UNIVERSIDADE FEDERAL DE SANTA CATARINA

METADATA INGESTER

Ferramenta de indexação e busca sobre

arquivos de metadados em imagens

Taise Barreiros Fernandes Marques

Florianópolis – SC

2012/2

UNIVERSIDADE FEDERAL DE SANTA CATARINA

DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA

CURSO DE CIÊNCIAS DA COMPUTAÇÃO

METADATA INGESTER

Ferramenta de indexação e busca sobre

arquivos de metadados em imagens

Taise Barreiros Fernandes Marques

Trabalho de conclusão de curso

apresentado como parte dos

requisitos para obtenção do grau de

Bacharel em Ciências da Computação

Florianópolis – SC

2012/2

Taise Barreiros Fernandes Marques

METADATA INGESTER

Ferramenta de indexação e busca sobre

arquivos de metadados em imagens

Trabalho de conclusão de curso apresentado como parte dos requisitos para obtenção do grau de Bacharel em Ciências da Computação

Orientadora: Carina Dorneles

Banca Examinadora:

Roberto Willrich

Ronaldo dos Santos Mello

1. SUMÁRIO

2. LISTA DE FIGURAS

3. RESUMO

O fácil acesso à internet e à informação resultou no aumento do armazenamento de dados, principalmente em repositórios. E para que esses dados possam ser recuperados, o metadado vem sendo uma ferramenta importante para a organização e identificação rápida sem a necessidade de acesso direto a esses dados. Percebendo a necessidade de uma ferramenta para a recuperação de dados, este trabalho propõe um framework que permita a recuperação de informações em metadados associados a outros dados. Inicialmente, o framework trata metadados associados a arquivos de imagens, por serem mais comuns e numerosos. Este framework reconhece três tipos de informações associadas às imagens, que são os metadados textuais, os metadados embutidos e as informações sobre coloração da imagem, fornecendo quatro tipos de buscas para a recuperação dessa informação, que são buscas simples (retornando palavras estruturalmente iguais à que foi fornecida), por similaridade (retornando palavras estruturalmente semelhantes), por sinônimo (retornando palavras que possuem o mesmo sentido da que foi fornecida) e por substring (retornando palavras que contenham a palavra fornecida). O framework foi implementado afim de por em prática as definições teóricas e permitir a realização de testes de revocação (fração dos documentos relevantes, observada dentre os documentos examinados) e precisão (fração dos documentos já examinados que são relevantes), que comprovaram um resultado satisfatório no retorno dos dados.

4. ABSTRACT

Easy access to the internet and information resulted in increased data storage, mainly in repositories. And in order to these data can be recovered, the metadata has been a important tool for organizing and rapid identification without the direct access to these data. Realizing the need to recover that data, this paper proposes a framework that allows the retrieval of information in metadata associated with other data. Initially the framework treats metadata associated with images, because they are more common and numerous. It recognizes tree types of information associated with images that are textual metadata, embedded metadata and color information of the image, providing four types of searches for the recovered information which are simple search (returning words structurally identical to the one provided), similarity search (returning structurally similar words), synonymous search (returning words that have the same sense of that was provided) and substring search (words that contain the word provided). The framework was implemented in order to put in practice the theoretical definitions and allow tests of recall (fraction of relevant documents found among the documents examined) and precision (fraction of documents previously examined that are relevant), who provided a satisfactory result in the return of data.

5. INTRODUÇÃO

A explosão da Internet e dos repositórios de conteúdos digitais tem trazido uma grande quantidade de dados ao alcance de todos. Com o tempo, a quantidade de dados disponíveis se tornou tão vasta que surgiu a necessidade de formas alternativas e mais dinâmicas de encontrar a informação (GOSPODNETIC & HATCHER, 2005). Nesse contexto, o metadado acabou se tornando popular e amplamente utilizado.

O metadado consiste em um grupo de informações sobre o dado ao qual ele está associado, auxiliando assim na organização desses dados. Estes incluem elementos de descrição do conteúdo dos dados e qualquer informação relevante para a recuperação dos seus conteúdos. Podem ser destacadas as seguintes vantagens na utilização e disponibilização de metadados: estabelecimento de padrões de dados diante da heterogeneidade de informações contidas em rede, como por exemplo, a Internet; facilidade e maior precisão na recuperação das informações desejadas; troca de informações entre aplicações e organizações (GARCIA, MOURA, & CAMPOS, 1999).

Um tipo de metadado, mais recentemente difundido principalmente por alguns modelos de câmeras digitais, é o metadado em imagens que podem ser tanto embutido na própria imagem como em arquivos externos à imagem. O modelo do metadado embutido depende do formato da imagem, por exemplo, em imagens no formato de compactação JPEG ou sem compactação no formato TIFF, o modelo do metadado embutido é o chamado EXIF (*CIPA DC-008-Translation-2012*). Como entre os arquivos de imagens o formato mais comum é JPG ou JPEG, o metadado EXIF se tornou

um padrão. Este metadado torna acessível as informações que envolvem uma imagem, mas que não estão evidentes na própria imagem, como por exemplo, a data e hora em que ela foi capturada e até mesmo seu geo-posicionamento. Já o metadado em arquivo externo, normalmente, é um arquivo de texto, que pode ter ou não a formatação baseada na linguagem de marcação XML (W3C.Org). Nesse arquivo externo podem ser acrescentadas outras informações pelo próprio usuário de maneira a dar um sentido semântico e pessoal a esse metadado, como por exemplo, a informação que a foto foi capturada no aniversário de um amigo. Porém, nem sempre esse metadado está completo e no mesmo formato, dificultando a manipulação e recuperação dessa informação.

Existem algumas ferramentas que disponibilizam recursos para auxiliar nesse processo de recuperação da informação, mas essas ferramentas estão ou em bibliotecas, como no *Image Metadata JPEG* (CPAN.ORG) ou embutidos em outros grupos de ferramentas comerciais em forma de um *plugin*, assim como no software *Zoom Search Engine* (ZOOM_WRENSOFT.COM) com o acréscimo do *plugin ImageInfo* (WRENSOFT_PLUGINS.COM).

A proposta deste trabalho, em uma primeira etapa, é oferecer um *framework* de busca a informações em arquivos de metadados associados a arquivos multimídia, totalmente independente de qualquer outra ferramenta e totalmente gratuita, disponibilizando operações de indexação e busca, integrando informações de três fontes distintas do arquivo: estrutura (tamanho, formato, coloração, etc), metadado embutido(data de captura, câmera utilizada, etc) e metadado textual (dados cadastrados pelo próprio usuário que atribui a semântica do arquivo, como por

exemplo, a ocasião em que a foto foi tirada). Essas informações podem ser cruzadas livremente, utilizando-se apenas uma ou combinações de várias ou até mesmo todas as características dos arquivos extraídas das fontes citadas acima. Podem ser utilizadas buscas simples, por similaridade, por sinônimos, por substring ou até mesmo uma combinação das mesmas. Sendo que a busca simples retorna palavras estruturalmente iguais à que foi fornecida, a busca por similaridade retorna palavras estruturalmente semelhantes, a busca por sinônimos retorna palavras que possuem o mesmo sentido da que foi fornecida e a busca por substring retorna palavras que contenham a palavra fornecida.

Em uma segunda etapa, foi realizada a implementação dessas funcionalidades em uma ferramenta chamada *Metadata Ingester* (MI), onde podem ser executados os testes sobre todas as funcionalidades propostas.

Além desse capítulo introdutório, o trabalho conta com outros quatro capítulos: C*apítulo 2* que evidencia o embasamento teórico envolvido no projeto, necessário como base para a boa compreensão das etapas seguintes; *Capítulo 3* que descreve as ferramentas relacionadas ao assunto em questão, mostrando o recurso que essas disponibilizam para a indexação e busca de informações em metadados. *Capítulo 4* que apresenta e descreve o projeto desenvolvido nomeado como Metadata Ingester; *Capítulo 5* que cita os trabalhos futuros e melhorias que podem ser desenvolvidas sobre o Metadata Ingester; *Capítulo 6* que menciona todas as referências bibliográficas utilizadas como base informativa para o desenvolvimento desse projeto.

6. FUNDAMENTAÇÃO TEÓRICA

6.1. INDEXAÇÃO E BUSCA

Buscas referenciam apenas uma pequena proporção dos registros de um arquivo, sendo assim, não é eficiente para o sistema ler cada registro, o ideal é que o sistema seja capaz de localizá-los diretamente (SILBERSCHATZ, KORTH, & SUDARSHAN, 2006, pp. 321, Cap12). Se esses registros estiverem organizados, o seu acesso e leitura são mais eficientes e rápidos, aumentando a agilidade de todo o processo de busca. Para permitir essas formas de acesso, foram projetadas estruturas especiais que são associadas aos arquivos chamadas índices (SILBERSCHATZ, KORTH, & SUDARSHAN, 2006). Indexação é o processamento de uma informação original para uma referência cruzada eficiente facilitando as buscas rápidas (GOSPODNETIC & HATCHER, 2005).

No contexto de acesso à informação, existem sistemas chamados *Search Engine* (motor de busca). Esses sistemas oferecem uma facilidade na geração de índices eficientes, além de recursos para a recuperação desses dados, disponibilizando esses resultados normalmente em formato de listas, chamadas *hits* (SHERMAN & PRICE, 2001). A categoria de *Search Engine* pode ser representada por bibliotecas de recuperação de informação, como Egothor (EGOTHOR), Lucene (LUCENE_APACHE.ORG) e Xapian (XAPIAN.ORG), além de aplicações de indexação e busca, como Microsoft Index Server (MSDN Microsoft), Namazu (NAMAZU.ORG) e Swish (SWISH.ORG).

6.1.1. INDEXAÇÃO

Os arquivos sequenciais indexados são um dos esquemas de índice mais antigos usados nos sistemas de banco de dados. Para permitir a rápida recuperação de registros na ordem da chave de busca, os registros são armazenados sequencialmente, e aqueles fora de ordem são encadeados (SILBERSCHATZ, KORTH, & SUDARSHAN, 2006, pp. 351, Cap12).

As estruturas de índices tipicamente fornecem um caminho secundário de acesso, que proveem formas alternativas de acessar os dados sem afetar o local físico desses dados em disco. Eles permitem um acesso eficiente aos dados baseando-se nos campos indexados que são usados para construir o índice (NAVATHE & ELMASRI, 1999, pp. 134, Cap.6).

Não existem estruturas de índices melhores ou piores. Cada estrutura de índice tem um campo de atuação e neste campo esta estrutura se destaca. Assim como dentro do contexto dos Search Engines a estrutura de índices mais utilizada é o índice invertido. Esta estrutura tem como foco principal a otimização da velocidade na recuperação do documento que contém determinada palavra chave, além de tornar eficiente a utilização de espaço em disco. E possui a clássica descrição: “No lugar de fazer a pergunta ‘Quais palavras estão contidas nesse documento?’ é feita a pergunta ‘Quais documentos contém essa palavra?’ ” (GOSPODNETIC & HATCHER, 2005). A ideia básica é manter um dicionário de termos e para cada termo existe uma lista com as referências dos documentos que essa palavra aparece (MANNING, RAGHAVAN, & SCHÜTZE, 2008).

Um exemplo prático dessa estrutura: Tendo os documentos D0 que contém as palavras "*it is what is*", D1 com "*what is it*" e D2 com "*it is a banana*", resulta na seguinte estrutura de índices invertidos (levando-se em consideração que os inteiros 0, 1 e 2 vão representar D0, D1 e D2, respectivamente) mostrada na Figura1:

"it"	*{0, 1, 2}*
"is"	*{0,1,2}*
"what"	*{0,1}*
"a"	*{2}*
"banana"	*{2}*

Figura 1 - Exemplo de estrutura de índice invertido

O índice invertido representado na figura acima armazena a informação de que a palavra "*it*" está presente nos documentos D0, D1 e D2, representado pelo conjunto {0, 1, 2}, a palavra "*what*" está presente nos documentos D0 e D1 representado pelo conjunto {0, 1}, a palavra "*a*" está presente no documento D2, representado pelo conjunto {2} e a palavra banana está presente no documento D2 representado pelo conjunto {2}, Então em uma consulta por "*what*", "*is*" e "*it*", resultaria na seguinte operação de intersecção entre conjuntos:

$$\{0,1\} \cap \{0,1,2\} \cap \{0,1,2\} = \{0,1\}$$

Após o cálculo das intersecções entre os conjuntos que representam os documentos que contém as palavras "*what*", "*is*" e "*it*", o resultado seria o conjunto de documentos que contém as três palavras simultaneamente, que no exemplo dado acima é o conjunto {0,1}, que representa os documentos D0 e D1.

6.1.2. BUSCA

Para se recuperar uma palavra de um documento é possível simplesmente vasculhar cada documento verificando se essa palavra está ou não presente. Porém essa abordagem é ineficiente quando o conjunto de documentos a ser pesquisado é de grande escala ou mesmo se cada documento for extenso. Para evitar problemas dessa natureza, todos os *Search Engines* utilizam, como um conceito básico, o recurso de índices, processando a informação original para um eficiente alvo de referência cruzada, a fim de facilitar a rápida busca posterior, eliminando a necessidade de uma lenta busca sequencial (GOSPODNETIC & HATCHER, 2005).

De acordo com *Gospodnetic* e *Hatcher*, em *Lucene in Action* de 2005, o núcleo dos *Search Engines* atuais são índices invertidos, pelo fato destas estruturas tornarem eficiente o uso de espaço em disco enquanto permitem rápidas buscas por palavras chaves.

Já em uma camada inferior, todos os *Search Engines* dependem de um acesso extremamente rápido a um índice invertido para atingir a taxa de transferência da busca requerida. Em particular, para cada consulta o *Search Engine* precisa passar por todas as listas invertidas correspondentes aos termos da pesquisa, a fim de identificar um conjunto limitado de documentos promissores, que podem então ser analisados de uma forma mais completa em uma fase posterior (YAN, DING, & SUEL, 2009).

6.2. APACHE LUCENE

Apache Lucene (LUCENE_APACHE.ORG) é uma biblioteca para indexação e busca disponibilizada pela *Apache Software Foundation* (APACHE.ORG). É um membro da

popular família de projetos da *Apache Jakarta* (JAKARTA.ORG) e é atualmente, e tem sido por alguns anos, a mais popular biblioteca de Recuperação de Informações Java gratuita. Utiliza modernos algoritmos e conceitos na indexação, como índices invertidos, e recuperação de resultados, como escalonamento sobre *cluster*. O Lucene disponibiliza um simples, porém poderoso, conjunto de funcionalidades que requer o mínimo de entendimento de indexação textual e buscas. Apenas é necessário aprender a manipular tais recursos para a integração com a aplicação a qual o Lucene é utilizado (GOSPODNETIC & HATCHER, 2005).

Pela sua praticidade e robustez, esta biblioteca é utilizada em consolidadas aplicações web como: AOL (http://www.aol.com/), Apple (http://www.apple.com/), Disney (http://www.disney.com.br/), Eclipse (http://www.eclipse.org/), IBM (http://www.ibm.com/us/en/), LinkedIn (http://www.linkedin.com/). E por ser amplamente difundida, conta com uma extensa documentação e suporte, facilitando a busca por informações e resolução de problemas. A combinação desses fatores resultou na escolha desta biblioteca para a indexação e busca do trabalho em questão.

6.3. METADADOS ASSOCIADOS ÀS IMAGENS

Metadados são modelos de representação ou abstração dos dados, com o objetivo de descrição da coleção e identificação das características de cada componente da coleção. Os metadados têm um papel muito importante na administração de dados, pois é a partir deles que as informações são selecionadas, processadas, e consultadas (MOURA, 2005). Através deles é possível descrever as informações contidas em um

determinado documento, permitindo assim a obtenção de informações sobre o mesmo sem que seja necessário utilizá-lo diretamente.

O metadado pode ser representado de várias formas, pelo fato de que a sua definição é dada de maneira abstrata: “Dados sobre dados”. Dentro do contexto de arquivos de multimídia, o metadado pode ser descrito como uma entidade que contém informações estruturadas sobre arquivos. Para arquivos de imagem, podemos trabalhar o metadado basicamente em dois formatos: embutido na imagem e em arquivos de texto.

O metadado embutido em imagens mais conhecido é o chamado Exif e contém informações diretamente ligadas à imagem, como câmera que foi utilizada para capturar a fotografia ou mesmo a data em que esse fato ocorreu. Diferentemente do metadado embarcado, o metadado em arquivo externo pode ser criado ou editado pelo usuário sempre que for necessário. Outra diferença é o fato de não existir um padrão fixo dos valores contidos nos metadados. Este metadado pode ser entendido como uma anotação do próprio usuário nesta imagem, dando uma conotação semântica a essas informações.

6.3.1. METADADO EMBUTIDO

Cada tipo de arquivo multimídia possui um metadado básico acoplado a ele e que dispõe de informações essenciais sobre este arquivo, como data da última modificação, formato, tamanho, dentre outros. Para arquivos de imagens não é diferente, porém, além desse metadado básico existem extensões que melhor

descrevem este arquivo. Para arquivos com formato de compactação JPEG ou TIFF é utilizado o metadado *Extended Information File* (ExIF). O Exif está contido como um cabeçalho, presente geralmente em fotografias tiradas com uma máquina digital, mas pode ser acrescentado posteriormente através de ferramentas especializadas. Este metadado armazena informações, tais como qual a fabricante da câmera que tirou a fotografia, ou até mesmo o modo em que a foto foi tirada. Estas informações normalmente são utilizadas em investigações criminais (CIPA DC-008-Translation-2012). O formato Exif foi criado pela *Japan Electronic Industry Development Association* (JEITA) e é referenciado como formato preferencial para câmeras digitais na ISO 12234-1. Este metadado é armazenado em uma área chamada de segmento de do arquivo (CIPA DC-008-Translation-2012).

Caso este metadado esteja presente em uma imagem, o mesmo pode ser acessado pelo usuário verificando-se as propriedades da imagem. Por exemplo, se o usuário utilizar o Sistema Operacional Windows é possível acessar as informações clicando com o botão direito do mouse e acessando o menu “Propriedades” e logo após buscar a aba “Detalhes”. Esta aba contem diversas informações, como dados sobre a câmera que foi utilizada para tirar a fotografia, demonstrada na Figura 2 abaixo.

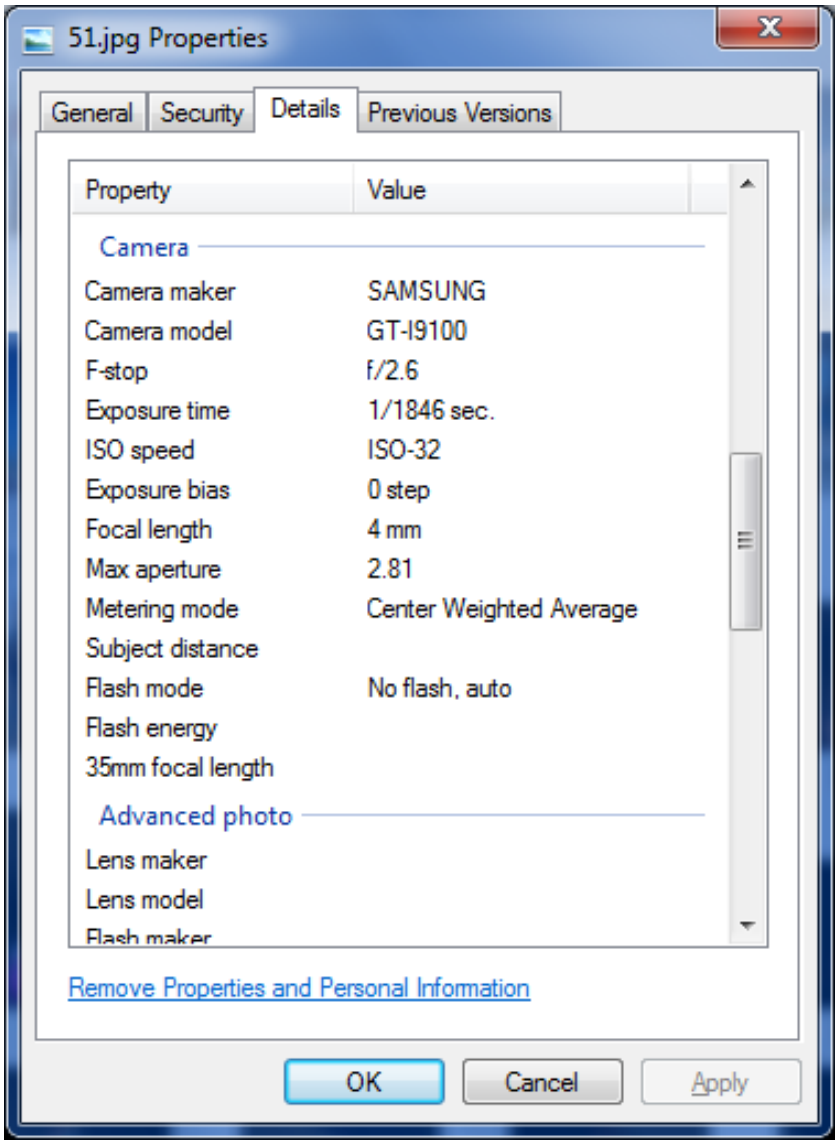

Figura 2 - Exemplo de informações contidas em um Metadado Exif

Neste projeto foi definida a utilização do metadado embarcado Exif, pois imagens no formato de compressão JPEG/JPG formam a grande parcela do montante de arquivos de imagens com metadados textuais associados. Porém, como citado no **Capítulo 6 – Trabalhos Futuros**, outros metadados podem ser acrescentados como fonte de dados.

6.3.2. METADADO TEXTUAL

A associação de metadados textuais é uma das soluções mais utilizadas para a estruturação do conteúdo de arquivos de multimídia. Em geral, estes metadados são feitos a partir da compreensão e da necessidade do usuário que está manipulando o arquivo. Embora as descrições obtidas possam ser imprecisas e incompletas, elas adicionam muito mais conhecimento e semântica ao conteúdo de um arquivo multimídia do que poderia ser obtido através das técnicas atuais de visão computacional e de reconhecimento de padrões de imagem e áudio (NETO, ROCHA, & SANTOS, 2005).

Estes arquivos textuais podem ter a formatação interna variável, dependendo da preferência do usuário. Em geral, esses arquivos podem ser arquivos planos (sem formatação) ou com marcações (com formatação). Os arquivos com formatação normalmente utilizam a marcação XML (W3C.Org), por ser um padrão de marcação difundido.

6.3.2.1. METADADO TEXTUAL PLANO

A indexação e busca em arquivos planos ou não estruturados (como arquivos de texto simples) são tratados apenas como agrupamento de palavras. “Estes dados possuem as mesmas informações que os dados tradicionais estruturados, porém sua forma é textual o que dificulta a identificação de suas características importantes” (WIVES, 1999). Dados contidos em documentos não estruturados não dispõem de qualquer esquema associado e o processo de recuperação de informações consiste em localizar

documentos relevantes, com base na entrada do usuário, como palavras-chave ou documentos de exemplo (SILBERSCHATZ & KORTH & SUDARSHAN, 2006).

Este tipo de metadado pode ser tratado como um repositório de informações vinculadas ao arquivo e que podem conter dados semânticos sobre este arquivo. Uma tecnologia que se aplica neste tipo de informação é a Mineração de Textos. A mineração de textos vem das técnicas de recuperação de informações e é diferente de um mecanismo de busca. Na busca o usuário já sabe o que quer encontrar. A tecnologia usada em mineração de textos ajuda o usuário a descobrir informações desconhecidas (ARANHA & PASSOS, 2006).

Uma possível abordagem em buscas sobre arquivos de texto plano é tratar as informações contidas nos textos como palavras agrupadas e as operações relacionadas à semântica podem ser alcançadas sobre buscas em palavras sinônimas, em partes da palavra ou mesmo em palavras similares. Como por exemplo, na busca sobre a palavra fotografia, podem ser pesquisados os seus sinônimos imagem e retrato, a palavra parcial foto ou mesmo palavras similares como fotográfica e fotografias.

6.3.2.2. METADADO TEXTUAL COM MARCAÇÃO XML

O formato XML (*eXtensible Markup Language*) é uma recomendação da *World Wide Web Consortium* (W3C.Org) para representação de informações estruturadas. Seu principal objetivo é ser uma linguagem que possa ser lida por software, e integrar-se com as demais linguagens. Sua filosofia é incorporada por vários princípios importantes, como separação do conteúdo da formatação, simplicidade e legibilidade,

tanto para humanos quanto para computadores, possibilidade de criação de tags sem limitação, dentre outros (W3C.Org).

O XML é uma linguagem de marcação. “Por linguagem de marcação, entende-se um conjunto de convenções utilizadas para a codificação de textos. Uma linguagem de marcação deve especificar que marcas são permitidas, quais são exigidas, como se deve fazer distinção entre marcas e o texto e qual o significado da marcação” (ALMEIDA, 2002). Essas marcações podem ser utilizadas como as características a serem indexadas, diminuindo o tamanho dos índices e aumentando a relevância das informações indexadas, visto que o índice irá conter a estrutura principal do arquivo e não todas as suas palavras.

Este formato também é utilizado em arquivos textuais de metadados. Esses arquivos têm como principal finalidade documentar e organizar, de forma estruturada dados das organizações, minimizando a duplicação de esforços e facilitando a manutenção desses dados (PRABHAKARAN, 2007). Podem ser aplicados em uma grande variedade de acervos, desde dados bancários, bibliotecas tradicionais e digitais e documentos multimídia.

A função das marcações XML dentro do contexto de arquivos textuais de metadados é basicamente estruturar de forma padronizada as informações contidas nesse arquivo, para uma melhor visualização e busca desses dados. Porém, mesmo que possam ser padronizadas para um grupo de arquivos, não existe uma definição de quais campos pertencem a esses arquivos de uma forma geral. Cada usuário determina que marcações existentes em cada grupo de arquivos textuais de metadados associados a imagens. Por exemplo, em uma viagem um usuário capturou 100 fotografias e posteriormente criou 100 arquivos textuais de

metadados e associou um arquivo a cada fotografia. Para descrever as imagens ele utilizou a seguinte estrutura de marcações XML, demonstrada na Figura 3:

```
<FOTOGRAFIA>
    <ARQUIVO>...</ARQUIVO>
    <CIDADE>...</CIDADE>
    <ESTADO>...</ESTADO>
    <DESCRICAO>...</DESCRICAO>
    <PESSOAS>...</PESSOAS>
    <DATA>...</DATA>
</FOTOGRAFIA>
```

Figura 3 - Exemplo de Estruturação de um Metadado Textual em XML

Sendo que as marcações devem ser utilizadas da seguinte forma:

- FOTOGRAFIA: marcação raiz de todas as outras marcações.

- ARQUIVO: descreve qual fotografia está sendo descrita.

- CIDADE: descreve qual cidade em que a fotografia foi capturada.

- ESTADO: descreve qual estado em que a fotografia foi capturada.

- DESCRICAO: descreve o momento em que a imagem foi capturada, pode ser falado sobre o clima, se foi em alguma ocasião especial ou mesmo se a imagem está relacionada à uma outra imagem.

- PESSOAS: descreve quais pessoas estavam na fotografia.

- DATA: descreve a data em que a imagem foi tirada.

7. TRABALHOS RELACIONADOS

7.1. IMAGE METADATA JPEG

Image Metadata JPEG (CPAN.ORG) é uma biblioteca utilizada basicamente para manusear os metadados de um arquivo, ou seja, editar, remover ou acrescentar. Pode ser acrescentada livremente em uma aplicação sendo desenvolvida, porém apenas identifica informações de metadados em imagens no formato JPEG, ou seja, identifica apenas metadados do tipo Exif.

O objetivo desta biblioteca é apenas manipular metadados, não incluindo os recursos de indexação e busca desses dados, nem muito menos lidar com informações da estrutura da imagem ou de arquivos de metadados textuais.

7.2. ZOOM SEARCH ENGINE COM PLUGIN IMAGEINFO

O *Zoom Search Engine* (ZOOM_WRENSOFT.COM) permite que sejam acrescentadas buscas em softwares desenvolvidos em sites, intranet ou CD/DVD. Acrescentando o *plugin ImageInfo* (WRENSOFT_PLUGINS.COM) é possível buscar informações em imagens nos formatos TIFF, JPG, GIF e PNG. Porém, esta ferramenta tem uma restrição de gratuidade, ou seja, se o site possui mais de 50 páginas a ferramenta passa a ser paga.

O objetivo desta ferramenta é indexação e busca em sites específicos. Sendo assim, não trabalha com informações de um arquivo, como os metadados e muito menos a utilização desses dados em diferentes tipos de buscas.

7.3. SADMAN SOFTWARE SEARCH

O objetivo da ferramenta *SadMan Software Search* (SADMAN.COM) é buscar por partes de palavras em determinados arquivos usando buscas com combinações lógicas (AND, OR e AND NOT) com até duas palavras, por expressões regulares. Além disso, ela permite a visualização da linha que retornou da busca definida pelo usuário, além de gerar novas buscas utilizando os resultados de uma busca anterior.

Porém, o objetivo é apenas encontrar um arquivo que contém determinada palavra e/ou palavras no seu conteúdo, não contemplando outras características como a sua data de criação ou sua coloração, ou mesmo se existe um arquivo de metadado textual associado a esse documento.

7.4. TABELA COMPARATIVA

A Figura 4 abaixo mostra a tabela comparativa entre os aplicativos citados acima e as características utilizadas como requisito de funcionamento do projeto proposto neste trabalho.

	Metadata Ingester	SadMan	Zoom Search Engine	Image Metadata JPEG
1 - Gratuito	Sim	Não, mas possui versão de demonstração	Parcialmente gratuito	Sim
2 - Precisa de *plugin*	Não	Não	Sim	Não
3 - Trabalha sobre que tipo de informação	**Arquivos de imagens, Metadados Exif e Textuais**	Arquivos de texto	Sites, arquivos de imagens e arquivos textuais	Arquivos de imagens no formato JPEG
4 - Extrai informações sobre a coloração da imagem	**Sim**	Não	Não	Não
5 - Extrai informações do metadado Exif da imagem	Sim	Não	Não	Sim
6 - Extrai informações do metadado textual	**Sim**	Não	Não	Não
7 - Gera índices	Sim	Não	Sim	Não
8 - Busca pela palavra	Sim	Sim	Sim	Não
9 - Busca pelos sinônimos da palavra	**Sim**	Não	Sim	Não
10 - Busca por palavras semelhantes da palavra	Sim	Sim	Não	Não
11 - Busca por parte da palavra	Sim	Sim	Sim	Não
12 - Busca por informações de coloração da imagem	**Sim**	Não	Não	Não
13 - Busca composta entre as 5 buscas anteriores	**Sim**	Não	Não	Não

Figura 4 - Comparação entre Funcionalidades dos Aplicativos Relacionados

Na Figura 4 é possível visualizar as diferenças e semelhanças entre os sistemas e também verificar que dentro do objetivo proposto pelo trabalho, o *Metadata Ingester* (MI) atende a todos os requisitos. A principal diferença entre o MI e os outros sistemas é a possibilidade de extração de dados de metadados textuais e informações de coloração de imagem e a combinação de buscas pela palavra, por similaridade, por sinônimos e por substring sobre essas informações.

8. METADATA INGESTER

8.1. VISÃO GERAL

O Metadata Ingester (MI) é um sistema de busca e indexação de arquivos de metadados no formato de texto plano e em formato XML que utiliza a biblioteca de recuperação de informação Apache Lucene (LUCENE_APACHE.ORG) como base. O MI executa uma prévia interpretação dos arquivos textuais e o resultado é utilizado tanto pelo Lucene quanto pelo próprio MI para a criação dos arquivos de indexação. Finalizado o processo de criação de índices o MI recebe buscas do usuário, faz uma busca prévia nos seus arquivos de índices, formata os dados e envia ao Lucene para a recuperação da informação nos índices criados.

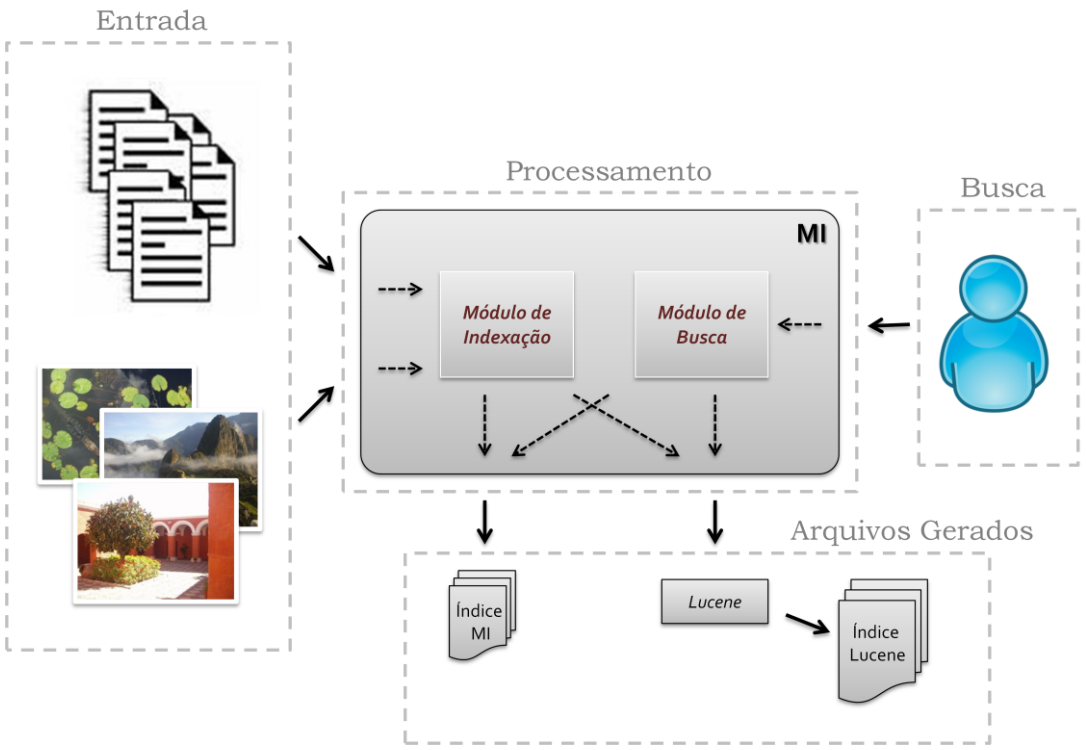

Figura 5 - Visão Geral Metadata Ingester

Na Figura 5 está representada a visão geral do Metadata Ingester (MI). Pode-se subdividir a imagem em 4 (quatro) áreas distintas:

- Entrada: os arquivos de metadados textuais e de imagens são fornecidos ao MI.

- Processamento: os índices serão gerados e as buscas serão executadas.

- Busca: o usuário configura e inicia a execução das buscas.

- Arquivos Gerados: os arquivos de índices serão salvos em memória para que possam ser utilizados nas buscas dos usuários.

8.2. MODULO DE INDEXAÇÃO

Este módulo trabalha com 4 etapas distintas: na *Primeira Etapa* são lidos e interpretados os arquivos de metadados textuais. Na Segunda Etapa são extraídos os metadados embarcados na imagem. Na Terceira Etapa os dados sobre a coloração da imagem são coletados, e na Quarta Etapa todos esses dados são usados para a geração dos arquivos de índices.

8.2.1. PRIMEIRA ETAPA

Nesta etapa inicial, o MI tem o objetivo de gerar dois grupos de índices: Índices sobre os metadados textuais planos e sobre os metadados textuais com marcação XML.

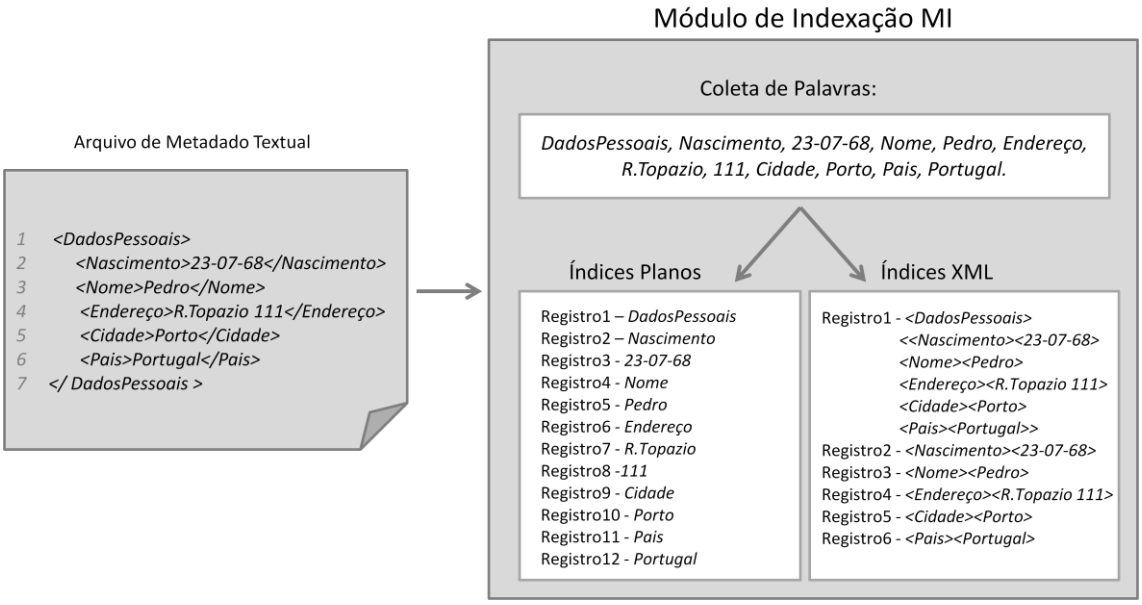

Figura 6 - Exemplo de Coleta de Informações de Metadados Textuais

Na Figura 6 pode-se visualizar o comportamento do MI em relação aos metadados textuais. A partir do grupo de arquivos textuais fornecido pelo usuário, o MI cria a indexação dos arquivos primeiramente como se esses arquivos não possuíssem marcação, e posteriormente identifica quais desses arquivos possuem a marcação XML e para esses arquivos acrescenta o segundo grupo de índices utilizando as marcações. Sendo assim o usuário não precisa saber, dentre os arquivos textuais, quais possuem e quais não possuem formatação. O próprio MI identifica os arquivos que podem ser também indexados como arquivos XML.

8.2.2. SEGUNDA ETAPA

Nesta etapa os dados referentes ao metadado da imagem são coletados. Para este processo, foi utilizada a biblioteca Apache Sanselan (SANSELAN.APACHE) que tem o objetivo de extrair metadados de imagens. Esta biblioteca possui recursos para a recuperação de valores do metadado embutido no padrão EXIF e também para a recuperação de informações referente à estrutura da imagem, como tamanho, largura e altura, por exemplo. Uma observação importante é que nem todas as imagens possuem o metadado EXIF, porém todas as imagens possuem as informações referente à estrutura da imagem. Sendo assim, se em uma busca de imagens houver a solicitação de uma informação que estava contida no metadado Exif, apenas as imagens que possuem esse metadado atendem os requisitos da busca e retornam como resultados.

As informações retiradas do metadado EXIF são as seguintes:

- ***Data de Criação***: Representa a data em que a imagem foi tirada com a câmera digital.

- ***Câmera utilizada***: Representa a fabricante da câmera digital utilizada.

- ***Resolução X e Y***: Representa a resolução em relação à altura e largura da imagem.

As informações retiradas das informações sobre a estruturada imagem são:

- ***Tamanho***: Representa o tamanho em disco do arquivo em Kbytes

- ***Última modificação***: Representa a data da última modificação sofrida pelo arquivo

- ***Largura***: Representa a largura da imagem em pixels

- ***Altura***: representa a altura da imagem em pixels

- ***Formato***: Representa o formato de compressão da imagem (JPEG, JPG, GIF, PNG, TIFF, etc)

- ***Modelo de Cores***: Representa o padrão de cores utilizado na imagem (RGB, CMYK, Tons de Cinza, etc).

8.2.3. TERCEIRA ETAPA

Esta etapa consiste em obter informações da coloração da imagem. Isto significa extrair informações como cor predominante e porcentagem de cada cor da imagem.

A tomada de cores de uma imagem é uma decisão difícil, pois trata-se de um universo de 24 milhões analisadas a cada pixel da imagem, ou seja, em uma imagem de 1024x768 a quantidade de iterações seria de 18.874.368.000.000. Esse processo geraria um custo computacional muito elevado, incrementando excessivamente o tempo de processamento ao indexar um grupo de imagens e dependendo do hardware utilizado seria até mesmo impossível concluir o processamento.

Para resolver o problema de extração de coloração de imagem, desenvolveu-se o um recurso [1] para ser embarcado ao MI. Este recurso baseia-se em uma técnica bastante difundida em computação 3D para reconhecimento de padrões, o algoritmo *K-Nearest Neighbor Algorithm* (BREMNER, et al., 2005, p. 594), para determinar a cor de cada pixel dentro de um conjunto de cores pré-determinado partindo do código RBG da cor do pixel em questão. O algoritmo *K-Nearest Neighbor Algorithm* (KNN) disponibiliza

uma importante ferramenta de classificação de informação para reconhecimento de classes de objetos no domínio de reconhecimento de padrões (SINGH, HADDON, & MARKOU, 1999).

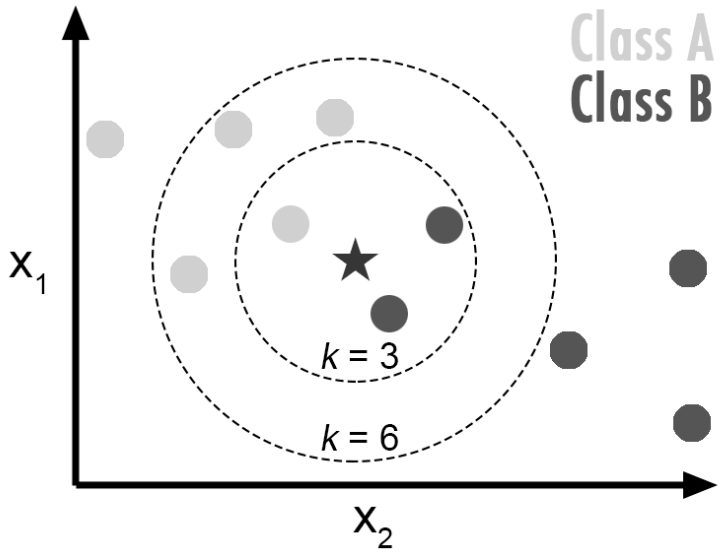

Figura 7 - Exemplo de utilização do Algoritmo KNN

Um exemplo de utilização do Algoritmo KNN é mostrado na Figura 7. Neste conjunto de Treinamento são utilizadas 2 (duas) classes que possuem os seguintes rótulos: “Class A” representada pelos círculos na coloração de cinza claro e “Class B” representada pelos círculos na coloração de cinza escuro. O valor de k representa a quantidade de objetos que estão presentes no raio determinado. E foram utilizadas (2) duas características das classes denominadas de “x1” e “x2”. No exemplo dado a definição do rótulo de classe para o objeto a ser pesquisado (representado pela estrela) se dá pela classe que possuem a maior quantidade de objetos pertencentes aquele raio. Como no exemplo, no raio menor o objeto receberia o rótulo de classe como “Class B” e no raio maior receberia “Class A”.

O Algoritmo KNN que serviu como base possui as seguintes etapas:

Treinamento: Os exemplos de treinamento são vetores multidimensionais, cada um com um rótulo de classe. A etapa consiste em armazenar os vetores de características e rótulos de classes das amostras de treinamento.

Classificação: Nesta etapa, k é uma constante definida pelo usuário, e um vetor não rotulado (que é o ponto de teste) é classificado atribuindo-se um rótulo que deve ser igual ao mais frequente dentre os exemplos de treinamento mais próximos desse ponto de teste.

Para o desenvolvimento do recurso utilizado neste trabalho, foram feitas algumas adaptações no algoritmo base, resultando na seguinte abordagem:

- Os exemplos de treinamento foram substituídos por cores, cada rótulo contem uma cor da lista de cores e sua característica contém o código RGB da respectiva cor.

- Cada componente de cor que compõe o RGB é uma coordenada no espaço, ou seja, o componente *Red* é a coordenada x, o componente *Green* é a coordenada y e o componente *Blue* é a coordenada z.

- O espaço de trabalho é um ambiente 3D, com as cores plotadas nesse universo, sendo suas coordenadas definidas pelos componentes de cor R, G e B que se torna, respectivamente, x, y e z.

- A definição de qual é a cor mais próxima da cor sendo pesquisada é determinada através do cálculo da distância entre dois pontos no espaço R^3. Seja a cor A com RGB(r_1, g_1, b_1) e a cor B com RGB(r_2, g_2, b_2) a distância entre essas duas cores é determinada pela fórmula de *Distância Euclediana* para o cálculo de distâncias em um

[1] O recurso de extração da característica de coloração da imagem foi cortesia de um grande amigo e companheiro o Engenheiro de Software, Julio Cesar Lopes Marques.

espaço tridimensional: $\sqrt{(r_1 - r_2)^2 + (g_1 - g_2)^2 + (b_1 - b_2)^2} = distância.$ Sendo assim o ponto que estiver mais próximo do ponto sendo pesquisado representa a cor que dará o seu nome à cor sendo pesquisada.

- O conjunto finito de cores principais a serem detectadas em cada pixel da imagem ficou definido como: ***Branco***, ***Preto***, ***Cinza***, ***Azul***, ***Verde***, ***Amarelo***, ***Vermelho***, ***Rosa***, ***Lilás***, ***Laranja*** e ***Marrom***.

- A quantidade de cores a serem descobertas é determinada pela quantidade de pixels da imagem, visto que a cada pixel é determinada qual cor que ele representa.

Para auxiliar a aplicação do algoritmo foi criada uma lista com diversos tons pertencentes às cores que compõe o conjunto base de cores e seus respectivos códigos RGB. Sendo assim com mais pontos no espaço, a busca pela cor mais próxima se torna mais precisa.

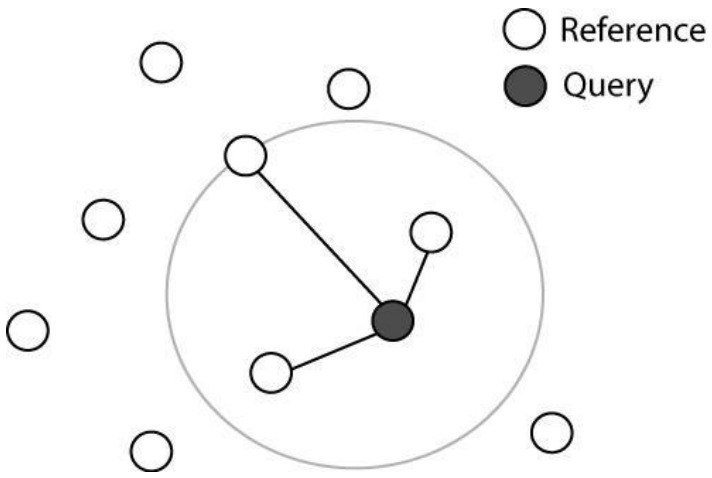

Figura 8 - Exemplo de utilização do Algoritmo KNN

Na Figura 8 é demonstrado um exemplo de utilização do Algoritmo KNN visando a abordagem utilizada neste projeto. Onde o objeto "*Query*" representa a cor à ser pesquisada e os objetos "*Reference*" representam as cores que foram definidas como base. Após o cálculo da distância Euclediana será descoberta a cor que possui a menor distância em relação à cor pesquisada e assim o nome da cor sendo pesquisada será igual ao nome da cor de menor distância.

O retorno da aplicação do algoritmo é uma lista com as cores pré-determinadas e a suas respectivas porcentagens encontradas na imagem. A cor predominante é determinada como a cor que possui a maior porcentagem de presença na imagem.

8.2.4. QUARTA ETAPA

Nesta etapa a criação propriamente dita dos arquivos de índices é feita com base nas informações coletadas nas etapas anteriores. Porém, para que os próximos passos desta etapa possam ser entendidos por completo, é necessária a explicação do processo de geração de índices pelo Lucene.

A implementação do processo de indexação do Lucene conta com um conjunto de objetos, cada um com uma função específica. Os objetos principais são: *Document* e *Field.* O objeto *Document* é responsável por representar cada arquivo a ser indexado e é composto por objetos *Field*. O objeto Field é responsável por representar cada informação sobre o arquivo a ser indexado e é composto por dois atributos *Name* e *Value* (GOSPODNETIC & HATCHER, 2005, p. 27). A Figura 9 representa a árvore de estruturação entre os objetos *Document* e *Field*.

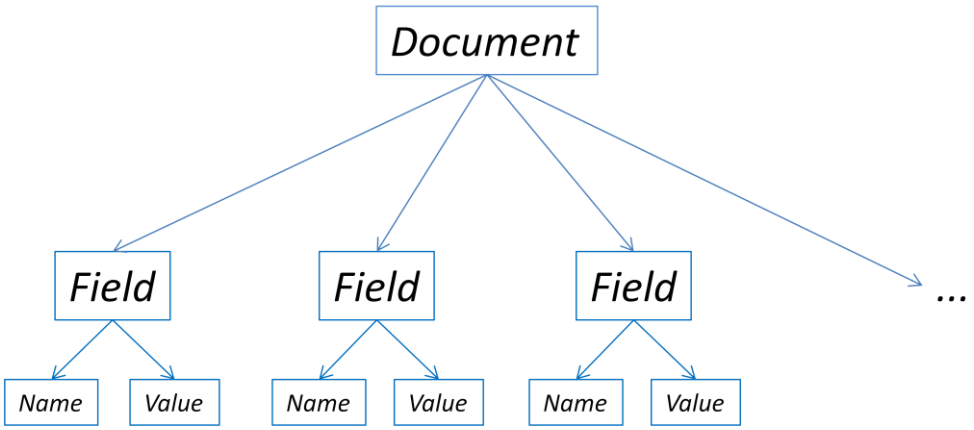

Figura 9 - Estruturação dos Objetos Document e Field

Internamente ao objeto *Field*, os atributos *Name* e *Value* representam, respectivamente, uma chave, que identifica apenas uma instância do objeto *Field*, e um valor, que representa uma informação do arquivo indexado. Caso exista mais de uma instância do objeto *Field* em uma instância de um objeto *Document* com o mesmo valor para o atributo *Name*, esses objetos vão ser unificados e o seus valores concatenados, a fim de manter único o valor do atributo *Name* entre as instâncias de objetos *Field*. A Figura 10 simboliza um exemplo de um arquivo de texto a ser indexado pelo Lucene.

```
<TITULO>Criando índices com o Lucene</TITULO>
<AUTOR>João da Silva</AUTOR>
<AUTOR>Maria de Lourdes Souza</AUTOR>
<AUTOR>José Alencar</AUTOR>
```

Figura 10 - Exemplo de arquivo de texto a ser indexado

Para esse arquivo é executada a seguinte sequencia de criação de objetos demonstrada pela Figura 11.

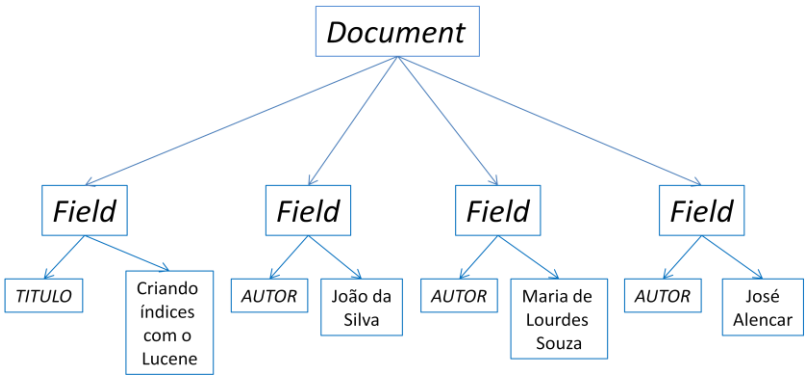

Figura 11 - Exemplo de sequencia de criação de objetos para a indexação

Porém, os objetos que o Lucene realmente cria, são representados pela Figura 12.

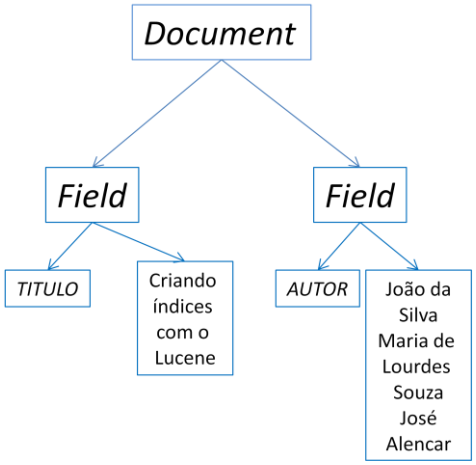

Figura 12 - Exemplo de objetos criados pelo Lucene

Com essas informações sobre a geração de índices pelo Lucene, é possível o entendimento da necessidade de se determinar nomes para cada objeto *Field* criado

no processo de geração de índices. Para cada uma das informações coletadas nas 3 etapas anteriores é utilizada uma política de nomeação de objetos *Field*.

Nas informações coletadas na Primeira Etapa vindas dos arquivos de metadado textual plano, o objeto *Document* é composto de um objeto *Field* com o nome de *Path* e o valor com o caminho do arquivo em disco e um objeto *Field* com o nome *Content* e o valor com as palavras pertencentes ao arquivo, e das informações vindas dos arquivos de metadado textual em XML. O objeto *Document* é composto por um objeto Field com o nome de *Path* e diversos outros objetos *Field*, um para cada marcação XML que o arquivo de metadado contém.

Para o mesmo exemplo de arquivo de metadado textual demonstrado na Figura 10, são criados os seguintes objetos representados pela Figura 13 e Figura 14, para indexação, analisando o arquivo como texto plano e texto em XML, respectivamente.

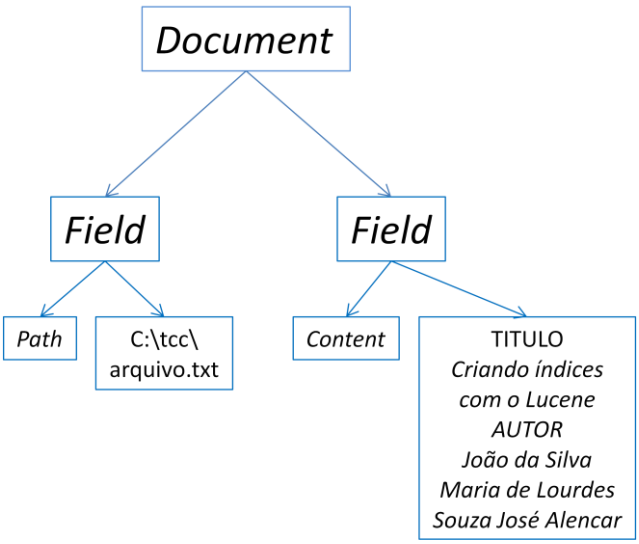

Figura 13 - Exemplo de criação de índice em arquivo plano

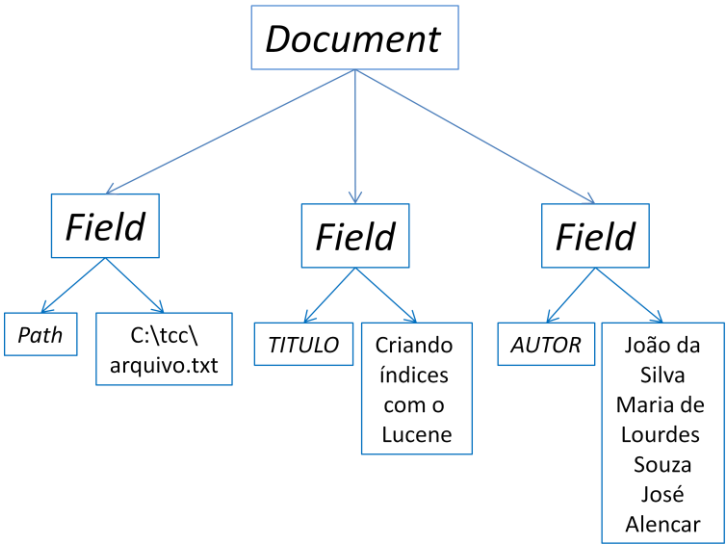

Figura 14 - Exemplo de criação de índice em arquivo XML

Para que esses objetos Field possam ser acessados no momento da busca, é necessário o conhecimento dos nomes exatos atribuídos aos objetos Field, para que possa ser recuperada a informação do valor desse objeto e consequentemente a comparação com o valor que o usuário solicitou na busca. Para os arquivos de metadados planos, os nomes das duas instâncias do objeto Field são previamente definidos como *Path* e *Content*, porém nos arquivos de metadados com marcação XML, apenas 1 nome é conhecido, o *Path*, pois todas as instâncias restantes recebem o nome conforme a marcação que ela representa. Para que esses nomes não se percam, uma estrutura de índice auxiliar é gerada pelo MI, para que seja viável o retorno dos resultados de forma segura e precisa. A possibilidade de esse segundo grupo de índices reduzir a eficiência da busca foi analisada, mas houve a detecção de que quando há marcações nos arquivos de metadados textuais, as mesmas se repetem entre os arquivos. Então, mesmo que tenhamos uma quantidade extensa de arquivos, a quantidade de

marcações que devem ser armazenadas é restrita. Essa lista é persistida juntamente com os arquivos de índices gerados pelo Lucene e na inicialização do MI. Esta lista é carregada em memória facilitando o acesso à suas informações.

Nas informações coletadas na Segunda Etapa, vindas dos metadados embarcados básicos e *Exif*, o objeto *Document* é composto de um objeto Field com o nome de Path e valor como o caminho do arquivo e um objeto Field para cada uma das informações citadas no item **5.2.2. Segunda Etapa**, deste mesmo capítulo. Já que não existe uma variabilidade dos nomes dos objetos Field, não é necessária uma estrutura auxiliar para armazenar os nomes utilizados. Na Figura 15 é mostrado um exemplo de criação de objetos para a geração de índices.

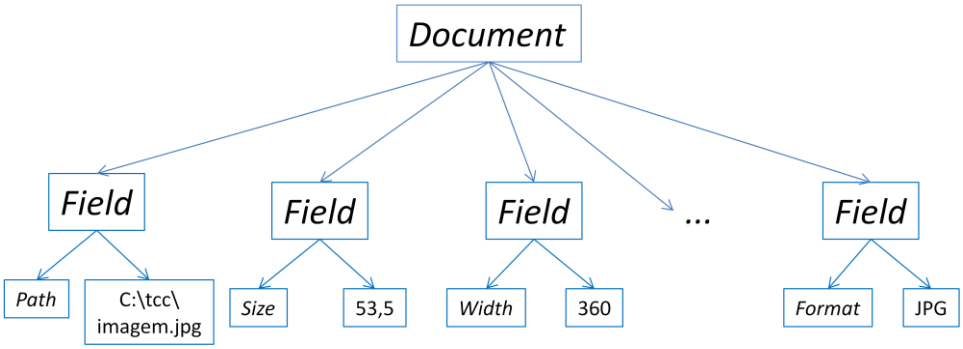

Figura 15 - Exemplo de criação de índice sobre os metadados embutidos

Nas informações coletadas na Terceira Etapa, vindas das informações de coloração das imagens, o objeto *Document* é composto de um objeto Field com o nome de Path e valor como o caminho do arquivo e um objeto para cada uma das cores definidas no item **5.2.3. Terceira Etapa**, deste mesmo capítulo e um último objeto Field com o nome de *PredominantColor* e valor com o nome da cor predominante na imagem. Assim como para as informações da Segunda Etapa, nesta etapa não é necessária um

índice auxiliar para manter os nomes definidos, visto que também já estão definidos. A Figura 16 representa um exemplo de criação de objetos para a geração de índices:

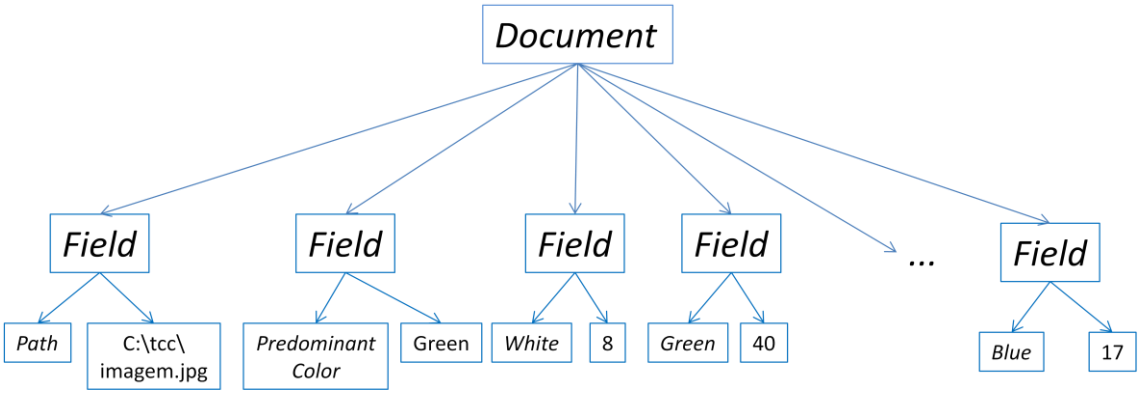

Figura 16 - Exemplo de criação de índice sobre a coloração da imagem

8.3. MÓDULO DE BUSCA

O módulo de busca trata a combinação de 4 (quatro) tipos distintos de busca (simples, que retorna palavras estruturalmente iguais à que foi fornecida, substring, que retorna palavras que contenham a palavra fornecida, similaridade, que retorna palavras estruturalmente semelhantes e sinônimo, que retorna palavras que possuem o mesmo sentido da que foi fornecida) com 3 (três) diferentes fontes de dados (arquivos de metadados textuais, metadados embutidos e informações de coloração), permitindo ao usuário uma grande liberdade na criação de buscas.

O princípio básico da busca tradicional é: a partir de um valor determinado pelo usuário descobrir que informações devem ser retornadas como resultado. No caso do projeto proposto as informações a serem retornadas ao usuário estão armazenadas em forma de índices gerados pelo Lucene e pelo MI, e, portanto, para recuperá-las deve-se pesquisar nesses índices.

8.3.1. FONTES DE DADOS

Assim como na indexação, dentro do módulo de busca são tratados 3 tipos de dados de diferentes fontes: arquivos de metadados textuais, metadados embarcados e informações de coloração.

8.3.1.1. METADADO TEXTUAL

A busca em informações de arquivos de metadado textual pode utilizar arquivos um dos arquivos de índices, apenas os gerados pelo Lucene ou apenas os gerados pelo MI, ou mesmo ambos, dependendo do formato de metadados textual em que a busca se baseia. Caso a busca aconteça sobre os arquivos de metadados planos, apenas os índices gerados pelo Lucene deve ser utilizados, assim como demonstrado na Figura 17.

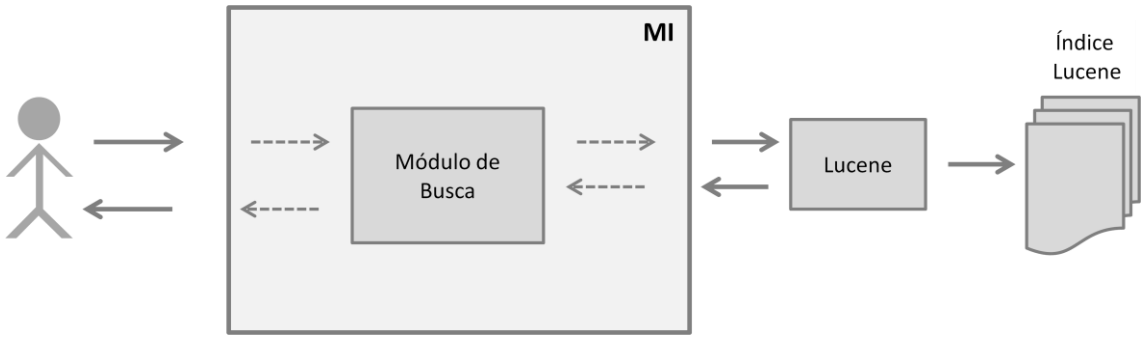

Figura 17 - Busca sobre Metadados Textuais Planos

Caso a busca aconteça sobre os arquivos de metadados textuais com marcação XML, os arquivos de índices gerados tanto pelo Lucene quanto pelo MI são utilizados, assim como evidenciado na Figura 18.

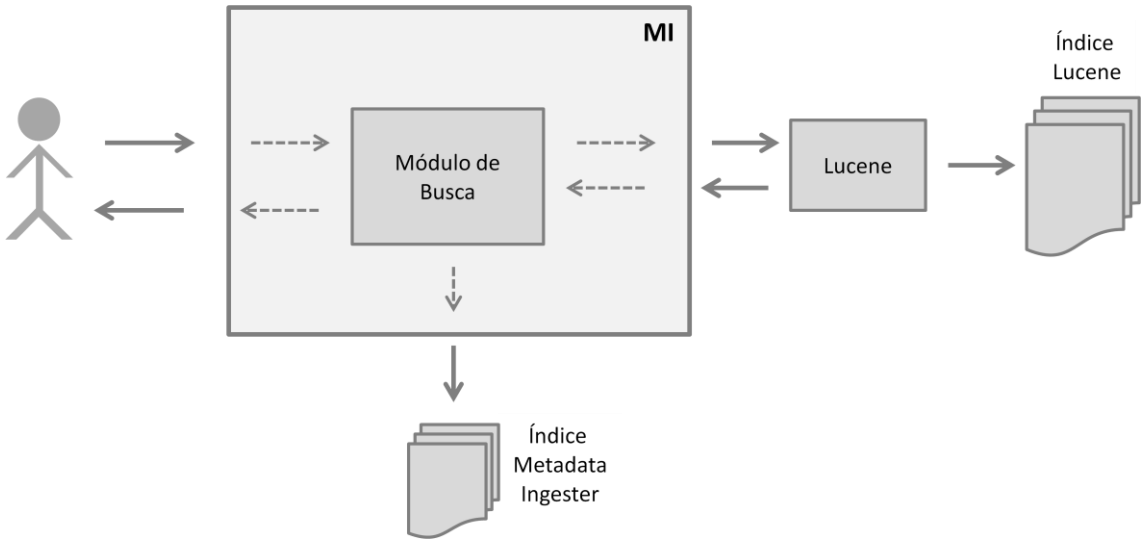

Figura 18 - Busca sobre Metadados Textuais com Marcação XML

Na busca associada aos índices gerados pelos arquivos de metadados textuais simples, ou seja, em formato de texto plano, é necessário o fornecimento pelo usuário apenas da palavra-chave a ser pesquisada. Como explicado no item **5.2.4. Quarta Etapa**, neste capítulo, a busca nos índices do Lucene necessitam do nome do objeto *Field* que foi indexado. Como o MI identificou que a solicitação de busca é sobre metadados textuais planos, a busca nos índices contém o nome com a palavra *Content* e o valor que foi solicitado pelo usuário. Após a configuração da busca, a mesma é enviada ao Lucene, que executa a busca nos arquivos de índices de metadado textuais planos. O resultado desse processo é a lista com os valores de todos os campos *Path* (que possui o caminho em disco do arquivo) dos objetos *Document* que contém a palavra pesquisada pelo usuário. Esta lista é mostrada ao usuário pelo MI.

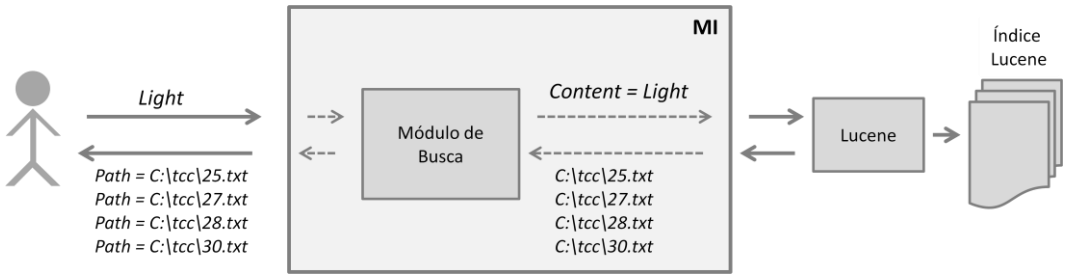

Figura 19 - Exemplo de Busca sobre Metadados Textuais Planas

A Figura 19 exemplifica uma busca pela palavra *Light* sobre os índices gerados pelo Lucene. O MI identifica que é uma busca sobre metadados textuais planos e configura a busca com o nome do objeto *Field* como *Content* e o valor *Light*. Assim o Lucene é capaz de retornar o resultado com os caminhos dos arquivos que contém essa palavra.

Para buscas associadas aos índices gerados pelos metadados textuais em formato XML além da palavra a ser pesquisada, existe a necessidade de sua respectiva marcação XML. A marcação assume o papel do nome do objeto *Field* da busca e a palavra assume o papel de valor desse objeto. O nome da marcação fornecido pelo usuário pode não ser exatamente o que foi utilizado para a geração dos índices do Lucene. Então, o MI executa uma busca inicial no arquivo de índice criado por ele, com o objetivo de fornecer ao usuário quais marcações foram utilizadas na indexação. Esta busca pode também ser executadas de 4 formas, assim como para busca de palavras: Simples, Similaridade, Sinônimos e Substring.

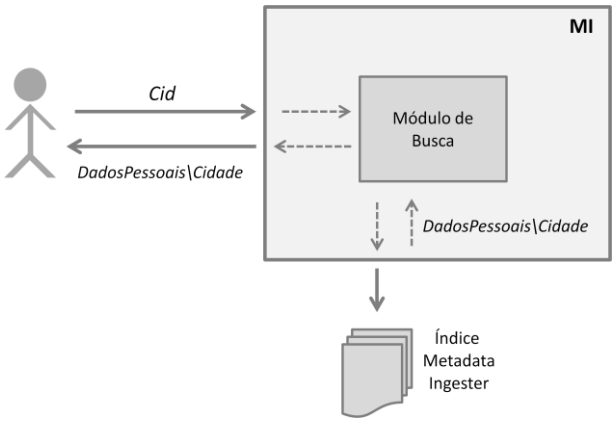

Figura 20 - Exemplo de Busca sobre o Índice MI

Na Figura 20 é exemplificado o processo de busca sobre o índice gerado pelo MI com as informações das marcações XML. Para que a busca seja executada nos índices do Lucene, é necessária recuperação do nome completo da chave que a informação foi indexada. Caso o usuário não saiba como o nome foi indexado ele poderá executar uma busca simples e/ou por similaridade e/ou por substring e/ou por sinônimo. Dentre as marcações resultantes da busca o usuário poderá selecionar qual que realmente é a que ele procura.

Após esse processo, o MI já possui o nome do objeto Field indexado pelo Lucene, então é possível configurar a busca com o nome recuperado pelo MI e o valor com a palavra solicitada pelo usuário. Como é exemplificado na Figura 21, a busca foi configurada com o nome “DadosPessoais\Cidade” e com o valor “Porto” e recebeu os seguinte endereços de arquivos que contém esses dados “C:\tcc\25.xml”, “C:\tcc\35.xml”, “C:\tcc\38.xml”, “C:\tcc\41.xml”.

