UNIVERSIDADE FEDERAL FLUMINENSE
INSTITUTO DE COMPUTAÇÃO
DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

BRUNO ERTHAL DE ABREU
TONI TIAGO DA SILVA PACHECO

# CATALOGO DIGITAL DE IMAGENS

NITERÓI
2010

BRUNO ERTHAL DE ABREU
TONI TIAGO DA SILVA PACHECO

# CATALOGO DIGITAL DE IMAGENS

Trabalho de Conclusão apresentado ao Curso de Graduação em Ciência da Computação da Universidade Federal Fluminense, como requisito parcial para obtenção do grau de Bacharel em Ciência da Computação.

ORIENTADOR: PROF. LUIZ VALTER BRAND GOMES

NITERÓI
2010

BRUNO ERTHAL DE ABREU
TONI TIAGO DA SILVA PACHECO

# CATALOGO DIGITAL DE IMAGENS

Trabalho de Conclusão apresentado ao Curso de Graduação em Ciência da Computação da Universidade Federal Fluminense, como requisito parcial para obtenção do grau de Bacharel em Ciência da Computação.

BANCA EXAMINADORA

---

Prof. Luiz Valter Brand Gomes - Orientador
Universidade Federal Fluminense

---

Prof. Leonardo Cruz da Costa
Universidade Federal Fluminense

---

Profa. Rosangela Lopes Lima
Universidade Federal Fluminense

NITERÓI
2010

## RESUMO

O presente estudo consiste no desenvolvimento de catálogo digital utilizando a linguagem JAVA, que permita o cadastro de imagens com suas respectivas características e classificações para posterior visualização facilitando, desta forma, a classificação de objetos a partir de comparações com os dados cadastrados. O problema central relacionado a essa proposta é a dificuldade encontrada por profissionais de diversas áreas de conhecimento em identificar e classificar objetos de uma classe adequadamente, de modo a gerar dados relevantes à área de estudos. O sistema desenvolvido possibilitará a identificação e classificação de objetos de mesma natureza em um intervalo de tempo muito menor, proporcionando a modernização das atividades de identificação de tipos ou espécies, alem da padronização de análises.

Usabilidade, eficiência e reusabilidade nortearam o desenvolvimento desta aplicação.



**Palavras-chave:** Catálogo digital de imagens; linguagem JAVA; análise de imagens; sistema de decisão.

## ABSTRACT

The present study consists in the development of a digital catalog using the JAVA language, which allows the cadastre of images with their features and classifications for posterior visualization, facilitating, this way, the classification of objects from the comparison with the registered data. The central problem related to this proposal is the difficulty encountered by professionals of diverse areas of knowledge in accordingly identifying and classifying objects, generating relevant data to the area of study. The developed system will make possible the identification and classification of objects of the same nature in a smaller time interval, providing the salutary modernization of the activities of identification of types and species, and the standardization of analysis.

Usability, efficiency and reuse guided the development of this application.

# LISTA DE FIGURAS

## LISTA DE ABREVIATURAS E SIGLAS

| | |
|---|---|
| MFC | Microsoft Foundation Classes |
| MDI | Multiple Document Interface |
| JDK | Java Development Kit |
| IDE | Integrated Development Environment |

# SUMÁRIO

# 1. INTRODUÇÃO

O projeto Catálogo de Imagens tem a finalidade de organizar as imagens e as informações pertinentes ao objeto fotografado, pois um enorme conjunto de imagens sem as informações àquelas inerentes se torna inócuo.

A organização destas imagens será feita através de uma classificação que permite as imagens serem agrupadas de acordo com determinadas caracteríscas daquele objeto. Para que as imagens possam ser classificadas da maneira mais eficiente, o Catálogo de Imagens implementa ferramentas auxiliares para classificação através de perguntas ou por comparação com imagens já cadastradas no sistema, semelhantes à imagem que está sendo trabalhada. Uma vez classificada e incluída no sistema, a imagem irá ajudar a classificar outras imagens futuras.

Para elucidar os conceitos utilizados, o segundo capítulo, tratará sobre o que é como e onde é feita a classificação de um objeto e seu uso para a elaboração de um catálogo.

O terceiro capítulo apresentará a arquitetura proposta e suas principais visões.

No quarto capítulo são enumeradas as principais ações do usuário na forma de casos de uso.

O quinto capítulo abordará as soluções adotadas para os principais problemas de implementação. Um resumo das principais classes e interfaces também será apresentado.

O sexto capítulo tratará sobre o armazenamento e os tipos das informações, sobre o SGBD utilizado e a estrutura implementada.

O sétimo capítulo exemplificará nosso projeto através de uma aplicação para catalogar espécies da família das orquídeas (*orchidaceae*).

E por fim, o oitavo e o último capítulo, apresentarão sugestões de desenvolvimento de trabalhos futuros a partir de nosso projeto.

## 2. CLASSIFICAÇÃO DE OBJETOS

Mesmo sem perceber, o ser humano classifica tudo e todos diariamente. Classificam-se roupas em roupa de ficar em casa, de sair ou de trabalhar; classifica-se relacionamentos sociais em desconhecidos, colegas ou amigos. Classificar é importante para que se possa aplicar com melhor eficiência os recursos e informações a que se tem acesso.

Classificação consiste em separar elementos agrupando-os por características diversas de forma que os elementos desses grupos quantitativamente menores resultantes sejam mais semelhantes entre si. Esta metodologia é adotada em várias áreas de estudo como na geologia, botânica, zoologia dentre outras.

Ao classificar um elemento podem-se adquirir novas informações pertinentes ao grupo ao qual ele pertence. Por exemplo, em zoologia, a classificação de um animal como mamífero, permite deduzir o tempo e tipo de gestação, formação do esqueleto, forma de reprodução etc.

Trabalhar com grupos distintos possibilita descobertas que não seriam possíveis de outras formas, tanto da relação de um elemento com outros elementos do mesmo grupo como da relação intergrupos. Assim foram os estudos precursores de genética, os quais se utilizavam ervilhas separadas em três grupos analisando os cruzamentos dentro e fora de cada grupo.

Um exemplo é perfuração de um poço de petróleo que tem um custo elevadíssimo, razão pela quais informações litológicas são de suma importância para indústria petrolífera, determinando inclusive a viabilidade ou não do empreendimento. A classificação das rochas sugere a capacidade de produção do poço e também ditam a maneira como o poço será perfurado, quais profundidades são críticas durante a perfuração e qual o tipo de estrutura será usada para a manutenção daquele poço.

A Classificação é feita da seguinte forma:

Primeiro avalia-se as características restritivas, aquelas que por si só são suficientes para dizer se um elemento pertence aquele grupo ou não; por exemplo, se um animal amamenta é uma característica restritiva para o grupo dos mamíferos.

Se houver mais de uma classificação possível para aquele conjunto de características restritivas avaliam-se então outras características. Cada uma das outras características têm um peso com relação a uma classificação e estes pesos deverão ser

somados para cada classificação e ao final, a que tiver o maior somatório de pesos, será a classificação sugerida.

A partir da classificação de diversos objetos pode-se reuni-los organizadamente criando um catálogo. Este servirá para realizar consultas, servindo principalmente como ferramenta auxiliar no processo de classificação de um novo objeto dentro da categoria analisada. Quanto mais objetos existirem no catálogo maior a possibilidade de encontrar um objeto muito parecido ou mesmo igual ao novo objeto.

