principal característica servir de base para outras ontologias por ser independente de um problema ou domínio específico.
- b) Ontologias de domínio:** Constituem ontologias que possuem um conjunto de termos relacionados de forma a especializar-se com base nos conceitos introduzidos na ontologia de nível superior.
- c) Ontologias de tarefas:** Busca a especialização a partir de ontologias genéricas com o intuito de referir-se a resolução de problemas, entretanto esta também pode ser utilizada para outras aplicações fazendo com que também seja conhecida como ontologia de aplicação genérica. Através da especialização de conceitos inseridos nas ontologias de nível superior, descreve um vocabulário conexo a uma tarefa ou atividade genérica.
- d) Ontologias de aplicação:** Possuem menor grau de reusabilidade, sendo usadas dentro de aplicações. Especializa conceitos de ontologias de domínio e de tarefas.

A partir da idéia de que uma ontologia é um grupo de termos que descrevem um domínio, a organização desses termos de forma hierárquica (taxonomia) pode ser utilizada para constituir a estrutura de uma base de conhecimento.

2.1.1 O emprego da Ontologia em Bases de Conhecimento

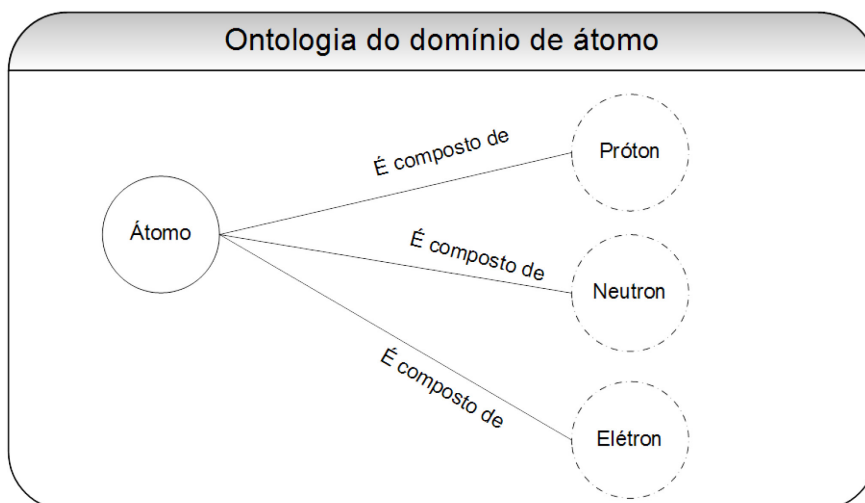
A ontologia é responsável por ministrar um agregado de conceitos e termos para referir a um determinado domínio, ao passo que a base de conhecimento emprega esses termos para modelar certa realidade. Se essa realidade for alterada a base de conhecimento também será alterada, entretanto desde que o domínio não seja alterado, a ontologia permanecerá inalterada (GUIMARÃES, 2002, p.50).

É importante ressaltar que uma ontologia sobre determinado domínio deve identificar conceitos relevantes sobre este, assim como referir-se de forma compartilhada, ou seja, os termos usados para constituir a ontologia do domínio devem apontar concordância de conceitos entre um grupo de pessoas e não ser restrito a apenas alguns indivíduos. Segundo Studer (1998 apud ROUSSEY, 2005, p.1), *“An ontology is a formal and explicit specification of a share conceptualization”*.

Além disso, na etapa de agregar conhecimento sobre determinado domínio, é de extrema relevância que o engenheiro do conhecimento, assim descrito por Russel e Norvig (2004, p. 253), “[...] já deve ser um especialista no domínio ou talvez precise trabalhar com especialistas reais para extrair o que eles conhecem – um processo chamado **aquisição do conhecimento**. [...] A idéia é compreender o escopo da base de conhecimento, determinada pela tarefa, e entender como o domínio funciona”.

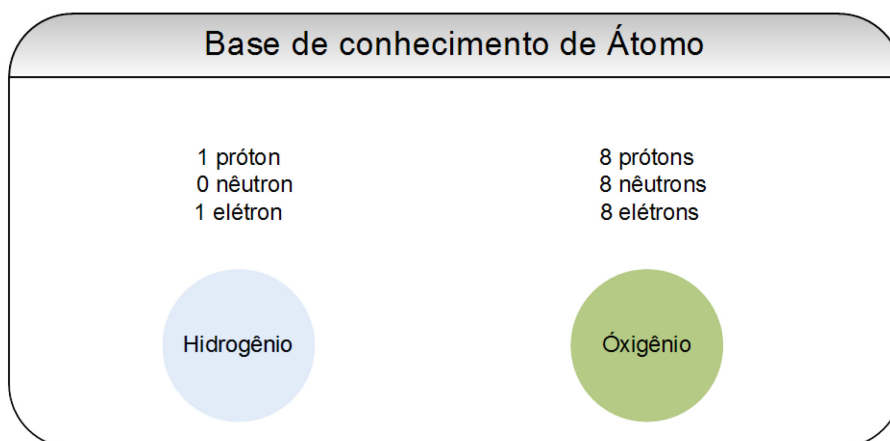
Para exemplificar pode-se visualizar na figura 1 o domínio de um “Átomo”. Uma possível ontologia para este domínio poderia ser constituída de termos específicos como, por exemplo, próton, nêutron, elétron, etc., e relações entre estes termos como um átomo é composto de prótons. Tais termos podem vir a constituir o esqueleto de uma base de conhecimento sobre átomos de elementos da tabela periódica, como apresenta a figura 2. Ao inserir informações sobre um átomo de hidrogênio na base de conhecimento, este contém 1 próton, 1 elétron e 0 nêutron, ao passo que ao inserir informações da estrutura de um átomo de oxigênio na base

de conhecimento este contém 8 prótons, 8 nêutrons e 8 elétrons. Portanto embora as realidades dos dois átomos sejam distintas, a ontologia aplicada na base de conhecimentos continua sendo a mesma. Adaptado de (GUIMARÃES, 2002, p.50).



Fonte: Adaptado de (GUIMARÃES, 2002, p.52)

Figura 1: Ontologia do domínio de Átomo.



Fonte: Adaptado de (GUIMARÃES, 2002, p.52)

Figura 2: Utilização da Ontologia de Átomo em Base de Conhecimento

2.1.2 Abordagens para o desenvolvimento de Ontologias

O desenvolvimento de uma ontologia pode ser realizado através de várias abordagens, sendo que em cada caso deve-se escolher a mais apropriada seguindo como critério a classe da ontologia a ser desenvolvida.

De acordo com Holsapple e Joshi (2002 apud OLIVEIRA, 2006, p.14) o processo de criação de ontologia pode ser abordado das seguintes formas:

- **Abordagem partindo de uma inspiração:** A partir de um questionamento sobre o porquê a ontologia é necessária, o desenvolvedor constrói a ontologia que visa solucionar a necessidade encontrada fazendo uso de sua imaginação, criatividade e visão pessoal. Nesta abordagem há a vantagem de ser possível a descoberta de uma ontologia inovadora para a resolução do problema do domínio em questão. Entretanto, esta mesma abordagem pode vir de uma visão muito superficial do próprio desenvolvedor, podendo gerar problemas em sua adoção;
- **Abordagem indutiva:** A ontologia é construída a partir de observação, exame e análise de casos específicos no domínio de interesse. A ontologia decorrida do estudo de um caso específico pode ser usada para outros casos do mesmo domínio. Nessa abordagem pode se destacar como vantagem o uso de uma mesma ontologia para a solução de vários outros casos dentro de um domínio específico. Contudo pode-se destacar como desvantagem dessa abordagem a incapacidade dessa ontologia cobrir domínios menos específicos, ou seja, mais amplos;
- **Abordagem dedutiva:** A ontologia é desenvolvida utilizando-se alguns princípios gerais e empregando-os para que se possa construir uma ontologia específica. Processos como escolher e alterar as noções gerais são incluídas para que se possa personalizar um subconjunto do domínio. A partir de uma ontologia mais genérica de um domínio, pode-se chegar a uma ontologia mais específica restrita dentro de um subconjunto do domínio. Por exemplo, partindo-se de um domínio mais genérico de computadores pode-se chegar a um domínio de laptops² que é um subconjunto de computadores.

² Computador portátil

Como característica elementar desta abordagem pode-se destacar que para se gerar uma ontologia mais específica, há a necessidade de que inicialmente já exista uma ontologia genérica;

- **Abordagem sintética:** Nessa abordagem, o desenvolvedor identifica um conjunto base de ontologias, em que nenhuma substitua a outra. A fim de construir uma ontologia unificada, é realizada uma composição das características dessas ontologias, junto com outros conceitos que pertencem ao domínio em questão. Essa ontologia unificada pode fornecer uma forma conexa para a comunicação entre os usuários das ontologias múltiplas. Uma das barreiras a ser quebrada para o sucesso do uso desta abordagem está na dificuldade de estabelecer uma ligação harmônica entre um grande grupo de ontologias;
- **Abordagem colaborativa:** A construção da ontologia tenta reproduzir a experiência e os pontos de vista das pessoas envolvidas no desenvolvimento desta, onde seu processo de criação se encontra de forma compartilhada entre um grupo de pessoas. Nesta abordagem pode-se destacar o aumento das chances de sucesso na aceitação da ontologia para o emprego em determinado domínio. Uma vez que esta ontologia é constituída de conhecimento compartilhado entre um grupo de pessoas onde o nível de conhecimento sobre o assunto é mais versatilizado, tornando a ontologia mais elaborada do que nas abordagens anteriores.

2.1.3 Metodologia para o desenvolvimento de Ontologias

O emprego da ontologia na ciência da computação ainda é de fato muito novo.

De acordo com Oliveira (2006, p.15), as metodologias para a construção de ontologias ainda se encontram em desenvolvimento, o que faz com que seja uma prática comum entre os desenvolvedores de ontologias, pularem a fase de desenvolvimento de conceitos relacionados ao domínio e seguirem diretamente para a fase de implementação. Isto pode ser um erro grave, e pode trazer vários

problemas para o desenvolvedor e para o sistema futuramente, tais como dificuldade de reuso da ontologia, problemas de entendimento do perito na ontologia com o código da aplicação (uma vez que os conceitos do domínio estarão juntos do código-fonte), dificuldades na construção de ontologias mais complexas, etc.

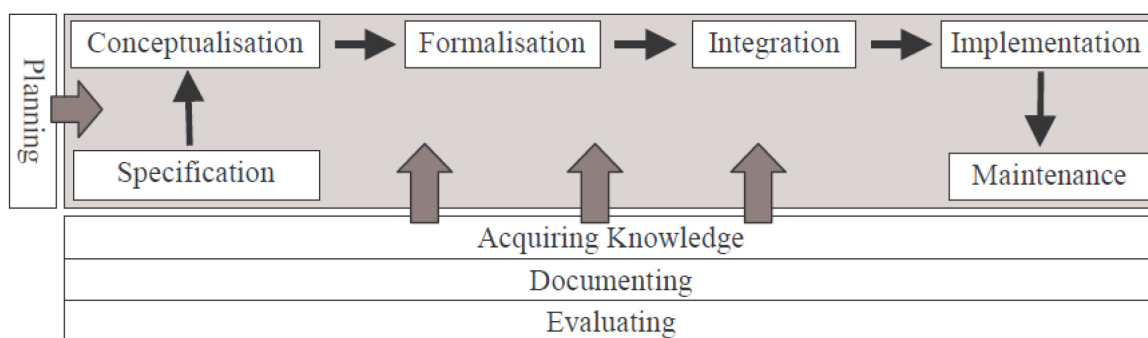
Uma das metodologias mais conhecidas para o desenvolvimento de ontologias é a *Methontology*, a única metodologia destinada ao desenvolvimento de ontologias que especifica o ciclo de desenvolvimento e que baseia-se nas normas do *Institute of Electrical and Electronic Engineer* (IEEE). Adaptado de (OLIVEIRA, 2006,p.19)

Segundo Fernandez et al (1997 apud OLIVEIRA,2006,p.20), a *Methontology* foi inicialmente proposta por Mariano Fernandez, Asunción Gómez – Perez e Natalia Juristo e desenvolvida pelo laboratório de Inteligência Artificial da Universidade Politécnica de Madri. Esta metodologia se baseia em um ciclo de vida evolutivo baseado em protótipos juntamente com o método de reengenharia para a construção de ontologias.