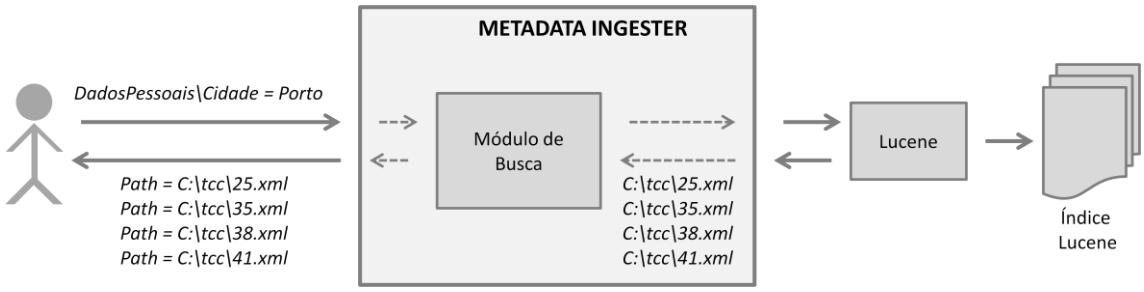

Figura 21 - Exemplo de Busca sobre Metadados em XML

8.3.1.2. METADADOS EMBUTIDOS

A busca em informações indexadas a partir do Metadado da imagem acontece primeiramente localizando-se o nome da chave referente à característica que o usuário solicitou na busca dentro do arquivo de constantes criado anteriormente. Após a descoberta do nome da chave, o MI configura a busca com o nome da chave e o valor que o usuário determinou.

Dependendo da característica a ser buscada é possível trabalhar, não apenas com busca de palavras, mas também com intervalos de valores. Tanto em intervalos de números quanto de intervalos envolvendo datas, utilizando os recursos do Lucene chamados "*NumericRangeQuery*" e "*TermRangeFilter*", respectivamente.

8.3.1.3. INFORMAÇÕES DE COLORAÇÃO

Assim como no tipo de dados de Metadado a busca inicia-se a partir da localização da palavra que representa a chave indexada no índice do Lucene. Com o nome da chave localizada, a busca é configurada com o nome da chave e o valor que foi fornecido pelo usuário.

Nesse tipo de dado também é aplicado o recurso de consultas com intervalos de valores, para que possamos disponibilizar ao usuário a opção de buscar intervalos do tipo: buscar imagens que possuem entre 20% e 40% de preto.

8.3.2. TIPOS DE BUSCA

O MI permite 4 tipos de busca que podem ser executados separadamente ou combinados, resultando em diversas configurações de busca. Sendo que elas:

- Busca simples - retorna palavras estruturalmente iguais à que foi fornecida

- Busca por similaridade - retorna palavras estruturalmente semelhantes

- Busca por sinônimos - retorna palavras que possuem o mesmo sentido da que foi fornecida

- Busca por substring - retorna palavras que contenham a palavra fornecida.

Além dos tipos de buscas diferenciados, o MI permite criar combinações entre os tipos de dados.

8.3.2.1. SIMPLES

Neste tipo de busca, o Lucene é configurado para buscar o valor exato fornecido pelo usuário, juntamente com o nome da chave. Nos resultados, apenas deve conter os arquivos que possuem exatamente a palavra que foi buscada. Ela nem mesmo retorna

palavras compostas pela palavra pesquisada, se elas estiverem concatenadas ou separadas por hífen de uma outra palavra.

No caso de busca no índice gerado pelo MI com as marcações XML indexadas, a procura pela palavra é uma simples varredura na lista de marcações, que é carregada em memória na inicialização do sistema, até encontrar a igualdade entre a palavra buscada e a marcação na lista.

8.3.2.2. SUBSTRING

Esta busca utiliza o conceito de substring. Este conceito, em programação, é muito utilizado e indica que se uma palavra é substring de uma outra, a primeira é parte da segunda. Como por exemplo, na palavra “girassol” a palavra “sol” ou a palavra “gira” é substring de “girassol”. Porém, não existe a necessidade de a substring ter um significado, por exemplo, o conjunto de caracteres “rasso” também é substring de “girassol”.

Para a aplicação desse conceito foi utilizado o recurso de *WildCardQuery* do Lucene que permite que seja informada a palavra a ser buscada através de uma expressão. No MI a busca foi configurada com o valor de “* palavra *” que determina ao Lucene que a palavra a ser buscada pode vir concatenada por qualquer palavra, ou até mesmo nenhuma, tanto antes quanto depois. Com isso é possível que qualquer palavra que contenha a substring buscada seja retornada como resultado da busca.

Na Figura 20, é mostrado um exemplo prático da utilização do recurso de *WildcardQuery*.

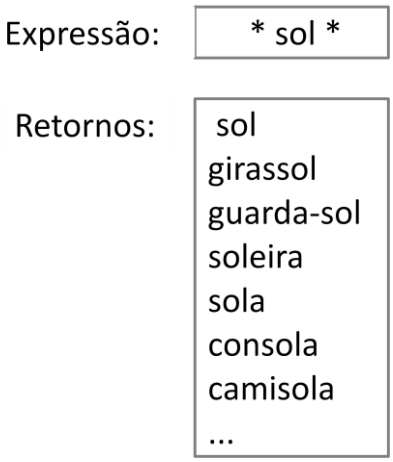

Figura 22 - Exemplo de retorno de uma busca por substring

Na Figura acima a palavra utilizada como base é “sol”. Os asteriscos, presentes antes e depois da palavra base, representam a possibilidade de qualquer conjunto de caracteres seja acrescido à palavra base. Sendo assim, por exemplo, a palavra “camisola” é uma palavra válida de retorno, pois antes da palavra base está o conjunto de caracteres “cami” e depois está “a”. A própria palavra “sol” é um retorno válido, visto que os asteriscos podem representar que nenhum conjunto de caracteres é concatenado à palavra base.

Na busca de marcações por substring no índice gerado pelo MI é utilizada a função nativa do próprio Java chamada “*indexOf*” (JavaSE String). Esta função retorna a posição da substring na palavra e caso a substring não pertença à palavra o valor retornado é -1. Sendo assim, o MI fornece ao usuário todos os marcadores que o resultado da função executada com a palavra dada pelo usuário tenha sido diferente de -1.

8.3.2.3. SIMILARIDADE

Neste projeto, a identificação de semelhança entre palavras através da similaridade é feita através do recurso *FuzzyQuery* do Lucene. Esta ferramenta permite que determinemos o índice de similaridade entre a palavra pesquisada e as palavras que estão nos índices criados pelo Lucene. A similaridade é calculada através do algoritmo de distância de *Leveshtein* (GOSPODNETIC & HATCHER, 2005), que determina o quão similar duas palavras são através da seguinte fórmula:

$$1 - \frac{distance}{\min\left(textlen, targetlen\right)}$$

Entendendo a fórmula:

- *distance* = Representa o valor de edição da palavra para que se tornasse a palavra alvo. Estas operações ganham o nome de "*Edit Distance"* e os pesos de cada operação são calculados da seguinte forma: inserções e remoções de letras tem o mesmo peso e substituições de letras tem duas vezes o peso de uma inserção. Por exemplo, a distância entre as palavras "caixa" e "abaixa" é resulta em 3 e é medida da seguinte forma:

    - caixa para **a**caixa -> inserção da letra a -> custo 1

    - acaixa para a**b**aixa -> substituição da letra c pela b -> custo 2

- *min(textlen, targetlen)* = Resulta no menor tamanho entre textlen (tamanho da palavra pesquisada) e targetlen (tamanho da palavra alvo da pesquisa).

O recurso acima foi utilizado para a busca nos índices gerados pelo Lucene. Já nas buscas no índice gerado pelo MI foi utilizada a biblioteca *SimMetrics*

(SIMMETRICS.ORG) desenvolvida na Universidade de Sheffield (SHEFFIELD.UK), UK. Esta biblioteca implementa diversos algoritmos de similaridade, como o de *Leveshtein* citado acima, utilizado para verificar a similaridade entre a palavra fornecida pelo usuário e a lista de marcações dos arquivos de metadados textuais em formato XML.

8.3.2.4. SINÔNIMO

Sinônimos são palavras que são estruturalmente diferentes, porém possuem a mesma semântica associada. Para que fosse possível utilizar essa opção como uma das buscas, foi incorporada a biblioteca *Apache Lucene-WordNet* (LUCENE_WORDNET.APACHE) que utiliza a base Prolog *WordNet* (WORDNET.EDU) para definir quais palavras são sinônimos da palavra a ser pesquisada. *WordNet* é uma extensa base de dados léxica de palavras na língua Inglesa desenvolvida na *Princeton University* (PRINCETON.EDU) e que é disponível gratuitamente. Após obter a lista de sinônimos, uma busca simples para cada item da lista é executada e os resultados são mostrados ao usuário juntamente aos resultados da busca com a palavra base fornecida pelo mesmo.

Para as buscas em sinônimos de marcações XML, o mesmo processo é executado, ou seja, dada a palavra fornecida pelo usuário, uma lista de sinônimos é criada. Caso esta palavra possua sinônimos, a cada item na lista uma busca simples é executada e os retornos das buscas são mostrados ao usuário para que ele selecione a que melhor se adeque à marcação que ele deseja.

8.3.3. COMBINAÇÃO ENTRE FONTES DE DADOS E TIPOS DE BUSCA

Como foi explicado acima, foram desenvolvidos neste projeto diversos tipos de busca que tratam diferentes fontes de dados. A grande questão neste ponto do projeto é: Como combinar os resultados dessas diferentes buscas em diferentes dados?

Em primeiro lugar é relembrada a diferença entre os índices de metadados textuais com e sem marcação. O índice gerado a partir de metadados textuais com marcação XML está incluído no índice de metadados textuais sem marcação, visto que o processo de geração de índices acontece da seguinte forma: inicialmente é gerado o índice dos arquivos de metadados textuais como se eles não possuíssem marcação alguma, apenas um aglomerado de palavras, e posteriormente é verificado se esse metadado textual possui a marcação XML. Em caso positivo esse arquivo é acrescentado ao índice de metadado textual com marcação.

Sendo assim, não faz sentido combinar esse dois índices de metadados textuais, já que tudo que contiver no índice de metadados textuais com marcação XML também está contido no índice de metadados textuais planas. Por esse motivo, quando é configurada a busca para procurar em ambos os índices de metadados textuais (com marcação XML e plano) o resultado ao usuário não deve ser combinado, visto que o resultado dessa combinação é o mesmo que se fosse executada a busca apenas nos arquivos de metadados textuais planos. Obviamente, esse caso só acontece caso o usuário selecione a fonte de dados de metadados textuais e também solicite ambos os formatos de metadados textuais.

A exemplo dos arquivos de metadados textuais, apenas ocorre combinações entre resultados de busca dentre os itens configurados pelo usuário. Isso significa que só são combinados os resultados dos tipos de buscas (simples, sinônimos, substring ou similaridade) sobre as fontes de dados (metadado externo textual, metadado embarcado e informações sobre imagem) que o usuário selecionou. Por exemplo: Em uma busca por similaridade e sinônimos, em arquivos de metadados textuais planas pela palavra “ground” e com cor predominante “verde”, são combinados os resultados da busca por similaridade nos índices de metadados textuais planos, os resultados da busca de todos os sinônimos nos índices de metadados textuais planos e os resultados da busca simples nos índices de características da imagem por cor predominante igual à “verde”.

Em uma busca, uma característica pode ser mais importante que outra, ou seja, uma característica é mais relevante que outra. Em um caso prático, a relevância acontece, por exemplo, caso o usuário deseje que a característica x apareça com certeza em todos os resultados, mas se a característica y estiver presente é um resultado mais próximo do que ele estava esperando. Além do desejo do usuário, também se deve levar em consideração que informações de resultados de consultas exatas estão sendo combinadas com consultas aproximadas. Em outras palavras, as buscas que acontecem sobre as características extraídas da estrutura da imagem (como tamanho, padrão de cores, formato de compressão, resolução, dentre outros) possuem retornos exatos, ou seja, todas as imagens que estão no grupo de resultados contêm as características solicitadas. Já os retornos das buscas nos índices de metadados textuais retornam resultados aproximados, já que essas buscas trabalham com a semelhança entre os dados configurados na busca e as informações indexadas. Sendo assim, quando existe

a consulta tanto sobre os arquivos de metadados textuais quanto nos dados estruturais da imagem, os resultados dessa última devem ser usados para tornar mais precisos os resultados da primeira. Por exemplo, o usuário configurou uma busca por sinônimos da palavra “ambiente” e que a imagem tenha tamanho de 125Kb. Para cada item da lista de resultados da busca sobre os arquivos de metadados textuais, é feita uma varredura na lista de resultados da busca sobre os metadados embarcados. Caso o item esteja em ambas as listas, os valores de *score* são somados. Após todos os itens serem verificados, as duas listas são mescladas e os resultados são reorganizados pelo valor do *score*. Neste momento, estão nas primeiras posições os resultados que possuem uma probabilidade maior de ser o resultado desejado pelo usuário.

Para executar a reorganização dos resultados conforme a relevância das pesquisas, foi desenvolvido um esquema de definição de relevâncias, onde o usuário pode definir conforme lhe for melhor adequado. Esses valores são aplicados aos resultados utilizando a fórmula de média ponderada aos Scores (pontuação de relevância desse resultado dada pelo próprio Lucene após uma consulta) de cada resultado do grupo de resultados retornado de cada busca.

Aplicando a seguinte tabela de relevâncias, apresentada na Figura 23, ao exemplo dados acima ‘*Em uma busca por similaridade e sinônimos, em arquivos de metadados textuais sem formatação pela palavra “ground” e com cor predominante “verde” ’*.

	Valor Relevância
Busca Similaridade	3
Busca Sinônimos	2
Cor Predominante	4

Figura 23 - Exemplo de Valores de Relevância

Supondo que os determinados resultados de cada busca tenham retornado os arquivos e scores mostrados na Figura 24.

Sinônimos:

Arquivos	Score
E:\tcc\annotations\02\2145.eng	79
E:\tcc\annotations\02\2165.eng	79
E:\tcc\annotations\02\2389.eng	53
E:\tcc\annotations\02\2047.eng	53
E:\tcc\annotations\02\2014.eng	33

Cor Predominante:

Arquivos	Score
E:\tcc\image\02\2047.jpg	100
E:\tcc\image\02\2389.jpg	100
E:\tcc\image\02\2007.jpg	100
E:\tcc\image\02\2029.jpg	100
E:\tcc\image\02\2225.jpg	100

Similaridade:

Arquivos	Score
E:\tcc\annotations\02\2047.eng	63
E:\tcc\annotations\02\2225.eng	63
E:\tcc\annotations\02\2014.eng	63
E:\tcc\annotations\02\2145.eng	63
E:\tcc\annotations\02\2389.eng	55

Figura 24 - Exemplos de Retorno de Busca

Para este exemplo os cálculos de relevâncias aconteceriam da seguinte forma:

$$Arquivo\ E{:}\backslash tcc\backslash annotations\backslash 02\backslash 2145{:}\ \frac{(79*2)+(0*4)+\ (63*3)}{8} = 43{,}375$$

$$Arquivo\ E{:}\backslash tcc\backslash annotations\backslash 02\backslash 2165{:}\ \frac{(79*2)+(0*4)+\ (0*3)}{8} = 19{,}75$$

$$Arquivo\ E{:}\backslash tcc\backslash annotations\backslash 02\backslash 2389{:}\ \frac{(53*2)+(100*4)+\ (55*3)}{8} = 71{,}375$$

$$Arquivo\ E{:}\backslash tcc\backslash annotations\backslash 02\backslash 2047{:}\ \frac{(53*2)+(100*4)+\ (63*3)}{8} = 74{,}375$$

$$Arquivo\ E{:}\backslash tcc\backslash annotations\backslash 02\backslash 2014{:}\ \frac{(33*2)+(0*4)+\ (63*3)}{8} = 31{,}875$$

$$Arquivo\ E{:}\backslash tcc\backslash annotations\backslash 02\backslash 2007{:}\ \frac{(0*2)+(100*4)+\ (0\ *3)}{8} = 50{,}00$$

$$Arquivo\ E{:}\backslash tcc\backslash annotations\backslash 02\backslash 2029{:}\ \frac{(0*2)+(100*4)+\ (0\ *3)}{8} = 50{,}00$$

$$Arquivo\ E{:}\backslash tcc\backslash annotations\backslash 02\backslash 2225{:}\ \frac{(0*2)+(100*4)+\ (63\ *3)}{8} = 73{,}625$$

Juntando os resultados ordenados pela relevância em uma lista, tem-se o resultado apresentado na Figura 25.

Lista Final:

Arquivos	Score
E:\tcc\image\02\2047.eng	74,375
E:\tcc\image\02\2225.jpg	73,625
E:\tcc\image\02\2389.jpg	71,375
E:\tcc\image\02\2007.jpg	50,00
E:\tcc\image\02\2029.jpg	50,00
E:\tcc\annotations\02\2145.eng	43,375
E:\tcc\annotations\02\2014.eng	31,875
E:\tcc\annotations\02\2165.eng	19,75

Figura 25 - Exemplo de Combinação de Resultados

8.4. IMPLEMENTAÇÃO

A Figura 26, apresenta a estruturação de pacotes da implementação do *framework* proposto.

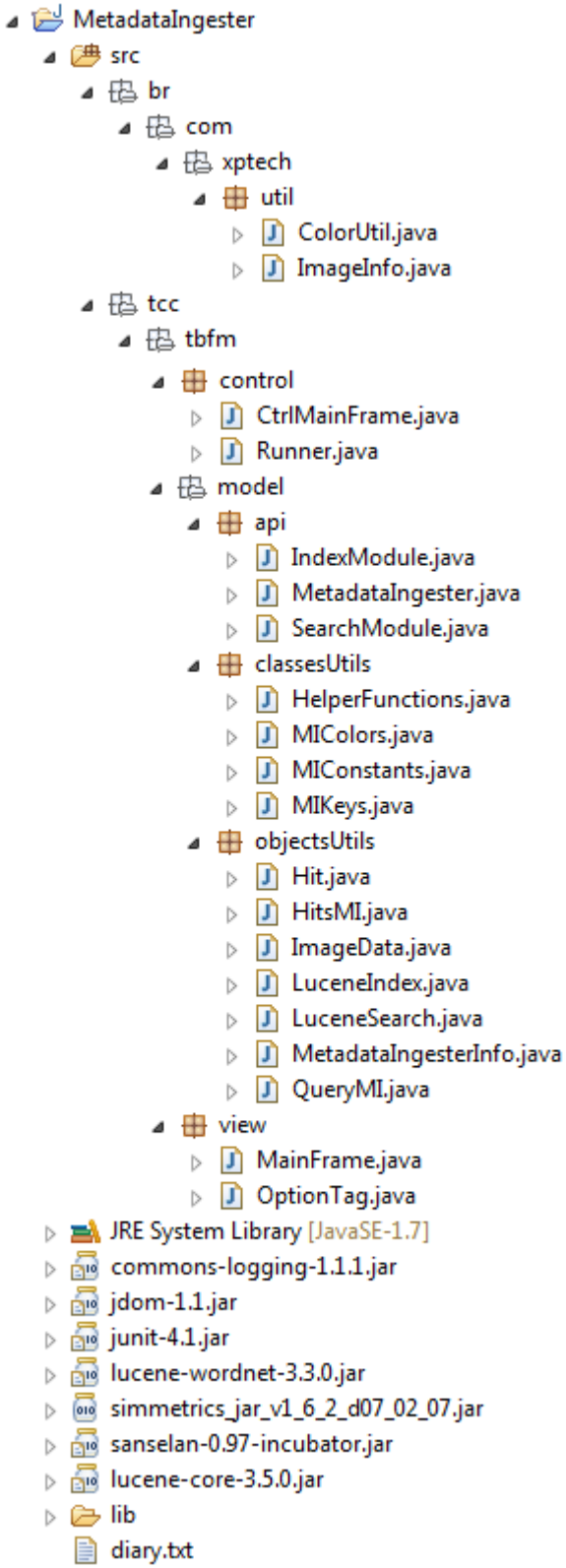

Figura 26 - Estrutura de Pacotes

Para a criação do projeto foi definida a estruturação de pacotes em dois ramos principais: tcc.tbfm e br.com.xptech.util. No ramo tcc.tbfm estão classes pertencentes à aplicação MI. No ramo br.com.xptech.util está o recurso de extração das características de coloração de imagens.

O ramo tcc.tbfm é subdividido conforme o modelo de Desenvolvimento de Software MVC (Model-View-Controller MSDN), que atualmente é considerado uma arquitetura padrão utilizada na Engenharia de Software. O modelo isola a lógica da interface do usuário, permitindo desenvolver, editar e testar cada parte da aplicação. A iteratividade entre os pacotes principais utilizados nesse padrão é mostrada na Figura 27:

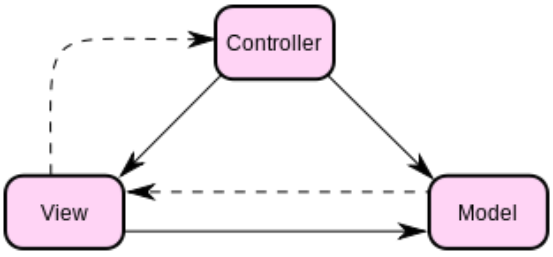

Figura 27 - Esquema gráfico do modelo MVC

8.4.1. PACOTE CONTROL

Neste pacote encontram-se as duas classes de controle da aplicação: *Runner* e *CtrlMainFrame*. A classe *Runner* é o ponto de entrada da aplicação. É esta classe que

cria o objeto de controle da tela principal chamado *CtrlMainFrame*. A classe *CtrlMainFrame* contém toda a lógica de todas as ações possíveis ao usuário na classe que renderiza a tela principal do MI, a *MainFrame* e a tela de configuração da marcação a ser buscada, a *OptionTag*. Sempre que algo acontece em qualquer uma das telas uma ação é executada pelo objeto de controle da tela criado na classe *Runner*.

8.4.2. PACOTE VIEW

Neste pacote, encontram-se outras duas classes que são responsáveis pela renderização das telas que permite ao usuário utilizar os recursos do MI: *MainFrame* e *OptionTag*.

A classe *MainFrame* permite ao usuário acesso aos seguintes recursos:

- ***Indexação***: O usuário pode definir o caminho em que o MI busca os arquivos de metadados textuais, as imagens e onde são armazenados os índices gerados pelo MI, caso o usuário queira gerar novos índices. Caso o usuário queira recuperar índices criados anteriormente então, as informações fornecidas são utilizadas para a busca das informações de índices, imagens e arquivos de metadados textuais.

- ***Busca***: O usuário pode escolher e combinar dentre os quatro tipos de buscas (simples, sinônimo, similaridade, substring) e os três tipos de dados (metadados textuais, metadados embutidos e coloração) para criar diversas buscas. Também é permitido ao usuário configurar as relevâncias de todos os itens que fazem parte da consulta, desde características da imagem até tipos de busca.

8.4.3. PACOTE MODEL

Este pacote foi subdividido em três outros pacotes chamados *api*, *classesUtils* e *objectsUtils*.

8.4.3.1. PACOTE *API*

No pacote *api* estão as três principais classes do aplicativo MI: *MetadataIngester*, *SearchModule* e *IndexModule*. A classe *MetadataIngester* representa o aplicativo propriamente dito e possui os objetos das classes *IndexModule* e *SearchModule* que representam, respectivamente, o módulo de indexação e busca da aplicação MI.

- *MetadataIngester* – É a classe correspondente ao objeto principal do projeto. É composta por dois outros objetos *IndexModule* e *SearchModule* responsáveis, respectivamente, pela indexação e busca.
- *IndexModule* – É responsável pela indexação dos dados físicos das imagens e seus arquivos de metadados textuais correspondentes. Conta com o auxilio da biblioteca Lucene para a geração dos índices baseados nas informações coletadas.
- *SearchModule* – É responsável pela busca dos dados. Correlaciona os dados físicos das imagens e seus arquivos de metadados textuais, com o objetivo de aumentar a acurácia dos resultados.

8.4.3.2. PACOTE *CLASSESUTILS*

No pacote *classesUtils* estão todas as classes utilizadas de forma estática, ou seja, não é necessária a criação de um objeto para a sua utilização. Este pacote é composto pelas seguintes classes:

- *HelperFunctions* – Contém funcionalidades bastante úteis na codificação, como por exemplo, limpar diretórios específicos, persistir e restaurar objetos.
- *MIColors* – Esta classe foi criada com a intenção de auxiliar a descoberta de que cor, dentro do conjunto base de cores, está mais próxima a cor de um determinado pixel da imagem. Esta classe contém uma listagem de diversos tons das cores utilizadas como base no MI.
- *MIKeys* – Contém as chaves utilizadas para a configuração da busca e também para a definição dos nomes das chaves para a geração dos documentos de indexação pelo Lucene. Possui chaves como:

IMAGE_WIDTH – *define largura da imagem*
IMAGE_HEIGHT – *define a altura da imagem*
IMAGE_SIZEKB – *define o tamanho em KB da imagem*
SEARCH_ANNOT_TXT *– define se a anotação à ser utilizada será a textual simples*
SEARCH_ANNOT_XML *– define se a anotação à ser utilizada será a em XML*
SEARCH_NUMBER_OF_RESULTS *– define o número de resultados à ser retornado*
SEARCH_WORD_VALUE *– define o valor que deve ser buscado*
SEARCH_PATH_VALUE *– define, para as anotações em XML, o caminho à ser buscado*

- *MIConstants* – Contém algumas definições de valores padrão para algumas variáveis, caso nenhum valor seja definido. Define, por exemplo, um valor padrão para a quantidade de resultados retornados para 50.

```
DEFAULT_SEARCH_NUMBER_OF_RESULTS = 50
DEFAULT_INDEX_FOLDER_TXT = "\\metadataTXTIndex"
DEFAULT_INDEX_FOLDER_XML = "\\metadataXMLIndex"
DEFAULT_INDEX_FOLDER_IMAGE = "\\metadataImageIndex"
DEFAULT_INDEX_TAG_FILE = "\\tagsListIndex.ser"
DEFAULT_RESULT_FILE_NAME = "\\result"
DEFAULT_RESULT_FILE_EXT = ".txt"
```

8.4.3.3. PACOTE *OBJECTUTILS*

No pacote *objectsUtils* estão todas as classes que são utilizadas como objetos, assim como os objetos auxiliares baseados na biblioteca Lucene para criação de índices e leitura dos mesmos, ou mesmo o objeto *QueryMI* que armazena todas as informações da busca. Este pacote possui as seguintes classes:

- *LuceneIndex e LuceneSearch* – São objetos que encapsulam as funcionalidades da biblioteca Lucene para indexação e busca, respectivamente.
- *ImageData* – É o objeto que extrai e armazena as informações físicas da imagem, para posteriormente servirem de base para a indexação e associação aos arquivos de metadados textuais em buscas. Essas informações são buscadas da própria imagem (como suas cores predominantes), do metadado geral (como data da última modificação) e do metadado específico de cada tipo

de imagem (como exif para imagens jpg e tiff que contém informações como a resolução, data de criação, câmera utilizada, dentre outros).

- *QueryMI* – Este objeto é passado como parâmetro na busca e pode ser configurado de acordo com as informações que devem ser buscadas. Possui duas estruturas HashTable contendo chaves e valores das informações que devem ser buscadas nos índices gerados a partir das imagens e de seus arquivos de metadados textuais.
- *Hit* – Este objeto apenas armazena informações sobre cada resultado de busca, como o seu score e o caminho do arquivo.
- *HitsMI* – Esta é a classe que é responsável por calculadas todas relevâncias de todos os resultados, gerar a nova lista com todos os nomes de todos os arquivos e escrever os arquivos de resultados.
- *MetadataIngesterInfo* – Este classe foi criada com a intenção de tornar o código implementado mais claro e conciso. Esta classe armazena informações que são utilizadas tanto pelo módulo de indexação quanto pelo módulo de busca, como o endereço onde são ou estão armazenados os índices, respectivamente.

8.4.4. BIBLIOTECAS UTILIZADAS

- *JDOM* (JDOM.ORG) – Utilizado para ler e manipular os arquivos de metadados textuais em formato XML. Oferece uma forma simples e eficiente de representação, leitura e escrita.

- *SimMetrics* (SIMMETRICS.ORG) – Biblioteca de métricas de similaridade ou distância entre palavras.
- *Apache SANSELAN* (SANSELAN.APACHE) – Promove o acesso ao metadado e informações extras acoplados à imagem.
- *Apache Lucene* (LUCENE_APACHE.ORG) – Biblioteca de Recuperação de Informações (IR – Information Retriever) e Busca de alto desempenho e fácil manuseio.
- *Apache Lucene-Wordnet* (LUCENE_WORDNET.APACHE) – Utiliza a base de dados Prolog do WordNet para buscas utilizando a biblioteca Apache Lucene.

8.4.5. MODELO LÓGICO

O relacionamento entre as classes utilizadas no projeto está ilustrado na Figura 28, de uma maneira simplificada. Os métodos e atributos foram omitidos para que a visão da estruturação das classes e seus relacionamentos fossem completa.

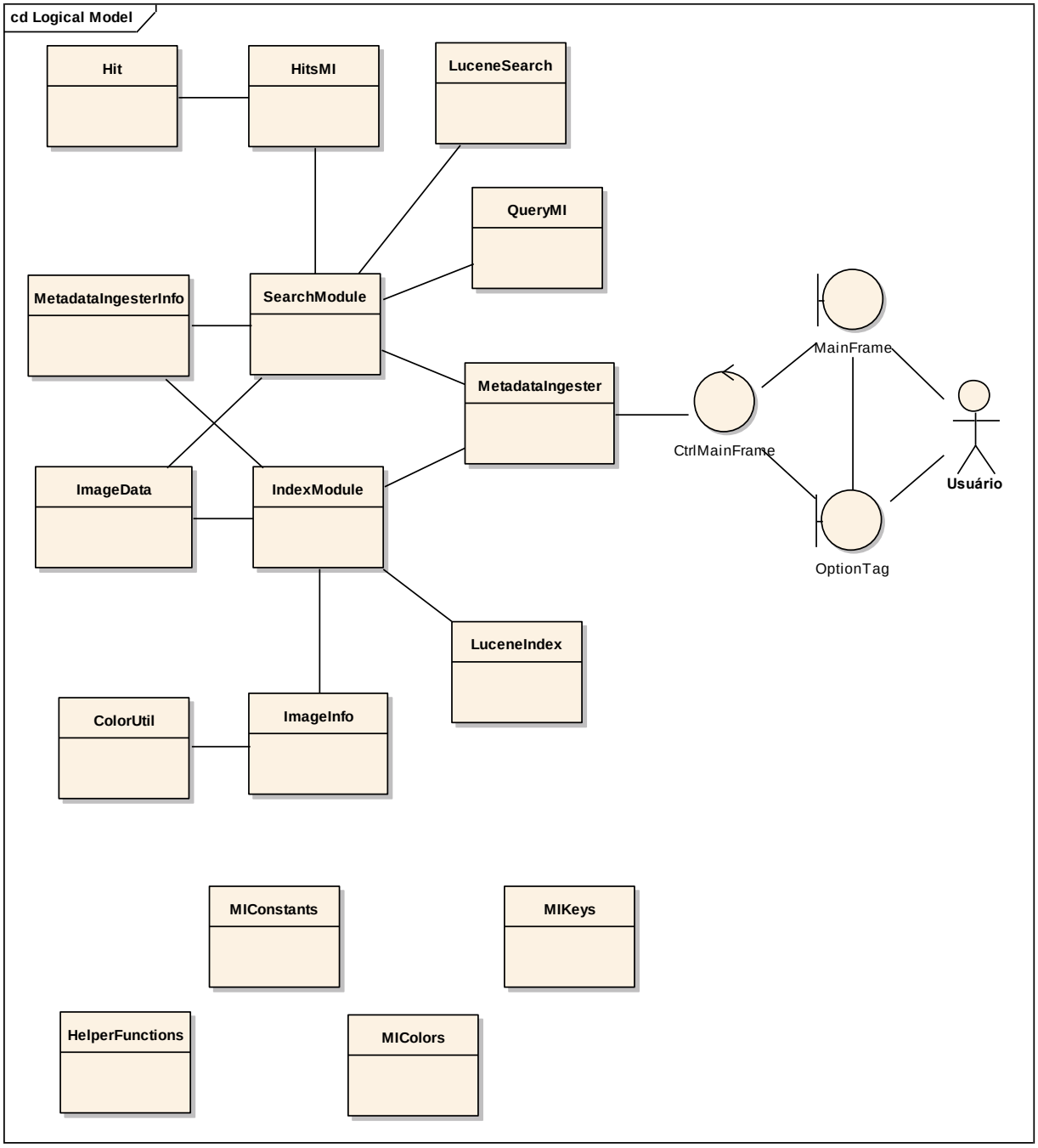

Figura 28 - Modelo Lógico

8.5. INTERFACE

A interface com o usuário desenvolvida foi bastante simplificada, afim de manter o foco no desenvolvimento do projeto e manter a fidelidade aos conceitos e definições

firmado no início desse trabalho. Porém, mesmo simples, a interface desenvolvida atende a todas as funcionalidades implementadas no projeto.

8.5.1. TELA PRINCIPAL

8.5.1.1. INDEXAÇÃO

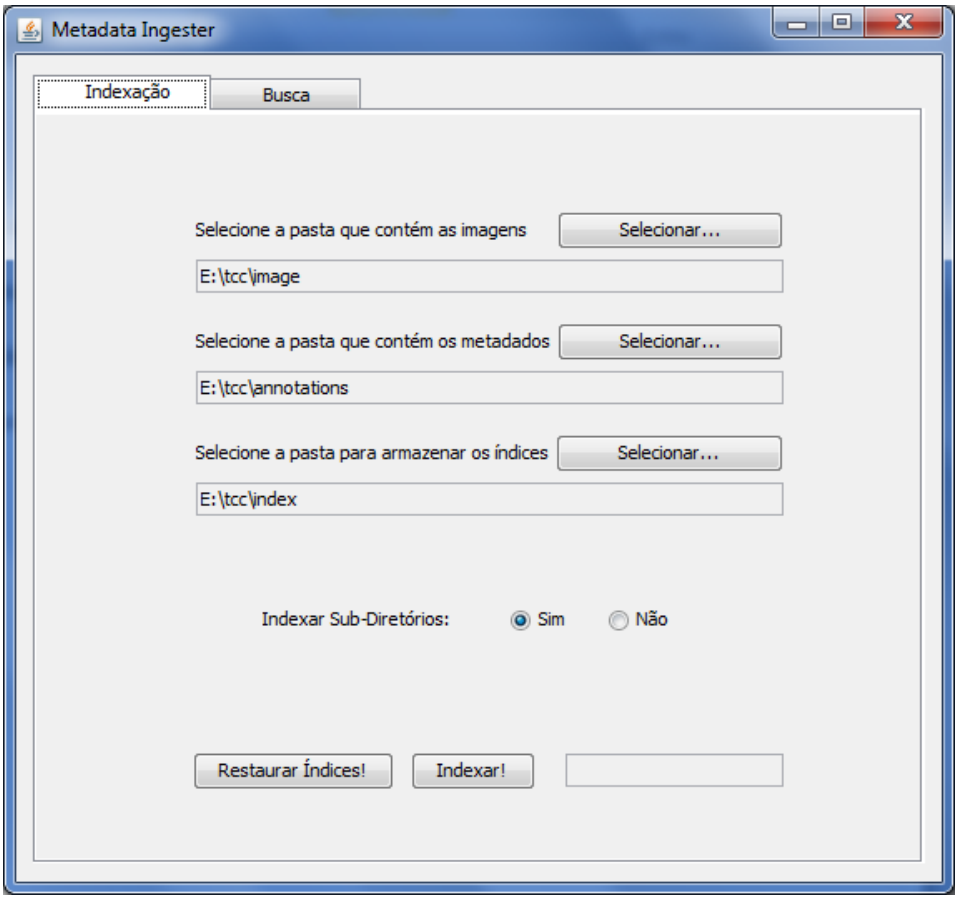

Figura 29 - Tela Principal (Indexação)

Na Tela Principal, representada pela Figura 29, o usuário pode optar entre gerar índices ou restaurar índices já gerados. Porém, em ambos os casos, é necessária a configuração de caminho de onde se encontram as imagens, os arquivos de metadados textuais e os índices. Apenas a opção de "Indexar Sub-Diretórios" não é

necessária a configuração no caso de restaurar índices já criados. Para o caso de criação de índices, esta opção determina se os subdiretórios, pertencentes ao caminho raiz fornecido, são indexados também.

8.5.1.2. BUSCA

Para executar buscas, primeiramente é necessária a configuração da mesma. Essa configuração pode ser total ou parcial, ou seja, a configuração pode utilizar todos os tipos de buscas e de fonte de dados ou utilizar apenas alguns tipos de buscas e fonte de dados. A configuração dos tipos de fontes de dados é feita em cada uma das 3 abas de busca, caso a aba seja mantida com a configuração original aquele tipo de dado não será acrescentado na busca. Por exemplo, na aba “Arquivo de Metadados Textuais” o valor padrão para o formato do metadado textual é “Nenhum”, caso essa configuração seja mantida o metadado textual não será utilizado. Nos casos das abas “Metadado Embutido” e “Coloração da Imagem” caso os campos não sejam preenchidos o tipo de dado que seria configurado não será utilizado na busca. A quarta aba chamada “Configurar Relevâncias” é utilizada para definir que resultados possuem uma maior prioridade que outros. Caso os valores da aba sejam mantidos com o padrão todos os resultados vão possuir a mesma relevância e a sequência em que esses resultados aparecerão dependerá apenas do *score* do *Lucene*.

Na aba “Metadados Textuais”, mostrada na Figura 30, deve ser configurada a busca em arquivos de metadados textuais, tanto com ou sem marcação XML. Nesta aba teremos as seguintes subdivisões:

- ***Selecione o Formato do Arquivo de Metadado Textual:*** Aqui deve ser selecionado em que formato de metadado textual a busca é realizada. Caso a opção "Nenhum" seja mantida selecionada, a busca em arquivos de metadados textuais não é realizada. Caso a opção "Plano" seja selecionada apenas em arquivos de metadados textuais sem formatação a busca é realizada. Caso a opção "XML" seja selecionada apenas a busca em arquivos de metadados textuais com formatação em XML é executada. Caso a opção "XML/Plano" seja selecionada ambos os índices de arquivos de metadados textuais são utilizados.

- ***Selecione a Marcação XML***: O usuário tem a opção de digitar a marcação XML. Porém, como foi explicado anteriormente, ela vai ser usada como chave para a busca do Lucene. Assim sendo, deve ser exatamente igual à que foi indexada. Mas também pode-se utilizar as mesmas buscas disponibilizadas às palavras e características para buscar a marcação XML através do botão "Buscar Marcação". Este botão dá acesso à tela "Configuração de Marcação XML" que será descrita em breve.

- ***Configure a Busca da Palavra***: Neste momento o usuário fornece a palavra a ser buscada no campo de texto "Palavra", que deve ser apenas uma palavra por busca, e utiliza os tipos de busca explicados anteriormente: Simples, Similaridade, Sinônimos e Substring. Pode ser feita a combinação dessas buscas apenas selecionando os tipos de buscas desejados. Relembrando: nas buscas simples é buscada a palavra exatamente como ela foi fornecida, nas buscas por substring a palavra é buscada dentro de outras palavras, nas buscas por sinônimos são feitas buscas simples utilizando a palavra como os sinônimos encontrados da palavra fornecida, nas buscas por similaridade é buscada

palavras que tenham no mínimo o índice de similaridade, fornecido no campo de texto “Porcentagem de Similaridade” com a palavra fornecida.

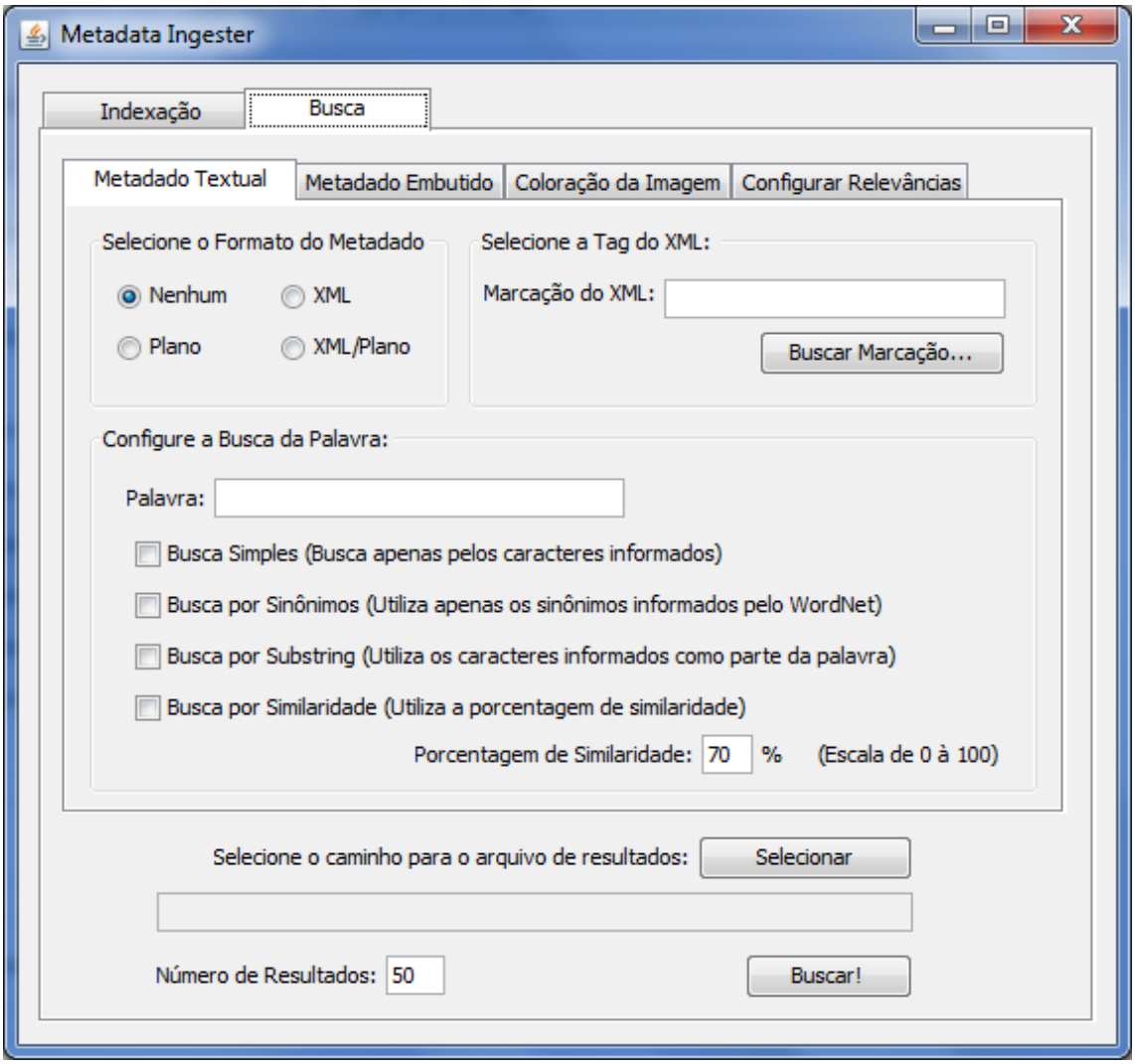

Figura 30 - Tela Principal (Busca Arquivo de Metadado)

Ao clicar em “Buscar Marcação” a tela “Configuração de Marcação XML” torna-se disponível como mostrada na Figura 31. Nela é executada a pesquisa no índice gerado pelo MI, que armazenou todas as marcações XML indexadas pelo Lucene. No campo “Buscar Por:” é fornecida a marcação, ou parte dela, que é pesquisada. As buscas possuem o mesmo conceito da tela anterior para a busca de palavra. Após a configuração de busca concluída e o botão “Listar Marcações” clicado, é listada as

marcações encontradas. Uma das marcações deve ser selecionada antes de selecionar "OK" da tela.

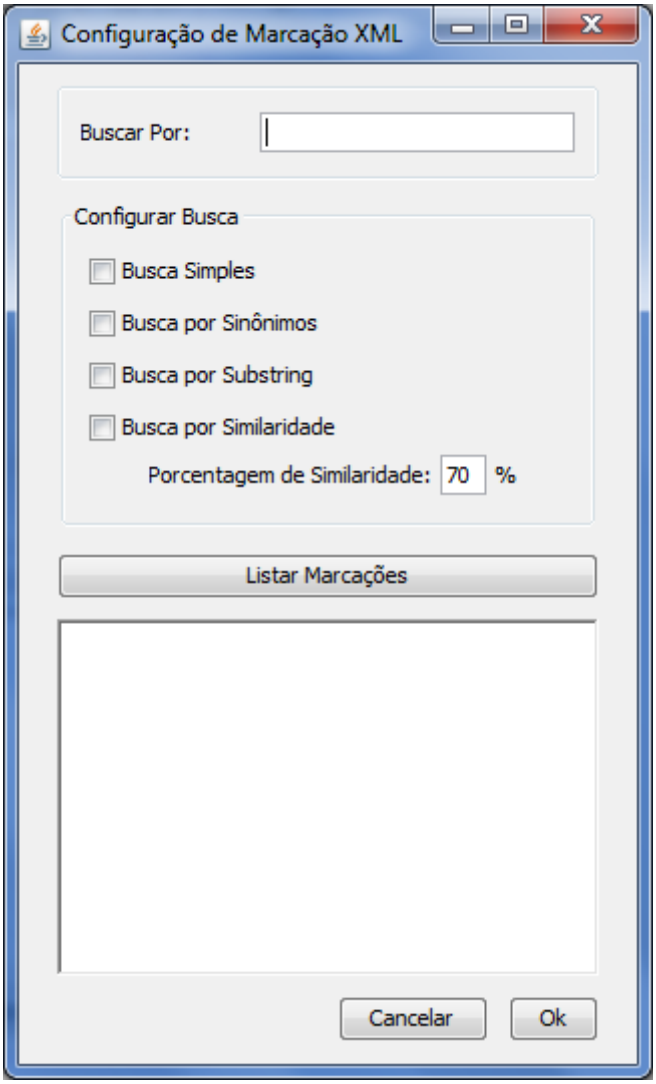

Figura 31 - Tela Secundária (Configurar Marcação XML)

Na aba "Metadado Embutido" (Figura 32), as informações sobre as características extraídas dos metadados das imagens são configuradas para a busca. As informações sobre largura, altura, tamanho, resolução X e Y, data de criação e data de última modificação podem ser configurados através de intervalos, mas caso o usuário queira um valor único ele deve preencher os dois campos de intervalo com o mesmo valor. As unidades utilizadas estão simbolizadas ao lado dos campos de texto, por exemplo, na

largura que é medida em pixel e simbolizada por *px*. Para as características: modelo de cores, formato e câmera é disponibilizada uma listagem de alguns valores padrão para ser selecionado.

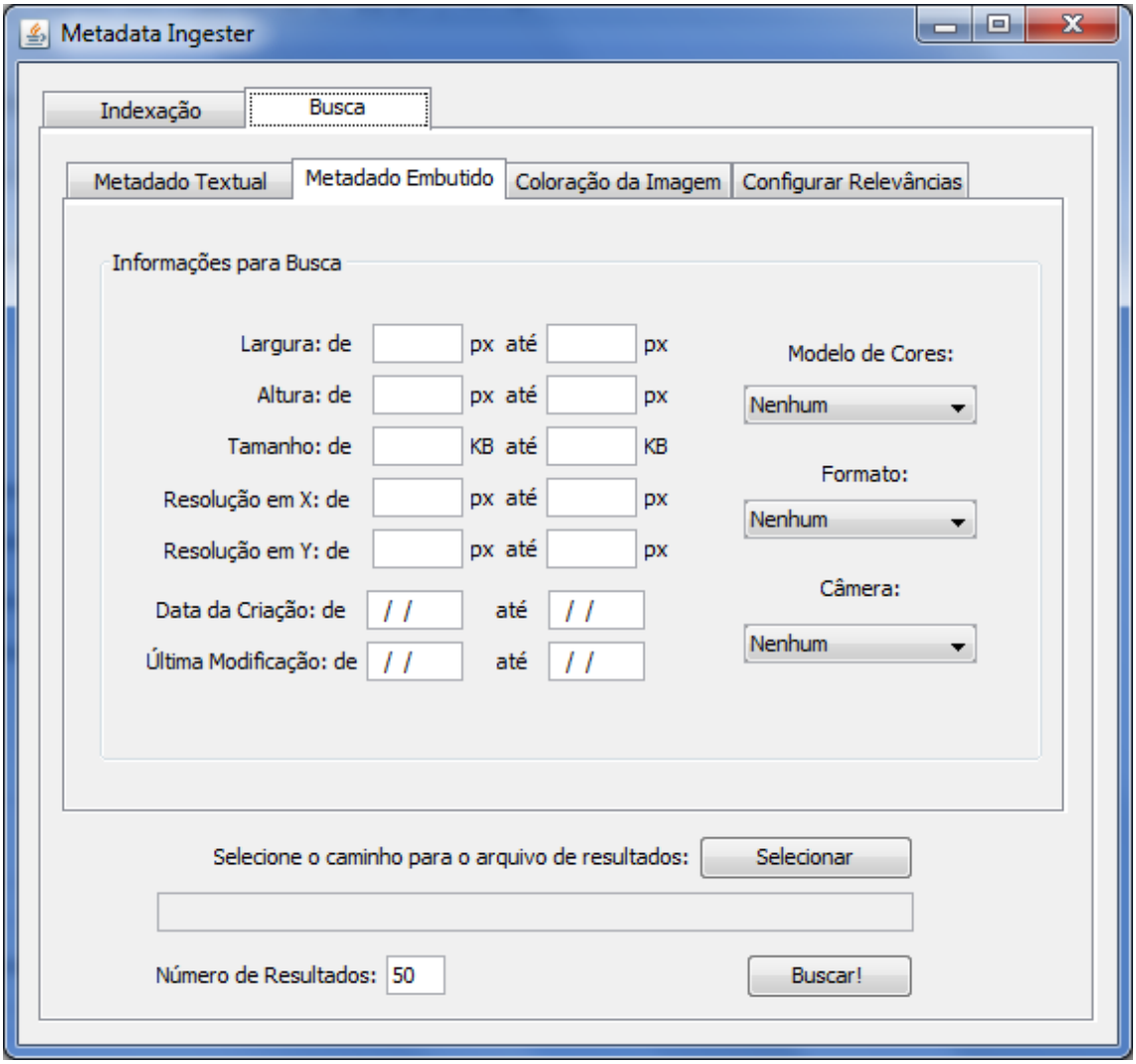

Figura 32 - Tela Principal (Busca Metadado)

Na aba "Coloração de Imagem" (Figura 33), são configuradas as informações sobre cor predominante e porcentagem de cada cor da imagem a ser buscada. Para a seleção de cor predominante foi disponibilizada a lista com o conjunto de cores definido no início do projeto e para a seleção das porcentagens de cada cor das imagens também foi disponibilizado o recurso de intervalos.

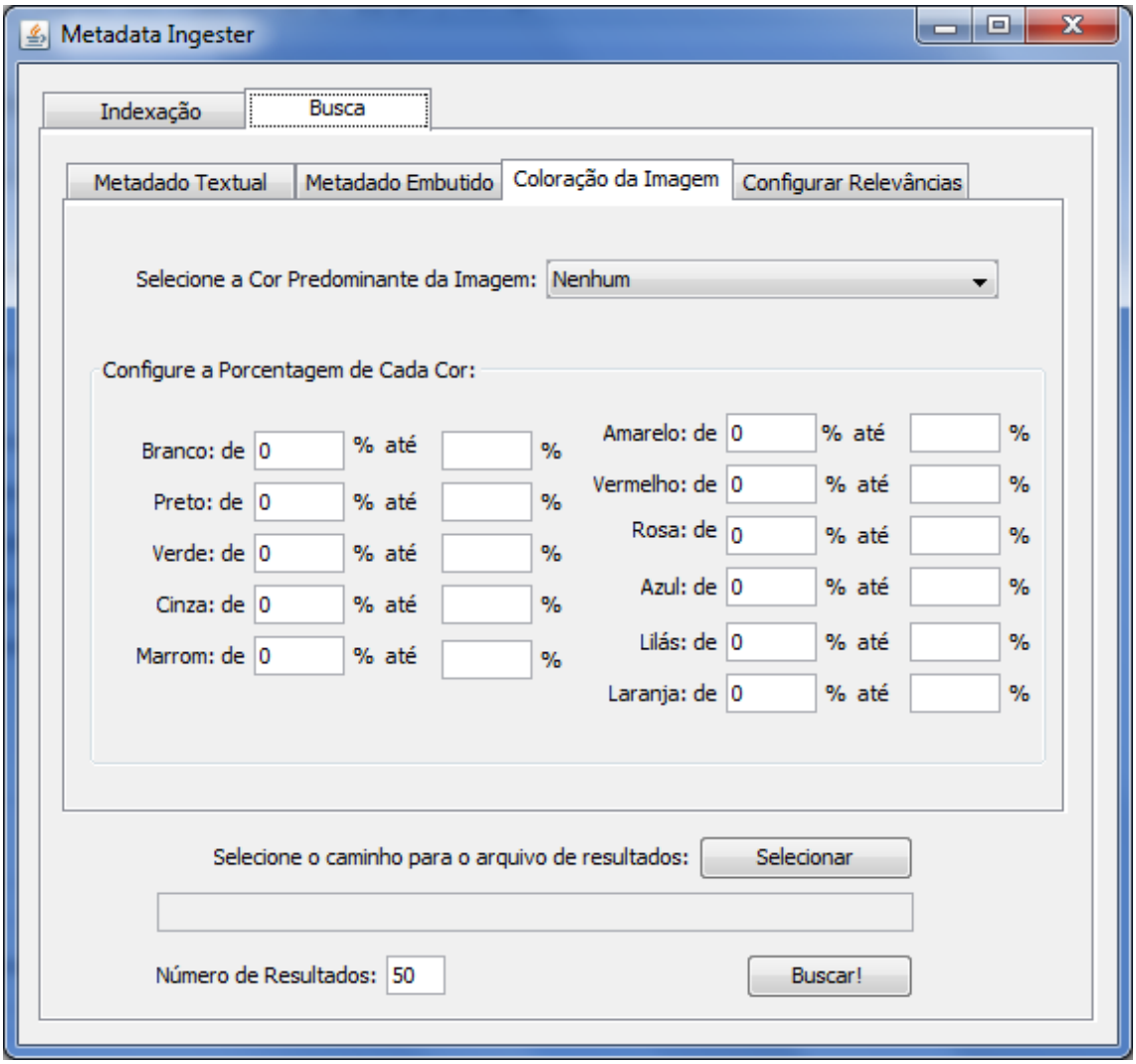

Figura 33 - Tela Principal (Busca Coloração da Imagem)

Na aba "Configurar Relevâncias" (Figura 34) é possível personalizar o cálculo de relevância de cada resultado encontrado pelo módulo de busca do MI. Essa configuração pode ser feita a cada nova busca, ou pode ser mantida de uma busca anterior. Por padrão foi mantida a mesma relevância para cada característica e fonte de dados, resultando em que nenhum resultado de busca é mais relevante que o outro.

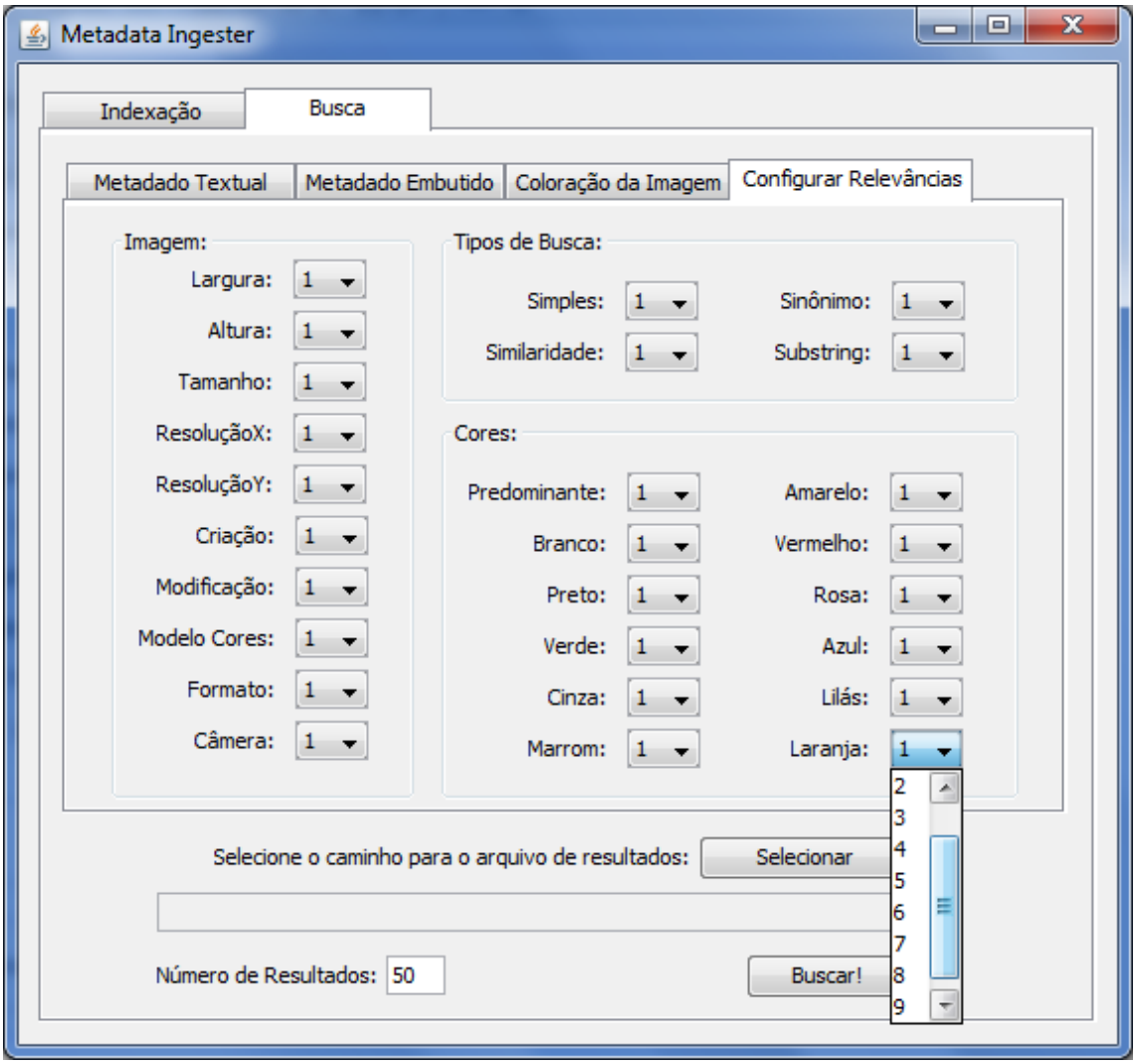

Figura 34 - Tela Principal (Busca Configurar Relevâncias)

A base da Tela Principal permite a configuração do caminho em que o arquivo de resultados da busca é criado. Nesta primeira etapa os resultados são retornados em formato de texto, com o caminho do arquivo que atende a todos os requisitos da busca. Como trabalho futuro, pode ser desenvolvida uma interface web para a disponibilização dos resultados com as imagens selecionadas. Também pode ser configurado o máximo de retornos que cada busca contém através do campo de texto "Número de Resultados".

9. EXPERIMENTOS

Os experimentos foram executados com o propósito de validar a *revocação* e *precisão* do MI. Precisão é a fração dos documentos já examinados que são relevantes, e revocação é a fração dos documentos relevantes observada dentre os documentos examinados (CARDOSO, 2000). Para o cálculo da média de revocação e precisão, foram coletados os resultados de 40 diferentes configurações de busca, distribuídas da seguinte forma: *10 Buscas Simples, 10 Buscas por Similaridade, 10 Buscas por Sinônimos* e *10 Buscas por Substring*. A Figura 33 representa os resultados obtidos após os experimentos.

Resultados das Buscas		
Tipo de Busca	Média Revocação	Média Precisão
Simples	0,85	0,80
Similaridade	0,84	0,81
Substring	0,82	0,81
Sinônimos	0,85	0,81

Figura 35 - Médias Resultantes dos Experimentos para Revocação e Precisão

As informações coletadas nas 40 buscas foram armazenadas em tabelas que possuem as seguintes colunas e estão disponíveis nos Anexos I, II, III e IV:

- *ID*: representa um identificador para busca a ser realizada.

- *Fontes*: representa as fontes de dados que serão utilizadas na busca.

- *Dados da Busca*: representa quais as informações que serão configuradas na busca.

- *Relevantes na Base*: representa a quantidade de arquivos que deveriam retornar da execução da busca.

- *Retorno*: representa a quantidade de arquivos que retornaram na execução da busca.

- *Relevantes no Retorno*: representa a quantidade de arquivos relevantes na base que retornaram no resultado da execução da busca.

- Precisão: calculada pela seguinte fórmula $Relevantes\ no\ Retorno/_{Retorno}$

- *Revocação*: calculada pela fórmula $Relevantes\ no\ Retorno/_{Relevantes\ na\ Base}$

10. CONCLUSÃO

A proposta deste projeto, descrita em mais detalhes na introdução deste documento, é fornecer um framework para indexação e busca sobre informações de imagens e aos metadados associados a ela. Permitindo a busca simples (que retorna arquivos que contenham palavras estruturalmente iguais à que foi fornecida), por similaridade (que retorna arquivos que contenham palavras estruturalmente semelhantes), por sinônimos (que retorna arquivos que contenham palavras que possuem o mesmo sentido da que foi fornecida) e por substring (que retorna palavras que contenham a palavra fornecida na sua estrutura) sobre informações recuperadas de metadados textuais, metadados embutidos e sobre a coloração da mesma. Além de gerar a implementação desse framework, com o objetivo de demonstrar a viabilidade da proposta.

Durante a execução do projeto foi possível verificar a lacuna que existe no mercado de sistemas que atendam às características propostas para o projeto através de pesquisas e de comparações entre os sistemas existentes. E neste contexto o MI pode ser considerado um sistema completo para auxiliar o processo de criação de índices e pesquisas combinadas sobre diversas fontes de dados.

Mesmo já trabalhando com uma grande quantidade de informações, o MI foi desenvolvido para que pudesse ser possível a sua evolução, ou seja, é possível evoluir o projeto atual para acrescentar fontes de dados e tipos de buscas diferenciadas sem qualquer problema na implementação dessas funcionalidades.

11. TRABALHOS FUTUROS

Todos os itens propostos neste projeto foram alcançados. Porém existem algumas tarefas que, caso sejam executadas, podem tornar a implementação do framework, o MI, mais otimizado e com uma melhor usabilidade:

- Possibilidade de criação de base de dados, com base nas informações das fontes de dados (arquivos de metadados textuais, metadado embutido e coloração da imagem).
- Otimização da indexação dos arquivos.
- Atualização incremental dos índices.
- Busca por palavras compostas.
- Permitir que o Lucene gerencie também os índices de marcações.
- Disponibilizar a busca por sinônimos em outras línguas, além do Inglês.
- Desenvolvimento da interface Web para a execução de buscas e retorno de resultados.
- Acrescentar o acesso à metadados embarcados de outros formatos de compressão além do Exif.

12. REFERÊNCIAS BIBLIOGRÁFICAS

ALMEIDA, M. B. (2002). Uma introdução ao XML, sua utilização na Internet e alguns conceitos complementares. *Ciência da Informação On-line, Volume31, n. 2* .

APACHE.ORG. (s.d.). Acesso em 15 de 11 de 2012, disponível em The Apache Software Foundation: http://www.apache.org/

ARANHA, C., & PASSOS, E. (2006). A Tecnologia de Mineração de Textos. *RESI-Revista Elerônica de Sistemas de Informação, Nº2, Volume5* , 1-8.

BREMNER, D., DEMAINE, E., ERICKSON, J., IACONO, J., LANGERMAN, S., MORIN, P., et al. (2005). Output-sensitive algorithms for computing nearest-neighbour decision boundaries. *Discrete and Computational Geometry* , 593-604.

CARDOSO, O. N. (2000). Recuperação de informação. *Infocomp: Revista de Computação da UFLA* .

CIPA DC-008-Translation-2012, C. (s.d.). *Standart of th Camera & Imaging Products Association*. Acesso em 20 de 02 de 2013, disponível em CIPA: http://www.cipa.jp/english/hyoujunka/kikaku/pdf/DC-008-2012_E.pdf

CPAN.ORG. (s.d.). Acesso em 11 de 12 de 2011, disponível em http://search.cpan.org/~bettelli/Image-MetaData-JPEG-0.153/lib/Image/MetaData/JPEG.pod

EGOTHOR. (s.d.). Acesso em 03 de 12 de 2011, disponível em http://www.egothor.org

GARCIA, S. d., MOURA, A. M., & CAMPOS, M. L. (1999). Metadados para Documentação e Recuperação de Imagens. *Dissertação (Mestrado em Sistemas e Computação) – Instituto Militar de Engenharia* . Rio de Janeiro.

GOSPODNETIC, O., & HATCHER, E. (2005). *Lucene in Action.* Greenwich: Manning Publications.

JAKARTA.ORG. (s.d.). Acesso em 15 de 11 de 2012, disponível em http://jakarta.apache.org/

JavaSE String. (s.d.). Acesso em 02 de 11 de 2012, disponível em http://docs.oracle.com/javase/1.4.2/docs/api/java/lang/String.html

JDOM.ORG. (s.d.). Acesso em 06 de 10 de 2012, disponível em http://www.jdom.org/

JEITA. (s.d.). Acesso em 14 de 11 de 2012, disponível em Japan Electronic and Information Technology Industries Association: http://www.jeita.or.jp/english/

LUCENE_APACHE.ORG. (s.d.). Acesso em 2012 de 06 de 10, disponível em Site da Apache Software Fundation: http://lucene.apache.org/

LUCENE_WORDNET.APACHE. (s.d.). Acesso em 10 de 06 de 2012, disponível em Site da Apache Software Fundation: http://lucene.apache.org/core/old_versioned_docs/versions/3_0_0/api/contrib-wordnet/org/apache/lucene/wordnet/package-summary.html

MANNING, C. D., RAGHAVAN, P., & SCHÜTZE, H. (2008). *Introduction to Information Retrieval.* New York: Cambridge University Press.

Model-View-Controller MSDN. (s.d.). Acesso em 03 de 11 de 2012, disponível em MSDN: http://msdn.microsoft.com/en-us/library/ff649643.aspx

MOURA, A. C. (2005). A importância dos metadados no uso das Geotecnologias e na difusão da Cartografia Digital. *II Seminário Nacional sobre Mapeamento Sistemático – CREA-MG* , 1 -18.

MSDN Microsoft. (s.d.). Acesso em 10 de 12 de 2011, disponível em Microsoft Index: http://msdn.microsoft.com/en-us/library/dd582937%28v=office.11%29.aspx

NAMAZU.ORG. (s.d.). Acesso em 10 de 12 de 2011, disponível em http://www.namazu.org/

NAVATHE, S., & ELMASRI, R. (1999). *Fundamentals of Database Systems, Third Edition.* Addison-Wesley.

NETO, A. R., ROCHA, E. M., & SANTOS, C. A. (2005). Uma proposta de avaliação baseada em anotações de vídeos digitais. *V Workshop de Informática Médica* .

PRABHAKARAN, B. (2007). *Multimedia Database Management Systems.* Springer .

PRINCETON.EDU. (s.d.). Acesso em 02 de 11 de 2012, disponível em http://www.princeton.edu/main/

SADMAN.COM. (s.d.). Acesso em 31 de 10 de 2012, disponível em SadMan Search: http://www.sadmansoftware.com/search/index.html

SANSELAN.APACHE. (s.d.). Acesso em 10 de 06 de 2012, disponível em Site da Apache Software Fundation: http://commons.apache.org/imaging/

SHEFFIELD.UK. (s.d.). Acesso em 02 de 11 de 2012, disponível em The University of Sheffield: http://www.shef.ac.uk/

SHERMAN, C., & PRICE, G. (2001). *The invisible Web: Uncovering information sources search engines can't see.* Medford: CyberAge Books.

SILBERSCHATZ, A., KORTH, H. F., & SUDARSHAN, S. (2006). *Sistema de banco de dados, 5ed.* Rio de Janeiro: Elsevier.

SIMMETRICS.ORG. (s.d.). Acesso em 10 de 06 de 2012, disponível em http://www.aktors.org/technologies/simmetrics/index.html

SINGH, S., HADDON, J., & MARKOU, M. (1999). NEAREST NEIGHBOUR STRATEGIES FOR IMAGE UNDERSTANDING. *Workshop on Advanced Concepts for Intelligent Vision Systems (ACIVS'99)* . Baden-Baden.

SWISH.ORG. (s.d.). Acesso em 10 de 12 de 2011, disponível em http://swish-e.org/

W3C.Org. (s.d.). Acesso em 03 de 11 de 2011, disponível em World Wide Web Consortium: http://www.w3.org

WIVES, L. K. (22 de 04 de 1999). Um estudo sobre Agrupamento de Documentos Textuais em Processamento de Informações não Estruturadas Usando Técnicas de "Clustering". *Universidade Federal do Rio Grande do Sul. Instituto de Informática. Programa de Pós-Graduação em Computação* . Porto Alegre, Brasil.

WORDNET.EDU. (s.d.). Acesso em 02 de 11 de 2012, disponível em http://wordnet.princeton.edu/

WRENSOFT_PLUGINS.COM. (s.d.). Acesso em 11 de 11 de 2012, disponível em http://www.wrensoft.com/zoom/plugins.html

XAPIAN.ORG. (s.d.). Acesso em 03 de 12 de 2011, disponível em http://xapian.org

YAN, H., DING, S., & SUEL, T. (2009). Inverted Index Compression and Query Processing with Optimized Document Ordering. *WWW '09 Proceedings of the 18th international conference on World wide web* , 401-410. New York, NY, USA: ACM.