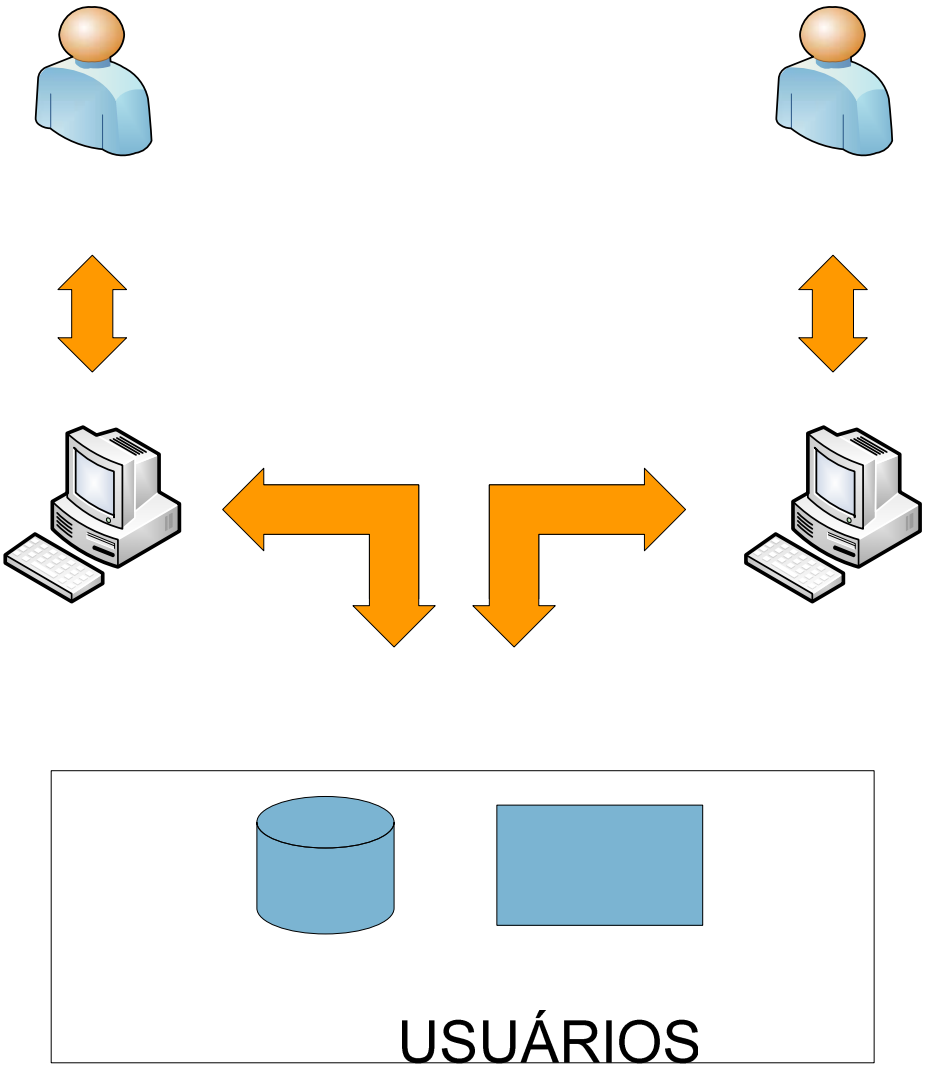
Áreas do conhecimento onde a imagem do objeto de estudo possua importância são beneficiadas com o uso de um catálogo de imagens. Através deste é possível ter acesso de uma só vez a diversos objetos dispostos de forma a melhorar e facilitar a visualização das imagens e respectivas características. Sendo este catálogo digital pode-se aplicar outros tipos de ferramentas como filtros para auxiliar o processo diminuindo o tempo de execução.

Um catálogo digital pode ter uma única base de dados acessada por vários usuários, concentrando as informações em único lugar tornando o catálogo cada vez mais completo.

## 3. ARQUITETURA PROPOSTA

A arquitetura é apresentada através de um conjunto de visões que juntas visam cobrir os principais aspectos técnicos relativos ao desenvolvimento do *software* em questão. O objetivo é capturar e formalizar as principais decisões tomadas com relação à arquitetura do *software*.

**Figura 1 - Arquitetura Proposta**




A arquitetura adotada, como na Figura 1, tem base na proposta de implementar um sistema que, através de uma interface de fácil utilização, permita aos usuários conseguirem executar descrições diagnósticas com substancial redução de tempo na análise. Para tanto, o sistema precisa acessar a base de imagens, que será criada com

dados selecionados pelo usuário especialista.

O usuário deverá capturar a imagem do objeto a ser analisado para que o sistema, a partir da imagem e da base de dados, possa executar uma série de tratamentos na imagem, com o objetivo de encontrar similaridades entre as características (módulo de análise automática).

O sistema possibilitará que sejam feitas consultas sobre características, padrões de textura e estrutura do objeto.

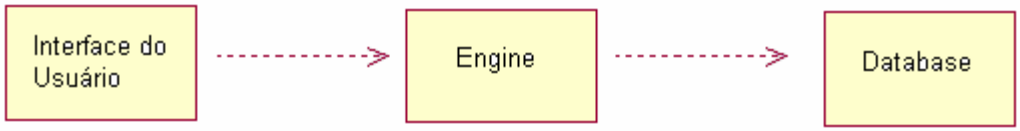
O sistema foi desenvolvido para ser utilizado em versão a ser instalada em qualquer máquina que esteja conectada a rede ou tenha cópia da base de dados e base de imagem localmente. O banco de dados e imagem deverá ser único e nele estarão as informações dos objetos padrões para comparação que represente claramente um tipo ou espécie.

## 3.1. Visão Geral

O *software* está basicamente em três camadas, como mostrado na Figura 2:

- Interface do usuário: Constrói a interface do sistema com acesso através de um aplicativo MDI e janelas de interação com o usuário. Composição básica pela biblioteca *Swing* e *Awt*.
- *Engine*: Controla o processamento das ações requeridas pelo usuário, aplica segurança através de identificação do usuário e aciona operações no banco de dados, retornando o conteúdo que será enviado ao navegador utilizado pelo cliente.
- *Database*: Define o banco de dados a ser utilizado para persistir as informações do sistema. Pode utilizar componentes de intermediação com a camada de lógica de negócio.

**Figura 2 – Diagrama Visão Geral**




## 3.2. Visão de Processos

Inicialmente cada máquina cliente terá a aplicação localmente instalada; assim cada máquina gerará apenas um processo. O programa fará requisições ao Banco de Dados, que estará no servidor de banco de dados, permitindo assim que mais de um

usuário possa utilizar o sistema simultaneamente.

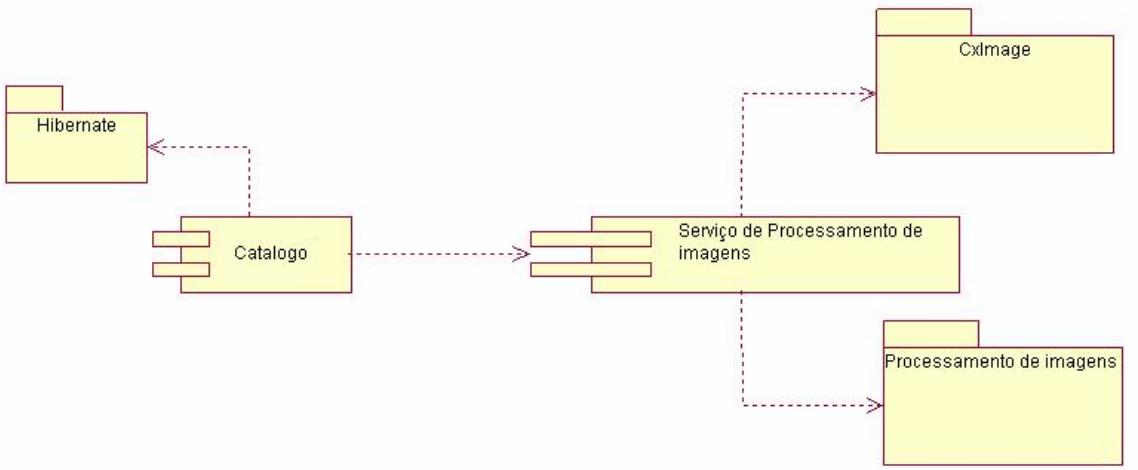
### 3.3. Visão de Implementação

O catálogo será desenvolvido como aplicativo para execução local. O sistema gerenciado de banco de dados a ser utilizado é o *PostgreSQL*. Para o seu desenvolvimento foi utilizada a linguagem Java, uma linguagem moderna que possui diversos *frameworks* que possibilitam o desenvolvimento rápido de aplicações robustas e poderosas. Dentre os frameworks da linguagem Java foram utilizados o Swing para construção de interfaces e o Hibernate para conexão com o banco de dados. Foi usada a ferramenta de desenvolvimento *Netbeans*. A Figura 3 mostra uma visão geral da implementação.

O sistema utilizará o *Java web start* para sua instalação. Os arquivos *.jar e *.exe do sistema serão copiados para o diretório raiz da instalação (<raiz>) e as bibliotecas *.jar serão copiadas para a pasta <raiz>/lib.