A figura 3 apresenta a estrutura de estados e atividades do ciclo de vida de desenvolvimento da ontologia proposto pela *Methontology*, onde segundo (OLIVEIRA,2006,p.20), esta é dividida pelas seguintes fases:

- **Planejamento** – Responsável por identificar as principais tarefas da ontologia e a utilização dos recursos a serem empregados.
- **Especificação** – Responsável por determinar o propósito da construção da ontologia, seu escopo e seus usuários. Esta fase tem como objetivo a elaboração da documentação da ontologia a ser desenvolvida.
- **Aquisição de Conhecimento** – Fase responsável pela captação de fontes de conhecimento, tais como entrevistas com o expert no domínio da ontologia a ser desenvolvida, livros e revistas sobre o domínio.
- **Conceitualização** – Fase responsável pela criação de um modelo conceitual para a descrição do problema e sua possível solução, a partir do vocabulário adquirido nas fases anteriores. É considerada a principal fase desta metodologia.
- **Formalização** – Esta fase é responsável pela criação de um modelo formal a partir do modelo conceitual desenvolvido no estágio anterior.

- **Integração** – Fase responsável pela integração da ontologia em desenvolvimento com outras existentes, de forma que se busque ontologias que melhor de adéquem ao seu propósito.
- **Implementação** – Fase de transformação da ontologia, para uma linguagem de programação, de modo que ela seja computável.
- **Avaliação** – Presente em todas as fases do ciclo de vida proposto pela Methontology, a avaliação trata da verificação e validação dos termos adquiridos de modo a reduzir a probabilidade de erro na implementação.
- **Documentação** – Executada em todas as fases do ciclo de vida do desenvolvimento da ontologia, a documentação é responsável por auxiliar na manutenção e reutilização da ontologia. É composta de documentos como especificação dos requisitos, modelo conceitual e avaliação.
- **Manutenção** – Fase responsável pela atualização da ontologia quando necessária.



Fonte: (FERNANDEZ et al. ,1997 apud ROUSSEY,2005)

Figura 3: Estados e atividades no ciclo de vida da ontologia

Segundo Guimarães (2002, p.68), deve se destacar as seguintes observações sobre a *Methontology*:

- As atividades de aquisição de conhecimento, avaliação e documentação são executadas em todos os estágios do ciclo de vida da ontologia;
- A primeira atividade a ser executada é o planejamento;
- A atividade de aquisição de conhecimento tem maior ênfase na fase de especificação, e vai decrementando à maneira que o ciclo de vida de construção da ontologia avança;

- A atividade de avaliação da ontologia tem maior ênfase nos estágios iniciais do ciclo de vida, diminuindo a probabilidade de erro, que poderia resultar na má construção da ontologia;
- A documentação é realizada durante todos os estágios do ciclo de vida da ontologia;

Esta metodologia não está restrita a um determinado domínio, podendo ser então aplicada em qualquer domínio.

2.2 TAXONOMIA

Segundo Bicudo (2004), a “Taxonomia é a ciência da identificação. Talvez, a mais velha de todas as ciências, pois nasceu com o homem, mas, com toda certeza, a mais necessária. Entretanto, paradoxalmente, é a menos valorizada de todas as ciências.”

Juntamente com a ontologia, a taxonomia foi, e ainda é amplamente utilizada para identificar, nomear e categorizar de forma hierárquica termos dentro de um domínio específico.

As taxonomias foram usadas explicitamente durante séculos em campos técnicos. Por, exemplo a biologia sistemática pretende fornecer uma taxonomia de todas as espécies vivas e extintas; a ciência da informação desenvolveu uma taxonomia de todos os campos do conhecimento, conhecida como o sistema decimal de Dewey; as autoridades tributárias e outros departamentos governamentais desenvolveram taxonomias extensas de ocupações e produtos comerciais(RUSSEL, NORVIG, 2004, p.312).

A utilização de taxonomia para a classificação e categorização de termos dentro de um determinado domínio pode facilitar o processo de pesquisa e compreensão deste, além de fazer com que objetivo de seu estudo se torne mais sistemático.

A organização de objetos em categorias é uma parte vital da representação de conhecimento. Embora a interação com o mundo ocorra no nível de objetos individuais, uma grande parte do raciocínio tem lugar no nível de categorias! (RUSSEL, NORVIG, 2004, p.311).

A partir da utilização da ontologia e taxonomia foi desenvolvido um modelo conceitual da hierarquia taxonômica da medicina que será explicado no próximo tópico.

2.3 PROPOSTA DE UM MODELO CONCEITUAL DE TAXONOMIA

A aplicação da taxonomia e ontologia dentro do objeto de estudo deste trabalho se dá pela classificação das patologias dentro de especialidades médicas, estas por sua vez são organizadas categoricamente em uma ou mais áreas da medicina.

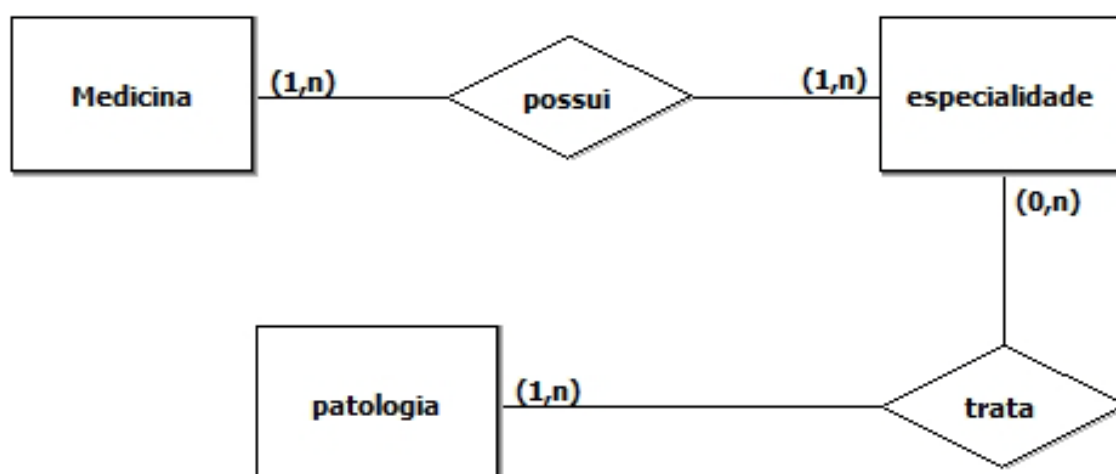


Figura 4: Modelo conceitual da hierarquia taxonômica da Medicina.

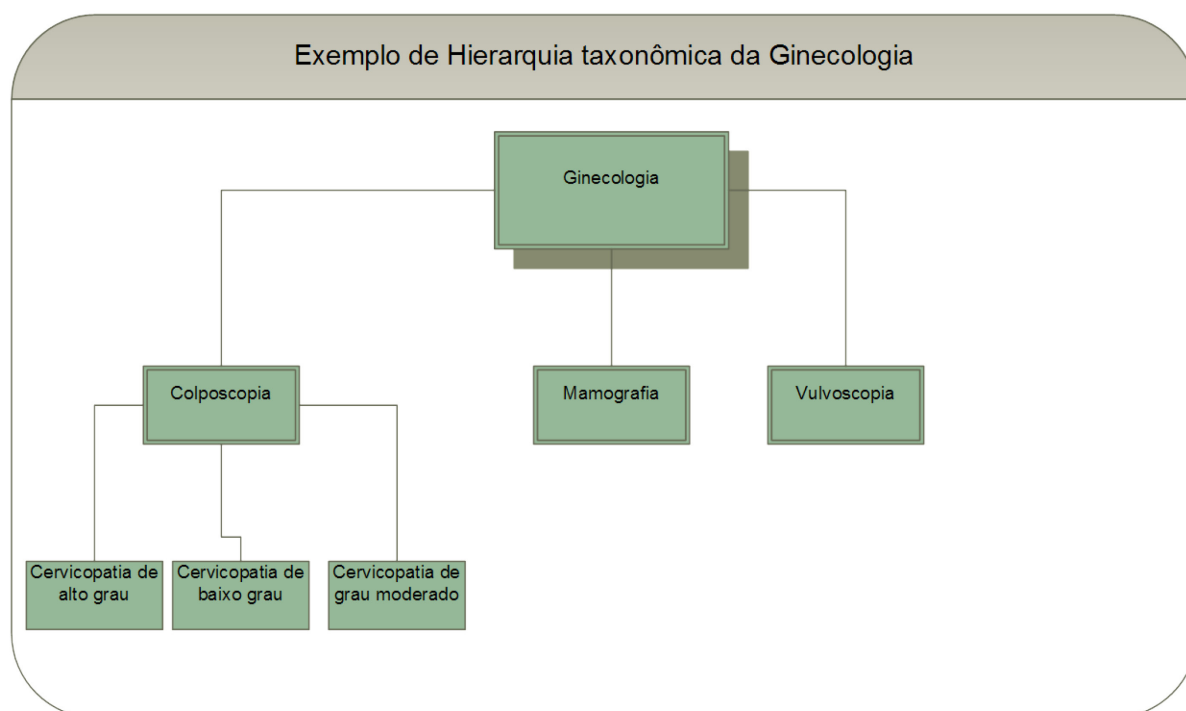


Figura 5: Exemplo de hierarquia taxonômica da Ginecologia.(exames ginecológicos)

Como pode-se visualizar na figura 4, a um modelo da hierarquia taxonômica aplicada à medicina pode ser desenvolvida como uma forma de modelo conceitual. Onde uma área médica pode conter uma ou mais especialidades médicas, e estas especialidades médicas podem tratar uma ou mais patologias, como pode ser exemplificado na figura 5.

Espera-se que este modelo possa servir como alicerce para a construção de uma base de conhecimento de imagens médicas a partir da indexação destas em uma base de dados.

3 ARMAZENAMENTO DE DADOS NÃO CONVENCIONAIS

3.1 DADOS NÃO CONVENCIONAIS

Também conhecidos como dados multimídia ou objetos multimídia, os dados não convencionais "[...] compreendem imagens (como fotografias ou desenhos), vídeos (como filmes, documentários ou vídeos caseiros), gravações (como canções, mensagens telefônicas ou falas) e documentos (como livros e artigos)[...]" (ELMASRI, NAVATHE, 2006, p. 562).

Uma forma cada vez mais popular de dados, os dados não convencionais são comumente armazenados fora do banco de dados (armazenamento do *path*³), geralmente em sistemas de arquivos. A manipulação de pequenos volumes de dados não convencionais é uma tarefa relativamente fácil em sistemas de arquivo, todavia quando se trata de um volume maior de informação, se torna obrigatório o uso SGBD para o armazenamento e recuperação destes objetos/informações.

Recentemente, tem crescido o interesse por banco de dados que armazenam dados multimídia, como imagens, áudio e vídeo. Hoje em dia, normalmente armazenam-se dados multimídia fora do banco de dados, em sistemas de arquivos. Geralmente, quando o número de objetos multimídia é relativamente pequeno, os recursos fornecidos por um banco de dados não são importantes. A funcionalidade do banco de dados torna-se importante quando é grande o número de objetos multimídia armazenados. Nesse caso, características como atualizações transacionais, facilidades de consulta e indexação tornam-se relevantes (SILBERSCHATZ; KORTH; SUDARSHAN, 1999, p.724).