ZOOM_WRENSOFT.COM. (s.d.). Acesso em 11 de 12 de 2011, disponível em WRENSOFT: http://www.wrensoft.com/zoom/index.html

13. Anexo I

Busca Simples							
ID	Fontes	Dados da Busca	Relevantes na base	Retorno	Relevantes no Retorno	Revocação	Precisão
1	Plano	*Palavra:* Sky	13	13	13	1	1
	XML	*Marcação:* DOC/DESCRIPTION *Palavra:* Sky	13	13	13	1	1
2	Plano e Coloração	*Palavra:* Sky *Cor Predominante:* Azul	52	64	47	0,90	0,73
	XML e Coloração	*Marcação:* DOC/DESCRIPTION *Palavra:* Sky *Cor Predominante:* Azul	52	64	47	0,90	0,73
3	Coloração	*Cor Predominante:* Verde	68	74	61	0,90	0,82
4	Imagem	*Tamanho:* de 60Kb até 65Kb	24	17	15	0,63	0,88
5	Plano e Imagem	*Palavra:* Sky *Tamanho:* 45Kb	22	25	18	0,82	0,72
	XML e Imagem	*Marcação: DOC/DESCRIPTION* *Palavra:* Sky *Tamanho:* 45Kb	22	25	18	0,82	0,72
6	Plano, Coloração e Imagem	*Palavra:* Sky *Cor Predominante:* Azul *Tamanho:* de 20Kb até 30Kb	72	69	58	0,81	0,84
	XML, Coloração e Imagem	*Marcação:* DOC/DESCRIPTION *Palavra:* Sky *Cor Predominante:* Azul *Tamanho:* de 20Kb até 30Kb	72	69	58	0,81	0,84
7	Plano e Imagem	*Palavra:* Ground *Tamanho:* de 30Kb até 35Kb	21	23	17	0,81	0,74
	XML e Imagem	*Marcação:* DOC/DESCRIPTION *Palavra:* Ground *Tamanho:* de 30Kb até 35Kb	21	23	17	0,81	0,74
8	Plano e Coloração	*Palavra:* Jungle *Cor Predominante: Verde*	77	82	61	0,79	0,74
	XML e Coloração	*Marcação: DOC/TITLE* *Palavra:* Jungle *Cor Predominante:* Verde	69	74	56	0,81	0,76
9	Coloração e Imagem	*Cor Predominante:* Vermelho *Tamanho: de 60Kb*	16	18	15	0,94	0,83
10	Plano e Imagem	*Palavra:* Water *Cor Predominante:* Branco	19	22	17	0,89	0,77
	XML e Imagem	*Marcação:* DOC/DESCRIPTION *Palavra:* Water *Cor Predominante:* Branco	19	22	17	0,89	0,77
					Média	0,85	0,80

14. Anexo II

Busca Similaridade							
ID	Fontes	Dados da Busca	Relevantes na base	Retorno	Relevantes no Retorno	Revocação	Precisão
1	*Plano*	*Palavra:* Sky	15	16	13	0,87	0,81
	XML	*Marcação:* DOC/DESCRIPTION *Palavra:* Sky	15	16	13	0,87	0,81
2	*Plano e Coloração*	*Palavra:* Sky *Cor Predominante:* Azul	54	63	48	0,89	0,76
	XML e Coloração	*Marcação:* DOC/DESCRIPTION *Palavra:* Sky *Cor Predominante:* Azul	54	63	48	0,89	0,76
3	*Coloração*	*Cor Predominante:* Verde	68	74	61	0,90	0,82
4	*Imagem*	*Tamanho:* de 60Kb até 65Kb	24	17	15	0,63	0,88
5	*Plano e Imagem*	*Palavra:* Sky *Tamanho:* 45Kb	26	28	23	0,88	0,8214286
	XML e Imagem	*Marcação: DOC/DESCRIPTION* *Palavra:* Sky *Tamanho:* 45Kb	26	28	23	0,88	0,8214286
6	*Plano, Coloração e Imagem*	*Palavra:* Sky *Cor Predominante:* Azul *Tamanho:* de 20Kb até 30Kb	74	70	59	0,80	0,84
	XML, Coloração e Imagem	*Marcação:* DOC/DESCRIPTION *Palavra:* Sky *Cor Predominante:* Azul *Tamanho:* de 20Kb até 30Kb	74	70	59	0,80	0,84
7	*Plano e Imagem*	*Palavra:* Ground *Tamanho:* de 30Kb até 35Kb	24	25	19	0,79	0,76
	XML e Imagem	*Marcação:* DOC/DESCRIPTION *Palavra:* Ground *Tamanho:* de 30Kb até 35Kb	24	25	19	0,79	0,76
8	*Plano e Coloração*	*Palavra:* Jungle *Cor Predominante:* Verde	81	82	69	0,85	0,84
	XML e Coloração	*Marcação:* DOC/TITLE *Palavra:* Jungle *Cor Predominante:* Verde	76	74	61	0,80	0,82
9	*Coloração e Imagem*	*Cor Predominante:* Vermelho *Tamanho: de 60Kb*	16	18	15	0,94	0,83
10	*Plano*	*Palavra:* Water *Cor Predominante:* Branco	21	24	18	0,86	0,75
	XML	*Marcação:* DOC/DESCRIPTION *Palavra:* Water *Cor Predominante:* Branco	21	24	18	0,86	0,75
					Média	0,84	0,81

15. Anexo III

Busca Substring							
ID	Fontes	Dados da Busca	Relevantes na base	Retorno	Relevantes no Retorno	Revocação	Precisão
1	*Plano*	*Palavra:* Sky	18	21	16	0,89	0,76
	XML	*Marcação:* DOC/DESCRIPTION *Palavra:* Sky	18	21	16	0,89	0,76
2	*Plano e Coloração*	*Palavra:* Sky *Cor Predominante:* Azul	59	63	49	0,83	0,78
	XML e Coloração	*Marcação:* DOC/DESCRIPTION *Palavra:* Sky *Cor Predominante:* Azul	59	63	49	0,83	0,78
3	*Coloração*	*Cor Predominante:* Verde	68	74	61	0,90	0,82
4	*Imagem*	*Tamanho:* de 60Kb até 65Kb	24	17	15	0,63	0,88
5	*Plano e Imagem*	*Palavra:* Sky *Tamanho:* 45Kb	28	26	21	0,75	0,81
	XML e Imagem	*Marcação: DOC/DESCRIPTION* *Palavra:* Sky *Tamanho:* 45Kb	28	26	21	0,75	0,81
6	*Plano, Coloração e Imagem*	*Palavra:* Sky *Cor Predominante:* Azul *Tamanho:* de 20Kb até 30Kb	78	81	65	0,83	0,80
	XML, Coloração e Imagem	*Marcação:* DOC/DESCRIPTION *Palavra:* Sky *Cor Predominante:* Azul *Tamanho:* de 20Kb até 30Kb	78	81	65	0,83	0,80
7	*Plano e Imagem*	*Palavra:* Ground *Tamanho:* de 30Kb até 35Kb	86	85	68	0,79	0,80
	XML e Imagem	*Marcação:* DOC/DESCRIPTION *Palavra:* Ground *Tamanho:* de 30Kb até 35Kb	86	85	68	0,79	0,80
8	*Plano e Coloração*	*Palavra:* Jungle *Cor Predominante:* Verde	78	83	62	0,79	0,75
	XML e Coloração	*Marcação:* DOC/TITLE *Palavra:* Jungle *Cor Predominante:* Verde	71	75	57	0,80	0,76
9	*Coloração e Imagem*	*Cor Predominante:* Vermelho *Tamanho: de 60Kb*	16	18	15	0,94	0,83
10	*Plano*	*Palavra:* Water *Cor Predominante:* Branco	34	32	28	0,82	0,88
	XML	*Marcação:* DOC/DESCRIPTION *Palavra:* Water *Cor Predominante:* Branco	34	32	28	0,82	0,88
					Média	0,82	0,81

16. Anexo IV

Busca Sinônimos							
ID	Fontes	Dados da Busca	Relevantes na base	Retorno	Relevantes no Retorno	Revocação	Precisão
1	*Plano*	*Palavra:* Sky	13	13	13	1,00	1,00
	XML	*Marcação:* DOC/DESCRIPTION *Palavra:* Sky	13	13	13	1,00	1,00
2	*Plano e Coloração*	*Palavra:* Sky *Cor Predominante:* Azul	52	64	47	0,90	0,73
	XML e Coloração	*Marcação:* DOC/DESCRIPTION *Palavra:* Sky *Cor Predominante:* Azul	52	64	47	0,90	0,73
3	*Coloração*	*Cor Predominante:* Verde	68	74	61	0,90	0,82
4	*Imagem*	*Tamanho:* de 60Kb até 65Kb	24	17	15	0,63	0,88
5	*Plano e Imagem*	*Palavra:* Sky *Tamanho:* 45Kb	22	25	18	0,82	0,72
	XML e Imagem	*Marcação: DOC/DESCRIPTION* *Palavra:* Sky *Tamanho:* 45Kb	22	25	18	0,82	0,72
6	*Plano, Coloração e Imagem*	*Palavra:* Sky *Cor Predominante:* Azul *Tamanho:* de 20Kb até 30Kb	72	69	58	0,81	0,84
	XML, Coloração e Imagem	*Marcação:* DOC/DESCRIPTION *Palavra:* Sky *Cor Predominante:* Azul *Tamanho:* de 20Kb até 30Kb	72	69	58	0,81	0,84
7	*Plano e Imagem*	*Palavra:* Ground *Tamanho:* de 30Kb até 35Kb	89	87	69	0,78	0,79
	XML e Imagem	*Marcação:* DOC/DESCRIPTION *Palavra:* Ground *Tamanho:* de 30Kb até 35Kb	89	87	69	0,78	0,79
8	*Plano e Coloração*	*Palavra:* Jungle *Cor Predominante:* Verde	77	82	61	0,79	0,74
	XML e Coloração	*Marcação:* DOC/TITLE *Palavra:* Jungle *Cor Predominante:* Verde	69	74	56	0,81	0,76
9	*Coloração e Imagem*	*Cor Predominante:* Vermelho *Tamanho: de 60Kb*	16	18	15	0,94	0,83
10	*Plano*	*Palavra:* Water *Cor Predominante:* Branco	19	22	17	0,89	0,77
	XML	*Marcação:* DOC/DESCRIPTION *Palavra:* Water *Cor Predominante:* Branco	19	22	17	0,89	0,77
					Média	0,85	0,81

Metadata Ingester - Ferramenta de indexação e busca sobre arquivos de metadados em imagens

Taise B. F. Marques

Centro Tecnológico – Universidade Federal de Santa Catarina(UFSC)
88040-900 – Santa Catarina – SC – Brazil

Abstract. *Easy access to the internet and information resulted in increased data storage, mainly in repositories. And in order to these data can be recovered, the metadata has been a important tool for organizing and rapid identification without the direct access to these data. Realizing the need to recover that data, this paper proposes a framework that allows the retrieval of information in metadata associated with other data. Initially the framework treats metadata associated with images, because they are more common and numerous. It recognizes tree types of information associated with images that are textual metadata, embedded metadata and color information of the image, providing four types of searches for the recovered information which are simple search (returning words structurally identical to the one provided), similarity search (returning structurally similar words), synonymous search (returning words that have the same sense of that was provided) and substring search (words that contain the word provided). The framework was implemented in order to put in practice the theoretical definitions and allow tests of recall (fraction of relevant documents found among the documents examined) and precision (fraction of documents previously examined that are relevant), who provided a satisfactory result in the return of data.*

Keywords**:** *Indexing, Search data, Metadata, Embedded Metadata, Textual Metadata.*

Resumo. *O fácil acesso à internet e à informação resultou no aumento do armazenamento de dados, principalmente em repositórios. E para que esses dados possam ser recuperados, o metadado vem sendo uma ferramenta importante para a organização e identificação rápida sem a necessidade de acesso direto a esses dados. Percebendo a necessidade de uma ferramenta para a recuperação de dados, este trabalho propõe um framework que permita a recuperação de informações em metadados associados a outros dados. Inicialmente, o framework trata metadados associados a arquivos de imagens, por serem mais comuns e numerosos. Este framework reconhece três tipos de informações associadas às imagens, que são os metadados textuais, os metadados embutidos e as informações sobre coloração da imagem, fornecendo quatro tipos de buscas para a recuperação dessa informação, que são buscas simples (retornando palavras estruturalmente iguais à que foi fornecida), por similaridade (retornando palavras estruturalmente semelhantes), por sinônimo (retornando palavras que possuem o mesmo sentido da que foi fornecida) e por substring (retornando palavras que contenham a palavra fornecida). O framework foi implementado afim de por em prática as definições teóricas e permitir a realização de testes de revocação (fração dos documentos relevantes,*

observada dentre os documentos examinados) e precisão (fração dos documentos já examinados que são relevantes), que comprovaram um resultado satisfatório no retorno dos dados.

Palavras-chave: Indexação, Busca de dados, Metadados, Metadados Embutidos, Metadados Textuais

1. Introdução

A explosão da Internet e dos repositórios de conteúdos digitais tem trazido uma grande quantidade de dados ao alcance de todos. Com o tempo, a quantidade de dados disponíveis se tornou tão vasta que surgiu a necessidade de formas alternativas e mais dinâmicas de encontrar a informação (GOSPODNETIC & HATCHER, 2005). Nesse contexto, o metadado acabou se tornando popular e amplamente utilizado.

O metadado consiste em um grupo de informações sobre o dado ao qual ele está associado, auxiliando assim na organização desses dados. Estes incluem elementos de descrição do conteúdo dos dados e qualquer informação relevante para a recuperação dos seus conteúdos. Podem ser destacadas as seguintes vantagens na utilização e disponibilização de metadados: estabelecimento de padrões de dados diante da heterogeneidade de informações contidas em rede, como por exemplo, a Internet; facilidade e maior precisão na recuperação das informações desejadas; troca de informações entre aplicações e organizações (GARCIA, MOURA, & CAMPOS, 1999).

Um tipo de metadado, dentre os arquivos de imagens, mais recentemente difundido principalmente por câmeras digitais, é o metadado em imagens que podem ser tanto embutido na própria imagem como em arquivos externos à imagem. O modelo do metadado embutido depende do formato da imagem, por exemplo, em imagens no formato de compactação JPEG ou sem compactação no formato TIFF, o modelo do metadado embutido é o chamado EXIF (CIPA DC-008-Translation-2012). Como entre os arquivos de imagens o formato mais comum é JPG ou JPEG, o metadado EXIF se tornou um padrão. Este metadado torna acessível as informações que envolvem uma imagem, mas que não estão evidentes na própria imagem, como por exemplo, a data e hora em que ela foi capturada e até mesmo seu geo-posicionamento. Já o metadado em arquivo externo, normalmente, é um arquivo de texto, que pode ter ou não a formatação baseada na linguagem de marcação XML (W3C.Org). Nesse arquivo externo podem ser acrescentadas outras informações pelo próprio usuário de maneira a dar um sentido semântico e pessoal a esse metadado, como por exemplo, a informação que a foto foi capturada no aniversário de um amigo. Porém, nem sempre esse metadado está completo e no mesmo formato, dificultando a manipulação e recuperação dessa informação.

Existem algumas ferramentas que disponibilizam recursos para auxiliar nesse processo de recuperação da informação, mas essas ferramentas estão ou em bibliotecas, como no Image Metadata JPEG (CPAN.ORG) ou embutidos em outros grupos de ferramentas comerciais em forma de um plugin, assim como no software Zoom Search Engine (ZOOM_WRENSOFT.COM) com o acréscimo do plugin ImageInfo (WRENSOFT_PLUGINS.COM).

A proposta inicial deste trabalho é oferecer um framework de busca a informações em arquivos de metadados associados a arquivos multimídia, totalmente independente de

qualquer outra ferramenta e totalmente gratuita, disponibilizando operações de indexação e busca, integrando informações de três fontes distintas do arquivo: estrutura (tamanho, formato, coloração, etc), metadado embutido(data de captura, câmera utilizada, etc) e metadado textual (dados cadastrados pelo próprio usuário que atribui a semântica do arquivo, como por exemplo, a ocasião em que a foto foi tirada). Essas informações podem ser cruzadas livremente, utilizando-se apenas uma ou combinações de várias ou até mesmo todas as características dos arquivos extraídas das fontes citadas acima. Podem ser utilizadas buscas simples, por similaridade, por sinônimos, por substring ou até mesmo uma combinação das mesmas. Sendo que a busca simples retorna palavras estruturalmente iguais à que foi fornecida, a busca por similaridade retorna palavras estruturalmente semelhantes, a busca por sinônimos retorna palavras que possuem o mesmo sentido da que foi fornecida e a busca por substring retorna palavras que contenham a palavra fornecida.

Foi realizada a implementação dessas funcionalidades em uma ferramenta chamada Metadata Ingester (MI), onde podem ser executados os testes sobre todas as funcionalidades propostas pelo framework.

2. Fundamentação Teórica

2.1. Indexação

As estruturas de índices tipicamente fornecem um caminho secundário de acesso, que proveem formas alternativas de acessar os dados sem afetar o local físico desses dados em disco. Eles permitem um acesso eficiente aos dados baseando-se nos campos indexados que são usados para construir o índice (NAVATHE & ELMASRI, 1999, pp. 134, Cap.6).

2.2. Busca

Para se recuperar uma palavra de um documento é possível simplesmente vasculhar cada documento verificando se essa palavra está ou não presente. Porém essa abordagem é ineficiente quando o conjunto de documentos a ser pesquisado é de grande escala ou mesmo se cada documento for extenso. Para evitar problemas dessa natureza, todos os *Search Engines* utilizam, como um conceito básico, o recurso de índices, processando a informação original para um eficiente alvo de referência cruzada, a fim de facilitar a rápida busca posterior, eliminando a necessidade de uma lenta busca sequencial (GOSPODNETIC & HATCHER, 2005).

2.3. Apache Lucene

Apache Lucene (LUCENE_APACHE.ORG) é uma biblioteca para indexação e busca disponibilizada pela Apache Software Foundation (APACHE.ORG). É um membro da popular família de projetos da Apache Jakarta (JAKARTA.ORG) e é atualmente, e tem sido por alguns anos, a mais popular biblioteca de Recuperação de Informações Java gratuita. Utiliza modernos algoritmos e conceitos na indexação, como índices invertidos, e recuperação de resultados, como escalonamento sobre cluster. O Lucene disponibiliza um simples, porém poderoso, conjunto de funcionalidades que requer o mínimo de entendimento de indexação textual e buscas. Apenas é necessário aprender a

manipular tais recursos para a integração com a aplicação a qual o Lucene é utilizado (GOSPODNETIC & HATCHER, 2005).

2.4. Metadado associados à imagens

2.4.1.Metadado Textual

A associação de metadados textuais é uma das soluções mais utilizadas para a estruturação do conteúdo de arquivos de multimídia. Em geral, estes metadados são feitos a partir da compreensão e da necessidade do usuário que está manipulando o arquivo. Embora as descrições obtidas possam ser imprecisas e incompletas, elas adicionam muito mais conhecimento e semântica ao conteúdo de um arquivo multimídia do que poderia ser obtido através das técnicas atuais de visão computacional e de reconhecimento de padrões de imagem e áudio (NETO, ROCHA, & SANTOS, 2005).

2.4.2.Metadado Embutido

Cada tipo de arquivo multimídia possui um metadado básico acoplado a ele e que dispõe de informações essenciais sobre este arquivo, como data da última modificação, formato, tamanho, dentre outros. Para arquivos de imagens não é diferente, porém, além desse metadado básico existem extensões que melhor descrevem este arquivo. Para arquivos com formato de compactação JPEG ou TIFF é utilizado o metadado Extended Information File (ExIF). O Exif está contido como um cabeçalho, presente geralmente em fotografias tiradas com uma máquina digital, mas pode ser acrescentado posteriormente através de ferramentas especializadas. Este metadado armazena informações, tais como qual a fabricante da câmera que tirou a fotografia, ou até mesmo o modo em que a foto foi tirada. Estas informações normalmente são utilizadas em investigações criminais (CIPA DC-008-Translation-2012). O formato Exif foi criado pela Japan Electronic Industry Development Association (JEITA) e é referenciado como formato preferencial para câmeras digitais na ISO 12234-1. Este metadado é armazenado em uma área chamada de segmento de do arquivo (CIPA DC-008-Translation-2012).

3. Trabalhos Relacionados

3.1. Image Metadata JPEG

Image Metadata JPEG (CPAN.ORG) é uma biblioteca utilizada basicamente para manusear os metadados de um arquivo, ou seja, editar, remover ou acrescentar. Pode ser acrescentada livremente em uma aplicação sendo desenvolvida, porém apenas identifica informações de metadados em imagens no formato JPEG, ou seja, identifica apenas metadados do tipo Exif. O objetivo desta biblioteca é apenas manipular metadados, não incluindo os recursos de indexação e busca desses dados, nem muito menos lidar com informações da estrutura da imagem ou de arquivos de metadados textuais.

3.2. Zoom Search Engine com Plugin ImageInfo

O Zoom Search Engine (ZOOM_WRENSOFT.COM) permite que sejam acrescentadas buscas em softwares desenvolvidos em sites, intranet ou CD/DVD. Acrescentando o plugin ImageInfo (WRENSOFT_PLUGINS.COM) é possível buscar informações em

imagens nos formatos TIFF, JPG, GIF e PNG. Porém, esta ferramenta tem uma restrição de gratuidade, ou seja, se o site possui mais de 50 páginas a ferramenta passa a ser paga. O objetivo desta ferramenta é indexação e busca em sites específicos. Sendo assim, não trabalha com informações de um arquivo, como os metadados e muito menos a utilização desses dados em diferentes tipos de buscas.

3.3. Sadman Software Search

O objetivo da ferramenta SadMan Software Search (SADMAN.COM) é buscar por partes de palavras em determinados arquivos usando buscas com combinações lógicas (AND, OR e AND NOT) com até duas palavras, por expressões regulares. Além disso, ela permite a visualização da linha que retornou da busca definida pelo usuário, além de gerar novas buscas utilizando os resultados de uma busca anterior. Porém, o objetivo é apenas encontrar um arquivo que contém determinada palavra e/ou palavras no seu conteúdo, não contemplando outras características como a sua data de criação ou sua coloração, ou mesmo se existe um arquivo de metadado textual associado a esse documento.

3.4. Tabela Comparativa

	Metadata Ingester	SadMan	Zoom Search Engine	Image Metadata JPEG
1 - Gratuito	Sim	Não, mas possui versão de demonstração	Parcialmente gratuito	Sim
2 - Precisa de *plugin*	Não	Não	Sim	Não
3 - Trabalha sobre que tipo de informação	**Arquivos de imagens, Metadados Exif e Textuais**	Arquivos de texto	Sites, arquivos de imagens e arquivos textuais	Arquivos de imagens no formato JPEG
4 - Extrai informações sobre a coloração da imagem	**Sim**	Não	Não	Não
5 - Extrai informações do metadado Exif da imagem	Sim	Não	Não	Sim
6 - Extrai informações do metadado textual	**Sim**	Não	Não	Não
7 - Gera índices	Sim	Não	Sim	Não
8 - Busca pela palavra	Sim	Sim	Sim	Não
9 - Busca pelos sinônimos da palavra	**Sim**	Não	Sim	Não
10 - Busca por palavras semelhantes da palavra	Sim	Sim	Não	Não
11 - Busca por parte da palavra	Sim	Sim	Sim	Não
12 - Busca por informações de coloração da imagem	**Sim**	Não	Não	Não
13 - Busca composta entre as 5 buscas anteriores	**Sim**	Não	Não	Não

Figura 1 - Comparação entre Funcionalidades dos Aplicativos Relacionados

Na Figura 1 é possível visualizar as diferenças e semelhanças entre os sistemas e também verificar que dentro do objetivo proposto pelo trabalho, o Metadata Ingester (MI) atende a todos os requisitos. A principal diferença entre o MI e os outros sistemas é a possibilidade de extração de dados de metadados textuais e informações de

coloração de imagem e a combinação de buscas pela palavra, por similaridade, por sinônimos e por substring sobre essas informações.

4. Metadata Ingester

4.1. Visão Geral

O Metadata Ingester (MI) é um sistema de busca e indexação de arquivos de metadados no formato de texto plano e em formato XML que utiliza a biblioteca de recuperação de informação Apache Lucene (LUCENE_APACHE.ORG) como base. O MI executa uma prévia interpretação dos arquivos textuais e o resultado é utilizado tanto pelo Lucene quanto pelo próprio MI para a criação dos arquivos de indexação. Finalizado o processo de criação de índices o MI recebe buscas do usuário, faz uma busca prévia nos seus arquivos de índices, formata os dados e envia ao Lucene para a recuperação da informação nos índices criados.

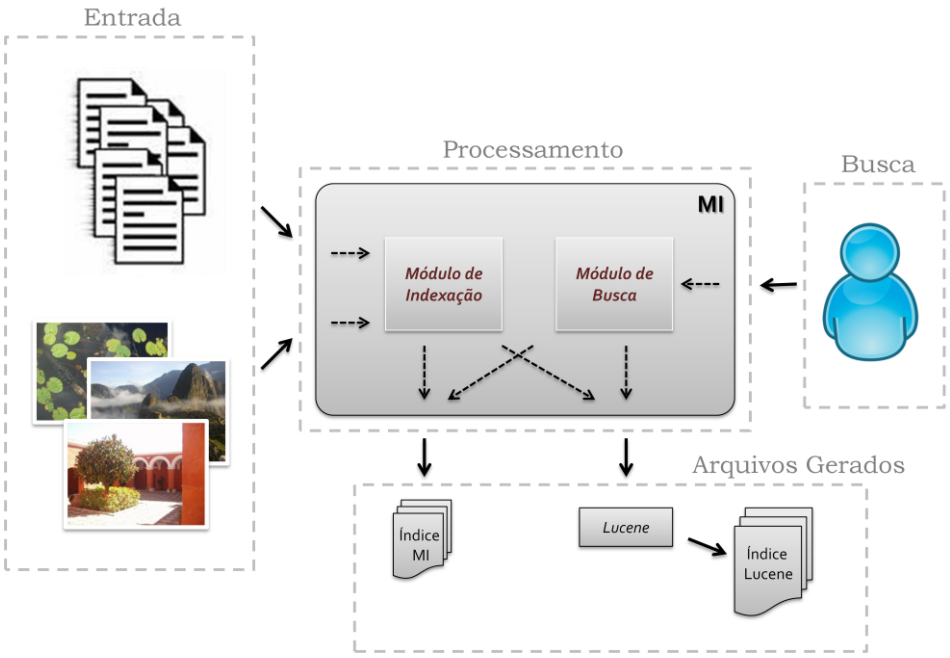

Figura 2 - Visão Geral Metadata Ingester

Na Figura 2 está representada a visão geral do Metadata Ingester (MI). Pode-se subdividir a imagem em 4 (quatro) áreas distintas:

- Entrada: os arquivos de metadados textuais e de imagens são fornecidos ao MI.

- Processamento: os índices serão gerados e as buscas serão executadas.

- Busca: o usuário configura e inicia a execução das buscas.

- Arquivos Gerados: os arquivos de índices serão salvos em memória para que possam ser utilizados nas buscas dos usuários.

4.2. Módulo Indexação

Este módulo trabalha com 4 etapas distintas: na Primeira Etapa são lidos e interpretados os arquivos de metadados textuais. Na Segunda Etapa são extraídos os metadados embarcados na imagem. Na Terceira Etapa os dados sobre a coloração da imagem são coletados, e na Quarta Etapa todos esses dados são usados para a geração dos arquivos de índices.

4.3. Módulo Busca

O módulo de busca trata a combinação de 4 (quatro) tipos distintos de busca (simples, que retorna palavras estruturalmente iguais à que foi fornecida, substring, que retorna palavras que contenham a palavra fornecida, similaridade, que retorna palavras estruturalmente semelhantes e sinônimo, que retorna palavras que possuem o mesmo sentido da que foi fornecida) com 3 (três) diferentes fontes de dados (arquivos de metadados textuais, metadados embutidos e informações de coloração), permitindo ao usuário uma grande liberdade na criação de buscas. O princípio básico da busca tradicional é: a partir de um valor determinado pelo usuário descobrir que informações devem ser retornadas como resultado. No caso do projeto proposto as informações a serem retornadas ao usuário estão armazenadas em forma de índices gerados pelo Lucene e pelo MI, e, portanto, para recuperá-las deve-se pesquisar nesses índices.

5. Experimentos

Os experimentos foram executados com o propósito de validar a revocação e precisão do MI. Precisão é a fração dos documentos já examinados que são relevantes, e revocação é a fração dos documentos relevantes observada dentre os documentos examinados (CARDOSO, 2000). Para o cálculo da média de revocação e precisão, foram coletados os resultados de 40 diferentes configurações de busca, distribuídas da seguinte forma: 10 Buscas Simples, 10 Buscas por Similaridade, 10 Buscas por Sinônimos e 10 Buscas por Substring. A Figura 3 representa os resultados obtidos após os experimentos.

Resultados das Buscas		
Tipo de Busca	Média Revocação	Média Precisão
Simples	0,85	0,80
Similaridade	0,84	0,81
Substring	0,82	0,81
Sinônimos	0,85	0,81

Figura 3 - Médias Resultantes dos Experimentos para Revocação e Precisão

As informações coletadas nas 40 buscas foram armazenadas em tabelas que possuem as seguintes colunas e estão disponíveis nos Anexos I, II, III e IV:

- ID: representa um identificador para busca a ser realizada.

- Fontes: representa as fontes de dados que serão utilizadas na busca.

- Dados da Busca: representa quais as informações que serão configuradas na busca.

- Relevantes na Base: representa a quantidade de arquivos que deveriam retornar da execução da busca.

- Retorno: representa a quantidade de arquivos que retornaram na execução da busca.

- Relevantes no Retorno: representa a quantidade de arquivos relevantes na base que retornaram no resultado da execução da busca.

- Precisão: calculada pela seguinte fórmula (Relevantes no Retorno)/Retorno

- Revocação: calculada pela fórmula (Relevantes no Retorno)/(Relevantes na Base)

6. Conclusão

A proposta deste projeto, descrita em mais detalhes na introdução deste documento, é fornecer um framework para indexação e busca sobre informações de imagens e aos metadados associados a ela. Permitindo a busca simples (que retorna arquivos que contenham palavras estruturalmente iguais à que foi fornecida), por similaridade (que retorna arquivos que contenham palavras estruturalmente semelhantes), por sinônimos (que retorna arquivos que contenham palavras que possuem o mesmo sentido da que foi fornecida) e por substring (que retorna palavras que contenham a palavra fornecida na sua estrutura) sobre informações recuperadas de metadados textuais, metadados embutidos e sobre a coloração da mesma. Além de gerar a implementação desse framework, com o objetivo de demonstrar a viabilidade da proposta.

Durante a execução do projeto foi possível verificar a lacuna que existe no mercado de sistemas que atendam às características propostas para o projeto através de pesquisas e de comparações entre os sistemas existentes. E neste contexto o MI pode ser considerado um sistema completo para auxiliar o processo de criação de índices e pesquisas combinadas sobre diversas fontes de dados.

Mesmo já trabalhando com uma grande quantidade de informações, o MI foi desenvolvido para que pudesse ser possível a sua evolução, ou seja, é possível evoluir o projeto atual para acrescentar fontes de dados e tipos de buscas diferenciadas sem qualquer problema na implementação dessas funcionalidades.

7. Referências

GOSPODNETIC, O., & HATCHER, E. (2005). Lucene in Action. Greenwich: Manning Publications.

GARCIA, S. d., MOURA, A. M., & CAMPOS, M. L. (1999). Metadados para Documentação e Recuperação de Imagens. Dissertação (Mestrado em Sistemas e Computação) – Instituto Militar de Engenharia . Rio de Janeiro.

CIPA DC-008-Translation-2012, C. (s.d.). Standart of th Camera & Imaging Products Association. Acesso em 20 de 02 de 2013, disponível em CIPA: http://www.cipa.jp/english/hyoujunka/kikaku/pdf/DC-008-2012_E.pdf

W3C.Org. (s.d.). Acesso em 03 de 11 de 2011, disponível em World Wide Web Consortium: http://www.w3.org

CPAN.ORG. (s.d.). Acesso em 11 de 12 de 2011, disponível em http://search.cpan.org/~bettelli/Image-MetaData-JPEG-0.153/lib/Image/MetaData/JPEG.pod

ZOOM_WRENSOFT.COM. (s.d.). Acesso em 11 de 12 de 2011, disponível em WRENSOFT: http://www.wrensoft.com/zoom/index.html

WRENSOFT_PLUGINS.COM. (s.d.). Acesso em 11 de 11 de 2012, disponível em http://www.wrensoft.com/zoom/plugins.html

NAVATHE, S., & ELMASRI, R. (1999). Fundamentals of Database Systems, Third Edition. Addison-Wesley.

GOSPODNETIC, O., & HATCHER, E. (2005). Lucene in Action. Greenwich: Manning Publications.

LUCENE_APACHE.ORG. (s.d.). Acesso em 2012 de 06 de 10, disponível em Site da Apache Software Fundation: http://lucene.apache.org/

APACHE.ORG. (s.d.). Acesso em 15 de 11 de 2012, disponível em The Apache Software Foundation: http://www.apache.org/

JAKARTA.ORG. (s.d.). Acesso em 15 de 11 de 2012, disponível em http://jakarta.apache.org/

NETO, A. R., ROCHA, E. M., & SANTOS, C. A. (2005). Uma proposta de avaliação baseada em anotações de vídeos digitais. V Workshop de Informática Médica .

CIPA DC-008-Translation-2012, C. (s.d.). Standart of th Camera & Imaging Products Association. Acesso em 20 de 02 de 2013, disponível em CIPA: http://www.cipa.jp/english/hyoujunka/kikaku/pdf/DC-008-201

JEITA. (s.d.). Acesso em 14 de 11 de 2012, disponível em Japan Electronic and Information Technology Industries Association: http://www.jeita.or.jp/english

CPAN.ORG. (s.d.). Acesso em 11 de 12 de 2011, disponível em http://search.cpan.org/~bettelli/Image-MetaData-JPEG-0.153/lib/Image/MetaData/JPEG.pod

ZOOM_WRENSOFT.COM. (s.d.). Acesso em 11 de 12 de 2011, disponível em WRENSOFT: http://www.wrensoft.com/zoom/index.html

WRENSOFT_PLUGINS.COM. (s.d.). Acesso em 11 de 11 de 2012, disponível em http://www.wrensoft.com/zoom/plugins.html

SADMAN.COM. (s.d.). Acesso em 31 de 10 de 2012, disponível em SadMan Search: http://www.sadmansoftware.com/search/index.html

CARDOSO, O. N. (2000). Recuperação de informação. Infocomp: Revista de Computação da UFLA .

Apêndice B – Classe *CtrlMainFrame*

```
package tcc.tbfm.control;

import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.util.ArrayList;
import java.util.Calendar;
import java.util.Hashtable;

import javax.swing.JOptionPane;

import tcc.tbfm.model.api.MetadataIngester;
import tcc.tbfm.model.classesUtils.MIConstants;
import tcc.tbfm.model.classesUtils.MIKeys;
import tcc.tbfm.model.objectsUtils.MetadataIngesterInfo;
import tcc.tbfm.model.objectsUtils.QueryMI;
import tcc.tbfm.view.MainFrame;
import tcc.tbfm.view.OptionTag;

public class CtrlMainFrame {

	private MainFrame mainFrame;
	private OptionTag optionTag;
	private MetadataIngester metadataIngester;

	public CtrlMainFrame() {
		mainFrame = new MainFrame();
		optionTag = new OptionTag();
		setActionListeners();
		mainFrame.setVisible(true);
	}

	private void setActionListeners() {
```

```
        mainFrame.getBtnIndexIt().addActionListener(btnIndexItListener());
        mainFrame.getBtnSearchIt().addActionListener(btnSearchItListener());
        mainFrame.getBtnSelectTag().addActionListener(btnSearchTagListener());
        mainFrame.getBtnReuseIndex().addActionListener(btnReuseIndexListener());

        optionTag.getBtnCancel().addActionListener(btnCancelOptionListener());
        optionTag.getBtnOk().addActionListener(btnOkOptionListener());
        optionTag.getBtnSearchTag().addActionListener(btnShowTagListener());

    }

    private ActionListener btnIndexItListener() {
        return new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                // Usar os setCmbColors, setCmbType, setCmbColorMode,
                // setCmbCamera
                if (mainFrame.getTxtMetadataPath().getText().equals("")) {
                    JOptionPane.showMessageDialog(null,
                            "Selecione o caminho para os metadados.");
                } else if (mainFrame.getTxtImagePath().getText().equals("")) {
                    JOptionPane.showMessageDialog(null,
                            "Selecione o caminho para as imagens.");
                } else if (mainFrame.getTxtIndexPath().getText().equals("")) {
                    JOptionPane
                            .showMessageDialog(null,
                                    "Selecione o caminho para o armazenamento dos índices.");
                } else {
                    MetadataIngesterInfo info = new MetadataIngesterInfo(
                            mainFrame.getTxtMetadataPath().getText(), mainFrame
                                    .getTxtImagePath().getText(), mainFrame
                                    .getTxtIndexPath().getText(),
                            mainFrame.isFollow());

                    if (metadataIngester == null)
                        metadataIngester = new MetadataIngester(info);
                    else
                        metadataIngester.setMetadataInfo(info);

                    metadataIngester.index();
```

```
                        mainFrame.setLblResult("Finalizado com Sucesso!");
                    }
                }
            };
    }

    private ActionListener btnReuseIndexListener() {
            return new ActionListener() {
                @Override
                public void actionPerformed(ActionEvent e) {
                    if (mainFrame.getTxtMetadataPath().getText().equals("")) {
                        JOptionPane.showMessageDialog(null,
                                "Selecione o caminho para os metadados.");
                    } else if (mainFrame.getTxtImagePath().getText().equals("")) {
                        JOptionPane.showMessageDialog(null,
                                "Selecione o caminho para as imagens.");
                    } else if (mainFrame.getTxtIndexPath().getText().equals("")) {
                        JOptionPane
                                .showMessageDialog(null,
                                        "Selecione o caminho para o armazenamento dos índices.");
                    } else {
                        MetadataIngesterInfo info = new MetadataIngesterInfo(
                                mainFrame.getTxtMetadataPath().getText(), mainFrame
                                        .getTxtImagePath().getText(), mainFrame
                                        .getTxtIndexPath().getText(),
                                mainFrame.isFollow());

                        if (metadataIngester == null)
                            metadataIngester = new MetadataIngester(info);
                        else
                            metadataIngester.setMetadataInfo(info);

                        mainFrame.setLblResult("Finalizado com Sucesso!");
                    }
                }
            };
    }

    private ActionListener btnSearchItListener() {
            return new ActionListener() {
```

```
@Override
public void actionPerformed(ActionEvent e) {
	Hashtable<String, String> annotationData = new Hashtable<>();
	Hashtable<String, String> imageStringData = new Hashtable<>();
	Hashtable<String, Integer[]> imageIntData = new Hashtable<>();
	Hashtable<String, Calendar[]> imageDateData = new Hashtable<>();
	Hashtable<String, String> searchConfig = new Hashtable<>();
	Hashtable<String, Integer> relevances = new Hashtable<>();
	if (mainFrame.getPathOfResults().equals("")) {
		JOptionPane
				.showMessageDialog(null,
						"Selecione a pasta para armazenar os resultados da busca.");
	} else {
		if (mainFrame.isSearchAnnotation()) {
			if (mainFrame.getTxtWord().equals("")) {
				JOptionPane.showMessageDialog(null,
						"Digite a palavra para a busca.");
			} else {
				String met = mainFrame.getRdBtnTextType();
				if (mainFrame.getTxtTag().getText().equals("")
						&& (met.equals("XML/Texto") || met
								.equals("XML"))) {
					JOptionPane.showMessageDialog(null,
							"Selecione a tag XML para a busca.");
				} else {
					annotationData.put(MIKeys.SEARCH_WORD_VALUE,
							mainFrame.getTxtWord());
					if (met.equals("XML")) {
						searchConfig.put(MIKeys.SEARCH_ANNOT_XML,
								"true");
						searchConfig.put(MIKeys.SEARCH_ANNOT_TXT,
								"false");
						annotationData.put(
								MIKeys.SEARCH_PATH_VALUE, mainFrame
										.getTxtTag().getText());
					} else if (met.equals("Texto")) {
						searchConfig.put(MIKeys.SEARCH_ANNOT_XML,
								"false");
						searchConfig.put(MIKeys.SEARCH_ANNOT_TXT,
								"true");
```

```
} else {
	searchConfig.put(MIKeys.SEARCH_ANNOT_XML,
			"true");
	searchConfig.put(MIKeys.SEARCH_ANNOT_TXT,
			"true");
	annotationData.put(
			MIKeys.SEARCH_PATH_VALUE, mainFrame
					.getTxtTag().getText());
}
if (mainFrame.isOnlySimpleSearch()) {
	searchConfig.put(
			MIKeys.SEARCH_WORD_ONLY_SOURCE,
			"true");
	searchConfig.put(MIKeys.SEARCH_WORD_SIMPLE,
			"true");
} else {
	if (mainFrame.isSimpleSearch()) {
		searchConfig.put(
				MIKeys.SEARCH_WORD_SIMPLE,
				"true");
	}
	if (mainFrame.isSimilaritySearch()) {
		searchConfig.put(
				MIKeys.SEARCH_WORD_SIMILARITY,
				"true");
		annotationData
				.put(MIKeys.SEARCH_WORD_SIMILARITY_VALUE,
						mainFrame.similarity());
	}
	if (mainFrame.isSynonysmSearch()) {
		searchConfig.put(
				MIKeys.SEARCH_WORD_SYNONYSM,
				"true");
	}
	if (mainFrame.isSubstringSearch()) {
		searchConfig.put(
				MIKeys.SEARCH_WORD_SUBSTRING,
				"true");
	}
}
```

```
                }
            }
        }

        if (mainFrame.hasHeigth())
            imageIntData.put(MIKeys.IMAGE_HEIGHT,
                    mainFrame.getTxtHeigth());
        if (mainFrame.hasWidth())
            imageIntData.put(MIKeys.IMAGE_WIDTH,
                    mainFrame.getTxtWidth());
        if (mainFrame.hasSize())
            imageIntData.put(MIKeys.IMAGE_SIZEKB,
                    mainFrame.getTxtSize());
        if (mainFrame.hasXResolution())
            imageIntData.put(MIKeys.IMAGE_X_RESOLUTION,
                    mainFrame.getTxtXResolution());
        if (mainFrame.hasYResolution())
            imageIntData.put(MIKeys.IMAGE_Y_RESOLUTION,
                    mainFrame.getTxtYResolution());

        if (mainFrame.hasPercentBlack())
            imageIntData.put(MIConstants.IMAGE_COLOR_BLACK,
                    mainFrame.getTxtPercentBlack());
        if (mainFrame.hasPercentBlue())
            imageIntData.put(MIConstants.IMAGE_COLOR_BLUE,
                    mainFrame.getTxtPercentBlue());
        if (mainFrame.hasPercentBrown())
            imageIntData.put(MIConstants.IMAGE_COLOR_BROWN,
                    mainFrame.getTxtPercentBrown());
        if (mainFrame.hasPercentGray())
            imageIntData.put(MIConstants.IMAGE_COLOR_GRAY,
                    mainFrame.getTxtPercentGray());
        if (mainFrame.hasPercentGreen())
            imageIntData.put(MIConstants.IMAGE_COLOR_GREEN,
                    mainFrame.getTxtPercentGreen());
        if (mainFrame.hasPercentOrange())
            imageIntData.put(MIConstants.IMAGE_COLOR_ORANGE,
                    mainFrame.getTxtPercentOrange());
        if (mainFrame.hasPercentPink())
            imageIntData.put(MIConstants.IMAGE_COLOR_PINK,
```

```
                    mainFrame.getTxtPercentPink());
        if (mainFrame.hasPercentPurple())
            imageIntData.put(MIConstants.IMAGE_COLOR_PURPLE,
                    mainFrame.getTxtPercentPurple());
        if (mainFrame.hasPercentRed())
            imageIntData.put(MIConstants.IMAGE_COLOR_RED,
                    mainFrame.getTxtPercentRed());
        if (mainFrame.hasPercentWhite())
            imageIntData.put(MIConstants.IMAGE_COLOR_WHITE,
                    mainFrame.getTxtPercentWhite());
        if (mainFrame.hasPercentYellow())
            imageIntData.put(MIConstants.IMAGE_COLOR_YELLOW,
                    mainFrame.getTxtPercentYellow());

        if (mainFrame.hasCreated()) {
            Calendar[] creation = new Calendar[] {
                    getDate(mainFrame.getTxtCreatedFrom()),
                    getDate(mainFrame.getTxtCreatedTo()) };
            imageDateData.put(MIKeys.IMAGE_CREATION, creation);
        }
        if (mainFrame.hasModification()) {
            Calendar[] modification = new Calendar[] {
                    getDate(mainFrame.getTxtModificationFrom()),
                    getDate(mainFrame.getTxtModificationTo()) };
            imageDateData.put(MIKeys.IMAGE_LAST_MODIFICATION,
                    modification);
        }
        if (!mainFrame.getCmbColor().equals(
                MIConstants.IMAGE_NOT_SELECTED_INFO)) {
            imageStringData.put(MIKeys.IMAGE_COLOR_TYPE,
                    mainFrame.getCmbColor());
        }
        if (!mainFrame.getCmbType().equals(
                MIConstants.IMAGE_NOT_SELECTED_INFO)) {
            imageStringData.put(MIKeys.IMAGE_FORMAT,
                    mainFrame.getCmbType());
        }
        if (!mainFrame.getCmbPredominantColor().equals(
                MIConstants.IMAGE_NOT_SELECTED_INFO)) {
            imageStringData.put(MIKeys.IMAGE_PREDOMINANT_COLOR,
```

```
                    mainFrame.getCmbPredominantColor());
        }
        if (!mainFrame.getCmbCamera().equals(
                    MIConstants.IMAGE_NOT_SELECTED_INFO)) {
            imageStringData.put(MIKeys.IMAGE_CAMERA,
                    mainFrame.getCmbCamera());
        }
        if (imageIntData.size() > 0 || imageDateData.size() > 0
                    || imageStringData.size() > 0) {
            searchConfig.put(MIKeys.SEARCH_IMAGE_DATA, "true");
        } else {
            searchConfig.put(MIKeys.SEARCH_IMAGE_DATA, "false");
        }

        searchConfig.put(MIKeys.SEARCH_NUMBER_OF_RESULTS,
                    mainFrame.getNumberOfResults());
        searchConfig.put(MIKeys.SEARCH_PATH_OF_RESULTS,
                    mainFrame.getPathOfResults());

        relevances.put(MIConstants.IMAGE_COLOR_BLACK,
                    mainFrame.getCmbRelevBlack());
        relevances.put(MIConstants.IMAGE_COLOR_BLUE,
                    mainFrame.getCmbRelevBlue());
        relevances.put(MIConstants.IMAGE_COLOR_BROWN,
                    mainFrame.getCmbRelevBrown());
        relevances.put(MIKeys.IMAGE_CAMERA,
                    mainFrame.getCmbRelevCamera());
        relevances.put(MIKeys.IMAGE_COLOR_TYPE,
                    mainFrame.getCmbRelevColorType());
        relevances.put(MIKeys.IMAGE_CREATION,
                    mainFrame.getCmbRelevCreated());
        relevances.put(MIKeys.IMAGE_FORMAT,
                    mainFrame.getCmbRelevFormat());
        relevances.put(MIConstants.IMAGE_COLOR_GRAY,
                    mainFrame.getCmbRelevGray());
        relevances.put(MIConstants.IMAGE_COLOR_GREEN,
                    mainFrame.getCmbRelevGreen());
        relevances.put(MIKeys.IMAGE_HEIGHT,
                    mainFrame.getCmbRelevHeight());
        relevances.put(MIKeys.IMAGE_LAST_MODIFICATION,
```

```
                        mainFrame.getCmbRelevModif());
                relevances.put(MIConstants.IMAGE_COLOR_ORANGE,
                        mainFrame.getCmbRelevOrange());
                relevances.put(MIConstants.IMAGE_COLOR_PINK,
                        mainFrame.getCmbRelevPink());
                relevances.put(MIKeys.IMAGE_PREDOMINANT_COLOR,
                        mainFrame.getCmbRelevPredominant());
                relevances.put(MIConstants.IMAGE_COLOR_PURPLE,
                        mainFrame.getCmbRelevPurple());
                relevances.put(MIConstants.IMAGE_COLOR_RED,
                        mainFrame.getCmbRelevRed());
                relevances.put(MIKeys.SEARCH_WORD_SIMILARITY,
                        mainFrame.getCmbRelevSimilarity());
                relevances.put(MIKeys.SEARCH_WORD_SIMPLE,
                        mainFrame.getCmbRelevSimple());
                relevances.put(MIKeys.IMAGE_SIZEKB,
                        mainFrame.getCmbRelevSize());
                relevances.put(MIKeys.SEARCH_WORD_SUBSTRING,
                        mainFrame.getCmbRelevSubstring());
                relevances.put(MIKeys.SEARCH_WORD_SYNONYSM,
                        mainFrame.getCmbRelevSynonysm());
                relevances.put(MIConstants.IMAGE_COLOR_WHITE,
                        mainFrame.getCmbRelevWhite());
                relevances.put(MIKeys.IMAGE_WIDTH,
                        mainFrame.getCmbRelevWidth());
                relevances.put(MIKeys.IMAGE_X_RESOLUTION,
                        mainFrame.getCmbRelevXResol());
                relevances.put(MIKeys.IMAGE_Y_RESOLUTION,
                        mainFrame.getCmbRelevYResol());
                relevances.put(MIConstants.IMAGE_COLOR_YELLOW,
                        mainFrame.getCmbRelevYellow());

                QueryMI query = new QueryMI(searchConfig, annotationData,
                        imageStringData, imageIntData, imageDateData,
                        relevances);
                metadataIngester.search(query);

            }
        }
};
```

```
}

private ActionListener btnSearchTagListener() {
	return new ActionListener() {
		@Override
		public void actionPerformed(ActionEvent e) {
			optionTag.setVisible(true);
		}
	};
}

private ActionListener btnCancelOptionListener() {
	return new ActionListener() {
		@Override
		public void actionPerformed(ActionEvent e) {
			optionTag.getTxtTag().setText("");
			optionTag.getBoxSimilarity().setSelected(false);
			optionTag.getBoxSimple().setSelected(false);
			optionTag.getBoxSubstring().setSelected(false);
			optionTag.getBoxSynonysm().setSelected(false);
			optionTag.setVisible(false);
		}
	};
}

private ActionListener btnOkOptionListener() {
	return new ActionListener() {
		@Override
		public void actionPerformed(ActionEvent e) {

			if (optionTag.getLstTags().getSelectedItem() == null) {
				JOptionPane.showMessageDialog(null,
						"Selecione uma tag da lista.");
			} else {
				mainFrame.getTxtTag().setText(
						optionTag.getLstTags().getSelectedItem());
				optionTag.getTxtTag().setText("");
				optionTag.getBoxSimilarity().setSelected(false);
				optionTag.getBoxSimple().setSelected(false);
				optionTag.getBoxSubstring().setSelected(false);
```

```
                        optionTag.getBoxSynonysm().setSelected(false);
                        optionTag.setVisible(false);
                    }
                }
        };
}

private ActionListener btnShowTagListener() {
        return new ActionListener() {
                @Override
                public void actionPerformed(ActionEvent e) {
                    if (optionTag.getTxtTag().getText().equals("")) {
                        JOptionPane.showMessageDialog(null,
                                "Digite a palavra para a busca.");
                    } else if (optionTag.hasSearchModeSelected()) {
                        Hashtable<String, String> annotationData = new Hashtable<String, String>();
                        Hashtable<String, String> config = new Hashtable<String, String>();
                        config.put(MIKeys.SEARCH_OF_POSSIBLES_PATHS, "true");
                        annotationData.put(MIKeys.SEARCH_PATH_VALUE, optionTag
                                .getTxtTag().getText());
                        if (optionTag.isOnlySimpleSearch()) {
                            config.put(MIKeys.SEARCH_PATH_ONLY_SOURCE, "true");
                        } else {
                            if (optionTag.isSimpleSearch()) {
                                config.put(MIKeys.SEARCH_PATH_SIMPLE, "true");
                            }
                            if (optionTag.isSimilaritySearch()) {
                                config.put(MIKeys.SEARCH_PATH_SIMILARITY, "true");
                                annotationData.put(
                                        MIKeys.SEARCH_PATH_SIMILARITY_VALUE,
                                        optionTag.similarity());
                            }
                            if (optionTag.isSynonysmSearch()) {
                                config.put(MIKeys.SEARCH_PATH_SYNONYSM, "true");
                            }
                            if (optionTag.isSubstringSearch()) {
                                config.put(MIKeys.SEARCH_PATH_SUBSTRING, "true");
                            }
                        }
                        QueryMI query = new QueryMI(config, annotationData, null,
```

```
                                        null, null, null);
                                ArrayList<String> tags = metadataIngester.search(query)
                                        .getHitsPath();
                                optionTag.getLstTags().removeAll();
                                for (String string : tags) {
                                        optionTag.getLstTags().add(string);
                                }

                        } else {
                                JOptionPane.showMessageDialog(null,
                                        "Selecione uma configuração de busca.");
                        }
                }
        };
    }

    public Calendar getDate(String dateString) {
        Calendar c = Calendar.getInstance();
        String[] date = dateString.split("/");
        int day = Integer.parseInt(date[0]);
        int month = Integer.parseInt(date[1]) - 1;
        int year = Integer.parseInt(date[2]);
        int hour = 0;
        int minutes = 0;
        int seconds = 0;
        c.set(year, month, day, hour, minutes, seconds);
        return c;
    }

}
```

Apêndice C – Classe *IndexModule*

```
package tcc.tbfm.model.api;

import tcc.tbfm.model.objectsUtils.LuceneIndex;
import tcc.tbfm.model.objectsUtils.MetadataIngesterInfo;

public class IndexModule {

	private LuceneIndex luceneUtilsIndex;

	public IndexModule(MetadataIngesterInfo metadataInfo) {
		luceneUtilsIndex = new LuceneIndex(metadataInfo);
	}

	public void setMetadataInfo(MetadataIngesterInfo info) {
		luceneUtilsIndex.setMetadataInfo(info);
	}

	public void startIndex() {
		long start = System.currentTimeMillis();
		System.out.println("Start:" + start);
		luceneUtilsIndex.indexIt();
		long elapsedTimeMillis = System.currentTimeMillis() - start;
		float elapsedTimeHour = elapsedTimeMillis / (60 * 60 * 1000F);
		System.out.println("Finished: " + elapsedTimeHour + "hours");
	}

}
```

Apêndice D – Classe *SearchModule*

```
package tcc.tbfm.model.api;

import java.util.ArrayList;
import tcc.tbfm.model.classesUtils.HelperFunctions;
import tcc.tbfm.model.classesUtils.MIConstants;
import tcc.tbfm.model.objectsUtils.HitsMI;
import tcc.tbfm.model.objectsUtils.LuceneSearch;
import tcc.tbfm.model.objectsUtils.MetadataIngesterInfo;
import tcc.tbfm.model.objectsUtils.QueryMI;
import uk.ac.shef.wit.simmetrics.similaritymetrics.AbstractStringMetric;
import uk.ac.shef.wit.simmetrics.similaritymetrics.MongeElkan;

public class SearchModule {

	private LuceneSearch luceneUtilsSearch;
	MetadataIngesterInfo info;

	public SearchModule(MetadataIngesterInfo info) {
		luceneUtilsSearch = new LuceneSearch(info);
		setMetadataInfo(info);
	}

	public void setMetadataInfo(MetadataIngesterInfo info) {
		luceneUtilsSearch.setMetadataInfo(info);
		this.info = info;
	}

	public HitsMI search(QueryMI queryMI) {
		HitsMI hits = new HitsMI(queryMI);
		if (queryMI.isSearchOfPossiblesPaths()) {
			hits.addHitsPath(getPathsToSearch(queryMI));
		} else {
			if (queryMI.isSearchWordOnlySource()) {
				hits.addHitsSimple(luceneUtilsSearch.simpleSearch(queryMI));
```

```
			} else {
				// Busca Simples
				if (queryMI.isSearchWordSimple())
					hits.addHitsSimple(luceneUtilsSearch.simpleSearch(queryMI));

				// Busca por Sinônimos
				if (queryMI.isSearchWordSynonyms())
					hits.addHitsSynonysm(luceneUtilsSearch
							.synonismSearch(queryMI));

				// Busca por Substring
				if (queryMI.isSearchWordSubstring())
					hits.addHitsSubstring(luceneUtilsSearch
							.substringSearch(queryMI));

				// Busca por Similaridade
				if (queryMI.isSearchWordSimilarity())
					hits.addHitsSimilarity(luceneUtilsSearch
							.similaritySearch(queryMI));
			}

			// Busca por dados da imagem
			if (queryMI.isSearchImageData())
				hits.addHitsImage(luceneUtilsSearch.imageSearch(queryMI));

		}

		return hits;

	}

	private ArrayList<String> getPathsToSearch(QueryMI query) {
		// TODO: mostrar o índice de similaridade de cada path
		ArrayList<String> pathsResult = new ArrayList<String>();
		ArrayList<String> tagsToSearch = new ArrayList<String>();
		String[] nodes = query.getPathValueToSearch().split("//");

		if (nodes.length == 1) {
			tagsToSearch.add(query.getPathValueToSearch());
		} else if (nodes.length > 1) {
```

```
                for (int i = 0; i < nodes.length; i++) {
                        tagsToSearch.add(nodes[i]);
                }
        }

        if (query.isSearchPathSubstring()) {
                HelperFunctions.addAll(
                                getPathsBySubstring(query.getPathValueToSearch()),
                                pathsResult);
        }

        if (query.isSearchPathSynonyms()) {
                for (String tagToSearch : tagsToSearch) {
                        HelperFunctions.addAll(getPathsBySynonysm(tagToSearch),
                                        pathsResult);
                }
        }

        if (query.isSearchPathSimilarity()) {
                for (String tagToSearch : tagsToSearch) {
                        HelperFunctions.addAll(
                                        getPathsBySimilarity(tagToSearch,
                                                        query.pathSimilarity()), pathsResult);
                }
        }

        return pathsResult;
}

private ArrayList<String> getPathsBySynonysm(String tag) {
        ArrayList<String> synonysm = HelperFunctions.getSynonysm(tag);
        ArrayList<String> paths = new ArrayList<String>();
        for (String string : synonysm) {
                paths.addAll(getPathsBySubstring(string));
        }
        return paths;
}

private ArrayList<String> getPathsBySubstring(String parcialPath) {
        ArrayList<String> paths = restoreTagList();
```

```
		ArrayList<String> totalPaths = new ArrayList<String>();
		for (String path : paths) {
			if (path.toUpperCase().indexOf(parcialPath.toUpperCase()) != -1) {
				totalPaths.add(path);
			}
		}
		return totalPaths;
	}

	private ArrayList<String> getPathsBySimilarity(String tagSource,
			float similarity) {
		// TODO: PESQUISAR SOBRE MONGO DB!!!
		ArrayList<String> resultPaths = new ArrayList<String>();
		ArrayList<String> originsPaths = restoreTagList();
		AbstractStringMetric metric = new MongeElkan();
		// verifica a tag e o caminho totais
		for (String path : originsPaths) {
			float simil = metric.getSimilarity(path, tagSource);
			if (simil >= similarity) {
				resultPaths.add(path);
			}
		}
		return resultPaths;
	}

	@SuppressWarnings("unchecked")
	private ArrayList<String> restoreTagList() {
		return (ArrayList<String>) HelperFunctions.getObject(info
				.getFolderIndex().toString()
				+ MIConstants.DEFAULT_INDEX_FOLDER_XML
				+ MIConstants.DEFAULT_INDEX_TAG_FILE);
	}

}
```

Apêndice E – Classe *MetadataIngester*

```
package tcc.tbfm.model.api;

import tcc.tbfm.model.objectsUtils.HitsMI;
import tcc.tbfm.model.objectsUtils.MetadataIngesterInfo;
import tcc.tbfm.model.objectsUtils.QueryMI;

public class MetadataIngester {

	private SearchModule searchModule;
	private IndexModule indexModule;
	private MetadataIngesterInfo metadataInfo;

	public MetadataIngester(MetadataIngesterInfo info) {
		this.metadataInfo = info;
		this.indexModule = new IndexModule(metadataInfo);
		this.searchModule = new SearchModule(metadataInfo);
	}

	public void setMetadataInfo(MetadataIngesterInfo info) {
		this.metadataInfo = info;
	}

	public void index() {
		indexModule.startIndex();
	}

	public HitsMI search(QueryMI query) {
		HitsMI hits = searchModule.search(query);
		hits.writeResultFile();
		return hits;
	}

}
```

Apêndice F – Classe *ConfigUtil*

```
package tcc.tbfm.model.classesUtils;

import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.util.Hashtable;
import java.util.Properties;

public class ConfigUtil {

	private static Hashtable<String, String> table;

	public synchronized static String getConfByKey(String key) {
		if (table == null) {
			loadProperties();
		}
		return table.get(key);
	}

	private static void loadProperties() {
		File fl = new File(
				"E:\\UFSC\\Estudo\\ProjetosJava\\MetadataIngester\\conf\\common.properties");
		if (!fl.exists()) {
			System.out.println("Erro Arquivo de configuracao nao encontrado!");
			System.exit(13);
		}
		Properties props = new Properties();
		try {
			props.load(new FileInputStream(fl));
		} catch (FileNotFoundException e) {
			System.out.println("Erro ao carregar Properties: " + e);
		} catch (IOException e) {
			System.out.println("Erro ao carregar Properties: " + e);
```

```
                }
                table = new Hashtable<String, String>();
                for (Object i : props.keySet()) {
                        table.put(i.toString(), props.getProperty(i.toString()));
                }
        }
}
```

Apêndice G – Classe *HelperFunctions*

```
package tcc.tbfm.model.classesUtils;

import java.io.BufferedInputStream;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.io.Serializable;
import java.util.ArrayList;
import java.util.Calendar;
import java.util.Enumeration;
import java.util.Hashtable;
import java.util.regex.Matcher;
import java.util.regex.Pattern;

import org.apache.lucene.search.ScoreDoc;
import org.apache.lucene.store.Directory;
import org.apache.lucene.wordnet.SynonymMap;

public class HelperFunctions {

	public static void encodingXML(File fl, String encoding) {
		StringBuffer stringb = new StringBuffer();
		try {
			FileInputStream fIn = new FileInputStream(fl);
			BufferedInputStream bIn = new BufferedInputStream(fIn);
			int c = 0;
			while ((c = bIn.read()) != -1) {
				stringb.append((char) c);
			}
			fIn.close();
```

```
                bIn.close();
                if (stringb.indexOf("<?xml version") == -1) {
                        StringBuffer stringEnconding = new StringBuffer();
                        stringEnconding.append("<?xml version=\"1.0\" encoding=\""
                                        + encoding + "\"?>\n");
                        stringEnconding.append(stringb);
                        FileOutputStream fOut = new FileOutputStream(fl);
                        for (int i = 0; i < stringEnconding.length(); i++) {
                                fOut.write(stringEnconding.charAt(i));
                        }
                        fOut.close();
                }
        } catch (FileNotFoundException e) {
                e.printStackTrace();
        } catch (IOException e) {
                e.printStackTrace();
        }
}

public static void cleanDirectory(Directory dir) throws IOException {
        String[] files = dir.listAll();
        for (String file : files) {
                dir.deleteFile(file);
        }
}

public static String arrayListToString(ArrayList<String> array) {
        String result = "";
        for (int i = 0; i < array.size(); i++) {
                if (i == array.size() - 1) {
                        result = result + array.get(i);
                } else {
                        result = result + array.get(i) + ", ";
                }
        }
        return result;
}

public static ArrayList<String> arrayToArrayList(String[] array) {
        ArrayList<String> arrayList = new ArrayList<String>();
```

```
	for (int i = 0; i < array.length; i++) {
		arrayList.add(array[i]);
	}
	return arrayList;
}

public static ArrayList<ScoreDoc> arrayToArrayList(ScoreDoc[] array) {
	ArrayList<ScoreDoc> arrayList = new ArrayList<ScoreDoc>();
	for (int i = 0; i < array.length; i++) {
		arrayList.add(array[i]);
	}
	return arrayList;
}

public static void persistObject(Serializable obj, String path)
		throws IOException {
	File stream = new File(path);
	FileOutputStream ostream = new FileOutputStream(stream);
	ObjectOutputStream p = new ObjectOutputStream(ostream);
	p.writeObject(obj);
	ostream.close();
	p.close();
}

public static Serializable getObject(String path) {
	File game = new File(path);
	FileInputStream istream;
	try {
		istream = new FileInputStream(game);
		ObjectInputStream p;
		p = new ObjectInputStream(istream);
		Serializable obj;
		obj = (Serializable) p.readObject();
		istream.close();
		p.close();
		return obj;
	} catch (FileNotFoundException e) {
		e.printStackTrace();
	} catch (IOException e) {
		e.printStackTrace();
```

```
		} catch (ClassNotFoundException e) {
			e.printStackTrace();
		}
		return null;
	}

	public static ArrayList<String> getSynonysm(String word) {
		ArrayList<String> words = new ArrayList<String>();
		SynonymMap map;
		try {
			map = new SynonymMap(new FileInputStream(
					ConfigUtil.getConfByKey("WORDNET_PROLOG")));
			String[] synonyms = map.getSynonyms(word.toLowerCase());
			words = arrayToArrayList(synonyms);
		} catch (FileNotFoundException e) {
			e.printStackTrace();
		} catch (IOException e) {
			e.printStackTrace();
		}
		return words;
	}

	public static void addAll(ArrayList<String> arrayToAdd,
			ArrayList<String> baseArray) {
		if (arrayToAdd.size() > 0) {
			for (String ar : arrayToAdd) {
				if (!baseArray.contains(ar)) {
					baseArray.add(ar);
				}
			}
		}
	}

	public static Calendar getDateFromStringDate(String dateTime) {

		Calendar c = Calendar.getInstance();

		String dateTimeAux = normalizeDateTime(dateTime);
		String[] dateTimeArray = dateTimeAux.split(" ");
```

```
	if (dateTimeArray.length == 2) {

		String[] date = dateTimeArray[0].split(":");
		if (date.length != 3)
			date = dateTimeArray[0].split("/");

		String[] hours = dateTimeArray[1].split(":");

		int day = Integer.parseInt(date[2]);
		int month = Integer.parseInt(date[1]);
		int year = Integer.parseInt(date[0]);
		int hour = Integer.parseInt(hours[0]);
		int minutes = Integer.parseInt(hours[1]);
		int seconds = Integer.parseInt(hours[2]);

		c.set(year, month, day, hour, minutes, seconds);

	} else
		return null;

	return c;

}

public static Calendar getDateFromMillis(long millis) {
	Calendar c = Calendar.getInstance();
	c.setTimeInMillis(millis);
	return c;
}

private static String normalizeDateTime(String dateTime) {
	Pattern padrao = Pattern
			.compile("(\\d\\d\\d\\d):?/?(\\d?\\d):?/?(\\d?\\d) (\\d?\\d):(\\d?\\d):(\\d?\\d)");
	Matcher pesquisa = padrao.matcher(dateTime);

	if (pesquisa.matches()) {
		return dateTime;
	} else {
		return "";
	}
```

```
	}

	public static String maxValue(Hashtable<String, Integer> integers) {
		int max = 0;
		String value = "";
		Enumeration<String> keys = integers.keys();
		while (keys.hasMoreElements()) {
			String strAux = keys.nextElement();
			if (integers.get(strAux) > max) {
				max = integers.get(strAux);
				value = strAux;
			}
		}
		return value;
	}

}
```

Apêndice H – Classe *Hit*

```
package tcc.tbfm.model.classesUtils;

public class Hit {

	private String content = "";
	private int score = 0;

	public Hit(String content, int score) {
		this.content = content;
		this.score = score;
	}

	public String getContent() {
		return content;
	}

	public void setContent(String content) {
		this.content = content;
	}

	public int getScore() {
		return score;
	}

	public void setScore(int score) {this.score = score;}

	@Override
	public boolean equals(Object obj) {
		Hit h = (Hit) obj;
		return h.getContent().equals(this.getContent());
	}

}
```