O módulo de análise de imagem funciona na forma de serviço a espera de requisições, portanto será instalada no servidor que tratará os pedidos de análise retornando para a aplicação a resposta da análise. O serviço deve ser instalado de forma que suas bibliotecas fiquem no mesmo diretório do arquivo executável (instalação padrão). Tendo então, diretório de instalação (<raiz>) o arquivo *.exe da aplicação e as bibliotecas *.dll.

**Figura 3 - Diagrama Visão de Implementação**



### 3.4. Visão de Implantação

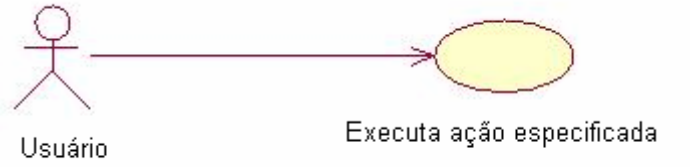
Os requisitos mínimos para o funcionamento do Sistema:

- Processador recomendado: Pentium 4 (ou superior),
- Memória recomendada: 1GB (ou maior quantidade);
- Espaço em disco: Variante dependendo da qualidade/quantidade das imagens e fatores relativos à base de dados.
- Banco de Dados *PostgreSQL* 8.3(ou superior) instalado no Servidor;

## 4. PRINCIPAIS CASOS DE USO

Todos os casos de usos que serão aqui mostrados seguem a mesma forma de interação. O usuário acessa diretamente uma funcionalidade do sistema como mostrado no diagrama da Figura 4.

**Figura 4 - Modelo de Casos de Uso**




- **Capturar Imagem em análise**

A imagem que está sendo analisada poderá ser capturada pelo usuário com objetivo de ser inserida na base de dados do sistema por um responsável. A imagem poderá ser utilizada também para uma posterior análise, com o objetivo de detalhar a análise inicial.

- **Classificar objeto**

O usuário classifica o objeto em análise.

- **Comparar Imagens**

O usuário compara a imagem em análise com as imagens da base de dados do sistema.

- **Consultar Índice**

O usuário consulta a base de dados do sistema utilizando índices textuais e remissivos, para recuperar exemplos de objetos da base de dados.

- **Inserir dados de análise**

O usuário, com permissão, insere dados de análise no sistema com o objetivo de enriquecer o acervo inicial do catálogo.

- **Responder perguntas**

O usuário responde a uma série de perguntas pertinentes as características da imagem a ser analisada, com o objetivo de filtrar a análise e assim aperfeiçoar o

processo.

- **Visualizar desenhos esquemáticos**

O usuário visualiza esquemas que irão representar os valores das propriedades utilizadas para classificar os objetos.

- **Visualizar Fotos**

O usuário visualiza fotos de padrões presentes na base de dados do sistema e suas respectivas descrições.

## 5. IMPLEMENTAÇÃO

Como citado anteriormente, a interface visa uma abordagem de fácil interação e a disponibilidade de filtros que possam ajudar o usuário a chegar a uma classificação confiável. Com isso é proporcionado não apenas uma visualização agradável, mas é disponibilizado também ferramentas para guiar o usuário até uma classificação ou lista de possíveis classificações.

O módulo de interface do sistema foi implementado usando o Java JDK 1.6 (*Swing* e *Awt*) e a *IDE Netbeans* 6.1.

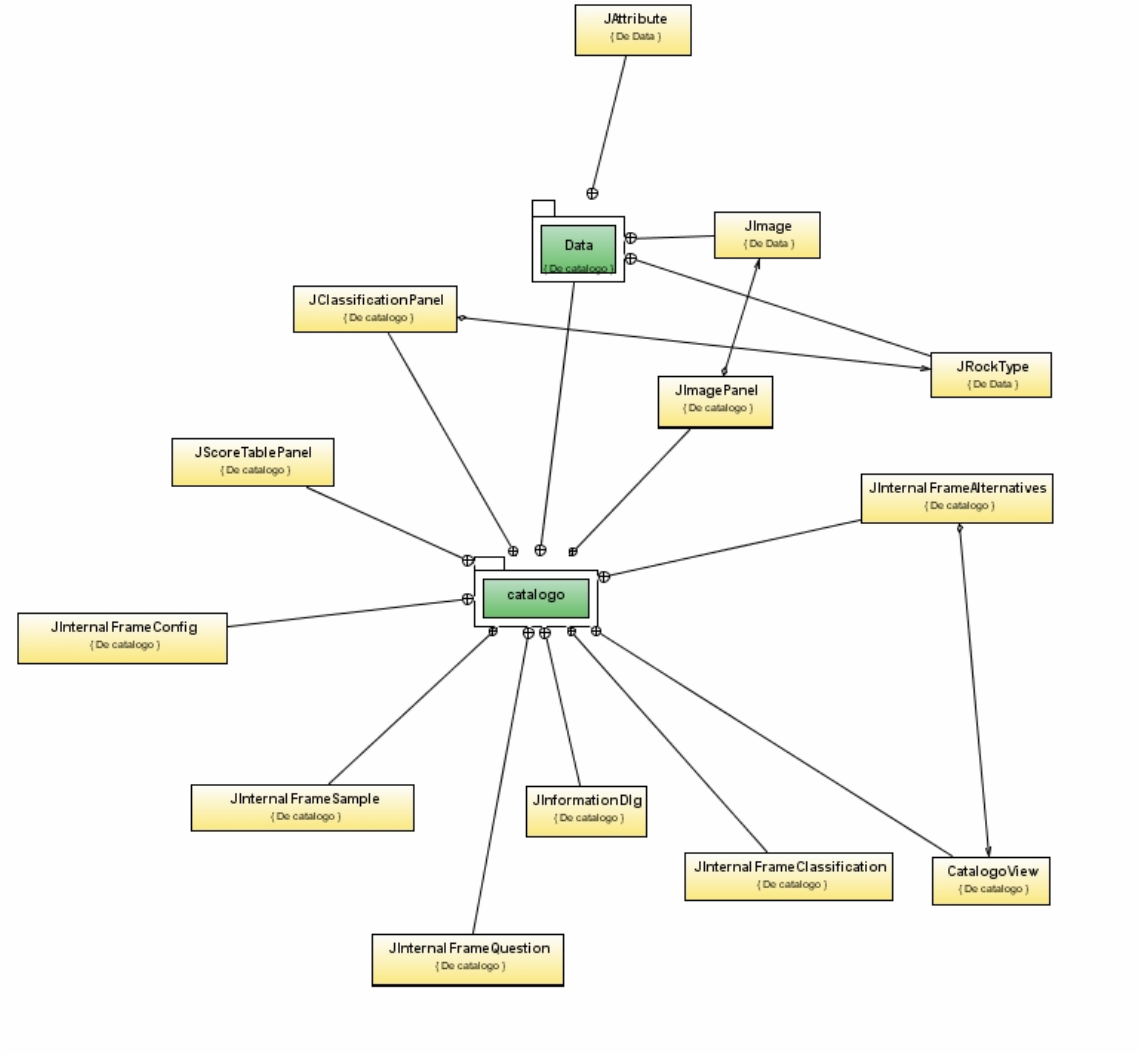
O módulo da *engine* controla basicamente a forma dos dados durante a forma da execução e funcionalidades de processamento desses objetos. Implementado utilizando *Awt*, tem como principais funcionalidades cálculos de pontuação entre as classificações e aplicação de filtro sobre os objetos a serem exibidas no módulo de interface.

O banco de dados foi projetado para adequar a aplicação a diferente tipo de objetos o que permite a utilização do catalogo em diferentes universos de problemas. No módulo de banco de dados o sistema utiliza o *Hibernate* para carregar informações sobre os tipos analisados características relevantes local das imagens e suas características, em resumo toda a configuração do sistema especialista por trás da interface.

O módulo de análise foi desenvolvido utilizando o *Microsoft Visual* C++ 2008 e a biblioteca *CxImage*. Tendo em vista algumas características que podem ser facilmente reconhecidas, este módulo pode recuperar estas características a partir da seleção da imagem a ser analisada, aplicando assim uma filtragem excluindo imagens com baixa similaridade. Esse módulo é disponibilizado na forma de serviço, para manter a portabilidade do sistema.