3.2 MANIPULAÇÃO DE DADOS NÃO CONVENCIONAIS EM SGBD

De acordo com Vaz (2000 apud SILVA, 2006), os SGBDs possuem formas distintas de armazenar dados não convencionais. Estas serão destacadas adiante:

- **Referências externas:** caracterizado por armazenar no banco de dados somente o path (caminho) onde ficam armazenados efetivamente os objetos

³ Caminho dos diretórios do sistema operacional

multimídia. O SGBD não gerencia o objeto multimídia devido ao armazenamento externo, desprovendo então de integridade e consistência dos dados;

- **Armazenamento de dados multimídia não interpretados (campo BLOB) :** caracterizado pelo uso de estruturas denominadas blocos binários ou BLOBs (*Binary Large Objects*) para a realização do armazenamento dos dados multimídia.

Ao utilizar campos BLOB, os dados são armazenados em uma sequência de bytes não interpretados, o que impossibilita a extração de características semânticas dos objetos. Fazendo com que o banco de dados se torne apenas um repositório de objetos multimídia.

Em consequência a aplicação que manipula estes objetos deve ser capaz de prover formas de recuperar e exibir-los de forma consistente.

Todavia, é importante ressaltar que a recuperação de dados de mídia contínua, como áudio e vídeo deve ser constante (streaming⁴), caso contrário, eventualmente, poderá haver inconsistência nos dados ou estouro do buffer do sistema.

A recuperação de alguns tipos de dados, como áudio e vídeo, tem o requisito de que a remessa dos dados precisa prosseguir em um ritmo constante garantido. Esses dados as vezes são chamados de **dados isócronos**, ou **dados de mídia contínua**. Por exemplo, se os dados de áudio não forem fornecidos em tempo, haverá lacunas no som. Se os dados forem fornecidos muito rapidamente, os buffers do sistema poderão estourar, resultando em perda dos dados. (SILBERSCHATZ, KORTH, SUDARSHAN, 1999, p.619).(grifos do autor)

- **Funções externas:** são geralmente utilizadas para completar a insuficiência de recursos para a manipulação de objetos multimídia nos SGBDs;
- **Sistemas orientados a objetos ou relacionais estendidos:** nestes sistemas o desenvolvedor pode determinar tipos de dados e referenciá-los para as aplicações. A utilização de armazenamento de dados multimídia através de orientação a objetos é uma tentativa de junção dos dois métodos relacionados anteriormente. Entretanto cabe ao SGBD a forma de tratamento da orientação a objeto e uso da semântica dos dados.

⁴ Distribuição de informação multimídia através de pacotes em uma rede

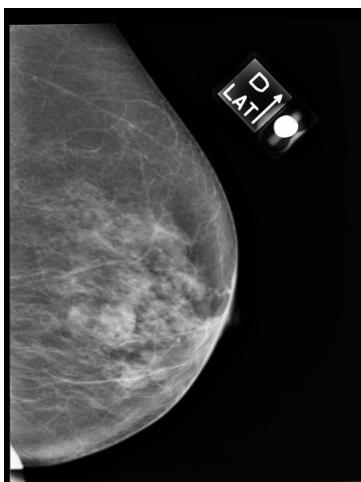
3.3 INDEXAÇÃO DE DADOS NÃO CONVENCIONAIS ATRAVÉS DE METADADOS

Ao armazenar dados não convencionais em banco de dados, pode-se usar estratégias de indexação com dados convencionais, de modo que ao realizar uma recuperação de imagens por exemplo, a busca se torne mais precisa e de fácil realização.

A indexação permite identificar e representar o conteúdo ou o assunto de um documento de forma a caracterizá-lo. Um documento deve ser indexado tendo-se em mente não apenas o seu armazenamento, mas principalmente sua futura busca e recuperação. (VAZ 2000 apud SILVA, 2006, pág. 10)

Uma técnica muito comum é a utilização de metadados, isto é, dados que descrevem outros dados, para indexar objetos multimídia, pois ao se associar estes metadados, pode-se ter uma descrição mais concisa do domínio onde o objeto multimídia se insere, suas características e informações adicionais.

Na figura 6, pode-se visualizar os metadados como: área médica, especialidade, e patologia, os quais são utilizados para a indexação de imagens médicas tal como a imagem mamográfica abaixo, auxiliando na recuperação automatizada das imagens relacionadas com uma patologia específica.



| Metadados | Valor |
|---------------|-------------|
| Área Médica | Ginecologia |
| Especialidade | Radiologia |
| Patologia | Bi-Rads |
| Categoria | 0 |
| Composição | 1 |

Figura 6: Metadados relacionados a imagem mamográfica.

De acordo com seu escopo, os metadados podem ser categorizados em diversos tipos. Segundo Niso e Vaz (2004 apud SILVA, 2006, pág.20) os tipos são classificados como:

Tipos gerais de metadados

- Metadados descritivos: são metadados que descrevem o objeto multimídia relacionado. Os metadados da figura 6 são avaliados como descritivos;
- Metadados estruturais: são metadados que definem a relação entre objetos. Determinam quais metadados podem ser utilizados a constituir um objeto composto;
- Metadados administrativos: colaboram na administração do sistema, sendo que estes determinam quando e como os metadados e objetos estarão disponíveis. Tais metadados determinam quais são os tipos de usuários que tem permissão a um determinado objeto multimídia.

Tipos de metadados voltados para objetos multimídia:

- Metadados para representação de tipos de mídia: os metadados para representação de tipos de mídia são responsáveis por descrever as necessidades técnicas para a representação dos objetos multimídia. Mídias de um mesmo gênero podem possuir diversas formas de digitalização, compactação e armazenamento. Os metadados pertinentes a representação de tipos de mídia fornecem informações para que o tipo de mídia seja manipulado de forma correta no momento de exibição. Metadados para o armazenamento de plug-ins, são tipos de metadados para representação de tipos de mídia;
- Metadados para classificação de conteúdo: responsáveis por correlacionar os metadados de acordo com os objetivos e regras implementados pelo sistema. Como por exemplo a indexação de metadados de objetos multimídia de acordo com o equipamento utilizado para sua aquisição.

- Metadados para composição de documentos: responsáveis por promover um conhecimento da interligação das mídias e dos documentos multimídia como páginas da Web.

A utilização de metadados para a indexação de dados não convencionais, promove uma facilitação na pesquisa destes, uma vez que cada metadado possui uma chave de identificação única que pode ser usado para consultas com base em SQL (*Structured Query Language*).

Ao empregar metadados para a indexação de dados não convencionais, deve-se alertar para o vocabulário utilizado, assim como os erros de grafia, que podem resultar em comprometimento da pesquisa e retorno dos dados.

Uma solução cabível para o vocabulário empregado como metadado é o uso de ontologias como apresentado no capítulo 2, permitindo descrever de forma concisa o domínio do objeto multimídia a ser armazenado.

4 SGBD POSTGRESQL

As seções de descrição do SGBD⁵ PostgreSQL e manipulação de dados não convencionais são baseadas no Manual do PostgreSQL⁶

4.1 DESCRIÇÃO DO SGBD POSTGRESQL

A história do PostgreSQL se inicia em 1986, onde liderado pelo professor Michael Stonebraker da Universidade da Califórnia juntamente com uma equipe de desenvolvedores e patrocinado pela DARPA (*Defence Advanced Research Projects Agency*), pelo ARO (*Army Research Office*), pela NFS (*National Science Foundation*) e pela ESL, Inc., deram início ao projeto até então nomeado de POSTGRES.

Desde então o POSTGRES passou por várias versões tendo seu número de usuários dobrado no ano de 1993. Entretanto a manutenção do código e suporte consumiam o tempo que deveria ser utilizado para pesquisa de banco de dados. Para contornar a situação o projeto POSTGRES teve seu fim na versão 4.2.

No ano de 1994, Andrew Yu e Jolly Chen implementaram um interpretador de linguagem SQL ao POSTGRES, alteraram seu nome para Postgres95 liberando seu código fonte na *Web* para que o SGBD se tornasse então um software livre.

No ano de 1996, o Postgres95 teve seu nome alterado para PostgreSQL iniciando-se assim a versão 6.0, disponível com este nome até os dias de hoje.

Atualmente, o SGBD PostgreSQL é tido como um dos Sistemas Gerenciadores de Banco de Dados mais robustos do mundo e atualmente apresenta as seguintes funcionalidades:

- Consultas preparadas;
- Cursores;
- Esquemas;
- Funções;
- Gatilhos;

⁵ Sistema Gerenciador de Banco de Dados

⁶ Disponível em <http://sourceforge.net/projects/pgdocptbr>

- Subconsultas;
- Transações e subtransações;
- Visões;
- Alta disponibilidade;
- Ambientes de desenvolvimento;
- Ferramentas de administração, monitoramento e análise;
- Online Backup;
- Replicação;
- Suporte as plataformas mais utilizadas;
- Suporte a várias línguas;
- Tablespace;
- Suporte a Objetos grandes (Large Objects) e XML (Extensible Markup Language);
- Suporte as linguagens de programação mais utilizadas incluindo C/C++, C#, Java, Perl, PHP, Python e outras;

4.2 MANIPULAÇÃO DE DADOS NÃO CONVENCIONAIS NO POSTGRESQL

O PostgreSQL prove duas formas de armazenamento de dados não convencionais, descritas a seguir:

- Utilizando a funcionalidade de objetos grandes (Large objects);
- Através do tipo de dado nomeado Bytea;

4.2.1 Manipulação de Objetos Grandes no PostgreSQL

Para suportar o armazenamento de dados não convencionais, o PostgreSQL prove a interface de armazenamento de objetos grandes, que possui a funcionalidade de acesso na forma de fluxo aos dados dos usuários.

Apesar de apresentar menor desempenho no acesso aos dados, esta forma de armazenamento fornece uma integridade de dados mais rigorosa, sendo que os dados são armazenados em uma tabela do sistema nomeada de *pg_largeobject* como é visualizado através da figura 7.

| Painel de saída | | | |
|-----------------|-------------|-------------------|--|
| Saída de Dados | | | |
| | Explain | Mensagens | Histórico |
| | loid oid | pageno integer | data bytea |
| 271 | 22983 | 1 | \006\0079\031\301\311\377\000\353P\006}\332\244\021\231\017\ |
| 272 | 22983 | 2 | |
| 273 | 22983 | 3 | |
| 274 | 22983 | 4 | |
| 275 | 22983 | 5 | |
| 276 | 22983 | 6 | \257\357\027\037\347\351R\253K)\221\237~\362\347\345+\307^ |
| 277 | 22983 | 7 | |
| 278 | 22983 | 8 | |
| 279 | 22983 | 9 | 7\357\302\363\264\222\241y\371\216:\232\216\335\244y \306~\C |
| 280 | 22983 | 10 | \232\004B\247\022\200\200\205\034\023\322\235\2746\025\0169\ |
| 281 | 22983 | 11 | AR\244x\030\343\0149\250\202\312S G\264\036\324e\230*7%Ns\ |
| 282 | 22983 | 12 | bB\344\263\034\266\321\234&\177\300\376\224\327\300\270\177 |
| 283 | 22983 | 13 | |
| 284 | 22983 | 14 | jY\021<FFR\012\310p\314p\316\007R}\207O\302\250]\351\211q& |
| 285 | 22983 | 15 | |
| 286 | 22983 | 16 | \231\021\332\030\306\245\230\257\013\234\034\216\230\374O\3! |
| 287 | 22983 | 17 | |
| 288 | 22983 | 18 | |

Figura 7:Tabela *pg_largeobject*.

O PostgreSQL desde a versão 7.1 possui o mecanismo apelidado de “*TOAST*” (fatias), isto é, o SGBD permite o armazenamento de valores de dados muito maiores que as páginas de dados, fazendo com que o objeto grande seja dividido em “pedaços”, e armazenado em linhas do banco de dados.

Um campo *large-object* pode armazenar dados com até 2 GB (*Gigabytes*), e dados fatiados com no máximo 1GB.

Para manipular objetos grandes existem funções preestabelecidas no PostgreSQL e que devem ser utilizadas dentro de um bloco de transação SQL. Estas funções serão mostradas a seguir de acordo com o manual do PostgreSQL.