Apêndice I – Classe *MIColors*

```
package tcc.tbfm.model.classesUtils;

import java.awt.Color;
import java.util.Hashtable;

public class MIColors {

	public static Hashtable<Color, String> colors = new Hashtable<>();

	static {
		colors.put(new Color(53, 49, 44), MIConstants.IMAGE_COLOR_GRAY);
		colors.put(new Color(117, 170, 148), MIConstants.IMAGE_COLOR_GREEN);
		colors.put(new Color(192, 232, 213), MIConstants.IMAGE_COLOR_GREEN);
		colors.put(new Color(116, 80, 133), MIConstants.IMAGE_COLOR_PURPLE);
		colors.put(new Color(144, 94, 38), MIConstants.IMAGE_COLOR_BROWN);
		colors.put(new Color(93, 138, 168), MIConstants.IMAGE_COLOR_BLUE);
		colors.put(new Color(190, 178, 154), MIConstants.IMAGE_COLOR_GRAY);
		colors.put(new Color(242, 240, 230), MIConstants.IMAGE_COLOR_GRAY);
		colors.put(new Color(225, 218, 203), MIConstants.IMAGE_COLOR_YELLOW);
		colors.put(new Color(149, 78, 44), MIConstants.IMAGE_COLOR_BROWN);
		colors.put(new Color(240, 248, 255), MIConstants.IMAGE_COLOR_BLUE);
		colors.put(new Color(227, 38, 54), MIConstants.IMAGE_COLOR_RED);
		colors.put(new Color(31, 106, 125), MIConstants.IMAGE_COLOR_BLUE);
		colors.put(new Color(238, 217, 196), MIConstants.IMAGE_COLOR_BROWN);
		colors.put(new Color(154, 134, 120), MIConstants.IMAGE_COLOR_BROWN);
		colors.put(new Color(173, 138, 59), MIConstants.IMAGE_COLOR_YELLOW);
		colors.put(new Color(205, 198, 197), MIConstants.IMAGE_COLOR_GRAY);
		colors.put(new Color(132, 135, 137), MIConstants.IMAGE_COLOR_GRAY);
		colors.put(new Color(229, 43, 80), MIConstants.IMAGE_COLOR_RED);
		colors.put(new Color(56, 123, 84), MIConstants.IMAGE_COLOR_GREEN);
		colors.put(new Color(255, 191, 0), MIConstants.IMAGE_COLOR_YELLOW);
		colors.put(new Color(138, 125, 114), MIConstants.IMAGE_COLOR_GRAY);
		colors.put(new Color(153, 102, 204), MIConstants.IMAGE_COLOR_PURPLE);
		colors.put(new Color(149, 135, 156), MIConstants.IMAGE_COLOR_PURPLE);
```

```
colors.put(new Color(245, 230, 234), MIConstants.IMAGE_COLOR_PURPLE);
colors.put(new Color(125, 157, 114), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(140, 206, 234), MIConstants.IMAGE_COLOR_BLUE);
colors.put(new Color(108, 70, 31), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(250, 235, 215), MIConstants.IMAGE_COLOR_WHITE);
colors.put(new Color(198, 142, 63), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(211, 169, 92), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(102, 179, 72), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(169, 82, 73), MIConstants.IMAGE_COLOR_PINK);
colors.put(new Color(222, 234, 220), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(251, 206, 177), MIConstants.IMAGE_COLOR_ORANGE);
colors.put(new Color(247, 240, 219), MIConstants.IMAGE_COLOR_WHITE);
colors.put(new Color(0, 255, 255), MIConstants.IMAGE_COLOR_BLUE);
colors.put(new Color(217, 221, 213), MIConstants.IMAGE_COLOR_GRAY);
colors.put(new Color(232, 243, 232), MIConstants.IMAGE_COLOR_WHITE);
colors.put(new Color(219, 228, 220), MIConstants.IMAGE_COLOR_GRAY);
colors.put(new Color(127, 255, 212), MIConstants.IMAGE_COLOR_BLUE);
colors.put(new Color(39, 74, 93), MIConstants.IMAGE_COLOR_BLUE);
colors.put(new Color(72, 74, 70), MIConstants.IMAGE_COLOR_GRAY);
colors.put(new Color(75, 83, 32), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(130, 122, 103), MIConstants.IMAGE_COLOR_GRAY);
colors.put(new Color(59, 68, 75), MIConstants.IMAGE_COLOR_GRAY);
colors.put(new Color(190, 186, 167), MIConstants.IMAGE_COLOR_GRAY);
colors.put(new Color(123, 160, 91), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(237, 213, 166), MIConstants.IMAGE_COLOR_YELLOW);
colors.put(new Color(55, 111, 137), MIConstants.IMAGE_COLOR_BLUE);
colors.put(new Color(68, 81, 114), MIConstants.IMAGE_COLOR_BLUE);
colors.put(new Color(33, 69, 89), MIConstants.IMAGE_COLOR_BLUE);
colors.put(new Color(220, 221, 221), MIConstants.IMAGE_COLOR_GRAY);
colors.put(new Color(213, 203, 178), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(156, 208, 59), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(43, 121, 122), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(61, 75, 82), MIConstants.IMAGE_COLOR_BLUE);
colors.put(new Color(255, 153, 102), MIConstants.IMAGE_COLOR_ORANGE);
colors.put(new Color(158, 103, 89), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(55, 37, 40), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(113, 47, 44), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(239, 248, 170), MIConstants.IMAGE_COLOR_YELLOW);
colors.put(new Color(149, 152, 107), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(99, 119, 90), MIConstants.IMAGE_COLOR_GREEN);
```

```
colors.put(new Color(249, 192, 196), MIConstants.IMAGE_COLOR_RED);
colors.put(new Color(41, 52, 50), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(240, 255, 255), MIConstants.IMAGE_COLOR_WHITE);
colors.put(new Color(111, 255, 255), MIConstants.IMAGE_COLOR_BLUE);
colors.put(new Color(37, 89, 127), MIConstants.IMAGE_COLOR_BLUE);
colors.put(new Color(169, 192, 28), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(92, 51, 23), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(132, 156, 169), MIConstants.IMAGE_COLOR_BLUE);
colors.put(new Color(60, 61, 62), MIConstants.IMAGE_COLOR_GRAY);
colors.put(new Color(251, 231, 178), MIConstants.IMAGE_COLOR_YELLOW);
colors.put(new Color(135, 132, 102), MIConstants.IMAGE_COLOR_GRAY);
colors.put(new Color(210, 198, 31), MIConstants.IMAGE_COLOR_YELLOW);
colors.put(new Color(182, 147, 92), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(247, 229, 183), MIConstants.IMAGE_COLOR_YELLOW);
colors.put(new Color(69, 46, 57), MIConstants.IMAGE_COLOR_PURPLE);
colors.put(new Color(44, 44, 50), MIConstants.IMAGE_COLOR_BLUE);
colors.put(new Color(81, 87, 79), MIConstants.IMAGE_COLOR_GRAY);
colors.put(new Color(123, 177, 141), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(53, 62, 100), MIConstants.IMAGE_COLOR_BLUE);
colors.put(new Color(143, 119, 119), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(235, 185, 179), MIConstants.IMAGE_COLOR_PINK);
colors.put(new Color(146, 111, 91), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(233, 215, 171), MIConstants.IMAGE_COLOR_YELLOW);
colors.put(new Color(245, 245, 220), MIConstants.IMAGE_COLOR_WHITE);
colors.put(new Color(134, 210, 193), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(111, 140, 159), MIConstants.IMAGE_COLOR_BLUE);
colors.put(new Color(188, 191, 168), MIConstants.IMAGE_COLOR_GRAY);
colors.put(new Color(244, 239, 224), MIConstants.IMAGE_COLOR_WHITE);
colors.put(new Color(51, 64, 70), MIConstants.IMAGE_COLOR_BLUE);
colors.put(new Color(62, 128, 39), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(174, 153, 210), MIConstants.IMAGE_COLOR_PURPLE);
colors.put(new Color(63, 55, 38), MIConstants.IMAGE_COLOR_GRAY);
colors.put(new Color(208, 193, 23), MIConstants.IMAGE_COLOR_YELLOW);
colors.put(new Color(47, 60, 83), MIConstants.IMAGE_COLOR_BLUE);
colors.put(new Color(72, 108, 122), MIConstants.IMAGE_COLOR_BLUE);
colors.put(new Color(181, 172, 148), MIConstants.IMAGE_COLOR_GRAY);
colors.put(new Color(255, 228, 196), MIConstants.IMAGE_COLOR_ORANGE);
colors.put(new Color(61, 43, 31), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(136, 137, 108), MIConstants.IMAGE_COLOR_GRAY);
colors.put(new Color(210, 219, 50), MIConstants.IMAGE_COLOR_YELLOW);
```

```
colors.put(new Color(254, 111, 94), MIConstants.IMAGE_COLOR_RED);
colors.put(new Color(231, 210, 200), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(0, 0, 0), MIConstants.IMAGE_COLOR_BLACK);
colors.put(new Color(35, 46, 38), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(44, 50, 39), MIConstants.IMAGE_COLOR_GRAY);
colors.put(new Color(224, 222, 215), MIConstants.IMAGE_COLOR_GRAY);
colors.put(new Color(51, 44, 34), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(56, 55, 64), MIConstants.IMAGE_COLOR_BLUE);
colors.put(new Color(30, 39, 44), MIConstants.IMAGE_COLOR_BLACK);
colors.put(new Color(44, 45, 60), MIConstants.IMAGE_COLOR_BLUE);
colors.put(new Color(83, 41, 52), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(36, 37, 43), MIConstants.IMAGE_COLOR_GRAY);
colors.put(new Color(229, 230, 223), MIConstants.IMAGE_COLOR_GRAY);
colors.put(new Color(229, 228, 219), MIConstants.IMAGE_COLOR_GRAY);
colors.put(new Color(67, 24, 47), MIConstants.IMAGE_COLOR_PURPLE);
colors.put(new Color(46, 24, 59), MIConstants.IMAGE_COLOR_PURPLE);
colors.put(new Color(217, 208, 193), MIConstants.IMAGE_COLOR_GRAY);
colors.put(new Color(255, 235, 205), MIConstants.IMAGE_COLOR_ORANGE);
colors.put(new Color(235, 225, 206), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(163, 227, 237), MIConstants.IMAGE_COLOR_BLUE);
colors.put(new Color(223, 177, 182), MIConstants.IMAGE_COLOR_PINK);
colors.put(new Color(0, 0, 255), MIConstants.IMAGE_COLOR_BLUE);
colors.put(new Color(98, 119, 126), MIConstants.IMAGE_COLOR_BLUE);
colors.put(new Color(153, 153, 204), MIConstants.IMAGE_COLOR_BLUE);
colors.put(new Color(227, 214, 233), MIConstants.IMAGE_COLOR_PURPLE);
colors.put(new Color(38, 43, 47), MIConstants.IMAGE_COLOR_BLACK);
colors.put(new Color(64, 143, 144), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(75, 45, 114), MIConstants.IMAGE_COLOR_PURPLE);
colors.put(new Color(53, 81, 79), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(75, 60, 142), MIConstants.IMAGE_COLOR_PURPLE);
colors.put(new Color(189, 186, 206), MIConstants.IMAGE_COLOR_GRAY);
colors.put(new Color(0, 98, 111), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(106, 91, 177), MIConstants.IMAGE_COLOR_PURPLE);
colors.put(new Color(216, 240, 210), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(120, 133, 122), MIConstants.IMAGE_COLOR_GRAY);
colors.put(new Color(22, 100, 97), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(138, 43, 226), MIConstants.IMAGE_COLOR_PURPLE);
colors.put(new Color(30, 52, 66), MIConstants.IMAGE_COLOR_BLUE);
colors.put(new Color(60, 67, 84), MIConstants.IMAGE_COLOR_BLUE);
colors.put(new Color(48, 92, 113), MIConstants.IMAGE_COLOR_BLUE);
```

```
colors.put(new Color(181, 80, 103), MIConstants.IMAGE_COLOR_PINK);
colors.put(new Color(42, 39, 37), MIConstants.IMAGE_COLOR_GRAY);
colors.put(new Color(121, 68, 59), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(174, 174, 173), MIConstants.IMAGE_COLOR_GRAY);
colors.put(new Color(223, 215, 210), MIConstants.IMAGE_COLOR_GRAY);
colors.put(new Color(0, 149, 182), MIConstants.IMAGE_COLOR_BLUE);
colors.put(new Color(219, 194, 171), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(76, 28, 36), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(76, 61, 78), MIConstants.IMAGE_COLOR_GRAY);
colors.put(new Color(67, 142, 172), MIConstants.IMAGE_COLOR_BLUE);
colors.put(new Color(146, 172, 180), MIConstants.IMAGE_COLOR_BLUE);
colors.put(new Color(37, 70, 54), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(124, 129, 124), MIConstants.IMAGE_COLOR_GRAY);
colors.put(new Color(167, 129, 153), MIConstants.IMAGE_COLOR_PURPLE);
colors.put(new Color(175, 108, 62), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(91, 61, 39), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(220, 182, 138), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(192, 124, 64), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(182, 133, 122), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(181, 166, 66), MIConstants.IMAGE_COLOR_YELLOW);
colors.put(new Color(81, 123, 120), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(198, 45, 66), MIConstants.IMAGE_COLOR_RED);
colors.put(new Color(248, 235, 221), MIConstants.IMAGE_COLOR_WHITE);
colors.put(new Color(250, 230, 223), MIConstants.IMAGE_COLOR_WHITE);
colors.put(new Color(102, 255, 0), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(87, 89, 93), MIConstants.IMAGE_COLOR_GRAY);
colors.put(new Color(146, 42, 49), MIConstants.IMAGE_COLOR_RED);
colors.put(new Color(236, 189, 44), MIConstants.IMAGE_COLOR_YELLOW);
colors.put(new Color(8, 232, 222), MIConstants.IMAGE_COLOR_BLUE);
colors.put(new Color(255, 85, 163), MIConstants.IMAGE_COLOR_PINK);
colors.put(new Color(251, 96, 127), MIConstants.IMAGE_COLOR_PINK);
colors.put(new Color(0, 66, 37), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(167, 151, 129), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(205, 127, 50), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(88, 76, 37), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(67, 76, 40), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(238, 204, 36), MIConstants.IMAGE_COLOR_YELLOW);
colors.put(new Color(165, 42, 42), MIConstants.IMAGE_COLOR_RED);
colors.put(new Color(83, 51, 30), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(89, 69, 55), MIConstants.IMAGE_COLOR_BROWN);
```

```
colors.put(new Color(60, 36, 27), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(230, 242, 234), MIConstants.IMAGE_COLOR_WHITE);
colors.put(new Color(110, 81, 80), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(165, 168, 143), MIConstants.IMAGE_COLOR_GRAY);
colors.put(new Color(188, 155, 27), MIConstants.IMAGE_COLOR_YELLOW);
colors.put(new Color(240, 220, 130), MIConstants.IMAGE_COLOR_YELLOW);
colors.put(new Color(72, 36, 39), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(117, 68, 43), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(41, 44, 47), MIConstants.IMAGE_COLOR_GRAY);
colors.put(new Color(43, 52, 73), MIConstants.IMAGE_COLOR_BLUE);
colors.put(new Color(128, 0, 32), MIConstants.IMAGE_COLOR_RED);
colors.put(new Color(222, 184, 135), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(35, 69, 55), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(208, 131, 99), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(88, 33, 36), MIConstants.IMAGE_COLOR_RED);
colors.put(new Color(255, 112, 52), MIConstants.IMAGE_COLOR_ORANGE);
colors.put(new Color(233, 116, 81), MIConstants.IMAGE_COLOR_RED);
colors.put(new Color(138, 51, 36), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(218, 148, 41), MIConstants.IMAGE_COLOR_YELLOW);
colors.put(new Color(157, 112, 46), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(104, 87, 140), MIConstants.IMAGE_COLOR_PURPLE);
colors.put(new Color(246, 224, 164), MIConstants.IMAGE_COLOR_YELLOW);
colors.put(new Color(241, 235, 218), MIConstants.IMAGE_COLOR_GRAY);
colors.put(new Color(74, 46, 50), MIConstants.IMAGE_COLOR_PURPLE);
colors.put(new Color(205, 82, 108), MIConstants.IMAGE_COLOR_PINK);
colors.put(new Color(76, 85, 68), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(91, 111, 85), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(95, 158, 160), MIConstants.IMAGE_COLOR_BLUE);
colors.put(new Color(152, 73, 97), MIConstants.IMAGE_COLOR_PINK);
colors.put(new Color(106, 73, 40), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(213, 177, 133), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(233, 140, 58), MIConstants.IMAGE_COLOR_ORANGE);
colors.put(new Color(61, 113, 136), MIConstants.IMAGE_COLOR_BLUE);
colors.put(new Color(32, 105, 55), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(128, 58, 75), MIConstants.IMAGE_COLOR_PURPLE);
colors.put(new Color(204, 164, 131), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(79, 77, 50), MIConstants.IMAGE_COLOR_GRAY);
colors.put(new Color(120, 134, 107), MIConstants.IMAGE_COLOR_GRAY);
colors.put(new Color(208, 138, 155), MIConstants.IMAGE_COLOR_PINK);
colors.put(new Color(255, 255, 153), MIConstants.IMAGE_COLOR_YELLOW);
```

```
colors.put(new Color(142, 81, 100), MIConstants.IMAGE_COLOR_PURPLE);
colors.put(new Color(78, 85, 82), MIConstants.IMAGE_COLOR_GRAY);
colors.put(new Color(254, 224, 165), MIConstants.IMAGE_COLOR_YELLOW);
colors.put(new Color(117, 72, 47), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(175, 193, 130), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(89, 39, 32), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(255, 213, 154), MIConstants.IMAGE_COLOR_PINK);
colors.put(new Color(235, 229, 213), MIConstants.IMAGE_COLOR_GRAY);
colors.put(new Color(27, 52, 39), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(196, 30, 58), MIConstants.IMAGE_COLOR_RED);
colors.put(new Color(201, 154, 160), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(0, 204, 153), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(230, 128, 149), MIConstants.IMAGE_COLOR_PINK);
colors.put(new Color(245, 249, 203), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(150, 0, 24), MIConstants.IMAGE_COLOR_RED);
colors.put(new Color(91, 58, 36), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(255, 166, 201), MIConstants.IMAGE_COLOR_PINK);
colors.put(new Color(248, 219, 224), MIConstants.IMAGE_COLOR_PINK);
colors.put(new Color(237, 145, 33), MIConstants.IMAGE_COLOR_YELLOW);
colors.put(new Color(240, 178, 83), MIConstants.IMAGE_COLOR_YELLOW);
colors.put(new Color(63, 84, 90), MIConstants.IMAGE_COLOR_BLUE);
colors.put(new Color(140, 168, 160), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(209, 179, 153), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(170, 181, 184), MIConstants.IMAGE_COLOR_GRAY);
colors.put(new Color(68, 35, 47), MIConstants.IMAGE_COLOR_PURPLE);
colors.put(new Color(39, 60, 90), MIConstants.IMAGE_COLOR_BLUE);
colors.put(new Color(224, 228, 220), MIConstants.IMAGE_COLOR_GRAY);
colors.put(new Color(224, 184, 177), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(146, 113, 167), MIConstants.IMAGE_COLOR_PURPLE);
colors.put(new Color(70, 52, 48), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(172, 225, 175), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(180, 192, 76), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(210, 210, 192), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(58, 78, 95), MIConstants.IMAGE_COLOR_BLUE);
colors.put(new Color(43, 63, 54), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(133, 113, 88), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(222, 49, 99), MIConstants.IMAGE_COLOR_PINK);
colors.put(new Color(0, 123, 167), MIConstants.IMAGE_COLOR_BLUE);
colors.put(new Color(42, 82, 190), MIConstants.IMAGE_COLOR_BLUE);
colors.put(new Color(253, 233, 224), MIConstants.IMAGE_COLOR_WHITE);
```

```
colors.put(new Color(90, 110, 65), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(223, 194, 129), MIConstants.IMAGE_COLOR_YELLOW);
colors.put(new Color(71, 88, 119), MIConstants.IMAGE_COLOR_BLUE);
colors.put(new Color(232, 205, 154), MIConstants.IMAGE_COLOR_YELLOW);
colors.put(new Color(238, 217, 182), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(237, 184, 199), MIConstants.IMAGE_COLOR_PINK);
colors.put(new Color(57, 64, 67), MIConstants.IMAGE_COLOR_GRAY);
colors.put(new Color(70, 70, 70), MIConstants.IMAGE_COLOR_GRAY);
colors.put(new Color(248, 234, 223), MIConstants.IMAGE_COLOR_WHITE);
colors.put(new Color(255, 200, 120), MIConstants.IMAGE_COLOR_YELLOW);
colors.put(new Color(164, 220, 230), MIConstants.IMAGE_COLOR_BLUE);
colors.put(new Color(208, 116, 139), MIConstants.IMAGE_COLOR_PINK);
colors.put(new Color(127, 255, 0), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(223, 255, 0), MIConstants.IMAGE_COLOR_YELLOW);
colors.put(new Color(65, 159, 89), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(179, 171, 182), MIConstants.IMAGE_COLOR_GRAY);
colors.put(new Color(44, 89, 113), MIConstants.IMAGE_COLOR_BLUE);
colors.put(new Color(136, 169, 91), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(149, 83, 47), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(222, 195, 113), MIConstants.IMAGE_COLOR_YELLOW);
colors.put(new Color(245, 205, 130), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(55, 45, 82), MIConstants.IMAGE_COLOR_PURPLE);
colors.put(new Color(245, 215, 220), MIConstants.IMAGE_COLOR_PINK);
colors.put(new Color(185, 78, 72), MIConstants.IMAGE_COLOR_RED);
colors.put(new Color(102, 111, 180), MIConstants.IMAGE_COLOR_PURPLE);
colors.put(new Color(91, 93, 86), MIConstants.IMAGE_COLOR_GRAY);
colors.put(new Color(240, 245, 187), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(208, 94, 52), MIConstants.IMAGE_COLOR_ORANGE);
colors.put(new Color(249, 247, 222), MIConstants.IMAGE_COLOR_WHITE);
colors.put(new Color(251, 243, 211), MIConstants.IMAGE_COLOR_WHITE);
colors.put(new Color(157, 211, 168), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(210, 105, 30), MIConstants.IMAGE_COLOR_ORANGE);
colors.put(new Color(56, 33, 97), MIConstants.IMAGE_COLOR_PURPLE);
colors.put(new Color(113, 169, 29), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(191, 101, 46), MIConstants.IMAGE_COLOR_ORANGE);
colors.put(new Color(202, 199, 183), MIConstants.IMAGE_COLOR_GRAY);
colors.put(new Color(125, 78, 56), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(36, 42, 46), MIConstants.IMAGE_COLOR_BLACK);
colors.put(new Color(251, 215, 204), MIConstants.IMAGE_COLOR_RED);
colors.put(new Color(227, 66, 52), MIConstants.IMAGE_COLOR_RED);
```

```
colors.put(new Color(93, 59, 46), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(142, 154, 33), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(159, 183, 10), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(210, 179, 169), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(110, 34, 51), MIConstants.IMAGE_COLOR_RED);
colors.put(new Color(244, 200, 219), MIConstants.IMAGE_COLOR_PINK);
colors.put(new Color(137, 126, 89), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(223, 239, 234), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(70, 54, 35), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(194, 188, 177), MIConstants.IMAGE_COLOR_GRAY);
colors.put(new Color(53, 62, 79), MIConstants.IMAGE_COLOR_BLUE);
colors.put(new Color(176, 169, 159), MIConstants.IMAGE_COLOR_GRAY);
colors.put(new Color(71, 86, 47), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(0, 71, 171), MIConstants.IMAGE_COLOR_BLUE);
colors.put(new Color(79, 56, 53), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(53, 40, 30), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(225, 218, 187), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(45, 48, 50), MIConstants.IMAGE_COLOR_GRAY);
colors.put(new Color(114, 103, 81), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(54, 45, 38), MIConstants.IMAGE_COLOR_BLACK);
colors.put(new Color(154, 70, 61), MIConstants.IMAGE_COLOR_RED);
colors.put(new Color(60, 47, 35), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(157, 138, 191), MIConstants.IMAGE_COLOR_PURPLE);
colors.put(new Color(202, 181, 178), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(155, 221, 255), MIConstants.IMAGE_COLOR_BLUE);
colors.put(new Color(99, 99, 115), MIConstants.IMAGE_COLOR_GRAY);
colors.put(new Color(76, 120, 92), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(160, 177, 174), MIConstants.IMAGE_COLOR_GRAY);
colors.put(new Color(130, 127, 121), MIConstants.IMAGE_COLOR_GRAY);
colors.put(new Color(210, 209, 205), MIConstants.IMAGE_COLOR_GRAY);
colors.put(new Color(221, 203, 70), MIConstants.IMAGE_COLOR_YELLOW);
colors.put(new Color(101, 77, 73), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(177, 221, 82), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(193, 111, 104), MIConstants.IMAGE_COLOR_RED);
colors.put(new Color(218, 138, 103), MIConstants.IMAGE_COLOR_ORANGE);
colors.put(new Color(119, 66, 44), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(153, 102, 102), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(149, 82, 76), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(255, 127, 80), MIConstants.IMAGE_COLOR_ORANGE);
colors.put(new Color(245, 208, 201), MIConstants.IMAGE_COLOR_RED);
```

```
colors.put(new Color(255, 64, 64), MIConstants.IMAGE_COLOR_RED);
colors.put(new Color(171, 110, 103), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(64, 77, 73), MIConstants.IMAGE_COLOR_GRAY);
colors.put(new Color(187, 181, 141), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(90, 76, 66), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(251, 236, 93), MIConstants.IMAGE_COLOR_YELLOW);
colors.put(new Color(248, 243, 196), MIConstants.IMAGE_COLOR_YELLOW);
colors.put(new Color(66, 66, 111), MIConstants.IMAGE_COLOR_PURPLE);
colors.put(new Color(141, 112, 42), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(255, 248, 220), MIConstants.IMAGE_COLOR_WHITE);
colors.put(new Color(147, 204, 234), MIConstants.IMAGE_COLOR_BLUE);
colors.put(new Color(100, 149, 237), MIConstants.IMAGE_COLOR_BLUE);
colors.put(new Color(233, 186, 129), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(121, 77, 96), MIConstants.IMAGE_COLOR_PURPLE);
colors.put(new Color(225, 248, 231), MIConstants.IMAGE_COLOR_WHITE);
colors.put(new Color(252, 213, 207), MIConstants.IMAGE_COLOR_RED);
colors.put(new Color(98, 93, 42), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(255, 183, 213), MIConstants.IMAGE_COLOR_PINK);
colors.put(new Color(191, 186, 175), MIConstants.IMAGE_COLOR_GRAY);
colors.put(new Color(27, 75, 53), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(68, 55, 54), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(135, 56, 47), MIConstants.IMAGE_COLOR_RED);
colors.put(new Color(166, 86, 72), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(219, 80, 121), MIConstants.IMAGE_COLOR_PINK);
colors.put(new Color(77, 62, 60), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(255, 253, 208), MIConstants.IMAGE_COLOR_WHITE);
colors.put(new Color(255, 227, 155), MIConstants.IMAGE_COLOR_YELLOW);
colors.put(new Color(238, 192, 81), MIConstants.IMAGE_COLOR_YELLOW);
colors.put(new Color(57, 50, 39), MIConstants.IMAGE_COLOR_BLACK);
colors.put(new Color(119, 113, 43), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(220, 20, 60), MIConstants.IMAGE_COLOR_RED);
colors.put(new Color(112, 105, 80), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(118, 60, 51), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(180, 226, 213), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(22, 91, 49), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(243, 134, 83), MIConstants.IMAGE_COLOR_ORANGE);
colors.put(new Color(120, 68, 48), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(245, 244, 193), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(245, 178, 197), MIConstants.IMAGE_COLOR_PINK);
colors.put(new Color(61, 133, 184), MIConstants.IMAGE_COLOR_BLUE);
```

```
colors.put(new Color(92, 129, 115), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(15, 70, 69), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(237, 210, 164), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(91, 62, 144), MIConstants.IMAGE_COLOR_PURPLE);
colors.put(new Color(102, 74, 45), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(254, 216, 93), MIConstants.IMAGE_COLOR_YELLOW);
colors.put(new Color(91, 137, 192), MIConstants.IMAGE_COLOR_BLUE);
colors.put(new Color(0, 0, 139), MIConstants.IMAGE_COLOR_BLUE);
colors.put(new Color(101, 67, 33), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(8, 69, 126), MIConstants.IMAGE_COLOR_BLUE);
colors.put(new Color(152, 105, 96), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(205, 91, 69), MIConstants.IMAGE_COLOR_ORANGE);
colors.put(new Color(0, 139, 139), MIConstants.IMAGE_COLOR_BLUE);
colors.put(new Color(184, 134, 11), MIConstants.IMAGE_COLOR_YELLOW);
colors.put(new Color(169, 169, 169), MIConstants.IMAGE_COLOR_GRAY);
colors.put(new Color(1, 50, 32), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(74, 118, 110), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(189, 183, 107), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(139, 0, 139), MIConstants.IMAGE_COLOR_PURPLE);
colors.put(new Color(85, 107, 47), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(255, 140, 0), MIConstants.IMAGE_COLOR_ORANGE);
colors.put(new Color(153, 50, 204), MIConstants.IMAGE_COLOR_PURPLE);
colors.put(new Color(3, 192, 60), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(231, 84, 128), MIConstants.IMAGE_COLOR_PINK);
colors.put(new Color(135, 31, 120), MIConstants.IMAGE_COLOR_PURPLE);
colors.put(new Color(139, 0, 0), MIConstants.IMAGE_COLOR_RED);
colors.put(new Color(69, 54, 43), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(233, 150, 122), MIConstants.IMAGE_COLOR_ORANGE);
colors.put(new Color(143, 188, 143), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(70, 83, 82), MIConstants.IMAGE_COLOR_GRAY);
colors.put(new Color(72, 61, 139), MIConstants.IMAGE_COLOR_PURPLE);
colors.put(new Color(47, 79, 79), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(23, 114, 69), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(151, 105, 79), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(255, 168, 18), MIConstants.IMAGE_COLOR_ORANGE);
colors.put(new Color(0, 206, 209), MIConstants.IMAGE_COLOR_BLUE);
colors.put(new Color(148, 0, 211), MIConstants.IMAGE_COLOR_PURPLE);
colors.put(new Color(133, 94, 66), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(120, 136, 120), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(159, 157, 145), MIConstants.IMAGE_COLOR_GRAY);
```

```
colors.put(new Color(230, 214, 205), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(133, 202, 135), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(204, 207, 130), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(227, 111, 138), MIConstants.IMAGE_COLOR_PINK);
colors.put(new Color(81, 65, 45), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(218, 50, 135), MIConstants.IMAGE_COLOR_PINK);
colors.put(new Color(25, 57, 37), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(52, 52, 103), MIConstants.IMAGE_COLOR_PURPLE);
colors.put(new Color(153, 85, 187), MIConstants.IMAGE_COLOR_PURPLE);
colors.put(new Color(204, 0, 204), MIConstants.IMAGE_COLOR_PURPLE);
colors.put(new Color(255, 20, 147), MIConstants.IMAGE_COLOR_PINK);
colors.put(new Color(22, 126, 101), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(0, 191, 255), MIConstants.IMAGE_COLOR_BLUE);
colors.put(new Color(25, 68, 60), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(181, 153, 142), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(72, 101, 49), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(153, 155, 149), MIConstants.IMAGE_COLOR_GRAY);
colors.put(new Color(130, 114, 164), MIConstants.IMAGE_COLOR_PURPLE);
colors.put(new Color(21, 96, 189), MIConstants.IMAGE_COLOR_BLUE);
colors.put(new Color(249, 228, 198), MIConstants.IMAGE_COLOR_YELLOW);
colors.put(new Color(161, 95, 59), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(237, 201, 175), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(237, 231, 224), MIConstants.IMAGE_COLOR_GRAY);
colors.put(new Color(231, 242, 233), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(50, 44, 43), MIConstants.IMAGE_COLOR_BLACK);//
colors.put(new Color(105, 105, 105), MIConstants.IMAGE_COLOR_GRAY);
colors.put(new Color(96, 124, 71), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(137, 45, 79), MIConstants.IMAGE_COLOR_PINK);
colors.put(new Color(205, 132, 49), MIConstants.IMAGE_COLOR_YELLOW);
colors.put(new Color(30, 144, 255), MIConstants.IMAGE_COLOR_BLUE);
colors.put(new Color(245, 241, 113), MIConstants.IMAGE_COLOR_YELLOW);
colors.put(new Color(106, 104, 115), MIConstants.IMAGE_COLOR_GRAY);//
colors.put(new Color(108, 91, 76), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(90, 79, 81), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(129, 110, 92), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(110, 95, 86), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(228, 207, 153), MIConstants.IMAGE_COLOR_YELLOW);
colors.put(new Color(233, 220, 190), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(210, 195, 163), MIConstants.IMAGE_COLOR_GRAY);
colors.put(new Color(119, 118, 114), MIConstants.IMAGE_COLOR_GRAY);
```

```
colors.put(new Color(111, 210, 190), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(251, 235, 155), MIConstants.IMAGE_COLOR_YELLOW);
colors.put(new Color(81, 79, 74), MIConstants.IMAGE_COLOR_GRAY);
colors.put(new Color(229, 202, 192), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(172, 155, 155), MIConstants.IMAGE_COLOR_GRAY);
colors.put(new Color(240, 223, 187), MIConstants.IMAGE_COLOR_YELLOW);
colors.put(new Color(176, 172, 148), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(184, 167, 34), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(251, 242, 219), MIConstants.IMAGE_COLOR_WHITE);
colors.put(new Color(71, 82, 110), MIConstants.IMAGE_COLOR_PURPLE);
colors.put(new Color(170, 140, 188), MIConstants.IMAGE_COLOR_PURPLE);
colors.put(new Color(0, 135, 159), MIConstants.IMAGE_COLOR_BLUE);
colors.put(new Color(230, 216, 212), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(49, 51, 55), MIConstants.IMAGE_COLOR_BLACK);
colors.put(new Color(50, 52, 56), MIConstants.IMAGE_COLOR_GRAY);
colors.put(new Color(164, 175, 205), MIConstants.IMAGE_COLOR_PURPLE);
colors.put(new Color(63, 57, 57), MIConstants.IMAGE_COLOR_GRAY);
colors.put(new Color(194, 178, 128), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(214, 209, 192), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(201, 97, 56), MIConstants.IMAGE_COLOR_ORANGE);
colors.put(new Color(38, 98, 85), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(193, 216, 197), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(151, 164, 154), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(249, 228, 197), MIConstants.IMAGE_COLOR_YELLOW);
colors.put(new Color(153, 0, 102), MIConstants.IMAGE_COLOR_PINK);
colors.put(new Color(16, 52, 166), MIConstants.IMAGE_COLOR_BLUE);
colors.put(new Color(57, 57, 44), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(143, 78, 69), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(125, 249, 255), MIConstants.IMAGE_COLOR_BLUE);
colors.put(new Color(102, 0, 255), MIConstants.IMAGE_COLOR_PURPLE);
colors.put(new Color(204, 255, 0), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(191, 0, 255), MIConstants.IMAGE_COLOR_PURPLE);
colors.put(new Color(36, 54, 64), MIConstants.IMAGE_COLOR_BLUE);
colors.put(new Color(27, 138, 107), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(41, 123, 118), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(80, 200, 120), MIConstants.IMAGE_COLOR_BLUE);
colors.put(new Color(110, 57, 116), MIConstants.IMAGE_COLOR_PURPLE);
colors.put(new Color(80, 73, 74), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(124, 113, 115), MIConstants.IMAGE_COLOR_GRAY);
colors.put(new Color(41, 89, 139), MIConstants.IMAGE_COLOR_BLUE);
```

```
colors.put(new Color(245, 215, 82), MIConstants.IMAGE_COLOR_YELLOW);
colors.put(new Color(39, 66, 52), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(139, 165, 143), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(218, 177, 96), MIConstants.IMAGE_COLOR_YELLOW);
colors.put(new Color(78, 49, 45), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(45, 47, 40), MIConstants.IMAGE_COLOR_BLACK);
colors.put(new Color(50, 151, 96), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(205, 165, 156), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(38, 96, 79), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(38, 67, 52), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(243, 229, 220), MIConstants.IMAGE_COLOR_ORANGE);
colors.put(new Color(110, 90, 91), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(193, 154, 107), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(128, 24, 24), MIConstants.IMAGE_COLOR_RED);
colors.put(new Color(242, 230, 221), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(98, 86, 101), MIConstants.IMAGE_COLOR_PURPLE);
colors.put(new Color(165, 215, 133), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(77, 93, 83), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(209, 146, 117), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(99, 183, 108), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(79, 121, 66), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(135, 106, 104), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(234, 204, 74), MIConstants.IMAGE_COLOR_YELLOW);
colors.put(new Color(219, 224, 208), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(177, 89, 47), MIConstants.IMAGE_COLOR_ORANGE);
colors.put(new Color(99, 111, 34), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(117, 120, 90), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(97, 117, 91), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(105, 69, 84), MIConstants.IMAGE_COLOR_PURPLE);
colors.put(new Color(75, 90, 98), MIConstants.IMAGE_COLOR_BLUE);
colors.put(new Color(143, 63, 42), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(178, 34, 34), MIConstants.IMAGE_COLOR_RED);
colors.put(new Color(224, 152, 66), MIConstants.IMAGE_COLOR_YELLOW);
colors.put(new Color(206, 22, 32), MIConstants.IMAGE_COLOR_RED);
colors.put(new Color(49, 70, 67), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(190, 92, 72), MIConstants.IMAGE_COLOR_ORANGE);
colors.put(new Color(134, 40, 46), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(234, 134, 69), MIConstants.IMAGE_COLOR_ORANGE);
colors.put(new Color(225, 99, 79), MIConstants.IMAGE_COLOR_ORANGE);
colors.put(new Color(238, 220, 130), MIConstants.IMAGE_COLOR_YELLOW);
```

```
colors.put(new Color(113, 110, 97), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(122, 46, 77), MIConstants.IMAGE_COLOR_PURPLE);
colors.put(new Color(255, 250, 240), MIConstants.IMAGE_COLOR_WHITE);
colors.put(new Color(208, 234, 232), MIConstants.IMAGE_COLOR_BLUE);
colors.put(new Color(213, 199, 232), MIConstants.IMAGE_COLOR_PURPLE);
colors.put(new Color(167, 166, 157), MIConstants.IMAGE_COLOR_GRAY);
colors.put(new Color(34, 139, 34), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(253, 239, 219), MIConstants.IMAGE_COLOR_WHITE);
colors.put(new Color(101, 173, 178), MIConstants.IMAGE_COLOR_BLUE);
colors.put(new Color(255, 215, 160), MIConstants.IMAGE_COLOR_ORANGE);
colors.put(new Color(2, 157, 116), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(65, 86, 197), MIConstants.IMAGE_COLOR_BLUE);
colors.put(new Color(9, 249, 17), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(227, 91, 216), MIConstants.IMAGE_COLOR_PINK);
colors.put(new Color(192, 0, 0), MIConstants.IMAGE_COLOR_RED);
colors.put(new Color(191, 189, 193), MIConstants.IMAGE_COLOR_GRAY);
colors.put(new Color(222, 183, 217), MIConstants.IMAGE_COLOR_PURPLE);
colors.put(new Color(164, 210, 224), MIConstants.IMAGE_COLOR_BLUE);
colors.put(new Color(246, 74, 138), MIConstants.IMAGE_COLOR_PINK);
colors.put(new Color(134, 131, 122), MIConstants.IMAGE_COLOR_GRAY);
colors.put(new Color(180, 225, 187), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(229, 109, 117), MIConstants.IMAGE_COLOR_PINK);
colors.put(new Color(225, 228, 197), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(226, 242, 228), MIConstants.IMAGE_COLOR_WHITE);
colors.put(new Color(219, 229, 210), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(75, 163, 81), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(193, 84, 193), MIConstants.IMAGE_COLOR_PURPLE);
colors.put(new Color(255, 119, 255), MIConstants.IMAGE_COLOR_PINK);
colors.put(new Color(194, 214, 46), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(209, 144, 51), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(51, 80, 131), MIConstants.IMAGE_COLOR_BLUE);
colors.put(new Color(21, 99, 61), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(60, 59, 60), MIConstants.IMAGE_COLOR_GRAY);
colors.put(new Color(196, 86, 85), MIConstants.IMAGE_COLOR_PINK);
colors.put(new Color(44, 70, 65), MIConstants.IMAGE_COLOR_BLUE);
colors.put(new Color(220, 220, 220), MIConstants.IMAGE_COLOR_WHITE);
colors.put(new Color(220, 215, 209), MIConstants.IMAGE_COLOR_GRAY);
colors.put(new Color(216, 167, 35), MIConstants.IMAGE_COLOR_YELLOW);
colors.put(new Color(228, 155, 15), MIConstants.IMAGE_COLOR_ORANGE);
colors.put(new Color(197, 131, 46), MIConstants.IMAGE_COLOR_BROWN);
```

```
colors.put(new Color(49, 121, 109), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(231, 123, 117), MIConstants.IMAGE_COLOR_RED);
colors.put(new Color(203, 208, 207), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(192, 191, 199), MIConstants.IMAGE_COLOR_GRAY);
colors.put(new Color(248, 248, 255), MIConstants.IMAGE_COLOR_WHITE);
colors.put(new Color(86, 71, 134), MIConstants.IMAGE_COLOR_PURPLE);
colors.put(new Color(185, 173, 97), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(217, 223, 205), MIConstants.IMAGE_COLOR_GRAY);
colors.put(new Color(248, 234, 202), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(235, 212, 174), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(120, 177, 191), MIConstants.IMAGE_COLOR_BLUE);
colors.put(new Color(95, 129, 81), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(120, 110, 76), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(52, 83, 61), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(255, 215, 0), MIConstants.IMAGE_COLOR_YELLOW);
colors.put(new Color(213, 108, 48), MIConstants.IMAGE_COLOR_ORANGE);
colors.put(new Color(226, 178, 39), MIConstants.IMAGE_COLOR_YELLOW);
colors.put(new Color(202, 129, 54), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(153, 101, 21), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(241, 204, 43), MIConstants.IMAGE_COLOR_YELLOW);
colors.put(new Color(235, 222, 49), MIConstants.IMAGE_COLOR_YELLOW);
colors.put(new Color(249, 215, 126), MIConstants.IMAGE_COLOR_YELLOW);
colors.put(new Color(252, 194, 0), MIConstants.IMAGE_COLOR_YELLOW);
colors.put(new Color(234, 206, 106), MIConstants.IMAGE_COLOR_YELLOW);
colors.put(new Color(255, 193, 82), MIConstants.IMAGE_COLOR_ORANGE);
colors.put(new Color(255, 223, 0), MIConstants.IMAGE_COLOR_YELLOW);
colors.put(new Color(219, 219, 112), MIConstants.IMAGE_COLOR_YELLOW);
colors.put(new Color(55, 51, 50), MIConstants.IMAGE_COLOR_GRAY);
colors.put(new Color(41, 51, 43), MIConstants.IMAGE_COLOR_BLACK);
colors.put(new Color(253, 227, 54), MIConstants.IMAGE_COLOR_YELLOW);
colors.put(new Color(57, 159, 134), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(159, 211, 133), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(105, 136, 144), MIConstants.IMAGE_COLOR_BLUE);
colors.put(new Color(81, 85, 155), MIConstants.IMAGE_COLOR_BLUE);
colors.put(new Color(202, 184, 162), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(255, 205, 115), MIConstants.IMAGE_COLOR_ORANGE);
colors.put(new Color(139, 130, 101), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(197, 231, 205), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(123, 148, 140), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(157, 224, 147), MIConstants.IMAGE_COLOR_GREEN);
```

```
colors.put(new Color(65, 61, 75), MIConstants.IMAGE_COLOR_GRAY);
colors.put(new Color(56, 52, 40), MIConstants.IMAGE_COLOR_BLACK);
colors.put(new Color(74, 75, 70), MIConstants.IMAGE_COLOR_GRAY);
colors.put(new Color(0, 128, 0), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(62, 99, 52), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(57, 61, 42), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(82, 107, 45), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(191, 194, 152), MIConstants.IMAGE_COLOR_GRAY);
colors.put(new Color(38, 98, 66), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(156, 166, 100), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(169, 175, 153), MIConstants.IMAGE_COLOR_GRAY);
colors.put(new Color(35, 65, 78), MIConstants.IMAGE_COLOR_BLUE);
colors.put(new Color(44, 45, 36), MIConstants.IMAGE_COLOR_BLACK);
colors.put(new Color(222, 221, 203), MIConstants.IMAGE_COLOR_GRAY);
colors.put(new Color(173, 255, 47), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(193, 77, 54), MIConstants.IMAGE_COLOR_RED);
colors.put(new Color(128, 128, 128), MIConstants.IMAGE_COLOR_GRAY);
colors.put(new Color(159, 163, 167), MIConstants.IMAGE_COLOR_GRAY);
colors.put(new Color(189, 186, 174), MIConstants.IMAGE_COLOR_GRAY);
colors.put(new Color(209, 211, 204), MIConstants.IMAGE_COLOR_GRAY);
colors.put(new Color(161, 154, 127), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(147, 145, 160), MIConstants.IMAGE_COLOR_BLUE);
colors.put(new Color(70, 89, 69), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(149, 46, 49), MIConstants.IMAGE_COLOR_RED);
colors.put(new Color(52, 63, 92), MIConstants.IMAGE_COLOR_BLUE);
colors.put(new Color(116, 178, 168), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(164, 173, 176), MIConstants.IMAGE_COLOR_GRAY);
colors.put(new Color(172, 201, 178), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(113, 143, 138), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(72, 71, 83), MIConstants.IMAGE_COLOR_GRAY);
colors.put(new Color(44, 53, 57), MIConstants.IMAGE_COLOR_GRAY);
colors.put(new Color(122, 124, 118), MIConstants.IMAGE_COLOR_GRAY);
colors.put(new Color(152, 145, 113), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(158, 128, 34), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(99, 53, 40), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(44, 42, 53), MIConstants.IMAGE_COLOR_BLACK);
colors.put(new Color(237, 231, 200), MIConstants.IMAGE_COLOR_WHITE);
colors.put(new Color(85, 143, 147), MIConstants.IMAGE_COLOR_BLUE);
colors.put(new Color(242, 229, 191), MIConstants.IMAGE_COLOR_YELLOW);
colors.put(new Color(251, 240, 214), MIConstants.IMAGE_COLOR_WHITE);
```

```
colors.put(new Color(241, 234, 215), MIConstants.IMAGE_COLOR_WHITE);
colors.put(new Color(230, 219, 199), MIConstants.IMAGE_COLOR_GRAY);
colors.put(new Color(232, 212, 162), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(82, 24, 250), MIConstants.IMAGE_COLOR_BLUE);
colors.put(new Color(63, 255, 0), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(201, 52, 19), MIConstants.IMAGE_COLOR_RED);
colors.put(new Color(203, 206, 192), MIConstants.IMAGE_COLOR_GRAY);
colors.put(new Color(234, 183, 106), MIConstants.IMAGE_COLOR_ORANGE);
colors.put(new Color(59, 43, 44), MIConstants.IMAGE_COLOR_BLACK);
colors.put(new Color(87, 132, 193), MIConstants.IMAGE_COLOR_BLUE);
colors.put(new Color(153, 82, 43), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(210, 218, 237), MIConstants.IMAGE_COLOR_BLUE);
colors.put(new Color(79, 42, 44), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(174, 187, 193), MIConstants.IMAGE_COLOR_BLUE);
colors.put(new Color(148, 140, 126), MIConstants.IMAGE_COLOR_GRAY);
colors.put(new Color(70, 71, 62), MIConstants.IMAGE_COLOR_GRAY);
colors.put(new Color(223, 115, 255), MIConstants.IMAGE_COLOR_PURPLE);
colors.put(new Color(105, 104, 75), MIConstants.IMAGE_COLOR_GRAY);
colors.put(new Color(152, 125, 115), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(146, 140, 60), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(122, 148, 97), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(167, 160, 126), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(115, 99, 48), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(223, 241, 214), MIConstants.IMAGE_COLOR_WHITE);
colors.put(new Color(245, 239, 235), MIConstants.IMAGE_COLOR_WHITE);
colors.put(new Color(246, 245, 215), MIConstants.IMAGE_COLOR_WHITE);
colors.put(new Color(73, 136, 154), MIConstants.IMAGE_COLOR_BLUE);
colors.put(new Color(96, 138, 90), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(171, 73, 92), MIConstants.IMAGE_COLOR_PINK);
colors.put(new Color(161, 169, 168), MIConstants.IMAGE_COLOR_GRAY);
colors.put(new Color(253, 164, 112), MIConstants.IMAGE_COLOR_ORANGE);
colors.put(new Color(187, 142, 52), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(100, 125, 134), MIConstants.IMAGE_COLOR_BLUE);
colors.put(new Color(37, 52, 43), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(244, 0, 161), MIConstants.IMAGE_COLOR_PINK);
colors.put(new Color(92, 60, 109), MIConstants.IMAGE_COLOR_PURPLE);
colors.put(new Color(240, 255, 240), MIConstants.IMAGE_COLOR_WHITE);
colors.put(new Color(232, 237, 105), MIConstants.IMAGE_COLOR_YELLOW);
colors.put(new Color(205, 109, 147), MIConstants.IMAGE_COLOR_PINK);
colors.put(new Color(100, 136, 148), MIConstants.IMAGE_COLOR_BLUE);
```

```
colors.put(new Color(109, 86, 44), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(129, 91, 40), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(255, 0, 204), MIConstants.IMAGE_COLOR_PINK);
colors.put(new Color(255, 105, 180), MIConstants.IMAGE_COLOR_PINK);
colors.put(new Color(78, 46, 83), MIConstants.IMAGE_COLOR_PURPLE);
colors.put(new Color(167, 117, 44), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(206, 239, 228), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(53, 94, 59), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(139, 126, 119), MIConstants.IMAGE_COLOR_GRAY);
colors.put(new Color(178, 153, 75), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(175, 227, 214), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(202, 225, 217), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(239, 149, 174), MIConstants.IMAGE_COLOR_PINK);
colors.put(new Color(176, 227, 19), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(205, 92, 92), MIConstants.IMAGE_COLOR_RED);
colors.put(new Color(79, 48, 31), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(75, 0, 130), MIConstants.IMAGE_COLOR_PURPLE);
colors.put(new Color(156, 91, 52), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(0, 47, 167), MIConstants.IMAGE_COLOR_BLUE);
colors.put(new Color(255, 79, 0), MIConstants.IMAGE_COLOR_ORANGE);
colors.put(new Color(3, 180, 200), MIConstants.IMAGE_COLOR_BLUE);
colors.put(new Color(98, 66, 43), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(203, 205, 205), MIConstants.IMAGE_COLOR_GRAY);
colors.put(new Color(112, 110, 102), MIConstants.IMAGE_COLOR_GRAY);
colors.put(new Color(134, 80, 64), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(0, 153, 0), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(248, 237, 219), MIConstants.IMAGE_COLOR_WHITE);
colors.put(new Color(255, 255, 240), MIConstants.IMAGE_COLOR_WHITE);
colors.put(new Color(61, 50, 93), MIConstants.IMAGE_COLOR_PURPLE);
colors.put(new Color(65, 54, 40), MIConstants.IMAGE_COLOR_BLACK);
colors.put(new Color(61, 63, 125), MIConstants.IMAGE_COLOR_BLUE);
colors.put(new Color(0, 168, 107), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(226, 121, 69), MIConstants.IMAGE_COLOR_ORANGE);
colors.put(new Color(202, 231, 226), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(63, 46, 76), MIConstants.IMAGE_COLOR_PURPLE);
colors.put(new Color(41, 41, 47), MIConstants.IMAGE_COLOR_BLACK);
colors.put(new Color(103, 72, 52), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(47, 117, 50), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(206, 114, 89), MIConstants.IMAGE_COLOR_ORANGE);
colors.put(new Color(37, 151, 151), MIConstants.IMAGE_COLOR_BLUE);
```

```
colors.put(new Color(95, 44, 47), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(165, 11, 94), MIConstants.IMAGE_COLOR_PINK);
colors.put(new Color(68, 121, 142), MIConstants.IMAGE_COLOR_BLUE);
colors.put(new Color(187, 208, 201), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(19, 104, 67), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(70, 61, 62), MIConstants.IMAGE_COLOR_GRAY);
colors.put(new Color(238, 242, 147), MIConstants.IMAGE_COLOR_WHITE);
colors.put(new Color(122, 170, 224), MIConstants.IMAGE_COLOR_BLUE);
colors.put(new Color(93, 83, 70), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(135, 135, 133), MIConstants.IMAGE_COLOR_GRAY);
colors.put(new Color(41, 171, 135), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(176, 196, 196), MIConstants.IMAGE_COLOR_BLUE);
colors.put(new Color(116, 145, 142), MIConstants.IMAGE_COLOR_BLUE);
colors.put(new Color(220, 191, 172), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(108, 94, 83), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(36, 83, 54), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(197, 195, 176), MIConstants.IMAGE_COLOR_GRAY);
colors.put(new Color(45, 45, 36), MIConstants.IMAGE_COLOR_BLACK);
colors.put(new Color(254, 220, 193), MIConstants.IMAGE_COLOR_ORANGE);
colors.put(new Color(87, 109, 142), MIConstants.IMAGE_COLOR_BLUE);
colors.put(new Color(76, 187, 23), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(77, 80, 60), MIConstants.IMAGE_COLOR_GRAY);
colors.put(new Color(108, 50, 46), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(95, 182, 156), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(240, 230, 140), MIConstants.IMAGE_COLOR_YELLOW);
colors.put(new Color(191, 192, 171), MIConstants.IMAGE_COLOR_GRAY);
colors.put(new Color(58, 53, 50), MIConstants.IMAGE_COLOR_GRAY);
colors.put(new Color(73, 118, 79), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(105, 93, 135), MIConstants.IMAGE_COLOR_PURPLE);
colors.put(new Color(88, 53, 128), MIConstants.IMAGE_COLOR_PURPLE);
colors.put(new Color(224, 147, 171), MIConstants.IMAGE_COLOR_PINK);
colors.put(new Color(123, 120, 90), MIConstants.IMAGE_COLOR_GRAY);
colors.put(new Color(128, 78, 44), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(254, 181, 82), MIConstants.IMAGE_COLOR_ORANGE);
colors.put(new Color(249, 208, 84), MIConstants.IMAGE_COLOR_YELLOW);
colors.put(new Color(66, 137, 41), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(186, 192, 14), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(198, 218, 54), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(198, 169, 94), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(255, 255, 102), MIConstants.IMAGE_COLOR_YELLOW);
```

```
colors.put(new Color(110, 141, 113), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(230, 230, 250), MIConstants.IMAGE_COLOR_WHITE);
colors.put(new Color(204, 204, 255), MIConstants.IMAGE_COLOR_BLUE);
colors.put(new Color(255, 240, 245), MIConstants.IMAGE_COLOR_WHITE);
colors.put(new Color(189, 187, 215), MIConstants.IMAGE_COLOR_PURPLE);
colors.put(new Color(251, 174, 210), MIConstants.IMAGE_COLOR_PINK);
colors.put(new Color(251, 160, 227), MIConstants.IMAGE_COLOR_PINK);
colors.put(new Color(124, 252, 0), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(144, 106, 84), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(253, 233, 16), MIConstants.IMAGE_COLOR_YELLOW);
colors.put(new Color(255, 250, 205), MIConstants.IMAGE_COLOR_WHITE);
colors.put(new Color(150, 132, 40), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(153, 154, 134), MIConstants.IMAGE_COLOR_GRAY);
colors.put(new Color(46, 55, 73), MIConstants.IMAGE_COLOR_BLUE);
colors.put(new Color(173, 216, 230), MIConstants.IMAGE_COLOR_BLACK);
colors.put(new Color(240, 128, 128), MIConstants.IMAGE_COLOR_PINK);
colors.put(new Color(224, 255, 255), MIConstants.IMAGE_COLOR_BLUE);
colors.put(new Color(238, 221, 130), MIConstants.IMAGE_COLOR_YELLOW);
colors.put(new Color(250, 250, 210), MIConstants.IMAGE_COLOR_WHITE);
colors.put(new Color(144, 238, 144), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(211, 211, 211), MIConstants.IMAGE_COLOR_GRAY);
colors.put(new Color(255, 182, 193), MIConstants.IMAGE_COLOR_PINK);
colors.put(new Color(255, 160, 122), MIConstants.IMAGE_COLOR_ORANGE);
colors.put(new Color(32, 178, 170), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(135, 206, 250), MIConstants.IMAGE_COLOR_BLUE);
colors.put(new Color(132, 112, 255), MIConstants.IMAGE_COLOR_BLUE);
colors.put(new Color(119, 136, 153), MIConstants.IMAGE_COLOR_GRAY);
colors.put(new Color(176, 196, 222), MIConstants.IMAGE_COLOR_BLUE);
colors.put(new Color(133, 99, 99), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(255, 255, 224), MIConstants.IMAGE_COLOR_WHITE);
colors.put(new Color(247, 162, 51), MIConstants.IMAGE_COLOR_ORANGE);
colors.put(new Color(200, 162, 200), MIConstants.IMAGE_COLOR_PURPLE);
colors.put(new Color(148, 112, 196), MIConstants.IMAGE_COLOR_PURPLE);
colors.put(new Color(193, 159, 179), MIConstants.IMAGE_COLOR_PURPLE);
colors.put(new Color(233, 238, 235), MIConstants.IMAGE_COLOR_GRAY);
colors.put(new Color(122, 172, 33), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(0, 255, 0), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(50, 205, 50), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(95, 151, 39), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(137, 172, 39), MIConstants.IMAGE_COLOR_GREEN);
```

```
colors.put(new Color(250, 240, 230), MIConstants.IMAGE_COLOR_WHITE);
colors.put(new Color(199, 205, 216), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(150, 44, 84), MIConstants.IMAGE_COLOR_PINK);
colors.put(new Color(83, 75, 79), MIConstants.IMAGE_COLOR_GRAY);
colors.put(new Color(49, 42, 41), MIConstants.IMAGE_COLOR_BLACK);
colors.put(new Color(219, 217, 194), MIConstants.IMAGE_COLOR_GRAY);
colors.put(new Color(179, 187, 183), MIConstants.IMAGE_COLOR_GRAY);
colors.put(new Color(72, 144, 132), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(49, 110, 160), MIConstants.IMAGE_COLOR_BLUE);
colors.put(new Color(162, 165, 128), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(57, 62, 46), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(157, 156, 180), MIConstants.IMAGE_COLOR_PURPLE);
colors.put(new Color(185, 172, 187), MIConstants.IMAGE_COLOR_PURPLE);
colors.put(new Color(174, 148, 171), MIConstants.IMAGE_COLOR_PURPLE);
colors.put(new Color(82, 36, 38), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(139, 80, 75), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(76, 51, 71), MIConstants.IMAGE_COLOR_BLACK);
colors.put(new Color(171, 154, 28), MIConstants.IMAGE_COLOR_YELLOW);
colors.put(new Color(41, 45, 79), MIConstants.IMAGE_COLOR_BLUE);
colors.put(new Color(78, 85, 65), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(120, 46, 44), MIConstants.IMAGE_COLOR_RED);
colors.put(new Color(171, 141, 63), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(105, 125, 137), MIConstants.IMAGE_COLOR_BLUE);
colors.put(new Color(203, 232, 232), MIConstants.IMAGE_COLOR_BLUE);
colors.put(new Color(255, 185, 123), MIConstants.IMAGE_COLOR_ORANGE);
colors.put(new Color(183, 227, 168), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(45, 60, 84), MIConstants.IMAGE_COLOR_BLUE);
colors.put(new Color(71, 62, 35), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(255, 0, 255), MIConstants.IMAGE_COLOR_PINK);
colors.put(new Color(170, 240, 209), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(248, 244, 255), MIConstants.IMAGE_COLOR_WHITE);
colors.put(new Color(202, 52, 53), MIConstants.IMAGE_COLOR_RED);
colors.put(new Color(165, 101, 49), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(42, 41, 34), MIConstants.IMAGE_COLOR_BLACK);
colors.put(new Color(227, 185, 130), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(105, 95, 80), MIConstants.IMAGE_COLOR_GRAY);
colors.put(new Color(80, 85, 85), MIConstants.IMAGE_COLOR_GRAY);
colors.put(new Color(11, 218, 81), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(151, 151, 111), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(102, 183, 225), MIConstants.IMAGE_COLOR_BLUE);
```

```
colors.put(new Color(58, 69, 49), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(165, 151, 132), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(118, 109, 124), MIConstants.IMAGE_COLOR_GRAY);
colors.put(new Color(141, 144, 161), MIConstants.IMAGE_COLOR_BLUE);
colors.put(new Color(181, 123, 46), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(142, 35, 35), MIConstants.IMAGE_COLOR_RED);
colors.put(new Color(205, 82, 91), MIConstants.IMAGE_COLOR_PINK);
colors.put(new Color(245, 183, 153), MIConstants.IMAGE_COLOR_ORANGE);
colors.put(new Color(231, 114, 0), MIConstants.IMAGE_COLOR_ORANGE);
colors.put(new Color(226, 175, 128), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(127, 193, 92), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(150, 167, 147), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(228, 219, 85), MIConstants.IMAGE_COLOR_YELLOW);
colors.put(new Color(53, 34, 53), MIConstants.IMAGE_COLOR_BLACK);
colors.put(new Color(184, 138, 61), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(66, 99, 159), MIConstants.IMAGE_COLOR_BLUE);
colors.put(new Color(128, 0, 0), MIConstants.IMAGE_COLOR_RED);
colors.put(new Color(43, 46, 38), MIConstants.IMAGE_COLOR_BLACK);
colors.put(new Color(183, 168, 163), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(60, 55, 72), MIConstants.IMAGE_COLOR_GRAY);
colors.put(new Color(235, 200, 129), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(87, 83, 75), MIConstants.IMAGE_COLOR_GRAY);
colors.put(new Color(54, 92, 125), MIConstants.IMAGE_COLOR_BLUE);
colors.put(new Color(142, 77, 69), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(82, 75, 75), MIConstants.IMAGE_COLOR_GRAY);
colors.put(new Color(224, 176, 255), MIConstants.IMAGE_COLOR_PURPLE);
colors.put(new Color(145, 95, 109), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(240, 145, 169), MIConstants.IMAGE_COLOR_PINK);
colors.put(new Color(200, 177, 192), MIConstants.IMAGE_COLOR_PURPLE);
colors.put(new Color(115, 194, 251), MIConstants.IMAGE_COLOR_BLUE);
colors.put(new Color(140, 99, 56), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(102, 205, 170), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(0, 0, 205), MIConstants.IMAGE_COLOR_BLUE);
colors.put(new Color(175, 64, 53), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(234, 234, 174), MIConstants.IMAGE_COLOR_YELLOW);
colors.put(new Color(186, 85, 211), MIConstants.IMAGE_COLOR_PURPLE);
colors.put(new Color(147, 112, 219), MIConstants.IMAGE_COLOR_PURPLE);
colors.put(new Color(60, 179, 113), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(123, 104, 238), MIConstants.IMAGE_COLOR_BLUE);
colors.put(new Color(0, 250, 154), MIConstants.IMAGE_COLOR_GREEN);
```

```
colors.put(new Color(72, 209, 204), MIConstants.IMAGE_COLOR_BLUE);
colors.put(new Color(199, 21, 133), MIConstants.IMAGE_COLOR_PINK);
colors.put(new Color(166, 128, 100), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(224, 183, 194), MIConstants.IMAGE_COLOR_PINK);
colors.put(new Color(52, 41, 49), MIConstants.IMAGE_COLOR_BLACK);
colors.put(new Color(254, 186, 173), MIConstants.IMAGE_COLOR_PINK);
colors.put(new Color(195, 185, 221), MIConstants.IMAGE_COLOR_PURPLE);
colors.put(new Color(213, 210, 209), MIConstants.IMAGE_COLOR_GRAY);
colors.put(new Color(225, 219, 208), MIConstants.IMAGE_COLOR_GRAY);
colors.put(new Color(79, 78, 72), MIConstants.IMAGE_COLOR_GRAY);
colors.put(new Color(115, 52, 58), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(85, 74, 60), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(110, 61, 52), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(212, 175, 55), MIConstants.IMAGE_COLOR_YELLOW);
colors.put(new Color(187, 116, 49), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(74, 59, 106), MIConstants.IMAGE_COLOR_PURPLE);
colors.put(new Color(155, 61, 61), MIConstants.IMAGE_COLOR_RED);
colors.put(new Color(102, 106, 109), MIConstants.IMAGE_COLOR_GRAY);
colors.put(new Color(33, 48, 62), MIConstants.IMAGE_COLOR_BLUE);
colors.put(new Color(25, 25, 112), MIConstants.IMAGE_COLOR_BLUE);
colors.put(new Color(33, 38, 58), MIConstants.IMAGE_COLOR_BLACK);
colors.put(new Color(36, 46, 40), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(63, 54, 35), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(246, 244, 147), MIConstants.IMAGE_COLOR_YELLOW);
colors.put(new Color(158, 51, 50), MIConstants.IMAGE_COLOR_RED);
colors.put(new Color(243, 229, 192), MIConstants.IMAGE_COLOR_YELLOW);
colors.put(new Color(220, 217, 205), MIConstants.IMAGE_COLOR_GRAY);
colors.put(new Color(89, 86, 72), MIConstants.IMAGE_COLOR_GRAY);
colors.put(new Color(245, 245, 204), MIConstants.IMAGE_COLOR_WHITE);
colors.put(new Color(218, 234, 111), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(55, 62, 65), MIConstants.IMAGE_COLOR_BLACK);
colors.put(new Color(80, 99, 85), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(64, 117, 119), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(62, 50, 103), MIConstants.IMAGE_COLOR_BLUE);
colors.put(new Color(245, 255, 250), MIConstants.IMAGE_COLOR_WHITE);
colors.put(new Color(152, 255, 152), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(224, 216, 167), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(198, 234, 221), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(55, 63, 67), MIConstants.IMAGE_COLOR_GRAY);
colors.put(new Color(165, 169, 178), MIConstants.IMAGE_COLOR_GRAY);
```

```
colors.put(new Color(186, 185, 169), MIConstants.IMAGE_COLOR_GRAY);
colors.put(new Color(255, 228, 225), MIConstants.IMAGE_COLOR_PINK);
colors.put(new Color(96, 90, 103), MIConstants.IMAGE_COLOR_GRAY);
colors.put(new Color(88, 47, 43), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(255, 228, 181), MIConstants.IMAGE_COLOR_ORANGE);
colors.put(new Color(111, 55, 45), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(151, 70, 60), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(255, 152, 137), MIConstants.IMAGE_COLOR_PINK);
colors.put(new Color(107, 37, 44), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(85, 77, 66), MIConstants.IMAGE_COLOR_GRAY);
colors.put(new Color(165, 139, 111), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(122, 118, 121), MIConstants.IMAGE_COLOR_GRAY);
colors.put(new Color(57, 59, 60), MIConstants.IMAGE_COLOR_GRAY);
colors.put(new Color(122, 197, 180), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(131, 120, 199), MIConstants.IMAGE_COLOR_PURPLE);
colors.put(new Color(245, 243, 206), MIConstants.IMAGE_COLOR_WHITE);
colors.put(new Color(206, 205, 184), MIConstants.IMAGE_COLOR_GRAY);
colors.put(new Color(192, 178, 215), MIConstants.IMAGE_COLOR_PURPLE);
colors.put(new Color(240, 196, 32), MIConstants.IMAGE_COLOR_YELLOW);
colors.put(new Color(158, 209, 211), MIConstants.IMAGE_COLOR_BLUE);
colors.put(new Color(68, 45, 33), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(86, 80, 81), MIConstants.IMAGE_COLOR_GRAY);
colors.put(new Color(0, 95, 91), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(173, 223, 173), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(26, 179, 133), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(160, 159, 156), MIConstants.IMAGE_COLOR_GRAY);
colors.put(new Color(153, 122, 141), MIConstants.IMAGE_COLOR_PURPLE);
colors.put(new Color(169, 132, 79), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(158, 126, 83), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(197, 75, 140), MIConstants.IMAGE_COLOR_PINK);
colors.put(new Color(136, 79, 64), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(82, 77, 91), MIConstants.IMAGE_COLOR_GRAY);
colors.put(new Color(255, 219, 88), MIConstants.IMAGE_COLOR_YELLOW);
colors.put(new Color(214, 139, 128), MIConstants.IMAGE_COLOR_RED);
colors.put(new Color(253, 174, 69), MIConstants.IMAGE_COLOR_ORANGE);
colors.put(new Color(33, 66, 30), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(216, 221, 218), MIConstants.IMAGE_COLOR_GRAY);
colors.put(new Color(78, 93, 78), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(163, 154, 135), MIConstants.IMAGE_COLOR_GRAY);
colors.put(new Color(233, 230, 220), MIConstants.IMAGE_COLOR_WHITE);
```

```
colors.put(new Color(255, 222, 173), MIConstants.IMAGE_COLOR_ORANGE);
colors.put(new Color(0, 0, 128), MIConstants.IMAGE_COLOR_BLUE);
colors.put(new Color(0, 102, 204), MIConstants.IMAGE_COLOR_BLUE);
colors.put(new Color(184, 198, 190), MIConstants.IMAGE_COLOR_GRAY);
colors.put(new Color(238, 199, 162), MIConstants.IMAGE_COLOR_ORANGE);
colors.put(new Color(77, 77, 255), MIConstants.IMAGE_COLOR_BLUE);
colors.put(new Color(255, 153, 51), MIConstants.IMAGE_COLOR_ORANGE);
colors.put(new Color(255, 110, 199), MIConstants.IMAGE_COLOR_PINK);
colors.put(new Color(147, 170, 185), MIConstants.IMAGE_COLOR_BLUE);
colors.put(new Color(119, 168, 171), MIConstants.IMAGE_COLOR_BLUE);
colors.put(new Color(37, 37, 37), MIConstants.IMAGE_COLOR_GRAY);
colors.put(new Color(170, 165, 131), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(102, 111, 111), MIConstants.IMAGE_COLOR_GRAY);
colors.put(new Color(109, 59, 36), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(0, 0, 156), MIConstants.IMAGE_COLOR_BLUE);
colors.put(new Color(228, 195, 133), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(235, 199, 158), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(221, 131, 116), MIConstants.IMAGE_COLOR_RED);
colors.put(new Color(41, 169, 139), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(51, 46, 46), MIConstants.IMAGE_COLOR_BLACK);
colors.put(new Color(162, 61, 84), MIConstants.IMAGE_COLOR_PINK);
colors.put(new Color(37, 63, 78), MIConstants.IMAGE_COLOR_BLUE);
colors.put(new Color(169, 157, 157), MIConstants.IMAGE_COLOR_GRAY);
colors.put(new Color(161, 153, 134), MIConstants.IMAGE_COLOR_GRAY);
colors.put(new Color(29, 57, 60), MIConstants.IMAGE_COLOR_BLUE);
colors.put(new Color(164, 184, 143), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(188, 146, 41), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(126, 74, 59), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(252, 237, 197), MIConstants.IMAGE_COLOR_WHITE);
colors.put(new Color(0, 143, 112), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(76, 169, 115), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(204, 119, 34), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(223, 240, 226), MIConstants.IMAGE_COLOR_WHITE);
colors.put(new Color(250, 243, 220), MIConstants.IMAGE_COLOR_WHITE);
colors.put(new Color(49, 51, 48), MIConstants.IMAGE_COLOR_BLACK);
colors.put(new Color(138, 51, 53), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(115, 80, 59), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(207, 181, 59), MIConstants.IMAGE_COLOR_YELLOW);
colors.put(new Color(253, 245, 230), MIConstants.IMAGE_COLOR_WHITE);
colors.put(new Color(121, 104, 120), MIConstants.IMAGE_COLOR_GRAY);
```

```
colors.put(new Color(192, 46, 76), MIConstants.IMAGE_COLOR_RED);
colors.put(new Color(128, 128, 0), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(107, 142, 35), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(181, 179, 92), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(136, 128, 100), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(116, 112, 40), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(154, 185, 115), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(194, 230, 236), MIConstants.IMAGE_COLOR_BLUE);
colors.put(new Color(72, 65, 43), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(168, 195, 188), MIConstants.IMAGE_COLOR_BLUE);
colors.put(new Color(152, 126, 126), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(57, 85, 85), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(255, 165, 0), MIConstants.IMAGE_COLOR_ORANGE);
colors.put(new Color(255, 69, 0), MIConstants.IMAGE_COLOR_ORANGE);
colors.put(new Color(168, 83, 53), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(234, 227, 205), MIConstants.IMAGE_COLOR_YELLOW);
colors.put(new Color(218, 112, 214), MIConstants.IMAGE_COLOR_PINK);
colors.put(new Color(241, 235, 217), MIConstants.IMAGE_COLOR_WHITE);
colors.put(new Color(37, 91, 119), MIConstants.IMAGE_COLOR_BLUE);
colors.put(new Color(194, 142, 136), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(210, 211, 179), MIConstants.IMAGE_COLOR_GRAY);
colors.put(new Color(129, 137, 136), MIConstants.IMAGE_COLOR_GRAY);
colors.put(new Color(211, 219, 203), MIConstants.IMAGE_COLOR_GRAY);
colors.put(new Color(45, 56, 58), MIConstants.IMAGE_COLOR_GRAY);
colors.put(new Color(255, 96, 55), MIConstants.IMAGE_COLOR_ORANGE);
colors.put(new Color(40, 53, 58), MIConstants.IMAGE_COLOR_BLUE);
colors.put(new Color(109, 154, 120), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(209, 234, 234), MIConstants.IMAGE_COLOR_BLUE);
colors.put(new Color(212, 181, 176), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(134, 75, 54), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(122, 113, 92), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(0, 157, 196), MIConstants.IMAGE_COLOR_BLUE);
colors.put(new Color(79, 64, 55), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(126, 179, 148), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(104, 40, 96), MIConstants.IMAGE_COLOR_PURPLE);
colors.put(new Color(152, 118, 84), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(221, 173, 175), MIConstants.IMAGE_COLOR_RED);
colors.put(new Color(171, 205, 239), MIConstants.IMAGE_COLOR_BLUE);
colors.put(new Color(238, 232, 170), MIConstants.IMAGE_COLOR_YELLOW);
colors.put(new Color(152, 251, 152), MIConstants.IMAGE_COLOR_GREEN);
```

```
colors.put(new Color(189, 202, 168), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(249, 132, 229), MIConstants.IMAGE_COLOR_PINK);
colors.put(new Color(156, 141, 114), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(250, 218, 221), MIConstants.IMAGE_COLOR_PINK);
colors.put(new Color(249, 245, 159), MIConstants.IMAGE_COLOR_YELLOW);
colors.put(new Color(239, 214, 218), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(99, 109, 112), MIConstants.IMAGE_COLOR_GRAY);
colors.put(new Color(195, 190, 187), MIConstants.IMAGE_COLOR_GRAY);
colors.put(new Color(188, 152, 126), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(175, 238, 238), MIConstants.IMAGE_COLOR_BLUE);
colors.put(new Color(219, 112, 147), MIConstants.IMAGE_COLOR_PINK);
colors.put(new Color(32, 57, 44), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(54, 72, 47), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(234, 228, 220), MIConstants.IMAGE_COLOR_WHITE);
colors.put(new Color(235, 247, 228), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(223, 185, 146), MIConstants.IMAGE_COLOR_ORANGE);
colors.put(new Color(84, 79, 58), MIConstants.IMAGE_COLOR_YELLOW);
colors.put(new Color(255, 239, 213), MIConstants.IMAGE_COLOR_YELLOW);
colors.put(new Color(124, 45, 55), MIConstants.IMAGE_COLOR_RED);
colors.put(new Color(72, 128, 132), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(208, 200, 176), MIConstants.IMAGE_COLOR_YELLOW);
colors.put(new Color(251, 235, 80), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(49, 39, 96), MIConstants.IMAGE_COLOR_PURPLE);
colors.put(new Color(191, 205, 192), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(48, 93, 53), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(119, 221, 119), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(99, 146, 131), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(211, 229, 239), MIConstants.IMAGE_COLOR_BLUE);
colors.put(new Color(42, 37, 81), MIConstants.IMAGE_COLOR_BLUE);
colors.put(new Color(186, 171, 135), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(64, 64, 72), MIConstants.IMAGE_COLOR_GRAY);
colors.put(new Color(255, 203, 164), MIConstants.IMAGE_COLOR_ORANGE);
colors.put(new Color(255, 218, 185), MIConstants.IMAGE_COLOR_ORANGE);
colors.put(new Color(255, 204, 153), MIConstants.IMAGE_COLOR_ORANGE);
colors.put(new Color(250, 223, 173), MIConstants.IMAGE_COLOR_ORANGE);
colors.put(new Color(122, 68, 52), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(209, 226, 49), MIConstants.IMAGE_COLOR_YELLOW);
colors.put(new Color(222, 209, 198), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(234, 224, 200), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(118, 109, 82), MIConstants.IMAGE_COLOR_BROWN);
```

```
colors.put(new Color(37, 153, 178), MIConstants.IMAGE_COLOR_BLUE);
colors.put(new Color(215, 231, 208), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(172, 185, 232), MIConstants.IMAGE_COLOR_BLUE);
colors.put(new Color(194, 169, 219), MIConstants.IMAGE_COLOR_PURPLE);
colors.put(new Color(172, 182, 178), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(195, 205, 230), MIConstants.IMAGE_COLOR_BLUE);
colors.put(new Color(28, 57, 187), MIConstants.IMAGE_COLOR_BLUE);
colors.put(new Color(0, 166, 147), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(50, 18, 122), MIConstants.IMAGE_COLOR_BLUE);
colors.put(new Color(247, 127, 190), MIConstants.IMAGE_COLOR_PINK);
colors.put(new Color(104, 51, 50), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(204, 51, 51), MIConstants.IMAGE_COLOR_RED);
colors.put(new Color(254, 40, 162), MIConstants.IMAGE_COLOR_PINK);
colors.put(new Color(236, 88, 0), MIConstants.IMAGE_COLOR_ORANGE);
colors.put(new Color(205, 133, 63), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(115, 61, 31), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(122, 114, 41), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(218, 151, 144), MIConstants.IMAGE_COLOR_RED);
colors.put(new Color(145, 160, 146), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(130, 102, 99), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(248, 234, 151), MIConstants.IMAGE_COLOR_YELLOW);
colors.put(new Color(91, 160, 208), MIConstants.IMAGE_COLOR_BLUE);
colors.put(new Color(253, 215, 228), MIConstants.IMAGE_COLOR_PINK);
colors.put(new Color(0, 165, 80), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(117, 101, 86), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(189, 192, 126), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(1, 121, 111), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(42, 47, 35), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(255, 192, 203), MIConstants.IMAGE_COLOR_PINK);
colors.put(new Color(255, 102, 255), MIConstants.IMAGE_COLOR_PINK);
colors.put(new Color(216, 180, 182), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(246, 204, 215), MIConstants.IMAGE_COLOR_RED);
colors.put(new Color(243, 215, 182), MIConstants.IMAGE_COLOR_ORANGE);
colors.put(new Color(191, 179, 178), MIConstants.IMAGE_COLOR_GRAY);
colors.put(new Color(157, 84, 50), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(245, 230, 196), MIConstants.IMAGE_COLOR_YELLOW);
colors.put(new Color(252, 219, 210), MIConstants.IMAGE_COLOR_RED);
colors.put(new Color(186, 120, 42), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(187, 205, 165), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(229, 127, 61), MIConstants.IMAGE_COLOR_ORANGE);
```

```
colors.put(new Color(191, 141, 60), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(62, 89, 76), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(221, 160, 221), MIConstants.IMAGE_COLOR_PURPLE);
colors.put(new Color(101, 28, 38), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(229, 242, 231), MIConstants.IMAGE_COLOR_WHITE);
colors.put(new Color(138, 167, 204), MIConstants.IMAGE_COLOR_BLUE);
colors.put(new Color(106, 31, 68), MIConstants.IMAGE_COLOR_PURPLE);
colors.put(new Color(221, 220, 219), MIConstants.IMAGE_COLOR_GRAY);
colors.put(new Color(223, 157, 91), MIConstants.IMAGE_COLOR_ORANGE);
colors.put(new Color(59, 67, 108), MIConstants.IMAGE_COLOR_BLUE);
colors.put(new Color(244, 240, 155), MIConstants.IMAGE_COLOR_YELLOW);
colors.put(new Color(139, 152, 216), MIConstants.IMAGE_COLOR_BLUE);
colors.put(new Color(240, 213, 85), MIConstants.IMAGE_COLOR_YELLOW);
colors.put(new Color(239, 220, 212), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(132, 92, 64), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(176, 224, 230), MIConstants.IMAGE_COLOR_BLUE);
colors.put(new Color(136, 60, 50), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(202, 180, 212), MIConstants.IMAGE_COLOR_PURPLE);
colors.put(new Color(226, 205, 213), MIConstants.IMAGE_COLOR_PURPLE);
colors.put(new Color(228, 222, 142), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(248, 246, 223), MIConstants.IMAGE_COLOR_WHITE);
colors.put(new Color(246, 227, 218), MIConstants.IMAGE_COLOR_WHITE);
colors.put(new Color(0, 51, 102), MIConstants.IMAGE_COLOR_BLUE);
colors.put(new Color(221, 0, 255), MIConstants.IMAGE_COLOR_PURPLE);
colors.put(new Color(204, 136, 153), MIConstants.IMAGE_COLOR_PINK);
colors.put(new Color(110, 51, 38), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(89, 186, 163), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(186, 192, 180), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(255, 117, 24), MIConstants.IMAGE_COLOR_ORANGE);
colors.put(new Color(83, 73, 49), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(128, 0, 128), MIConstants.IMAGE_COLOR_PURPLE);
colors.put(new Color(101, 45, 193), MIConstants.IMAGE_COLOR_BLUE);
colors.put(new Color(150, 120, 182), MIConstants.IMAGE_COLOR_PURPLE);
colors.put(new Color(80, 64, 77), MIConstants.IMAGE_COLOR_GRAY);
colors.put(new Color(205, 174, 112), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(242, 237, 221), MIConstants.IMAGE_COLOR_WHITE);
colors.put(new Color(235, 226, 210), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(217, 217, 243), MIConstants.IMAGE_COLOR_BLUE);
colors.put(new Color(195, 152, 139), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(203, 201, 192), MIConstants.IMAGE_COLOR_GRAY);
```

```
colors.put(new Color(106, 84, 69), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(35, 47, 44), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(255, 53, 94), MIConstants.IMAGE_COLOR_PINK);
colors.put(new Color(220, 198, 160), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(102, 112, 40), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(179, 193, 177), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(252, 174, 96), MIConstants.IMAGE_COLOR_ORANGE);
colors.put(new Color(43, 46, 37), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(111, 116, 123), MIConstants.IMAGE_COLOR_GRAY);
colors.put(new Color(210, 125, 70), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(115, 74, 18), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(255, 51, 204), MIConstants.IMAGE_COLOR_PINK);
colors.put(new Color(227, 11, 92), MIConstants.IMAGE_COLOR_PINK);
colors.put(new Color(69, 52, 48), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(255, 0, 0), MIConstants.IMAGE_COLOR_RED);
colors.put(new Color(112, 31, 40), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(203, 111, 74), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(102, 42, 44), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(255, 63, 52), MIConstants.IMAGE_COLOR_RED);
colors.put(new Color(93, 31, 30), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(125, 65, 56), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(173, 82, 46), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(187, 51, 133), MIConstants.IMAGE_COLOR_PINK);
colors.put(new Color(91, 52, 46), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(209, 239, 159), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(169, 141, 54), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(32, 63, 88), MIConstants.IMAGE_COLOR_BLUE);
colors.put(new Color(121, 132, 136), MIConstants.IMAGE_COLOR_BLUE);
colors.put(new Color(160, 205, 217), MIConstants.IMAGE_COLOR_BLUE);
colors.put(new Color(246, 222, 218), MIConstants.IMAGE_COLOR_RED);
colors.put(new Color(178, 110, 51), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(50, 63, 117), MIConstants.IMAGE_COLOR_BLUE);
colors.put(new Color(55, 54, 63), MIConstants.IMAGE_COLOR_GRAY);
colors.put(new Color(61, 70, 83), MIConstants.IMAGE_COLOR_GRAY);
colors.put(new Color(239, 236, 222), MIConstants.IMAGE_COLOR_WHITE);
colors.put(new Color(239, 245, 209), MIConstants.IMAGE_COLOR_WHITE);
colors.put(new Color(89, 89, 171), MIConstants.IMAGE_COLOR_BLUE);
colors.put(new Color(161, 82, 38), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(183, 198, 26), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(137, 217, 200), MIConstants.IMAGE_COLOR_GREEN);
```

```
colors.put(new Color(221, 173, 86), MIConstants.IMAGE_COLOR_ORANGE);
colors.put(new Color(0, 204, 204), MIConstants.IMAGE_COLOR_BLUE);
colors.put(new Color(90, 77, 65), MIConstants.IMAGE_COLOR_GRAY);
colors.put(new Color(147, 162, 186), MIConstants.IMAGE_COLOR_BLUE);
colors.put(new Color(157, 68, 45), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(199, 163, 132), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(109, 120, 118), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(216, 98, 91), MIConstants.IMAGE_COLOR_RED);
colors.put(new Color(125, 103, 87), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(255, 198, 158), MIConstants.IMAGE_COLOR_ORANGE);
colors.put(new Color(234, 184, 82), MIConstants.IMAGE_COLOR_ORANGE);
colors.put(new Color(161, 71, 67), MIConstants.IMAGE_COLOR_RED);
colors.put(new Color(142, 89, 60), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(254, 171, 154), MIConstants.IMAGE_COLOR_RED);
colors.put(new Color(138, 45, 82), MIConstants.IMAGE_COLOR_PURPLE);
colors.put(new Color(144, 93, 93), MIConstants.IMAGE_COLOR_PURPLE);
colors.put(new Color(251, 238, 232), MIConstants.IMAGE_COLOR_WHITE);
colors.put(new Color(188, 143, 143), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(169, 64, 100), MIConstants.IMAGE_COLOR_PURPLE);
colors.put(new Color(65, 105, 225), MIConstants.IMAGE_COLOR_BLUE);
colors.put(new Color(181, 75, 115), MIConstants.IMAGE_COLOR_PINK);
colors.put(new Color(107, 63, 160), MIConstants.IMAGE_COLOR_PURPLE);
colors.put(new Color(224, 17, 95), MIConstants.IMAGE_COLOR_RED);
colors.put(new Color(241, 237, 212), MIConstants.IMAGE_COLOR_WHITE);
colors.put(new Color(128, 70, 27), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(125, 101, 92), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(183, 65, 14), MIConstants.IMAGE_COLOR_RED);
colors.put(new Color(139, 69, 19), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(255, 102, 0), MIConstants.IMAGE_COLOR_ORANGE);
colors.put(new Color(244, 196, 48), MIConstants.IMAGE_COLOR_YELLOW);
colors.put(new Color(152, 159, 122), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(183, 152, 38), MIConstants.IMAGE_COLOR_YELLOW);
colors.put(new Color(165, 206, 236), MIConstants.IMAGE_COLOR_BLUE);
colors.put(new Color(23, 123, 77), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(250, 128, 114), MIConstants.IMAGE_COLOR_RED);
colors.put(new Color(255, 214, 123), MIConstants.IMAGE_COLOR_YELLOW);
colors.put(new Color(44, 110, 49), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(68, 87, 97), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(78, 108, 157), MIConstants.IMAGE_COLOR_BLUE);
colors.put(new Color(175, 147, 125), MIConstants.IMAGE_COLOR_BROWN);
```

```
colors.put(new Color(222, 203, 129), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(254, 219, 183), MIConstants.IMAGE_COLOR_ORANGE);
colors.put(new Color(244, 164, 96), MIConstants.IMAGE_COLOR_ORANGE);
colors.put(new Color(146, 0, 10), MIConstants.IMAGE_COLOR_RED);
colors.put(new Color(153, 152, 167), MIConstants.IMAGE_COLOR_GRAY);
colors.put(new Color(225, 213, 166), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(8, 37, 103), MIConstants.IMAGE_COLOR_BLUE);
colors.put(new Color(85, 91, 44), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(244, 234, 228), MIConstants.IMAGE_COLOR_WHITE);
colors.put(new Color(245, 222, 196), MIConstants.IMAGE_COLOR_ORANGE);
colors.put(new Color(111, 99, 160), MIConstants.IMAGE_COLOR_PURPLE);
colors.put(new Color(173, 217, 209), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(255, 36, 0), MIConstants.IMAGE_COLOR_RED);
colors.put(new Color(74, 45, 87), MIConstants.IMAGE_COLOR_BLACK);
colors.put(new Color(126, 37, 48), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(107, 106, 108), MIConstants.IMAGE_COLOR_GRAY);
colors.put(new Color(135, 135, 111), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(255, 216, 0), MIConstants.IMAGE_COLOR_YELLOW);
colors.put(new Color(48, 142, 160), MIConstants.IMAGE_COLOR_BLUE);
colors.put(new Color(106, 100, 102), MIConstants.IMAGE_COLOR_GRAY);
colors.put(new Color(238, 231, 200), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(102, 255, 102), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(61, 64, 49), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(239, 149, 72), MIConstants.IMAGE_COLOR_ORANGE);
colors.put(new Color(223, 221, 214), MIConstants.IMAGE_COLOR_GRAY);
colors.put(new Color(46, 139, 87), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(194, 213, 196), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(138, 174, 164), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(219, 129, 126), MIConstants.IMAGE_COLOR_RED);
colors.put(new Color(119, 183, 208), MIConstants.IMAGE_COLOR_BLUE);
colors.put(new Color(50, 20, 20), MIConstants.IMAGE_COLOR_BLACK);
colors.put(new Color(105, 50, 110), MIConstants.IMAGE_COLOR_PURPLE);
colors.put(new Color(255, 245, 238), MIConstants.IMAGE_COLOR_WHITE);
colors.put(new Color(55, 65, 42), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(255, 186, 0), MIConstants.IMAGE_COLOR_ORANGE);
colors.put(new Color(107, 66, 38), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(252, 233, 215), MIConstants.IMAGE_COLOR_WHITE);
colors.put(new Color(154, 192, 182), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(96, 154, 184), MIConstants.IMAGE_COLOR_BLUE);
colors.put(new Color(248, 246, 168), MIConstants.IMAGE_COLOR_YELLOW);
```

```
colors.put(new Color(51, 204, 153), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(0, 158, 96), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(52, 54, 58), MIConstants.IMAGE_COLOR_GRAY);
colors.put(new Color(0, 73, 78), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(27, 70, 54), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(230, 178, 166), MIConstants.IMAGE_COLOR_RED);
colors.put(new Color(116, 89, 55), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(121, 136, 171), MIConstants.IMAGE_COLOR_BLUE);
colors.put(new Color(78, 78, 76), MIConstants.IMAGE_COLOR_GRAY);
colors.put(new Color(132, 40, 51), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(232, 153, 190), MIConstants.IMAGE_COLOR_PINK);
colors.put(new Color(252, 15, 192), MIConstants.IMAGE_COLOR_PINK);
colors.put(new Color(97, 102, 107), MIConstants.IMAGE_COLOR_GRAY);
colors.put(new Color(104, 107, 80), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(233, 217, 169), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(160, 82, 45), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(187, 173, 161), MIConstants.IMAGE_COLOR_GRAY);
colors.put(new Color(192, 192, 192), MIConstants.IMAGE_COLOR_GRAY);
colors.put(new Color(103, 190, 144), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(166, 213, 208), MIConstants.IMAGE_COLOR_BLUE);
colors.put(new Color(105, 41, 59), MIConstants.IMAGE_COLOR_PURPLE);
colors.put(new Color(104, 118, 110), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(157, 180, 170), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(135, 206, 235), MIConstants.IMAGE_COLOR_BLUE);
colors.put(new Color(106, 90, 205), MIConstants.IMAGE_COLOR_BLUE);
colors.put(new Color(112, 128, 144), MIConstants.IMAGE_COLOR_BLUE);
colors.put(new Color(0, 51, 153), MIConstants.IMAGE_COLOR_BLUE);
colors.put(new Color(73, 98, 103), MIConstants.IMAGE_COLOR_BLUE);
colors.put(new Color(187, 95, 52), MIConstants.IMAGE_COLOR_ORANGE);
colors.put(new Color(255, 250, 250), MIConstants.IMAGE_COLOR_WHITE);
colors.put(new Color(227, 227, 220), MIConstants.IMAGE_COLOR_GRAY);
colors.put(new Color(234, 247, 201), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(214, 240, 205), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(228, 215, 229), MIConstants.IMAGE_COLOR_PURPLE);
colors.put(new Color(236, 229, 218), MIConstants.IMAGE_COLOR_GRAY);
colors.put(new Color(207, 190, 165), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(133, 73, 76), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(233, 236, 241), MIConstants.IMAGE_COLOR_WHITE);
colors.put(new Color(221, 107, 56), MIConstants.IMAGE_COLOR_ORANGE);
colors.put(new Color(201, 181, 154), MIConstants.IMAGE_COLOR_BROWN);
```

```
colors.put(new Color(111, 99, 75), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(75, 67, 59), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(123, 137, 118), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(55, 93, 79), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(108, 79, 63), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(139, 95, 77), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(255, 28, 174), MIConstants.IMAGE_COLOR_PINK);
colors.put(new Color(179, 196, 216), MIConstants.IMAGE_COLOR_BLUE);
colors.put(new Color(126, 205, 221), MIConstants.IMAGE_COLOR_BLUE);
colors.put(new Color(167, 252, 0), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(0, 255, 127), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(163, 189, 156), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(241, 241, 198), MIConstants.IMAGE_COLOR_YELLOW);
colors.put(new Color(184, 202, 157), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(143, 125, 107), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(50, 84, 130), MIConstants.IMAGE_COLOR_BLUE);
colors.put(new Color(133, 136, 133), MIConstants.IMAGE_COLOR_GRAY);
colors.put(new Color(160, 161, 151), MIConstants.IMAGE_COLOR_GRAY);
colors.put(new Color(227, 221, 57), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(70, 130, 180), MIConstants.IMAGE_COLOR_BLUE);
colors.put(new Color(67, 70, 75), MIConstants.IMAGE_COLOR_GRAY);
colors.put(new Color(131, 61, 62), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(148, 106, 129), MIConstants.IMAGE_COLOR_PURPLE);
colors.put(new Color(64, 99, 86), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(114, 74, 161), MIConstants.IMAGE_COLOR_PURPLE);
colors.put(new Color(140, 156, 156), MIConstants.IMAGE_COLOR_BLUE);
colors.put(new Color(238, 239, 223), MIConstants.IMAGE_COLOR_GRAY);
colors.put(new Color(198, 234, 128), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(143, 182, 156), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(56, 176, 222), MIConstants.IMAGE_COLOR_BLUE);
colors.put(new Color(239, 142, 56), MIConstants.IMAGE_COLOR_ORANGE);
colors.put(new Color(248, 175, 169), MIConstants.IMAGE_COLOR_PINK);
colors.put(new Color(218, 192, 26), MIConstants.IMAGE_COLOR_YELLOW);
colors.put(new Color(199, 97, 85), MIConstants.IMAGE_COLOR_RED);
colors.put(new Color(255, 204, 51), MIConstants.IMAGE_COLOR_ORANGE);
colors.put(new Color(192, 81, 74), MIConstants.IMAGE_COLOR_RED);
colors.put(new Color(254, 76, 64), MIConstants.IMAGE_COLOR_RED);
colors.put(new Color(250, 157, 73), MIConstants.IMAGE_COLOR_ORANGE);
colors.put(new Color(255, 180, 55), MIConstants.IMAGE_COLOR_ORANGE);
colors.put(new Color(184, 212, 187), MIConstants.IMAGE_COLOR_GREEN);
```

```
colors.put(new Color(195, 214, 189), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(0, 123, 119), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(124, 159, 47), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(139, 134, 133), MIConstants.IMAGE_COLOR_GRAY);
colors.put(new Color(37, 47, 47), MIConstants.IMAGE_COLOR_BLACK);
colors.put(new Color(218, 230, 221), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(249, 225, 118), MIConstants.IMAGE_COLOR_YELLOW);
colors.put(new Color(238, 145, 141), MIConstants.IMAGE_COLOR_PINK);
colors.put(new Color(215, 206, 197), MIConstants.IMAGE_COLOR_GRAY);
colors.put(new Color(219, 208, 202), MIConstants.IMAGE_COLOR_GRAY);
colors.put(new Color(246, 174, 120), MIConstants.IMAGE_COLOR_ORANGE);
colors.put(new Color(220, 114, 42), MIConstants.IMAGE_COLOR_ORANGE);
colors.put(new Color(216, 204, 155), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(133, 53, 52), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(210, 180, 140), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(30, 47, 60), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(242, 133, 0), MIConstants.IMAGE_COLOR_ORANGE);
colors.put(new Color(255, 204, 0), MIConstants.IMAGE_COLOR_YELLOW);
colors.put(new Color(212, 111, 49), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(179, 112, 132), MIConstants.IMAGE_COLOR_PINK);
colors.put(new Color(222, 241, 221), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(37, 60, 72), MIConstants.IMAGE_COLOR_BLUE);
colors.put(new Color(186, 192, 179), MIConstants.IMAGE_COLOR_GRAY);
colors.put(new Color(72, 60, 50), MIConstants.IMAGE_COLOR_GRAY);
colors.put(new Color(139, 133, 137), MIConstants.IMAGE_COLOR_GRAY);
colors.put(new Color(100, 58, 72), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(73, 101, 105), MIConstants.IMAGE_COLOR_BLUE);
colors.put(new Color(43, 75, 64), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(208, 240, 192), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(248, 131, 194), MIConstants.IMAGE_COLOR_PINK);
colors.put(new Color(0, 128, 128), MIConstants.IMAGE_COLOR_BLUE);
colors.put(new Color(37, 72, 85), MIConstants.IMAGE_COLOR_BLUE);
colors.put(new Color(60, 33, 38), MIConstants.IMAGE_COLOR_BLACK);
colors.put(new Color(205, 87, 0), MIConstants.IMAGE_COLOR_ORANGE);
colors.put(new Color(244, 208, 164), MIConstants.IMAGE_COLOR_ORANGE);
colors.put(new Color(226, 114, 91), MIConstants.IMAGE_COLOR_RED);
colors.put(new Color(236, 230, 126), MIConstants.IMAGE_COLOR_YELLOW);
colors.put(new Color(252, 176, 87), MIConstants.IMAGE_COLOR_ORANGE);
colors.put(new Color(177, 148, 143), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(216, 191, 216), MIConstants.IMAGE_COLOR_PURPLE);
```

```
colors.put(new Color(77, 77, 75), MIConstants.IMAGE_COLOR_GRAY);
colors.put(new Color(146, 56, 48), MIConstants.IMAGE_COLOR_RED);
colors.put(new Color(185, 195, 190), MIConstants.IMAGE_COLOR_GRAY);
colors.put(new Color(24, 67, 67), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(252, 128, 165), MIConstants.IMAGE_COLOR_PINK);
colors.put(new Color(240, 245, 144), MIConstants.IMAGE_COLOR_YELLOW);
colors.put(new Color(50, 67, 54), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(217, 214, 207), MIConstants.IMAGE_COLOR_GRAY);
colors.put(new Color(221, 214, 225), MIConstants.IMAGE_COLOR_PURPLE);
colors.put(new Color(159, 113, 95), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(62, 38, 49), MIConstants.IMAGE_COLOR_BLACK);
colors.put(new Color(45, 37, 65), MIConstants.IMAGE_COLOR_BLACK);
colors.put(new Color(79, 99, 72), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(255, 99, 71), MIConstants.IMAGE_COLOR_RED);
colors.put(new Color(231, 158, 136), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(253, 14, 53), MIConstants.IMAGE_COLOR_RED);
colors.put(new Color(53, 61, 117), MIConstants.IMAGE_COLOR_BLUE);
colors.put(new Color(55, 78, 136), MIConstants.IMAGE_COLOR_BLUE);
colors.put(new Color(156, 172, 165), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(109, 175, 167), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(221, 237, 233), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(226, 129, 59), MIConstants.IMAGE_COLOR_ORANGE);
colors.put(new Color(126, 132, 36), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(128, 93, 128), MIConstants.IMAGE_COLOR_PURPLE);
colors.put(new Color(197, 79, 51), MIConstants.IMAGE_COLOR_ORANGE);
colors.put(new Color(174, 201, 235), MIConstants.IMAGE_COLOR_BLUE);
colors.put(new Color(0, 117, 94), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(142, 114, 199), MIConstants.IMAGE_COLOR_PURPLE);
colors.put(new Color(249, 211, 190), MIConstants.IMAGE_COLOR_ORANGE);
colors.put(new Color(227, 172, 61), MIConstants.IMAGE_COLOR_YELLOW);
colors.put(new Color(222, 166, 129), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(88, 84, 82), MIConstants.IMAGE_COLOR_GRAY);
colors.put(new Color(245, 204, 35), MIConstants.IMAGE_COLOR_YELLOW);
colors.put(new Color(165, 110, 117), MIConstants.IMAGE_COLOR_PURPLE);
colors.put(new Color(174, 144, 65), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(64, 224, 208), MIConstants.IMAGE_COLOR_BLUE);
colors.put(new Color(108, 218, 231), MIConstants.IMAGE_COLOR_BLUE);
colors.put(new Color(54, 62, 29), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(173, 98, 66), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(227, 229, 177), MIConstants.IMAGE_COLOR_YELLOW);
```

```
colors.put(new Color(248, 228, 227), MIConstants.IMAGE_COLOR_RED);
colors.put(new Color(218, 192, 205), MIConstants.IMAGE_COLOR_PURPLE);
colors.put(new Color(244, 246, 236), MIConstants.IMAGE_COLOR_WHITE);
colors.put(new Color(193, 145, 86), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(102, 2, 60), MIConstants.IMAGE_COLOR_PURPLE);
colors.put(new Color(255, 111, 255), MIConstants.IMAGE_COLOR_PINK);
colors.put(new Color(18, 10, 143), MIConstants.IMAGE_COLOR_BLUE);
colors.put(new Color(212, 87, 78), MIConstants.IMAGE_COLOR_RED);
colors.put(new Color(56, 44, 56), MIConstants.IMAGE_COLOR_BLACK);
colors.put(new Color(42, 43, 65), MIConstants.IMAGE_COLOR_BLACK);
colors.put(new Color(82, 57, 54), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(204, 182, 155), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(253, 239, 211), MIConstants.IMAGE_COLOR_ORANGE);
colors.put(new Color(200, 8, 21), MIConstants.IMAGE_COLOR_RED);
colors.put(new Color(44, 87, 120), MIConstants.IMAGE_COLOR_BLUE);
colors.put(new Color(139, 125, 130), MIConstants.IMAGE_COLOR_PURPLE);
colors.put(new Color(72, 83, 26), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(255, 77, 0), MIConstants.IMAGE_COLOR_ORANGE);
colors.put(new Color(92, 64, 51), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(205, 205, 205), MIConstants.IMAGE_COLOR_GRAY);
colors.put(new Color(168, 85, 51), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(86, 73, 133), MIConstants.IMAGE_COLOR_PURPLE);
colors.put(new Color(95, 146, 40), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(77, 177, 200), MIConstants.IMAGE_COLOR_BLUE);
colors.put(new Color(149, 82, 100), MIConstants.IMAGE_COLOR_PURPLE);
colors.put(new Color(197, 143, 157), MIConstants.IMAGE_COLOR_PURPLE);
colors.put(new Color(46, 34, 73), MIConstants.IMAGE_COLOR_BLACK);
colors.put(new Color(238, 130, 238), MIConstants.IMAGE_COLOR_PINK);
colors.put(new Color(159, 95, 159), MIConstants.IMAGE_COLOR_PURPLE);
colors.put(new Color(247, 70, 138), MIConstants.IMAGE_COLOR_PINK);
colors.put(new Color(64, 130, 109), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(75, 95, 86), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(249, 228, 150), MIConstants.IMAGE_COLOR_YELLOW);
colors.put(new Color(151, 213, 179), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(227, 223, 217), MIConstants.IMAGE_COLOR_GRAY);
colors.put(new Color(255, 153, 128), MIConstants.IMAGE_COLOR_ORANGE);
colors.put(new Color(128, 55, 144), MIConstants.IMAGE_COLOR_PURPLE);
colors.put(new Color(68, 50, 64), MIConstants.IMAGE_COLOR_PURPLE);
colors.put(new Color(54, 56, 60), MIConstants.IMAGE_COLOR_GRAY);
colors.put(new Color(91, 110, 145), MIConstants.IMAGE_COLOR_BLUE);
```

```
colors.put(new Color(76, 78, 49), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(228, 226, 220), MIConstants.IMAGE_COLOR_GRAY);
colors.put(new Color(132, 145, 55), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(182, 236, 222), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(0, 110, 78), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(214, 202, 61), MIConstants.IMAGE_COLOR_YELLOW);
colors.put(new Color(242, 205, 187), MIConstants.IMAGE_COLOR_ORANGE);
colors.put(new Color(238, 179, 158), MIConstants.IMAGE_COLOR_ORANGE);
colors.put(new Color(253, 215, 216), MIConstants.IMAGE_COLOR_RED);
colors.put(new Color(76, 107, 136), MIConstants.IMAGE_COLOR_BLUE);
colors.put(new Color(142, 53, 55), MIConstants.IMAGE_COLOR_RED);
colors.put(new Color(92, 81, 47), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(229, 130, 58), MIConstants.IMAGE_COLOR_ORANGE);
colors.put(new Color(212, 207, 197), MIConstants.IMAGE_COLOR_GRAY);
colors.put(new Color(241, 145, 154), MIConstants.IMAGE_COLOR_RED);
colors.put(new Color(245, 222, 179), MIConstants.IMAGE_COLOR_ORANGE);
colors.put(new Color(223, 215, 189), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(210, 144, 98), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(239, 230, 230), MIConstants.IMAGE_COLOR_GRAY);
colors.put(new Color(255, 255, 255), MIConstants.IMAGE_COLOR_WHITE);
colors.put(new Color(215, 238, 228), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(231, 229, 232), MIConstants.IMAGE_COLOR_WHITE);
colors.put(new Color(248, 246, 216), MIConstants.IMAGE_COLOR_WHITE);
colors.put(new Color(218, 214, 204), MIConstants.IMAGE_COLOR_GRAY);
colors.put(new Color(245, 245, 245), MIConstants.IMAGE_COLOR_WHITE);
colors.put(new Color(122, 137, 184), MIConstants.IMAGE_COLOR_BLUE);
colors.put(new Color(227, 212, 116), MIConstants.IMAGE_COLOR_YELLOW);
colors.put(new Color(231, 228, 222), MIConstants.IMAGE_COLOR_GRAY);
colors.put(new Color(255, 51, 153), MIConstants.IMAGE_COLOR_PINK);
colors.put(new Color(253, 91, 120), MIConstants.IMAGE_COLOR_RED);
colors.put(new Color(190, 202, 96), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(83, 115, 111), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(223, 230, 207), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(105, 117, 92), MIConstants.IMAGE_COLOR_GREEN);
colors.put(new Color(70, 44, 119), MIConstants.IMAGE_COLOR_PURPLE);
colors.put(new Color(82, 44, 53), MIConstants.IMAGE_COLOR_BROWN);
colors.put(new Color(208, 195, 131), MIConstants.IMAGE_COLOR_YELLOW);
colors.put(new Color(249, 232, 226), MIConstants.IMAGE_COLOR_WHITE);
colors.put(new Color(201, 160, 220), MIConstants.IMAGE_COLOR_PURPLE);
colors.put(new Color(162, 158, 205), MIConstants.IMAGE_COLOR_BLUE);
```

```
            colors.put(new Color(251, 240, 115), MIConstants.IMAGE_COLOR_YELLOW);
            colors.put(new Color(48, 38, 33), MIConstants.IMAGE_COLOR_BLACK);
            colors.put(new Color(70, 54, 41), MIConstants.IMAGE_COLOR_BROWN);
            colors.put(new Color(98, 103, 70), MIConstants.IMAGE_COLOR_GREEN);
            colors.put(new Color(69, 64, 43), MIConstants.IMAGE_COLOR_BROWN);
            colors.put(new Color(43, 50, 48), MIConstants.IMAGE_COLOR_BLACK);
            colors.put(new Color(85, 69, 69), MIConstants.IMAGE_COLOR_BROWN);
            colors.put(new Color(117, 135, 110), MIConstants.IMAGE_COLOR_GREEN);
            colors.put(new Color(255, 255, 0), MIConstants.IMAGE_COLOR_YELLOW);
            colors.put(new Color(154, 205, 50), MIConstants.IMAGE_COLOR_GREEN);
            colors.put(new Color(115, 99, 62), MIConstants.IMAGE_COLOR_BROWN);
            colors.put(new Color(255, 174, 66), MIConstants.IMAGE_COLOR_ORANGE);
            colors.put(new Color(244, 159, 53), MIConstants.IMAGE_COLOR_ORANGE);
            colors.put(new Color(255, 197, 187), MIConstants.IMAGE_COLOR_RED);
            colors.put(new Color(130, 106, 33), MIConstants.IMAGE_COLOR_BROWN);
            colors.put(new Color(107, 90, 90), MIConstants.IMAGE_COLOR_GRAY);
            colors.put(new Color(178, 198, 177), MIConstants.IMAGE_COLOR_GREEN);
            colors.put(new Color(198, 114, 59), MIConstants.IMAGE_COLOR_ORANGE);
            colors.put(new Color(59, 60, 56), MIConstants.IMAGE_COLOR_BLACK);
            colors.put(new Color(129, 166, 170), MIConstants.IMAGE_COLOR_BLUE);
            colors.put(new Color(235, 194, 175), MIConstants.IMAGE_COLOR_RED);
            colors.put(new Color(222, 227, 227), MIConstants.IMAGE_COLOR_GRAY);
            colors.put(new Color(221, 194, 131), MIConstants.IMAGE_COLOR_YELLOW);
            colors.put(new Color(162, 149, 137), MIConstants.IMAGE_COLOR_BROWN);
            colors.put(new Color(23, 70, 46), MIConstants.IMAGE_COLOR_GREEN);
            colors.put(new Color(205, 213, 213), MIConstants.IMAGE_COLOR_GRAY);
        }

    }
```