## 5.1. PRINCIPAIS CLASSES DE DADOS E INTERFACE

**Figura 5 - Diagrama de Classes (Principais)**




### 5.1.1. INTERFACE PRINCIPAL

Esta interface é a interface básica do sistema. Ela permite a visualização das imagens cadastradas e também acesso a outras funcionalidades que podem ser acessadas pela barra de botões a direita da interface. A classe CatálogoView atua como *container* para a área de visualização de imagens e interfaces de outras funcionalidades. As demais funcionalidades do sistema são implementadas utilizando *subframes* o que diminui a necessidade de diálogos modais, um conceito simples de interfaces tentando tornar mais amigável. A relação entre as principais classes que formam o sistema pode ser observada na Figura 5.

A área de visualização possui um modo de rolagem. O usuário ativa a direção em que deseja rolar o painel no pequeno direcional na inferior direita (figura 8). Assim o

sistema ativa uma rotina temporizada que movimenta o painel até que o fim o limite seja atingido e a rotina é então executada na direção contrária. Essa animação permanece mudando de direção até que o usuário clique no botão central do direcional, parar.

O deslocamento é executado em uma *thread* separada para que a *thread* principal possa desenhar a área de visualização. O método abaixo desloca a janela de tempo em tempo, onde esse tempo é igual à variável “WAITTIME” em milissegundos. A rotina recalcula a posição que a *view* deve estar no instante utilizando a expressão: posição inicial mais deslocamento total multiplicado pela razão do tempo decorrido pelo tempo total.

```
//inicializando variáveis de tempo a serem utilizadas.
long lTimeLastUpdate = GregorianCalendar.getInstance().getTimeInMillis();
long lTimeStart = GregorianCalendar.getInstance().getTimeInMillis();
long lEllapsed = 0;

//inicializando deslocamentos necessários para rolar uma imagem
int nHorizontalDelta = ((nDirection & (LEFT|RIGHT)) != 0) ? (PHOTOWIDTH +
PHOTOMARGIN) : 0;
int nVerticalDelta = ((nDirection & (UP|DOWN)) != 0) ? (PHOTOHEIGH +
PHOTOMARGIN) : 0;

//inversão de deltas caso deslocamento para esquerda ou para cima
if (nDirection & LEFT ) nHorizontalDelta *= -1;
if (nDirection & UP ) nVerticalDelta *= -1;

//posição inicial do cursor da janela
int nHBarPos = jMainScrollPane.getHorizontalScrollBar().getValue();
int nVBarPos = jMainScrollPane.getVerticalScrollBar().getValue();

do
{
       //verifica se o tempo decorrido é relevante para uma
       //próxima atualização de posição
       if ((GregorianCalendar.getInstance().getTimeInMillis() -
       lTimeLastUpdate) > WAITTIME)
       {

       //recalcula o delocamento.
       //posição = posicão_inicial + delta x Tdecorrido/Tanimaçao.
       m_horizontalbarpos = (int)(nHBarPos + nHorizontalDelta * lEllapsed /
       PHOTOANIMATION);
       m_verticalbarpos = (int)(nVBarPos + nVerticalDelta * lEllapsed /
       PHOTOANIMATION);

       //Movimenta a view para nova posição.
       jMainScrollPane.getHorizontalScrollBar().setValue(m_horizontalbarpos);
       jMainScrollPane.getVerticalScrollBar().setValue(m_verticalbarpos);

       //atualiza o tempo da ultima atualização
       lTimeLastUpdate = GregorianCalendar.getInstance().getTimeInMillis();
       }

       //atualiza o dempo decorrido total
       lEllapsed = GregorianCalendar.getInstance().getTimeInMillis() -
       lTimeStart;
}
```

```
while(lEllapsed < PHOTOANIMATIONTIME);

//posição final.
m_horizontalbarpos = nHBarPos + nHorizontalDelta;
m_verticalbarpos   = nVBarPos + nVerticalDelta;
jMainScrollPane.getHorizontalScrollBar().setValue(m_horizontalbarpos);
jMainScrollPane.getVerticalScrollBar().setValue(m_verticalbarpos);
```

Outro ponto de destaque na interface principal é o problema que pode ocorrer por falta de memória suficiente ao carregar todas as imagens. Em casos com imagens de alta resolução e muitas imagens cadastradas, carregar todas as imagens seria inviável e fatalmente o sistema ficaria instável.

Esse problema foi solucionado com a idéia de manter carregadas somente imagens que estão dentro da área de visualização. As demais imagens devem ser carregadas sob demanda e imagens que se tornarem desnecessárias, devem ser descarregadas da memória. A técnica soluciona o problema de falta de recursos, porem quando necessária a utilização da animação de deslizamento percebe-se um pequeno atraso para visualização das imagens, pelo processo de leitura em disco ou uma possível transferência por rede. Uma pequena alteração resolve esse detalhe. Utiliza-se uma área de *buffer*, onde colunas e linhas adicionais são carregadas para evitar o efeito de atraso de carregamento. Na Figura 6 é ilustrada a solução, mostrando ao centro a área de visualização, área buffer com as imagens imediatamente próximas e mais externamente a área não carregada.

**Figura 6 - Buffer de Visualização**

| | | | | | | |
|---|---|---|---|---|---|---|
| 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 8 | 9 | 10 | 11 | 12 | 13 | 14 |
| 15 | 16 | 17 | 18 | 19 | 20 | 21 |
| 22 | 23 | 24 | 25 | 26 | 27 | 28 |
| 29 | 30 | 31 | 32 | 33 | 34 | 35 |
| 36 | 37 | 38 | 39 | 40 | 41 | 44 |

Após cada movimento na barra de rolagem ou pela animação de deslizamento.

Deve-se atualizar a área de buffer e devolver ao sistema a memória não mais utilizada pelas imagens que não são mais visíveis. A rotina abaixo é executada em uma *thread* para atualizar o *buffer* de visualização.

```
Public void updateBuffer()
{
    for (JThumbPanel panel : m_ImagePanels)
    {
         //testa se o painel deve ser carregado
        if (isBuffered(panel))
        {
            //se ainda nao carregado: carregue
             if (null == p.getImageIcon())
                 panel.load(PHOTOWIDTH,PHOTOHEIGHT);
        }
        else
            //libera referencias a possivel imagem
            //carregada posteriormente.
             panel.setImage(null);
    }
}
private boolean isBuffered(JThumbPanel p)
{
    //calcula o retangulo de buffer + area visivel
    int left = (int) m_view.getX() - m_nBufferedSize;
    int top = (int) m_view.getY() - m_nBufferedSize;
    int right = (int) (m_view.getWidth() + m_view.getX()) + m_nBufferedSize;
    int bottom = (int) (m_view.getHeight() + m_view.getY()) + m_nBufferedSize;

    //retorna se painel está na area visivel
    return
       p.getLine()   >= top    &&
       p.getLine()   <= bottom &&
       p.getColumn() >= left   &&
       p.getColumn() <= right;
}
```

### 5.1.2. FILTRO POR CARACTERISTICAS

*JInternalFrameQuestion* é a classe responsável pela exibição de características relevantes para discriminar imagens exibidas. Uma lista de perguntas e seus possíveis valores são mostrados e o usuário seleciona quais as propriedades mais se assemelham a imagem que está sendo analisada.

As características listadas são processadas pela classe “JFilter” desenvolvida para aplicação dos métodos de filtragem de dados e também para avaliar quais os tipos de filtros podem ser aplicados com o atual conjunto de imagens. Por exemplo, na filtragem por característica se todas as imagens exibidas apresentarem a mesma cor dominante, não será exibido à pergunta “Qual a cor dominante?”.

Uma vez utilizando o modo de guia por perguntas, o usuário deve ler e

responder as perguntas que ele possui informação sobre o objeto que esta sendo analisado. A classe “JFilter” novamente entra em ação, reordenando os objetos de forma que imagens são dispostas em ordem decrescente de características comuns.

As classes “JInternalFrameQuestion” juntamente com “JQuestionPanel” controlam também alguns tipos diferentes de entradas, por exemplo, seleção múltipla e seleção exclusiva.

Outra questão comumente abordada é uma pergunta se tornar necessária apenas após a reposta de outra pergunta. Para isso as classes citadas organizam as perguntas em níveis, que podem ser configurados no banco de dados.

### 5.1.3. O MÉTODO DE PONTUAÇÃO POR CARACTERISTICAS

Ao utilizar a visualização de pontos pela barra a direita, a tabela exibida mostra o critério de avaliação que foi utilizado durante a análise. Cada característica tem sua relevância associada preenchida no banco de dados. Por exemplo, algumas raras flores apresentam a coloração negra. Logo, essa característica é um forte indicador para essas espécies. A dada característica pode ser marcada como restritiva pelo especialista.