4.2.1.1 Funções utilizadas com SQL

Foram expostas anteriormente funções do lado cliente, nomeadas também como *client side*, utilizadas para manipulação de objetos grandes. Entretanto somente algumas funções são utilizadas juntamente com SQL. Para exemplificar, será criada uma tabela com dois atributos:

```
create table imagem(
    id_imagem serial primary key,
    figura oid);
```

Onde :

- **id_imagem:** é a chave primária com tipo de dado serial da tabela imagem;
- **figura:** é o OID (Object Identifier) de indexação para a tabela pg_largeobject que armazenará o objeto grande;

As funções que podem ser chamadas junto com SQL serão exploradas na tabela 1 a seguir de acordo com o Manual do PostgreSQL:

Tabela 1 - Funções de manipulação de objetos grandes com SQL

| Operação | SQL |
|---|--|
| Retornar o OID do objeto grande novo e vazio | <code>SELECT lo_creat(-1);</code> |
| Apagar o objeto grande com OID igual a 173454 | <code>SELECT lo_unlink(173454);</code> |
| Inserir uma imagem no banco de dados | <code>INSERT INTO imagem (id_imagem, figura) VALUES (default, lo_import('/etc/imagem.jpeg'));</code> |
| Exportar um objeto grande para uma pasta do sistema operacional | <code>SELECT lo_export(imagem.figura, '/tmp/imagem.jpeg') FROM imagem WHERE id_imagem = 1;</code> |

Fonte : Adaptado do Manual do PostgreSQL

4.2.2 BYTEA

O PostgreSQL possui um tipo de dado que armazena cadeias binárias, isto é, uma seqüência de *bytes* ou octetos de *bits*, denominado *Bytea*. Este tipo de dado permite o armazenamento de octetos zero e outros “não imprimíveis” (geralmente octetos fora da faixa 32 a 126).

O tipo de dados *Bytea* deve ser definido na criação da tabela do banco de dados como no exemplo a seguir:

```
create table imagem(
    id_imagem serial primary key,
    imagem bytea
);
```

A inserção de dados em campos *bytea* deve ser realizado em uma seqüência de escape. A implementação de uma seqüência de escape deve ser feita convertendo o octeto em um número octal de três dígitos precedido por duas contrabarras, apresentado na tabela 2.

Tabela 2 - Octetos com seqüência de escape para literais *bytea*

| Valor decimal do octeto | Descrição | Representação da entrada com escape | Exemplo | Representação da saída |
|-------------------------|---------------------------|-------------------------------------|------------------------|------------------------|
| 0 | octeto zero | '\\000' | SELECT '\\000'::bytea; | \000 |
| 39 | apóstrofo | '\'' ou '\\047' | SELECT '\''::bytea; | ' |
| 92 | contrabarra | '\\\\' ou '\\134' | SELECT '\\\\'::bytea; | \\ |
| 0 a 31 e 127 a 255 | octetos “não imprimíveis” | '\\xxx' (valor octal) | SELECT '\\001'::bytea; | \001 |

Fonte: Adaptado do Manual do PostgreSQL.

A razão pela qual se deve utilizar duas contra barras para a seqüência de escape é que, a primeira contra barra de cada par é interpretada como um caractere de escape pelo analisador de literais cadeia de caracteres, e portanto consumida,

deixando a segunda contra barra do par. A contra barra restante é reconhecida pela função de entrada de *bytea* como o início de um valor octal de três dígitos ou como escape de outra contra barra. Por exemplo, o literal cadeia de caracteres passado para o servidor como '\\001' se torna '\001', após passar pelo analisador de literais cadeias de caracteres. O '\001' é então enviado para a função de entrada de *bytea*, onde é convertido em um único octeto com valor decimal igual a 1. (Manual do PostgreSQL, p.90)

A utilização do tipo de dado *bytea* para o armazenamento de arquivos ainda não tem suas funções nativas em SQL para o escape de caracteres de arquivos binários, portanto fica a cargo da aplicação cliente realizar este tipo de seqüência de escape e inserir os dados.

5 FERRAMENTA DE INDEXAÇÃO E VISUALIZAÇÃO DE IMAGENS MÉDICAS

5.1 DESCRIÇÃO DO SISTEMA

Como parte dos objetivos deste trabalho foi desenvolvida uma ferramenta de indexação e recuperação de imagens médicas utilizando banco de dados PostgreSQL.

Se trata de uma ferramenta Web com o propósito inovador de realização de *upload* de imagens médicas de um computador cliente para o servidor de banco de dados PostgreSQL utilizando a plataforma *Web*, onde estas imagens serão armazenadas e indexadas por meio de suas respectivas patologias.

A ferramenta desenvolvida conta com um grau de segurança considerável, uma vez que as informações contidas são sigilosas e de interesse somente das pessoas diretamente envolvidas, como médicos e estudantes de medicina.

Possui um módulo visualizador de imagens, promovendo facilitadores no que diz respeito a recuperação desta por patologia.

Utiliza controle de usuários por nível de acesso, onde cada tipo de usuário possui restrições distintas para a utilização do sistema;

A implementação desta ferramenta baseou-se em tecnologias totalmente livres e seguras podendo ser utilizada para diversos fins, como por exemplo:

- Utilizar no mundo acadêmico para auxílio ao aprendizado da medicina, uma vez que permite armazenar juntamente com a imagem, seu diagnóstico e anamnese;
- Servir como um repositório de imagens médicas, reduzindo a quantidade de imagens armazenadas fisicamente, como também o custo operacional agregado, incrementando a velocidade de recuperação e visualização;
- Utilizar para diversas áreas da medicina que utilizam imagens para auxílio ao diagnóstico, uma vez que esta ferramenta foi desenvolvida, já com segmentação de módulos por área médica, onde cada médico poderá realizar a manipulação de suas imagens sem afetar o ambiente do outro usuário;

- Apresentar além das funcionalidades descritas acima, que esta ferramenta pode ser facilmente ampliada para fins alheios ao diagnóstico por imagens, como por exemplo, o suporte a dados de pacientes.

5.2 REQUISITOS DO SISTEMA

5.2.1 Requisitos Funcionais

De acordo com Xexéu (2007,p.44) "Os requisitos funcionais representam o que o sistema deve fazer ou alguma função esperada do sistema que agregue valor aos seus usuários."

Os requisitos funcionais desta ferramenta são:

- *Upload* de imagens médica junto com diagnóstico e observações;
- Cadastro de Áreas Médicas;
- Cadastro de Subáreas ou especialidades;
- Cadastro de Patologias;
- Cadastro de Usuários;
- Troca de senha de usuário;
- Controle de acesso de usuários (*Login*);
- Módulo de visualização de imagens médicas;
- Módulo de desativação/ativação de imagens para visualização;
- Módulo de Desativação de usuários;
- Gerador de relatórios.

5.2.2 Requisitos não Funcionais

Segundo Xexéu (2007,p.44) os requisitos não funcionais representam a forma com que os requisitos funcionais devem ser alcançados. O sistema deverá ser:

- Implementado utilizando tecnologias open source como PHP (PHP: Hipertext Preprocessor), jQuery e Apache Server;
- De fácil e rápida utilização por médicos e alunos de medicina, não necessitando de treinamento específico para sua manipulação;
- Seguro e estável, utilizando técnicas de autenticação de usuários (login), sessão, restrições de usuários e criptografia md5⁷.

5.2.3 Restrições do Sistema

“Restrições de projeto de sistemas são os limites impostos ao sistema para que funcione no seu ambiente operacional.” (XEXÉU, 2007, p.46)

A ferramenta apresenta, a princípio, as seguintes restrições:

- Da suporte a imagens com extensão JPEG (Join Pictures Expert Group), ou JPG, não sendo suportadas imagens com extensões diferentes desta como DICOM (Digital Imaging Communications in Medicine), PNG (Portable Network Graphics), etc.
- Não realiza processamento de imagem, podendo ser um dos módulos adicionais;
- Não possui suporte a dados de pacientes, podendo então ser implementado em módulos futuros.

5.3 TECNOLOGIAS UTILIZADAS

Para que o desenvolvimento se torne viável tanto economicamente, quanto de tempo de desenvolvimento, foram utilizadas tecnologias livres e com ampla documentação.

Logo mais são descritas às tecnologias utilizadas e uma breve descrição de sua história e utilização.

⁷ Mecanismo de *hash* (criptação) de dados

5.3.1 Servidor Apache

Um servidor *Web* foi desenvolvido no *National Center for Super Computing Applications* (NCSA) em 1995. Logo após a saída de seu principal desenvolvedor, várias pessoas e grupos passaram a adaptar este servidor *Web* às suas necessidades.

A partir disso foi criado o *Apache Group*, uma comunidade para manutenção do servidor da NCSA, que usou seu código fonte para criar um novo servidor denominado Apache.

Derivado originalmente do código do núcleo do servidor *Web* do NCSA e de uma grande quantidade de correções, o servidor Apache é agora o tema principal nas conversas da comunidade de servidores *Web*. Em apenas três anos, ele alcançou a posição de líder no mercado de servidores. (KABIR, 2002,p. 5)

Atualmente a comunidade *Apache Group* não tem fins lucrativos, operando por meio da *Web*, onde qualquer pessoa que tenha os conhecimentos necessários pode se integrar à equipe de desenvolvimento do servidor, porém quando é implementada alguma extensão, o novo código é submetido ao *Apache Group*, onde o mesmo realiza uma bateria de testes e controle de qualidade. Se aprovado, o código é integrado à distribuição principal do Apache. (KABIR, 2002, pág. 5)

Atualmente o servido Apache é o mais utilizado no mundo. Sua distribuição se encontra na versão 2.2, correspondente a versão utilizada para o desenvolvimento da ferramenta proposta neste trabalho.

5.3.2 PHP : Hipertext Preprocessor

PHP cujo nome é recursivo (*PHP: Hipertext Preprocessor*), é uma linguagem de desenvolvimento de *scripts* do lado servidor, geralmente incorporada no HTML (*HyperText Markup Language*).

Inicialmente criado por Rasmus Lerdorf, o PHP cujo nome inicial era “*Personal Home Page Tools*”, passou por várias versões até se parecer com a utilizada nos dias de hoje.

Atualmente, o PHP se encontra na versão 6.0, sendo que esta linguagem apresenta como algumas vantagens a extensibilidade de bibliotecas, uma vasta gama de suporte a conexão com banco de dados diversos, suporte a processamento de imagens, gerador de PDF(*Portable Document Format*), uma vasta documentação e muitos outros fatores que fazem com que o PHP seja uma das melhores linguagens para desenvolvimento de sistemas para a Web.

O PHP não é uma panacéia para todos os problemas de desenvolvimento Web, mas tem inúmeras vantagens. É construído por desenvolvedores Web para desenvolvedores Web e é suportado por uma grande e entusiástica comunidade. No centro do palco, ele se destaca por sua leveza, capacidade, confiabilidade e facilidade de uso. Oferece o melhor tipo e conectividade para todos os tipos de servidores de back - end e... será que mencionamos que o PHP é gratuito? Conhecer o PHP é amá-lo pelas muitas tarefas simples de desenvolvimento Web. (CONVERSE, PARK, 2003, p. 15)

5.3.3 jQuery

Criada por John Resig em 2006, jQuery é uma biblioteca *JavaScript* de código aberto, e de uso conforme a licença estabelecida pelo MIT (*Massachusetts Institute of Technology*) e pelo GPL (*GNU General Public License*).

Samy (2008) cita que a biblioteca JQuery é destinada a adicionar interatividade e dinamismo às paginas Web, isto proporciona ao desenvolvedor funcionalidades necessárias à criação de *scripts* que visem incrementar, de forma progressiva e não obstrutiva, a usabilidade, a acessibilidade e o *design*, fazendo com que a experiência do usuário se enriqueça.