Apêndice J – Classe *MIConstants*

```java
package tcc.tbfm.model.classesUtils;

public class MIConstants {

	// valor padrão
	public static final int DEFAULT_SEARCH_NUMBER_OF_RESULTS = 50;
	public static final float DEFAULT_SEARCH_SIMILARITY_VALUE = 70;
	public static final String DEFAULT_SEARCH_PATH = "Content";
	public static final String DEFAULT_INDEX_FOLDER_TXT = "\\metadataTXTIndex";
	public static final String DEFAULT_INDEX_FOLDER_XML = "\\metadataXMLIndex";
	public static final String DEFAULT_INDEX_FOLDER_IMAGE = "\\metadataImageIndex";
	public static final String DEFAULT_INDEX_TAG_FILE = "\\tagsListIndex.ser";
	public static final String DEFAULT_RESULT_FILE_NAME = "\\result";
	public static final String DEFAULT_RESULT_FILE_EXT = ".txt";
	public static final String IMAGE_NOT_SELECTED_INFO = "Nenhum";

	// cores
	public static final String IMAGE_COLOR_WHITE = "Branco";
	public static final String IMAGE_COLOR_YELLOW = "Amarelo";
	public static final String IMAGE_COLOR_BLUE = "Azul";
	public static final String IMAGE_COLOR_RED = "Vermelho";
	public static final String IMAGE_COLOR_BLACK = "Preto";
	public static final String IMAGE_COLOR_PURPLE = "Violeta";
	public static final String IMAGE_COLOR_ORANGE = "Laranja";
	public static final String IMAGE_COLOR_GRAY = "Cinza";
	public static final String IMAGE_COLOR_PINK = "Rosa";
	public static final String IMAGE_COLOR_GREEN = "Verde";
	public static final String IMAGE_COLOR_BROWN = "Marrom";

}
```

Apêndice K – Classe *MIKeys*

```java
package tcc.tbfm.model.classesUtils;

public class MIKeys {

	// chaves das propriedades das imagens
	public static final String IMAGE_WIDTH = "width";
	public static final String IMAGE_HEIGHT = "height";
	public static final String IMAGE_SIZEKB = "sizeKB";
	public static final String IMAGE_LAST_MODIFICATION = "lastModification";
	public static final String IMAGE_CREATION = "creation";
	public static final String IMAGE_X_RESOLUTION = "xResolution";
	public static final String IMAGE_Y_RESOLUTION = "yResolution";
	public static final String IMAGE_FORMAT = "format";
	public static final String IMAGE_COLOR_TYPE = "colorType";
	public static final String IMAGE_CAMERA = "camera";
	public static final String IMAGE_PREDOMINANT_COLOR = "predominantColor";

	// chaves para buscas em anotações
	public static final String SEARCH_ANNOT_TXT = "txtAnnotation";
	public static final String SEARCH_ANNOT_XML = "xmlAnnotation";
	public static final String SEARCH_IMAGE_DATA = "imageData";
	public static final String SEARCH_OF_POSSIBLES_PATHS = "searchOfPossiblesPaths";
	public static final String SEARCH_NUMBER_OF_RESULTS = "searchNumberOfResults";
	public static final String SEARCH_PATH_OF_RESULTS = "searchPathrOfResults";

	// chaves para tags
	public static final String SEARCH_PATH_VALUE = "searchPathValue";
	public static final String SEARCH_PATH_ONLY_SOURCE = "searchPathOnlySource";
	public static final String SEARCH_PATH_SYNONYSM = "searchPathSynonyms";
	public static final String SEARCH_PATH_SIMPLE = "searchPathSimple";
	public static final String SEARCH_PATH_SUBSTRING = "searchPathSubstring";
	public static final String SEARCH_PATH_SIMILARITY = "searchPathSimilarity";
	public static final String SEARCH_PATH_SIMILARITY_VALUE = "searchPathSimilarityValue";
```

```java
        // chaves para valor
        public static final String SEARCH_WORD_VALUE = "searchWordValue";
        public static final String SEARCH_WORD_ONLY_SOURCE = "searchWordOnlySource";
        public static final String SEARCH_WORD_SYNONYSM = "searchWordSynonyms";
        public static final String SEARCH_WORD_SIMPLE = "searchWordSimple";
        public static final String SEARCH_WORD_SUBSTRING = "searchWordSubstring";
        public static final String SEARCH_WORD_SIMILARITY = "searchWordSimilarity";
        public static final String SEARCH_WORD_SIMILARITY_VALUE = "searchWordSimilarityValue";

}
```

Apêndice L – Classe *HitsMI*

```java
package tcc.tbfm.model.objectsUtils;

import java.io.BufferedWriter;
import java.io.FileWriter;
import java.io.IOException;
import java.util.ArrayList;
import java.util.Calendar;
import java.util.Enumeration;
import java.util.Hashtable;

import tcc.tbfm.model.classesUtils.Hit;
import tcc.tbfm.model.classesUtils.MIConstants;
import tcc.tbfm.model.classesUtils.MIKeys;

public class HitsMI {

	private QueryMI query;
	private ArrayList<Hit> hitsSimpleTxt;
	private ArrayList<Hit> hitsSimpleXml;
	private ArrayList<Hit> hitsSubstringTxt;
	private ArrayList<Hit> hitsSubstringXml;
	private ArrayList<String> hitsPath;
	private ArrayList<Hit> hitsSimilarityTxt;
	private ArrayList<Hit> hitsSimilarityXml;
	private Hashtable<String, Hashtable<String, ArrayList<Hit>>> hitsSynonysm;
	private Hashtable<String, ArrayList<Hit>> hitsImage;
	private ArrayList<Hit> hitsMergedXml;
	private ArrayList<Hit> hitsMergedTxt;

	public HitsMI(QueryMI query) {
		this.hitsSimpleTxt = new ArrayList<>();
		this.hitsSimpleXml = new ArrayList<>();
		this.hitsSubstringTxt = new ArrayList<>();
		this.hitsSubstringXml = new ArrayList<>();
```

```
        this.hitsPath = new ArrayList<>();
        this.hitsSimilarityTxt = new ArrayList<>();
        this.hitsSimilarityXml = new ArrayList<>();
        this.hitsSynonysm = new Hashtable<>();
        this.hitsImage = new Hashtable<>();
        this.query = query.clone();
}

public ArrayList<String> getHitsPath() {
        return hitsPath;
}

public void addHitsPath(ArrayList<String> path) {
        hitsPath = path;
}

public void addHitsSimple(Hashtable<String, ArrayList<Hit>> simple) {
        if (simple.containsKey(MIKeys.SEARCH_ANNOT_TXT))
                hitsSimpleTxt = simple.get(MIKeys.SEARCH_ANNOT_TXT);

        if (simple.containsKey(MIKeys.SEARCH_ANNOT_XML))
                hitsSimpleXml = simple.get(MIKeys.SEARCH_ANNOT_XML);
}

public void addHitsSubstring(Hashtable<String, ArrayList<Hit>> substring) {
        if (substring.containsKey(MIKeys.SEARCH_ANNOT_TXT))
                hitsSubstringTxt = substring.get(MIKeys.SEARCH_ANNOT_TXT);

        if (substring.containsKey(MIKeys.SEARCH_ANNOT_XML))
                hitsSubstringXml = substring.get(MIKeys.SEARCH_ANNOT_XML);
}

public void addHitsSimilarity(Hashtable<String, ArrayList<Hit>> similarity) {
        if (similarity.containsKey(MIKeys.SEARCH_ANNOT_TXT))
                hitsSimilarityTxt = similarity.get(MIKeys.SEARCH_ANNOT_TXT);

        if (similarity.containsKey(MIKeys.SEARCH_ANNOT_XML))
                hitsSimilarityXml = similarity.get(MIKeys.SEARCH_ANNOT_XML);
}
```

```
public void addHitsSynonysm(
            Hashtable<String, Hashtable<String, ArrayList<Hit>>> synonysm) {
      hitsSynonysm = synonysm;
}

public void addHitsImage(Hashtable<String, ArrayList<Hit>> image) {
      hitsImage = image;
}

public void writeResultFile() {
      if (!query.isSearchOfPossiblesPaths()) {
            writeResultFileTotalResults();
            writeResultFileMergeResults();
      }
}

private void mergeResults() {

      ArrayList<Hit> hitsImgRelevances = getHitsWithRelevance(MIKeys.SEARCH_IMAGE_DATA);

      if (query.isSearchTxtAnnotation()) {
            ArrayList<Hit> hitsTxtRelevances = getHitsWithRelevance(MIKeys.SEARCH_ANNOT_TXT);
            hitsMergedTxt = new ArrayList<>();
            for (Hit hit : hitsTxtRelevances) {
                  if (hitsImgRelevances.contains(hit)) {
                        Hit valueImg = hitsImgRelevances.get(hitsImgRelevances
                                    .indexOf(hit));
                        hit.setScore(hit.getScore() + valueImg.getScore());
                  }
                  hitsMergedTxt.add(hit);
            }

            for (Hit hit : hitsImgRelevances)
                  if (!hitsTxtRelevances.contains(hit))
                        hitsMergedTxt.add(hit);

            for (Hit hit : hitsMergedTxt) {
                  hit.setScore(hit.getScore() / 2);
            }
```

```
        }

        if (query.isSearchXmlAnnotation()) {
                ArrayList<Hit> hitsXmlRelevances = getHitsWithRelevance(MIKeys.SEARCH_ANNOT_XML);
                hitsMergedXml = new ArrayList<>();
                for (Hit hit : hitsXmlRelevances) {
                        if (hitsImgRelevances.contains(hit)) {
                                Hit valueImg = hitsImgRelevances.get(hitsImgRelevances
                                                .indexOf(hit));
                                hit.setScore(hit.getScore() + valueImg.getScore());
                        }
                        hitsMergedXml.add(hit);
                }

                for (Hit hit : hitsImgRelevances)
                        if (!hitsXmlRelevances.contains(hit))
                                hitsMergedXml.add(hit);

                for (Hit hit : hitsMergedXml) {
                        hit.setScore(hit.getScore() / 2);
                }

        }
}

private ArrayList<Hit> getHitsWithRelevance(String metadataKind) {
        ArrayList<Hit> hitsRelevance = new ArrayList<>();
        // executando a média ponderada, considerando a relevância de cada busca
        if (metadataKind.equals(MIKeys.SEARCH_ANNOT_TXT)
                        || metadataKind.equals(MIKeys.SEARCH_ANNOT_XML)) {
                ArrayList<Hit> hitsSimilarity;
                ArrayList<Hit> hitsSimple;
                ArrayList<Hit> hitsSubstring;
                if (metadataKind.equals(MIKeys.SEARCH_ANNOT_TXT)) {
                        hitsSimilarity = hitsSimilarityTxt;
                        hitsSubstring = hitsSubstringTxt;
                        hitsSimple = hitsSimpleTxt;
                } else {
                        hitsSimilarity = hitsSimilarityXml;
                        hitsSubstring = hitsSubstringXml;
```

```
		hitsSimple = hitsSimpleXml;
	}

	int relevanceTotal = 0;

	if (query.isSearchWordSimilarity()) {
		relevanceTotal = +query
				.getRelevanceOfKey(MIKeys.SEARCH_WORD_SIMILARITY);
		for (Hit hit : hitsSimilarity) {
			if (hitsRelevance.contains(hit)) {
				Hit merg = hitsRelevance
						.get(hitsRelevance.indexOf(hit));
				merg.setScore(merg.getScore()
						+ (hit.getScore() * query
								.getRelevanceOfKey(MIKeys.SEARCH_WORD_SIMILARITY)));
			} else {
				hitsRelevance.add(hit);
			}
		}
	}

	if (query.isSearchWordSimple()) {
		relevanceTotal = +query
				.getRelevanceOfKey(MIKeys.SEARCH_WORD_SIMPLE);
		for (Hit hit : hitsSimple) {
			if (hitsRelevance.contains(hit)) {
				Hit merg = hitsRelevance
						.get(hitsRelevance.indexOf(hit));
				merg.setScore(merg.getScore()
						+ (hit.getScore() * query
								.getRelevanceOfKey(MIKeys.SEARCH_WORD_SIMPLE)));
			} else {
				hitsRelevance.add(hit);
			}
		}
	}

	if (query.isSearchWordSubstring()) {
		relevanceTotal = +query
				.getRelevanceOfKey(MIKeys.SEARCH_WORD_SUBSTRING);
```

```
                        for (Hit hit : hitsSubstring) {
                                if (hitsRelevance.contains(hit)) {
                                        Hit merg = hitsRelevance
                                                        .get(hitsRelevance.indexOf(hit));
                                        merg.setScore(merg.getScore()
                                                        + (hit.getScore() * query
                                                                        .getRelevanceOfKey(MIKeys.SEARCH_WORD_SUBSTRING)));
                                } else {
                                        hitsRelevance.add(hit);
                                }
                        }
                }

                if (query.isSearchWordSynonyms()) {
                        relevanceTotal = +query
                                        .getRelevanceOfKey(MIKeys.SEARCH_WORD_SYNONYSM);
                        Enumeration<String> keys = hitsSynonysm.keys();
                        while (keys.hasMoreElements()) {
                                String synonysm = keys.nextElement();
                                for (Hit hit : hitsSynonysm.get(synonysm).get(metadataKind)) {
                                        if (hitsRelevance.contains(hit)) {
                                                Hit merg = hitsRelevance.get(hitsRelevance
                                                                .indexOf(hit));
                                                merg.setScore(merg.getScore()
                                                                + (hit.getScore() * query

.getRelevanceOfKey(MIKeys.SEARCH_WORD_SYNONYSM)));
                                        } else {
                                                hitsRelevance.add(hit);
                                        }
                                }
                        }
                }

                for (Hit hit : hitsRelevance) {
                        hit.setScore(hit.getScore() / relevanceTotal);
                }

        } else if (metadataKind.equals(MIKeys.SEARCH_IMAGE_DATA)) {
                // executando a média ponderada, considerando a relevância de cada
```

```
                // característica
                Enumeration<String> keys = hitsImage.keys();
                int relevanceTotal = 0;
                while (keys.hasMoreElements()) {
                        String key = keys.nextElement();
                        relevanceTotal += query.getRelevanceOfKey(key);
                        for (Hit hit : hitsImage.get(key)) {
                                if (hitsRelevance.contains(hit)) {
                                        Hit merg = hitsRelevance
                                                        .get(hitsRelevance.indexOf(hit));
                                        merg.setScore(merg.getScore()
                                                        + (hit.getScore() * query
                                                                        .getRelevanceOfKey(key)));
                                } else {
                                        hitsRelevance.add(hit);
                                }
                        }
                }
                for (Hit hit : hitsRelevance) {
                        hit.setScore(hit.getScore() / relevanceTotal);
                }
        }
        return hitsRelevance;
}

private void writeResultFileMergeResults() {
        FileWriter fileWriter;
        BufferedWriter out;
        try {
                fileWriter = new FileWriter(query.pathOfResults()
                                + MIConstants.DEFAULT_RESULT_FILE_NAME
                                + Calendar.getInstance().getTimeInMillis() + "Merge"
                                + MIConstants.DEFAULT_RESULT_FILE_EXT);
                out = new BufferedWriter(fileWriter);
                if (query.isSearchImageData()
                                && (query.isSearchTxtAnnotation() || query
                                                .isSearchXmlAnnotation())) {
                        mergeResults();
                        writeLine(
                                        out,
```

```
                        "-------------------------------------------------------------------------\n");
                writeLine(out, "\nExecutando busca");
                writeLine(out, "-----------\n");
                if (query.isSearchTxtAnnotation()) {
                        writeLine(out,
                                        "\n>>Em Anotações TXT e em Metadados da Imagem\n");
                        for (Hit hit : hitsMergedTxt) {
                                writeLine(out,
                                                hit.getContent() + " Score: " + hit.getScore());
                        }
                }
                if (query.isSearchXmlAnnotation()) {
                        writeLine(out,
                                        "\n>>Em Anotações XML e em Metadados da Imagem\n");
                        for (Hit hit : hitsMergedXml) {
                                writeLine(out,
                                                hit.getContent() + " Score: " + hit.getScore());
                        }
                }

        } else {
                writeLine(out,
                                "Sem dados para mesclar, configurada apenas uma busca!\n");
        }

        out.close();

    } catch (IOException e) {
            e.printStackTrace();
    }
}

private void writeResultFileTotalResults() {
    FileWriter fileWriter;
    BufferedWriter out;
    try {
            fileWriter = new FileWriter(query.pathOfResults()
                            + MIConstants.DEFAULT_RESULT_FILE_NAME
                            + Calendar.getInstance().getTimeInMillis() + "Total"
                            + MIConstants.DEFAULT_RESULT_FILE_EXT);
```

```
out = new BufferedWriter(fileWriter);
writeLine(
            out,
            "--------------------------------------------------------------------------------\n");
writeLine(out,
            "\nExecutando busca nos índices dos metadados externos à imagem por: "
                        + query.getWordValueToSearch());
writeLine(out, "-----------\n");
// Busca Simples
if (query.isSearchWordOnlySource()) {

      writeLine(out,
                  "\n--------Busca apenas pela palavra fornecida: "
                              + query.getWordValueToSearch() + "\n");

      if (query.isSearchTxtAnnotation()) {
            writeLine(out, "\n>>Em metadados TXT\n");
            if (hitsSimpleTxt.size() > 0) {
                  writeLine(out, "\nFound " + hitsSimpleTxt.size()
                              + " hits.\n");
                  int i = 1;
                  for (Hit hit : hitsSimpleTxt) {
                        writeLine(out, i + "-" + hit.getContent()
                                    + " Score: " + hit.getScore());
                        i++;
                  }
            } else
                  writeLine(out, "\nValor fornecido não encontrado!\n");
      }

      if (query.isSearchXmlAnnotation()) {
            writeLine(
                        out,
                        "\n>>Em metadados XML (com chave: "
                                    + query.getPathValueToSearch() + ").\n");
            if (hitsSimpleXml.size() > 0) {
                  writeLine(out, "\nFound " + hitsSimpleXml.size()
                              + " hits.\n");
                  int i = 1;
                  for (Hit hit : hitsSimpleXml) {
```

```
                        writeLine(out, i + "-" + hit.getContent()
                                + " Score: " + hit.getScore());
                        i++;
                    }
                } else
                    out.write("\nValor fornecido não encontrado!\n");
                out.newLine();
            }

        } else {
            if (!query.isSearchWordSimilarity()
                    && !query.isSearchWordSynonyms()
                    && !query.isSearchWordSubstring()
                    && !query.isSearchWordSimple())
                writeLine(out,
                        "Não foi configurado busca nos metadados externos à imagem!!");
            else {
                // Busca Simples
                if (query.isSearchWordSimple()) {
                    writeLine(out, "\n--------Busca pelo valor fornecido: "
                            + query.getWordValueToSearch() + ".\n");

                    if (query.isSearchTxtAnnotation()) {
                        writeLine(out, "\n>>Em metadados TXT\n");
                        if (hitsSimpleTxt.size() > 0) {
                            writeLine(out,
                                    "\nFound " + hitsSimpleTxt.size()
                                            + " hits.\n");
                            int i = 1;
                            for (Hit hit : hitsSimpleTxt) {
                                writeLine(out, i + "-" + hit.getContent()
                                        + " Score: " + hit.getScore());
                                i++;
                            }
                        } else
                            writeLine(out,
                                    "\nValor fornecido não encontrado!\n");
                    }

                    if (query.isSearchXmlAnnotation()) {
```

```
                writeLine(out, "\n>>Em metadados XML (com chave: "
                            + query.getPathValueToSearch() + ").\n");
                if (hitsSimpleXml.size() > 0) {
                      writeLine(out,
                                  "\nFound " + hitsSimpleXml.size()
                                              + " hits.\n");
                      int i = 1;
                      for (Hit hit : hitsSimpleXml) {
                            writeLine(out, i + "-" + hit.getContent()
                                        + " Score: " + hit.getScore());
                            i++;
                      }
                } else
                      writeLine(out,
                                  "\nValor fornecido não encontrado!\n");
          }

    }
    // Busca por Sinônimos
    if (query.isSearchWordSynonyms()) {
          writeLine(out,
                      "\n--------Busca por sinônimos do valor fornecido: "
                                  + query.getWordValueToSearch() + ".\n");
          if (hitsSynonysm.isEmpty())
                writeLine(out,
                            "\nNão foram encontrados sinônimos.\n");
          else {
                Enumeration<String> synonysms = hitsSynonysm.keys();
                while (synonysms.hasMoreElements()) {
                      String synon = (String) synonysms.nextElement();
                      writeLine(out, "\nSinônimo: " + synon + "\n");

                      ArrayList<Hit> synonTxt = hitsSynonysm.get(
                                  synon).get(MIKeys.SEARCH_ANNOT_TXT);
                      if (query.isSearchTxtAnnotation()) {
                            writeLine(out, "\n>>Em metadados TXT\n");
                            if (synonTxt != null) {
                                  writeLine(out,
                                              "\nFound " + synonTxt.size()
                                                          + " hits.\n");
```

```
                int i = 1;
                for (Hit hit : synonTxt) {
                      writeLine(
                                  out,
                                  i + "-" + hit.getContent()
                                              + " Score: "
                                              + hit.getScore());
                      i++;
                }

          } else
                writeLine(out,
                            "\nSinônimo não encontrado.\n");
    }

    ArrayList<Hit> synonXml = hitsSynonysm.get(
                synon).get(MIKeys.SEARCH_ANNOT_XML);

    if (query.isSearchXmlAnnotation()) {
          writeLine(
                      out,
                      "\n>>Em metadados XML (com chave: "
                                  + query.getPathValueToSearch()
                                  + ").\n");
          if (synonXml != null) {
                writeLine(out,
                            "\nFound " + synonXml.size()
                                        + " hits.\n");
                int i = 1;
                for (Hit hit : synonXml) {
                      writeLine(
                                  out,
                                  i + "-" + hit.getContent()
                                              + " Score: "
                                              + hit.getScore());
                      i++;
                }
          } else
                writeLine(out,
                            "\nSinônimo não encontrado.\n");
```

```
                        }
                    }
                }
            }
            // Busca por Substring
            if (query.isSearchWordSubstring()) {
                writeLine(out,
                        "\n--------Busca utilizando o valor fornecido '"
                                + query.getWordValueToSearch()
                                + "' como substring.\n");
                if (query.isSearchTxtAnnotation()) {
                    writeLine(out, "\n>>Em metadados TXT\n");
                    if (hitsSubstringTxt.size() > 0) {
                        writeLine(out,
                                "\nFound " + hitsSubstringTxt.size()
                                        + " hits.\n");
                        int i = 1;
                        for (Hit hit : hitsSubstringTxt) {
                            writeLine(out, i + "-" + hit.getContent()
                                    + " Score: " + hit.getScore());
                            i++;
                        }
                    } else
                        writeLine(out,
                                "\nValor fornecido não encontrado!\n");
                }

                if (query.isSearchXmlAnnotation()) {
                    writeLine(out, "\n>>Em metadados XML (com chave: "
                            + query.getPathValueToSearch() + ").\n");
                    if (hitsSubstringXml.size() > 0) {
                        writeLine(out,
                                "\nFound " + hitsSubstringXml.size()
                                        + " hits.\n");
                        int i = 1;
                        for (Hit hit : hitsSubstringXml) {
                            writeLine(out, i + "-" + hit.getContent()
                                    + " Score: " + hit.getScore());
                            i++;
                        }
```

```
                } else
                        writeLine(out,
                                        "\nValor fornecido não encontrado!\n");
        }

}
// Busca por Similaridade
if (query.isSearchWordSimilarity()) {
        writeLine(out,
                        "\n--------Busca utilizando o valor fornecido '"
                                        + query.getWordValueToSearch()
                                        + "' por similaridade.\n");

        if (query.isSearchTxtAnnotation()) {
                writeLine(out, "\n>>Em metadados TXT\n");
                if (hitsSimilarityTxt.size() > 0) {
                        writeLine(out,
                                        "\nFound " + hitsSimilarityTxt.size()
                                                        + " hits.\n");
                        int i = 1;
                        for (Hit hit : hitsSimilarityTxt) {
                                writeLine(out, i + "-" + hit.getContent()
                                                + " Score: " + hit.getScore());
                                i++;
                        }
                } else
                        writeLine(out,
                                        "\nValor fornecido não encontrado!\n");
        }

        if (query.isSearchXmlAnnotation()) {
                writeLine(out, "\n>>Em metadados XML (com chave: "
                                + query.getPathValueToSearch() + ").\n");
                if (hitsSimilarityXml.size() > 0) {
                        writeLine(out,
                                        "\nFound " + hitsSimilarityXml.size()
                                                        + " hits.\n");
                        int i = 1;
                        for (Hit hit : hitsSimilarityXml) {
                                writeLine(out, i + "-" + hit.getContent()
```

```
                                                + " Score: " + hit.getScore());
                                        i++;
                                }
                        } else
                                writeLine(out,
                                                "\nValor fornecido não encontrado!\n");
                }

        }
    }
}
writeLine(
        out,
        "-------------------------------------------------------------------------\n\n");

// Busca por dados da imagem
if (query.isSearchImageData()) {
    writeLine(
                out,
                "-------------------------------------------------------------------------\n");
    writeLine(out,
                "\nExecutando busca nos índices dos metadados internos à imagem.\n");
    writeLine(out, "-----------\n");

    if (hitsImage.size() > 0) {
        Enumeration<String> imgKeys = hitsImage.keys();
        while (imgKeys.hasMoreElements()) {
            String imgKey = (String) imgKeys.nextElement();
            String value = query.getValueOfKeyOfImgHash(imgKey);
            ArrayList<Hit> imgHits = hitsImage.get(imgKey);
            writeLine(out, "");
            writeLine(out, ">>Característica da imagem '" + imgKey
                        + "' com valor " + value);
            writeLine(out, "\nFound " + imgHits.size() + " hits.\n");
            int i = 1;
            for (Hit hit : imgHits) {
                writeLine(out, i + "-" + hit.getContent()
                            + " Score: " + hit.getScore());
                i++;
            }
```

```
                                }
                        } else
                                writeLine(out, "\nNão foi configurada a busca!\n");

                }
                writeLine(
                                out,
                                "--------------------------------------------------------------------------------\n");
                out.close();

        } catch (IOException e) {
                e.printStackTrace();
        }
    }

    private void writeLine(BufferedWriter out, String line) {
        try {
                out.write(line);
                out.newLine();
        } catch (IOException e) {
                e.printStackTrace();
        }
    }

}
```

Apêndice M – Classe *ImageData*

```
package tcc.tbfm.model.objectsUtils;

import java.io.File;
import java.io.IOException;
import java.util.Calendar;
import java.util.Hashtable;

import org.apache.sanselan.ImageInfo;
import org.apache.sanselan.ImageReadException;
import org.apache.sanselan.Sanselan;
import org.apache.sanselan.common.IImageMetadata;
import org.apache.sanselan.formats.jpeg.JpegImageMetadata;
import org.apache.sanselan.formats.tiff.TiffField;
import org.apache.sanselan.formats.tiff.constants.TiffConstants;

import tcc.tbfm.model.classesUtils.HelperFunctions;
import tcc.tbfm.model.classesUtils.MIKeys;

public class ImageData {

	private boolean safeInformations;

	private int sizeKB;
	private int width;
	private int height;
	private int xResolution; // by exif
	private int yResolution; // by exif

	private String predominantColor;
	private Hashtable<String, Integer> percentOfColors;
	private String format;
	private String colorType;
	private String camera; // by exif
```

```
private Calendar lastModification;
private Calendar creation; // by exif

public ImageData(File imageFile) {

	initData();

	sizeKB = ((int) imageFile.length() / 1024);
	lastModification = HelperFunctions.getDateFromMillis(imageFile
			.lastModified());

	try {

		getInformationFromImageInfo(imageFile);

		getColorInformationFromFile(imageFile);

		getInformationFromMetadata(imageFile);

		safeInformations = true;

	} catch (ImageReadException e1) {
		safeInformations = false;
	} catch (IOException e1) {
		safeInformations = false;
	} catch (Exception e) {
		safeInformations = false;
	}

}

private void getInformationFromMetadata(File imageFile) {

	try {

		IImageMetadata sanselanMetadata = Sanselan.getMetadata(imageFile);
		JpegImageMetadata jpegMetadata = (JpegImageMetadata) sanselanMetadata;

		if (jpegMetadata != null) {
```

```
TiffField tiffField;

tiffField = jpegMetadata
            .findEXIFValue(TiffConstants.EXIF_TAG_CREATE_DATE);
if (tiffField != null) {
      Object fieldDate = tiffField.getValue();
      String dateTime = fieldDate.toString();
      creation = HelperFunctions.getDateFromStringDate(dateTime
                  .trim());
}

tiffField = jpegMetadata
            .findEXIFValue(TiffConstants.EXIF_TAG_MAKE);
if (tiffField != null) {
      Object fieldCamera = tiffField.getValue();
      camera = fieldCamera.toString();
}

tiffField = jpegMetadata
            .findEXIFValue(TiffConstants.EXIF_TAG_XRESOLUTION);
if (tiffField != null) {
      Object fieldXResolution = tiffField.getValue();
      try {
            xResolution = Integer.parseInt(fieldXResolution
                        .toString());
      } catch (NumberFormatException e) {
            xResolution = 0;
      }
}

tiffField = jpegMetadata
            .findEXIFValue(TiffConstants.EXIF_TAG_YRESOLUTION);
if (tiffField != null) {
      Object fieldYResolution = tiffField.getValue();
      try {
            yResolution = Integer.parseInt(fieldYResolution
                        .toString());
      } catch (NumberFormatException e) {
            yResolution = 0;
      }
```

```
                    }
                }

        } catch (ImageReadException e) {
        } catch (IOException e) {
        }
}

private void getInformationFromImageInfo(File imageFile)
            throws ImageReadException, IOException {
        ImageInfo image_info = Sanselan.getImageInfo(imageFile);
        height = image_info.getHeight();
        width = image_info.getWidth();
        format = image_info.getFormat().name;
        colorType = image_info.getColorTypeDescription();
}

public boolean IsSafeInformations() {
        return safeInformations;
}

private void getColorInformationFromFile(File imageFile) throws Exception {
        br.com.xptech.util.ImageInfo ii = new br.com.xptech.util.ImageInfo();
        percentOfColors = ii.colorContents(imageFile.getAbsolutePath());
        predominantColor = HelperFunctions.maxValue(percentOfColors);
}

private void initData() {

        safeInformations = false;

        width = 0;
        height = 0;
        sizeKB = 0;
        lastModification = null;
        format = "";
        colorType = "";

        creation = null;
        xResolution = 0;
```

```
        yResolution = 0;
        camera = "";

        predominantColor = "";
        percentOfColors = null;

    }

    public Hashtable<String, String> getStringHashTable() {
        Hashtable<String, String> hashTableImageData = new Hashtable<String, String>();
        if (!predominantColor.isEmpty())
            hashTableImageData.put(MIKeys.IMAGE_PREDOMINANT_COLOR,
                    predominantColor);
        if (!colorType.isEmpty())
            hashTableImageData.put(MIKeys.IMAGE_COLOR_TYPE, colorType);
        if (!format.isEmpty())
            hashTableImageData.put(MIKeys.IMAGE_FORMAT, format);
        if (!camera.isEmpty())
            hashTableImageData.put(MIKeys.IMAGE_CAMERA, camera);
        return hashTableImageData;
    }

    public Hashtable<String, Integer> getIntHashTable() {
        Hashtable<String, Integer> hashTableImageData = new Hashtable<String, Integer>();
        if (!percentOfColors.isEmpty())
            hashTableImageData = percentOfColors;
        if (width != 0)
            hashTableImageData.put(MIKeys.IMAGE_WIDTH, width);
        if (height != 0)
            hashTableImageData.put(MIKeys.IMAGE_HEIGHT, height);
        if (sizeKB != 0)
            hashTableImageData.put(MIKeys.IMAGE_SIZEKB, sizeKB);
        if (xResolution != 0)
            hashTableImageData.put(MIKeys.IMAGE_X_RESOLUTION, xResolution);
        if (yResolution != 0)
            hashTableImageData.put(MIKeys.IMAGE_Y_RESOLUTION, yResolution);
        return hashTableImageData;
    }

    public Hashtable<String, Calendar> getDateHashTable() {
```

```
            Hashtable<String, Calendar> hashTableImageData = new Hashtable<String, Calendar>();
            if (lastModification != null)
                  hashTableImageData.put(MIKeys.IMAGE_LAST_MODIFICATION,
                              lastModification);
            if (creation != null)
                  hashTableImageData.put(MIKeys.IMAGE_CREATION, creation);
            return hashTableImageData;
      }

}
```