Um caso ainda mais extremo são as características restritivas. Em determinados estudos, uma característica é necessária, como se pode ver na área mineralógica, um fragmento ser atraído ou não pelo imã define a pertinência a uma classe.

Esta é uma interface que mostra ao usuário onde e como cada uma das classificações foi avaliada. Classificações eliminadas por uma possível característica restritiva e classificações, ainda possíveis, visualizadas na lista “JInternalFrameClassification” são mostradas em uma tabela comparando as similaridades das características do tipo com todas as características respondidas pelo usuário ou obtidas automaticamente.

*JFilter* mais uma vez, por centralizar os métodos de controle dos filtros e repostas quanto às características, é o responsável por processar os dados referentes à tabela de pontuação.

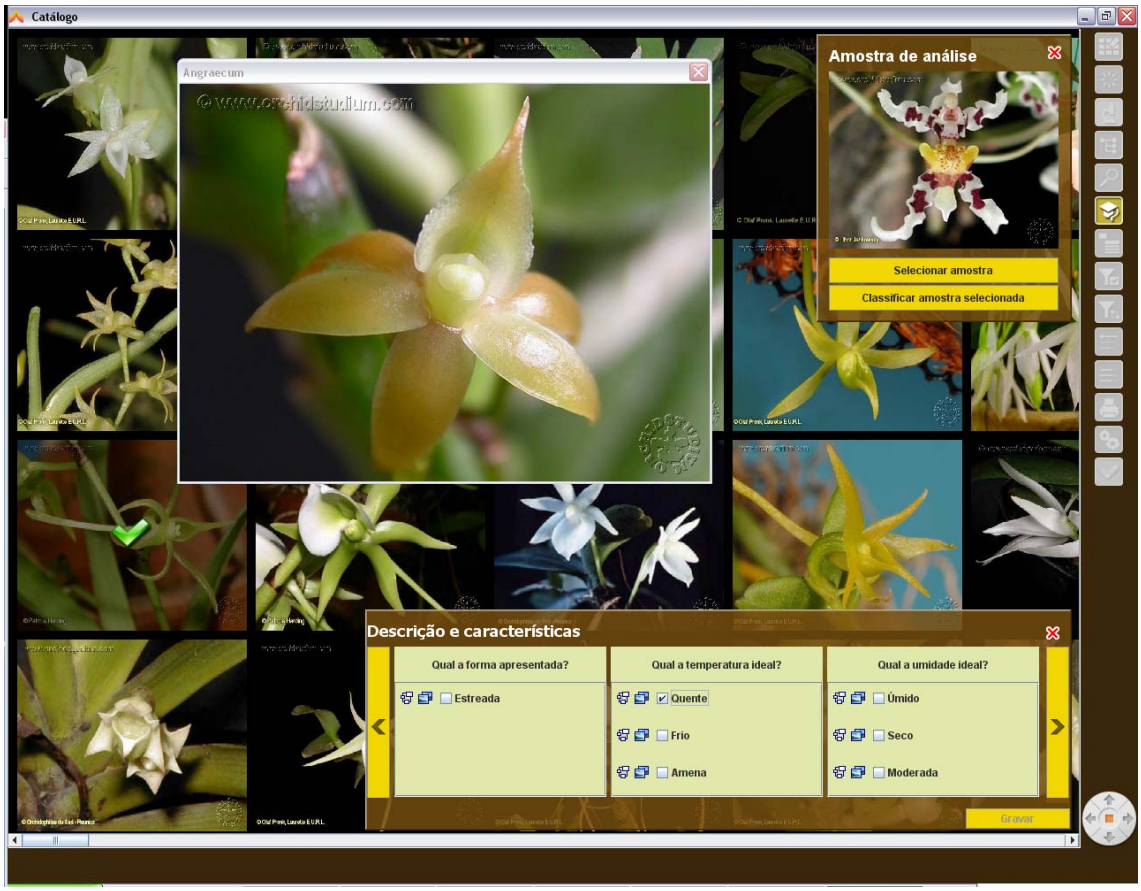
### 5.1.4. EXCLUSÃO DE CLASSIFICAÇÕES

“JInternalFrameClassification” é responsável por exibir a lista com todas as classificações possíveis para os filtros aplicados até o momento. O usuário pode nessa interface remover alguma classificação da lista. Essa funcionalidade se mostra por vezes

muito útil quando é desejado remover da interface de visualização todas as imagem de uma dada classificação.

Tendo assim após todos os módulos conectados a interface inicial como disposta na Figura 7, onde se pode ver a barra de ferramentas a direita dando acesso as funcionalidades e configurações do sistema.

**Figura 7 - A Interface Principal e os Módulos do Sistema**




## 5.2. MÓDULO DE ANÁLISE DE IMAGEM

O módulo de analise de imagens estrito utilizando C++ e MFC possibilita a análise automática de características. Pode-se, por exemplo, capturar a cor dominante na imagem, detecção de fendas e estruturas circulares.

O catalogo de imagens, ao selecionar a imagem de analise, envia uma copia da imagem para o servidor de analise configurado. O serviço de analise então começa o processamento da imagem a procura de cor dominante, presença de formas elípticas, detecção de pontas e simetrias na imagem.

Para uma melhor eficiência na analise de imagens, é ideal que a cor de fundo seja uma cor pura ou bem distinta do objeto. Logo, aplicando uma máscara, apenas o objeto alvo seria analisado, porem a padronização das imagens pode não ser uma tarefa simples.

A primeira etapa do processamento de imagem é a eliminação do fundo. Considerando que haja uma variação média entre a cor de fundo e cor do objeto, utilizando um ponto referente ao fundo, busca-se pontos vizinhos com cores próximas, Nestes pixels selecionados são modificados para uma cor de mascara.

A segunda etapa utiliza o algoritmo contorno de borda. Com a borda do objeto a ser analisado pode-se obter informações quanto à forma do objeto e cor do objeto.

Na terceira rotina avalia-se o objeto encontrado dentro do contorno encontrado. Avalia-se então dentro desta área a cor dominante na imagem. Devido à variação de luz e qualidade de lente dos equipamentos, a representação dessas imagens apresenta uma grande variação da representação RGB, mesmo tendo fotos do mesmo objeto (variação de luz, por exemplo). Foi utilizado o mapeamento de cores HSV/HSB, onde basicamente três características são extraídas: *Hue* (matiz), Saturation (saturação) e Value/Brightness (valor/brilho). Com esse padrão consegue-se então verificar a tonalidade baseada no matiz e saturação e o brilho nos indica talvez um possível reflexo de superfície ou variação entre preto e branco.

No quarto e ultimo estagio de analise busca-se por formas elípticas dentro da área delimitada pela borda gerada no segundo estágio. Dentro dessa área também se busca por simetrias de forma e cor. A busca por estruturas pontiagudas também são aplicadas nesse estágio.

### 5.2.1. REMOÇÃO DE FUNDO

A rotina para remoção da área de fundo foi implementada utilizando uma seleção de pontos de forma recursiva. O sistema gera um histograma das bordas para detectar as cores com maiores ocorrência nas áreas ao redor do centro, onde o objeto deve estar posicionado em imagens de boa qualidade. Verifica-se então, no histograma qual a composição da cor de maior incidência. Aos mais externos é verificada se a cor selecionada é similar. Caso positivo, ao ponto atual é atribuída uma cor de mascara e todos os pontos vizinhos passam pelo mesmo teste, caso contrario, o ponto continua inalterado.

Uma cópia da imagem é gerada e é aplicado um filtro conhecido como *threshold*. Esse filtro converte todos os pontos abaixo do valor de corte para a cor preta e todas as cores acima para a cor branca. O valor de corte é calculado, no caso descrito, como o valor médio dos pontos válidos em escala de cinza. A imagem cópia então, depois de passar pelo filtro *threshold*, é obtido um objeto ainda mais destacado do fundo de imagem.

Desta imagem resultante calcula-se então o contorno de borda do objeto. Que se pode por fim desenhar os pontos de borda na imagem original tendo demarcada toda a área do objeto de análise.