O foco principal da biblioteca jQuery é a simplicidade. Por que submeter os desenvolvedores ao martírio de escrever longos e complexos códigos para criar simples efeitos? (SAMY,2010 apud John Resig)

Além de dinamismo em suas páginas jQuery propõe, confiabilidade, portabilidade entre *browsers*, velocidade no desenvolvimento, interação com o DOM

(*Document Object Model*), usabilidade com Ajax, compatibilidade com a utilização de seletores CSS (*Cascading Style Sheets*), extensões para desenvolvimento de *plugins* e muito mais.

A biblioteca jQuery está disponível para *download* no site <http://jquery.com>, sendo que esta é nada mais que um arquivo JavaScript, que deve ser chamado à página Web onde será utilizada. Como no exemplo abaixo.

```
<head>
  <script type="text/javascript" src="/caminho/jquery-1.3.2.js"></script>
</head>
```

Atualmente a biblioteca jQuery é freqüentemente utilizada por empresas como Google, Dell e várias outras no mundo da tecnologia, o que se permite dizer que esta é muito confiável, podendo ter grande utilização no mundo do desenvolvimento *Web* nos próximos anos.

6 IMPLEMENTAÇÃO DO SISTEMA

6.1 MODELAGEM DO BANCO DE DADOS

O desenvolvimento de um projeto de banco de dados é constituído de três fases:

- Modelagem Conceitual;
- Modelagem Lógica;
- Modelagem Física.

A seguir são descritas as três fases, assim como a implementação destas no projeto de desenvolvimento do banco de dados utilizado para a integração com a ferramenta de indexação de imagens médicas desenvolvida neste trabalho.

6.1.1 Modelagem Conceitual

A primeira fase de desenvolvimento de um banco de dados é a modelagem conceitual.

Um modelo conceitual é uma descrição do banco de dados de forma independente de implementação em um SGBD. O modelo conceitual registra que dados podem aparecer no banco de dados, mas não registra como estes dados estão armazenados a nível de SGBD. (HEUSER,2008, p.5)

A partir do conhecimento adquirido sobre o domínio da medicina pode-se utilizar uma abordagem denominada entidade-relacionamento (ER), gerando como resultado um diagrama conhecido como diagrama entidade-relacionamento (DER).

O diagrama resultante da abordagem entidade-relacionamento deste trabalho é apresentado na figura 8. Este modelo apresenta uma abordagem geral do esquema da base de dados e das funções a serem implementadas pela ferramenta de indexação de imagens médicas desenvolvida.

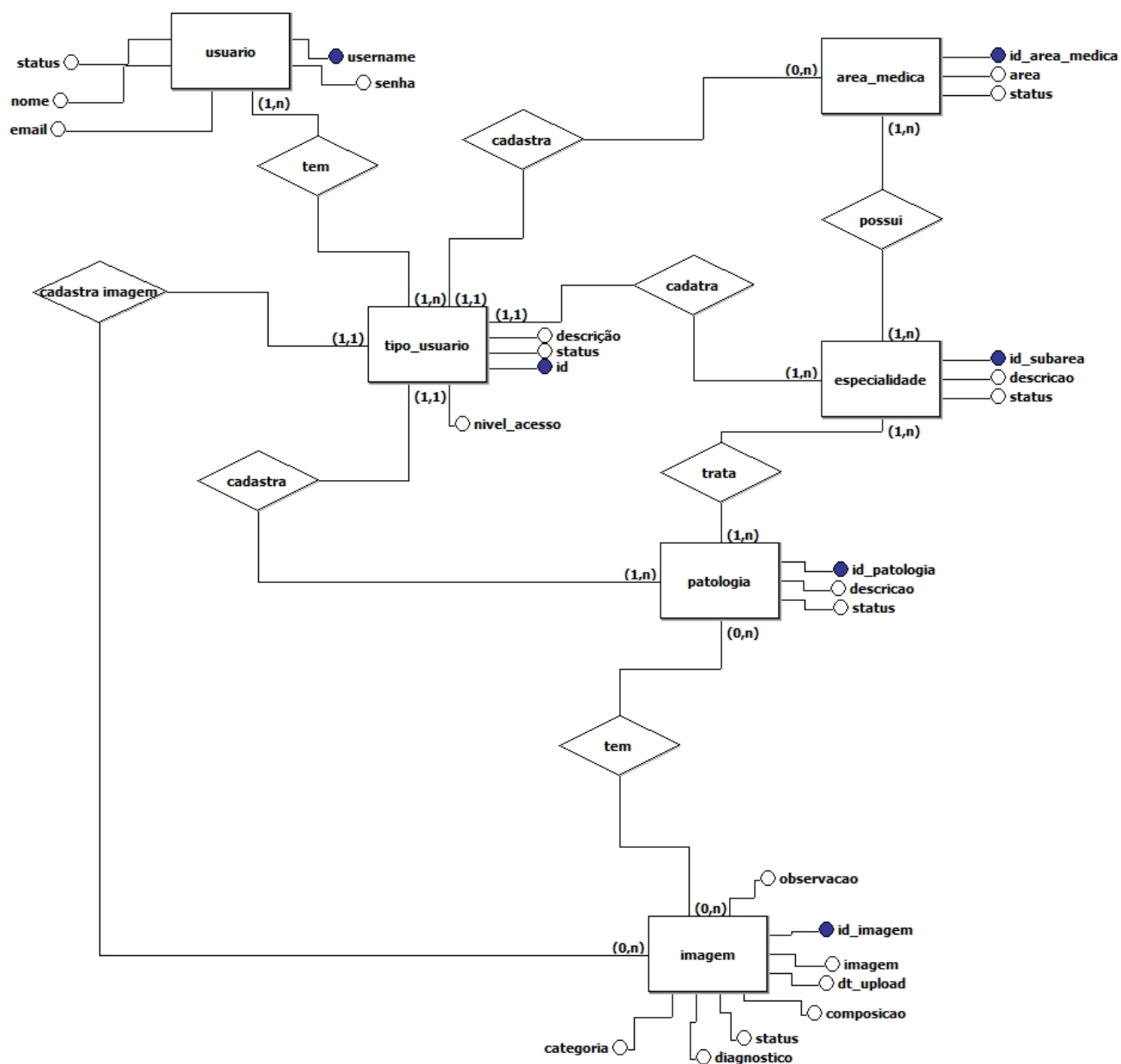


Figura 8: Modelagem conceitual do banco de dados

6.1.2 Modelagem Lógica

A segunda fase de desenvolvimento de um banco de dados é a modelagem lógica, que nada mais é do que a transição do DER para um modelo constituído de tabelas e relações entre tabelas. Sendo que uma tabela pode conter várias colunas definidas por tipos de dados específicos.

De acordo com Heuser (2008,p.6), "Um modelo lógico é uma descrição de um banco de dados no nível de abstração visto pelo usuário do SGBD. Assim, o modelo lógico é dependente do tipo particular de SGBD que está sendo usado."

O modelo lógico resultante da transição do DER gerado na figura 8, é apresentado na figura 9.

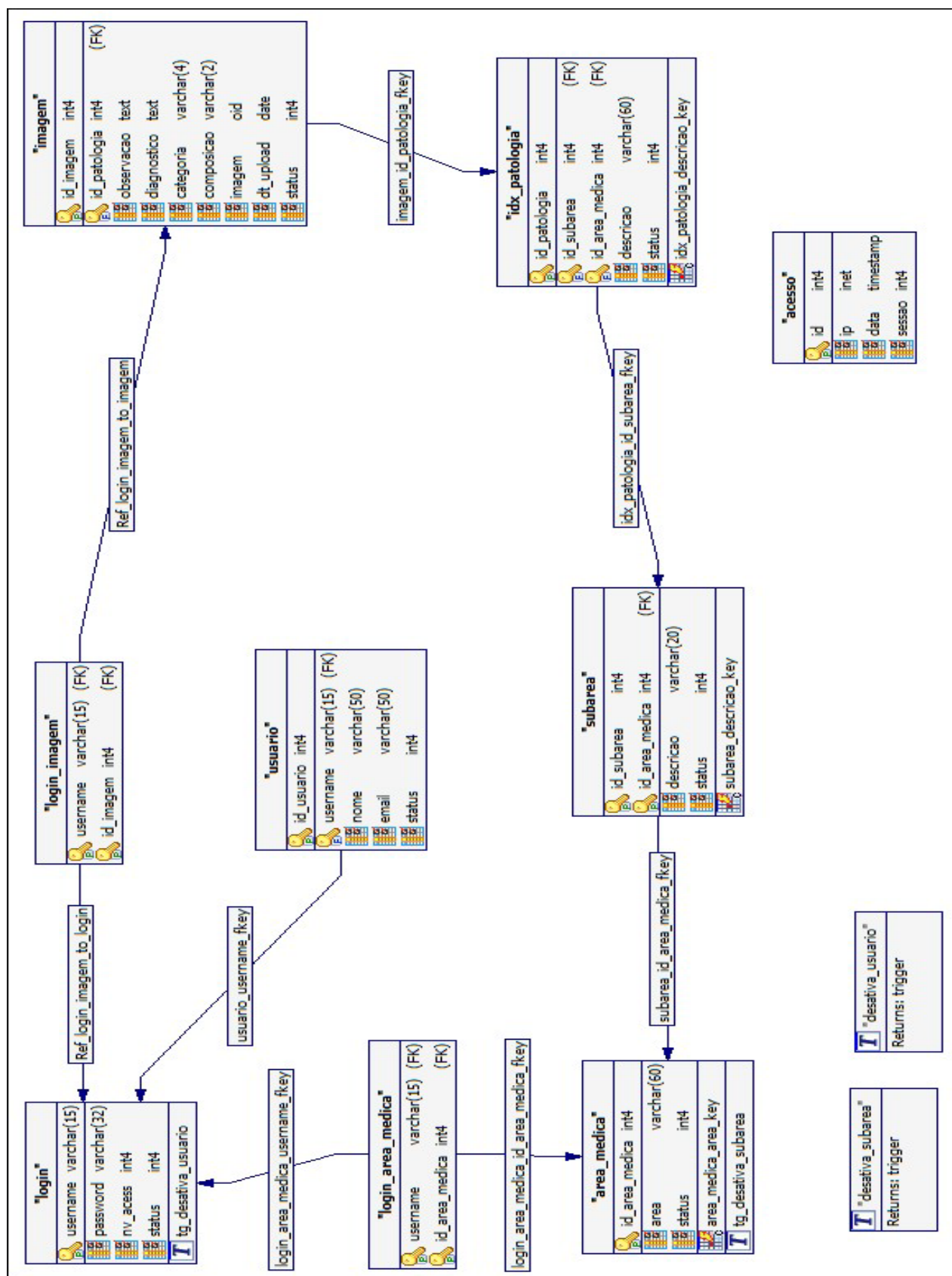


Figura 9: Modelagem Lógica do banco de dados

6.1.3 Modelagem Física

A terceira e última fase do desenvolvimento de um banco de dados é a modelagem física.

A modelagem física é constituída pela “tradução” do modelo lógico para uma linguagem que seja interpretada pelo SGBD, ou seja, linguagem SQL.

Nesta fase deve se preocupar com as particularidades de cada SGBD, suas linguagens específicas e seus tipos de dados.

A modelagem física do projeto de banco de dados deste trabalho se encontra na seção A do apêndice.

6.2 MÉTODO DE UPLOAD DE IMAGENS

Para a realização do *upload* de imagens médicas para o servidor de banco de dados foi desenvolvido um método denominado “realizaUpload”. Este é apresentado a seguir de forma simplificada:

```
public function realizaUpload($arquivo_temporario,$patologia,  
$observacao,$diagnostico,$categoria,$composicao);
```

O método “realizaUpload” recebe como parâmetros, os dados da imagem armazenada em um diretório temporário do servidor de banco de dados através do vetor \$_FILES do PHP, a patologia selecionada pelo usuário para indexação da imagem, observações e diagnóstico relacionados à imagem, categoria e composição da imagem (para imagens mamográficas).