Apêndice N – Classe *LuceneIndex*

```
package tcc.tbfm.model.objectsUtils;

import java.io.File;
import java.io.FileReader;
import java.io.IOException;
import java.util.ArrayList;
import java.util.Calendar;
import java.util.Enumeration;
import java.util.Hashtable;
import java.util.List;

import org.apache.lucene.analysis.Analyzer;
import org.apache.lucene.analysis.standard.StandardAnalyzer;
import org.apache.lucene.document.DateTools;
import org.apache.lucene.document.Document;
import org.apache.lucene.document.Field;
import org.apache.lucene.document.NumericField;
import org.apache.lucene.index.CorruptIndexException;
import org.apache.lucene.index.IndexWriter;
import org.apache.lucene.index.IndexWriterConfig;
import org.apache.lucene.store.Directory;
import org.apache.lucene.store.FSDirectory;
import org.apache.lucene.store.LockObtainFailedException;
import org.apache.lucene.util.Version;
import org.jdom.Element;
import org.jdom.JDOMException;
import org.jdom.input.SAXBuilder;

import tcc.tbfm.model.classesUtils.HelperFunctions;
import tcc.tbfm.model.classesUtils.MIConstants;

public class LuceneIndex {

	private ArrayList<String> setPaths = new ArrayList<String>();
```

```
private boolean isIndexInit;
private boolean isFollow;
private IndexWriter writerAnnotation;
private IndexWriter writerXmlAnnotation;
private IndexWriter writerImageData;
private Analyzer analyzerAnnotation;
private Analyzer analyzerXmlAnnotation;
private Analyzer analyzerImageData;
private Directory indexDirectoryAnnotation;
private Directory indexDirectoryXmlAnnotation;
private Directory indexDirectoryImageData;
private File imageDirectory;
private File annotationDirectory;
private File indexDirectory;

public LuceneIndex(MetadataIngesterInfo info) {
	setMetadataInfo(info);
}

public void setMetadataInfo(MetadataIngesterInfo info) {
	this.isFollow = info.isFollow();
	// lê os diretórios dos arquivos
	imageDirectory = info.getFolderImage();
	annotationDirectory = info.getFolderMetadata();
	indexDirectory = info.getFolderIndex();
	// lê o dir configurado para o índice
	File dirAnnotation = new File(info.getFolderIndex().toString()
			+ MIConstants.DEFAULT_INDEX_FOLDER_TXT);
	if (!dirAnnotation.exists())
		dirAnnotation.mkdirs();
	File dirXmlAnnotation = new File(info.getFolderIndex().toString()
			+ MIConstants.DEFAULT_INDEX_FOLDER_XML);
	if (!dirXmlAnnotation.exists())
		dirXmlAnnotation.mkdirs();
	File dirImages = new File(info.getFolderIndex().toString()
			+ MIConstants.DEFAULT_INDEX_FOLDER_IMAGE);
	if (!dirImages.exists())
		dirImages.mkdirs();
	// Diretório virtual para o índice
	try {
```

```
                indexDirectoryAnnotation = FSDirectory.open(dirAnnotation);
                indexDirectoryXmlAnnotation = FSDirectory.open(dirXmlAnnotation);
                indexDirectoryImageData = FSDirectory.open(dirImages);
        } catch (IOException e) {
                e.printStackTrace();
        }
}

// --------------Inicialização---------------
public void configureIndexer() {
        if (!isIndexInit) {

                // Configura o analyzer para extrair os dados para o indice
                analyzerAnnotation = new StandardAnalyzer(Version.LUCENE_33);
                analyzerXmlAnnotation = new StandardAnalyzer(Version.LUCENE_33);
                analyzerImageData = new StandardAnalyzer(Version.LUCENE_33);

                // aplica as configurações padrão
                IndexWriterConfig confAnnotation = new IndexWriterConfig(
                                Version.LUCENE_33, analyzerAnnotation);
                IndexWriterConfig confXmlAnnotation = new IndexWriterConfig(
                                Version.LUCENE_33, analyzerXmlAnnotation);
                IndexWriterConfig confImageData = new IndexWriterConfig(
                                Version.LUCENE_33, analyzerImageData);

                // Cria os arquivos com tamanho ilimitado.
                try {
                        writerAnnotation = new IndexWriter(indexDirectoryAnnotation,
                                        confAnnotation);
                        writerXmlAnnotation = new IndexWriter(
                                        indexDirectoryXmlAnnotation, confXmlAnnotation);
                        writerImageData = new IndexWriter(indexDirectoryImageData,
                                        confImageData);
                } catch (CorruptIndexException e) {
                        e.printStackTrace();
                } catch (LockObtainFailedException e) {
                        e.printStackTrace();
                } catch (IOException e) {
                        e.printStackTrace();
                }
```

```
		}
		isIndexInit = true;
	}

	public void indexIt() {
		// TODO: garantir que só serão indexadas as informações completas, ou
		// seja, a imagem e o seu respectivo
		// arquivo de anotação. Não devem ser indexadas informações sem um ou
		// outro.
		if (imageDirectory.exists()) {
			if (annotationDirectory.exists()) {
				cleanIndexDirectory();
				configureIndexer();
				for (File f : imageDirectory.listFiles()) {
					parse(f, true);
				}
				for (File f : annotationDirectory.listFiles()) {
					parse(f, false);
				}
				closeWriters();
				persistTagList();
			} else {
				System.out.println("Diretório "
						+ annotationDirectory.toString() + " não existe!");
			}
		} else {
			System.out.println("Diretório " + imageDirectory.toString()
					+ " não existe!");
		}
		// TODO: gerar um relatório/log com as informações sobre os arquivos que
		// não foram indexados
	}

	private void cleanIndexDirectory() {
		try {
			HelperFunctions.cleanDirectory(indexDirectoryAnnotation);
			HelperFunctions.cleanDirectory(indexDirectoryXmlAnnotation);
			HelperFunctions.cleanDirectory(indexDirectoryImageData);
		} catch (IOException e) {
```

```
                e.printStackTrace();
        }
}

private void parse(File fl, boolean IsImage) {
        // é um diretório
        if (fl.listFiles() != null) {
                // acessa cada item retornado
                for (File f : fl.listFiles()) {
                        // é um diretório e vamos verificar todas as subpastas
                        if (f.isDirectory() && isFollow) {
                                parse(f, IsImage);
                                // é um arquivo
                        } else if (!f.isDirectory()) {
                                handleFile(f, IsImage);
                        }
                }
                // é um arquivo
        } else {
                handleFile(fl, IsImage);
        }
}

private void handleFile(File fl, boolean IsImage) {
        try {
                if (IsImage) {
                        // adiciona os dados das imagens aos índices
                        appendImageData(fl.toString());
                } else {
                        // adiciona o arquivo de anotação por completo no índice
                        appendAnnotation(fl.toString(), new FileReader(fl));
                        // adiciona o xml válido do arquivo de anotação no índice
                        appendXMLAnnotation(fl.toString(), fileToHashTable(fl));
                }
        } catch (IOException e) {
                System.out.println("Erro: " + e);
                e.printStackTrace();
        } catch (Exception e) {
                System.out.println("Erro: " + e);
                e.printStackTrace();
```

```
	}
}

private Hashtable<String, String> fileToHashTable(File fl) {
	Hashtable<String, String> annotation = new Hashtable<String, String>();
	SAXBuilder sb = new SAXBuilder();
	try {
		HelperFunctions.encodingXML(fl, "ISO-8859-1");
		org.jdom.Document d = sb.build(fl);
		Element root = d.getRootElement();
		addAnnotation(annotation, root, root.getName());
	} catch (JDOMException e) {
		e.printStackTrace();
	} catch (IOException e) {
		System.out.println(e + fl.toString());
	}
	return annotation;
}

private void addAnnotation(Hashtable<String, String> an, Element n,
		String path) {
	if (n.getChildren().size() > 0) {
		@SuppressWarnings("unchecked")
		List<Element> kids = n.getChildren();
		for (Element node : kids) {
			addAnnotation(an, node, path + "/" + node.getName());
		}
	}
	if (!n.isRootElement()) {
		if (!setPaths.contains(path)) {
			setPaths.add(path);
		}
		an.put(path, n.getValue());
	}
}

private void appendAnnotation(String fileName, FileReader file)
		throws Exception, IOException {
	if (isIndexInit) {
		Document docAnnotations = new Document();
```

```
                docAnnotations.add(new Field("Path", fileName, Field.Store.YES,
                            Field.Index.ANALYZED));
                docAnnotations
                            .add(new Field(MIConstants.DEFAULT_SEARCH_PATH, file));
                writerAnnotation.addDocument(docAnnotations);
        }
}

private void appendXMLAnnotation(String fileName,
                Hashtable<String, String> file) throws Exception, IOException {
        if (isIndexInit) {
                Document docXmlAnnotations = new Document();
                docXmlAnnotations.add(new Field("Path", fileName, Field.Store.YES,
                            Field.Index.ANALYZED));
                Enumeration<String> e = file.keys();
                while (e.hasMoreElements()) {
                        String key = e.nextElement();
                        docXmlAnnotations.add(new Field(key, file.get(key),
                                    Field.Store.YES, Field.Index.ANALYZED));
                }
                writerXmlAnnotation.addDocument(docXmlAnnotations);
        }
}

private void appendImageData(String fileName) throws Exception, IOException {
        if (isIndexInit) {
                ImageData imageData = new ImageData(new File(fileName));
                if (imageData.IsSafeInformations()) {
                        Document docImageData = new Document();
                        docImageData.add(new Field("Path", fileName, Field.Store.YES,
                                    Field.Index.ANALYZED));

                        Hashtable<String, String> imageStringInfo = imageData
                                    .getStringHashTable();
                        Enumeration<String> stringInfo = imageStringInfo.keys();
                        while (stringInfo.hasMoreElements()) {
                                String key = stringInfo.nextElement();
                                docImageData.add(new Field(key, imageStringInfo.get(key),
                                            Field.Store.YES, Field.Index.ANALYZED));
                        }
```

```
                        Hashtable<String, Calendar> imageDateInfo = imageData
                                    .getDateHashTable();
                        Enumeration<String> dateInfo = imageDateInfo.keys();
                        while (dateInfo.hasMoreElements()) {
                              String key = dateInfo.nextElement();
                              docImageData.add(new Field(key, DateTools.dateToString(
                                          imageDateInfo.get(key).getTime(),
                                          DateTools.Resolution.SECOND), Field.Store.YES,
                                          Field.Index.NOT_ANALYZED));
                        }

                        Hashtable<String, Integer> imageIntInfo = imageData
                                    .getIntHashTable();
                        Enumeration<String> intInfo = imageIntInfo.keys();
                        while (intInfo.hasMoreElements()) {
                              String key = intInfo.nextElement();
                              NumericField field = new NumericField(key);
                              field.setIntValue(imageIntInfo.get(key));
                              docImageData.add(field);
                        }

                        writerImageData.addDocument(docImageData);
                  }
            }
      }

      private void closeWriters() {
            try {
                  writerAnnotation.prepareCommit();
                  writerAnnotation.commit();
                  writerAnnotation.close();
                  writerXmlAnnotation.prepareCommit();
                  writerXmlAnnotation.commit();
                  writerXmlAnnotation.close();
                  writerImageData.prepareCommit();
                  writerImageData.commit();
                  writerImageData.close();
            } catch (CorruptIndexException e) {
                  System.out.println("Nao foi possivel fechar IndexWriter. " + e);
```

```
			} catch (IOException e) {
				System.out.println("Nao foi possivel fechar IndexWriter. " + e);
			}
		}

		private void persistTagList() {
			try {
				HelperFunctions.persistObject(setPaths, indexDirectory.getPath()
						+ MIConstants.DEFAULT_INDEX_FOLDER_XML
						+ MIConstants.DEFAULT_INDEX_TAG_FILE);
			} catch (IOException e) {
				e.printStackTrace();
			}
		}

}
```

Apêndice O – Classe *LuceneSearch*

```
package tcc.tbfm.model.objectsUtils;

import java.io.File;
import java.io.IOException;
import java.util.ArrayList;
import java.util.Calendar;
import java.util.Enumeration;
import java.util.Hashtable;

import org.apache.lucene.analysis.StopAnalyzer;
import org.apache.lucene.document.DateTools;
import org.apache.lucene.document.DateTools.Resolution;
import org.apache.lucene.document.Document;
import org.apache.lucene.index.CorruptIndexException;
import org.apache.lucene.index.Term;
import org.apache.lucene.queryParser.QueryParser;
import org.apache.lucene.search.FuzzyQuery;
import org.apache.lucene.search.IndexSearcher;
import org.apache.lucene.search.MatchAllDocsQuery;
import org.apache.lucene.search.NumericRangeQuery;
import org.apache.lucene.search.Query;
import org.apache.lucene.search.ScoreDoc;
import org.apache.lucene.search.TermQuery;
import org.apache.lucene.search.TermRangeFilter;
import org.apache.lucene.search.TopDocs;
import org.apache.lucene.search.TopScoreDocCollector;
import org.apache.lucene.search.WildcardQuery;
import org.apache.lucene.store.Directory;
import org.apache.lucene.store.FSDirectory;
import org.apache.lucene.util.Version;

import tcc.tbfm.model.classesUtils.HelperFunctions;
import tcc.tbfm.model.classesUtils.Hit;
import tcc.tbfm.model.classesUtils.MIConstants;
```

```
import tcc.tbfm.model.classesUtils.MIKeys;

public class LuceneSearch {

	private boolean isSearchInit = false;
	private IndexSearcher searcherAnnotation;
	private Directory indexDirectoryAnnotation;
	private Directory indexDirectoryXmlAnnotation;
	private Directory indexDirectoryImageData;

	public LuceneSearch(MetadataIngesterInfo info) {
		setMetadataInfo(info);
	}

	public void setMetadataInfo(MetadataIngesterInfo info) {
		// lê o dir configurado para o indice
		String dirAnnotation = info.getFolderIndex().toString()
				+ MIConstants.DEFAULT_INDEX_FOLDER_TXT;
		String dirXmlAnnotation = info.getFolderIndex().toString()
				+ MIConstants.DEFAULT_INDEX_FOLDER_XML;
		String dirImagesData = info.getFolderIndex().toString()
				+ MIConstants.DEFAULT_INDEX_FOLDER_IMAGE;

		// Diretório virtual para o índice
		try {
			indexDirectoryAnnotation = FSDirectory
					.open(new File(dirAnnotation));
			indexDirectoryXmlAnnotation = FSDirectory.open(new File(
					dirXmlAnnotation));
			indexDirectoryImageData = FSDirectory.open(new File(dirImagesData));
		} catch (IOException e) {
			e.printStackTrace();
		}
	}

	public Hashtable<String, ArrayList<Hit>> imageSearch(QueryMI queryMI) {
		Query query;
		Hashtable<String, ArrayList<Hit>> result = new Hashtable<String, ArrayList<Hit>>();
		try {
			initSearchAnnotation(indexDirectoryImageData);
```

```
if (isSearchInit) {
	Enumeration<String> imageStringQuery = queryMI
			.getImageDataStringToSearch().keys();
	while (imageStringQuery.hasMoreElements()) {
		String key = imageStringQuery.nextElement();
		query = new QueryParser(Version.LUCENE_33, key,
				new StopAnalyzer(Version.LUCENE_33)).parse(queryMI
				.getImageDataStringToSearch().get(key));
		TopScoreDocCollector collector = TopScoreDocCollector
				.create(queryMI.numberOfResults(), true);
		searcherAnnotation.search(query, collector);
		result.put(key,
				getHitResults(collector.topDocs().scoreDocs));
	}
	Enumeration<String> imageDateQuery = queryMI
			.getImageDataDateToSearch().keys();
	while (imageDateQuery.hasMoreElements()) {
		String key = imageDateQuery.nextElement();
		Calendar[] value = queryMI.getImageDataDateToSearch().get(
				key);
		String dateFrom = DateTools.dateToString(
				value[0].getTime(), Resolution.SECOND);
		String dateTo = DateTools.dateToString(value[1].getTime(),
				Resolution.SECOND);
		query = new MatchAllDocsQuery();
		Term lowerTerm = new Term(key, dateFrom);
		Term upperTerm = new Term(key, dateTo);
		TermRangeFilter filter = new TermRangeFilter(key,
				lowerTerm.text(), upperTerm.text(), true, true);
		TopDocs hits = searcherAnnotation.search(query, filter,
				queryMI.numberOfResults());
		result.put(key, getHitResults(hits.scoreDocs));
	}
	Enumeration<String> imageIntQuery = queryMI
			.getImageDataIntegerToSearch().keys();
	while (imageIntQuery.hasMoreElements()) {
		String key = imageIntQuery.nextElement();
		Integer[] value = queryMI.getImageDataIntegerToSearch()
				.get(key);
		// página215
```

```
                        query = NumericRangeQuery.newIntRange(key, value[0],
                                    value[1], true, true);
                        TopScoreDocCollector collector = TopScoreDocCollector
                                    .create(queryMI.numberOfResults(), true);
                        searcherAnnotation.search(query, collector);
                        result.put(key,
                                    getHitResults(collector.topDocs().scoreDocs));
                  }
            }
      } catch (Exception e) {
            e.printStackTrace();
      }

      return result;
}

public Hashtable<String, ArrayList<Hit>> simpleSearch(QueryMI queryMI) {
      Query query;
      Hashtable<String, ArrayList<Hit>> result = new Hashtable<String, ArrayList<Hit>>();
      try {
            if (queryMI.isSearchXmlAnnotation()) {
                  initSearchAnnotation(indexDirectoryXmlAnnotation);
                  if (isSearchInit) {
                        query = new TermQuery(new Term(
                                    queryMI.getPathValueToSearch(),
                                    queryMI.getWordValueToSearch()));
                        TopScoreDocCollector collector = TopScoreDocCollector
                                    .create(queryMI.numberOfResults(), true);
                        searcherAnnotation.search(query, collector);
                        result.put(MIKeys.SEARCH_ANNOT_XML,
                                    getHitResults(collector.topDocs().scoreDocs));
                  } else {
                        result.put(MIKeys.SEARCH_ANNOT_XML, new ArrayList<Hit>());
                  }
            }
            if (queryMI.isSearchTxtAnnotation()) {
                  initSearchAnnotation(indexDirectoryAnnotation);
                  if (isSearchInit) {
                        query = new TermQuery(new Term(
                                    MIConstants.DEFAULT_SEARCH_PATH,
```

```
                                        queryMI.getWordValueToSearch()));
                                TopScoreDocCollector collector = TopScoreDocCollector
                                        .create(queryMI.numberOfResults(), true);
                                searcherAnnotation.search(query, collector);
                                result.put(MIKeys.SEARCH_ANNOT_TXT,
                                        getHitResults(collector.topDocs().scoreDocs));
                        } else {
                                result.put(MIKeys.SEARCH_ANNOT_XML, new ArrayList<Hit>());
                        }
                }
        } catch (Exception e) {
                e.printStackTrace();
        }

        return result;

}

public Hashtable<String, ArrayList<Hit>> substringSearch(QueryMI queryMI) {
        Query query;
        Hashtable<String, ArrayList<Hit>> result = new Hashtable<String, ArrayList<Hit>>();
        try {
                if (queryMI.isSearchXmlAnnotation()) {
                        initSearchAnnotation(indexDirectoryXmlAnnotation);
                        if (isSearchInit) {
                                query = new WildcardQuery(new Term(
                                        queryMI.getPathValueToSearch(), "*"
                                                + queryMI.getWordValueToSearch() + "*"));
                                TopScoreDocCollector collector = TopScoreDocCollector
                                        .create(queryMI.numberOfResults(), true);
                                searcherAnnotation.search(query, collector);
                                result.put(MIKeys.SEARCH_ANNOT_XML,
                                        getHitResults(collector.topDocs().scoreDocs));
                        } else {
                                result.put(MIKeys.SEARCH_ANNOT_XML, new ArrayList<Hit>());
                        }
                }
                if (queryMI.isSearchTxtAnnotation()) {
                        initSearchAnnotation(indexDirectoryAnnotation);
                        if (isSearchInit) {
```

```
                                query = new WildcardQuery(new Term(
                                            MIConstants.DEFAULT_SEARCH_PATH, "*"
                                                        + queryMI.getWordValueToSearch() + "*"));
                                TopScoreDocCollector collector = TopScoreDocCollector
                                            .create(queryMI.numberOfResults(), true);
                                searcherAnnotation.search(query, collector);
                                result.put(MIKeys.SEARCH_ANNOT_TXT,
                                            getHitResults(collector.topDocs().scoreDocs));
                          } else {
                                result.put(MIKeys.SEARCH_ANNOT_TXT, new ArrayList<Hit>());
                          }
                    }
              } catch (Exception e) {
                    e.printStackTrace();
              }

              return result;

        }

        public Hashtable<String, Hashtable<String, ArrayList<Hit>>> synonismSearch(
                    QueryMI queryMI) {
              Hashtable<String, Hashtable<String, ArrayList<Hit>>> result = new Hashtable<String, Hashtable<String,
ArrayList<Hit>>>();
              QueryMI queryAux = queryMI.clone();
              ArrayList<String> synonysms = HelperFunctions.getSynonysm(queryAux
                          .getWordValueToSearch());
              if (synonysms.size() > 0) {
                    for (String synonysm : synonysms) {
                          queryAux.getAnnotationDataToSearch().put(
                                      MIKeys.SEARCH_WORD_VALUE, synonysm);
                          result.put(synonysm, simpleSearch(queryAux));
                    }
              }
              return result;
        }

        public Hashtable<String, ArrayList<Hit>> similaritySearch(QueryMI queryMI) {
              Query query;
              Hashtable<String, ArrayList<Hit>> result = new Hashtable<String, ArrayList<Hit>>();
```

```
try {
	float similarity = queryMI.wordSimilarity();
	if (similarity == 0) {
		similarity = MIConstants.DEFAULT_SEARCH_SIMILARITY_VALUE;
	}
	if (queryMI.isSearchXmlAnnotation()) {
		initSearchAnnotation(indexDirectoryXmlAnnotation);
		if (isSearchInit) {
			query = new FuzzyQuery(new Term(
					queryMI.getPathValueToSearch(),
					queryMI.getWordValueToSearch()), similarity);
			TopScoreDocCollector collector = TopScoreDocCollector
					.create(queryMI.numberOfResults(), true);
			searcherAnnotation.search(query, collector);
			result.put(MIKeys.SEARCH_ANNOT_XML,
					getHitResults(collector.topDocs().scoreDocs));
		} else {
			result.put(MIKeys.SEARCH_ANNOT_XML, new ArrayList<Hit>());
		}
	}
	if (queryMI.isSearchTxtAnnotation()) {
		initSearchAnnotation(indexDirectoryAnnotation);
		if (isSearchInit) {
			query = new FuzzyQuery(new Term(
					MIConstants.DEFAULT_SEARCH_PATH,
					queryMI.getWordValueToSearch()), similarity);
			TopScoreDocCollector collector = TopScoreDocCollector
					.create(queryMI.numberOfResults(), true);
			searcherAnnotation.search(query, collector);
			result.put(MIKeys.SEARCH_ANNOT_TXT,
					getHitResults(collector.topDocs().scoreDocs));
		} else {
			result.put(MIKeys.SEARCH_ANNOT_TXT, new ArrayList<Hit>());
		}
	}
} catch (Exception e) {
	e.printStackTrace();
}

return result;
```

```
}

/*
 * public ScoreDoc[] compoundWordSearch(QueryMI queryMI){ PhraseQuery term;
 * Hashtable<String, ArrayList<String>> result = new Hashtable<String,
 * ArrayList<String>>(); try { if (queryMI.isSearchXmlAnnotation()) { term =
 * new PhraseQuery(); term.setSlop(0);
 * initSearchAnnotation(indexDirectoryXmlAnnotation); if( isSearchInit ){
 * term.add(new Term(MIConstants.DEFAULT_SEARCH_PATH, valueToSearch[0]));
 * term.add(new Term(MIConstants.DEFAULT_SEARCH_PATH, valueToSearch[1]));
 * TopScoreDocCollector collector =
 * TopScoreDocCollector.create(queryMI.numberOfResults(), true);
 * searcherAnnotation.search(term, collector);
 * result.put(MIKeys.SEARCH_ANNOT_XML,
 * getWordResults(collector.topDocs().scoreDocs)); }else{
 * result.put(MIKeys.SEARCH_ANNOT_XML, new ArrayList<String>()); } } if(
 * isSearchInit ){ PhraseQuery term = new PhraseQuery(); term.setSlop(0); if
 * (isXMLAnnotation) { term.add(new Term(valueToSearch[0],
 * valueToSearch[1])); term.add(new Term(valueToSearch[0],
 * valueToSearch[2])); }else{ term.add(new
 * Term(MIConstants.DEFAULT_SEARCH_PATH, valueToSearch[0])); term.add(new
 * Term(MIConstants.DEFAULT_SEARCH_PATH, valueToSearch[1])); }
 * TopScoreDocCollector collector =
 * TopScoreDocCollector.create(numberOfresults, true);
 * searcherAnnotation.search(term, collector); return
 * collector.topDocs().scoreDocs; }else{
 * System.out.println("Searcher não inicializado!"); } } catch
 * (CorruptIndexException e) { e.printStackTrace(); } catch (IOException e)
 * { System.out.println("Nao foi possivel executar query. " + e); } return
 * null; }
 */

@SuppressWarnings("deprecation")
public void initSearchAnnotation(Directory indexDirectory) {
	isSearchInit = false;
	try {
		searcherAnnotation = new ~~IndexSearcher~~(indexDirectory, true);
		isSearchInit = true;
	} catch (CorruptIndexException e) {
```

```
                e.printStackTrace();
            } catch (IOException e) {
                e.printStackTrace();
            }
        }

        private ArrayList<Hit> getHitResults(ScoreDoc[] hits) {
            ArrayList<Hit> hitsCleared = new ArrayList<Hit>();
            if (hits.length > 0) {
                try {
                    for (int i = 0; i < hits.length; i++) {
                        int docId = hits[i].doc;
                        float score = hits[i].score;
                        Document d = searcherAnnotation.doc(docId);
                        if (!hitsCleared.contains(d.get("Path"))) {
                            int sim = (int) ((score) * 100);
                            if (sim > 100)
                                sim = 100;
                            hitsCleared.add(new Hit(d.get("Path"), sim));
                        }
                    }

                } catch (CorruptIndexException e) {
                    e.printStackTrace();
                } catch (IOException e) {
                    e.printStackTrace();
                }
            }
            return hitsCleared;
        }

}
```

Apêndice P – Classe *MetadataIngesterInfo*

```
package tcc.tbfm.model.objectsUtils;

import java.io.File;

public class MetadataIngesterInfo {

	private File folderMetadata;
	private File folderImage;
	private File folderIndex;
	private boolean follow;

	public MetadataIngesterInfo(String folderMetadata, String folderImage,
			String folderIndex, boolean follow) {
		this.folderMetadata = new File(folderMetadata);
		this.folderImage = new File(folderImage);
		this.folderIndex = new File(folderIndex);
		this.setFollow(follow);
	}

	public File getFolderMetadata() {
		return folderMetadata;
	}

	public void setFolderMetadata(String folderMetadata) {
		this.folderMetadata = new File(folderMetadata);
	}

	public File getFolderImage() {
		return folderImage;
	}

	public void setFolderImage(String folderImage) {
		this.folderImage = new File(folderImage);
	}
```

```
	public File getFolderIndex() {
		return folderIndex;
	}

	public void setFolderIndex(String folderIndex) {
		this.folderIndex = new File(folderIndex);
	}

	public boolean isFollow() {
		return follow;
	}

	public void setFollow(boolean follow) {
		this.follow = follow;
	}

}
```

Apêndice Q – Classe *QueryMI*

```
package tcc.tbfm.model.objectsUtils;

import java.util.Calendar;
import java.util.Hashtable;

import tcc.tbfm.model.classesUtils.MIConstants;
import tcc.tbfm.model.classesUtils.MIKeys;

public class QueryMI {

	// Informações sobre anotações
	private Hashtable<String, String> annotationDataToSearch;

	// informações sobre a imagem
	private Hashtable<String, String> imageDataStringToSearch;
	private Hashtable<String, Integer[]> imageDataIntToSearch;
	private Hashtable<String, Calendar[]> imageDataDateToSearch;

	// informações sobre a busca
	private Hashtable<String, String> searchConfig;

	// informações sobre relevância
	private Hashtable<String, Integer> relevances;

	public QueryMI(Hashtable<String, String> searchConfig,
			Hashtable<String, String> annotationDataToSearch,
			Hashtable<String, String> imageDataStringToSearch,
			Hashtable<String, Integer[]> imageDataIntToSearch,
			Hashtable<String, Calendar[]> imageDataDateToSearch,
			Hashtable<String, Integer> relevances) {
		this.searchConfig = searchConfig;
		this.annotationDataToSearch = annotationDataToSearch == null ? new Hashtable<String, String>()
				: annotationDataToSearch;
		this.imageDataStringToSearch = imageDataStringToSearch == null ? new Hashtable<String, String>()
```

```
                : imageDataStringToSearch;
        this.imageDataIntToSearch = imageDataIntToSearch == null ? new Hashtable<String, Integer[]>()
                : imageDataIntToSearch;
        this.imageDataDateToSearch = imageDataDateToSearch == null ? new Hashtable<String, Calendar[]>()
                : imageDataDateToSearch;
        this.relevances = relevances == null ? new Hashtable<String, Integer>()
                : relevances;
}

public int numberOfResults() {
        if (searchConfig.containsKey(MIKeys.SEARCH_NUMBER_OF_RESULTS)) {
                return Integer.parseInt(searchConfig
                                .get(MIKeys.SEARCH_NUMBER_OF_RESULTS));
        } else {
                return MIConstants.DEFAULT_SEARCH_NUMBER_OF_RESULTS;
        }
}

public String pathOfResults() {
        if (searchConfig.containsKey(MIKeys.SEARCH_PATH_OF_RESULTS)) {
                return searchConfig.get(MIKeys.SEARCH_PATH_OF_RESULTS);
        } else {
                return "";
        }
}

public Hashtable<String, String> getImageDataStringToSearch() {
        return imageDataStringToSearch;
}

public Hashtable<String, Integer[]> getImageDataIntegerToSearch() {
        return imageDataIntToSearch;
}

public Hashtable<String, Calendar[]> getImageDataDateToSearch() {
        return imageDataDateToSearch;
}

public Hashtable<String, String> getAnnotationDataToSearch() {
        return annotationDataToSearch;
```

```
}

public int getRelevanceOfKey(String key) {
	return relevances.get(key);
}

public String getValueOfKeyOfImgHash(String key) {

	if (imageDataStringToSearch != null) {
		if (imageDataStringToSearch.containsKey(key))
			return "'" + imageDataStringToSearch.get(key) + "'";
	}

	if (imageDataIntToSearch != null) {
		if (imageDataIntToSearch.containsKey(key))
			return "de '" + imageDataIntToSearch.get(key)[0] + "' até '"
					+ imageDataIntToSearch.get(key)[1] + "'";
	}

	if (imageDataDateToSearch != null) {
		if (imageDataDateToSearch.containsKey(key)) {
			Calendar[] valueCal = imageDataDateToSearch.get(key);
			return "de '" + valueCal[0].get(Calendar.DATE) + "/"
					+ (valueCal[0].get(Calendar.MONTH) + 1) + "/"
					+ valueCal[0].get(Calendar.YEAR) + "' até '"
					+ valueCal[1].get(Calendar.DATE) + "/"
					+ (valueCal[1].get(Calendar.MONTH) + 1) + "/"
					+ valueCal[1].get(Calendar.YEAR) + "'";
		}
	}

	return "";

}

public Hashtable<String, String> getSearchConfig() {
	return searchConfig;
}

public float wordSimilarity() {
```

```
		float similarity = MIConstants.DEFAULT_SEARCH_SIMILARITY_VALUE;
		if (annotationDataToSearch
				.containsKey(MIKeys.SEARCH_WORD_SIMILARITY_VALUE)) {
			String results = annotationDataToSearch
					.get(MIKeys.SEARCH_WORD_SIMILARITY_VALUE);
			if (!results.equals("0")) {
				similarity = Float.parseFloat(results);
			}
		}
		return similarity / (float) 100;
	}

	public float pathSimilarity() {
		float similarity = MIConstants.DEFAULT_SEARCH_SIMILARITY_VALUE;
		if (annotationDataToSearch
				.containsKey(MIKeys.SEARCH_PATH_SIMILARITY_VALUE)) {
			String results = annotationDataToSearch
					.get(MIKeys.SEARCH_PATH_SIMILARITY_VALUE);
			if (!results.equals("0")) {
				similarity = Float.parseFloat(results);
			}
		}
		return similarity / (float) 100;
	}

	public boolean isSearchXmlAnnotation() {
		if (searchConfig.containsKey(MIKeys.SEARCH_ANNOT_XML)) {
			return searchConfig.get(MIKeys.SEARCH_ANNOT_XML).equalsIgnoreCase(
					"true");
		} else {
			return false;
		}
	}

	public boolean isSearchTxtAnnotation() {
		if (searchConfig.containsKey(MIKeys.SEARCH_ANNOT_TXT)) {
			return searchConfig.get(MIKeys.SEARCH_ANNOT_TXT).equalsIgnoreCase(
					"true");
		} else {
			return false;
```

```
	}
}

public boolean isSearchImageData() {
	if (searchConfig.containsKey(MIKeys.SEARCH_IMAGE_DATA)) {
		return searchConfig.get(MIKeys.SEARCH_IMAGE_DATA).equalsIgnoreCase(
				"true");
	} else {
		return false;
	}
}

public boolean isSearchWordSimilarity() {
	if (searchConfig.containsKey(MIKeys.SEARCH_WORD_SIMILARITY)) {
		return searchConfig.get(MIKeys.SEARCH_WORD_SIMILARITY)
				.equalsIgnoreCase("true");
	} else {
		return false;
	}
}

public boolean isSearchPathSimilarity() {
	if (searchConfig.containsKey(MIKeys.SEARCH_PATH_SIMILARITY)) {
		return searchConfig.get(MIKeys.SEARCH_PATH_SIMILARITY)
				.equalsIgnoreCase("true");
	} else {
		return false;
	}
}

public boolean isSearchWordSynonyms() {
	if (searchConfig.containsKey(MIKeys.SEARCH_WORD_SYNONYSM)) {
		return searchConfig.get(MIKeys.SEARCH_WORD_SYNONYSM)
				.equalsIgnoreCase("true");
	} else {
		return false;
	}
}

public boolean isSearchPathSynonyms() {
```

```
	if (searchConfig.containsKey(MIKeys.SEARCH_PATH_SYNONYSM)) {
		return searchConfig.get(MIKeys.SEARCH_PATH_SYNONYSM)
				.equalsIgnoreCase("true");
	} else {
		return false;
	}
}

public boolean isSearchWordSubstring() {
	if (searchConfig.containsKey(MIKeys.SEARCH_WORD_SUBSTRING)) {
		return searchConfig.get(MIKeys.SEARCH_WORD_SUBSTRING)
				.equalsIgnoreCase("true");
	} else {
		return false;
	}
}

public boolean isSearchPathSubstring() {
	if (searchConfig.containsKey(MIKeys.SEARCH_PATH_SUBSTRING)) {
		return searchConfig.get(MIKeys.SEARCH_PATH_SUBSTRING)
				.equalsIgnoreCase("true");
	} else {
		return false;
	}
}

public boolean isSearchWordSimple() {
	if (searchConfig.containsKey(MIKeys.SEARCH_WORD_SIMPLE)) {
		return searchConfig.get(MIKeys.SEARCH_WORD_SIMPLE)
				.equalsIgnoreCase("true");
	} else {
		return false;
	}
}

public boolean isSearchPathSimple() {
	if (searchConfig.containsKey(MIKeys.SEARCH_PATH_SIMPLE)) {
		return searchConfig.get(MIKeys.SEARCH_PATH_SIMPLE)
				.equalsIgnoreCase("true");
	} else {
```

```
			return false;
		}
	}

	public boolean isSearchWordOnlySource() {
		if (searchConfig.containsKey(MIKeys.SEARCH_WORD_ONLY_SOURCE)) {
			return searchConfig.get(MIKeys.SEARCH_WORD_ONLY_SOURCE)
					.equalsIgnoreCase("true");
		} else {
			return false;
		}
	}

	public boolean isSearchPathOnlySource() {
		if (searchConfig.containsKey(MIKeys.SEARCH_PATH_ONLY_SOURCE)) {
			return searchConfig.get(MIKeys.SEARCH_PATH_ONLY_SOURCE)
					.equalsIgnoreCase("true");
		} else {
			return false;
		}
	}

	public boolean isSearchOfPossiblesPaths() {
		if (searchConfig.containsKey(MIKeys.SEARCH_OF_POSSIBLES_PATHS)) {
			return searchConfig.get(MIKeys.SEARCH_OF_POSSIBLES_PATHS)
					.equalsIgnoreCase("true");
		} else {
			return false;
		}
	}

	public String getWordValueToSearch() {
		String valueToSearch = "";
		if (annotationDataToSearch.containsKey(MIKeys.SEARCH_WORD_VALUE)) {
			String results = annotationDataToSearch
					.get(MIKeys.SEARCH_WORD_VALUE);
			if (!results.isEmpty()) {
				valueToSearch = results;
			}
		}
```

```
		return valueToSearch;
	}

	public String getPathValueToSearch() {
		String pathToSearch = "";
		if (annotationDataToSearch.containsKey(MIKeys.SEARCH_PATH_VALUE)) {
			String results = annotationDataToSearch
					.get(MIKeys.SEARCH_PATH_VALUE);
			if (!results.isEmpty()) {
				pathToSearch = results;
			}
		}
		return pathToSearch;
	}

	@SuppressWarnings("unchecked")
	@Override
	public QueryMI clone() {
		Hashtable<String, String> annotationDataToSearchAux = (Hashtable<String, String>) annotationDataToSearch
				.clone();
		Hashtable<String, String> searchConfigAux = (Hashtable<String, String>) searchConfig
				.clone();
		Hashtable<String, Integer[]> intImgAux = (Hashtable<String, Integer[]>) imageDataIntToSearch
				.clone();
		Hashtable<String, String> strImgAux = (Hashtable<String, String>) imageDataStringToSearch
				.clone();
		Hashtable<String, Calendar[]> dtImgAux = (Hashtable<String, Calendar[]>) imageDataDateToSearch
				.clone();
		Hashtable<String, Integer> relevAux = (Hashtable<String, Integer>) relevances
				.clone();
		return new QueryMI(searchConfigAux, annotationDataToSearchAux,
				strImgAux, intImgAux, dtImgAux, relevAux);
	}

}
```

Apêndice R – Classe *Runner*

```
package tcc.tbfm.run;

import javax.swing.JDialog;
import javax.swing.JFrame;
import javax.swing.UIManager;

import tcc.tbfm.control.CtrlMainFrame;

public final class Runner {

	public static void main(String[] args) {

		JFrame.setDefaultLookAndFeelDecorated(true);
		JDialog.setDefaultLookAndFeelDecorated(true);
		try {
			UIManager
					.setLookAndFeel("com.sun.java.swing.plaf.windows.WindowsLookAndFeel");
		} catch (Exception ex) {
			System.out.println("Failed loading L&F: ");
			System.out.println(ex);
		}
		new CtrlMainFrame();

	}

}
```

Apêndice S – Classe *MainFrame*

```
package tcc.tbfm.view;

import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.util.Enumeration;

import javax.swing.JFileChooser;
import javax.swing.JRadioButton;

import tcc.tbfm.model.classesUtils.MIConstants;

public class MainFrame extends javax.swing.JFrame {

	private static final long serialVersionUID = -3982686670719306200L;

	public MainFrame() {
		initComponents();
		setActionListeners();
		setCmbColors();
		setCmbType();
		setCmbColorMode();
		setCmbCamera();
		// setDefaultValues();
	}

	@SuppressWarnings({ "unchecked", "rawtypes" })
	private void setCmbColors() {
		cmbPredominantColor.setModel(new javax.swing.DefaultComboBoxModel(
				new String[] {}));
		cmbPredominantColor.addItem(MIConstants.IMAGE_NOT_SELECTED_INFO);
		cmbPredominantColor.addItem(MIConstants.IMAGE_COLOR_BLACK);
		cmbPredominantColor.addItem(MIConstants.IMAGE_COLOR_BLUE);
		cmbPredominantColor.addItem(MIConstants.IMAGE_COLOR_BROWN);
		cmbPredominantColor.addItem(MIConstants.IMAGE_COLOR_GRAY);
```

```
        cmbPredominantColor.addItem(MIConstants.IMAGE_COLOR_GREEN);
        cmbPredominantColor.addItem(MIConstants.IMAGE_COLOR_ORANGE);
        cmbPredominantColor.addItem(MIConstants.IMAGE_COLOR_PINK);
        cmbPredominantColor.addItem(MIConstants.IMAGE_COLOR_PURPLE);
        cmbPredominantColor.addItem(MIConstants.IMAGE_COLOR_RED);
        cmbPredominantColor.addItem(MIConstants.IMAGE_COLOR_WHITE);
        cmbPredominantColor.addItem(MIConstants.IMAGE_COLOR_YELLOW);
    }

    @SuppressWarnings({ "unchecked", "rawtypes" })
    private void setCmbType() {
        cmbType.setModel(new javax.swing.DefaultComboBoxModel(new String[] {
                    MIConstants.IMAGE_NOT_SELECTED_INFO, "JPG", "JPEG", "GIF",
                    "TIFF", "PNG", "BMP" }));
    }

    @SuppressWarnings({ "unchecked", "rawtypes" })
    private void setCmbColorMode() {
        cmbColor.setModel(new javax.swing.DefaultComboBoxModel(
                    new String[] { MIConstants.IMAGE_NOT_SELECTED_INFO, "RGB",
                                "GRAYSCALE", "CMYK" }));
    }

    @SuppressWarnings({ "unchecked", "rawtypes" })
    private void setCmbCamera() {
        cmbCamera.setModel(new javax.swing.DefaultComboBoxModel(new String[] {
                    MIConstants.IMAGE_NOT_SELECTED_INFO, "NIKON" }));
    }

    private void setDefaultValues() {
        txtImage.setText("E:\\tcc\\image");
        txtMetadata.setText("E:\\tcc\\annotations");
        txtIndex.setText("E:\\tcc\\index");
        txtPathResult.setText("E:\\tcc");
    }

    public javax.swing.JButton getBtnIndexIt() {
        return btnIndexIt;
    }
```

```
public javax.swing.JButton getBtnReuseIndex() {
	return btnReuseIndex;
}

public javax.swing.JButton getBtnSearchIt() {
	return btnSearch;
}

public javax.swing.JButton getBtnSelectTag() {
	return btnSelectTag;
}

public javax.swing.JTextField getTxtMetadataPath() {
	return txtMetadata;
}

public javax.swing.JTextField getTxtImagePath() {
	return txtImage;
}

public javax.swing.JTextField getTxtIndexPath() {
	return txtIndex;
}

public void setLblResult(String res) {
	lblIndexResult.setText(res);
}

public boolean hasWidth() {
	return !((txtWidthFrom.getText().equals("0") || txtWidthFrom.getText()
			.equals("")) && (txtWidthTo.getText().equals("0") || txtWidthTo
			.getText().equals("")));
}

public Integer[] getTxtWidth() {
	Integer from = Integer.parseInt(txtWidthFrom.getText().equals("") ? "0"
			: txtWidthFrom.getText());
	Integer to = Integer.parseInt(txtWidthTo.getText().equals("") ? "0"
			: txtWidthTo.getText());
	return new Integer[] { from, to };
```

```
}

public boolean hasHeigth() {
	return !((txtHeigthFrom.getText().equals("0") || txtHeigthFrom
			.getText().equals("")) && (txtHeigthTo.getText().equals("0") || txtHeigthTo
			.getText().equals("")));
}

public Integer[] getTxtHeigth() {
	Integer from = Integer
			.parseInt(txtHeigthFrom.getText().equals("") ? "0"
					: txtHeigthFrom.getText());
	Integer to = Integer.parseInt(txtHeigthTo.getText().equals("") ? "0"
			: txtHeigthTo.getText());
	return new Integer[] { from, to };
}

public boolean hasSize() {
	return !((txtSizeFrom.getText().equals("0") || txtSizeFrom.getText()
			.equals("")) && (txtSizeTo.getText().equals("0") || txtSizeTo
			.getText().equals("")));
}

public Integer[] getTxtSize() {
	Integer from = Integer.parseInt(txtSizeFrom.getText().equals("") ? "0"
			: txtSizeFrom.getText());
	Integer to = Integer.parseInt(txtSizeTo.getText().equals("") ? "0"
			: txtSizeTo.getText());
	return new Integer[] { from, to };
}

public boolean hasXResolution() {
	return !((txtXResolutionFrom.getText().equals("0") || txtXResolutionFrom
			.getText().equals("")) && (txtXResolutionTo.getText().equals(
			"0") || txtXResolutionTo.getText().equals("")));
}

public Integer[] getTxtXResolution() {
	Integer from = Integer
			.parseInt(txtXResolutionFrom.getText().equals("") ? "0"
```

```
                                : txtXResolutionFrom.getText());
        Integer to = Integer
                    .parseInt(txtXResolutionTo.getText().equals("") ? "0"
                                : txtXResolutionTo.getText());
        return new Integer[] { from, to };
}

public boolean hasYResolution() {
        return !((txtYResolutionFrom.getText().equals("0") || txtYResolutionFrom
                    .getText().equals("")) && (txtYResolutionTo.getText().equals(
                    "0") || txtYResolutionTo.getText().equals("")));
}

public Integer[] getTxtYResolution() {
        Integer from = Integer
                    .parseInt(txtYResolutionFrom.getText().equals("") ? "0"
                                : txtYResolutionFrom.getText());
        Integer to = Integer
                    .parseInt(txtYResolutionTo.getText().equals("") ? "0"
                                : txtYResolutionTo.getText());
        return new Integer[] { from, to };
}

public boolean hasPercentWhite() {
        return !((txtWhiteFrom.getText().equals("0") || txtWhiteFrom.getText()
                    .equals("")) && (txtWhiteTo.getText().equals("0") || txtWhiteTo
                    .getText().equals("")));
}

public Integer[] getTxtPercentWhite() {
        Integer from = Integer.parseInt(txtWhiteFrom.getText().equals("") ? "0"
                    : txtWhiteFrom.getText());
        Integer to = Integer.parseInt(txtWhiteTo.getText().equals("") ? "0"
                    : txtWhiteTo.getText());
        return new Integer[] { from, to };
}

public boolean hasPercentBlack() {
        return !((txtBlackFrom.getText().equals("0") || txtBlackFrom.getText()
                    .equals("")) && (txtBlackTo.getText().equals("0") || txtBlackTo
```

```
                    .getText().equals("")));
    }

    public Integer[] getTxtPercentBlack() {
          Integer from = Integer.parseInt(txtBlackFrom.getText().equals("") ? "0"
                      : txtBlackFrom.getText());
          Integer to = Integer.parseInt(txtBlackTo.getText().equals("") ? "0"
                      : txtBlackTo.getText());
          return new Integer[] { from, to };
    }

    public boolean hasPercentBlue() {
          return !((txtBlueFrom.getText().equals("0") || txtBlueFrom.getText()
                      .equals("")) && (txtBlueTo.getText().equals("0") || txtBlueTo
                      .getText().equals("")));
    }

    public Integer[] getTxtPercentBlue() {
          Integer from = Integer.parseInt(txtBlueFrom.getText().equals("") ? "0"
                      : txtBlueFrom.getText());
          Integer to = Integer.parseInt(txtBlueTo.getText().equals("") ? "0"
                      : txtBlueTo.getText());
          return new Integer[] { from, to };
    }

    public boolean hasPercentBrown() {
          return !((txtBrownFrom.getText().equals("0") || txtBrownFrom.getText()
                      .equals("")) && (txtBrownTo.getText().equals("0") || txtBrownTo
                      .getText().equals("")));
    }

    public Integer[] getTxtPercentBrown() {
          Integer from = Integer.parseInt(txtBrownFrom.getText().equals("") ? "0"
                      : txtBrownFrom.getText());
          Integer to = Integer.parseInt(txtBrownTo.getText().equals("") ? "0"
                      : txtBrownTo.getText());
          return new Integer[] { from, to };
    }

    public boolean hasPercentGray() {
```

```
        return !((txtGrayFrom.getText().equals("0") || txtGrayFrom.getText()
                        .equals("")) && (txtGrayTo.getText().equals("0") || txtGrayTo
                        .getText().equals("")));
    }

    public Integer[] getTxtPercentGray() {
        Integer from = Integer.parseInt(txtGrayFrom.getText().equals("") ? "0"
                        : txtGrayFrom.getText());
        Integer to = Integer.parseInt(txtGrayTo.getText().equals("") ? "0"
                        : txtGrayTo.getText());
        return new Integer[] { from, to };
    }

    public boolean hasPercentGreen() {
        return !((txtGreenFrom.getText().equals("0") || txtGreenFrom.getText()
                        .equals("")) && (txtGreenTo.getText().equals("0") || txtGreenTo
                        .getText().equals("")));
    }

    public Integer[] getTxtPercentGreen() {
        Integer from = Integer.parseInt(txtGreenFrom.getText().equals("") ? "0"
                        : txtGreenFrom.getText());
        Integer to = Integer.parseInt(txtGreenTo.getText().equals("") ? "0"
                        : txtGreenTo.getText());
        return new Integer[] { from, to };
    }

    public boolean hasPercentOrange() {
        return !((txtOrangeFrom.getText().equals("0") || txtOrangeFrom
                        .getText().equals("")) && (txtOrangeTo.getText().equals("0") || txtOrangeTo
                        .getText().equals("")));
    }

    public Integer[] getTxtPercentOrange() {
        Integer from = Integer
                        .parseInt(txtOrangeFrom.getText().equals("") ? "0"
                                        : txtOrangeFrom.getText());
        Integer to = Integer.parseInt(txtOrangeTo.getText().equals("") ? "0"
                        : txtOrangeTo.getText());
        return new Integer[] { from, to };
```

```
}

public boolean hasPercentPink() {
	return !((txtPinkFrom.getText().equals("0") || txtPinkFrom.getText()
			.equals("")) && (txtPinkTo.getText().equals("0") || txtPinkTo
			.getText().equals("")));
}

public Integer[] getTxtPercentPink() {
	Integer from = Integer.parseInt(txtPinkFrom.getText().equals("") ? "0"
			: txtPinkFrom.getText());
	Integer to = Integer.parseInt(txtPinkTo.getText().equals("") ? "0"
			: txtPinkTo.getText());
	return new Integer[] { from, to };
}

public boolean hasPercentPurple() {
	return !((txtPurpleFrom.getText().equals("0") || txtPurpleFrom
			.getText().equals("")) && (txtPurpleTo.getText().equals("0") || txtPurpleTo
			.getText().equals("")));
}

public Integer[] getTxtPercentPurple() {
	Integer from = Integer
			.parseInt(txtPurpleFrom.getText().equals("") ? "0"
					: txtPurpleFrom.getText());
	Integer to = Integer.parseInt(txtPurpleTo.getText().equals("") ? "0"
			: txtPurpleTo.getText());
	return new Integer[] { from, to };
}

public boolean hasPercentRed() {
	return !((txtRedFrom.getText().equals("0") || txtRedFrom.getText()
			.equals("")) && (txtRedTo.getText().equals("0") || txtRedTo
			.getText().equals("")));
}

public Integer[] getTxtPercentRed() {
	Integer from = Integer.parseInt(txtRedFrom.getText().equals("") ? "0"
			: txtRedFrom.getText());
```

```
		Integer to = Integer.parseInt(txtRedTo.getText().equals("") ? "0"
				: txtRedTo.getText());
		return new Integer[] { from, to };
	}

	public boolean hasPercentYellow() {
		return !((txtYellowFrom.getText().equals("0") || txtYellowFrom
				.getText().equals("")) && (txtYellowTo.getText().equals("0") || txtYellowTo
				.getText().equals("")));
	}

	public Integer[] getTxtPercentYellow() {
		Integer from = Integer
				.parseInt(txtYellowFrom.getText().equals("") ? "0"
						: txtYellowFrom.getText());
		Integer to = Integer.parseInt(txtYellowTo.getText().equals("") ? "0"
				: txtYellowTo.getText());
		return new Integer[] { from, to };
	}

	public boolean hasCreated() {
		return !(txtCreatedFrom.getText().equals("  /  /    ") || txtCreatedTo
				.getText().equals("  /  /    "));
	}

	public String getTxtCreatedFrom() {
		return txtCreatedFrom.getText();
	}

	public String getTxtCreatedTo() {
		return txtCreatedTo.getText();
	}

	public boolean hasModification() {
		return !(txtModifyFrom.getText().equals("  /  /    ") || txtModifyTo
				.getText().equals("  /  /    "));
	}

	public String getTxtModificationFrom() {
		return txtModifyFrom.getText().equals("") ? "0" : txtModifyFrom
```

```
                    .getText();
}

public String getTxtModificationTo() {
      return txtModifyTo.getText().equals("") ? "0" : txtModifyTo.getText();
}

public String getCmbCamera() {
      return cmbCamera.getSelectedItem().toString();
}

public String getCmbColor() {
      return cmbColor.getSelectedItem().toString();
}

public String getCmbType() {
      return cmbType.getSelectedItem().toString();
}

public String getCmbPredominantColor() {
      return cmbPredominantColor.getSelectedItem().toString();
}

public int getCmbRelevWidth() {
      return cmbRelWidth.getItemAt(cmbRelWidth.getSelectedIndex());
}

public int getCmbRelevHeight() {
      return cmbRelHeight.getItemAt(cmbRelHeight.getSelectedIndex());
}

public int getCmbRelevSize() {
      return cmbRelSize.getItemAt(cmbRelSize.getSelectedIndex());
}

public int getCmbRelevXResol() {
      return cmbRelXResol.getItemAt(cmbRelXResol.getSelectedIndex());
}

public int getCmbRelevYResol() {
```

```
        return cmbRelYResol.getItemAt(cmbRelYResol.getSelectedIndex());
}

public int getCmbRelevCreated() {
        return cmbRelCreated.getItemAt(cmbRelCreated.getSelectedIndex());
}

public int getCmbRelevModif() {
        return cmbRelModif.getItemAt(cmbRelModif.getSelectedIndex());
}

public int getCmbRelevCamera() {
        return cmbRelCamera.getItemAt(cmbRelCamera.getSelectedIndex());
}

public int getCmbRelevColorType() {
        return cmbRelColorType.getItemAt(cmbRelColorType.getSelectedIndex());
}

public int getCmbRelevFormat() {
        return cmbRelFormat.getItemAt(cmbRelFormat.getSelectedIndex());
}

public int getCmbRelevSimple() {
        return cmbRelSimple.getItemAt(cmbRelSimple.getSelectedIndex());
}

public int getCmbRelevSimilarity() {
        return cmbRelSimilarity.getItemAt(cmbRelSimilarity.getSelectedIndex());
}

public int getCmbRelevSynonysm() {
        return cmbRelSynonysm.getItemAt(cmbRelSynonysm.getSelectedIndex());
}

public int getCmbRelevSubstring() {
        return cmbRelSubstring.getItemAt(cmbRelSubstring.getSelectedIndex());
}

public int getCmbRelevPredominant() {
```

```
        return cmbRelPredominant
                    .getItemAt(cmbRelPredominant.getSelectedIndex());
}

public int getCmbRelevBlack() {
        return cmbRelBlack.getItemAt(cmbRelBlack.getSelectedIndex());
}

public int getCmbRelevBlue() {
        return cmbRelBlue.getItemAt(cmbRelBlue.getSelectedIndex());
}

public int getCmbRelevBrown() {
        return cmbRelBrown.getItemAt(cmbRelBrown.getSelectedIndex());
}

public int getCmbRelevGray() {
        return cmbRelGray.getItemAt(cmbRelGray.getSelectedIndex());
}

public int getCmbRelevGreen() {
        return cmbRelGreen.getItemAt(cmbRelGreen.getSelectedIndex());
}

public int getCmbRelevOrange() {
        return cmbRelOrange.getItemAt(cmbRelOrange.getSelectedIndex());
}

public int getCmbRelevPink() {
        return cmbRelPink.getItemAt(cmbRelPink.getSelectedIndex());
}

public int getCmbRelevPurple() {
        return cmbRelPurple.getItemAt(cmbRelPurple.getSelectedIndex());
}

public int getCmbRelevRed() {
        return cmbRelRed.getItemAt(cmbRelRed.getSelectedIndex());
}
```

```
public int getCmbRelevWhite() {
	return cmbRelWhite.getItemAt(cmbRelWhite.getSelectedIndex());
}

public int getCmbRelevYellow() {
	return cmbRelYellow.getItemAt(cmbRelYellow.getSelectedIndex());
}

public boolean isSimpleSearch() {
	return boxSimple.isSelected();
}

public boolean isSimilaritySearch() {
	return boxSimilarity.isSelected();
}

public String similarity() {
	return txtSimilarity.getText();
}

public boolean isSubstringSearch() {
	return boxSubstring.isSelected();
}

public boolean isSynonysmSearch() {
	return boxSynonysm.isSelected();
}

public boolean isSearchAnnotation() {
	return !getRdBtnTextType().equals("Nenhum");
}

public boolean isOnlySimpleSearch() {
	return (isSynonysmSearch() == false && isSubstringSearch() == false
			&& isSimilaritySearch() == false && isSimpleSearch() == true);
}

public String getRdBtnTextType() {
	for (@SuppressWarnings("rawtypes")
	Enumeration e = btnMetadataFormtat.getElements(); e.hasMoreElements();) {
```

```
			JRadioButton b = (JRadioButton) e.nextElement();
			if (b.isSelected()) {
				return b.getText();
			}
		}
		return "";
	}

	public javax.swing.JTextField getTxtTag() {
		return txtTag;
	}

	public String getTxtWord() {
		return txtWord.getText();
	}

	public String getNumberOfResults() {
		if (txtNumberResults.getText().equalsIgnoreCase("")) {
			return MIConstants.DEFAULT_SEARCH_NUMBER_OF_RESULTS + "";
		} else {
			return txtNumberResults.getText();
		}
	}

	public String getPathOfResults() {
		return txtPathResult.getText();
	}

	public boolean isFollow() {
		for (@SuppressWarnings("rawtypes")
		Enumeration e = btnFollow.getElements(); e.hasMoreElements();) {
			JRadioButton b = (JRadioButton) e.nextElement();
			if (b.isSelected()) {
				return b.getText().equals("Sim");
			}
		}
		return false;
	}

	private ActionListener selectMetadataListener() {
```

```
		return new ActionListener() {
			@Override
			public void actionPerformed(ActionEvent e) {
				selectFile.setFileSelectionMode(JFileChooser.DIRECTORIES_ONLY);
				int opt = selectFile.showOpenDialog(null);
				if (opt != JFileChooser.CANCEL_OPTION) {
					String path = selectFile.getSelectedFile()
							.getAbsolutePath();
					txtMetadata.setText(path);
				}
			}

		};
	}

	private ActionListener selectIndexListener() {
		return new ActionListener() {
			@Override
			public void actionPerformed(ActionEvent e) {
				selectFile.setFileSelectionMode(JFileChooser.DIRECTORIES_ONLY);
				int opt = selectFile.showOpenDialog(null);
				if (opt != JFileChooser.CANCEL_OPTION) {
					String path = selectFile.getSelectedFile()
							.getAbsolutePath();
					txtIndex.setText(path);
				}
			}

		};
	}

	private ActionListener selectImageListener() {
		return new ActionListener() {
			@Override
			public void actionPerformed(ActionEvent e) {
				selectFile.setFileSelectionMode(JFileChooser.DIRECTORIES_ONLY);
				int opt = selectFile.showOpenDialog(null);
				if (opt != JFileChooser.CANCEL_OPTION) {
					String path = selectFile.getSelectedFile()
							.getAbsolutePath();
```

```
                        txtImage.setText(path);
                    }
                }

        };
    }

    private ActionListener selectResultFolderListener() {
        return new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                selectFile.setFileSelectionMode(JFileChooser.DIRECTORIES_ONLY);
                int opt = selectFile.showOpenDialog(null);
                if (opt != JFileChooser.CANCEL_OPTION) {
                    String path = selectFile.getSelectedFile()
                                .getAbsolutePath();
                    txtPathResult.setText(path);
                }
            }

        };
    }

    private void setActionListeners() {
        btnPathResult.addActionListener(selectResultFolderListener());
        btnImage.addActionListener(selectImageListener());
        btnMetadata.addActionListener(selectMetadataListener());
        btnIndex.addActionListener(selectIndexListener());
    }

    private void initComponents() {

        btnFollow = new javax.swing.ButtonGroup();
        btnMetadataFormtat = new javax.swing.ButtonGroup();
        tabPnl = new javax.swing.JTabbedPane();
        pnlIndex = new javax.swing.JPanel();
        txtMetadata = new javax.swing.JTextField();
        lblMetadata = new javax.swing.JLabel();
        btnMetadata = new javax.swing.JButton();
        btnIndex = new javax.swing.JButton();
```

```
txtIndex = new javax.swing.JTextField();
lblIndex = new javax.swing.JLabel();
btnIndexIt = new javax.swing.JButton();
btnImage = new javax.swing.JButton();
txtImage = new javax.swing.JTextField();
lblImage = new javax.swing.JLabel();
lblFollow = new javax.swing.JLabel();
rbtnFollowYes = new javax.swing.JRadioButton();
rbtnFollowNo = new javax.swing.JRadioButton();
lblIndexResult = new javax.swing.JTextField();
btnReuseIndex = new javax.swing.JButton();
pnlSearch = new javax.swing.JPanel();
tabPnlSearch = new javax.swing.JTabbedPane();
pnlTextMetadata = new javax.swing.JPanel();
pnlAnnFormat = new javax.swing.JPanel();
btnNone = new javax.swing.JRadioButton();
btnXML = new javax.swing.JRadioButton();
btnText = new javax.swing.JRadioButton();
btnXMLText = new javax.swing.JRadioButton();
pnlTagSearch = new javax.swing.JPanel();
lblTag = new javax.swing.JLabel();
txtTag = new javax.swing.JTextField();
btnSelectTag = new javax.swing.JButton();
pnlWordSearch = new javax.swing.JPanel();
txtWord = new javax.swing.JTextField();
lblWord = new javax.swing.JLabel();
boxSimilarity = new javax.swing.JCheckBox();
boxSynonysm = new javax.swing.JCheckBox();
lblPerc = new javax.swing.JLabel();
boxSubstring = new javax.swing.JCheckBox();
lblScale = new javax.swing.JLabel();
boxSimple = new javax.swing.JCheckBox();
lblSimilarity = new javax.swing.JLabel();
txtSimilarity = new javax.swing.JFormattedTextField();
pnlImageMetadata = new javax.swing.JPanel();
pnlImageSearch = new javax.swing.JPanel();
panel1 = new javax.swing.JPanel();
lblWidth = new javax.swing.JLabel();
txtWidthFrom = new javax.swing.JTextField();
lblHeigth = new javax.swing.JLabel();
```

```
txtHeigthFrom = new javax.swing.JTextField();
lblSize = new javax.swing.JLabel();
txtSizeFrom = new javax.swing.JTextField();
txtXResolutionFrom = new javax.swing.JTextField();
lblXResolution = new javax.swing.JLabel();
txtYResolutionFrom = new javax.swing.JTextField();
lblYResolution = new javax.swing.JLabel();
lblCreated = new javax.swing.JLabel();
lblModify = new javax.swing.JLabel();
txtCreatedFrom = new javax.swing.JFormattedTextField();
txtModifyFrom = new javax.swing.JFormattedTextField();
jLabel35 = new javax.swing.JLabel();
txtWidthTo = new javax.swing.JFormattedTextField();
jLabel36 = new javax.swing.JLabel();
jLabel37 = new javax.swing.JLabel();
txtHeigthTo = new javax.swing.JFormattedTextField();
jLabel38 = new javax.swing.JLabel();
jLabel39 = new javax.swing.JLabel();
txtSizeTo = new javax.swing.JFormattedTextField();
jLabel40 = new javax.swing.JLabel();
jLabel41 = new javax.swing.JLabel();
jLabel42 = new javax.swing.JLabel();
txtYResolutionTo = new javax.swing.JFormattedTextField();
txtXResolutionTo = new javax.swing.JFormattedTextField();
jLabel43 = new javax.swing.JLabel();
jLabel44 = new javax.swing.JLabel();
jLabel45 = new javax.swing.JLabel();
jLabel46 = new javax.swing.JLabel();
txtCreatedTo = new javax.swing.JFormattedTextField();
txtModifyTo = new javax.swing.JFormattedTextField();
panel2 = new javax.swing.JPanel();
lblColor = new javax.swing.JLabel();
cmbColor = new javax.swing.JComboBox<String>();
cmbType = new javax.swing.JComboBox<String>();
lblType = new javax.swing.JLabel();
lblCamera = new javax.swing.JLabel();
cmbCamera = new javax.swing.JComboBox<String>();
pnlImageColor = new javax.swing.JPanel();
jPanel1 = new javax.swing.JPanel();
jPanel7 = new javax.swing.JPanel();
```

```
txtYellowFrom = new javax.swing.JFormattedTextField();
txtRedFrom = new javax.swing.JFormattedTextField();
txtPinkFrom = new javax.swing.JFormattedTextField();
txtBlueFrom = new javax.swing.JFormattedTextField();
txtPurpleFrom = new javax.swing.JFormattedTextField();
jLabel13 = new javax.swing.JLabel();
jLabel18 = new javax.swing.JLabel();
jLabel6 = new javax.swing.JLabel();
jLabel19 = new javax.swing.JLabel();
jLabel20 = new javax.swing.JLabel();
jLabel21 = new javax.swing.JLabel();
txtOrangeFrom = new javax.swing.JFormattedTextField();
jLabel1 = new javax.swing.JLabel();
jLabel2 = new javax.swing.JLabel();
jLabel3 = new javax.swing.JLabel();
jLabel4 = new javax.swing.JLabel();
jLabel5 = new javax.swing.JLabel();
jLabel7 = new javax.swing.JLabel();
txtYellowTo = new javax.swing.JFormattedTextField();
txtRedTo = new javax.swing.JFormattedTextField();
txtPinkTo = new javax.swing.JFormattedTextField();
txtBlueTo = new javax.swing.JFormattedTextField();
txtPurpleTo = new javax.swing.JFormattedTextField();
txtOrangeTo = new javax.swing.JFormattedTextField();
jLabel8 = new javax.swing.JLabel();
jLabel10 = new javax.swing.JLabel();
jLabel11 = new javax.swing.JLabel();
jLabel12 = new javax.swing.JLabel();
jLabel14 = new javax.swing.JLabel();
jLabel15 = new javax.swing.JLabel();
jPanel3 = new javax.swing.JPanel();
jLabel9 = new javax.swing.JLabel();
jLabel22 = new javax.swing.JLabel();
jLabel23 = new javax.swing.JLabel();
jLabel24 = new javax.swing.JLabel();
txtWhiteFrom = new javax.swing.JFormattedTextField();
txtBlackFrom = new javax.swing.JFormattedTextField();
txtGreenFrom = new javax.swing.JFormattedTextField();
txtGrayFrom = new javax.swing.JFormattedTextField();
jLabel25 = new javax.swing.JLabel();
```

```
txtBrownFrom = new javax.swing.JFormattedTextField();
jLabel16 = new javax.swing.JLabel();
jLabel17 = new javax.swing.JLabel();
jLabel26 = new javax.swing.JLabel();
jLabel27 = new javax.swing.JLabel();
jLabel28 = new javax.swing.JLabel();
txtBrownTo = new javax.swing.JFormattedTextField();
jLabel29 = new javax.swing.JLabel();
jLabel30 = new javax.swing.JLabel();
txtGrayTo = new javax.swing.JFormattedTextField();
txtGreenTo = new javax.swing.JFormattedTextField();
jLabel31 = new javax.swing.JLabel();
jLabel32 = new javax.swing.JLabel();
txtBlackTo = new javax.swing.JFormattedTextField();
txtWhiteTo = new javax.swing.JFormattedTextField();
jLabel33 = new javax.swing.JLabel();
jLabel34 = new javax.swing.JLabel();
cmbPredominantColor = new javax.swing.JComboBox<String>();
jPanel2 = new javax.swing.JPanel();
jPanel4 = new javax.swing.JPanel();
jLabel47 = new javax.swing.JLabel();
cmbRelSimple = new javax.swing.JComboBox<Integer>();
cmbRelSimilarity = new javax.swing.JComboBox<Integer>();
jLabel48 = new javax.swing.JLabel();
cmbRelSynonysm = new javax.swing.JComboBox<Integer>();
jLabel49 = new javax.swing.JLabel();
cmbRelSubstring = new javax.swing.JComboBox<Integer>();
jLabel50 = new javax.swing.JLabel();
jPanel5 = new javax.swing.JPanel();
jLabel51 = new javax.swing.JLabel();
cmbRelWidth = new javax.swing.JComboBox<Integer>();
cmbRelHeight = new javax.swing.JComboBox<Integer>();
jLabel52 = new javax.swing.JLabel();
cmbRelSize = new javax.swing.JComboBox<Integer>();
jLabel53 = new javax.swing.JLabel();
cmbRelXResol = new javax.swing.JComboBox<Integer>();
jLabel54 = new javax.swing.JLabel();
jLabel55 = new javax.swing.JLabel();
cmbRelYResol = new javax.swing.JComboBox<Integer>();
jLabel56 = new javax.swing.JLabel();
```

```
jLabel57 = new javax.swing.JLabel();
cmbRelCreated = new javax.swing.JComboBox<Integer>();
cmbRelModif = new javax.swing.JComboBox<Integer>();
jLabel58 = new javax.swing.JLabel();
cmbRelColorType = new javax.swing.JComboBox<Integer>();
cmbRelFormat = new javax.swing.JComboBox<Integer>();
jLabel59 = new javax.swing.JLabel();
jLabel60 = new javax.swing.JLabel();
cmbRelCamera = new javax.swing.JComboBox<Integer>();
jPanel6 = new javax.swing.JPanel();
jLabel61 = new javax.swing.JLabel();
jLabel62 = new javax.swing.JLabel();
jLabel63 = new javax.swing.JLabel();
jLabel64 = new javax.swing.JLabel();
jLabel65 = new javax.swing.JLabel();
jLabel66 = new javax.swing.JLabel();
cmbRelBrown = new javax.swing.JComboBox<Integer>();
cmbRelGray = new javax.swing.JComboBox<Integer>();
cmbRelGreen = new javax.swing.JComboBox<Integer>();
cmbRelBlack = new javax.swing.JComboBox<Integer>();
cmbRelWhite = new javax.swing.JComboBox<Integer>();
cmbRelPredominant = new javax.swing.JComboBox<Integer>();
cmbRelPink = new javax.swing.JComboBox<Integer>();
cmbRelRed = new javax.swing.JComboBox<Integer>();
cmbRelYellow = new javax.swing.JComboBox<Integer>();
jLabel67 = new javax.swing.JLabel();
jLabel68 = new javax.swing.JLabel();
jLabel69 = new javax.swing.JLabel();
jLabel70 = new javax.swing.JLabel();
jLabel71 = new javax.swing.JLabel();
cmbRelBlue = new javax.swing.JComboBox<Integer>();
cmbRelPurple = new javax.swing.JComboBox<Integer>();
cmbRelOrange = new javax.swing.JComboBox<Integer>();
jLabel72 = new javax.swing.JLabel();
btnSearch = new javax.swing.JButton();
lblNumberResults = new javax.swing.JLabel();
txtNumberResults = new javax.swing.JFormattedTextField();
lblPathResult = new javax.swing.JLabel();
txtPathResult = new javax.swing.JTextField();
btnPathResult = new javax.swing.JButton();
```

```
setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
setTitle("Metadata Ingester");

txtMetadata.setEditable(false);

lblMetadata.setText("Selecione a pasta que contém os metadados");

btnMetadata.setText("Selecionar...");

btnIndex.setText("Selecionar...");

txtIndex.setEditable(false);

lblIndex.setText("Selecione a pasta para armazenar os índices");

btnIndexIt.setText("Indexar!");

btnImage.setText("Selecionar...");

txtImage.setEditable(false);

lblImage.setText("Selecione a pasta que contém as imagens");

lblFollow.setText("Indexar Sub-Diretórios:");

btnFollow.add(rbtnFollowYes);
rbtnFollowYes.setSelected(true);
rbtnFollowYes.setText("Sim");

btnFollow.add(rbtnFollowNo);
rbtnFollowNo.setText("Não");

lblIndexResult.setBackground(new java.awt.Color(240, 240, 240));
lblIndexResult.setColumns(10);
lblIndexResult.setHorizontalAlignment(javax.swing.JTextField.CENTER);

btnReuseIndex.setText("Restaurar Índices!");

javax.swing.GroupLayout pnlIndexLayout = new javax.swing.GroupLayout(
```

```
                pnlIndex);
        pnlIndex.setLayout(pnlIndexLayout);
        pnlIndexLayout
                .setHorizontalGroup(pnlIndexLayout
                        .createParallelGroup(
                                javax.swing.GroupLayout.Alignment.LEADING)
                        .addGroup(
                                javax.swing.GroupLayout.Alignment.TRAILING,
                                pnlIndexLayout
                                        .createSequentialGroup()
                                        .addGroup(
                                                pnlIndexLayout
                                                        .createParallelGroup(

javax.swing.GroupLayout.Alignment.TRAILING)
                                                        .addGroup(
                                                                pnlIndexLayout

.createSequentialGroup()

.addContainerGap()

.addComponent(

        btnReuseIndex)

.addPreferredGap(

        javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)

.addComponent(

        btnIndexIt)

.addGap(18,

        18,

        18)
```

```
.addComponent(
        lblIndexResult,
        javax.swing.GroupLayout.PREFERRED_SIZE,
        131,
        javax.swing.GroupLayout.PREFERRED_SIZE))
                                                                    .addGroup(
                                                                            pnlIndexLayout
.createSequentialGroup()
.addGap(96,
        96,
        96)
.addGroup(
        pnlIndexLayout
                .createParallelGroup(
                        javax.swing.GroupLayout.Alignment.LEADING)
                .addComponent(
                        txtMetadata,
                        javax.swing.GroupLayout.DEFAULT_SIZE,
                        353,
                        Short.MAX_VALUE)
                .addGroup(
```

```
                pnlIndexLayout
                        .createSequentialGroup()
                        .addComponent(
                                lblMetadata)
                        .addPreferredGap(
                                javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                        .addComponent(
                                btnMetadata,
                                javax.swing.GroupLayout.DEFAULT_SIZE,
                                136,
                                Short.MAX_VALUE))
        .addGroup(
                pnlIndexLayout
                        .createSequentialGroup()
                        .addComponent(
                                lblImage)
                        .addGap(18,
                                18,
                                18)
                        .addComponent(
```

```
                                        btnImage,
                                        javax.swing.GroupLayout.DEFAULT_SIZE,
                                        136,
                                        Short.MAX_VALUE))
                .addComponent(
                        txtIndex,
                        javax.swing.GroupLayout.DEFAULT_SIZE,
                        353,
                        Short.MAX_VALUE)
                .addGroup(
                        pnlIndexLayout
                                .createSequentialGroup()
                                .addComponent(
                                        lblIndex)
                                .addPreferredGap(
                                        javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                                .addComponent(
                                        btnIndex,
                                        javax.swing.GroupLayout.DEFAULT_SIZE,
                                        137,
```

```
                                                        Short.MAX_VALUE))
                        .addComponent(
                                        txtImage,
                                        javax.swing.GroupLayout.Alignment.TRAILING,
                                        javax.swing.GroupLayout.PREFERRED_SIZE,
                                        353,
                                        javax.swing.GroupLayout.PREFERRED_SIZE))))
                                                        .addGap(126, 126, 126))
                                        .addGroup(
                                                        pnlIndexLayout.createSequentialGroup()
                                                        .addGap(136, 136, 136)
                                                        .addComponent(lblFollow)
                                                        .addGap(32, 32, 32)
                                                        .addComponent(rbtnFollowYes)
                                                        .addGap(18, 18, 18)
                                                        .addComponent(rbtnFollowNo)
                                                        .addContainerGap(146, Short.MAX_VALUE)));
        pnlIndexLayout
                        .setVerticalGroup(pnlIndexLayout
                                        .createParallelGroup(
                                                        javax.swing.GroupLayout.Alignment.LEADING)
                                        .addGroup(
                                                        pnlIndexLayout
                                                        .createSequentialGroup()
                                                        .addGap(58, 58, 58)
                                                        .addGroup(
                                                                        pnlIndexLayout
                                                                        .createParallelGroup(
javax.swing.GroupLayout.Alignment.BASELINE)
                                                                        .addComponent(lblImage)
                                                                        .addComponent(btnImage))
                                                        .addPreferredGap(
```

```
javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                                                .addComponent(
                                                        txtImage,
                                                        javax.swing.GroupLayout.PREFERRED_SIZE,
                                                        javax.swing.GroupLayout.DEFAULT_SIZE,
                                                        javax.swing.GroupLayout.PREFERRED_SIZE)
                                                .addGap(18, 18, 18)
                                                .addGroup(
                                                        pnlIndexLayout
                                                                .createParallelGroup(

javax.swing.GroupLayout.Alignment.BASELINE)
                                                                .addComponent(
                                                                        lblMetadata)
                                                                .addComponent(
                                                                        btnMetadata))
                                                .addPreferredGap(

javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                                                .addComponent(
                                                        txtMetadata,
                                                        javax.swing.GroupLayout.PREFERRED_SIZE,
                                                        javax.swing.GroupLayout.DEFAULT_SIZE,
                                                        javax.swing.GroupLayout.PREFERRED_SIZE)
                                                .addGap(18, 18, 18)
                                                .addGroup(
                                                        pnlIndexLayout
                                                                .createParallelGroup(

javax.swing.GroupLayout.Alignment.BASELINE)
                                                                .addComponent(lblIndex)
                                                                .addComponent(btnIndex))
                                                .addPreferredGap(

javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                                                .addComponent(
                                                        txtIndex,
                                                        javax.swing.GroupLayout.PREFERRED_SIZE,
                                                        javax.swing.GroupLayout.DEFAULT_SIZE,
```

```
                                                        javax.swing.GroupLayout.PREFERRED_SIZE)
                                                .addGap(51, 51, 51)
                                                .addGroup(
                                                        pnlIndexLayout
                                                                .createParallelGroup(
javax.swing.GroupLayout.Alignment.BASELINE)
                                                                .addComponent(lblFollow)
                                                                .addComponent(
                                                                        rbtnFollowNo)
                                                                .addComponent(
                                                                        rbtnFollowYes))
                                                .addPreferredGap(
javax.swing.LayoutStyle.ComponentPlacement.RELATED,
                                                        68, Short.MAX_VALUE)
                                                .addGroup(
                                                        pnlIndexLayout
                                                                .createParallelGroup(
javax.swing.GroupLayout.Alignment.BASELINE)
                                                                .addComponent(
                                                                        lblIndexResult,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
                                                                .addComponent(
                                                                        btnIndexIt)
                                                                .addComponent(
                                                                        btnReuseIndex))
                                                .addGap(41, 41, 41)));

        tabPnl.addTab("        Indexação      ", pnlIndex);

        pnlAnnFormat.setBorder(javax.swing.BorderFactory
                        .createTitledBorder("Selecione o Formato do Metadado"));
```

```
        btnMetadataFormtat.add(btnNone);
        btnNone.setSelected(true);
        btnNone.setText("Nenhum");

        btnMetadataFormtat.add(btnXML);
        btnXML.setText("XML");

        btnMetadataFormtat.add(btnText);
        btnText.setText("Texto");

        btnMetadataFormtat.add(btnXMLText);
        btnXMLText.setText("XML/Texto");

        javax.swing.GroupLayout pnlAnnFormatLayout = new javax.swing.GroupLayout(
                    pnlAnnFormat);
        pnlAnnFormat.setLayout(pnlAnnFormatLayout);
        pnlAnnFormatLayout
                    .setHorizontalGroup(pnlAnnFormatLayout
                                .createParallelGroup(
                                            javax.swing.GroupLayout.Alignment.LEADING)
                                .addGroup(
                                            pnlAnnFormatLayout
                                                        .createSequentialGroup()
                                                        .addContainerGap()
                                                        .addGroup(
                                                                    pnlAnnFormatLayout
                                                                                .createParallelGroup(

javax.swing.GroupLayout.Alignment.LEADING)
                                                                                .addComponent(btnNone)
                                                                                .addComponent(btnText))
                                                        .addGap(18, 18, 18)
                                                        .addGroup(
                                                                    pnlAnnFormatLayout
                                                                                .createParallelGroup(

javax.swing.GroupLayout.Alignment.LEADING)
                                                                                .addComponent(
                                                                                            btnXMLText)
                                                                                .addComponent(btnXML))
```

```
                                                        .addContainerGap(10, Short.MAX_VALUE)));
        pnlAnnFormatLayout
                    .setVerticalGroup(pnlAnnFormatLayout
                                .createParallelGroup(
                                            javax.swing.GroupLayout.Alignment.LEADING)
                                .addGroup(
                                            pnlAnnFormatLayout
                                                        .createSequentialGroup()
                                                        .addContainerGap()
                                                        .addGroup(
                                                                    pnlAnnFormatLayout
                                                                                .createParallelGroup(

javax.swing.GroupLayout.Alignment.BASELINE)
                                                                                .addComponent(btnNone)
                                                                                .addComponent(btnXML))
                                                        .addPreferredGap(

javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
                                                        .addGroup(
                                                                    pnlAnnFormatLayout
                                                                                .createParallelGroup(

javax.swing.GroupLayout.Alignment.BASELINE)
                                                                                .addComponent(btnText)
                                                                                .addComponent(
                                                                                            btnXMLText))
                                                        .addContainerGap(15, Short.MAX_VALUE)));

        pnlTagSearch.setBorder(javax.swing.BorderFactory
                    .createTitledBorder("Selecione a Tag do XML: "));

        lblTag.setText("Tag do XML:");

        txtTag.setColumns(15);

        btnSelectTag.setText("Buscar Tag...");

        javax.swing.GroupLayout pnlTagSearchLayout = new javax.swing.GroupLayout(
                    pnlTagSearch);
```

```
        pnlTagSearch.setLayout(pnlTagSearchLayout);
        pnlTagSearchLayout
                    .setHorizontalGroup(pnlTagSearchLayout
                                .createParallelGroup(
                                            javax.swing.GroupLayout.Alignment.LEADING)
                                .addGroup(
                                            pnlTagSearchLayout
                                                        .createSequentialGroup()
                                                        .addGap(4, 4, 4)
                                                        .addGroup(
                                                                    pnlTagSearchLayout
                                                                                .createParallelGroup(

javax.swing.GroupLayout.Alignment.TRAILING)
                                                                                .addComponent(
                                                                                            btnSelectTag)
                                                                                .addGroup(


javax.swing.GroupLayout.Alignment.LEADING,

pnlTagSearchLayout

.createSequentialGroup()

.addComponent(

        lblTag)

.addPreferredGap(

        javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addComponent(

        txtTag,

        javax.swing.GroupLayout.PREFERRED_SIZE,

        187,
```

```
        javax.swing.GroupLayout.PREFERRED_SIZE)))
                                                        .addContainerGap()));
        pnlTagSearchLayout
                    .setVerticalGroup(pnlTagSearchLayout
                                .createParallelGroup(
                                            javax.swing.GroupLayout.Alignment.LEADING)
                                .addGroup(
                                            pnlTagSearchLayout
                                                        .createSequentialGroup()
                                                        .addGap(10, 10, 10)
                                                        .addGroup(
                                                                    pnlTagSearchLayout
                                                                                .createParallelGroup(

javax.swing.GroupLayout.Alignment.LEADING)
                                                                                .addComponent(lblTag)
                                                                                .addComponent(
                                                                                            txtTag,

javax.swing.GroupLayout.PREFERRED_SIZE,

javax.swing.GroupLayout.DEFAULT_SIZE,

javax.swing.GroupLayout.PREFERRED_SIZE))
                                                        .addPreferredGap(

javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                                                        .addComponent(btnSelectTag)
                                                        .addContainerGap(12, Short.MAX_VALUE)));

        pnlWordSearch.setBorder(javax.swing.BorderFactory
                    .createTitledBorder("Configure a Busca da Palavra:"));

        lblWord.setText("Palavra:");

        boxSimilarity
                    .setText("Busca por Similaridade (Utiliza a porcentagem de similaridade)");

        boxSynonysm
```

```
                .setText("Busca por Sinônimos (Utiliza apenas os sinônimos informados pelo WordNet)");

        lblPerc.setText("%");

        boxSubstring
                .setText("Busca por Substring (Utiliza os caracteres informados como parte da palavra)");

        lblScale.setText("(Escala de 0 à 100)");

        boxSimple
                .setText("Busca Simples (Busca apenas pelos caracteres informados)");

        lblSimilarity.setText("Porcentagem de Similaridade:");

        txtSimilarity
                .setFormatterFactory(new javax.swing.text.DefaultFormatterFactory(
                        new javax.swing.text.NumberFormatter(
                                new java.text.DecimalFormat("#0"))));
        txtSimilarity.setText("70");

        javax.swing.GroupLayout pnlWordSearchLayout = new javax.swing.GroupLayout(
                pnlWordSearch);
        pnlWordSearch.setLayout(pnlWordSearchLayout);
        pnlWordSearchLayout
                .setHorizontalGroup(pnlWordSearchLayout
                        .createParallelGroup(
                                javax.swing.GroupLayout.Alignment.LEADING)
                        .addGroup(
                                pnlWordSearchLayout
                                        .createSequentialGroup()
                                        .addGap(15, 15, 15)
                                        .addGroup(
                                                pnlWordSearchLayout
                                                        .createParallelGroup(

javax.swing.GroupLayout.Alignment.TRAILING)
                                                        .addGroup(

javax.swing.GroupLayout.Alignment.LEADING,
```

```
pnlWordSearchLayout
.createParallelGroup(
      javax.swing.GroupLayout.Alignment.LEADING)
.addComponent(
      boxSimple)
.addGroup(
      pnlWordSearchLayout
                  .createSequentialGroup()
                  .addComponent(
                              lblWord)
                  .addPreferredGap(
                              javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                  .addComponent(
                              txtWord,
                              javax.swing.GroupLayout.PREFERRED_SIZE,
                              208,
                              javax.swing.GroupLayout.PREFERRED_SIZE))
.addComponent(
      boxSimilarity)
.addGroup(
```

```
javax.swing.GroupLayout.Alignment.TRAILING,
pnlWordSearchLayout
                .createSequentialGroup()
                .addGroup(
                                pnlWordSearchLayout
                                                .createParallelGroup(
                                                                javax.swing.GroupLayout.Alignment.TRAILING,
                                                                false)
                                                .addComponent(
                                                                boxSynonysm,
                                                                javax.swing.GroupLayout.Alignment.LEADING,
                                                                javax.swing.GroupLayout.DEFAULT_SIZE,
                                                                javax.swing.GroupLayout.DEFAULT_SIZE,
                                                                Short.MAX_VALUE)
                                                .addComponent(
                                                                boxSubstring,
                                                                javax.swing.GroupLayout.Alignment.LEADING))
                .addGap(47,
                                47,
                                47)))
```

```
                                                                        .addGroup(
javax.swing.GroupLayout.Alignment.LEADING,
pnlWordSearchLayout
.createSequentialGroup()
.addGap(146,
      146,
      146)
.addComponent(
      lblSimilarity)
.addPreferredGap(
      javax.swing.LayoutStyle.ComponentPlacement.RELATED)
.addGroup(
      pnlWordSearchLayout
                  .createParallelGroup(
                              javax.swing.GroupLayout.Alignment.LEADING)
                  .addGroup(
                              javax.swing.GroupLayout.Alignment.TRAILING,
                              pnlWordSearchLayout
                                          .createSequentialGroup()
                                          .addComponent(
```

```
                                                        txtSimilarity,
                                                        javax.swing.GroupLayout.PREFERRED_SIZE,
                                                        26,
                                                        javax.swing.GroupLayout.PREFERRED_SIZE)
                                                .addGap(15,
                                                        15,
                                                        15))
                                .addComponent(
                                        lblPerc,
                                        javax.swing.GroupLayout.Alignment.TRAILING))
        .addGap(18,
                18,
                18)
        .addComponent(
                lblScale)))
                                                        .addContainerGap(15, Short.MAX_VALUE)));
                pnlWordSearchLayout
                                .setVerticalGroup(pnlWordSearchLayout
                                        .createParallelGroup(
                                                javax.swing.GroupLayout.Alignment.LEADING)
                                        .addGroup(
                                                javax.swing.GroupLayout.Alignment.TRAILING,
                                                pnlWordSearchLayout
                                                        .createSequentialGroup()
                                                        .addContainerGap(
```

```
                                        javax.swing.GroupLayout.DEFAULT_SIZE,
                                        Short.MAX_VALUE)
                                .addGroup(
                                        pnlWordSearchLayout
                                                .createParallelGroup(
javax.swing.GroupLayout.Alignment.BASELINE)
                                                .addComponent(
                                                        txtWord,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
                                                .addComponent(lblWord))
                                .addPreferredGap(
javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
                                .addComponent(boxSimple)
                                .addPreferredGap(
javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
                                .addComponent(boxSynonysm)
                                .addGap(3, 3, 3)
                                .addComponent(boxSubstring)
                                .addPreferredGap(
javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
                                .addComponent(boxSimilarity)
                                .addPreferredGap(
javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                                .addGroup(
                                        pnlWordSearchLayout
                                                .createParallelGroup(
javax.swing.GroupLayout.Alignment.BASELINE)
                                                .addComponent(lblPerc)
                                                .addComponent(
```

```
                                                                txtSimilarity,

javax.swing.GroupLayout.PREFERRED_SIZE,

javax.swing.GroupLayout.DEFAULT_SIZE,

javax.swing.GroupLayout.PREFERRED_SIZE)
                                                            .addComponent(
                                                                    lblSimilarity)
                                                            .addComponent(
                                                                    lblScale,

javax.swing.GroupLayout.PREFERRED_SIZE,
                                                                    17,

javax.swing.GroupLayout.PREFERRED_SIZE))
                                                    .addGap(5, 5, 5)));

        javax.swing.GroupLayout pnlTextMetadataLayout = new javax.swing.GroupLayout(
                    pnlTextMetadata);
        pnlTextMetadata.setLayout(pnlTextMetadataLayout);
        pnlTextMetadataLayout
                    .setHorizontalGroup(pnlTextMetadataLayout
                                .createParallelGroup(
                                            javax.swing.GroupLayout.Alignment.LEADING)
                                .addGroup(
                                            javax.swing.GroupLayout.Alignment.TRAILING,
                                            pnlTextMetadataLayout
                                                        .createSequentialGroup()
                                                        .addContainerGap()
                                                        .addGroup(
                                                                    pnlTextMetadataLayout
                                                                            .createParallelGroup(

javax.swing.GroupLayout.Alignment.TRAILING)
                                                                            .addComponent(
                                                                                    pnlWordSearch,

javax.swing.GroupLayout.Alignment.LEADING,
```

```
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
                                                                    .addGroup(
pnlTextMetadataLayout
.createSequentialGroup()
.addComponent(
      pnlAnnFormat,
      javax.swing.GroupLayout.PREFERRED_SIZE,
      javax.swing.GroupLayout.DEFAULT_SIZE,
      javax.swing.GroupLayout.PREFERRED_SIZE)
.addPreferredGap(
      javax.swing.LayoutStyle.ComponentPlacement.RELATED)
.addComponent(
      pnlTagSearch,
      javax.swing.GroupLayout.DEFAULT_SIZE,
      290,
      Short.MAX_VALUE)))
                                                        .addContainerGap()));
      pnlTextMetadataLayout
                  .setVerticalGroup(pnlTextMetadataLayout
                              .createParallelGroup(
                                          javax.swing.GroupLayout.Alignment.LEADING)
```

```
                                .addGroup(
                                        pnlTextMetadataLayout
                                                .createSequentialGroup()
                                                .addContainerGap()
                                                .addGroup(
                                                        pnlTextMetadataLayout
                                                                .createParallelGroup(
javax.swing.GroupLayout.Alignment.LEADING,
                                                                        false)
                                                                .addComponent(
                                                                        pnlAnnFormat,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
                                                                .addComponent(
                                                                        pnlTagSearch,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE))
                                                .addPreferredGap(
javax.swing.LayoutStyle.ComponentPlacement.RELATED,
                                                        javax.swing.GroupLayout.DEFAULT_SIZE,
                                                        Short.MAX_VALUE)
                                                .addComponent(
                                                        pnlWordSearch,
                                                        javax.swing.GroupLayout.PREFERRED_SIZE,
                                                        javax.swing.GroupLayout.DEFAULT_SIZE,
                                                        javax.swing.GroupLayout.PREFERRED_SIZE)
                                                .addGap(7, 7, 7)));

        tabPnlSearch.addTab("   Metadado Textual   ", pnlTextMetadata);
```

```
pnlImageSearch.setBorder(javax.swing.BorderFactory
            .createTitledBorder("Informações para Busca"));

lblWidth.setText("Largura: de");

txtWidthFrom.setColumns(5);

lblHeigth.setText("Altura: de");

txtHeigthFrom.setColumns(5);

lblSize.setText("Tamanho: de");

txtSizeFrom.setColumns(5);

txtXResolutionFrom.setColumns(5);

lblXResolution.setText("Resolução em X: de");

txtYResolutionFrom.setColumns(5);

lblYResolution.setText("Resolução em Y: de");

lblCreated.setText("Data da Criação: de");

lblModify.setText("Última Modificação: de");

txtCreatedFrom.setColumns(10);
try {
      txtCreatedFrom
                  .setFormatterFactory(new javax.swing.text.DefaultFormatterFactory(
                              new javax.swing.text.MaskFormatter("##/##/####")));
} catch (java.text.ParseException ex) {
      ex.printStackTrace();
}

try {
      txtModifyFrom
                  .setFormatterFactory(new javax.swing.text.DefaultFormatterFactory(
                              new javax.swing.text.MaskFormatter("##/##/####")));
```

```
} catch (java.text.ParseException ex) {
      ex.printStackTrace();
}

jLabel35.setText("px  até");

txtWidthTo.setColumns(5);

jLabel36.setText("px");

jLabel37.setText("px");

txtHeigthTo.setColumns(5);

jLabel38.setText("px  até");

jLabel39.setText("KB  até");

txtSizeTo.setColumns(5);

jLabel40.setText("KB");

jLabel41.setText("px  até");

jLabel42.setText("px  até");

txtYResolutionTo.setColumns(5);

txtXResolutionTo.setColumns(5);

jLabel43.setText("px");

jLabel44.setText("px");

jLabel45.setText("até");

jLabel46.setText(" até");

txtCreatedTo.setColumns(10);
try {
```

```
            txtCreatedTo
                        .setFormatterFactory(new javax.swing.text.DefaultFormatterFactory(
                                    new javax.swing.text.MaskFormatter("##/##/####")));
      } catch (java.text.ParseException ex) {
            ex.printStackTrace();
      }

      try {
            txtModifyTo
                        .setFormatterFactory(new javax.swing.text.DefaultFormatterFactory(
                                    new javax.swing.text.MaskFormatter("##/##/####")));
      } catch (java.text.ParseException ex) {
            ex.printStackTrace();
      }

      javax.swing.GroupLayout panel1Layout = new javax.swing.GroupLayout(
                  panel1);
      panel1.setLayout(panel1Layout);
      panel1Layout
                  .setHorizontalGroup(panel1Layout
                              .createParallelGroup(
                                          javax.swing.GroupLayout.Alignment.LEADING)
                              .addGroup(
                                          panel1Layout
                                                      .createSequentialGroup()
                                                      .addContainerGap()
                                                      .addGroup(
                                                                  panel1Layout
                                                                              .createParallelGroup(

javax.swing.GroupLayout.Alignment.TRAILING)
                                                                              .addGroup(
                                                                                          panel1Layout

.createSequentialGroup()

.addComponent(

      lblModify)
```

```
.addPreferredGap(
      javax.swing.LayoutStyle.ComponentPlacement.RELATED)
.addComponent(
      txtModifyFrom,
      javax.swing.GroupLayout.PREFERRED_SIZE,
      49,
      javax.swing.GroupLayout.PREFERRED_SIZE))
                                                                  .addGroup(
                                                                        panel1Layout
.createSequentialGroup()
.addComponent(
      lblCreated)
.addPreferredGap(
      javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
.addComponent(
      txtCreatedFrom,
      javax.swing.GroupLayout.PREFERRED_SIZE,
      49,
      javax.swing.GroupLayout.PREFERRED_SIZE))
                                                                  .addGroup(
                                                                        panel1Layout
.createSequentialGroup()
```

```
.addComponent(
        lblSize)
.addGap(10,
        10,
        10)
.addComponent(
        txtSizeFrom,
        javax.swing.GroupLayout.PREFERRED_SIZE,
        javax.swing.GroupLayout.DEFAULT_SIZE,
        javax.swing.GroupLayout.PREFERRED_SIZE))
                                                                    .addGroup(
                                                                            panel1Layout
.createSequentialGroup()
.addComponent(
        lblHeigth)
.addPreferredGap(
        javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
.addComponent(
        txtHeigthFrom,
        javax.swing.GroupLayout.PREFERRED_SIZE,
        javax.swing.GroupLayout.DEFAULT_SIZE,
```

```
        javax.swing.GroupLayout.PREFERRED_SIZE))
                                                                .addGroup(
                                                                        panel1Layout
.createSequentialGroup()
.addComponent(
        lblWidth)
.addPreferredGap(
        javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
.addComponent(
        txtWidthFrom,
        javax.swing.GroupLayout.PREFERRED_SIZE,
        javax.swing.GroupLayout.DEFAULT_SIZE,
        javax.swing.GroupLayout.PREFERRED_SIZE))
                                                                .addGroup(
                                                                        panel1Layout
.createSequentialGroup()
.addComponent(
        lblYResolution)
.addGap(11,
        11,
        11)
.addComponent(
```

```
        txtYResolutionFrom,
        javax.swing.GroupLayout.PREFERRED_SIZE,
        javax.swing.GroupLayout.DEFAULT_SIZE,
        javax.swing.GroupLayout.PREFERRED_SIZE))
                                                                .addGroup(
                                                                        panel1Layout
.createSequentialGroup()
.addComponent(
        lblXResolution)
.addGap(11,
        11,
        11)
.addComponent(
        txtXResolutionFrom,
        javax.swing.GroupLayout.PREFERRED_SIZE,
        javax.swing.GroupLayout.DEFAULT_SIZE,
        javax.swing.GroupLayout.PREFERRED_SIZE)))
                                                        .addPreferredGap(
javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                                                        .addGroup(
                                                                panel1Layout
                                                                        .createParallelGroup(
javax.swing.GroupLayout.Alignment.LEADING)
```

```
                                                                .addGroup(
                                                                        panel1Layout
.createSequentialGroup()
.addGroup(
      panel1Layout
                  .createParallelGroup(
                              javax.swing.GroupLayout.Alignment.LEADING)
                  .addComponent(
                              jLabel35)
                  .addComponent(
                              jLabel38))
.addPreferredGap(
      javax.swing.LayoutStyle.ComponentPlacement.RELATED)
.addGroup(
      panel1Layout
                  .createParallelGroup(
                              javax.swing.GroupLayout.Alignment.LEADING)
                  .addGroup(
                              panel1Layout
                                          .createSequentialGroup()
                                          .addComponent(
```

```
                                        txtWidthTo,
                                        javax.swing.GroupLayout.PREFERRED_SIZE,
                                        javax.swing.GroupLayout.DEFAULT_SIZE,
                                        javax.swing.GroupLayout.PREFERRED_SIZE)
                                .addPreferredGap(
                                        javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                                .addComponent(
                                        jLabel36))
                .addGroup(
                        panel1Layout
                                .createSequentialGroup()
                                .addComponent(
                                        txtHeigthTo,
                                        javax.swing.GroupLayout.PREFERRED_SIZE,
                                        javax.swing.GroupLayout.DEFAULT_SIZE,
                                        javax.swing.GroupLayout.PREFERRED_SIZE)
                                .addPreferredGap(
                                        javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                                .addComponent(
                                        jLabel37))))
```

```
                                                                .addGroup(
                                                                        panel1Layout
.createSequentialGroup()
.addComponent(
      jLabel39)
.addPreferredGap(
      javax.swing.LayoutStyle.ComponentPlacement.RELATED)
.addComponent(
      txtSizeTo,
      javax.swing.GroupLayout.PREFERRED_SIZE,
      javax.swing.GroupLayout.DEFAULT_SIZE,
      javax.swing.GroupLayout.PREFERRED_SIZE)
.addPreferredGap(
      javax.swing.LayoutStyle.ComponentPlacement.RELATED)
.addComponent(
      jLabel40))
                                                                .addGroup(
                                                                        panel1Layout
.createSequentialGroup()
.addGroup(
      panel1Layout
                  .createParallelGroup(
```

```
                                javax.swing.GroupLayout.Alignment.LEADING)
                        .addComponent(
                                jLabel41)
                        .addComponent(
                                jLabel42))
        .addPreferredGap(
                javax.swing.LayoutStyle.ComponentPlacement.RELATED)
        .addGroup(
                panel1Layout
                        .createParallelGroup(
                                javax.swing.GroupLayout.Alignment.LEADING)
                        .addGroup(
                                panel1Layout
                                        .createSequentialGroup()
                                        .addComponent(
                                                txtXResolutionTo,
                                                javax.swing.GroupLayout.PREFERRED_SIZE,
                                                javax.swing.GroupLayout.DEFAULT_SIZE,
                                                javax.swing.GroupLayout.PREFERRED_SIZE)
                                        .addPreferredGap(
```

```
                                        javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                                .addComponent(
                                        jLabel43))
                        .addGroup(
                                panel1Layout
                                .createSequentialGroup()
                                .addComponent(
                                        txtYResolutionTo,
                                        javax.swing.GroupLayout.PREFERRED_SIZE,
                                        javax.swing.GroupLayout.DEFAULT_SIZE,
                                        javax.swing.GroupLayout.PREFERRED_SIZE)
                                .addPreferredGap(
                                        javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                                .addComponent(
                                        jLabel44))))
                                                        .addGroup(
                                                                panel1Layout
.createSequentialGroup()
.addGap(10,
       10,
       10)
```

```
.addGroup(
        panel1Layout
                        .createParallelGroup(
                                        javax.swing.GroupLayout.Alignment.TRAILING)
                        .addComponent(
                                        jLabel45)
                        .addComponent(
                                        jLabel46))
.addPreferredGap(
        javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
.addGroup(
        panel1Layout
                        .createParallelGroup(
                                        javax.swing.GroupLayout.Alignment.LEADING)
                        .addComponent(
                                        txtModifyTo,
                                        javax.swing.GroupLayout.PREFERRED_SIZE,
                                        49,
                                        javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addComponent(
```

```
                                txtCreatedTo,

                                javax.swing.GroupLayout.PREFERRED_SIZE,

                                49,

                                javax.swing.GroupLayout.PREFERRED_SIZE))))
                                                .addContainerGap(
                                                                javax.swing.GroupLayout.DEFAULT_SIZE,
                                                                Short.MAX_VALUE)));
        panel1Layout
                        .setVerticalGroup(panel1Layout
                                        .createParallelGroup(
                                                        javax.swing.GroupLayout.Alignment.LEADING)
                                        .addGroup(
                                                        panel1Layout
                                                                        .createSequentialGroup()
                                                                        .addContainerGap()
                                                                        .addGroup(
                                                                                        panel1Layout
                                                                                                        .createParallelGroup(

javax.swing.GroupLayout.Alignment.BASELINE)
                                                                                                        .addComponent(
                                                                                                                        txtWidthFrom,

javax.swing.GroupLayout.PREFERRED_SIZE,

javax.swing.GroupLayout.DEFAULT_SIZE,

javax.swing.GroupLayout.PREFERRED_SIZE)
                                                                                                        .addComponent(lblWidth)
                                                                                                        .addComponent(jLabel35)
                                                                                                        .addComponent(
                                                                                                                        txtWidthTo,

javax.swing.GroupLayout.PREFERRED_SIZE,

javax.swing.GroupLayout.DEFAULT_SIZE,
```

```
javax.swing.GroupLayout.PREFERRED_SIZE)
                                                        .addComponent(jLabel36))
                                                .addPreferredGap(

javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                                                .addGroup(
                                                        panel1Layout
                                                        .createParallelGroup(

javax.swing.GroupLayout.Alignment.BASELINE)
                                                        .addComponent(
                                                                txtHeigthFrom,

javax.swing.GroupLayout.PREFERRED_SIZE,

javax.swing.GroupLayout.DEFAULT_SIZE,

javax.swing.GroupLayout.PREFERRED_SIZE)
                                                        .addComponent(lblHeigth)
                                                        .addComponent(
                                                                txtHeigthTo,

javax.swing.GroupLayout.PREFERRED_SIZE,

javax.swing.GroupLayout.DEFAULT_SIZE,

javax.swing.GroupLayout.PREFERRED_SIZE)
                                                        .addComponent(jLabel37)
                                                        .addComponent(jLabel38))
                                                .addPreferredGap(

javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                                                .addGroup(
                                                        panel1Layout
                                                        .createParallelGroup(

javax.swing.GroupLayout.Alignment.BASELINE)
                                                        .addComponent(
                                                                txtSizeFrom,
```

```
javax.swing.GroupLayout.PREFERRED_SIZE,

javax.swing.GroupLayout.DEFAULT_SIZE,

javax.swing.GroupLayout.PREFERRED_SIZE)
                                                                    .addComponent(lblSize)
                                                                    .addComponent(
                                                                                txtSizeTo,

javax.swing.GroupLayout.PREFERRED_SIZE,

javax.swing.GroupLayout.DEFAULT_SIZE,

javax.swing.GroupLayout.PREFERRED_SIZE)
                                                                    .addComponent(jLabel40)
                                                                    .addComponent(jLabel39))
                                                        .addPreferredGap(

javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                                                        .addGroup(
                                                                    panel1Layout
                                                                    .createParallelGroup(

javax.swing.GroupLayout.Alignment.LEADING)
                                                                    .addGroup(
                                                                                panel1Layout

.createSequentialGroup()

.addGroup(

      panel1Layout

                  .createParallelGroup(

                              javax.swing.GroupLayout.Alignment.LEADING)

                  .addGroup(
```

```
                                panel1Layout
                                        .createSequentialGroup()
                                        .addPreferredGap(
                                                javax.swing.LayoutStyle.ComponentPlacement.RELATED,
                                                4,
                                                javax.swing.GroupLayout.PREFERRED_SIZE)
                                        .addComponent(
                                                lblXResolution))
                        .addComponent(
                                txtXResolutionFrom,
                                javax.swing.GroupLayout.PREFERRED_SIZE,
                                javax.swing.GroupLayout.DEFAULT_SIZE,
                                javax.swing.GroupLayout.PREFERRED_SIZE))
.addPreferredGap(
        javax.swing.LayoutStyle.ComponentPlacement.RELATED)
.addGroup(
        panel1Layout
                        .createParallelGroup(
                                javax.swing.GroupLayout.Alignment.LEADING)
                        .addGroup(
```

```
                                panel1Layout
                                        .createSequentialGroup()
                                        .addPreferredGap(
                                                javax.swing.LayoutStyle.ComponentPlacement.RELATED,
                                                4,
                                                javax.swing.GroupLayout.PREFERRED_SIZE)
                                        .addComponent(
                                                lblYResolution))
                        .addComponent(
                                txtYResolutionFrom,
                                javax.swing.GroupLayout.PREFERRED_SIZE,
                                javax.swing.GroupLayout.DEFAULT_SIZE,
                                javax.swing.GroupLayout.PREFERRED_SIZE)))
                                                                        .addGroup(
                                                                                panel1Layout
.createSequentialGroup()
.addGroup(
        panel1Layout
                        .createParallelGroup(
                                javax.swing.GroupLayout.Alignment.BASELINE)
                        .addComponent(
```

```
                                    jLabel41)
                        .addComponent(
                                    txtXResolutionTo,
                                    javax.swing.GroupLayout.PREFERRED_SIZE,
                                    javax.swing.GroupLayout.DEFAULT_SIZE,
                                    javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addComponent(
                                    jLabel43))
.addPreferredGap(
      javax.swing.LayoutStyle.ComponentPlacement.RELATED)
.addGroup(
      panel1Layout
                        .createParallelGroup(
                                    javax.swing.GroupLayout.Alignment.BASELINE)
                        .addComponent(
                                    txtYResolutionTo,
                                    javax.swing.GroupLayout.PREFERRED_SIZE,
                                    javax.swing.GroupLayout.DEFAULT_SIZE,
                                    javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addComponent(
```

```
                                jLabel44)
                        .addComponent(
                                jLabel42))))
                                                        .addPreferredGap(
javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
                                                        .addGroup(
                                                                panel1Layout
                                                                        .createParallelGroup(
javax.swing.GroupLayout.Alignment.LEADING)
                                                                        .addGroup(
                                                                                panel1Layout
.createSequentialGroup()
.addGroup(
      panel1Layout
                        .createParallelGroup(
                                javax.swing.GroupLayout.Alignment.BASELINE)
                        .addComponent(
                                lblCreated)
                        .addComponent(
                                txtCreatedFrom,
                                javax.swing.GroupLayout.PREFERRED_SIZE,
                                javax.swing.GroupLayout.DEFAULT_SIZE,
                                javax.swing.GroupLayout.PREFERRED_SIZE))
```

```
.addPreferredGap(
      javax.swing.LayoutStyle.ComponentPlacement.RELATED)
.addGroup(
      panel1Layout
                  .createParallelGroup(
                              javax.swing.GroupLayout.Alignment.BASELINE)
                  .addComponent(
                              lblModify)
                  .addComponent(
                              txtModifyFrom,
                              javax.swing.GroupLayout.PREFERRED_SIZE,
                              javax.swing.GroupLayout.DEFAULT_SIZE,
                              javax.swing.GroupLayout.PREFERRED_SIZE)))
                                                                        .addGroup(
                                                                              panel1Layout
.createSequentialGroup()
.addGroup(
      panel1Layout
                  .createParallelGroup(
                              javax.swing.GroupLayout.Alignment.BASELINE)
                  .addComponent(
```

```
                                txtCreatedTo,
                                javax.swing.GroupLayout.PREFERRED_SIZE,
                                javax.swing.GroupLayout.DEFAULT_SIZE,
                                javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addComponent(
                                jLabel46))
.addPreferredGap(
        javax.swing.LayoutStyle.ComponentPlacement.RELATED)
.addGroup(
        panel1Layout
                        .createParallelGroup(
                                javax.swing.GroupLayout.Alignment.BASELINE)
                        .addComponent(
                                jLabel45)
                        .addComponent(
                                txtModifyTo,
                                javax.swing.GroupLayout.PREFERRED_SIZE,
                                javax.swing.GroupLayout.DEFAULT_SIZE,
                                javax.swing.GroupLayout.PREFERRED_SIZE))))
                                                .addContainerGap(
                                                        javax.swing.GroupLayout.DEFAULT_SIZE,
```

```
                                                                    Short.MAX_VALUE)));

        lblColor.setText("Modelo de Cores:");

        lblType.setText("Formato:");

        lblCamera.setText("Câmera:");

        javax.swing.GroupLayout panel2Layout = new javax.swing.GroupLayout(
                    panel2);
        panel2.setLayout(panel2Layout);
        panel2Layout
                    .setHorizontalGroup(panel2Layout
                                .createParallelGroup(
                                            javax.swing.GroupLayout.Alignment.LEADING)
                                .addGroup(
                                            panel2Layout
                                                        .createSequentialGroup()
                                                        .addGroup(
                                                                    panel2Layout
                                                                                .createParallelGroup(

javax.swing.GroupLayout.Alignment.LEADING,
                                                                                            false)
                                                                                .addGroup(
                                                                                            panel2Layout

.createSequentialGroup()

.addGap(33,

        33,

        33)

.addComponent(

        lblColor))
                                                                                .addGroup(
                                                                                            panel2Layout
```

```
.createSequentialGroup()
.addGap(50,
      50,
      50)
.addComponent(
      lblType))
                                                                  .addGroup(
                                                                        panel2Layout
.createSequentialGroup()
.addContainerGap()
.addGroup(
      panel2Layout
                  .createParallelGroup(
                        javax.swing.GroupLayout.Alignment.LEADING,
                        false)
                  .addComponent(
                        cmbType,
                        0,
                        javax.swing.GroupLayout.DEFAULT_SIZE,
                        Short.MAX_VALUE)
                  .addComponent(
```

```
                                cmbColor,
                                0,
                                118,
                                Short.MAX_VALUE)))
                                                                .addGroup(
                                                                        panel2Layout
.createSequentialGroup()
.addGap(49,
        49,
        49)
.addComponent(
        lblCamera))
                                                                .addGroup(
                                                                        panel2Layout
.createSequentialGroup()
.addContainerGap()
.addComponent(
        cmbCamera,
        0,
        javax.swing.GroupLayout.DEFAULT_SIZE,
        Short.MAX_VALUE)))
                                                        .addContainerGap(19, Short.MAX_VALUE)));
        panel2Layout
```

```
                .setVerticalGroup(panel2Layout
                        .createParallelGroup(
                                javax.swing.GroupLayout.Alignment.LEADING)
                        .addGroup(
                                panel2Layout
                                        .createSequentialGroup()
                                        .addContainerGap()
                                        .addComponent(lblColor)
                                        .addGap(9, 9, 9)
                                        .addComponent(
                                                cmbColor,
                                                javax.swing.GroupLayout.PREFERRED_SIZE,
                                                javax.swing.GroupLayout.DEFAULT_SIZE,
                                                javax.swing.GroupLayout.PREFERRED_SIZE)
                                        .addGap(18, 18, 18)
                                        .addComponent(lblType)
                                        .addPreferredGap(

javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                                        .addComponent(
                                                cmbType,
                                                javax.swing.GroupLayout.PREFERRED_SIZE,
                                                javax.swing.GroupLayout.DEFAULT_SIZE,
                                                javax.swing.GroupLayout.PREFERRED_SIZE)
                                        .addGap(18, 18, 18)
                                        .addComponent(lblCamera)
                                        .addPreferredGap(

javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
                                        .addComponent(
                                                cmbCamera,
                                                javax.swing.GroupLayout.PREFERRED_SIZE,
                                                javax.swing.GroupLayout.DEFAULT_SIZE,
                                                javax.swing.GroupLayout.PREFERRED_SIZE)
                                        .addContainerGap(
                                                javax.swing.GroupLayout.DEFAULT_SIZE,
                                                Short.MAX_VALUE)));

        javax.swing.GroupLayout pnlImageSearchLayout = new javax.swing.GroupLayout(
                        pnlImageSearch);
```

```
pnlImageSearch.setLayout(pnlImageSearchLayout);
pnlImageSearchLayout.setHorizontalGroup(pnlImageSearchLayout
            .createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(
                        pnlImageSearchLayout
                                    .createSequentialGroup()
                                    .addContainerGap()
                                    .addComponent(panel1,
                                                javax.swing.GroupLayout.PREFERRED_SIZE,
                                                javax.swing.GroupLayout.DEFAULT_SIZE,
                                                javax.swing.GroupLayout.PREFERRED_SIZE)
                                    .addGap(18, 18, 18)
                                    .addComponent(panel2,
                                                javax.swing.GroupLayout.PREFERRED_SIZE,
                                                javax.swing.GroupLayout.DEFAULT_SIZE,
                                                javax.swing.GroupLayout.PREFERRED_SIZE)
                                    .addContainerGap(
                                                javax.swing.GroupLayout.DEFAULT_SIZE,
                                                Short.MAX_VALUE)));
pnlImageSearchLayout.setVerticalGroup(pnlImageSearchLayout
            .createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(
                        pnlImageSearchLayout
                                    .createSequentialGroup()
                                    .addContainerGap()
                                    .addComponent(panel1,
                                                javax.swing.GroupLayout.PREFERRED_SIZE,
                                                javax.swing.GroupLayout.DEFAULT_SIZE,
                                                javax.swing.GroupLayout.PREFERRED_SIZE)
                                    .addContainerGap(25, Short.MAX_VALUE))
            .addGroup(
                        pnlImageSearchLayout
                                    .createSequentialGroup()
                                    .addGap(19, 19, 19)
                                    .addComponent(panel2,
                                                javax.swing.GroupLayout.DEFAULT_SIZE,
                                                javax.swing.GroupLayout.DEFAULT_SIZE,
                                                Short.MAX_VALUE).addGap(34, 34, 34)));

javax.swing.GroupLayout pnlImageMetadataLayout = new javax.swing.GroupLayout(
```

```
            pnlImageMetadata);
pnlImageMetadata.setLayout(pnlImageMetadataLayout);
pnlImageMetadataLayout.setHorizontalGroup(pnlImageMetadataLayout
            .createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(
                        pnlImageMetadataLayout
                                    .createSequentialGroup()
                                    .addGap(15, 15, 15)
                                    .addComponent(pnlImageSearch,
                                                javax.swing.GroupLayout.DEFAULT_SIZE,
                                                javax.swing.GroupLayout.DEFAULT_SIZE,
                                                Short.MAX_VALUE).addContainerGap()));
pnlImageMetadataLayout.setVerticalGroup(pnlImageMetadataLayout
            .createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(
                        pnlImageMetadataLayout
                                    .createSequentialGroup()
                                    .addGap(22, 22, 22)
                                    .addComponent(pnlImageSearch,
                                                javax.swing.GroupLayout.PREFERRED_SIZE,
                                                javax.swing.GroupLayout.DEFAULT_SIZE,
                                                javax.swing.GroupLayout.PREFERRED_SIZE)
                                    .addContainerGap(21, Short.MAX_VALUE)));

tabPnlSearch.addTab("Metadado em Imagem", pnlImageMetadata);

jPanel1.setBorder(javax.swing.BorderFactory
            .createTitledBorder("Configure a Porcentagem de Cada Cor:"));

txtYellowFrom.setColumns(5);
txtYellowFrom.setText("0");

txtRedFrom.setColumns(5);
txtRedFrom.setText("0");

txtPinkFrom.setColumns(5);
txtPinkFrom.setText("0");

txtBlueFrom.setColumns(5);
txtBlueFrom.setText("0");
```

```
txtPurpleFrom.setColumns(5);
txtPurpleFrom.setText("0");

jLabel13.setText("Amarelo: de");

jLabel18.setText("Vermelho: de");

jLabel6.setText("Rosa: de");

jLabel19.setText("Azul: de");

jLabel20.setText("Lilás: de");

jLabel21.setText("Laranja: de");

txtOrangeFrom.setColumns(5);
txtOrangeFrom.setText("0");

jLabel1.setText("%  até");

jLabel2.setText("%  até");

jLabel3.setText("%  até");

jLabel4.setText("%  até");

jLabel5.setText("%  até");

jLabel7.setText("%  até");

txtYellowTo.setColumns(5);

txtRedTo.setColumns(5);

txtPinkTo.setColumns(5);

txtBlueTo.setColumns(5);

txtPurpleTo.setColumns(5);
```

```
        txtOrangeTo.setColumns(5);

        jLabel8.setText("%");

        jLabel10.setText("%");

        jLabel11.setText("%");

        jLabel12.setText("%");

        jLabel14.setText("%");

        jLabel15.setText("%");

        javax.swing.GroupLayout jPanel7Layout = new javax.swing.GroupLayout(
                    jPanel7);
        jPanel7.setLayout(jPanel7Layout);
        jPanel7Layout
                    .setHorizontalGroup(jPanel7Layout
                                .createParallelGroup(
                                            javax.swing.GroupLayout.Alignment.LEADING)
                                .addGroup(
                                            jPanel7Layout
                                                        .createSequentialGroup()
                                                        .addContainerGap(
                                                                    javax.swing.GroupLayout.DEFAULT_SIZE,
                                                                    Short.MAX_VALUE)
                                                        .addGroup(
                                                                    jPanel7Layout
                                                                                .createParallelGroup(

javax.swing.GroupLayout.Alignment.LEADING)
                                                                                .addGroup(

javax.swing.GroupLayout.Alignment.TRAILING,
                                                                                            jPanel7Layout

.createSequentialGroup()
```

```
.addGroup(
        jPanel7Layout
                        .createParallelGroup(
                                        javax.swing.GroupLayout.Alignment.LEADING)
                        .addComponent(
                                        jLabel20,
                                        javax.swing.GroupLayout.Alignment.TRAILING)
                        .addComponent(
                                        jLabel13,
                                        javax.swing.GroupLayout.Alignment.TRAILING)
                        .addComponent(
                                        jLabel18,
                                        javax.swing.GroupLayout.Alignment.TRAILING)
                        .addComponent(
                                        jLabel19,
                                        javax.swing.GroupLayout.Alignment.TRAILING)
                        .addComponent(
                                        jLabel6,
                                        javax.swing.GroupLayout.Alignment.TRAILING))
.addPreferredGap(
```

```
        javax.swing.LayoutStyle.ComponentPlacement.RELATED)
.addGroup(
        jPanel7Layout
                        .createParallelGroup(
                                        javax.swing.GroupLayout.Alignment.TRAILING)
                        .addGroup(
                                        jPanel7Layout
                                                        .createParallelGroup(
                                                                        javax.swing.GroupLayout.Alignment.LEADING)
                                                        .addComponent(
                                                                        txtBlueFrom,
                                                                        javax.swing.GroupLayout.PREFERRED_SIZE,
                                                                        javax.swing.GroupLayout.DEFAULT_SIZE,
                                                                        javax.swing.GroupLayout.PREFERRED_SIZE)
                                                        .addComponent(
                                                                        txtPinkFrom,
                                                                        javax.swing.GroupLayout.PREFERRED_SIZE,
                                                                        javax.swing.GroupLayout.DEFAULT_SIZE,
                                                                        javax.swing.GroupLayout.PREFERRED_SIZE)
                                                        .addComponent(
```

```
                                                txtRedFrom,
                                                javax.swing.GroupLayout.PREFERRED_SIZE,
                                                javax.swing.GroupLayout.DEFAULT_SIZE,
                                                javax.swing.GroupLayout.PREFERRED_SIZE)
                                        .addComponent(
                                                txtYellowFrom,
                                                javax.swing.GroupLayout.PREFERRED_SIZE,
                                                javax.swing.GroupLayout.DEFAULT_SIZE,
                                                javax.swing.GroupLayout.PREFERRED_SIZE))
                        .addComponent(
                                txtPurpleFrom,
                                javax.swing.GroupLayout.PREFERRED_SIZE,
                                javax.swing.GroupLayout.DEFAULT_SIZE,
                                javax.swing.GroupLayout.PREFERRED_SIZE)))
                                                                        .addGroup(
javax.swing.GroupLayout.Alignment.TRAILING,
                                                                                jPanel7Layout
.createSequentialGroup()
.addComponent(
      jLabel21)
.addPreferredGap(
```

```
        javax.swing.LayoutStyle.ComponentPlacement.RELATED)
.addComponent(
        txtOrangeFrom,
        javax.swing.GroupLayout.PREFERRED_SIZE,
        javax.swing.GroupLayout.DEFAULT_SIZE,
        javax.swing.GroupLayout.PREFERRED_SIZE)))
                                                    .addGroup(
                                                            jPanel7Layout
                                                                    .createParallelGroup(
javax.swing.GroupLayout.Alignment.LEADING)
                                                                    .addGroup(
                                                                            jPanel7Layout
.createSequentialGroup()
.addGap(2,
        2,
        2)
.addComponent(
        jLabel1))
                                                                    .addGroup(
                                                                            jPanel7Layout
.createSequentialGroup()
.addPreferredGap(
        javax.swing.LayoutStyle.ComponentPlacement.RELATED)
```

```
.addGroup(
        jPanel7Layout
                        .createParallelGroup(
                                        javax.swing.GroupLayout.Alignment.LEADING)
                        .addComponent(
                                        jLabel7)
                        .addComponent(
                                        jLabel5)
                        .addComponent(
                                        jLabel3)
                        .addComponent(
                                        jLabel4)
                        .addComponent(
                                        jLabel2))))
                                                        .addGap(10, 10, 10)
                                                        .addGroup(
                                                                        jPanel7Layout
                                                                                        .createParallelGroup(
javax.swing.GroupLayout.Alignment.TRAILING)
                                                                                        .addComponent(
                                                                                                        txtOrangeTo,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
```

```
javax.swing.GroupLayout.PREFERRED_SIZE)
                                                                                .addGroup(
                                                                                        jPanel7Layout
.createParallelGroup(
        javax.swing.GroupLayout.Alignment.LEADING)
.addComponent(
        txtBlueTo,
        javax.swing.GroupLayout.PREFERRED_SIZE,
        javax.swing.GroupLayout.DEFAULT_SIZE,
        javax.swing.GroupLayout.PREFERRED_SIZE)
.addComponent(
        txtPinkTo,
        javax.swing.GroupLayout.PREFERRED_SIZE,
        javax.swing.GroupLayout.DEFAULT_SIZE,
        javax.swing.GroupLayout.PREFERRED_SIZE)
.addComponent(
        txtRedTo,
        javax.swing.GroupLayout.PREFERRED_SIZE,
        javax.swing.GroupLayout.DEFAULT_SIZE,
        javax.swing.GroupLayout.PREFERRED_SIZE)
.addComponent(
```

```
        txtYellowTo,
        javax.swing.GroupLayout.PREFERRED_SIZE,
        javax.swing.GroupLayout.DEFAULT_SIZE,
        javax.swing.GroupLayout.PREFERRED_SIZE))
                                                                .addComponent(
                                                                        txtPurpleTo,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
                                                        .addPreferredGap(
javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                                                        .addGroup(
                                                                jPanel7Layout
                                                                        .createParallelGroup(
javax.swing.GroupLayout.Alignment.LEADING)
                                                                        .addComponent(jLabel15)
                                                                        .addComponent(jLabel14)
                                                                        .addComponent(jLabel12)
                                                                        .addComponent(jLabel11)
                                                                        .addComponent(jLabel10)
                                                                        .addComponent(jLabel8))));
        jPanel7Layout
                        .setVerticalGroup(jPanel7Layout
                                        .createParallelGroup(
                                                        javax.swing.GroupLayout.Alignment.LEADING)
                                        .addGroup(
                                                        jPanel7Layout
                                                                        .createSequentialGroup()
                                                                        .addGap(13, 13, 13)
                                                                        .addGroup(
                                                                                        jPanel7Layout
```

```
                                                                                .createParallelGroup(
javax.swing.GroupLayout.Alignment.LEADING)
                                                                                .addGroup(
                                                                                        jPanel7Layout
.createSequentialGroup()
.addGroup(
      jPanel7Layout
                  .createParallelGroup(
                              javax.swing.GroupLayout.Alignment.BASELINE)
                  .addComponent(
                              txtYellowTo,
                              javax.swing.GroupLayout.PREFERRED_SIZE,
                              javax.swing.GroupLayout.DEFAULT_SIZE,
                              javax.swing.GroupLayout.PREFERRED_SIZE)
                  .addComponent(
                              jLabel8))
.addPreferredGap(
      javax.swing.LayoutStyle.ComponentPlacement.RELATED)
.addGroup(
      jPanel7Layout
                  .createParallelGroup(
```

```
                                javax.swing.GroupLayout.Alignment.BASELINE)
                        .addComponent(
                                txtRedTo,
                                javax.swing.GroupLayout.PREFERRED_SIZE,
                                javax.swing.GroupLayout.DEFAULT_SIZE,
                                javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addComponent(
                                jLabel10))
.addPreferredGap(
        javax.swing.LayoutStyle.ComponentPlacement.RELATED)
.addGroup(
        jPanel7Layout
                        .createParallelGroup(
                                javax.swing.GroupLayout.Alignment.BASELINE)
                        .addComponent(
                                txtPinkTo,
                                javax.swing.GroupLayout.PREFERRED_SIZE,
                                javax.swing.GroupLayout.DEFAULT_SIZE,
                                javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addComponent(
```

```
                                jLabel11))
.addPreferredGap(
      javax.swing.LayoutStyle.ComponentPlacement.RELATED)
.addGroup(
      jPanel7Layout
                  .createParallelGroup(
                              javax.swing.GroupLayout.Alignment.BASELINE)
                  .addComponent(
                              txtBlueTo,
                              javax.swing.GroupLayout.PREFERRED_SIZE,
                              javax.swing.GroupLayout.DEFAULT_SIZE,
                              javax.swing.GroupLayout.PREFERRED_SIZE)
                  .addComponent(
                              jLabel12))
.addGap(8,
      8,
      8)
.addGroup(
      jPanel7Layout
                  .createParallelGroup(
```

```
                                javax.swing.GroupLayout.Alignment.BASELINE)
                        .addComponent(
                                txtPurpleTo,
                                javax.swing.GroupLayout.PREFERRED_SIZE,
                                javax.swing.GroupLayout.DEFAULT_SIZE,
                                javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addComponent(
                                jLabel14))
.addPreferredGap(
        javax.swing.LayoutStyle.ComponentPlacement.RELATED)
.addGroup(
        jPanel7Layout
                        .createParallelGroup(
                                javax.swing.GroupLayout.Alignment.BASELINE)
                        .addComponent(
                                txtOrangeTo,
                                javax.swing.GroupLayout.PREFERRED_SIZE,
                                javax.swing.GroupLayout.DEFAULT_SIZE,
                                javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addComponent(
```

```
                                jLabel15)))
                                                                                .addGroup(
                                                                                        jPanel7Layout
.createSequentialGroup()
.addGroup(
      jPanel7Layout
                  .createParallelGroup(
                              javax.swing.GroupLayout.Alignment.BASELINE)
                  .addComponent(
                              jLabel13)
                  .addComponent(
                              txtYellowFrom,
                              javax.swing.GroupLayout.PREFERRED_SIZE,
                              javax.swing.GroupLayout.DEFAULT_SIZE,
                              javax.swing.GroupLayout.PREFERRED_SIZE)
                  .addComponent(
                              jLabel1))
.addPreferredGap(
      javax.swing.LayoutStyle.ComponentPlacement.RELATED)
.addGroup(
      jPanel7Layout
```

```
                        .createParallelGroup(
                                    javax.swing.GroupLayout.Alignment.BASELINE)
                        .addComponent(
                                    jLabel18)
                        .addComponent(
                                    txtRedFrom,
                                    javax.swing.GroupLayout.PREFERRED_SIZE,
                                    javax.swing.GroupLayout.DEFAULT_SIZE,
                                    javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addComponent(
                                    jLabel2))
            .addPreferredGap(
                  javax.swing.LayoutStyle.ComponentPlacement.RELATED)
            .addGroup(
                  jPanel7Layout
                        .createParallelGroup(
                                    javax.swing.GroupLayout.Alignment.LEADING)
                        .addComponent(
                                    jLabel6)
                        .addGroup(
```

```
                                jPanel7Layout
                                        .createSequentialGroup()
                                        .addGroup(
                                                jPanel7Layout
                                                        .createParallelGroup(
javax.swing.GroupLayout.Alignment.BASELINE)
                                                        .addComponent(
                                                                txtPinkFrom,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
                                                        .addComponent(
                                                                jLabel4))
                                        .addPreferredGap(
                                                javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                                        .addGroup(
                                                jPanel7Layout
                                                        .createParallelGroup(
```

```
javax.swing.GroupLayout.Alignment.BASELINE)
                                                        .addComponent(
                                                                txtBlueFrom,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
                                                        .addComponent(
                                                                jLabel19)
                                                        .addComponent(
                                                                jLabel3))
                                        .addGap(8,
                                                8,
                                                8)
                                        .addGroup(
                                                jPanel7Layout
                                                        .createParallelGroup(
javax.swing.GroupLayout.Alignment.BASELINE)
```

```
                                                                .addComponent(
                                                                        txtPurpleFrom,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
                                                                .addComponent(
                                                                        jLabel20)
                                                                .addComponent(
                                                                        jLabel5))))
.addPreferredGap(
        javax.swing.LayoutStyle.ComponentPlacement.RELATED)
.addGroup(
        jPanel7Layout
                        .createParallelGroup(
                                javax.swing.GroupLayout.Alignment.BASELINE)
                        .addComponent(
                                jLabel21)
                        .addComponent(
```

```
                        txtOrangeFrom,
                        javax.swing.GroupLayout.PREFERRED_SIZE,
                        javax.swing.GroupLayout.DEFAULT_SIZE,
                        javax.swing.GroupLayout.PREFERRED_SIZE)
                .addComponent(
                        jLabel7))))
                                        .addContainerGap(
                                                javax.swing.GroupLayout.DEFAULT_SIZE,
                                                Short.MAX_VALUE)));

        jLabel9.setText("Branco: de");

        jLabel22.setText("Preto: de");

        jLabel23.setText("Cinza: de");

        jLabel24.setText("Verde: de");

        txtWhiteFrom.setColumns(5);
        txtWhiteFrom.setText("0");

        txtBlackFrom.setColumns(5);
        txtBlackFrom.setText("0");

        txtGreenFrom.setColumns(5);
        txtGreenFrom.setText("0");

        txtGrayFrom.setColumns(5);
        txtGrayFrom.setText("0");

        jLabel25.setText("Marrom: de");

        txtBrownFrom.setColumns(5);
        txtBrownFrom.setText("0");
```

```
jLabel16.setText("%  até");

jLabel17.setText("%  até");

jLabel26.setText("%  até");

jLabel27.setText("%  até");

jLabel28.setText("%  até");

txtBrownTo.setColumns(5);

jLabel29.setText("%");

jLabel30.setText("%");

txtGrayTo.setColumns(5);

txtGreenTo.setColumns(5);

jLabel31.setText("%");

jLabel32.setText("%");

txtBlackTo.setColumns(5);

txtWhiteTo.setColumns(5);

jLabel33.setText("%");

javax.swing.GroupLayout jPanel3Layout = new javax.swing.GroupLayout(
            jPanel3);
jPanel3.setLayout(jPanel3Layout);
jPanel3Layout
            .setHorizontalGroup(jPanel3Layout
                        .createParallelGroup(
                                    javax.swing.GroupLayout.Alignment.LEADING)
                        .addGroup(
                                    jPanel3Layout
```

```
                                                        .createSequentialGroup()
                                                        .addContainerGap(
                                                                    javax.swing.GroupLayout.DEFAULT_SIZE,
                                                                    Short.MAX_VALUE)
                                                        .addGroup(
                                                                    jPanel3Layout
                                                                                .createParallelGroup(
javax.swing.GroupLayout.Alignment.TRAILING)
                                                                                .addGroup(
javax.swing.GroupLayout.Alignment.LEADING,
                                                                                            jPanel3Layout
.createSequentialGroup()
.addGap(3,
      3,
      3)
.addGroup(
      jPanel3Layout
                  .createParallelGroup(
                              javax.swing.GroupLayout.Alignment.TRAILING)
                  .addGroup(
                              jPanel3Layout
                                          .createSequentialGroup()
                                          .addComponent(
                                                      jLabel22)
```

```
                                        .addGap(1,
                                                1,
                                                1))
                        .addComponent(
                                jLabel9)
                        .addComponent(
                                jLabel23)
                        .addComponent(
                                jLabel24))
        .addPreferredGap(
                javax.swing.LayoutStyle.ComponentPlacement.RELATED)
        .addGroup(
                jPanel3Layout
                        .createParallelGroup(
                                javax.swing.GroupLayout.Alignment.LEADING)
                        .addComponent(
                                txtWhiteFrom,
                                javax.swing.GroupLayout.PREFERRED_SIZE,
                                javax.swing.GroupLayout.DEFAULT_SIZE,
                                javax.swing.GroupLayout.PREFERRED_SIZE)
```

```
                        .addComponent(
                                txtBlackFrom,
                                javax.swing.GroupLayout.PREFERRED_SIZE,
                                javax.swing.GroupLayout.DEFAULT_SIZE,
                                javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addComponent(
                                txtGreenFrom,
                                javax.swing.GroupLayout.PREFERRED_SIZE,
                                javax.swing.GroupLayout.DEFAULT_SIZE,
                                javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addComponent(
                                txtGrayFrom,
                                javax.swing.GroupLayout.PREFERRED_SIZE,
                                javax.swing.GroupLayout.DEFAULT_SIZE,
                                javax.swing.GroupLayout.PREFERRED_SIZE)))
                                                                .addGroup(
                                                                        jPanel3Layout
.createSequentialGroup()
.addComponent(
      jLabel25)
.addPreferredGap(
```

```
        javax.swing.LayoutStyle.ComponentPlacement.RELATED)
.addComponent(
        txtBrownFrom,
        javax.swing.GroupLayout.PREFERRED_SIZE,
        javax.swing.GroupLayout.DEFAULT_SIZE,
        javax.swing.GroupLayout.PREFERRED_SIZE)))
                                                        .addPreferredGap(
javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                                                        .addGroup(
                                                                jPanel3Layout
                                                                        .createParallelGroup(
javax.swing.GroupLayout.Alignment.LEADING)
                                                                        .addComponent(jLabel16)
                                                                        .addComponent(jLabel28)
                                                                        .addComponent(jLabel27)
                                                                        .addComponent(jLabel26)
                                                                        .addComponent(jLabel17))
                                                        .addGap(12, 12, 12)
                                                        .addGroup(
                                                                jPanel3Layout
                                                                        .createParallelGroup(
javax.swing.GroupLayout.Alignment.TRAILING)
                                                                        .addGroup(
                                                                                jPanel3Layout
.createParallelGroup(
        javax.swing.GroupLayout.Alignment.LEADING)
.addComponent(
```

```
                txtGrayTo,
                javax.swing.GroupLayout.PREFERRED_SIZE,
                javax.swing.GroupLayout.DEFAULT_SIZE,
                javax.swing.GroupLayout.PREFERRED_SIZE)
        .addComponent(
                txtGreenTo,
                javax.swing.GroupLayout.PREFERRED_SIZE,
                javax.swing.GroupLayout.DEFAULT_SIZE,
                javax.swing.GroupLayout.PREFERRED_SIZE)
        .addComponent(
                txtBlackTo,
                javax.swing.GroupLayout.PREFERRED_SIZE,
                javax.swing.GroupLayout.DEFAULT_SIZE,
                javax.swing.GroupLayout.PREFERRED_SIZE)
        .addComponent(
                txtWhiteTo,
                javax.swing.GroupLayout.PREFERRED_SIZE,
                javax.swing.GroupLayout.DEFAULT_SIZE,
                javax.swing.GroupLayout.PREFERRED_SIZE))
                                                        .addComponent(
                                                                txtBrownTo,
```

```
javax.swing.GroupLayout.PREFERRED_SIZE,

javax.swing.GroupLayout.DEFAULT_SIZE,

javax.swing.GroupLayout.PREFERRED_SIZE))
                                                        .addPreferredGap(

javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                                                        .addGroup(
                                                                jPanel3Layout
                                                                        .createParallelGroup(

javax.swing.GroupLayout.Alignment.LEADING)
                                                                        .addComponent(jLabel32)
                                                                        .addComponent(jLabel33)
                                                                        .addComponent(jLabel31)
                                                                        .addComponent(jLabel30)
                                                                        .addComponent(jLabel29))));
        jPanel3Layout
                        .setVerticalGroup(jPanel3Layout
                                        .createParallelGroup(
                                                        javax.swing.GroupLayout.Alignment.LEADING)
                                        .addGroup(
                                                        jPanel3Layout
                                                                .createSequentialGroup()
                                                                .addContainerGap()
                                                                .addGroup(
                                                                        jPanel3Layout
                                                                                .createParallelGroup(

javax.swing.GroupLayout.Alignment.LEADING)
                                                                                .addGroup(
                                                                                        jPanel3Layout

.createSequentialGroup()

.addGroup(

        jPanel3Layout
```

```
                        .createParallelGroup(
                                    javax.swing.GroupLayout.Alignment.BASELINE)
                        .addComponent(
                                    txtWhiteTo,
                                    javax.swing.GroupLayout.PREFERRED_SIZE,
                                    javax.swing.GroupLayout.DEFAULT_SIZE,
                                    javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addComponent(
                                    jLabel33))
            .addPreferredGap(
                  javax.swing.LayoutStyle.ComponentPlacement.RELATED)
            .addGroup(
                  jPanel3Layout
                        .createParallelGroup(
                                    javax.swing.GroupLayout.Alignment.BASELINE)
                        .addComponent(
                                    txtBlackTo,
                                    javax.swing.GroupLayout.PREFERRED_SIZE,
                                    javax.swing.GroupLayout.DEFAULT_SIZE,
                                    javax.swing.GroupLayout.PREFERRED_SIZE)
```

```
                    .addComponent(
                                jLabel32))
.addPreferredGap(
      javax.swing.LayoutStyle.ComponentPlacement.RELATED)
.addGroup(
      jPanel3Layout
                    .createParallelGroup(
                                javax.swing.GroupLayout.Alignment.BASELINE)
                    .addComponent(
                                txtGreenTo,
                                javax.swing.GroupLayout.PREFERRED_SIZE,
                                javax.swing.GroupLayout.DEFAULT_SIZE,
                                javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addComponent(
                                jLabel31))
.addPreferredGap(
      javax.swing.LayoutStyle.ComponentPlacement.RELATED)
.addGroup(
      jPanel3Layout
                    .createParallelGroup(
```

```
                                javax.swing.GroupLayout.Alignment.BASELINE)
                        .addComponent(
                                txtGrayTo,
                                javax.swing.GroupLayout.PREFERRED_SIZE,
                                javax.swing.GroupLayout.DEFAULT_SIZE,
                                javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addComponent(
                                jLabel30))
.addGap(8,
        8,
        8)
.addGroup(
        jPanel3Layout
                        .createParallelGroup(
                                javax.swing.GroupLayout.Alignment.BASELINE)
                        .addComponent(
                                txtBrownTo,
                                javax.swing.GroupLayout.PREFERRED_SIZE,
                                javax.swing.GroupLayout.DEFAULT_SIZE,
                                javax.swing.GroupLayout.PREFERRED_SIZE)
```

```
                    .addComponent(
                                jLabel29)))
                                                                                .addComponent(jLabel16)
                                                                                .addGroup(
                                                                                            jPanel3Layout
.createSequentialGroup()
.addGroup(
      jPanel3Layout
                    .createParallelGroup(
                                javax.swing.GroupLayout.Alignment.BASELINE)
                    .addComponent(
                                txtWhiteFrom,
                                javax.swing.GroupLayout.PREFERRED_SIZE,
                                javax.swing.GroupLayout.DEFAULT_SIZE,
                                javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addComponent(
                                jLabel9))
.addPreferredGap(
      javax.swing.LayoutStyle.ComponentPlacement.RELATED)
.addGroup(
      jPanel3Layout
```

```
                    .createParallelGroup(
                                javax.swing.GroupLayout.Alignment.BASELINE)
                    .addComponent(
                                txtBlackFrom,
                                javax.swing.GroupLayout.PREFERRED_SIZE,
                                javax.swing.GroupLayout.DEFAULT_SIZE,
                                javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addComponent(
                                jLabel22)
                    .addComponent(
                                jLabel17))
.addPreferredGap(
      javax.swing.LayoutStyle.ComponentPlacement.RELATED)
.addGroup(
      jPanel3Layout
                    .createParallelGroup(
                                javax.swing.GroupLayout.Alignment.BASELINE)
                    .addComponent(
                                txtGreenFrom,
                                javax.swing.GroupLayout.PREFERRED_SIZE,
```

```
                                javax.swing.GroupLayout.DEFAULT_SIZE,
                                javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addComponent(
                                jLabel24)
                        .addComponent(
                                jLabel26))
.addPreferredGap(
        javax.swing.LayoutStyle.ComponentPlacement.RELATED)
.addGroup(
        jPanel3Layout
                        .createParallelGroup(
                                javax.swing.GroupLayout.Alignment.BASELINE)
                        .addComponent(
                                txtGrayFrom,
                                javax.swing.GroupLayout.PREFERRED_SIZE,
                                javax.swing.GroupLayout.DEFAULT_SIZE,
                                javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addComponent(
                                jLabel23)
                        .addComponent(
```

```
                                jLabel27))
.addPreferredGap(
      javax.swing.LayoutStyle.ComponentPlacement.RELATED)
.addGroup(
      jPanel3Layout
                  .createParallelGroup(
                              javax.swing.GroupLayout.Alignment.BASELINE)
                  .addComponent(
                              txtBrownFrom,
                              javax.swing.GroupLayout.PREFERRED_SIZE,
                              javax.swing.GroupLayout.DEFAULT_SIZE,
                              javax.swing.GroupLayout.PREFERRED_SIZE)
                  .addComponent(
                              jLabel25)
                  .addComponent(
                              jLabel28))))
                                                .addContainerGap(18, Short.MAX_VALUE)));

      javax.swing.GroupLayout jPanel1Layout = new javax.swing.GroupLayout(
                  jPanel1);
      jPanel1.setLayout(jPanel1Layout);
      jPanel1Layout
                  .setHorizontalGroup(jPanel1Layout
                              .createParallelGroup(
```

```
                                        javax.swing.GroupLayout.Alignment.LEADING)
                                .addGroup(
                                        javax.swing.GroupLayout.Alignment.TRAILING,
                                        jPanel1Layout
                                                .createSequentialGroup()
                                                .addContainerGap()
                                                .addComponent(
                                                        jPanel3,
                                                        javax.swing.GroupLayout.DEFAULT_SIZE,
                                                        javax.swing.GroupLayout.DEFAULT_SIZE,
                                                        Short.MAX_VALUE)
                                                .addPreferredGap(

javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                                                .addComponent(
                                                        jPanel7,
                                                        javax.swing.GroupLayout.PREFERRED_SIZE,
                                                        javax.swing.GroupLayout.DEFAULT_SIZE,
                                                        javax.swing.GroupLayout.PREFERRED_SIZE)
                                                .addGap(12, 12, 12)));
        jPanel1Layout.setVerticalGroup(jPanel1Layout
                        .createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                        .addGroup(
                                        javax.swing.GroupLayout.Alignment.TRAILING,
                                        jPanel1Layout
                                                .createSequentialGroup()
                                                .addContainerGap(
                                                        javax.swing.GroupLayout.DEFAULT_SIZE,
                                                        Short.MAX_VALUE)
                                                .addComponent(jPanel3,
                                                        javax.swing.GroupLayout.PREFERRED_SIZE,
                                                        javax.swing.GroupLayout.DEFAULT_SIZE,
                                                        javax.swing.GroupLayout.PREFERRED_SIZE)
                                                .addGap(21, 21, 21))
                        .addGroup(
                                        jPanel1Layout
                                                .createSequentialGroup()
                                                .addComponent(jPanel7,
                                                        javax.swing.GroupLayout.PREFERRED_SIZE,
                                                        javax.swing.GroupLayout.DEFAULT_SIZE,
```

```
                                                javax.swing.GroupLayout.PREFERRED_SIZE)
                                        .addContainerGap(
                                                javax.swing.GroupLayout.DEFAULT_SIZE,
                                                Short.MAX_VALUE)));

        jLabel34.setText("Selecione a Cor Predominante da Imagem:");

        javax.swing.GroupLayout pnlImageColorLayout = new javax.swing.GroupLayout(
                    pnlImageColor);
        pnlImageColor.setLayout(pnlImageColorLayout);
        pnlImageColorLayout
                    .setHorizontalGroup(pnlImageColorLayout
                                .createParallelGroup(
                                        javax.swing.GroupLayout.Alignment.LEADING)
                                .addGroup(
                                        pnlImageColorLayout
                                                .createSequentialGroup()
                                                .addContainerGap()
                                                .addGroup(
                                                        pnlImageColorLayout
                                                                .createParallelGroup(

javax.swing.GroupLayout.Alignment.LEADING)
                                                                .addGroup(

pnlImageColorLayout

.createSequentialGroup()

.addComponent(

        jPanel1,

        javax.swing.GroupLayout.DEFAULT_SIZE,

        488,

        Short.MAX_VALUE)

.addGap(6,
```

```
        6,
        6))
                                                                                .addGroup(
javax.swing.GroupLayout.Alignment.TRAILING,
pnlImageColorLayout
.createSequentialGroup()
.addComponent(
        jLabel34)
.addPreferredGap(
        javax.swing.LayoutStyle.ComponentPlacement.RELATED)
.addComponent(
        cmbPredominantColor,
        javax.swing.GroupLayout.PREFERRED_SIZE,
        229,
        javax.swing.GroupLayout.PREFERRED_SIZE)
.addGap(32,
        32,
        32)))));
        pnlImageColorLayout
                    .setVerticalGroup(pnlImageColorLayout
                                .createParallelGroup(
                                            javax.swing.GroupLayout.Alignment.LEADING)
                                .addGroup(
```

```
                                                javax.swing.GroupLayout.Alignment.TRAILING,
                                                pnlImageColorLayout
                                                            .createSequentialGroup()
                                                            .addGap(27, 27, 27)
                                                            .addGroup(
                                                                        pnlImageColorLayout
                                                                                    .createParallelGroup(

javax.swing.GroupLayout.Alignment.BASELINE)
                                                                                    .addComponent(jLabel34)
                                                                                    .addComponent(

cmbPredominantColor,

javax.swing.GroupLayout.PREFERRED_SIZE,

javax.swing.GroupLayout.DEFAULT_SIZE,

javax.swing.GroupLayout.PREFERRED_SIZE))
                                                            .addPreferredGap(

javax.swing.LayoutStyle.ComponentPlacement.RELATED,
                                                                        29, Short.MAX_VALUE)
                                                            .addComponent(
                                                                        jPanel1,
                                                                        javax.swing.GroupLayout.PREFERRED_SIZE,
                                                                        javax.swing.GroupLayout.DEFAULT_SIZE,
                                                                        javax.swing.GroupLayout.PREFERRED_SIZE)
                                                            .addGap(19, 19, 19)));

      tabPnlSearch.addTab("Coloração da Imagem", pnlImageColor);

      jPanel4.setBorder(javax.swing.BorderFactory
                  .createTitledBorder("Tipos de Busca:"));

      jLabel47.setText("Simples:");

      cmbRelSimple.setModel(new javax.swing.DefaultComboBoxModel<Integer>(
                  new Integer[] { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 }));
```

```
        cmbRelSimilarity
                    .setModel(new javax.swing.DefaultComboBoxModel<Integer>(
                                new Integer[] { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 }));

        jLabel48.setText("Similaridade:");

        cmbRelSynonysm.setModel(new javax.swing.DefaultComboBoxModel<Integer>(
                    new Integer[] { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 }));

        jLabel49.setText("Sinônimo:");

        cmbRelSubstring.setModel(new javax.swing.DefaultComboBoxModel<Integer>(
                    new Integer[] { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 }));

        jLabel50.setText("Substring:");

        javax.swing.GroupLayout jPanel4Layout = new javax.swing.GroupLayout(
                    jPanel4);
        jPanel4.setLayout(jPanel4Layout);
        jPanel4Layout
                    .setHorizontalGroup(jPanel4Layout
                                .createParallelGroup(
                                            javax.swing.GroupLayout.Alignment.LEADING)
                                .addGroup(
                                            jPanel4Layout
                                                        .createSequentialGroup()
                                                        .addGap(19, 19, 19)
                                                        .addGroup(
                                                                    jPanel4Layout
                                                                                .createParallelGroup(

javax.swing.GroupLayout.Alignment.LEADING)
                                                                                .addGroup(

javax.swing.GroupLayout.Alignment.TRAILING,
                                                                                            jPanel4Layout

.createSequentialGroup()

.addComponent(
```

```
        jLabel47)
.addPreferredGap(
        javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
.addComponent(
        cmbRelSimple,
        javax.swing.GroupLayout.PREFERRED_SIZE,
        javax.swing.GroupLayout.DEFAULT_SIZE,
        javax.swing.GroupLayout.PREFERRED_SIZE))
                                                                .addGroup(
                                                                        jPanel4Layout
.createSequentialGroup()
.addComponent(
        jLabel48)
.addPreferredGap(
        javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
.addComponent(
        cmbRelSimilarity,
        javax.swing.GroupLayout.PREFERRED_SIZE,
        javax.swing.GroupLayout.DEFAULT_SIZE,
        javax.swing.GroupLayout.PREFERRED_SIZE)))
                                                        .addGap(39, 39, 39)
                                                        .addGroup(
```

```
                                                                jPanel4Layout
                                                                        .createParallelGroup(
javax.swing.GroupLayout.Alignment.TRAILING)
                                                                        .addGroup(
                                                                                jPanel4Layout
.createSequentialGroup()
.addComponent(
      jLabel50)
.addPreferredGap(
      javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
.addComponent(
      cmbRelSubstring,
      javax.swing.GroupLayout.PREFERRED_SIZE,
      javax.swing.GroupLayout.DEFAULT_SIZE,
      javax.swing.GroupLayout.PREFERRED_SIZE))
                                                                        .addGroup(
                                                                                jPanel4Layout
.createSequentialGroup()
.addComponent(
      jLabel49)
.addPreferredGap(
      javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
.addComponent(
```

```
        cmbRelSynonysm,

        javax.swing.GroupLayout.PREFERRED_SIZE,

        javax.swing.GroupLayout.DEFAULT_SIZE,

        javax.swing.GroupLayout.PREFERRED_SIZE)))
                                                        .addContainerGap(23, Short.MAX_VALUE)));
        jPanel4Layout
                    .setVerticalGroup(jPanel4Layout
                                .createParallelGroup(
                                            javax.swing.GroupLayout.Alignment.LEADING)
                                .addGroup(
                                            javax.swing.GroupLayout.Alignment.TRAILING,
                                            jPanel4Layout
                                                        .createSequentialGroup()
                                                        .addContainerGap(
                                                                    javax.swing.GroupLayout.DEFAULT_SIZE,
                                                                    Short.MAX_VALUE)
                                                        .addGroup(
                                                                    jPanel4Layout
                                                                                .createParallelGroup(

javax.swing.GroupLayout.Alignment.LEADING)
                                                                                .addGroup(
                                                                                            jPanel4Layout

.createSequentialGroup()

.addGroup(

        jPanel4Layout

                    .createParallelGroup(

                                javax.swing.GroupLayout.Alignment.BASELINE)

                    .addComponent(
```

```
                                cmbRelSynonysm,
                                javax.swing.GroupLayout.PREFERRED_SIZE,
                                javax.swing.GroupLayout.DEFAULT_SIZE,
                                javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addComponent(
                                jLabel49))
        .addPreferredGap(
                javax.swing.LayoutStyle.ComponentPlacement.RELATED)
        .addGroup(
                jPanel4Layout
                        .createParallelGroup(
                                javax.swing.GroupLayout.Alignment.BASELINE)
                        .addComponent(
                                cmbRelSubstring,
                                javax.swing.GroupLayout.PREFERRED_SIZE,
                                javax.swing.GroupLayout.DEFAULT_SIZE,
                                javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addComponent(
                                jLabel50)))
                                                                        .addGroup(
                                                                                jPanel4Layout
```

```
.createSequentialGroup()
.addGroup(
      jPanel4Layout
                  .createParallelGroup(
                              javax.swing.GroupLayout.Alignment.BASELINE)
                  .addComponent(
                              cmbRelSimple,
                              javax.swing.GroupLayout.PREFERRED_SIZE,
                              javax.swing.GroupLayout.DEFAULT_SIZE,
                              javax.swing.GroupLayout.PREFERRED_SIZE)
                  .addComponent(
                              jLabel47))
.addPreferredGap(
      javax.swing.LayoutStyle.ComponentPlacement.RELATED)
.addGroup(
      jPanel4Layout
                  .createParallelGroup(
                              javax.swing.GroupLayout.Alignment.BASELINE)
                  .addComponent(
                              cmbRelSimilarity,
```

```
                        javax.swing.GroupLayout.PREFERRED_SIZE,

                        javax.swing.GroupLayout.DEFAULT_SIZE,

                        javax.swing.GroupLayout.PREFERRED_SIZE)

                .addComponent(

                        jLabel48))))
                                                .addContainerGap()));

        jPanel5.setBorder(javax.swing.BorderFactory
                .createTitledBorder("Imagem:"));

        jLabel51.setText("Largura:");

        cmbRelWidth.setModel(new javax.swing.DefaultComboBoxModel<Integer>(
                new Integer[] { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 }));

        cmbRelHeight.setModel(new javax.swing.DefaultComboBoxModel<Integer>(
                new Integer[] { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 }));

        jLabel52.setText("Altura:");

        cmbRelSize.setModel(new javax.swing.DefaultComboBoxModel<Integer>(
                new Integer[] { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 }));

        jLabel53.setText("Tamanho:");

        cmbRelXResol.setModel(new javax.swing.DefaultComboBoxModel<Integer>(
                new Integer[] { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 }));

        jLabel54.setText("ResoluçãoX:");

        jLabel55.setText("ResoluçãoY:");

        cmbRelYResol.setModel(new javax.swing.DefaultComboBoxModel<Integer>(
                new Integer[] { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 }));
```

```
        jLabel56.setText("Modificação:");

        jLabel57.setText("Criação:");

        cmbRelCreated.setModel(new javax.swing.DefaultComboBoxModel<Integer>(
                    new Integer[] { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 }));

        cmbRelModif.setModel(new javax.swing.DefaultComboBoxModel<Integer>(
                    new Integer[] { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 }));

        jLabel58.setText("Modelo Cores:");

        cmbRelColorType.setModel(new javax.swing.DefaultComboBoxModel<Integer>(
                    new Integer[] { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 }));

        cmbRelFormat.setModel(new javax.swing.DefaultComboBoxModel<Integer>(
                    new Integer[] { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 }));

        jLabel59.setText("Formato:");

        jLabel60.setText("Câmera:");

        cmbRelCamera.setModel(new javax.swing.DefaultComboBoxModel<Integer>(
                    new Integer[] { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 }));

        javax.swing.GroupLayout jPanel5Layout = new javax.swing.GroupLayout(
                    jPanel5);
        jPanel5.setLayout(jPanel5Layout);
        jPanel5Layout
                    .setHorizontalGroup(jPanel5Layout
                                .createParallelGroup(
                                            javax.swing.GroupLayout.Alignment.LEADING)
                                .addGroup(
                                            jPanel5Layout
                                                        .createSequentialGroup()
                                                        .addGroup(
                                                                    jPanel5Layout
                                                                                .createParallelGroup(

javax.swing.GroupLayout.Alignment.LEADING)
```

```
                                                                .addGroup(
                                                                        jPanel5Layout
.createSequentialGroup()
.addGap(19,
        19,
        19)
.addGroup(
        jPanel5Layout
                .createParallelGroup(
                        javax.swing.GroupLayout.Alignment.LEADING)
                .addGroup(
                        jPanel5Layout
                                .createParallelGroup(
                                        javax.swing.GroupLayout.Alignment.LEADING)
                                .addGroup(
                                        jPanel5Layout
                                                .createParallelGroup(

javax.swing.GroupLayout.Alignment.TRAILING)
                                                .addGroup(
                                                        jPanel5Layout
```

```
.createSequentialGroup()
                                                                                .addComponent(
jLabel54)
.addPreferredGap(
javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
                                                                                .addComponent(
cmbRelXResol,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
                                                                    .addGroup(
                                                                            jPanel5Layout
.createParallelGroup(
javax.swing.GroupLayout.Alignment.LEADING,
```

```
false)
                                                                                .addGroup(
jPanel5Layout
        .createSequentialGroup()
        .addGap(8,
                        8,
                        8)
        .addGroup(
                        jPanel5Layout
                                        .createParallelGroup(
                                                        javax.swing.GroupLayout.Alignment.TRAILING)
                                        .addComponent(
                                                        jLabel52)
```

```
                        .addComponent(
                                jLabel51))
            .addPreferredGap(
                    javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
            .addGroup(
                    jPanel5Layout
                            .createParallelGroup(
                                    javax.swing.GroupLayout.Alignment.LEADING)
                            .addComponent(
                                    cmbRelWidth,
                                    javax.swing.GroupLayout.PREFERRED_SIZE,
                                    javax.swing.GroupLayout.DEFAULT_SIZE,
                                    javax.swing.GroupLayout.PREFERRED_SIZE)
```

```
                                .addComponent(
                                        cmbRelHeight,
                                        javax.swing.GroupLayout.PREFERRED_SIZE,
                                        javax.swing.GroupLayout.DEFAULT_SIZE,
                                        javax.swing.GroupLayout.PREFERRED_SIZE)))
                                                                        .addGroup(
javax.swing.GroupLayout.Alignment.TRAILING,
jPanel5Layout
        .createSequentialGroup()
        .addComponent(
                jLabel53)
        .addPreferredGap(
                javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
```

```
        .addComponent(
                        cmbRelSize,
                        javax.swing.GroupLayout.PREFERRED_SIZE,
                        javax.swing.GroupLayout.DEFAULT_SIZE,
                        javax.swing.GroupLayout.PREFERRED_SIZE))))
                                        .addGroup(
                                                        javax.swing.GroupLayout.Alignment.TRAILING,
                                                        jPanel5Layout
                                                                        .createSequentialGroup()
                                                                        .addComponent(
                                                                                        jLabel55)
                                                                        .addPreferredGap(
javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
                                                                        .addComponent(
                                                                                        cmbRelYResol,
javax.swing.GroupLayout.PREFERRED_SIZE,
```

```
javax.swing.GroupLayout.DEFAULT_SIZE,

javax.swing.GroupLayout.PREFERRED_SIZE)))
                    .addGroup(
                                javax.swing.GroupLayout.Alignment.TRAILING,
                                jPanel5Layout
                                            .createSequentialGroup()
                                            .addComponent(
                                                        jLabel57)
                                            .addPreferredGap(
                                                        javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
                                            .addComponent(
                                                        cmbRelCreated,
                                                        javax.swing.GroupLayout.PREFERRED_SIZE,
                                                        javax.swing.GroupLayout.DEFAULT_SIZE,
                                                        javax.swing.GroupLayout.PREFERRED_SIZE))
                    .addGroup(
                                javax.swing.GroupLayout.Alignment.TRAILING,
                                jPanel5Layout
                                            .createSequentialGroup()
```

```
                                        .addComponent(
                                                jLabel56)
                                        .addPreferredGap(
                                                javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
                                        .addComponent(
                                                cmbRelModif,
                                                javax.swing.GroupLayout.PREFERRED_SIZE,
                                                javax.swing.GroupLayout.DEFAULT_SIZE,
                                                javax.swing.GroupLayout.PREFERRED_SIZE))))
                                                        .addGroup(
                                                                jPanel5Layout
.createSequentialGroup()
.addContainerGap()
.addGroup(
      jPanel5Layout
                  .createParallelGroup(
                          javax.swing.GroupLayout.Alignment.LEADING)
                  .addGroup(
                          jPanel5Layout
                                  .createSequentialGroup()
                                  .addComponent(
```

```
                                        jLabel58)
                                .addPreferredGap(
                                        javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
                                .addComponent(
                                        cmbRelColorType,
                                        javax.swing.GroupLayout.PREFERRED_SIZE,
                                        javax.swing.GroupLayout.DEFAULT_SIZE,
                                        javax.swing.GroupLayout.PREFERRED_SIZE))
            .addGroup(
                    javax.swing.GroupLayout.Alignment.TRAILING,
                    jPanel5Layout
                            .createSequentialGroup()
                            .addComponent(
                                    jLabel59)
                            .addPreferredGap(
                                    javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
                            .addComponent(
                                    cmbRelFormat,
                                    javax.swing.GroupLayout.PREFERRED_SIZE,
                                    javax.swing.GroupLayout.DEFAULT_SIZE,
```

```
                                                javax.swing.GroupLayout.PREFERRED_SIZE))
                        .addGroup(
                                        javax.swing.GroupLayout.Alignment.TRAILING,
                                        jPanel5Layout
                                                        .createSequentialGroup()
                                                        .addComponent(
                                                                        jLabel60)
                                                        .addPreferredGap(
                                                                        javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
                                                        .addComponent(
                                                                        cmbRelCamera,
                                                                        javax.swing.GroupLayout.PREFERRED_SIZE,
                                                                        javax.swing.GroupLayout.DEFAULT_SIZE,
                                                                        javax.swing.GroupLayout.PREFERRED_SIZE)))))
                                                        .addContainerGap(19, Short.MAX_VALUE)));
        jPanel5Layout
                        .setVerticalGroup(jPanel5Layout
                                        .createParallelGroup(
                                                        javax.swing.GroupLayout.Alignment.LEADING)
                                        .addGroup(
                                                        jPanel5Layout
                                                                        .createSequentialGroup()
                                                                        .addGroup(
                                                                                        jPanel5Layout
                                                                                                        .createParallelGroup(
```

```
javax.swing.GroupLayout.Alignment.BASELINE)
                                                                .addComponent(
                                                                        cmbRelWidth,

javax.swing.GroupLayout.PREFERRED_SIZE,

javax.swing.GroupLayout.DEFAULT_SIZE,

javax.swing.GroupLayout.PREFERRED_SIZE)
                                                                .addComponent(jLabel51))
                                                        .addPreferredGap(

javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                                                        .addGroup(
                                                                jPanel5Layout
                                                                .createParallelGroup(

javax.swing.GroupLayout.Alignment.BASELINE)
                                                                .addComponent(jLabel52)
                                                                .addComponent(
                                                                        cmbRelHeight,

javax.swing.GroupLayout.PREFERRED_SIZE,

javax.swing.GroupLayout.DEFAULT_SIZE,

javax.swing.GroupLayout.PREFERRED_SIZE))
                                                        .addPreferredGap(

javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                                                        .addGroup(
                                                                jPanel5Layout
                                                                .createParallelGroup(

javax.swing.GroupLayout.Alignment.BASELINE)
                                                                .addComponent(
                                                                        cmbRelSize,

javax.swing.GroupLayout.PREFERRED_SIZE,
```

```
javax.swing.GroupLayout.DEFAULT_SIZE,

javax.swing.GroupLayout.PREFERRED_SIZE)
                                                .addComponent(jLabel53))
                                        .addPreferredGap(

javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                                        .addGroup(
                                                jPanel5Layout
                                                .createParallelGroup(

javax.swing.GroupLayout.Alignment.BASELINE)
                                                .addComponent(
                                                        cmbRelXResol,

javax.swing.GroupLayout.PREFERRED_SIZE,

javax.swing.GroupLayout.DEFAULT_SIZE,

javax.swing.GroupLayout.PREFERRED_SIZE)
                                                .addComponent(jLabel54))
                                        .addPreferredGap(

javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                                        .addGroup(
                                                jPanel5Layout
                                                .createParallelGroup(

javax.swing.GroupLayout.Alignment.BASELINE)
                                                .addComponent(
                                                        cmbRelYResol,

javax.swing.GroupLayout.PREFERRED_SIZE,

javax.swing.GroupLayout.DEFAULT_SIZE,

javax.swing.GroupLayout.PREFERRED_SIZE)
                                                .addComponent(jLabel55))
                                        .addPreferredGap(
```

```
javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                                                .addGroup(
                                                        jPanel5Layout
                                                                .createParallelGroup(

javax.swing.GroupLayout.Alignment.BASELINE)
                                                                .addComponent(
                                                                        cmbRelCreated,

javax.swing.GroupLayout.PREFERRED_SIZE,

javax.swing.GroupLayout.DEFAULT_SIZE,

javax.swing.GroupLayout.PREFERRED_SIZE)
                                                                .addComponent(jLabel57))
                                                .addPreferredGap(

javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                                                .addGroup(
                                                        jPanel5Layout
                                                                .createParallelGroup(

javax.swing.GroupLayout.Alignment.BASELINE)
                                                                .addComponent(
                                                                        cmbRelModif,

javax.swing.GroupLayout.PREFERRED_SIZE,

javax.swing.GroupLayout.DEFAULT_SIZE,

javax.swing.GroupLayout.PREFERRED_SIZE)
                                                                .addComponent(jLabel56))
                                                .addPreferredGap(

javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                                                .addGroup(
                                                        jPanel5Layout
                                                                .createParallelGroup(
```

```
javax.swing.GroupLayout.Alignment.BASELINE)
                                                                .addComponent(
cmbRelColorType,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
                                                                .addComponent(jLabel58))
                                                        .addPreferredGap(
javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                                                        .addGroup(
                                                                jPanel5Layout
                                                                .createParallelGroup(
javax.swing.GroupLayout.Alignment.BASELINE)
                                                                .addComponent(
                                                                        cmbRelFormat,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
                                                                .addComponent(jLabel59))
                                                        .addPreferredGap(
javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                                                        .addGroup(
                                                                jPanel5Layout
                                                                .createParallelGroup(
javax.swing.GroupLayout.Alignment.BASELINE)
                                                                .addComponent(
                                                                        cmbRelCamera,
```

```
javax.swing.GroupLayout.PREFERRED_SIZE,

javax.swing.GroupLayout.DEFAULT_SIZE,

javax.swing.GroupLayout.PREFERRED_SIZE)
                                                                    .addComponent(jLabel60))
                                                        .addContainerGap(15, Short.MAX_VALUE)));

        jPanel6.setBorder(javax.swing.BorderFactory
                    .createTitledBorder("Cores:"));

        jLabel61.setText("Predominante:");

        jLabel62.setText("Branco:");

        jLabel63.setText("Preto:");

        jLabel64.setText("Verde:");

        jLabel65.setText("Cinza:");

        jLabel66.setText("Marrom:");

        cmbRelBrown.setModel(new javax.swing.DefaultComboBoxModel<Integer>(
                    new Integer[] { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 }));

        cmbRelGray.setModel(new javax.swing.DefaultComboBoxModel<Integer>(
                    new Integer[] { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 }));

        cmbRelGreen.setModel(new javax.swing.DefaultComboBoxModel<Integer>(
                    new Integer[] { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 }));

        cmbRelBlack.setModel(new javax.swing.DefaultComboBoxModel<Integer>(
                    new Integer[] { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 }));

        cmbRelWhite.setModel(new javax.swing.DefaultComboBoxModel<Integer>(
                    new Integer[] { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 }));

        cmbRelPredominant
```

```
                .setModel(new javax.swing.DefaultComboBoxModel<Integer>(
                                new Integer[] { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 }));

        cmbRelPink.setModel(new javax.swing.DefaultComboBoxModel<Integer>(
                new Integer[] { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 }));

        cmbRelRed.setModel(new javax.swing.DefaultComboBoxModel<Integer>(
                new Integer[] { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 }));

        cmbRelYellow.setModel(new javax.swing.DefaultComboBoxModel<Integer>(
                new Integer[] { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 }));

        jLabel67.setText("Amarelo:");

        jLabel68.setText("Lilás:");

        jLabel69.setText("Azul:");

        jLabel70.setText("Rosa:");

        jLabel71.setText("Vermelho:");

        cmbRelBlue.setModel(new javax.swing.DefaultComboBoxModel<Integer>(
                new Integer[] { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 }));

        cmbRelPurple.setModel(new javax.swing.DefaultComboBoxModel<Integer>(
                new Integer[] { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 }));

        cmbRelOrange.setModel(new javax.swing.DefaultComboBoxModel<Integer>(
                new Integer[] { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 }));

        jLabel72.setText("Laranja:");

        javax.swing.GroupLayout jPanel6Layout = new javax.swing.GroupLayout(
                jPanel6);
        jPanel6.setLayout(jPanel6Layout);
        jPanel6Layout
                .setHorizontalGroup(jPanel6Layout
                                .createParallelGroup(
                                                javax.swing.GroupLayout.Alignment.LEADING)
```

```
                                .addGroup(
                                        jPanel6Layout
                                                .createSequentialGroup()
                                                .addContainerGap()
                                                .addGroup(
                                                        jPanel6Layout
                                                                .createParallelGroup(
javax.swing.GroupLayout.Alignment.LEADING)
                                                                .addGroup(
                                                                        jPanel6Layout
.createParallelGroup(
      javax.swing.GroupLayout.Alignment.LEADING)
.addGroup(
      jPanel6Layout
                  .createParallelGroup(
                          javax.swing.GroupLayout.Alignment.TRAILING)
                  .addGroup(
                          jPanel6Layout
                                  .createSequentialGroup()
                                  .addComponent(
                                          jLabel64)
                                  .addPreferredGap(
                                          javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
                                  .addComponent(
```

```
                                                cmbRelGreen,
                                                javax.swing.GroupLayout.PREFERRED_SIZE,
                                                javax.swing.GroupLayout.DEFAULT_SIZE,
                                                javax.swing.GroupLayout.PREFERRED_SIZE))
                        .addGroup(
                                jPanel6Layout
                                        .createParallelGroup(
                                                javax.swing.GroupLayout.Alignment.LEADING,
                                                false)
                                        .addGroup(
                                                jPanel6Layout
                                                        .createSequentialGroup()
                                                        .addGroup(
                                                                jPanel6Layout
.createParallelGroup(
javax.swing.GroupLayout.Alignment.TRAILING)
                                                                                .addComponent(
jLabel62)
```

```
                                                                                .addComponent(
jLabel61))
                                                                        .addPreferredGap(
javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
                                                                        .addGroup(
                                                                                jPanel6Layout
.createParallelGroup(
javax.swing.GroupLayout.Alignment.LEADING)
                                                                                .addComponent(
cmbRelPredominant,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
                                                                                .addComponent(
cmbRelWhite,
```

```
javax.swing.GroupLayout.PREFERRED_SIZE,

javax.swing.GroupLayout.DEFAULT_SIZE,

javax.swing.GroupLayout.PREFERRED_SIZE)))
                                                .addGroup(
                                                        javax.swing.GroupLayout.Alignment.TRAILING,
                                                        jPanel6Layout
                                                                .createSequentialGroup()
                                                                .addComponent(
                                                                        jLabel63)
                                                                .addPreferredGap(

javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
                                                                .addComponent(
                                                                        cmbRelBlack,

javax.swing.GroupLayout.PREFERRED_SIZE,

javax.swing.GroupLayout.DEFAULT_SIZE,

javax.swing.GroupLayout.PREFERRED_SIZE))))
```

```
.addGroup(
	javax.swing.GroupLayout.Alignment.TRAILING,
	jPanel6Layout
			.createSequentialGroup()
			.addComponent(
					jLabel65)
			.addPreferredGap(
					javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
			.addComponent(
					cmbRelGray,
					javax.swing.GroupLayout.PREFERRED_SIZE,
					javax.swing.GroupLayout.DEFAULT_SIZE,
					javax.swing.GroupLayout.PREFERRED_SIZE)))
                                                                    .addGroup(
javax.swing.GroupLayout.Alignment.TRAILING,
                                                                                jPanel6Layout
.createSequentialGroup()
.addComponent(
	jLabel66)
.addPreferredGap(
	javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
```

```
.addComponent(
        cmbRelBrown,
        javax.swing.GroupLayout.PREFERRED_SIZE,
        javax.swing.GroupLayout.DEFAULT_SIZE,
        javax.swing.GroupLayout.PREFERRED_SIZE)))
                                                        .addPreferredGap(
javax.swing.LayoutStyle.ComponentPlacement.RELATED,
                                                                38, Short.MAX_VALUE)
                                                        .addGroup(
                                                                jPanel6Layout
                                                                        .createParallelGroup(
javax.swing.GroupLayout.Alignment.LEADING)
                                                                        .addGroup(
                                                                                jPanel6Layout
.createParallelGroup(
        javax.swing.GroupLayout.Alignment.LEADING)
.addGroup(
        jPanel6Layout
                        .createParallelGroup(
                                javax.swing.GroupLayout.Alignment.TRAILING)
                        .addGroup(
                                jPanel6Layout
                                        .createSequentialGroup()
```

```
                                .addComponent(
                                        jLabel69)
                                .addPreferredGap(
                                        javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
                                .addComponent(
                                        cmbRelBlue,
                                        javax.swing.GroupLayout.PREFERRED_SIZE,
                                        javax.swing.GroupLayout.DEFAULT_SIZE,
                                        javax.swing.GroupLayout.PREFERRED_SIZE))
            .addGroup(
                    jPanel6Layout
                            .createParallelGroup(
                                    javax.swing.GroupLayout.Alignment.LEADING,
                                    false)
                            .addGroup(
                                    jPanel6Layout
                                            .createSequentialGroup()
                                            .addGroup(
                                                    jPanel6Layout
```

```
.createParallelGroup(
javax.swing.GroupLayout.Alignment.TRAILING)
                                                                    .addComponent(
jLabel71)
                                                                    .addComponent(
jLabel67))
                                                            .addPreferredGap(
javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
                                                            .addGroup(
                                                                    jPanel6Layout
.createParallelGroup(
javax.swing.GroupLayout.Alignment.LEADING)
                                                                    .addComponent(
cmbRelYellow,
javax.swing.GroupLayout.PREFERRED_SIZE,
```

```
javax.swing.GroupLayout.DEFAULT_SIZE,

javax.swing.GroupLayout.PREFERRED_SIZE)
                                                                                .addComponent(

cmbRelRed,

javax.swing.GroupLayout.PREFERRED_SIZE,

javax.swing.GroupLayout.DEFAULT_SIZE,

javax.swing.GroupLayout.PREFERRED_SIZE)))
                                                .addGroup(
                                                        javax.swing.GroupLayout.Alignment.TRAILING,
                                                        jPanel6Layout
                                                                .createSequentialGroup()
                                                                .addComponent(
                                                                        jLabel70)
                                                                .addPreferredGap(

javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
                                                                .addComponent(
```

```
                                                                        cmbRelPink,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))))
.addGroup(
      javax.swing.GroupLayout.Alignment.TRAILING,
      jPanel6Layout
                  .createSequentialGroup()
                  .addComponent(
                              jLabel68)
                  .addPreferredGap(
                              javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
                  .addComponent(
                              cmbRelPurple,
                              javax.swing.GroupLayout.PREFERRED_SIZE,
                              javax.swing.GroupLayout.DEFAULT_SIZE,
                              javax.swing.GroupLayout.PREFERRED_SIZE)))
                                                                        .addGroup(
javax.swing.GroupLayout.Alignment.TRAILING,
```

```
                                                                                                jPanel6Layout
.createSequentialGroup()
.addComponent(
      jLabel72)
.addPreferredGap(
      javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
.addComponent(
      cmbRelOrange,
      javax.swing.GroupLayout.PREFERRED_SIZE,
      javax.swing.GroupLayout.DEFAULT_SIZE,
      javax.swing.GroupLayout.PREFERRED_SIZE)))
                                                        .addGap(24, 24, 24)));
      jPanel6Layout
                  .setVerticalGroup(jPanel6Layout
                              .createParallelGroup(
                                          javax.swing.GroupLayout.Alignment.LEADING)
                              .addGroup(
                                          javax.swing.GroupLayout.Alignment.TRAILING,
                                          jPanel6Layout
                                                      .createSequentialGroup()
                                                      .addContainerGap(
                                                                  javax.swing.GroupLayout.DEFAULT_SIZE,
                                                                  Short.MAX_VALUE)
                                                      .addGroup(
                                                                  jPanel6Layout
                                                                              .createParallelGroup(
javax.swing.GroupLayout.Alignment.LEADING)
                                                                              .addGroup(
                                                                                          jPanel6Layout
```

```
.createSequentialGroup()
.addGroup(
      jPanel6Layout
                  .createParallelGroup(
                              javax.swing.GroupLayout.Alignment.BASELINE)
                  .addComponent(
                              cmbRelYellow,
                              javax.swing.GroupLayout.PREFERRED_SIZE,
                              javax.swing.GroupLayout.DEFAULT_SIZE,
                              javax.swing.GroupLayout.PREFERRED_SIZE)
                  .addComponent(
                              jLabel67))
.addPreferredGap(
      javax.swing.LayoutStyle.ComponentPlacement.RELATED)
.addGroup(
      jPanel6Layout
                  .createParallelGroup(
                              javax.swing.GroupLayout.Alignment.BASELINE)
                  .addComponent(
                              jLabel71)
```

```
                    .addComponent(
                                cmbRelRed,
                                javax.swing.GroupLayout.PREFERRED_SIZE,
                                javax.swing.GroupLayout.DEFAULT_SIZE,
                                javax.swing.GroupLayout.PREFERRED_SIZE))
.addPreferredGap(
      javax.swing.LayoutStyle.ComponentPlacement.RELATED)
.addGroup(
      jPanel6Layout
                    .createParallelGroup(
                                javax.swing.GroupLayout.Alignment.BASELINE)
                    .addComponent(
                                cmbRelPink,
                                javax.swing.GroupLayout.PREFERRED_SIZE,
                                javax.swing.GroupLayout.DEFAULT_SIZE,
                                javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addComponent(
                                jLabel70))
.addPreferredGap(
      javax.swing.LayoutStyle.ComponentPlacement.RELATED)
```

```
.addGroup(
        jPanel6Layout
                        .createParallelGroup(
                                        javax.swing.GroupLayout.Alignment.BASELINE)
                        .addComponent(
                                        cmbRelBlue,
                                        javax.swing.GroupLayout.PREFERRED_SIZE,
                                        javax.swing.GroupLayout.DEFAULT_SIZE,
                                        javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addComponent(
                                        jLabel69))
.addPreferredGap(
        javax.swing.LayoutStyle.ComponentPlacement.RELATED)
.addGroup(
        jPanel6Layout
                        .createParallelGroup(
                                        javax.swing.GroupLayout.Alignment.BASELINE)
                        .addComponent(
                                        cmbRelPurple,
                                        javax.swing.GroupLayout.PREFERRED_SIZE,
```

```
                                javax.swing.GroupLayout.DEFAULT_SIZE,
                                javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addComponent(
                                jLabel68))
        .addPreferredGap(
                javax.swing.LayoutStyle.ComponentPlacement.RELATED)
        .addGroup(
                jPanel6Layout
                        .createParallelGroup(
                                javax.swing.GroupLayout.Alignment.BASELINE)
                        .addComponent(
                                cmbRelOrange,
                                javax.swing.GroupLayout.PREFERRED_SIZE,
                                javax.swing.GroupLayout.DEFAULT_SIZE,
                                javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addComponent(
                                jLabel72)))
                                                                        .addGroup(
                                                                                jPanel6Layout
        .createSequentialGroup()
        .addGroup(
```

```
        jPanel6Layout
                        .createParallelGroup(
                                        javax.swing.GroupLayout.Alignment.BASELINE)
                        .addComponent(
                                        cmbRelPredominant,
                                        javax.swing.GroupLayout.PREFERRED_SIZE,
                                        javax.swing.GroupLayout.DEFAULT_SIZE,
                                        javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addComponent(
                                        jLabel61))
.addPreferredGap(
        javax.swing.LayoutStyle.ComponentPlacement.RELATED)
.addGroup(
        jPanel6Layout
                        .createParallelGroup(
                                        javax.swing.GroupLayout.Alignment.BASELINE)
                        .addComponent(
                                        jLabel62)
                        .addComponent(
                                        cmbRelWhite,
```

```
                                javax.swing.GroupLayout.PREFERRED_SIZE,
                                javax.swing.GroupLayout.DEFAULT_SIZE,
                                javax.swing.GroupLayout.PREFERRED_SIZE))
.addPreferredGap(
      javax.swing.LayoutStyle.ComponentPlacement.RELATED)
.addGroup(
      jPanel6Layout
                  .createParallelGroup(
                                javax.swing.GroupLayout.Alignment.BASELINE)
                  .addComponent(
                                cmbRelBlack,
                                javax.swing.GroupLayout.PREFERRED_SIZE,
                                javax.swing.GroupLayout.DEFAULT_SIZE,
                                javax.swing.GroupLayout.PREFERRED_SIZE)
                  .addComponent(
                                jLabel63))
.addPreferredGap(
      javax.swing.LayoutStyle.ComponentPlacement.RELATED)
.addGroup(
      jPanel6Layout
```

```
                        .createParallelGroup(
                                    javax.swing.GroupLayout.Alignment.BASELINE)
                        .addComponent(
                                    cmbRelGreen,
                                    javax.swing.GroupLayout.PREFERRED_SIZE,
                                    javax.swing.GroupLayout.DEFAULT_SIZE,
                                    javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addComponent(
                                    jLabel64))
.addPreferredGap(
      javax.swing.LayoutStyle.ComponentPlacement.RELATED)
.addGroup(
      jPanel6Layout
                        .createParallelGroup(
                                    javax.swing.GroupLayout.Alignment.BASELINE)
                        .addComponent(
                                    cmbRelGray,
                                    javax.swing.GroupLayout.PREFERRED_SIZE,
                                    javax.swing.GroupLayout.DEFAULT_SIZE,
                                    javax.swing.GroupLayout.PREFERRED_SIZE)
```

```
                    .addComponent(
                                jLabel65))
.addPreferredGap(
      javax.swing.LayoutStyle.ComponentPlacement.RELATED)
.addGroup(
      jPanel6Layout
                    .createParallelGroup(
                                javax.swing.GroupLayout.Alignment.BASELINE)
                    .addComponent(
                                cmbRelBrown,
                                javax.swing.GroupLayout.PREFERRED_SIZE,
                                javax.swing.GroupLayout.DEFAULT_SIZE,
                                javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addComponent(
                                jLabel66))))
                                                .addContainerGap()));
      javax.swing.GroupLayout jPanel2Layout = new javax.swing.GroupLayout(
                    jPanel2);
      jPanel2.setLayout(jPanel2Layout);
      jPanel2Layout
                    .setHorizontalGroup(jPanel2Layout
                                .createParallelGroup(
                                            javax.swing.GroupLayout.Alignment.LEADING)
                                .addGroup(
```

```
                                        jPanel2Layout
                                                .createSequentialGroup()
                                                .addGap(21, 21, 21)
                                                .addComponent(
                                                        jPanel5,
                                                        javax.swing.GroupLayout.PREFERRED_SIZE,
                                                        javax.swing.GroupLayout.DEFAULT_SIZE,
                                                        javax.swing.GroupLayout.PREFERRED_SIZE)
                                                .addPreferredGap(

javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
                                                .addGroup(
                                                        jPanel2Layout
                                                                .createParallelGroup(

javax.swing.GroupLayout.Alignment.LEADING,
                                                                                        false)
                                                                .addComponent(
                                                                        jPanel6,

javax.swing.GroupLayout.DEFAULT_SIZE,

javax.swing.GroupLayout.DEFAULT_SIZE,

Short.MAX_VALUE)
                                                                .addComponent(
                                                                        jPanel4,

javax.swing.GroupLayout.DEFAULT_SIZE,

javax.swing.GroupLayout.DEFAULT_SIZE,

Short.MAX_VALUE))
                                                .addContainerGap(20, Short.MAX_VALUE)));
        jPanel2Layout
                        .setVerticalGroup(jPanel2Layout
                                        .createParallelGroup(
                                                        javax.swing.GroupLayout.Alignment.LEADING)
                                        .addGroup(
                                                        javax.swing.GroupLayout.Alignment.TRAILING,
```

```
                                        jPanel2Layout
                                                .createSequentialGroup()
                                                .addContainerGap()
                                                .addGroup(
                                                        jPanel2Layout
                                                                .createParallelGroup(
javax.swing.GroupLayout.Alignment.TRAILING)
                                                                .addComponent(
                                                                        jPanel5,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
                                                                .addGroup(
javax.swing.GroupLayout.Alignment.LEADING,
                                                                        jPanel2Layout
.createSequentialGroup()
.addComponent(
      jPanel4,
      javax.swing.GroupLayout.PREFERRED_SIZE,
      javax.swing.GroupLayout.DEFAULT_SIZE,
      javax.swing.GroupLayout.PREFERRED_SIZE)
.addPreferredGap(
      javax.swing.LayoutStyle.ComponentPlacement.RELATED)
.addComponent(
      jPanel6,
```

```
javax.swing.GroupLayout.DEFAULT_SIZE,

javax.swing.GroupLayout.DEFAULT_SIZE,

Short.MAX_VALUE)))
                                                    .addContainerGap()));

tabPnlSearch.addTab("Configurar Relevâncias", jPanel2);

btnSearch.setText("Buscar!");

lblNumberResults.setText("Número de Resultados:");

txtNumberResults
            .setFormatterFactory(new javax.swing.text.DefaultFormatterFactory(
                        new javax.swing.text.NumberFormatter(
                                    new java.text.DecimalFormat("#0"))));
txtNumberResults.setText("50");

lblPathResult
            .setText("Selecione o caminho para o arquivo de resultados:");

txtPathResult.setEditable(false);

btnPathResult.setText("Selecionar");

javax.swing.GroupLayout pnlSearchLayout = new javax.swing.GroupLayout(
            pnlSearch);
pnlSearch.setLayout(pnlSearchLayout);
pnlSearchLayout
            .setHorizontalGroup(pnlSearchLayout
                        .createParallelGroup(
                                    javax.swing.GroupLayout.Alignment.LEADING)
                        .addGroup(
                                    pnlSearchLayout
                                                .createSequentialGroup()
                                                .addContainerGap()
                                                .addComponent(
                                                            tabPnlSearch,
```

```
                                                                javax.swing.GroupLayout.PREFERRED_SIZE,
                                                                509,
                                                                javax.swing.GroupLayout.PREFERRED_SIZE)
                                                        .addContainerGap(12, Short.MAX_VALUE))
                                        .addGroup(
                                                pnlSearchLayout
                                                        .createSequentialGroup()
                                                        .addGap(58, 58, 58)
                                                        .addGroup(
                                                                pnlSearchLayout
                                                                        .createParallelGroup(
javax.swing.GroupLayout.Alignment.TRAILING)
                                                                        .addComponent(
                                                                                txtPathResult,
javax.swing.GroupLayout.Alignment.LEADING,
javax.swing.GroupLayout.DEFAULT_SIZE,
                                                                                383,
Short.MAX_VALUE)
                                                                        .addGroup(
javax.swing.GroupLayout.Alignment.LEADING,
                                                                                pnlSearchLayout
.createSequentialGroup()
.addComponent(
      lblNumberResults)
.addPreferredGap(
      javax.swing.LayoutStyle.ComponentPlacement.RELATED)
.addComponent(
      txtNumberResults,
```

```
        javax.swing.GroupLayout.PREFERRED_SIZE,
        30,
        javax.swing.GroupLayout.PREFERRED_SIZE)
.addPreferredGap(
        javax.swing.LayoutStyle.ComponentPlacement.RELATED,
        152,
        Short.MAX_VALUE)
.addComponent(
        btnSearch,
        javax.swing.GroupLayout.PREFERRED_SIZE,
        85,
        javax.swing.GroupLayout.PREFERRED_SIZE))
                                                                .addGroup(
                                                                        pnlSearchLayout
.createSequentialGroup()
.addComponent(
        lblPathResult)
.addPreferredGap(
        javax.swing.LayoutStyle.ComponentPlacement.RELATED)
.addComponent(
        btnPathResult,
```

```
        javax.swing.GroupLayout.PREFERRED_SIZE,

        109,

        javax.swing.GroupLayout.PREFERRED_SIZE)))
                                                        .addGap(90, 90, 90)));
        pnlSearchLayout
                    .setVerticalGroup(pnlSearchLayout
                                .createParallelGroup(
                                            javax.swing.GroupLayout.Alignment.LEADING)
                                .addGroup(
                                            pnlSearchLayout
                                                        .createSequentialGroup()
                                                        .addContainerGap()
                                                        .addComponent(
                                                                    tabPnlSearch,
                                                                    javax.swing.GroupLayout.PREFERRED_SIZE,
                                                                    333,
                                                                    javax.swing.GroupLayout.PREFERRED_SIZE)
                                                        .addPreferredGap(

javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
                                                        .addGroup(
                                                                    pnlSearchLayout
                                                                                .createParallelGroup(

javax.swing.GroupLayout.Alignment.BASELINE)

                                                                                .addComponent(
                                                                                            lblPathResult)
                                                                                .addComponent(
                                                                                            btnPathResult))
                                                        .addPreferredGap(

javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                                                        .addComponent(
                                                                    txtPathResult,
                                                                    javax.swing.GroupLayout.PREFERRED_SIZE,
                                                                    javax.swing.GroupLayout.DEFAULT_SIZE,
                                                                    javax.swing.GroupLayout.PREFERRED_SIZE)
```

```
                                                        .addPreferredGap(
javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
                                                        .addGroup(
                                                                pnlSearchLayout
                                                                        .createParallelGroup(
javax.swing.GroupLayout.Alignment.BASELINE)
                                                                        .addComponent(
lblNumberResults)
                                                                        .addComponent(
txtNumberResults,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
                                                                        .addComponent(btnSearch))
                                                        .addContainerGap(
                                                                javax.swing.GroupLayout.DEFAULT_SIZE,
                                                                Short.MAX_VALUE)));

        tabPnl.addTab("         Busca         ", pnlSearch);

        javax.swing.GroupLayout layout = new javax.swing.GroupLayout(
                        getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(layout.createParallelGroup(
                        javax.swing.GroupLayout.Alignment.LEADING).addGroup(
                        layout.createSequentialGroup()
                                        .addContainerGap()
                                        .addComponent(tabPnl,
                                                        javax.swing.GroupLayout.PREFERRED_SIZE, 536,
                                                        javax.swing.GroupLayout.PREFERRED_SIZE)
                                        .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
                                                        Short.MAX_VALUE)));
        layout.setVerticalGroup(layout.createParallelGroup(
```

```
                        javax.swing.GroupLayout.Alignment.LEADING).addGroup(
                        layout.createSequentialGroup()
                                    .addContainerGap()
                                    .addComponent(tabPnl,
                                                javax.swing.GroupLayout.PREFERRED_SIZE, 475,
                                                javax.swing.GroupLayout.PREFERRED_SIZE)
                                    .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
                                                Short.MAX_VALUE)));

            pack();
      }// </editor-fold>

      /**
       * @param args
       *            the command line arguments
       */
      public static void main(String args[]) {
            java.awt.EventQueue.invokeLater(new Runnable() {
                  public void run() {
                        new MainFrame().setVisible(true);
                  }
            });
      }

      // Variables declaration - do not modify
      private javax.swing.JCheckBox boxSimilarity;
      private javax.swing.JCheckBox boxSimple;
      private javax.swing.JCheckBox boxSubstring;
      private javax.swing.JCheckBox boxSynonysm;
      private javax.swing.ButtonGroup btnFollow;
      private javax.swing.JButton btnImage;
      private javax.swing.JButton btnIndex;
      private javax.swing.JButton btnIndexIt;
      private javax.swing.JButton btnMetadata;
      private javax.swing.ButtonGroup btnMetadataFormtat;
      private javax.swing.JRadioButton btnNone;
      private javax.swing.JButton btnPathResult;
      private javax.swing.JButton btnReuseIndex;
      private javax.swing.JButton btnSearch;
      private javax.swing.JButton btnSelectTag;
```

```
private javax.swing.JRadioButton btnText;
private javax.swing.JRadioButton btnXML;
private javax.swing.JRadioButton btnXMLText;
private javax.swing.JComboBox<String> cmbCamera;
private javax.swing.JComboBox<String> cmbColor;
private javax.swing.JComboBox<String> cmbPredominantColor;
private javax.swing.JComboBox<Integer> cmbRelBlack;
private javax.swing.JComboBox<Integer> cmbRelBlue;
private javax.swing.JComboBox<Integer> cmbRelBrown;
private javax.swing.JComboBox<Integer> cmbRelCamera;
private javax.swing.JComboBox<Integer> cmbRelColorType;
private javax.swing.JComboBox<Integer> cmbRelCreated;
private javax.swing.JComboBox<Integer> cmbRelFormat;
private javax.swing.JComboBox<Integer> cmbRelGray;
private javax.swing.JComboBox<Integer> cmbRelGreen;
private javax.swing.JComboBox<Integer> cmbRelHeight;
private javax.swing.JComboBox<Integer> cmbRelModif;
private javax.swing.JComboBox<Integer> cmbRelOrange;
private javax.swing.JComboBox<Integer> cmbRelPink;
private javax.swing.JComboBox<Integer> cmbRelPredominant;
private javax.swing.JComboBox<Integer> cmbRelPurple;
private javax.swing.JComboBox<Integer> cmbRelRed;
private javax.swing.JComboBox<Integer> cmbRelSimilarity;
private javax.swing.JComboBox<Integer> cmbRelSimple;
private javax.swing.JComboBox<Integer> cmbRelSize;
private javax.swing.JComboBox<Integer> cmbRelSubstring;
private javax.swing.JComboBox<Integer> cmbRelSynonysm;
private javax.swing.JComboBox<Integer> cmbRelWhite;
private javax.swing.JComboBox<Integer> cmbRelWidth;
private javax.swing.JComboBox<Integer> cmbRelXResol;
private javax.swing.JComboBox<Integer> cmbRelYResol;
private javax.swing.JComboBox<Integer> cmbRelYellow;
private javax.swing.JComboBox<String> cmbType;
private javax.swing.JLabel jLabel1;
private javax.swing.JLabel jLabel10;
private javax.swing.JLabel jLabel11;
private javax.swing.JLabel jLabel12;
private javax.swing.JLabel jLabel13;
private javax.swing.JLabel jLabel14;
private javax.swing.JLabel jLabel15;
```

```
private javax.swing.JLabel jLabel16;
private javax.swing.JLabel jLabel17;
private javax.swing.JLabel jLabel18;
private javax.swing.JLabel jLabel19;
private javax.swing.JLabel jLabel2;
private javax.swing.JLabel jLabel20;
private javax.swing.JLabel jLabel21;
private javax.swing.JLabel jLabel22;
private javax.swing.JLabel jLabel23;
private javax.swing.JLabel jLabel24;
private javax.swing.JLabel jLabel25;
private javax.swing.JLabel jLabel26;
private javax.swing.JLabel jLabel27;
private javax.swing.JLabel jLabel28;
private javax.swing.JLabel jLabel29;
private javax.swing.JLabel jLabel3;
private javax.swing.JLabel jLabel30;
private javax.swing.JLabel jLabel31;
private javax.swing.JLabel jLabel32;
private javax.swing.JLabel jLabel33;
private javax.swing.JLabel jLabel34;
private javax.swing.JLabel jLabel35;
private javax.swing.JLabel jLabel36;
private javax.swing.JLabel jLabel37;
private javax.swing.JLabel jLabel38;
private javax.swing.JLabel jLabel39;
private javax.swing.JLabel jLabel4;
private javax.swing.JLabel jLabel40;
private javax.swing.JLabel jLabel41;
private javax.swing.JLabel jLabel42;
private javax.swing.JLabel jLabel43;
private javax.swing.JLabel jLabel44;
private javax.swing.JLabel jLabel45;
private javax.swing.JLabel jLabel46;
private javax.swing.JLabel jLabel47;
private javax.swing.JLabel jLabel48;
private javax.swing.JLabel jLabel49;
private javax.swing.JLabel jLabel5;
private javax.swing.JLabel jLabel50;
private javax.swing.JLabel jLabel51;
```

```
private javax.swing.JLabel jLabel52;
private javax.swing.JLabel jLabel53;
private javax.swing.JLabel jLabel54;
private javax.swing.JLabel jLabel55;
private javax.swing.JLabel jLabel56;
private javax.swing.JLabel jLabel57;
private javax.swing.JLabel jLabel58;
private javax.swing.JLabel jLabel59;
private javax.swing.JLabel jLabel6;
private javax.swing.JLabel jLabel60;
private javax.swing.JLabel jLabel61;
private javax.swing.JLabel jLabel62;
private javax.swing.JLabel jLabel63;
private javax.swing.JLabel jLabel64;
private javax.swing.JLabel jLabel65;
private javax.swing.JLabel jLabel66;
private javax.swing.JLabel jLabel67;
private javax.swing.JLabel jLabel68;
private javax.swing.JLabel jLabel69;
private javax.swing.JLabel jLabel7;
private javax.swing.JLabel jLabel70;
private javax.swing.JLabel jLabel71;
private javax.swing.JLabel jLabel72;
private javax.swing.JLabel jLabel8;
private javax.swing.JLabel jLabel9;
private javax.swing.JPanel jPanel1;
private javax.swing.JPanel jPanel2;
private javax.swing.JPanel jPanel3;
private javax.swing.JPanel jPanel4;
private javax.swing.JPanel jPanel5;
private javax.swing.JPanel jPanel6;
private javax.swing.JPanel jPanel7;
private javax.swing.JLabel lblCamera;
private javax.swing.JLabel lblColor;
private javax.swing.JLabel lblCreated;
private javax.swing.JLabel lblFollow;
private javax.swing.JLabel lblHeigth;
private javax.swing.JLabel lblImage;
private javax.swing.JLabel lblIndex;
private javax.swing.JTextField lblIndexResult;
```

```
private javax.swing.JLabel lblMetadata;
private javax.swing.JLabel lblModify;
private javax.swing.JLabel lblNumberResults;
private javax.swing.JLabel lblPathResult;
private javax.swing.JLabel lblPerc;
private javax.swing.JLabel lblScale;
private javax.swing.JLabel lblSimilarity;
private javax.swing.JLabel lblSize;
private javax.swing.JLabel lblTag;
private javax.swing.JLabel lblType;
private javax.swing.JLabel lblWidth;
private javax.swing.JLabel lblWord;
private javax.swing.JLabel lblXResolution;
private javax.swing.JLabel lblYResolution;
private javax.swing.JPanel panel1;
private javax.swing.JPanel panel2;
private javax.swing.JPanel pnlAnnFormat;
private javax.swing.JPanel pnlImageColor;
private javax.swing.JPanel pnlImageMetadata;
private javax.swing.JPanel pnlImageSearch;
private javax.swing.JPanel pnlIndex;
private javax.swing.JPanel pnlSearch;
private javax.swing.JPanel pnlTagSearch;
private javax.swing.JPanel pnlTextMetadata;
private javax.swing.JPanel pnlWordSearch;
private javax.swing.JRadioButton rbtnFollowNo;
private javax.swing.JRadioButton rbtnFollowYes;
private javax.swing.JTabbedPane tabPnl;
private javax.swing.JTabbedPane tabPnlSearch;
private javax.swing.JFormattedTextField txtBlackFrom;
private javax.swing.JFormattedTextField txtBlackTo;
private javax.swing.JFormattedTextField txtBlueFrom;
private javax.swing.JFormattedTextField txtBlueTo;
private javax.swing.JFormattedTextField txtBrownFrom;
private javax.swing.JFormattedTextField txtBrownTo;
private javax.swing.JFormattedTextField txtCreatedFrom;
private javax.swing.JFormattedTextField txtCreatedTo;
private javax.swing.JFormattedTextField txtGrayFrom;
private javax.swing.JFormattedTextField txtGrayTo;
private javax.swing.JFormattedTextField txtGreenFrom;
```

```
        private javax.swing.JFormattedTextField txtGreenTo;
        private javax.swing.JTextField txtHeigthFrom;
        private javax.swing.JFormattedTextField txtHeigthTo;
        private javax.swing.JTextField txtImage;
        private javax.swing.JTextField txtIndex;
        private javax.swing.JTextField txtMetadata;
        private javax.swing.JFormattedTextField txtModifyFrom;
        private javax.swing.JFormattedTextField txtModifyTo;
        private javax.swing.JFormattedTextField txtNumberResults;
        private javax.swing.JFormattedTextField txtOrangeFrom;
        private javax.swing.JFormattedTextField txtOrangeTo;
        private javax.swing.JTextField txtPathResult;
        private javax.swing.JFormattedTextField txtPinkFrom;
        private javax.swing.JFormattedTextField txtPinkTo;
        private javax.swing.JFormattedTextField txtPurpleFrom;
        private javax.swing.JFormattedTextField txtPurpleTo;
        private javax.swing.JFormattedTextField txtRedFrom;
        private javax.swing.JFormattedTextField txtRedTo;
        private javax.swing.JFormattedTextField txtSimilarity;
        private javax.swing.JTextField txtSizeFrom;
        private javax.swing.JFormattedTextField txtSizeTo;
        private javax.swing.JTextField txtTag;
        private javax.swing.JFormattedTextField txtWhiteFrom;
        private javax.swing.JFormattedTextField txtWhiteTo;
        private javax.swing.JTextField txtWidthFrom;
        private javax.swing.JFormattedTextField txtWidthTo;
        private javax.swing.JTextField txtWord;
        private javax.swing.JTextField txtXResolutionFrom;
        private javax.swing.JFormattedTextField txtXResolutionTo;
        private javax.swing.JTextField txtYResolutionFrom;
        private javax.swing.JFormattedTextField txtYResolutionTo;
        private javax.swing.JFormattedTextField txtYellowFrom;
        private javax.swing.JFormattedTextField txtYellowTo;
        private static final javax.swing.JFileChooser selectFile = new javax.swing.JFileChooser();

}
```

Apêndice T – Classe *OptionTag*

```
package tcc.tbfm.view;

public class OptionTag extends javax.swing.JFrame {

	private static final long serialVersionUID = -1963046590587945959L;

	public OptionTag() {
		initComponents();
	}

	public javax.swing.JButton getBtnOk() {
		return btnOk;
	}

	public javax.swing.JButton getBtnCancel() {
		return btnCancel;
	}

	public javax.swing.JButton getBtnSearchTag() {
		return btnSearchTag;
	}

	public boolean isSimpleSearch() {
		return boxSimple.isSelected();
	}

	public boolean isSimilaritySearch() {
		return boxSimilarity.isSelected();
	}

	public String similarity() {
		return txtSimilarity.getText();
	}
```

```
public boolean isSubstringSearch() {
	return boxSubstring.isSelected();
}

public boolean isSynonysmSearch() {
	return boxSynonysm.isSelected();
}

public boolean isOnlySimpleSearch() {
	return (isSynonysmSearch() == false && isSubstringSearch() == false
			&& isSimilaritySearch() == false && isSimpleSearch() == true);
}

public boolean hasSearchModeSelected() {
	return (isSynonysmSearch() == true || isSubstringSearch() == true
			|| isSimilaritySearch() == true || isSimpleSearch() == true);
}

public java.awt.List getLstTags() {
	return lstTags;
}

public javax.swing.JTextField getTxtTag() {
	return txtTag;
}

public javax.swing.JCheckBox getBoxSynonysm() {
	return boxSynonysm;
}

public javax.swing.JCheckBox getBoxSimilarity() {
	return boxSimilarity;
}

public javax.swing.JCheckBox getBoxSimple() {
	return boxSimple;
}

public javax.swing.JCheckBox getBoxSubstring() {
	return boxSubstring;
```

```
}

private void initComponents() {

        lblTag = new javax.swing.JLabel();
        txtTag = new javax.swing.JTextField();
        btnSearchTag = new javax.swing.JButton();
        lblTags1 = new javax.swing.JLabel();
        btnOk = new javax.swing.JButton();
        btnCancel = new javax.swing.JButton();
        lstTags = new java.awt.List();
        lblSimilarity = new javax.swing.JLabel();
        lblMethod2 = new javax.swing.JLabel();
        txtSimilarity = new javax.swing.JFormattedTextField();
        lblMethod1 = new javax.swing.JLabel();
        boxSimilarity = new javax.swing.JCheckBox();
        lblPerc = new javax.swing.JLabel();
        boxSynonysm = new javax.swing.JCheckBox();
        boxSubstring = new javax.swing.JCheckBox();
        lblScale = new javax.swing.JLabel();
        boxSimple = new javax.swing.JCheckBox();
        lblTags2 = new javax.swing.JLabel();

        setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);

        lblTag.setText("Buscar Por:");

        txtTag.setColumns(20);

        btnSearchTag.setText("Buscar Tags");

        lblTags1.setText("Selecione uma");

        btnOk.setText("Ok");

        btnCancel.setText("Cancelar");

        lblSimilarity.setText("Porcentagem de Similaridade:");

        lblMethod2.setText("de Busca:");
```

```
            txtSimilarity.setText("70");

            lblMethod1.setText("Configuração ");

            boxSimilarity
                        .setText("Busca por Similaridade (Utiliza a porcentagem de similaridade)");

            lblPerc.setText("%");

            boxSynonysm
                        .setText("Busca por Sinônimos (Utiliza apenas os sinônimos informados pelo WordNet)");

            boxSubstring
                        .setText("Busca por Substring (Utiliza os caracteres informados como parte da tag
buscada)");

            lblScale.setText("(Escala de 0 à 100)");

            boxSimple
                        .setText("Busca Simples (Busca apenas pelos caracteres informados)");

            lblTags2.setText("Tag Encontrada:");

            javax.swing.GroupLayout layout = new javax.swing.GroupLayout(
                        getContentPane());
            getContentPane().setLayout(layout);
            layout.setHorizontalGroup(layout
                        .createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                        .addGroup(
                                    layout.createSequentialGroup()
                                                .addGap(46, 46, 46)
                                                .addGroup(
                                                            layout.createParallelGroup(

      javax.swing.GroupLayout.Alignment.TRAILING)
                                                                        .addGroup(

      javax.swing.GroupLayout.Alignment.LEADING,
```

```
layout.createSequentialGroup()
                                                                                .addGroup(
layout.createParallelGroup(
      javax.swing.GroupLayout.Alignment.LEADING)
      .addComponent(
                  lblMethod1)
      .addGroup(
                  layout.createSequentialGroup()
                              .addGap(10,
                                          10,
                                          10)
                              .addComponent(
                                          lblMethod2))
      .addComponent(
                  lblTag,
                  javax.swing.GroupLayout.Alignment.TRAILING))
.addPreferredGap(
javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
                                                                                .addGroup(
layout.createParallelGroup(
      javax.swing.GroupLayout.Alignment.LEADING,
```

```
        false)
        .addComponent(
                    boxSimple)
        .addComponent(
                    boxSimilarity)
        .addComponent(
                    boxSubstring,
                    javax.swing.GroupLayout.DEFAULT_SIZE,
                    javax.swing.GroupLayout.DEFAULT_SIZE,
                    Short.MAX_VALUE)
        .addComponent(
                    boxSynonysm,
                    javax.swing.GroupLayout.DEFAULT_SIZE,
                    javax.swing.GroupLayout.DEFAULT_SIZE,
                    Short.MAX_VALUE)
        .addComponent(
                    txtTag)))
                                                                .addGroup(
layout.createSequentialGroup()
                                                                    .addGroup(
layout.createParallelGroup(
```

```
javax.swing.GroupLayout.Alignment.LEADING)
.addComponent(
                btnSearchTag,
                javax.swing.GroupLayout.Alignment.TRAILING)
.addGroup(
                javax.swing.GroupLayout.Alignment.TRAILING,
                layout.createSequentialGroup()
                                .addGroup(
                                                layout.createParallelGroup(
                                                                javax.swing.GroupLayout.Alignment.LEADING)
                                                                .addComponent(
                                                                                lblTags1)
                                                                .addComponent(
                                                                                lblTags2))
                                .addGap(31,
                                                31,
                                                31)
                                .addGroup(
                                                layout.createParallelGroup(
                                                                javax.swing.GroupLayout.Alignment.TRAILING)
```

```
                                                .addGroup(
                                                        layout.createSequentialGroup()
                                                                .addComponent(
                                                                        lblSimilarity)
                                                                .addPreferredGap(
javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                                                                .addComponent(
                                                                        txtSimilarity,
javax.swing.GroupLayout.PREFERRED_SIZE,
                                                                        23,
javax.swing.GroupLayout.PREFERRED_SIZE)
                                                                .addPreferredGap(
javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                                                                .addComponent(
                                                                        lblPerc)
                                                                .addPreferredGap(
javax.swing.LayoutStyle.ComponentPlacement.RELATED)
```

```
                                .addComponent(
                                        lblScale))
                        .addGroup(
                                layout.createSequentialGroup()
                                        .addComponent(
                                                btnCancel)
                                        .addPreferredGap(
javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
                                        .addComponent(
                                                btnOk)
                                        .addGap(3,
                                                3,
                                                3))
                        .addComponent(
                                lstTags,
javax.swing.GroupLayout.Alignment.LEADING,
                                javax.swing.GroupLayout.PREFERRED_SIZE,
                                374,
```

```
        javax.swing.GroupLayout.PREFERRED_SIZE))))
                                                                                .addGap(9, 9,
9)))
                                                .addContainerGap(57, Short.MAX_VALUE)));
                layout.setVerticalGroup(layout
                                .createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                                .addGroup(
                                                javax.swing.GroupLayout.Alignment.TRAILING,
                                                layout.createSequentialGroup()
                                                                .addGap(25, 25, 25)
                                                                .addGroup(
                                                                                layout.createParallelGroup(

        javax.swing.GroupLayout.Alignment.LEADING)
                                                                                                .addGroup(

        layout.createSequentialGroup()
                                                                                .addComponent(

        lblTag)
                                                                                .addGap(18, 18,

        18)
                                                                                .addComponent(

        lblMethod1)

        .addPreferredGap(

        javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
                                                                                .addComponent(

        lblMethod2))
                                                                                .addGroup(

        layout.createSequentialGroup()
                                                                                .addComponent(
```

```
txtTag,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
                                                                                .addGap(18, 18,
18)
                                                                                .addComponent(
boxSimple)
.addPreferredGap(
javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
                                                                                .addComponent(
boxSynonysm)
.addPreferredGap(
javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
                                                                                .addComponent(
boxSubstring)
.addPreferredGap(
javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
                                                                                .addComponent(
boxSimilarity)))
                                                .addGap(10, 10, 10)
                                                .addGroup(
                                                            layout.createParallelGroup(
javax.swing.GroupLayout.Alignment.BASELINE)
```

```
                                                                .addComponent(lblSimilarity)
                                                                .addComponent(
                                                                        txtSimilarity,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
                                                                .addComponent(lblPerc)
                                                                .addComponent(lblScale))
                                                .addGap(18, 18, 18)
                                                .addComponent(btnSearchTag)
                                                .addGap(33, 33, 33)
                                                .addGroup(
                                                        layout.createParallelGroup(
javax.swing.GroupLayout.Alignment.LEADING)
                                                                .addGroup(
layout.createSequentialGroup()
                                                                        .addComponent(
lblTags1)
.addPreferredGap(
javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
                                                                        .addComponent(
lblTags2)
.addPreferredGap(
javax.swing.LayoutStyle.ComponentPlacement.RELATED,
153,
javax.swing.GroupLayout.PREFERRED_SIZE))
                                                                .addGroup(
```

```
javax.swing.GroupLayout.Alignment.TRAILING,
layout.createSequentialGroup()
                                                                                .addComponent(
lstTags,
javax.swing.GroupLayout.PREFERRED_SIZE,
134,
javax.swing.GroupLayout.PREFERRED_SIZE)
.addPreferredGap(
javax.swing.LayoutStyle.ComponentPlacement.RELATED,
35,
Short.MAX_VALUE)
                                                                                .addGroup(
layout.createParallelGroup(
      javax.swing.GroupLayout.Alignment.BASELINE)
      .addComponent(
                  btnOk)
      .addComponent(
                  btnCancel))))
                                          .addGap(21, 21, 21)));
      pack();
}
private javax.swing.JCheckBox boxSimilarity;
```

```
        private javax.swing.JCheckBox boxSimple;
        private javax.swing.JCheckBox boxSubstring;
        private javax.swing.JCheckBox boxSynonysm;
        private javax.swing.JButton btnCancel;
        private javax.swing.JButton btnOk;
        private javax.swing.JButton btnSearchTag;
        private javax.swing.JLabel lblMethod1;
        private javax.swing.JLabel lblMethod2;
        private javax.swing.JLabel lblPerc;
        private javax.swing.JLabel lblScale;
        private javax.swing.JLabel lblSimilarity;
        private javax.swing.JLabel lblTag;
        private javax.swing.JLabel lblTags1;
        private javax.swing.JLabel lblTags2;
        private java.awt.List lstTags;
        private javax.swing.JFormattedTextField txtSimilarity;
        private javax.swing.JTextField txtTag;

}
```

Apêndice U – Classe *ColorUtil*

```
package br.com.xptech.util;

import java.awt.Color;
import java.util.Enumeration;
import tcc.tbfm.model.classesUtils.MIColors;

public class ColorUtil {

	static Color random = new Color(255, 0, 255);

	public static String nameThatColor(Color toName) {
		random = toName;
		double currDiff = 653555D;
		String toString = "desconhecida";
		Enumeration<Color> colors = MIColors.colors.keys();
		while (colors.hasMoreElements()) {
			Color c = colors.nextElement();
			double d = Math.floor(dist(c));
			if (d < currDiff) {
				currDiff = d;
				toString = MIColors.colors.get(c);
			}
		}
		return toString;
	}

	private static double dist(Color cc) {
		int x, y, z;
		x = Math.abs(cc.getRed() - random.getRed());
		y = Math.abs(cc.getGreen() - random.getGreen());
		z = Math.abs(cc.getBlue() - random.getBlue());
		return Math.sqrt(x * x + y * y + z * z);
	}

}
```

Apêndice V – Classe *ImageInfo*

```
package br.com.xptech.util;

import java.awt.Color;
import java.awt.image.BufferedImage;
import java.io.File;
import java.io.IOException;
import java.util.HashMap;
import java.util.Hashtable;
import java.util.Iterator;
import java.util.Map;
import javax.imageio.ImageIO;
import javax.imageio.ImageReader;
import javax.imageio.stream.ImageInputStream;

public class ImageInfo {

	public int[] getRGBArr(int pixel) {
		int alpha = (pixel >> 24) & 0xff;
		int red = (pixel >> 16) & 0xff;
		int green = (pixel >> 8) & 0xff;
		int blue = (pixel) & 0xff;
		return new int[] { red, green, blue, alpha };

	}

	public boolean isGray(int[] rgbArr) {
		int rgDiff = rgbArr[0] - rgbArr[1];
		int rbDiff = rgbArr[0] - rgbArr[2];
		// Filter out black, white and grays...... (tolerance within 10 pixels)
		int tolerance = 10;
		if (rgDiff > tolerance || rgDiff < -tolerance)
			if (rbDiff > tolerance || rbDiff < -tolerance) {
				return false;
			}
```

```
        return true;
}

public Hashtable<String, Integer> colorContents(String imgPath) throws Exception {
        File file = new File(imgPath);
        ImageInputStream is = ImageIO.createImageInputStream(file);
        Iterator<ImageReader> iter = ImageIO.getImageReaders(is);

        if (!iter.hasNext()) {
                System.out.println("Cannot load the specified file " + file);
                System.exit(1);
        }
        ImageReader imageReader = (ImageReader) iter.next();
        imageReader.setInput(is);

        BufferedImage image = imageReader.read(0);
        int height = image.getHeight();
        int width = image.getWidth();

        Map<String, Integer> m = new HashMap<>();
        for (int i = 0; i < width; i++) {
                for (int j = 0; j < height; j++) {
                        int rgb = image.getRGB(i, j);
                        int[] rgbArr = getRGBArr(rgb);
                        Color color = new Color(rgbArr[0], rgbArr[1], rgbArr[2]);
                        String nameColor = ColorUtil.nameThatColor(color);
                        if (m.containsKey(nameColor)) {
                                Integer newVal = m.get(nameColor) + 1;
                                m.remove(nameColor);
                                m.put(nameColor, newVal);
                        } else {m.put(nameColor, 1);}
                }
        }
        long total = 0L;
        for (String f : m.keySet()) {
                total += m.get(f);
        }
        Hashtable<String, Integer> results = new Hashtable<>();
        for (String f : m.keySet()) {
                results.put(f, (int) (((float) m.get(f) / (float) total) * 100));
```

```
		}
		return results;
	}

	public Map<Integer, Integer> imageMap(String path) throws IOException {
		File file = new File(path);
		ImageInputStream is = ImageIO.createImageInputStream(file);
		Iterator<ImageReader> iter = ImageIO.getImageReaders(is);

		if (!iter.hasNext()) {
			System.out.println("Cannot load the specified file " + file);
			System.exit(1);
		}
		ImageReader imageReader = (ImageReader) iter.next();
		imageReader.setInput(is);

		BufferedImage image = imageReader.read(0);
		int height = image.getHeight();
		int width = image.getWidth();

		Map<Integer, Integer> m = new HashMap<>();
		for (int i = 0; i < width; i++) {
			for (int j = 0; j < height; j++) {
				int rgb = image.getRGB(i, j);
				int[] rgbArr = getRGBArr(rgb);
				// Filter out grays....
				if (!isGray(rgbArr)) {
					Integer counter = (Integer) m.get(rgb);
					if (counter == null) counter = 0;
					counter++;
					m.put(rgb, counter);
				}
			}
		}
		return m;
	}
}
```