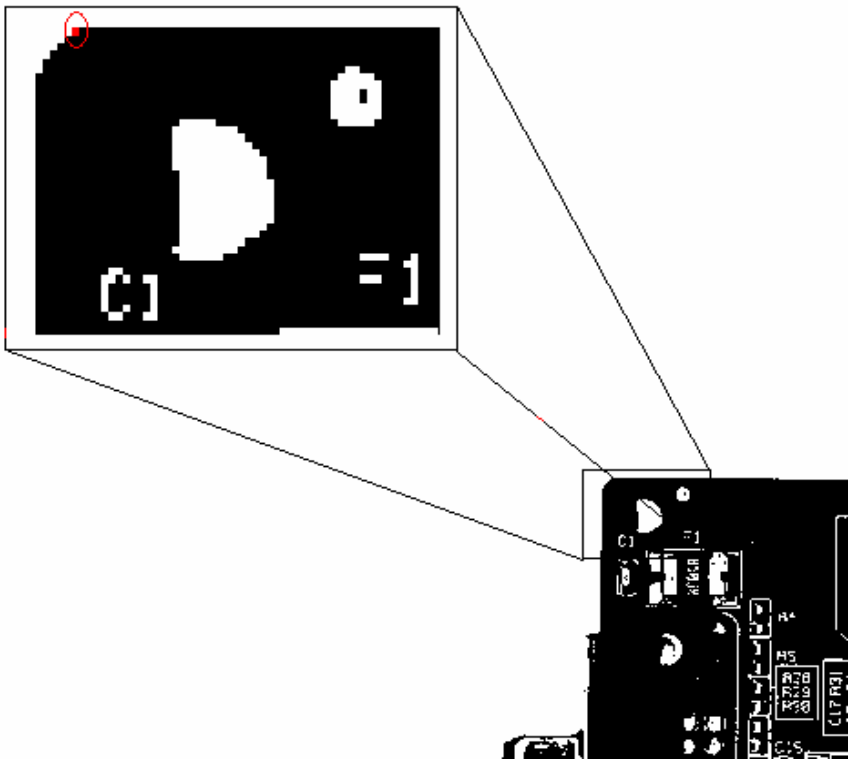
### 5.2.2. O CONTORNO DE BORDA

```
void Border( POINTS& points )
{
	CPoint pt = GetFirstPoint( );
	points.push_back( pt );

	DIRECTION directionFrom = DIR_RIGHT;
	do
	{
		pt = GetNextPoint( pt, directionFrom );
		points.push_back( pt );
	}
	while ( pt != points[0] );
}
```

A imagem é percorrida linha a linha da esquerda para a direita até encontrar um ponto preto. Este ponto é adicionado ao vetor de pontos da borda. O algoritmo utiliza como controle a direção para onde o cursor é deslocado do penúltimo para o ultimo ponto encontrado. Tendo isso e também a forma de varredura citada para encontrar o primeiro ponto, o controle de direção deve ser inicializado como vindo da direita, como mostrado na Figura 8.

Os próximos pontos da borda são obtidos por chamadas repetidas do método “GetNextPoint( ponto_atual, direção_do_ultimo_movimento )” até que retorne o primeiro ponto, o que caracteriza o fechamento da borda. Os pontos vizinhos ao ponto atual, como na Figura 9, são verificados em sentido horário, onde o primeiro a ser verificado deve ser o primeiro possível ponto de borda com base no ultimo movimento.

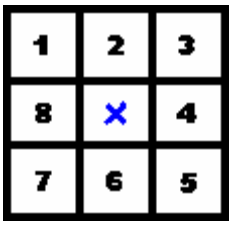
Figura 8 - Escolha de Ponto Inicial da Borda



**Primeiro ponto encontrado.**

```
Cpoint GetFirstPoint()
{
	for( unsigned y = 0; y < GetHeight(); y++ )
	{
		For( unsigned x = 0; x < GetWidth(); x++ )
		{
			if ( GetPixelColor(x,y) == RGB(0,0,0) )
			{
				return CPoint(x,y);
			}
		}
	}

	return CPoint(-1,-1);
}
```

Figura 9 - Mapeamento de Vizinhança Utilizado



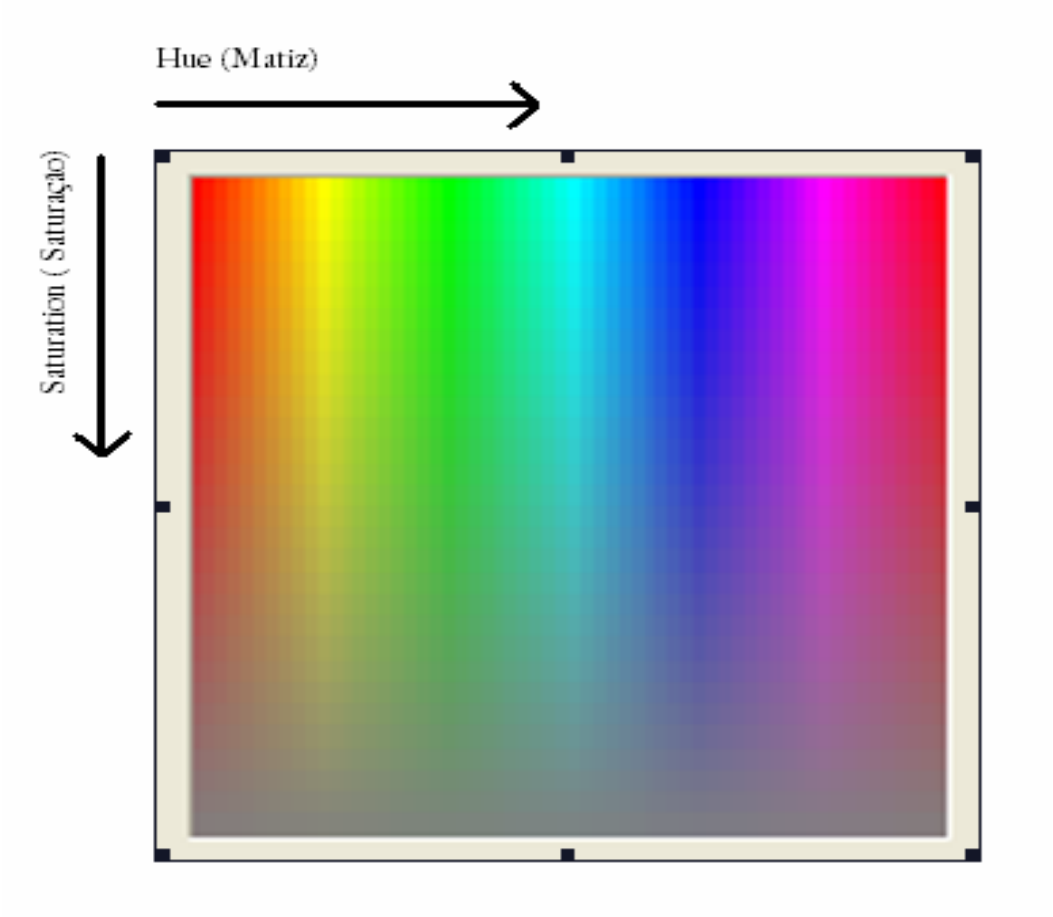

```
CPoint GetNextPoint( CPoint pt, DIRECTION& directionFrom)
{
```

```
      switch ( nFind )
      {
         case 2:
         case 3: return GetNeighborPoint( ptActual, directionFrom, 1);

         case 4:
         case 5: return GetNeighborPoint( ptActual, directionFrom, 3);

         case 6:
         case 7: return GetNeighborPoint( ptActual, directionFrom, 5);

         case 8:
         case 1: return GetNeighborPoint( ptActual, directionFrom, 7);

      };

      return CPoint(0,0);
}
```

### 5.2.3. ANALISE DE CORES DOMINANTES

Devido a problemas que possam ocasionar variação de cores, HSB mostrou bons resultados. Tendo em vista que esse modelo de cor extrai três características muito importantes: H: Matiz (hue), S: Saturação (saturation) e B: Brilho (brightness).

**Figura 10 - Paleta HSB, Hue x Saturation (Matiz x Saturação)**




A Imagem mostrada na Figura 10 ilustra uma paleta de matiz por saturação. Internamente o sistema divide a paleta de cores HSB em uma matriz 16x16. Cada área é identificada por correspondente uma cor, por exemplo, qualquer ponto que apresente matiz 0-15 e saturação 0-15 pertence à primeira área de delimitação de cor. O parâmetro “brilho” nos dá informações necessárias para identificar se a cor se caracteriza como clara, escura ou como luminosidade ideal.