É criada uma conexão com o SGBD PostgreSQL, utilizando o banco de dados denominado “medicina”.

```
include 'classes/conexao.class.php';  
$conexao = new ConexaoDB();  
$conexao->conexao("medicina");
```


Logo após é realizada a conversão de caracteres “/” por “\\”, do caminho de armazenamento temporário da imagem para inserção desta no SGBD PostgreSQL, utilizando a função do PHP *str_replace*.

```
$this->image = str_replace('\\', '/', $arquivo_temporario);
```

É iniciada uma transação e criada uma *query* para a inserção da imagem no banco de dados.

Logo após é executada a *query*.

```
pg_query("BEGIN");

$this->sql = "INSERT INTO imagem VALUES (default, $patologia,
'$observacao', '$diagnostico', '$categoria',
'$composicao', lo_import('$this->image'), date(now()), 1)";

$this->stat1 = pg_query($this->sql1);
```

É verificada a validade da execução da *query*, se esta foi executada com sucesso então é realizado um *COMMIT* na transação, caso contrário é realizado um *ROLLBACK*, e posteriormente a conexão é fechada.

```
if (!$this->stat1) {
    pg_query("ROLLBACK");

    } else {

        pg_query("COMMIT");
    }
    $conexao->close();
```

6.3 MÉTODO DE VISUALIZAÇÃO DE IMAGENS

Para recuperação de imagens inseridas no banco de dados foi desenvolvida uma função (ou método) denominada “geralmagem” que recebe como parâmetro o id da imagem a ser recuperada. Esta função é detalhada a seguir.

Após criada a conexão com o SGBD como descrito no método de *upload* de imagens, é criada uma consulta SQL para selecionar a imagem na base de dados a partir de seu id.

```
$sql = "SELECT imagem FROM imagem WHERE id_imagem = $id_imagem";
```

A consulta SQL criada é então executada pela função *pg_query* e seu resultado é armazenado na variável “\$result”.

Logo após é executada a função *pg_fetch_row*, onde seu resultado é retornado na forma de *array* (vetor) para a variável “\$data”.

```
$result = pg_query($sql);
$data = pg_fetch_row($result, 0);
```

É então verificado se a variável “\$data” contém valor.

```
if (!$data) {
    echo "Erro ao abrir imagem!!!";
} else {
```

Se a variável “\$data” contém valor então é iniciada uma transação. É aberta a *string* de *bytes* contida na variável “\$data” em modo de leitura fazendo com que a variável “\$ofp” se torne um ponteiro para a *string* de *bytes* de “\$data” através da função *pg_lo_open*.

É testado se a variável “\$ofp” possui valor, caso possua, então usa-se a função “*pg_lo_read*” que lê a *string* de *bytes* apontada por “\$ofp” e envia para a variável “\$im”.

Ao encerrar a transição da *string* de *bytes* para a variável “\$im”, é fechado o ponteiro “\$ofp” e então é encerrada a transação.

```
@pg_query("BEGIN");
$ofp = pg_lo_open($data[0], "r");
if (!$ofp) {
    echo "Erro ao abrir imagem!!!";
}
$im = pg_lo_read($ofp, 59999999);
pg_loclose($ofp);
```

```
pg_query("END");
```

Cria-se a variável “\$img” que recebe o valor convertido da *string* de *bytes* de \$im para uma imagem através da função *imagecreatefromstring*.

Utiliza-se um cabeçalho (*header*) com o padrão “image/jpeg”, e é transformada a imagem contida em “\$img” para o formato “jpeg” através da função *imagejpeg*. Logo após é “destruída” a variável “\$img” da memória do sistema.

```
$img = imagecreatefromstring($im);  
Header("Content-type: image/jpeg");  
imagejpeg($img);  
imagedestroy($img);  
}
```

Para finalizar, fecha-se a conexão com o SGBD utilizando a função *close*.

```
$conexao->close();
```

7 DEMONSTRAÇÃO DA FERRAMENTA

A ferramenta desenvolvida neste trabalho foi segmentada em módulos, dos quais são demonstrados neste capítulo.

A figura 10 apresenta a tela inicial do aplicativo, a qual permite identificar o usuário que efetuou o *login* e seus módulos de utilização, segmentado por níveis de usuário. Os níveis de usuários para o aplicativo desenvolvido corresponde a: 1 (aluno), 2 (médico) e 3 (administrador).

Nos próximos tópicos são apresentados, os módulos principais pertinentes ao nível de usuário médico.

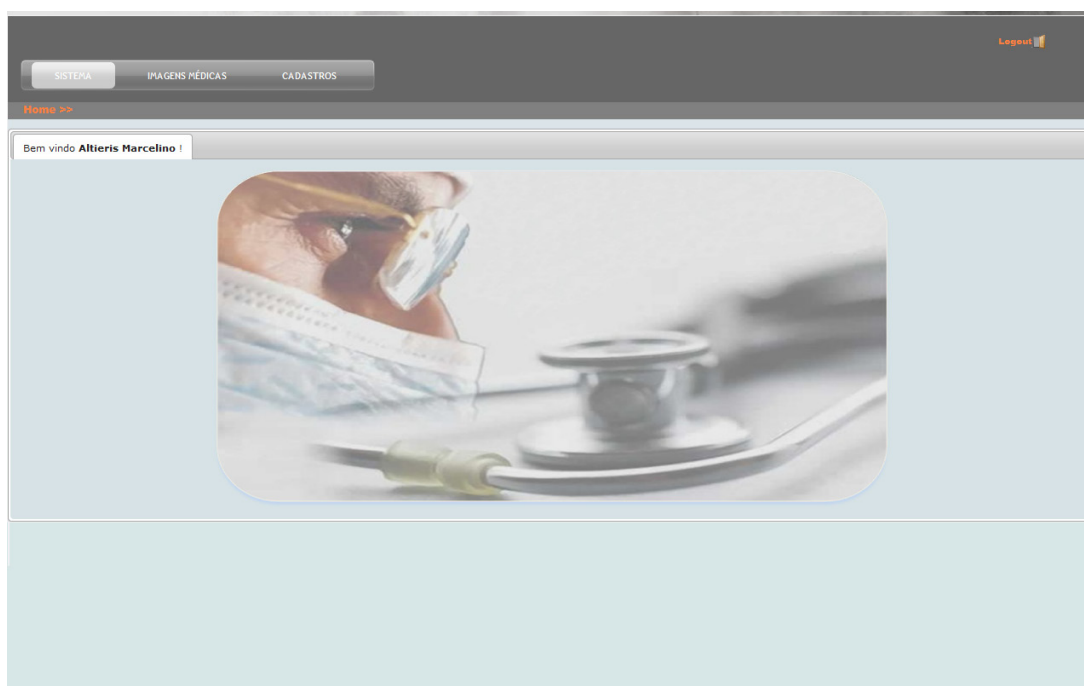


Figura 10: Tela principal do sistema

7.1 MÓDULO DE UPLOAD DE IMAGENS MÉDICAS

A tela para realização do *upload* de uma imagem médica é apresentada na figura 11, onde são selecionados a patologia, a imagem a ser carregada para o servidor de banco de dados, o diagnóstico e observações sobre a imagem.

Após preenchidos estes campos, o usuário deve clicar no botão enviar imagem para efetuar o envio da imagem para o servidor de banco de dados.

Figura 11: Tela de *upload* de imagens

A figura 12 apresenta a tela de “*upload* realizado”, esta tela é referente ao sucesso ou falha do carregamento da imagem para o servidor de banco de dados. Nesta tela pode-se alterar os dados da imagem em questão (inseridos na tela de *upload*).

Figura 12: *Upload* realizado

7.2 MÓDULO DE RECUPERAÇÃO DE IMAGENS MÉDICAS

A figura 13 apresenta a tela de busca de imagens, onde é selecionada a patologia para a busca das imagens e o número de imagens a serem visualizadas por página.

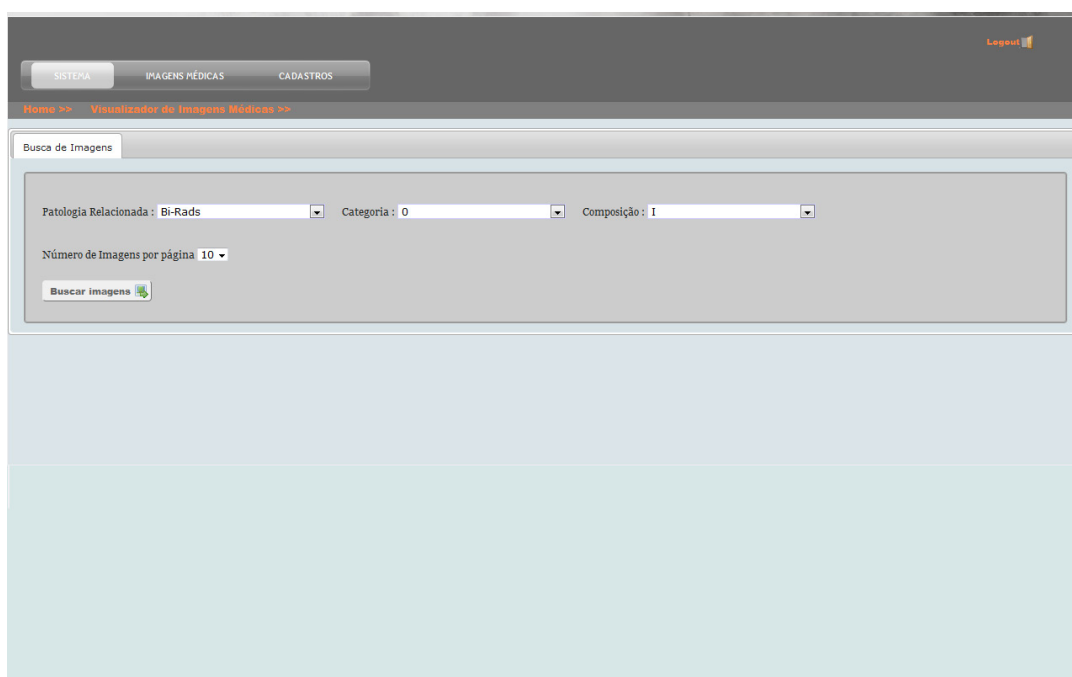


Figura 13: Tela de busca de imagens

O resultado da busca pode ser visualizado na figura 14, apresentando as miniaturas das imagens recuperadas a partir de uma consulta na base de dados, sobre a patologia mamária classificada através do Bi-Rads com a categoria 0 (zero) e composição 1 (um).

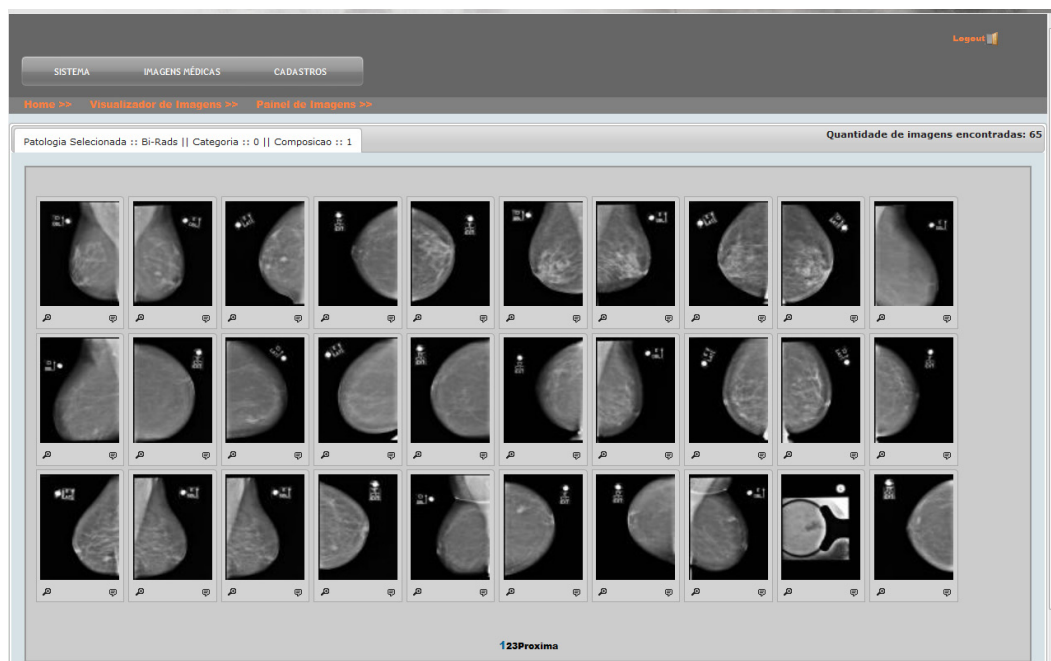


Figura 14: Painel de imagens recuperadas

A figura 15 apresenta uma mamografia ampliada, recuperada a partir de uma seleção na base de dados oriunda pela seleção da miniatura correspondente no painel de imagens apresentada na figura 14.

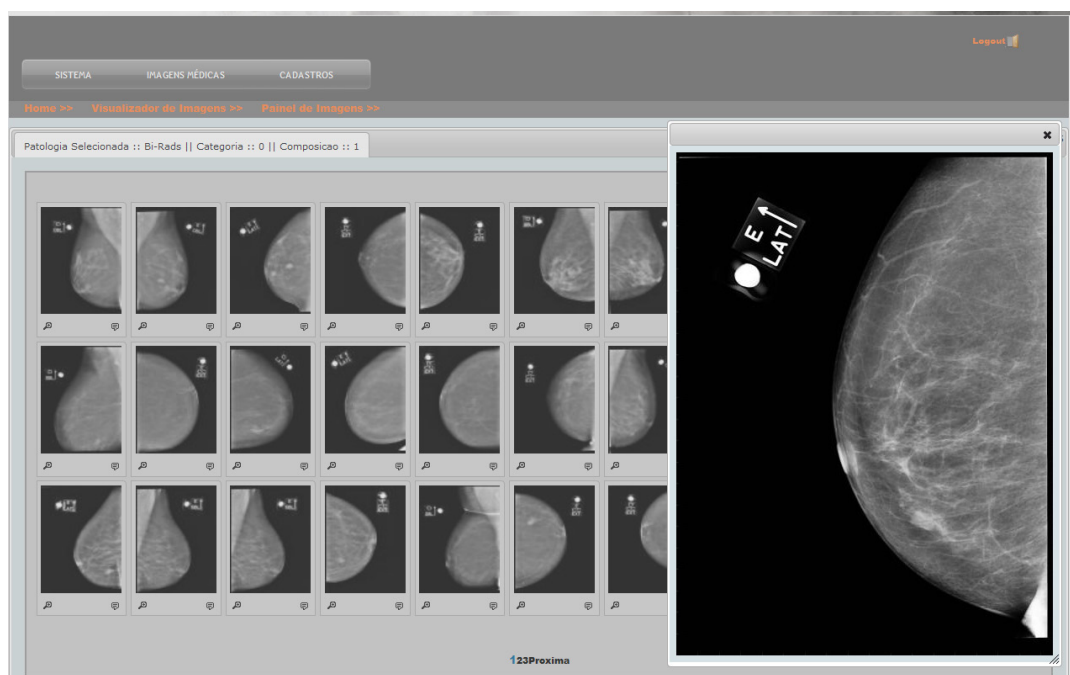


Figura 15: Mamografia ampliada

A figura 16 apresenta o diagnóstico e observações relativos a imagem selecionada no painel de imagens.

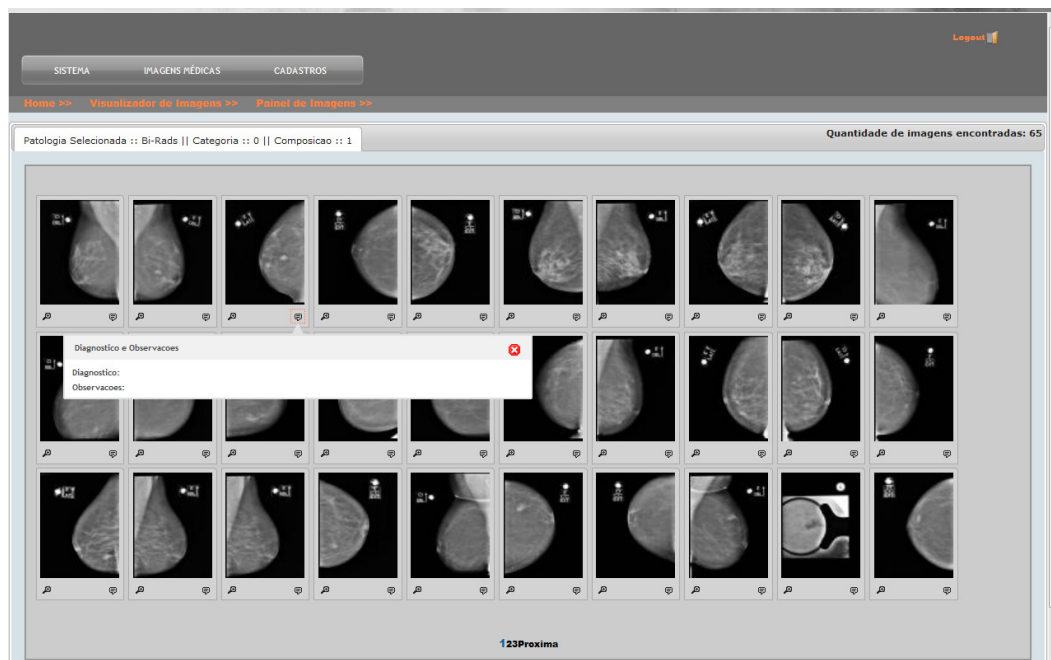


Figura 16: Diagnóstico e Observações relacionados a imagem selecionada

7.4 MÓDULO DE ATIVAÇÃO E DESATIVAÇÃO DE IMAGENS

Na figura 17 é possível visualizar a tela de recuperação de imagens para ativação ou desativação de imagens para visualização. Tal módulo promove facilitadores no que se refere à manipulação de imagens, devido à possibilidade de controlar as imagens as quais deverão ser recuperadas pelo sistema, fazendo a filtragem de imagens não relacionadas com a patologia selecionada, ou com diagnóstico errôneo sobre a imagem médica.

Através da seleção de uma determinada patologia, do número de imagens visualizadas por página, e das datas de *upload* das imagens, as imagens são recuperadas e visualizadas em uma tela semelhante à figura 18, bastando um clique na imagem de lixeira para a desativação da mesma.

Ativação/Desativação de Imagens

Ativar ou Desativar Imagens para Visualização: ☐ Ativar ☒ Desativar

Patologia Relacionada: Bi-Rads Categoria: 0 Composição: 1

Número de Imagens por página: 20

Selecionar período de upload para busca de imagens

Data Inicial: 01/10/2010

Data Final:

Selecionar Imagens

Figura 17: Tela de Busca de imagens para desativação/ativação

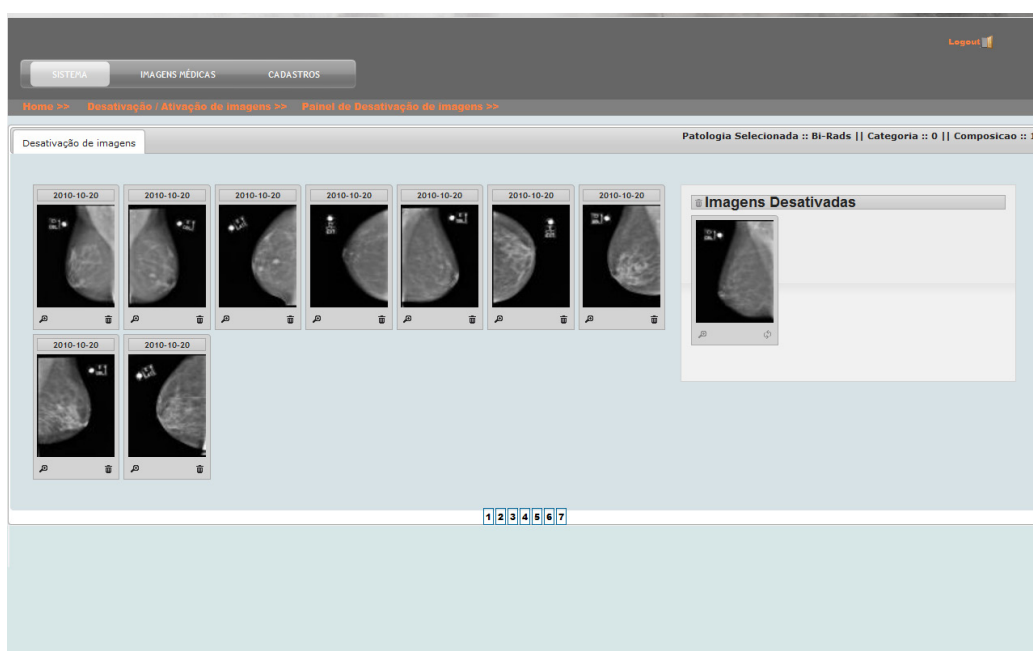


Figura 18: Desativação de imagens

8 CONSIDERAÇÕES FINAIS

Com o estudo constatou-se que a utilização de tecnologias *free*, como o PHP e o SGBD PostgreSQL, permitiu o desenvolvimento de uma ferramenta de indexação de imagens médicas em banco de dados não agregando nenhum custo no desenvolvimento.

Através do desenvolvimento da ferramenta proposta por este trabalho espera-se solucionar o problema da organização, armazenamento e disseminação de imagens médicas. Uma vez que a desorganização destas vem a causar problemas como a perda de exames, resultando em um novo exame do paciente e expondo-o desnecessariamente à radiação emitida pelos equipamentos de CAD.

Vale ressaltar que dentro do escopo deste trabalho foi obtido sucesso na utilização da ferramenta, porém para sua utilização em clínicas de diagnóstico por imagens deve-se realizar algumas alterações e ampliações em seus módulos, assim como os devidos testes em ambiente de produção.

Utilizando-se os conceitos de ontologia, taxonomia e metadados, um modelo de banco de dados genérico foi implementado para o armazenamento organizado de imagens médicas de diversas patologias, categorizadas por suas especialidades. Para este propósito, adotou-se o PostgreSQL como SGBD para a implementação do modelo de banco de dados, devido suas características e vantagens descritas no transcorrer do trabalho.

Espera-se, como trabalho futuro, realizar as ampliações e alterações necessárias para a utilização desta ferramenta em ambiente de produção, isto é, em clínicas de diagnóstico por imagens ou instituições acadêmicas voltadas para o ensino da medicina.

REFERÊNCIAS

BIBEAULT, Bear; KATZ, Yehuda. **jQuery in Action**. Greenwich : Manning,2008.

BICUDO, Carlos E. de M.;**Taxonomia**. Disponível em < <http://www.biotaneotropica.org.br/v4n1/pt/editorial>>.Acesso em 29 de jun. 2010.

CONVERSE, Tim; PARK, Joyce.**PHP a Bíblia**.Tradução da 2. ed. original de Edson Furmankiewicz. Rio de Janeiro. Campus,2003.

ELMASRI, Ramez; NAVATHE, Shamkant B. **Sistemas de Banco de Dados**. 4.ed. São Paulo:Pearson,2006.

GUIMARÃES,Francisco José Zamith; **Utilização de Ontologias no Domínio B2C**. Dissertação (Mestrado)-Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro, 2002.

HEUSER,Carlos Alberto.**Projeto de Banco de Dados**.2008.Disponível em < <http://groups-beta.google.com/group/digitalsource> >Acesso em 23 de out. 2010.

KABIR, J. Mohammed. **Apache Server 2**.Tradução:Vandenberg D. Souza. Rio de Janeiro. Campus,2002.

Manual do PostgreSQL Disponível em < <http://sourceforge.net/projects/pgdocptbr> >.Acesso em 29 de jun. 2010.

MATTHEW, Neil; STONES, Richard. **Beginning Databases with PostgreSQL From Novice to Professional, Second Edition**. New York: Apress, 2005.

MATTOS, Merissandra Cortês de; SIMÕES, Priscyla Waleska Targino de Azevedo;FARIAS,Renam Figueiredo. **Metodologia Methontology na construção de Ontologias**.

OLIVEIRA,Girlena Silva de. **Metodologia para construção de ontologias**. Monografia(Bacharelado em Ciência da Computação)-Universidade Federal da Bahia, Bahia, 2006.

ROUSSEY,Catherine.Guidelines to build ontologies : A bibliographic study. **COST Technical Committee "Transport and Urban Development"**, Nov.2005. Disponível em: <[http:// www.towntology.net/Documents/guidelines.pdf](http://www.towntology.net/Documents/guidelines.pdf)>. Acesso em 18 mai. 2010.

SILBERSCHATZ, Abraham; KORTH, Henry F. SUDARSHAN, S. **Sistema de Banco de Dados**. 5.ed. Rio de Janeiro:Elsevier,2006.

_____.**Sistema de Banco de Dados**. 3.ed.São Paulo:Pearson,1999.

SILVA,Ricardo Czelusniak da; **Benchmark em Banco de Dados Multimídia:Análise de desempenho em recuperação de objetos multimídia**.Dissertação (Pós-Graduação em Informática) - Universidade Federal do Paraná, Curitiba, 2006.

SILVA, Maurício Samy. JQuery: A Biblioteca do Programador JavaScript.São Paulo:Novatec, 2010.

XEXÉU,Geraldo.**Modelagem de Sistemas de Informação:Da análise de requisitos ao modelo de interface**, 2007. Disponível em < <http://wiki.xexeo.org> > Acesso em 23 de out.2010.

APÊNDICES

APENDICE A – MODELAGEM FÍSICA DO BANCO DE DADOS

```
-- CREATE TABLE login
CREATE TABLE login (
    username varchar(15) NOT NULL,
    password" varchar(32) NOT NULL,
    nv_acess integer NOT NULL,
    status integer NOT NULL,
    PRIMARY KEY("username")
);

-- CREATE TABLE usuario
CREATE TABLE usuario (
    id_usuario SERIAL NOT NULL,
    username varchar(15),
    nome varchar(50) NOT NULL,
    email varchar(50) NOT NULL,
    status integer NOT NULL,
    PRIMARY KEY("id_usuario"),
    FOREIGN KEY ("username") REFERENCES "login"("username")
);

-- CREATE TABLE area_medica
CREATE TABLE area_medica (
    id_area_medica SERIAL NOT NULL,
    area varchar(60) NOT NULL,
    status integer NOT NULL,
    PRIMARY KEY("id_area_medica")
);

-- CREATE TABLE subarea
CREATE TABLE subarea(
    id_subarea serial NOT NULL,
    id_area_medica integer NOT NULL,
    descricao character varying(20) NOT NULL,
    status integer NOT NULL,
    PRIMARY KEY (id_subarea, id_area_medica),
    FOREIGN KEY (id_area_medica) REFERENCES area_medica (id_area_medica)
);

-- CREATE TABLE idx_patologia
```

```

CREATE TABLE idx_patologia (
    id_patologia SERIAL NOT NULL,
    id_subarea integer NOT NULL,
    id_area_medica integer NOT NULL,
    descricao varchar(60) NOT NULL,
    status integer NOT NULL,
    PRIMARY KEY("id_patologia"),
    FOREIGN KEY ("id_subarea", "id_area_medica")
    REFERENCES "subarea"("id_subarea", "id_area_medica")
);

-- CREATE TABLE imagem
CREATE TABLE imagem (
    id_imagem SERIAL NOT NULL,
    id_patologia integer NOT NULL,
    observacao text,
    diagnostico text,
    categoria varchar(4),
    composicao varchar(2),
    imagem oid NOT NULL,
    dt_upload date NOT NULL,
    status integer NOT NULL,
    PRIMARY KEY("id_imagem"),
    FOREIGN KEY ("id_patologia")
    REFERENCES "idx_patologia"("id_patologia")
);

-- CREATE TABLE login_area_medica
CREATE TABLE login_area_medica (
    username varchar(15) NOT NULL,
    id_area_medica integer NOT NULL,
    PRIMARY KEY("username", "id_area_medica"),
    FOREIGN KEY ("username") REFERENCES "login"("username"),
    FOREIGN KEY ("id_area_medica") REFERENCES
    "area_medica"("id_area_medica")
);

-- CREATE TABLE login_imagem

```

```

CREATE TABLE login_imagem (
    username varchar(15) NOT NULL,
    id_imagem integer NOT NULL,
    PRIMARY KEY("username","id_imagem"),
    FOREIGN KEY ("username")REFERENCES "login"("username"),
    FOREIGN KEY ("id_imagem")REFERENCES "imagem"("id_imagem")
);

-- CREATE TABLE acesso
CREATE TABLE acesso (
    id_acesso integer,
    ip inet,
    data timestamp,
    sessao integer
);

-- CREATE STORED PROC : desativa_usuario
CREATE OR REPLACE FUNCTION desativa_usuario()
RETURNS trigger AS $$
begin
    update usuario set status = new.status where username = old.username;
    return null;
end;
$$
LANGUAGE plpgsql

CREATE TRIGGER tg_desativa_usuario AFTER UPDATE ON login FOR EACH ROW
EXECUTE PROCEDURE "desativa_usuario"();

```


ANEXOS

ANEXO A – FUNÇÕES DE MANIPULAÇÃO DE LARGE OBJECTS NO POSTGRESQL

A seguir será mostradas funções para manipulação de objetos grandes no SGBD PostgreSQL, de acordo com o Manual do PostgreSQL.

Para fins didáticos e práticos serão utilizadas a seguintes convenção para a variável de conexão com o banco de dados:

- **conn** : O argumento se refere a conexão utilizada para o estabelecimento da comunicação com o banco de dados para a criação do objeto grande.

1.1 Criação de Objeto Grande

Para criar um objeto grande novo no banco de dados deve-se utilizar a função :

```
Oid lo_creat(PGconn *conn, int modo);
```

Onde :

O argumento **modo** e composto por uma mascara de bits que refere aos diversos atributos do novo objeto. As permissões de acesso (leitura, escrita ou ambos) são controladas pelo OU lógico (|) dos bits de *INV_READ* e *INV_WRITE*.

O valor a ser retornado é o OID(*Object Identification Data*) atribuído ao objeto grande criado, ou *InvalidOid* (zero) se ocorrer algum erro na criação do objeto grande.

Exemplo:

```
inv_oid = lo_creat(conn, INV_READ|INV_WRITE);
```

1.2 Importação de Objeto Grande

Para realizar a importação de um arquivo do sistema operacional como um objeto grande deve ser chamada a função :

```
Oid lo_import(PGconn *conn, const char *nome_do_arquivo);
```

Onde:

O argumento **nome_do_arquivo** se refere ao nome do arquivo do sistema operacional que será importado para o novo objeto grande. É importante ressaltar que o arquivo deve residir no sistema de arquivos do cliente, uma vez que o arquivo é lido pela biblioteca de interface cliente e não pelo servidor.

O valor a ser retornado é o OID do novo objeto grande, ou *InvalidOid* (zero) se ocorrer algum erro na importação do arquivo.

Exemplo:

```
inv_oid = lo_import(conn,'caminho/do/arquivo/imagem.jpeg');
```

1.3 Exportação de Objeto Grande

Para realizar a exportação de um objeto grande para um arquivo do sistema operacional deve utilizar a função abaixo:

```
int lo_export(PGconn *conn, Oid lobjId, const char *nome_do_arquivo);
```

Onde:

O argumento **lobjId** é utilizado para especificar o OID do objeto grande a ser exportado.

O argumento **nome_do_arquivo** é utilizado para especificar o nome e caminho, onde o objeto grande será exportado.

Esta função tem como retorno 1 se bem sucedida, ou -1 um se ocorrer alguma falha de exportação.

Exemplo:

```
inv_oid = lo_export(conn,12345,'caminho/do/arquivo/imagem.jpeg');
```

1.4 Abertura de Objeto Grande existente

Para realizar a abertura de um objeto grande deve-se utilizar a seguinte função:

```
int lo_open(PGconn *conn, Oid lobjId, int modo);
```

Onde:

O argumento **lobjld** é utilizado para especificar o OID do objeto grande a ser aberto.

O argumento **modo** especifica o modo de abertura do objeto, INV_READ (leitura), INV_WRITE(escrita) , ou ambos.

Esta função tem como retorno um descritor do objeto grande (não negativo) para utilização posterior em funções como *lo_read*, *lo_write*, *lo_seek*, *lo_tell* e *lo_close*. Este descritor tem duração apenas para a transação corrente.

Em caso de erro na abertura do objeto grande, esta função retorna -1.

Exemplo:

```
fd = lo_open(conn,12345,INV_READ|INV_WRITE);
```

1.5 Escrita de dados em Objeto Grande

Para escrever em um objeto grande deve-se utilizar a função:

```
int lo_write(PGconn *conn, int fd, const char *buf, size_t len);
```

Onde:

O argumento **fd** se refere ao descritor do objeto grande retornado pela função *lo_open*.

O argumento **len** especifica o número de bytes a serem escritos no descritor do objeto grandes **fd** a partir do total de bytes denominado **buf**.

Esta função tem como retorno o número de bytes escritos em **fd**. Em caso de erro na escrita do objeto grande, esta função retorna um valor negativo.

1.6 Leitura de dados de Objeto Grande

Para realizar a leitura de um objeto grande deve-se utilizar a função:

```
int lo_read(PGconn *conn, int fd, char *buf, size_t len);
```

Onde :

O argumento **fd** se refere ao descritor do objeto grande retornado pela função *lo_open*.

O argumento **len** especifica o numero de bytes a serem lidos no descritor de objeto grande **fd** e coloca-os em **buf**.

Esta função tem como retorno o número de bytes lidos em **fd**. Em caso de erro na escrita do objeto grande, esta função retorna um valor negativo.

1.7 Procura em Objeto Grande

Para mover o ponteiro de posição corrente de leitura ou escrita ligada ao descritor do objeto grande deve-se utilizar a seguinte função:

```
int lo_lseek(PGconn *conn, int fd, int deslocamento, int donde);
```

Onde:

O argumento **fd** se refere ao descritor do objeto grande retornado pela função *lo_open*.

O argumento **deslocamento** é referente a nova posição especificada.

Os argumento **donde** é referente a posição de partida para a leitura do objeto grande e pode ter os seguintes valores:

SEEK_SET: procura a partir do inicio do objeto.

SEEK_CUR: procura a partir da posição corrente;

SEEK_END : procura a partir do fim do objeto.

A função tem como retorno o novo ponteiro de posição. Em caso de erro a função retorna -1.

1.8 Obtenção da posição de procura no Objeto Grande

A posição corrente de leitura ou escrita do descritor de objeto pode ser obtida a partir da função:

```
int lo_tell(PGconn *conn, int fd);
```

Onde :

O argumento **fd** se refere ao descritor do objeto grande retornado pela função *lo_open*.

Em caso de falha, essa função retorna o valor -1.

1.9 Fechamento do descritor do Objeto Grande

Para fechar um descritor de objeto grande deve-se utilizar a função:

```
int lo_close(PGconn *conn, int fd);
```

Onde:

O argumento **fd** se refere ao descritor do objeto grande retornado pela função *lo_open*.

Esta função tem como retorno o valor 0 (zero) se bem sucedida, caso contrário retorna um valor negativo.

1.10 Remoção de Objeto Grande

Para deletar um objeto grande do banco de dados deve-se utilizar a função:

```
int lo_unlink(PGconn *conn, Oid lobjId);
```

Onde:

O argumento **lobjID** se refere ao OID do objeto grande a ser removido.

Se bem sucedida, esta função retorna o valor 1, caso contrario o retorno tem valor -1.