Tendo assim, o sistema avalia pixel a pixel a área interna a borda calculada. Os pixels são avaliados quanto à pertinência nas áreas de classificação e uma ou duas cores podem ser consideradas como resposta caso não haja uma diferença maior que 20% entre elas.

### 5.2.4. ANALISE DE ESTRUTURAS

A rotina busca internamente a imagem, já com o fundo removido, procurando por novas bordas internas. O algoritmo calcula quais os dois pontos mais distantes entre si pertencentes a esta borda interna, assim então se encontra o maior eixo da elipse. O eixo menor é encontrado facilmente traçando uma reta perpendicular ao eixo já

encontrado. Calcula-se então a excentricidade da elipse e valores muito baixos (próximos de 0) nos mostram um uma possível fenda no objeto, enquanto valores mais próximo de 1 nos mostram uma estrutura circular interna. Em caso da avaliação nos mostrar uma possível estrutura circular aplica-se uma verificação para avaliar se a borda calculada pode ser aproximada a uma elipse ou quadrilátero. Com isso pode-se definir possíveis fendas ou formas circulares.

A rotina também busca na borda da imagem principal seqüências de pontos, onde ocorre mudança de sentido indo em direção do centro da imagem. Tais mudanças indicam alguma estrutura arredondada ou de pontiaguda. Pequenas variações devem ser desprezadas, tendo em vista que podem existir imperfeições na imagem. Assim as seções com grandes variações, os pontos constituintes são submetidos ao método de regressão linear de ordem dois. Caso esse teste retorne grande erro, deduz-se que tal variação não é uma parábola o que nos leva a detecção de uma ponta.

Por ultimo, o sistema avalia simetrias não muito refinadas. Um método simples é aplicado, onde a imagem é dividida ao meio verticalmente e em seqüência uma das metades espelhada e subtraída da outra metade. A obtenção de uma imagem resultante à subtração que apresente pequena quantidade de pixels validos, especialmente em pontas e mudanças de sentido em direção ao centro, mostram uma possível simetria.

# 6. BANCO DE DADOS

Para o armazenamento e consulta dos dados foi utilizado o SGBD *Postgre*SQL. Os dados armazenados são de dois tipos: Dados relativos ao funcionamento do sistema; Dados de informações sobre objetos armazenados.

## 6.1. Ferramenta SGBD PostgreSQL

*PostgreSQL* é um sistema gerenciador de banco de dados objeto relacional (SGBDOR), desenvolvido como projeto de código aberto.

O *PostgreSQL* implementa algumas funções importantes para o desenvolvimento do catálogo de imagens sendo elas:

- Consultas complexas
- Chaves estrangeiras
- Integridade Transacional

## 6.2. Dados relativos ao funcionamento do sistema

São os dados que são carregados assim que o sistema é iniciado, através do preenchimento desses dados configura-se o sistema para o tipo de aplicação que ele será utilizado. Esses dados funcionam como um domínio das possíveis características e classificações que os objetos fotografados podem ter.

Fazem parte deste grupo de dados as classificações, as características e os possíveis valores que estas características podem assumir além de imagens, descrições e outros atributos destes três primeiros.

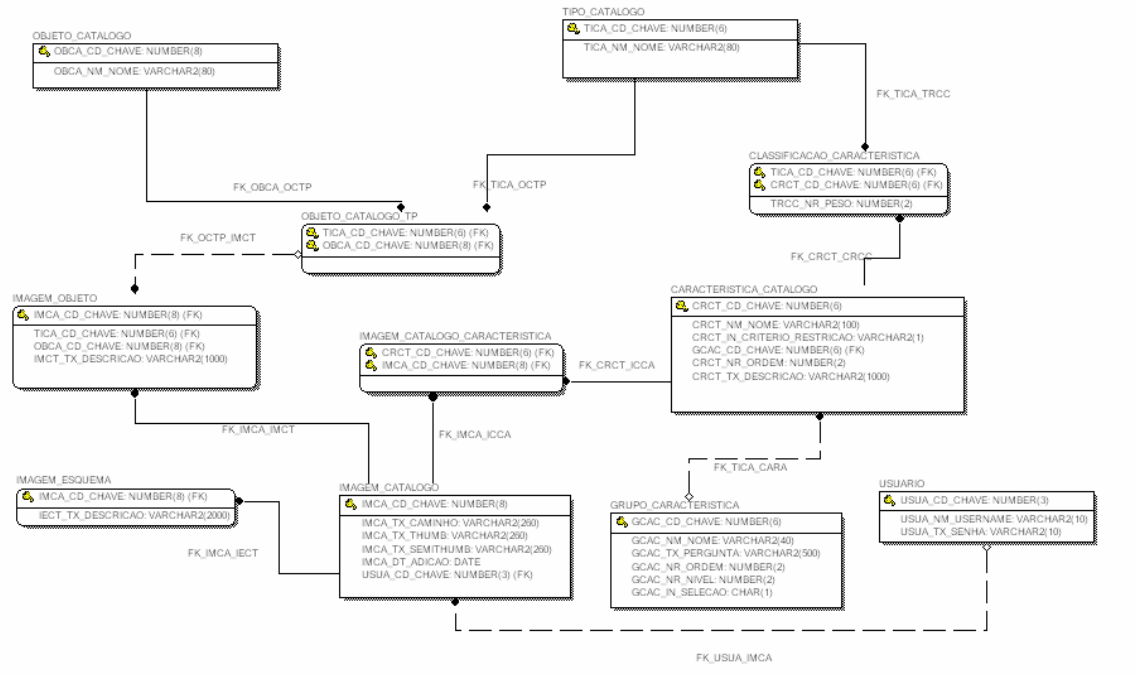
## 6.3. Informações sobre objetos armazenados

São os dados sobre um único exemplar do objeto analisado, como seu nome, sua classificação, suas características, suas fotos, etc. O gerenciamento desses dados é o propósito do sistema.

## 6.4. Esquema Entidade-Relacionamento

O modelo de entidade-relacionamento é mostrado na imagem da Figura 11.

**Figura 11 - Modelo Entidade-Relacionamento**



## 7. APLICAÇÕES DESENVOLVIDAS E RESULTADOS

Para ilustrar a aplicação do catálogo de imagens foi escolhida a área de botânica, em especial a família das orquídeas (*orchidaceae*). Por serem flores que apresentam um leque muito amplo de características, é um exemplo perfeito de aplicabilidade do sistema.

A grande utilização das orquídeas é a ornamentação. Por apresentarem muitíssimas e variadas formas, cores e tamanhos elas despertam o interesse de muitos colecionadores que se reúnem em associações orquidófilas espalhadas por todo o mundo, formando um numeroso grupo de possíveis usuários do sistema.

Para que o sistema fosse configurado como Catálogo de Orquídeas foram preenchidos os dados relativos ao funcionamento do sistema da seguinte forma:

Classificações

- Epidendrum
- BrassiaMiltonia
- Angraecum
- Catasetum
- Cattleya
- Cymbidium
- Dembrobium
- Encyclia
- Laelia
- Masdevallia
- Odontoglossum
- Oncidium
- Paphiopedilum
- Phalaenopsis
- Stanhopea
- Vanda
- Vanilla

Características e Valores:

- COR
  - Amarela
  - Branca
  - Verde
  - Rosa
  - Roxo
  - Vermelho
  - Violeta
  - Negra
  - Ocre

- FORMA
  - Estrelada

- TEMPERATURA
  - Quente
  - Frio
  - Amena

- UMIDADE
  - Úmido
  - Seco
  - Moderada

- LUMINOSIDADE
  - Direta
  - Indireta
  - Alta
  - Baixa
  - Moderada

- SUBSTRATO
    - Terra
    - Tronco
    - Rocha
    - Xaxim
    - Casca de madeira
    - Placa de cerâmica
    - brita com substrato

- ODOR
    - Suave
    - Forte
    - Perfumada

- LABELO
    - Curto
    - Largo
    - Achatado
    - Espesso
    - Reniforme ou riniforme
    - Grande

- CALO NA BASE LABELO
    - Sim
    - Não

- HASTE
    - Curta
    - Média
    - Longa
    - Rígida
    - Flexível
    - Fina

- SEXO
    - Monóica
    - Dióica

Nenhuma característica foi configurada como restrição e todos os pesos foram configurados iguais com o valor “1”.

# 8. CONCLUSÃO

A partir do desenvolvimento de catálogo de imagens foi criada uma ferramenta para auxiliar o trabalho de classificação e organização de imagens. Esta nova ferramenta poderá ser customizada para aplicações de fins específicos alterando apenas as informações contidas no seu banco de dados para os dados daquele fim.

Foram criados algoritmos para lidar com problemas advindos da grande quantidade de imagens em face de uma quantidade limitada de memória.

Como exemplo de utilização da aplicação, diversas imagens de orquídeas foram catalogadas com suas características e classificações. Essas imagens foram obtidas em *sites* especializados nesta espécie de planta.

# 9. TRABALHOS FUTUROS

Apesar do sistema Catalogo de imagens implementar ferramentas para auxiliar a classificação de imagens ela é feita manualmente, portanto frente a um número muito grande de imagens, o trabalho para realizar a classificação também será grande.

Referente aos algoritmos de análise de imagem nota-se melhorias ao utilizar métodos de detecção de bordas mostrados na literatura, o que pode aumentar a confiabilidade quando comparado ao método de contorno apresentado. Algoritmos de simetria e análise de pontas também podem utilizar de métodos matemáticos mais robustos.

A filtragem e ajuda para classificar um objeto, que hoje submete o usuário a uma série de perguntas, pode futuramente basear a forma de pontuação dinâmica em uma rede neural. Onde os pesos assumidos podem ser ajustados automaticamente dependendo classificações previamente realizadas.

Modelos tridimensionais dos objetos fotografados também poderiam ser armazenados e visualizados pelo catalogo. Esses modelos possuem informações espaciais do objeto que muitas vezes passam despercebidas em um ambiente bidimensional. Juntando os modelos com as imagens dos objetos seria obtida uma visão completa dos objetos texturizados pelas imagens.

## 10. REFERÊNCIAS


BAUER, Christian e KING, Gavin. *Hibernate in Action*. 1 ª ed. Manning Publications. EUA, 2004.

CADENHEAD. Rogers. *Aprenda em 21 Dias Java*. 4ª ed. Editora Campos. Rio de Janeiro, 2005.

PRESSMAN. Roger S. *Engenharia de Software*. 6ª ed. Editora McGraw-Hill. Rio de Janeiro, 2006

PROGRAMMING TUTORIALS AND SOURCE CODE EXEMPLES. Disponível em: <*www.java2s.com*>. Acesso em fevereiro de 2010.

ORQUIDEANA. Disponível em: <*http://www.orquideana.com.br/vanilla.html*>. Acesso em fevereiro de 2010.

A ORQUÍDEA. Disponível em: <*http://www.aorquidea.com.br*>. Acesso em fevereiro de 2010.

GLOSSÁRIO DE BOTÂNICA. Disponível em: <*http://w3.ufsm.br/herb/glossario.htm*>. Acesso em fevereiro de 2010.

AZEVEDO, Eduardo, LETA, Fabiana e CONCI, Aura. *Processamento de Imagens Digitais - Volume 2*. 1ª ed. Editora Elsevier. Rio de Janeiro, 2007

## 11. ANEXO

**Tabela de Classificação x Características das orquídeas cadastradas no sistema.**

| | | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Epidendrum | Brassia | Miltonia | Angraecum | Catasetum | Cattleya | Cymbidium | Dembrobium | Encyclia | Laelia | Masdevallia | Odontoglossum | Oncidium | Paphiopedilum | Phalaenopsis | Stanhopea | Vanda | Vanilla |
| | **COR(1)** | | | | | | | | | | | | | | | | | | |
| 1 | Amarela | | x | | x | | | | x | | | | | | | | | | |
| 2 | Branca | | x | x | x | | | | x | | | | | | | | | | |
| 3 | Verde | | x | | x | | | | | | | | | | | | | | x |
| 4 | Rosa | | | x | | | | | x | | | | | | | | | | |
| 5 | Roxo | | | x | | | | | | | | | | | | | | | |
| 6 | Vermelho | | | x | | | | | x | | | | | | | | | | |
| 7 | Violeta | | | x | | | | | | | | | | | | | | | |
| 8 | Negra | | | x | | | | | | | | | | | | | | | |
| 9 | Ocre | | | | x | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | | | | |
| | **FORMA(2)** | | | | | | | | | | | | | | | | | | |
| 10 | Estrelada | x | x | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | | | | |
| | **TEMPERATURA(3)** | | | | | | | | | | | | | | | | | | |
| 11 | Quente | x | | | x | x | x | | x | | x | x | | x | | x | x | | x |
| 12 | Frio | x | | x | | | | x | | | | x | | x | | | | | |
| 13 | Amena | | x | x | | | | x | | x | | | x | | | | | x | |
| | | | | | | | | | | | | | | | | | | | |
| | **UMIDADE(4)** | | | | | | | | | | | | | | | | | | |
| 14 | Úmido | | x | | x | x | x | | x | x | x | | | x | | | x | x | x |
| 15 | Seco | | | | | | | | | | | | | | | | | | |
| 16 | Moderada | | | | | | | | | x | | | x | | | | | | |
| | | | | | | | | | | | | | | | | | | | |
| | **LUMINOSIDADE(5)** | | | | | | | | | | | | | | | | | | |
| 17 | Direta | x | | | x | x | | | | | x | | | | | | | | |
| 18 | Indireta | | x | | x | x | | | | | x | | | x | | x | | | |
| 19 | Alta | | | | x | x | x | x | x | | | | x | x | | | | x | x |
| 20 | Baixa | | | x | | | | | | | | | | | | x | | | |
| 21 | Moderada | | | x | | x | | | | | x | | | x | x | | | | |
| | | | | | | | | | | | | | | | | | | | |
| | **SUBSTRATO(6)** | | | | | | | | | | | | | | | | | | |
| 22 | Terra | x | x | x | x | x | x | x | x | | | | | | x | | | | x |

| | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 23 | Tronco | x | | | x | | x | | x | | | | | | | | x | |
| 24 | Rocha | x | | | x | | | | | | | | | | | | | |
| 25 | Xaxim | x | | | | x | | | x | | | | | x | | x | | |
| 26 | Casca de madeira | | | | | x | x | | | | | | | | | x | | |
| 27 | Placa de cerâmica | | x | x | | | | | | | | | | | | | | |
| 28 | Brita com substrato | | | | | | | | | | | | | | | | x | |
| | | | | | | | | | | | | | | | | | | |
| | **ODOR(7)** | | | | | | | | | | | | | | | | | |
| 29 | Suave | | | x | | | | | | | | | | | | | | |
| 30 | Forte | x | | | | | | | | | | | | | | | | |
| 31 | Perfumada | | x | | x | x | | | | x | | | | | | | | |
| | | | | | | | | | | | | | | | | | | |
| | **LABELO(8)** | | | | | | | | | | | | | | | | | |
| 32 | Curto | | | | | | | | | | | x | | | | | | |
| 33 | Largo | | | x | | | | | | | | | | | | | | |
| 34 | Achatado | | | x | | | | | | | | | | | | | | |
| 35 | Espesso | | | | | | | | | | | | | | | | | |
| 36 | Reniforme ou riniforme | | | | | x | | | | | | | | | | | | |
| 37 | Grande | | | | | | x | | | | | | | | x | | | |
| | | | | | | | | | | | | | | | | | | |
| | **CALO NA BASE LABELO(9)** | | | | | | | | | | | | | | | | | |
| 38 | Sim | | | | | | | | | | | | | | | | | |
| 39 | Não | | | x | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | | | |
| | **HASTE(10)** | | | | | | | | | | | | | | | | | |
| 40 | Curta | | | | | | | | | | | | | | | | | |
| 41 | Média | | | | | | | | | | | | | | | | | |
| 42 | Longa | | x | | | | | | | | | | | | | | | |
| 43 | Rígida | | | | | | | | | x | | | | | | | | |
| 44 | Flexivel | | | | | | | | | | | | | | | | | |
| 45 | Fina | | | | | | x | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | | | |
| | **SEXO(11)** | | | | | | | | | | | | | | | | | |
| 46 | Monóica | | | | | x | | | | | | | | | | | | |
| 47 | Dióica | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | | | |