UNIVERSIDADE PAULISTA

B22816-4 MARCIO FERNANDES CRUZ

B56734-1 DIEGO DA SILVA SANTANA

B303GD-9 THIAGO GUY MOZOL VICENTE

“DESENVOLVIMENTO DE UM SISTEMA DE IDENTIFICAÇÃO E AUTENTICAÇÃO BIOMÉTRICA”

SÃO PAULO

2014

B22816-4 MARCIO FERNANDES CRUZ
B56734-1 DIEGO DA SILVA SANTANA
B303GD-9 THIAGO GUY MOZOL VICENTE

“DESENVOLVIMENTO DE UM SISTEMA DE IDENTIFICAÇÃO E AUTENTICAÇÃO BIOMÉTRICA”

Dados obtidos através dos métodos e aplicações estudados em aulas e com pesquisas para fins de obtenção de nota para o curso de Ciência da Computação 5/6° Semestre (Noturno) da Universidade Paulista (UNIP) sendo entrega ao Professor/Roberto.

Professor Roberto.

Ciência da Computação 5° e 6º Semestre – Noturno

SÃO PAULO
2014

Guia de normalização para apresentação de trabalhos acadêmicos da Universidade Paulista. / Ana Lúcia E. Pires... [et al]. – São Paulo, 2012

OBJETIVO DO TRABALHO

Este trabalho tem como objetivo elaborar as atividades Práticas Supervisionadas (APS), realizando uma pesquisa sobre os principais métodos da biometria, através de pesquisas na internet e disciplinas ministradas no semestre, sendo criado um projeto de software para desenvolver um programa para que possa verificar e comparar os histogramas de imagens dentro de uma base de dados.

Esse software possui uma base de dados interna de imagens e seus histogramas. A partir de uma imagem externa, sem identificação, através da técnica de comparação de histogramas, buscamos a imagem que possui o mesmo histograma e, consequentemente, identificamos qual é o nome do usuário que está relacionado àquela imagem.

Com tudo, o trabalho tem a finalidade de apresentar todo o contexto da biometria e suas tecnologias e o desenvolvimento e as funcionalidades de um programa para a verificação de histograma de imagem.

.

RESUMO

Esse trabalho tem demonstrado o conceito da biometria, suas tecnologias, como possibilitar o reconhecimento e autenticação segura de determinada pessoa utilizando suas características fisiológicas e comportamentais, como impressão digital, reconhecimento da íris, reconhecimento facial, geometria dos dedos e mãos e voz, e a escrita através de padrões de assinatura.

O trabalho apresenta o desenvolvimento de um programa de computador que terá a finalidade de comparar e verificar os histogramas de imagem. Esse software possui uma base de dados interna de imagens, cada uma delas contendo seu histograma diferenciado.

E a partir de uma imagem externa, sem identificação, através da técnica de comparação de histogramas, buscamos a imagem que possui o mesmo histograma e, consequentemente, identificamos qual é o nome do usuário que está relacionado àquela imagem. O programa terá uma interface simples, e haverá vários botões na tela para que o usuário possa utilizar para o funcionamento do software.

ABSTRACT

This work has demonstrated the concept of biometrics, its Technologies, and enable the recognition and secure authentication of a certain person using their physiological and behavioral characteristics such as fingerprint, iris recognition, face recognition, finger and hand geometry and voice, and written by signature patterns.

The paper presents the development of a computer program that will aim to compare and check the histogram image. This software has an internal database of images, each containing its distinctive histogram.

And from an external image without identification, using the technique of comparing histograms, we seek the image that has the same histogram and therefore identify what the user name that is related to that image. The program has a simple interface, and there Will be many buttons on the screen so the user can use to operate the software.

Sumário

1. Introdução 9

2. Fundamentos das principais técnicas biométricas 11

2.1 Principais técnicas biométricas **13**

2.1.1 Impressões Digitais: **13**

2.1.2 Reconhecimento Facial **14**

2.1.3 Geometria dos Dedos e Mãos **15**

2.1.4 Reconhecimento da íris **15**

2.1.5 Assinatura **16**

2.1.6 Reconhecimento por voz **16**

2.2 Segurança **17**

3. Plano de desenvolvimento da aplicação 19

3.1 Histogramas de imagem **19**

3.2 Comandos do aplicativo **21**

4. Projeto, estrutura e módulos que foram desenvolvidos. 25

4.1 MainBiometrico **25**

4.2 MainMenu **25**

4.2.1 Botão sair **25**

4.2.2 Botão de limpar a tela **25**

4.2.3 Botão importar o banco de imagens **26**

4.2.4 Botão exportar planilha **28**

4.3 Controller **29**

4.3.1 Carregar a lista de frequência **30**

4.3.2 Autenticar imagem **30**

5. Listagem do código fonte. 31

6. Referências bibliográficas 56

6.1 Livros **56**

6.2 Acessos de links via Internet **56**

Lista de figuras

Figura 1: imagem e seu histograma.....20

Figura 2: imagem com alto e baixo contraste.....21

Figura 3: Interface do programa.....21

Figura 4: Programa executado.....23

Figura 5: Histograma não localizado.....24

Lista de tabela

Tabela 1: Distribuição horizontal (por finalidade) das principais aplicações biométricas [BITE 2005].....13

1. Introdução

Com o significativo aumento das informações no mundo digital, serviços em rede (e-mail, ERP, Internet, entre outros) e com a necessidade de meios mais seguros de proteção é a motivação para implantar um sistema, em que a identificação e autenticação dos usuários são um dos principais aspectos considerados na garantia da segurança das informações.

Pensando nessa situação os pesquisadores de universidades conhecidas mundialmente perceberam que poderiam utilizar características do ser humano que o identificasse de modo único. Surge, então, a biometria e os sistemas biométricos. Estes sistemas no início de sua utilização eram tão pouco popularizados que o cidadão comum imaginava aquilo como algo futurístico. Hoje, porém está cada vez mais comum seu uso em banco, aeroportos, grandes empresas e em outros ambientes.

A biometria é mais bem definida como sendo as mensurações fisiológicas e as características de comportamento que podem ser utilizadas para verificação de identidade de uma pessoa. Elas incluem impressões digitais, voz, retina, íris, reconhecimento de face, imagem térmica, análise de assinatura, palma da mão e outras técnicas.

Inicialmente estas técnicas eram empregadas em aplicações especializadas de alta segurança, entretanto agora, pode-se notar sua utilização e proposta de uso em uma grande e crescente área de situações em utilizações públicas no nosso dia a dia, com a tecnologia digital custando cada vez menos, possibilitando a introdução no mercado de dispositivos que fazem a autenticação biométrica. As soluções de reconhecimento biométrico podem ter um custo inicial eventualmente superior aos demais meios, porém devido ao fato de dispensar mídias de reconhecimento (chips, cartões magnéticos, cartões inteligentes, crachás...) possuem custos de manutenção e utilização menores, o que viabiliza sua utilização.

Com o objetivo de apresentar neste trabalho o contexto da biometria, serão apresentadas algumas técnicas de biometria mais utilizadas e suas definições. Será desenvolvido um programa de computador que terá a finalidade de comparar histogramas de imagem. Por definição, histograma tem como objetivo de indicar a quantidade de pixel que uma imagem de determinado nível de cor, ou seja, ela faz uma representação gráfica ou numérica do número de pixels e linhas de cor encontrada em uma imagem.

Esse software possui uma base de dados interna de imagens e seus histogramas de todas as imagens. E a partir de uma imagem externa, sem identificação, buscamos a imagem que possui o mesmo histograma dentro da base de dados, e consequentemente, identificamos qual é o nome do usuário que está relacionado àquela imagem.

2. Fundamentos das principais técnicas biométricas

Conceitualmente, biometria é a medida de características únicas do indivíduo que podem ser utilizadas para reconhecer sua identidade (LIU & SILVERMAN, 2001). Este conceito pode ser aplicado à identificação de pessoas, sendo que os padrões físicos ou comportamentais dos indivíduos tornam-se seus métodos particulares de autenticação.

Biometria é o ramo da ciência que estuda a mensuração dos seres vivos (FERREIRA, 2004). “Em termos tecnológicos, refere-se à identificação automatizada de um indivíduo em particular, baseado em suas características físicas ou comportamentais” (GUMZ, 2002).

Os sistemas biométricos são utilizados para a autenticação de pessoas, e são definidos em dois modos de autenticação, sendo a verificação e identificação. Na verificação, a característica biométrica é apresentada pelo usuário juntamente com uma identidade alegada, usualmente por meio da digitação de um código de identificação. Na identificação, o usuário fornece apenas sua característica biométrica, para a identificação do usuário.

Os sistemas biométricos podem basear o seu funcionamento em diversas partes do corpo humano, por exemplo: os olhos, a palma da mão, as digitais do dedo, a retina ou íris dos olhos. A premissa em que se fundamentam é a de que cada indivíduo é único e possuí características físicas e de comportamento diferenciado, traços aos quais são característicos de cada ser humano.

A identificação biométrica pode ser dividida em duas classes principais sendo a Fisiológicas que esta relacionada com a forma do corpo, e a Comportamentais estão relacionados ao comportamento de uma pessoa. Qualquer característica fisiológica ou comportamental humana pode ser usada como característica biométrica desde que ela satisfaça alguns requisitos básicos [Clarke 1994]:

Universalidade: toda a população sendo autenticada deve possuir a característica. Na prática têm pessoas que não possuem impressões digitais.

Unicidade: uma característica biométrica deve ser única para cada pessoa, ou seja, a possibilidade de pessoas distintas possuírem características idênticas deve ser nula ou desprezível. Na prática, as características biométricas podem apresentar maior ou menor grau de unicidade, mas nenhuma delas pode ser considerada absolutamente única para cada indivíduo.

Permanência: a característica deve ser imutável. Na prática, existem alterações ocasionadas pelo envelhecimento, pela mudança das condições de saúde ou mesmo emocionais das pessoas e por mudanças nas condições do ambiente de coleta.

Coleta: a característica tem que ser passível de mensuração por meio de um dispositivo. Na prática, todas as características biométricas utilizadas comercialmente atendem a este requisito.

Aceitação: a coleta da característica deve ser tolerada pelo indivíduo em questão. Na prática, existem preocupações com higiene, com privacidade e questões culturais que diminuem a aceitação da coleta.

As tecnologias biométricas podem ser utilizadas em uma ampla variedade de aplicações, para proporcionar controle de acesso físico e lógico e fornecimento de unicidade. Existe uma taxonomia genérica de aplicações, segundo a qual todas as aplicações podem ser particionadas em sete categorias, pelo menos [Wayman 1999b]. Na prática, as aplicações dos nichos Governamental, Comercial e Forense (classificação vertical) podem ser classificadas por finalidade (classificação horizontal). Dentre os diversos conjuntos possíveis, dependendo do refinamento, é a classificação de alto nível de sete grupos usada no relatório BITE Market Report [BITE 2005]. Mostraremos o exemplo na tabela 1.

Tabela 1: Distribuição horizontal (por finalidade) das principais aplicações biométricas [BITE 2005].

Finalidade	**Utilização**
Identificação Criminal	28%
Controle de acesso e atendimento	22%
Identificação Civil	21%
Segurança de redes e de computadores	19%
Autenticação em pontos de vendas, ATM's e varejo.	4%
Autenticação telefônica e comércio eletrônico	3%
Vigilância e filtragem	3%

Fonte: BITE Market Report [BITE 2005].

2.1 Principais técnicas biométricas

Atualmente existem diversas técnicas biométricas, porem será demonstrado abaixo algumas técnicas principais e as suas finalidades.

2.1.1 Impressões Digitais: A impressão digital é composta por vários sulcos, que em sua formação apresentam diferenças chamadas de pontos de minúcias, ou seja, aquelas partem em que os sulcos se dividem ou onde terminam abruptamente As impressões digitais são únicas para cada dedo de uma pessoa, incluindo os gêmeos idênticos. Umas das tecnologias biométricas mais comercialmente disponível são os dispositivos de reconhecimento de impressões digitais para acesso de desktop e laptop que são agora amplamente disponíveis, a um custo baixo. Com esses dispositivos, os usuários não precisam digitar senhas, apenas um toque oferece acesso instantâneo. Há muitos anos os institutos oficiais de identificação de diversos países já realizam o reconhecimento de pessoas através do sistema de análise da impressão digital. Mostraremos abaixo algumas características da impressão digital.

- Métodos mais rápidos.
- Menor custo.
- Mais difundido.

- Muito confiável.
- Nunca muda.

2.1.2 Reconhecimento Facial: Essa técnica é realizada através de uma câmera digital, que captura as características de uma pessoa e pode ser capturadas por diversas maneiras. O reconhecimento facial em luz visível é tipicamente o modelo das principais características da porção central de uma imagem facial. Usando uma variedade de câmeras, os sistemas de luz visível extraem recursos da imagem capturada que não mudam ao longo do tempo, evitando características superficiais, tais como expressões faciais e cabelo.

Várias abordagens para a modelagem de imagens faciais no espectro visível são Análise de Componentes Principais, marcas locais, redes neurais, teoria dos grafos elástica, e uma análise de multi-resolução. Alguns dos desafios do reconhecimento facial no espectro visual incluem a redução do impacto das variáveis de iluminação e detecção de uma máscara ou uma fotografia. Alguns sistemas de reconhecimento facial podem exigir que um usuário ficasse estático ou imóvel a fim de capturar a imagem, embora muitos sistemas utilizem um processo em tempo real para detectar a cabeça de uma pessoa e localizar automaticamente o rosto.

Os principais benefícios do reconhecimento facial é que ela é não intrusivo, mãos-livres, contínuo e aceito pela maioria dos usuários. O reconhecimento facial usa características faciais distintivas, incluindo contornos superiores das órbitas, áreas em torno de maçãs do rosto, dos lados da boca e da localização do nariz e dos olhos. Mostraremos algumas características de reconhecimento facial.

- Alto custo.
- Menor confiabilidade.
- Maior tempo de leitura e pesquisa.

2.1.3 Geometria dos Dedos e Mãos: Estes métodos de autenticação pessoal estão bem estabelecidos. Para realizar a autenticação pessoal, um sistema pode medir tanto as características físicas dos dedos ou das mãos. Estes incluem o comprimento, largura, espessura e superfície da mão. Uma característica interessante é que alguns sistemas requerem uma pequena amostra biométrica alguns bytes. A geometria da mão vem ganhando aceitação em uma série de aplicações. Ela pode muitas vezes ser encontradas no controle de acesso físico em aplicações comerciais tipo banco e residenciais, no tempo e sistemas de atendimento e, em geral, em pedidos de autenticação pessoal. Estamos acostumados a impressões digitais, mas raramente pensamos em uma mão inteira como um identificador individual. Este método baseia-se em dispositivos que medem a duração e os ângulos de dedos individuais. É um processo mais amigável do que exames de retina, que ainda é complicado. Mostraremos abaixo algumas características da geometria dos dedos e mãos.

- Menor Custo.
- Menos confiável.
- Problemas com anéis.
- Necessita que o usuário encaixe a mão na posição correta.

2.1.4 Reconhecimento da íris: A tecnologia biométrica de reconhecimento pelo padrão da íris é muito confiável, e única para cada pessoa. Embora ela seja relativamente nova, vem se mostrando bastante precisa e estável. Para a aquisição do processo para obtê-la a imagens da íris, os sistemas comerciais utilizam câmeras monocromáticas, já que os métodos de extração de características não se utilizam da cor. A maioria dos sistemas requer que o usuário posicione os olhos dentro do campo de visão de uma câmera de foco estreito.

Para o processo da extração das características da íris para a criação de um Iriscode, funciona da seguinte forma: é localizada a imagem da íris na imagem adquirida, pelo centro da pupila, o padrão da íris é isolado da pupila, o padrão é de modulado para extração de sua formação de fase, quando são computados 256 bytes para a imagem da íris e os outros 256 bytes

representando a máscara para as áreas de ruído, para melhorar a precisão do comparador, perfazendo então um perfil de 512 bytes. Mostraremos abaixo algumas características da íris.

- Alto Custo
- Muito confiável
- Praticamente imutável com o passar dos anos

2.1.5 Assinatura: A assinatura pode ser definida como off-line ou estática aquela que é impostada em documentos de papel, escrita por meio convencional e posteriormente adquirida por meio de uma câmera ou scanner. Pode ser ainda on-line ou dinâmica, atuando num dispositivo eletrônico preparado para capturar, com alta resolução, as características dinâmicas temporais da assinatura, como a trajetória da caneta, a pressão, direção e elevação do traço.

O processo de aquisição pode ser baseado numa abordagem estática ou dinâmica. Várias abordagens de análise automatizada são baseadas em características como número de contornos interiores e número de componentes de inclinação. Entretanto, a falta de informação dinâmica torna o processo automatizado de verificação estática bastante vulnerável a fraudes.

O problema da verificação automática de assinaturas estáticas atraiu grande atenção nos últimos anos, mas os resultados não têm fornecido a precisão requerida por muitos problemas de segurança. A abordagem dinâmica é bem mais interessante. A verificação da assinatura está baseada nas características do processo de assinatura em si.

2.1.6 Reconhecimento por voz: O reconhecimento de voz é um dos sistemas menos invasivos, e a forma mais natural de uso é o sistema de reconhecimento de fala, e esses sistemas podem ser divididos em classes:

Texto fixo: O usuário pronuncia uma palavra ou frase pré-determinada, secreta, gravada durante a fase de registro.

Dependente do texto: O usuário é solicitado, pelo sistema de autenticação, a pronunciar algo específico, dentre as diversas opções

previamente registradas no sistema. Neste caso, a fase de registro é bastante longa. É similar ao protocolo de texto fixo, com um número maior de opções.

Independente do texto - O usuário pronuncia frases conforme seu desejo. O sistema processa qualquer discurso do usuário.

Conversacional: O usuário é interrogado, pelo sistema de autenticação, com perguntas cujas respostas são secretas, tornando-se um protocolo misto de conhecimento e biometria. É um protocolo similar ao dependente de texto, sendo que as frases previamente gravadas possuem certo grau de segredo.

Para o processo de aquisição, existem numerosos transdutores para transformar as ondas acústicas de voz em ondas eletromagnéticas. As quantidades de espaço de armazenamento necessárias para os dados de voz sem tratamento dependem da taxa de amostragem, níveis de quantização e número de canais (mono-canal na maioria das vezes).

Para a aplicação de ferramentas matemáticas, sem perda de generalidade, o sinal de voz deve ser representado por uma sequencia de vetores de características. O processo de extração pode se basear: na abordagem tradicional, por meio de PCA (Principal Component Analysis) e FA (Factor Analysis); na abordagem de estimativa de médias e covariâncias; e na estimativa de divergências [Campbell 1997]. O processo de comparação das características extraídas pode ser suportado por vários métodos.

2.2 Segurança

A segurança dos sistemas biométricos pode ser diferenciada em três aspectos importante.

- A precisão do sistema, representada pelas medidas clássicas estatísticas de taxas de falsa aceitação e falsa rejeição.
- A arquitetura do sistema e implementação do sistema em si, representada pela interconexão física e lógica entre suas diversas partes de componentes e a aplicação;
- E a robustez do sistema, representada pela sua capacidade de resistência à fraude e falsificação intencionais.

As precisões poderão ser avaliadas por meio de bancos de dados representativos e um conjunto básico de medidas aceitas. Com a arquitetura do sistema, existem procedimentos, embora sejam mais complexos, para avaliar a segurança de um projeto e a sua implementação de uma maneira padronizada. No entanto, a robustez é a mais difícil de ser avaliada, pois é fácil mostrar que um sistema biométrico pode ser fraudado, mas é muito mais difícil mostrar que um sistema biométrico não pode ser fraudado. Assim, independentemente de quão preciso é o sistema e de quão bem projetada é a sua arquitetura, não se pode enunciar de antemão conclusões sobre a sua resistência a ataques.

3. Plano de desenvolvimento da aplicação

A primeira questão a ser avaliada antes de ser traçado o caminho para o desenvolvimento do aplicativo foi traçar o cenário do aplicativo dentro do contexto da biometria. Foi traçado um plano em criar um software para comparar histograma de imagem. O software desenvolvido possui armazenado em um arquivo meta informações de imagens em disco, ou seja, o nome do arquivo e seu histograma. Isso foi feito para facilitar a busca e comparação de imagens.

Este aplicativo é multiplataforma e, o seu funcionamento consiste em arrastar uma imagem qualquer para uma área pré-definida no programa, em seguida, é feito um processamento para capturar o histograma dessa imagem e, posteriormente, buscar na base de dados dos histogramas da imagem.

3.1 Histogramas de imagem

Para o desenvolvimento deste aplicativo usaremos uma técnica chamada histograma. Segundo Silva (2001), em processamento de imagens, trabalha sempre com os tons de cinza (digital numbers ou DNs), localizado aos pixels de uma imagem. Histograma pode ser uma maneira simples de representar a distribuição do DNs de uma imagem. Ele fornece o percentual de pixels que a imagem tem de determinado nível de cinza ou cor.

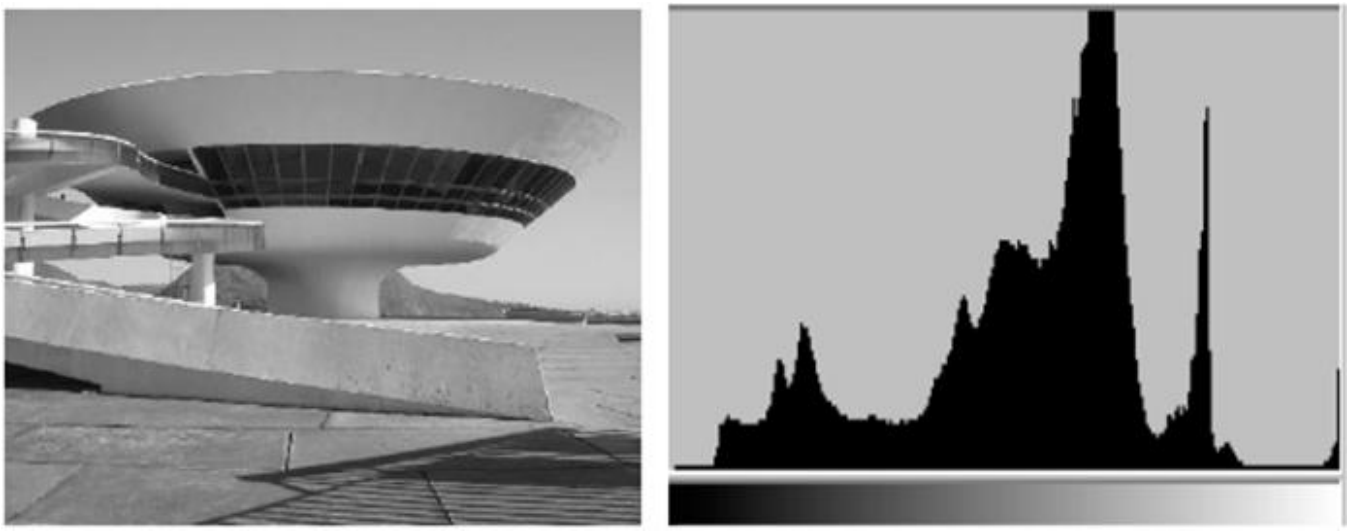
Cada pixel de uma imagem tem uma cor que foi produzida por uma combinação de cores primárias (vermelho verde, e azul do modelo RGB). Cada uma dessas cores pode ter um brilho que varia de 0 a 255 em uma imagem digital com profundidade de 8-bits. Um histograma RGB é produzido quando o computador varre a imagem em cada um desses valores de brilho RGB e conta quantos pixels tem em cada nível de 0 a 255.

O histograma também pode ser conhecido como a distribuição de intensidade e função de densidade de probabilidade, nesse caso é achar a probabilidade de achar um DNs de um dado valor dentro de uma imagem. Podemos observar outro aspecto importante em histograma, é que eles

representam dados digitais, também poderão ser chamados de discreto, que é representado por coluna discreta à distribuição de intensidade, e não podendo ser dividida ou quebrada correspondente a números inteiros. Esse conceito trata de realce de contraste em imagens.

Para que podemos ter uma visão melhor do que é um histograma de imagem, mostraremos um exemplo na figura 1.

Figura 1: imagem e seu histograma.

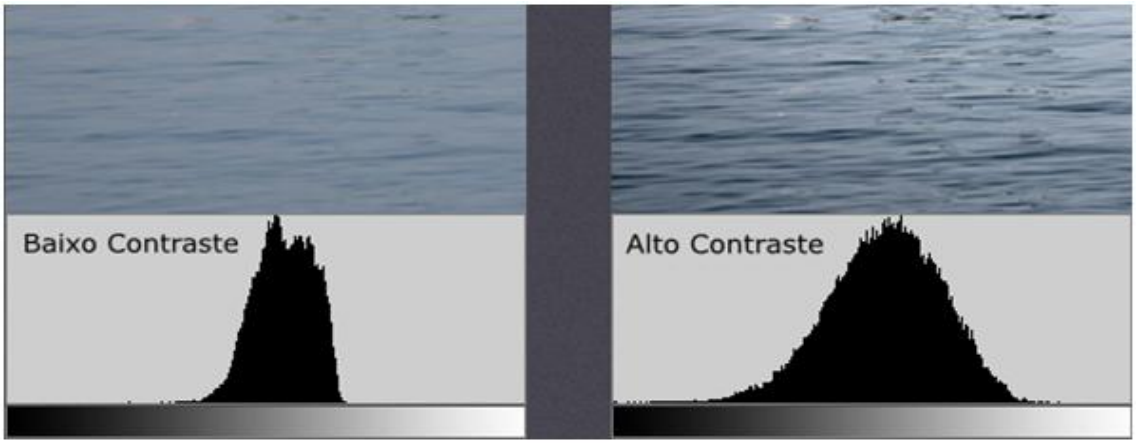


Fonte: http://saberinfo.comunidades.net/index.php?pagina=1155014242_04. Acesso em 22/09/2014.

Em uma imagem temos duas propriedades que podemos observar, o seu brilho que indica se uma imagem é mais clara ou escura e o contraste onde indica se há uma maior ou menor variedade de tons. Contraste é uma diferença de brilho entre as áreas claras e escuras de uma imagem. Histograma largo é típico de uma imagem com bastante contraste, enquanto histograma estreito é de uma imagem com menos contraste. Isso pode ser causado por uma combinação de fatores de luz e sujeito.

Mostraremos um exemplo na figura 2, uma imagem com alto e baixo contraste, onde há uma diferença de histograma entre as imagens.

Figura 2: imagem com alto e baixo contraste.

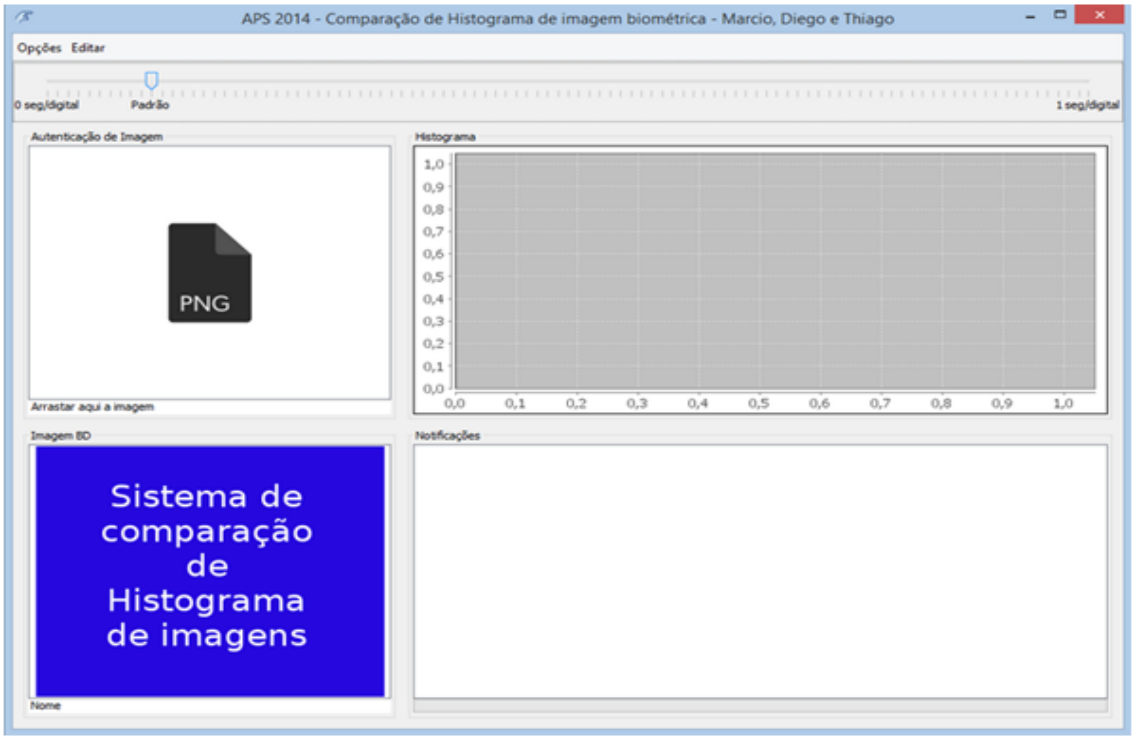


Fonte: http://www.cambridgeincolour.com/pt-br/tutorials/histograms1.htm. Acesso em 30/09/2014.

3.2 Comandos do aplicativo

Serão descrito os comandos que o usuário utilizará nessa aplicação. Na figura 3 que mostraremos abaixo, será a interface do programa utilizada pelo usuário.

Figura 3: Interface do programa.



Fonte: Adaptação própria.

Como vimos na figura 3, na parte superior da tela do programa temos o menu com dois comando, sendo opções e editar . O usuário clicando em opções, em seguida o programa mostrará três comando que mostraremos abaixo.

- ✓ Importar banco de imagens: Terá com sua principal finalidade de fazer as importações das imagens que estão armazenadas na base de dados do software.
- ✓ Exportar planilha eletrônica: Terá com sua principal finalidade de gera um arquivo em Excel identificando qual é o nível de acesso para cada imagem.
- ✓ Sair: Terá como sua principal finalidade de fechar a tela da interface do programa.
- ✓ No menu editar, temos um comando limpar tela: Tem como sua principal finalidade de limpar todos os dados que foi gerado no quadro de notificações.

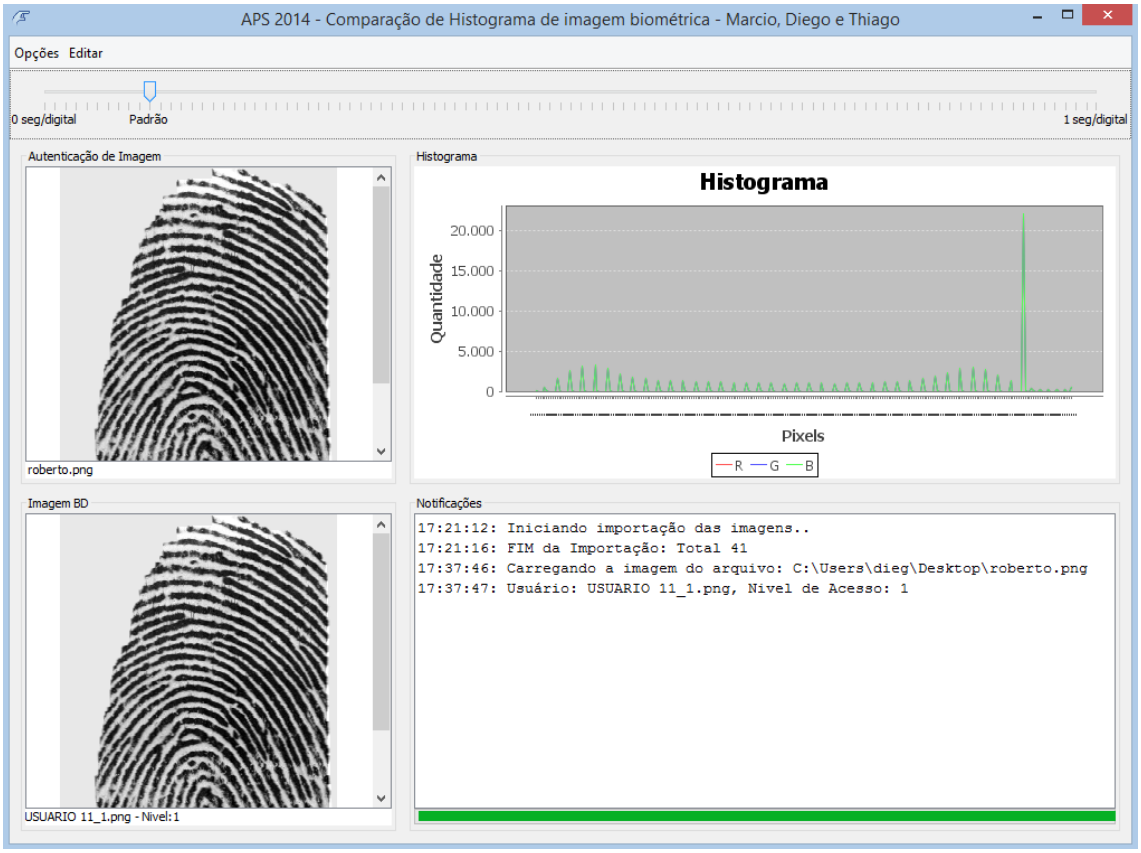
Na interface do programa foram criadas quatro áreas sendo, autenticação de imagem, imagem BD, histograma e notificações, mostraremos abaixo a função de cada um.

- ✓ Autenticação de imagem: É a área onde o usuário ira mover uma imagem externa para esse quadro, e automaticamente é feito a captura do histograma da imagem.
- ✓ Imagem BD: Após a comparação, se a imagem for encontrada, o programa ira mostrar a imagem idêntica que está localizada na base de dados nesta área, caso não encontrar ira mostrar uma mensagem de erro.
- ✓ Histograma: Tem como sua principal finalidade de mostrar o gráfico do histograma das imagens.
- ✓ Notificações: Têm como sua principal finalidade de armazena todas as informações dos comandos que o usuário utilizar no programa.

Na utilização do aplicativo, primeiramente o usuário terá que fazer a importação das imagens que esta localizada na base de dados do programa, após fazer essa importação, mostrara duas mensagens na área de notificações sendo iniciando importação das imagens e fim da importação com a quantidade de imagens importada. Em seguida o usuário ira mover uma imagem qualquer para uma área pré-definida chamada autenticação de imagem e em seguida o software faz o processamento para capturar o histograma dessa imagem e compara com o histograma de todas as imagens que esta localizada na base de dados.

Se a imagem for encontrada o programa ira mostra a imagem que esta localizada na base de dados na área do programa imagem BD. Para ter uma visão melhor, mostraremos na figura 4 um exemplo após ser feito a comparação.

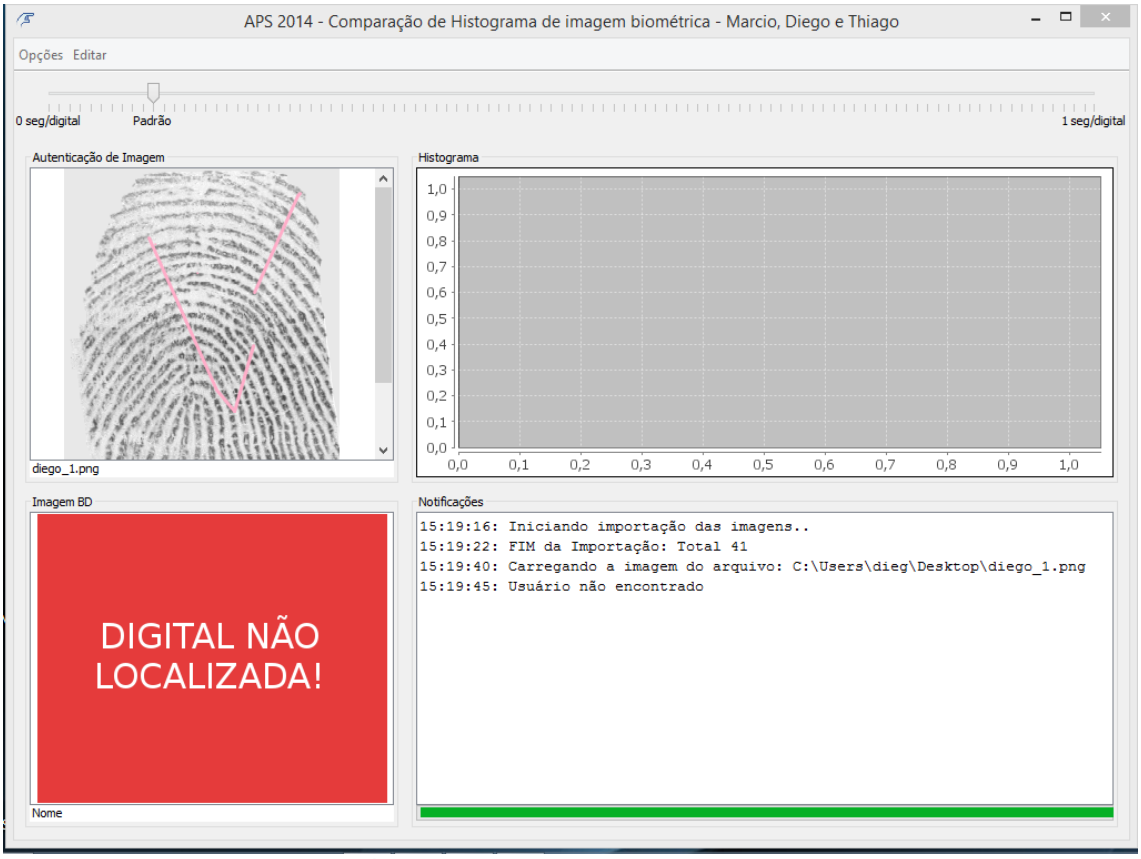
Figura 4: Programa executado.



Fonte: Adaptação própria.

A imagem que o usuário estiver utilizando para fazer a comparação do histograma, com as imagens que esta localizada na base de dado, caso não encontre o histograma idêntico dessa imagem, o programa mostra uma mensagem de erro. Mostraremos na figura 5, a interface do programa executado não encontrando o histograma idêntico da imagem.

Figura 5: Histograma não localizado.



Fonte: Adaptação própria.

4. Projeto, estrutura e módulos que foram desenvolvidos.

4.1 MainBiometrico

- Classe Principal

- Responsável por iniciar o programa.

- Inicia o MainMenu.

4.2 MainMenu

- Estende de JFrame, ou seja, ele herda tudo o que um frame (quadro) tem, e por tanto, apresenta a janela do programa.

Possuí um construtor que:

- Inicializa, customiza e adiciona os componentes (painéis, botões, etc.) no frame.
- Inicializa os layouts.
- Adiciona os ouvintes (listeners) em cada componente especifico que ficarão ouvindo os eventos das ações do usuário.
- Inicia o Controlador (Controller).

4.2.1 Botão sair

- Sai do programa.

4.2.2 Botão de limpar a tela

- Remove o arquivo da imagem e o seu nome no painel.
- Adiciona as imagens padrão.
- Inicia um histograma novo.
- Limpa a área de notificação.
- Inicia do zero a barra progressiva.

Deslizador de controle de tempo de busca:

- Fica ouvindo eventos de quando é deslizado.
- Se for deslizado, então o valor dele é guardado na variável “TempoDeBusca”, ou seja, teremos como referencia o valor do tempo de busca sempre que ocorrer o evento.

4.2.3 Botão importar o banco de imagens

- Acessa o Método “gerarBancoDeDadosImagem” através do Controlador.
- Notifica sua inicialização.
- Cria uma nova classe “GerarListaFrequencia” e, chama o método “getFrequencias”, ou seja, através desta classe pegamos as frequências.
- Cria um arquivo através do diretório já definido na classe “FuncoesGerais”, através do método “getPathDigitais”.
- É recebida uma lista de imagens.
- Cria uma lista de frequência.
- Inicia um loop com iterações de acordo com o tamanho da lista de arquivos.
- Verifica a cada iteração se a imagem é do formato “PNG”.
- Caso seja válido, então é adicionada uma nova frequência à lista de frequências.
- Na Classe frequência é passado por parâmetro o diretório com o nome da imagem.
- A partir do diretório completo é criada uma classe “Imagem”.
- É armazenado o nome do arquivo, a frequência da imagem, o nível de acesso que no caso é gerado aleatório.
- É criado um arquivo a partir do diretório completo.
- É armazenado o nome do usuário como “USUARIO” concatenado com o nome do arquivo.
- A Classe imagem é responsável por guardar suas frequências em matrizes do tipo de números inteiros, tais como, o RGB, HSV e o YUV.

- RGB possui uma dimensão de 256x3 já definida.
- HSV e YUV possuem a dimensão da largura da imagem multiplicada pela altura da imagem por três (largura*alturax3).
- Cria um Buffer (área de armazenamento temporária na memória à espera de entrada e saída) da imagem a partir de uma leitura do arquivo criado pelo diretório da imagem.
- Chama o método “CalculaFrequenciaDaImagem” passando o Buffer da imagem.
- Cria um índice.
- Inicia um loop com iterações de acordo com o tamanho da largura da imagem.
- Inicia um loop dentro com iterações de acordo com o tamanho da altura da imagem.
- O índice é incrementado a cada iteração da altura da imagem.
- Ou seja, será percorrida a matriz toda da imagem, pixel por pixel.
- É pego a cor do pixel pelo método “getRGB” do Buffer de imagem, onde é passado por parâmetro a linha e coluna atual da iteração.
- Através da cor, é pego de cada cor do RGB um número inteiro que representa a quantidade de tom de vermelho, verde e azul.
- Para cada coluna da matriz RGB (3 colunas) é incrementado uma unidade, sendo as linhas de acordo com o inteiro recebido para cada cor do RGB.
- É criado um RGB para o HSV, passando por parâmetro a quantidade de cada tom de cor do RGB e um Array do HSV no tamanho do índice.
- É criado um RGB para o YUV, passando por parâmetro a quantidade de cada tom de cor do RGB e um Array do YUV no tamanho do índice.
- Após a gerar a lista de frequência das imagens, ela é retornada, cabendo ao Controlador guarda-la.
- É criada uma classe “GravaFrequencia” (que estende de “GravaArquivo”) passando o caminho do arquivo de registro de frequências (o caminho já está definido na classe “FuncoesGerais”).
- É chamado o método “AbrirArquivo”, onde será criado e aberto o arquivo de acordo com o caminho já recebido.

- É chamado o método “SetLista”, onde é passado a lista de frequência, e em seguida, inicia um loop com iterações de acordo com a lista.
- A cada iteração é escrito um objeto de frequências dentro do arquivo de registro de frequências (.dat).
- O arquivo é preenchido com todas as frequências da lista criada anteriormente.
- O arquivo é fechado.
- É notificado o fim da importação e a quantidade de frequências criadas.

4.2.4 Botão exportar planilha

- Abre um dialogo pra escolher o destino da exportação.
- Faz algumas configurações no dialogo.
- É feita algumas verificações no diretório e extensão digitada.
- A extensão deve ser apenas em “xlsx”.
- Caso passe pelas validações é pego o caminho a ser exportado.
- É chamado o método “ExportarPlanilha” pelo Controlador, passando o caminho do arquivo.
- É criado uma classe “ExportaPlanilhaFrequencia”, passando a lista de frequência e o caminho do arquivo.
- Logo de inicio é criada uma planilha eletrônica com o nome de “Digitais importadas no banco”.
- Nesta classe é criada uma arvore de String (id do dado) e um Array de objetos (arquivo, usuário e nível de acesso).
- É criado um contador, iniciando-se no índice dois, pois no primeiro já foram inseridos os títulos (id, arquivo, usuário e nível de acesso);
- Inicia um loop com iterações de acordo com o tamanho da lista de frequências.
- A cada iteração é adicionado na arvore o índice, nome do arquivo, nome de usuário e nível de acesso da lista de frequências.
- As chaves da arvore são atribuídas a uma lista que não permite a repetição de elementos.
- Inicia um loop com iterações de acordo com o tamanho da nova lista.

- É pego e criada uma linha na planilha eletrônica.
- É pego um Array de objetos da arvore de acordo com a chave atual da iteração.
- Inicia um loop dentro com iterações de acordo com a quantidade de objetos do Array.
- É pego e criado uma célula na linha da planilha eletrônica.
- É verificado se o objeto atual da iteração é uma String ou um Integer.
- Caso seja uma String, o objeto é inserido como texto.
- Caso seja um Integer, objeto é inserido como um número inteiro.
- É criado um arquivo de acordo com o nome e caminho da planilha sugerido pelo dialogo.
- O arquivo da planilha eletrônica é colocado em uma classe responsável pela saída do arquivo, ou seja, o arquivo será escrito/criado depois de chamado o método “write” desta classe.
- O arquivo é criado.
- É fechada a corrente da classe de saída de arquivo.
- É notificado o caminho no qual o arquivo foi exportado.

4.3 Controller

- Responsável por:

- Guardar a lista de frequências.
- Controlar tudo, tendo acesso ao “MainMenu”.
- Fazer a ponte entre a “View” (área da interface do usuário) e o “Model” (modelos, por exemplo, a lista de frequências).

- Funções/Métodos:

- Gerar o banco de imagens.
- Carregar a lista de frequência.
- Exportar a planilha.
- Autenticar a imagem.

4.3.1 Carregar a lista de frequência

- Cria uma classe “LerFrequencia” que estende de “LerArquivo”, passando o caminho do arquivo de registro de frequências a partir da classe “FuncoesGerais”.
- Verifica se o arquivo existe.
- Se existir, então o arquivo é aberto por uma corrente e referenciada.
- Inicia um loop com iterações infinitas.
- Para cada iteração é adicionada a uma lista de frequência um objeto de frequência lida no arquivo.
- Quando apresentar uma exceção de fim de arquivo ou fim da corrente, a lista é retornada.
- Caso seja uma exceção diferente, então é notificado o erro.
- A corrente é fechada.
- Se não existir o arquivo no disco, então é retornada uma lista vazia de frequência.
- É notificado o total de registros de digitais encontradas.

4.3.2 Autenticar imagem

- É necessário que seja passado o arquivo para que se inicie.
- É criada uma frequência do arquivo passado.
- É iniciada a classe “ProcessoDeAutenticacao” por meio de um processo paralelo (Thread) para que se possa visualiza-la seu progresso durante a comparação entre todos os demais registros.
- É passada a frequência para o processo de autenticação.
- São desativados os comandos durante o processo.
- Inicia um loop com iterações de acordo com a lista de frequência.
- Para cada iteração a barra progressiva (loading) é alterada para a porcentagem estimada de busca (a iteração é multiplicada por cem e dividida pelo total da lista de frequência).
- É alterada a imagem do painel de imagens banco para a imagem atual da lista de frequência.

- É alterado o histograma do painel de histograma, onde é passada a frequência, e a partir dela, é criado um novo histograma.
- Verifica se o arquivo passado é igual ao atual da iteração.
- Na comparação por meio de uma classe de manipulação de Arrays é verificada profundamente a frequência de RGB de cada.
- Caso sejam iguais, então no painel de imagens do banco é inserido o nome e nível do arquivo de frequência.
- É notificado o nome de usuário e nível de acesso.
- A barra progressiva é passada para cem por cento.
- O loop termina.
- Caso o loop continue, então para cada iteração é aguardado o tempo de busca definido pelo deslizador.
- Caso não seja encontrada uma imagem igual, então o painel de histograma é renovado, o painel de imagens do banco é alterado para uma imagem de não encontrado e é notificado que não foi encontrado.
- Os botões de comando são ativados novamente.
- O processo de autenticação termina.

5. Listagem do código fonte.

```
package br.com.aps.biometria.controller;

import java.io.File;
import java.io.IOException;
import java.util.ArrayList;
import java.util.List;

import br.com.aps.biometria.model.*;
import br.com.aps.biometria.tipos.Frequencia;

public class Controller {

	private List<Frequencia> listaFrequencia;
	private PainelNotificacao painelNotificacao;
	private PainelImagemDB painelImagemDB;
	private PainelHistograma painelHistograma;
	private MainMenu mainMenu;

	public int getQuantidadeDigitais() {
		return listaFrequencia.size();
	}
```

```
	public void setPainelNotificacao(PainelNotificacao
painelNotificacao) {
		this.painelNotificacao = painelNotificacao;
	}

	public void setPainelImagemDB(PainelImagemDB painelImagemDB) {
		this.painelImagemDB = painelImagemDB;
	}

	public PainelNotificacao getPainelNotificacao() {
		return painelNotificacao;
	}

	public void setPainelHistograma(PainelHistograma
painelHistograma) {
		this.painelHistograma = painelHistograma;
	}

	public void setMainMenu(MainMenu mainMenu) {
		this.mainMenu = mainMenu;
	}

	public void gerarBancoDeDadosImagem() throws Exception {
		painelNotificacao.addText("Iniciando importação das
imagens..");

		// pegando imagens do disco e jogando num List
		this.listaFrequencia = new
GerarListaFrequencia().getFrequencias();

		// pegando do list e enviando para um .dat no disco
		GravaFrequencia grava = new GravaFrequencia();
		grava.abrirArquivo();
		grava.setLista(listaFrequencia);
		grava.fecharArquivo();

		painelNotificacao.addText("FIM da Importação: Total "
				+ getQuantidadeDigitais());
	}

	public void carregarListaFrequencia() throws Exception {
		LerFrequencia ler = new LerFrequencia();

		if (ler.existeArquivo()) {
			ler.abrirArquivo();
			listaFrequencia = ler.getLista();
			ler.fecharArquivo();
		} else
			listaFrequencia = new ArrayList<Frequencia>();

		painelNotificacao.addText("Total de registros de digitais:
"
				+ getQuantidadeDigitais());
	}

	public void exportarPlanilha(String arquivo) throws IOException
{
		new ExportaPlanilhaFrequencia(listaFrequencia, arquivo);
		painelNotificacao.addText("Arquivo de usuários exportado
para: "
				+ arquivo);
```

```
	}

	public void autenticarImagem(File arquivoAutenticacao) throws
IOException {
		Frequencia frequenciaArquivoAutenticacao = new Frequencia(
				arquivoAutenticacao.getPath());
		new Thread(new
ProcessoDeAutenticacao(frequenciaArquivoAutenticacao))
				.start();
	}

	public class ProcessoDeAutenticacao implements Runnable {

		private Frequencia frequenciaArquivoAutenticacao;

		public ProcessoDeAutenticacao(Frequencia
frequenciaArquivoAutenticacao) {
			this.frequenciaArquivoAutenticacao =
frequenciaArquivoAutenticacao;
		}

		@Override
		public void run() {

			mainMenu.ativarComandos(false);
			boolean encontrou = false;
			int i = 1, total = listaFrequencia.size();

			for (Frequencia registroBase : listaFrequencia) {
				painelNotificacao.setLoadingValue((i * 100) /
total);

	painelImagemDB.addImagem(registroBase.getNomeArquivo());

	painelHistograma.setFrequenciaImagem(registroBase);

				if
(frequenciaArquivoAutenticacao.equals(registroBase)) {

	painelImagemDB.setNome(registroBase.getNomeUsuario()
						+ " - Nivel:" +
registroBase.getNivelAcesso());
				painelNotificacao.addText("Usuário: "
						+
registroBase.getNomeUsuario()
						+ ", Nivel de Acesso: "
						+
registroBase.getNivelAcesso());


	painelHistograma.setFrequenciaImagem(registroBase);
					painelNotificacao.setLoadingValue(100);
					encontrou = true;
					break;
				}

				i++;
				try {
					Thread.sleep(mainMenu.getTempoDeBusca());
				} catch (InterruptedException e) {
					e.printStackTrace();
```

```
                        }
                  }

                  if (!encontrou) {
                        painelHistograma.clear();
                        painelNotificacao.addText("Usuário não
encontrado");

      painelImagemDB.addImagem(FuncoesGerais.getImagemNaoLocalizada())
;
                  }

                  mainMenu.ativarComandos(true);
                  mainMenu.repaint();
            }
      }

}package br.com.aps.biometria.model;

import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.util.List;
import java.util.Map;
import java.util.Set;
import java.util.TreeMap;

import org.apache.poi.ss.usermodel.Cell;
import org.apache.poi.ss.usermodel.Row;
import org.apache.poi.xssf.usermodel.XSSFSheet;
import org.apache.poi.xssf.usermodel.XSSFWorkbook;

import br.com.aps.biometria.tipos.Frequencia;

public class ExportaPlanilhaFrequencia {

      XSSFWorkbook workbook = new XSSFWorkbook();
      XSSFSheet sheet = workbook.createSheet("Digitais importadas no
Banco");

      public ExportaPlanilhaFrequencia(List<Frequencia>
listaFrequencia,
                  String nomePlanilha) throws IOException {
            super();

            Map<String, Object[]> data = new TreeMap<String,
Object[]>();

            data.put("1", new Object[] { "Arquivo", "Usuario", "Nivel
Acesso" });

            Integer cont = 2;

            for (int i = 0; i < listaFrequencia.size(); i++) {
                  data.put(cont.toString(), new Object[] {
                              listaFrequencia.get(i).getNomeArquivo(),
                              listaFrequencia.get(i).getNomeUsuario(),
                              listaFrequencia.get(i).getNivelAcesso()
});

                  cont++;
```

```
            }

            // Iterate over data and write to sheet
            Set<String> keyset = data.keySet();
            int rownum = 0;
            for (String key : keyset) {
                  Row row = sheet.createRow(rownum++);
                  Object[] objArr = data.get(key);
                  int cellnum = 0;
                  for (Object obj : objArr) {
                        Cell cell = row.createCell(cellnum++);
                        if (obj instanceof String)
                              cell.setCellValue((String) obj);
                        else if (obj instanceof Integer)
                              cell.setCellValue((Integer) obj);
                  }
            }

            FileOutputStream out = new FileOutputStream(new
File(nomePlanilha));
            workbook.write(out);
            out.close();
      }

}
package br.com.aps.biometria.file;

import java.awt.datatransfer.DataFlavor;
import java.awt.datatransfer.Transferable;
import java.awt.datatransfer.UnsupportedFlavorException;
import java.io.File;
import java.io.IOException;
import java.util.List;

import javax.swing.JOptionPane;
import javax.swing.TransferHandler;

public class FileTransfer extends TransferHandler {

      private static final long serialVersionUID = 1L;

      private File file;
      private boolean fileOnDrag, dragEnabled;

      public FileTransfer() {
            fileOnDrag = false;
            dragEnabled = true;
      }

      @Override
      public boolean canImport(TransferSupport fileDrag) {
            if (!fileDrag.isDrop() || !isDragEnabled())
                  return false;

            return
fileDrag.isDataFlavorSupported(DataFlavor.javaFileListFlavor);
      }

      @Override
      public boolean importData(TransferSupport fileDrag) {
```

```
            fileOnDrag = true;

            if (!canImport(fileDrag))
                  return false;

            try {
                  Transferable transferencia =
fileDrag.getTransferable();
                  Object dadosTransferencia = transferencia

      .getTransferData(DataFlavor.javaFileListFlavor);

                  @SuppressWarnings("unchecked")
                  List<File> itens = (List<File>) dadosTransferencia;

                  if (!itens.get(0).getName().endsWith(".png")) {
                        JOptionPane.showMessageDialog(null,
                                    "Extensões permitidas: \n- PNG ",
                                    "Autenticação de Imagem",
JOptionPane.WARNING_MESSAGE);
                        fileOnDrag = false;
                        return false;
                  }

                  if (itens.get(0).length() > 2097152) {
                        JOptionPane.showMessageDialog(null, "- O limite
máximo é de "
                                    + "2 MB.", "Autenticação de
Imagem",
                                    JOptionPane.WARNING_MESSAGE);
                        fileOnDrag = false;
                        return false;
                  }

                  file = itens.get(0);

            } catch (UnsupportedFlavorException e) {
                  return false;
            } catch (IOException e) {
                  return false;
            }

            return true;
      }

      public File getFile() {
            return file;
      }

      public void removeFile() {
            file = null;
      }

      public boolean isFileOnDrag() {
            return fileOnDrag;
      }

      public void setFileOnDrag(boolean fileOnDrag) {
            this.fileOnDrag = fileOnDrag;
      }
```

```
	public boolean isDragEnabled() {
		return dragEnabled;
	}

	public void setDragEnabled(boolean dragEnabled) {
		this.dragEnabled = dragEnabled;
	}

}package br.com.aps.biometria.tipos;

import java.io.File;
import java.io.IOException;
import java.io.Serializable;
import java.util.Arrays;

public class Frequencia implements Serializable {

	@Override
	public int hashCode() {
		final int prime = 31;
		int result = 1;
		result = prime * result + Arrays.hashCode(frequenciaRGB);
		return result;
	}

	@Override
	public boolean equals(Object obj) {
		if (this == obj)
			return true;
		if (obj == null)
			return false;
		if (getClass() != obj.getClass())
			return false;
		Frequencia other = (Frequencia) obj;
		if (!Arrays.deepEquals(frequenciaRGB, other.frequenciaRGB))
			return false;
		return true;
	}

	private static final long serialVersionUID = -
1969889687214191 03L;

	private String nomeArquivo;
	private String nomeUsuario;
	private int[][] frequenciaRGB;
	private int nivelAcesso; // 1 a 3

	public int getNivelAcesso() {
		return nivelAcesso;
	}

	public String getNomeUsuario() {
		return nomeUsuario;
	}

	public String getNomeArquivo() {
		return nomeArquivo;
	}

	public Frequencia(String nomeArquivo) throws IOException {
		Imagem imagem = new Imagem(nomeArquivo);
```

```
            this.nomeArquivo = nomeArquivo;
            this.frequenciaRGB = imagem.getFrequencia();
            this.nivelAcesso = (1 + (int) (Math.random() * 3));

            File file = new File(nomeArquivo);
            this.nomeUsuario = "USUARIO " + file.getName();
      }

      public int[][] getFrequenciaRGB() {
            return frequenciaRGB;
      }

      @Override
      public String toString() {
            return "Frequencia [nomeArquivo=" + nomeArquivo + ",
frequenciaRGB="
                        + Arrays.toString(frequenciaRGB) + "]";
      }

}
package br.com.aps.biometria.model;

public class FuncoesGerais {

      private static String pathSistema =
System.getProperty("user.dir") + "\\";

      private static String pathBaseDados = pathSistema +
"basedados\\";
      private static String pathDigitais = pathSistema + "digitais\\";
      private static String pathImage = pathSistema + "image\\";

      public static String getPathImage() {
            return pathImage;
      }

      public static String getPathSistema() {
            return pathSistema;
      }

      public static String getNomeArquivoRegistroFrequencia() {
            return pathBaseDados + "registrofrequencia.dat";
      }

      public static String getPathDigitais() {
            return pathDigitais;
      }

      public static String getImagemNaoLocalizada() {
            return pathImage + "naolocalizada.png";
      }

      public static String getImagemPadrao() {
            return pathImage + "telapadrao.png";
      }

      public static String getImagemArrastar() {
            return pathImage + "arrastar.png";
      }
```

```
	public static String getLogo() {
		return pathImage + "logo.png";
	}

}
package br.com.aps.biometria.model;

import java.io.File;
import java.io.IOException;
import java.util.ArrayList;
import java.util.List;

import br.com.aps.biometria.tipos.Frequencia;

public class GerarListaFrequencia {

	private List<Frequencia> frequencias;

	public List<Frequencia> getFrequencias() {
		return frequencias;
	}

	public GerarListaFrequencia() throws IOException {
		super();
		this.frequencias = gerarArrayImagens();
	}

	private List<Frequencia> gerarArrayImagens() throws IOException
{

		File diretorioImagens = new
File(FuncoesGerais.getPathDigitais());

		String[] listaArquivos = diretorioImagens.list();

		List<Frequencia> frequencias = new ArrayList<Frequencia>();

		for (int i = 1; i < listaArquivos.length; i++) {

			if
(listaArquivos[i].toString().toUpperCase().endsWith("PNG"))
				frequencias.add(new
Frequencia(FuncoesGerais.getPathDigitais()
					+ listaArquivos[i].toString()));
		}
		return frequencias;
	}

}
package br.com.aps.biometria.model;

import java.io.FileOutputStream;
import java.io.IOException;
import java.io.ObjectOutputStream;

public class GravaArquivo {
	protected ObjectOutputStream output;
	private String nomeArquivo;

	public GravaArquivo(String nomeArquivo) {
		super();
```

```
            this.nomeArquivo = nomeArquivo;
      }

      public void abrirArquivo() throws IOException {
            FileOutputStream file = new FileOutputStream(nomeArquivo);
            output = new ObjectOutputStream(file);
      }

      public void fecharArquivo() throws IOException {
            if (output != null)
                  output.close();
      }

}
package br.com.aps.biometria.model;

import java.io.IOException;
import java.util.List;

import br.com.aps.biometria.tipos.Frequencia;

public class GravaFrequencia extends GravaArquivo {

      public GravaFrequencia() {
            super(FuncoesGerais.getNomeArquivoRegistroFrequencia());
      }

      public void setLista(List<Frequencia> lista) throws IOException
{
            for (int i = 0; i < lista.size(); i++)
                  output.writeObject(lista.get(i));
      }

}
package br.com.aps.biometria.model;

import org.jfree.chart.JFreeChart;
import org.jfree.chart.ChartFactory;
import org.jfree.chart.plot.PlotOrientation;
import org.jfree.data.category.DefaultCategoryDataset;

import br.com.aps.biometria.tipos.Imagem;

public class Histograma {

      private JFreeChart grafico;
      private int[][] frequencia = new int[256][3];
      private Imagem foto;

      public Histograma(Imagem img) {
            this.foto = img;
            this.grafico = this.criaChart();
      }// fim do construtor

      public JFreeChart getChart() {
            return this.grafico;
      }

      private JFreeChart criaChart() {
            this.frequencia = this.foto.getFrequencia();
```

```
            DefaultCategoryDataset dataset = new
DefaultCategoryDataset();

            // insere os valores no dataset para carregar no grafico
            for (int x = 0; x < this.frequencia.length; x++) {
                  String v2 = String.valueOf(x);
                  // String v2 = x == 0 || 16 % x == 0?
String.valueOf(x) : "";
                  dataset.addValue(frequencia[x][0], "R", v2);
                  dataset.addValue(frequencia[x][1], "G", v2);
                  dataset.addValue(frequencia[x][2], "B", v2);
            }

            JFreeChart chart =
ChartFactory.createLineChart("Histograma", // chart

                                    // title
                        "Pixels", // domain axis label
                        "Quantidade", // range axis label
                        dataset, // data
                        PlotOrientation.VERTICAL, // orientation
                        true, // include legend
                        true, // tooltips?
                        false // URLs?
                        );
            return chart;
      }// fim do criaChart

}// fim da classe

package br.com.aps.biometria.tipos;

import java.awt.Color;
import java.awt.image.BufferedImage;
import java.io.File;
import java.io.IOException;
import java.util.Arrays;

import javax.imageio.ImageIO;

public class Imagem {

      private int[][] frequenciaRGB;
      private int[][] freqYUV;
      private int[][] freqHSV;

      public Imagem(String arquivoNoDisco) throws IOException {

            BufferedImage imagem = ImageIO.read(new
File(arquivoNoDisco));

            this.frequenciaRGB = new int[256][3];
            this.freqHSV = new int[imagem.getWidth() *
imagem.getHeight()][3];
            this.freqYUV = new int[imagem.getWidth() *
imagem.getHeight()][3];
            this.calculaFrequencia(imagem);

            imagem = null;
      }
```

```
	public int[][] getFrequencia() {
		return this.frequenciaRGB;
	}

	private void calculaFrequencia(BufferedImage imagem) {

		int indice = 0;

		for (int j = 0; j < imagem.getWidth(); j++)
			for (int k = 0; k < imagem.getHeight(); k++,
indice++) {
				Color cor = new Color(imagem.getRGB(j, k));

				int r = cor.getRed();
				int g = cor.getGreen();
				int b = cor.getBlue();

				this.frequenciaRGB[r][0] += 1;
				this.frequenciaRGB[g][1] += 1;
				this.frequenciaRGB[b][2] += 1;

				this.rgbParaHsv(r, g, b, freqHSV[indice]);
				this.rgbParaYuv(r, g, b, this.freqYUV[indice]);

			}// fim do for

	}// fim do calculaFrequencia

	/*
	 * Calcula frequencia dos 256 tons de cinza
	 */
	private void rgbParaYuv(int r, int g, int b, int yuv[]) {

		int y = (int) (0.299 * r + 0.587 * g + 0.114 * b);
		int u = (int) ((b - y) * 0.492f);
		int v = (int) ((r - y) * 0.877f);

		yuv[0] = y;
		yuv[1] = u;
		yuv[2] = v;
	}

	private void rgbParaHsv(int r, int g, int b, int hsv[]) {

		int min, max, delMax;

		if (r > g) {
			min = g;
			max = r;
		} else {
			min = r;
			max = g;
		}
		if (b > max)
			max = b;
		if (b < min)
			min = b;

		delMax = max - min;

		float H = 0, S;
```

```
            float V = max;

            if (delMax == 0) {
                  H = 0;
                  S = 0;
            } else {
                  S = delMax / 255f;
                  if (r == max)
                        H = ((g - b) / (float) delMax) * 60;
                  else if (g == max)
                        H = (2 + (b - r) / (float) delMax) * 60;
                  else if (b == max)
                        H = (4 + (r - g) / (float) delMax) * 60;
            }

            hsv[0] = (int) (H);
            hsv[1] = (int) (S * 100);
            hsv[2] = (int) (V * 100);
      }

      @Override
      public String toString() {
            return "Imagem [frequenciaRGB=" +
Arrays.toString(frequenciaRGB) + "]";
      }

}
package br.com.aps.biometria.model;

import java.io.File;
import java.io.FileInputStream;
import java.io.IOException;
import java.io.ObjectInputStream;

public class LerArquivo {

      protected ObjectInputStream input;
      private String nomeArquivo;

      public boolean existeArquivo() {
            File file = new File(nomeArquivo);
            return file.exists();
      }

      public void abrirArquivo() throws IOException {
            FileInputStream file = new FileInputStream(nomeArquivo);
            input = new ObjectInputStream(file);
      }

      public LerArquivo(String nomeArquivo) {
            super();
            this.nomeArquivo = nomeArquivo;
      }

      public void fecharArquivo() throws IOException {
            if (input != null)
                  input.close();
      }

}
package br.com.aps.biometria.model;
```

```
import java.io.EOFException;
import java.util.ArrayList;
import java.util.List;

import br.com.aps.biometria.tipos.Frequencia;

public class LerFrequencia extends LerArquivo {

	public LerFrequencia() {
		super(FuncoesGerais.getNomeArquivoRegistroFrequencia());
	}

	public List<Frequencia> getLista() throws Exception {
		List<Frequencia> lista = new ArrayList<Frequencia>();
		Frequencia registro;

		try {
			while (true) {
				registro = (Frequencia) input.readObject();
				lista.add(registro);
			}
		} catch (EOFException endOfFileException) {
			return lista;
		} catch (Exception exception) {
			throw new Exception(exception.getMessage());
		}

	}

}
package br.com.aps.biometria.controller;

import javax.swing.SwingUtilities;
import javax.swing.UIManager;
import javax.swing.UnsupportedLookAndFeelException;

public class MainBiometrico {

	public static void main(String[] args) {

		try {
			for (UIManager.LookAndFeelInfo info : UIManager
					.getInstalledLookAndFeels()) {
				if ("Windows".equals(info.getName())) {

	UIManager.setLookAndFeel(info.getClassName());
					break;
				}
			}
			SwingUtilities.invokeLater(new Runnable() {
				@Override
				public void run() {
					try {
						new MainMenu();
					} catch (Exception e) {
						// TODO Auto-generated catch block
						e.printStackTrace();
					}
				}
			});
```

```
            } catch (ClassNotFoundException | InstantiationException
                        | IllegalAccessException |
UnsupportedLookAndFeelException ex) {
                  System.err.println("Houve um erro na inicialização
(CLASSE: MAIN)");
                  System.err.println("CAUSA:");
                  ex.printStackTrace();
            }
      }

}
package br.com.aps.biometria.controller;

import java.awt.Dimension;
import java.awt.GridBagConstraints;
import java.awt.GridBagLayout;
import java.awt.Insets;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.io.File;
import java.io.IOException;
import java.util.Hashtable;

import javax.swing.ImageIcon;
import javax.swing.JFileChooser;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JMenu;
import javax.swing.JMenuBar;
import javax.swing.JMenuItem;
import javax.swing.JOptionPane;
import javax.swing.JSlider;
import javax.swing.event.ChangeEvent;
import javax.swing.event.ChangeListener;
import javax.swing.filechooser.FileFilter;

import br.com.aps.biometria.model.FuncoesGerais;

public class MainMenu extends JFrame {

      private static final long serialVersionUID = 1L;
      private PainelImagemAutenticacao painelImagemAutenticacao;
      private PainelHistograma painelHistogram;
      private PainelImagemDB painelImagemDB;
      private PainelNotificacao painelNotificacao;
      private JSlider controleDeTempoDeBusca;

      private JMenuBar menuBar;
      private JMenu menuOpcoes;
      private JMenu menuEditar;
      private JMenuItem itemImportarDB;
      private JMenuItem itemExportarPlanilha;
      private JMenuItem itemSair;
      private JMenuItem itemLimparTela;

      private int tempoDeBusca;
      private GridBagConstraints gbc;
      private Controller controller = new Controller();

      public MainMenu() throws Exception {
            loadComponents();
```

```
            loadLayouts();
            customizeComponents();
            addComponents();
            addListeners();
            limparTela();
            setVisible(true);
      }

      private void loadComponents() throws Exception {
            gbc = new GridBagConstraints();
            painelImagemAutenticacao = new PainelImagemAutenticacao();
            painelImagemDB = new PainelImagemDB();
            painelHistogram = new PainelHistograma();
            painelNotificacao = new PainelNotificacao();
            controleDeTempoDeBusca = new JSlider(JSlider.HORIZONTAL, 0,
1000,
                        (tempoDeBusca = 100));
            menuBar = new JMenuBar();
            menuOpcoes = new JMenu("Opções");
            menuEditar = new JMenu("Editar");

            itemImportarDB = new JMenuItem("Importar banco de
imagens");
            itemExportarPlanilha = new JMenuItem("Exportar planilha
eletrônica");
            itemSair = new JMenuItem("Sair");
            itemLimparTela = new JMenuItem("Limpar Tela");

            controller.setPainelImagemDB(painelImagemDB);
            controller.setPainelNotificacao(painelNotificacao);
            controller.setPainelHistograma(painelHistogram);
            controller.setMainMenu(this);
            controller.carregarListaFrequencia();

            painelImagemAutenticacao.setController(controller);
      }

      private void loadLayouts() {
            getContentPane().setLayout(new GridBagLayout());
      }

      private void customizeComponents() {
            setIconImage(new
ImageIcon(FuncoesGerais.getLogo()).getImage());
            setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
            setTitle("APS 2014 - Comparação de Histograma de imagem
biométrica - Marcio, Diego e Thiago");
            setSize(1020, 760);
            setMinimumSize(new Dimension(1020, 760));
            setLocationRelativeTo(null);
            setResizable(true);
            menuBar.setPreferredSize(new Dimension(0, 30));
            controleDeTempoDeBusca.setMajorTickSpacing(10);
            controleDeTempoDeBusca.setPaintTicks(true);
            Hashtable<Integer, JLabel> labelTable = new
Hashtable<Integer, JLabel>();
            labelTable.put(0, new JLabel("0 seg/digital"));
            labelTable.put(tempoDeBusca, new JLabel("Padrão"));
            labelTable.put(1000, new JLabel("1 seg/digital"));
            controleDeTempoDeBusca.setLabelTable(labelTable);
            controleDeTempoDeBusca.setPaintLabels(true);
```

```
}

private void addComponents() {
	setJMenuBar(menuBar);
	menuBar.add(menuOpcoes);
	menuOpcoes.add(itemImportarDB);
	menuOpcoes.add(itemExportarPlanilha);
	menuOpcoes.add(itemSair);
	menuBar.add(menuEditar);
	menuEditar.add(itemLimparTela);

	gbc.fill = GridBagConstraints.BOTH;
	gbc.anchor = GridBagConstraints.NORTHWEST;

	gbc.weightx = 300;
	gbc.weighty = 20;
	gbc.gridwidth = 3;
	gbc.gridheight = 1;
	gbc.gridx = 0;
	gbc.gridy = 0;
	gbc.insets = new Insets(0, 0, 0, 0);
	getContentPane().add(controleDeTempoDeBusca, gbc);

	gbc.weightx = 300;
	gbc.weighty = 300;
	gbc.gridwidth = 1;
	gbc.gridheight = 1;
	gbc.gridx = 1;
	gbc.gridy = 1;
	gbc.insets = new Insets(8, 8, 0, 0);
	getContentPane().add(painelImagemAutenticacao, gbc);

	gbc.weightx = 300;
	gbc.weighty = 300;
	gbc.gridwidth = 1;
	gbc.gridheight = 1;
	gbc.gridx = 1;
	gbc.gridy = 2;
	gbc.insets = new Insets(8, 8, 8, 0);
	getContentPane().add(painelImagemDB, gbc);

	gbc.weightx = 300;
	gbc.weighty = 300;
	gbc.gridwidth = 1;
	gbc.gridheight = 1;
	gbc.gridx = 2;
	gbc.gridy = 1;
	gbc.insets = new Insets(8, 8, 0, 8);
	getContentPane().add(painelHistogram, gbc);

	gbc.weightx = 300;
	gbc.weighty = 300;
	gbc.gridwidth = 1;
	gbc.gridheight = 1;
	gbc.gridx = 2;
	gbc.gridy = 2;
	gbc.insets = new Insets(8, 8, 8, 8);
	getContentPane().add(painelNotificacao, gbc);
}

public void addListeners() {
```

```
            itemSair.addActionListener(new ActionListener() {
                  @Override
                  public void actionPerformed(ActionEvent e) {
                        System.exit(0);
                  }
            });

            itemImportarDB.addActionListener(new ActionListener() {
                  @Override
                  public void actionPerformed(ActionEvent e) {
                        try {
                              controller.gerarBancoDeDadosImagem();
                        } catch (Exception e1) {
                              painelNotificacao.addText("Base de dados
não encontrada..");
                              e1.printStackTrace();
                        }
                  }
            });

            itemExportarPlanilha.addActionListener(new ActionListener()
{
                  @Override
                  public void actionPerformed(ActionEvent e) {
                        exportarExcel();
                  }
            });

            itemLimparTela.addActionListener(new ActionListener() {
                  @Override
                  public void actionPerformed(ActionEvent e) {
                        limparTela();
                  }
            });

            controleDeTempoDeBusca.addChangeListener(new
ChangeListener() {
                  @Override
                  public void stateChanged(ChangeEvent evt) {
                        tempoDeBusca =
controleDeTempoDeBusca.getValue();
                  }
            });
      }

      public void ativarComandos(boolean option) {
            painelImagemAutenticacao.setDragEnabled(option);
            menuEditar.setEnabled(option);
            menuOpcoes.setEnabled(option);
      }

      public void limparTela() {
            painelImagemAutenticacao.removeImagem();

      painelImagemAutenticacao.addImagem(FuncoesGerais.getImagemArrast
ar());
            painelImagemDB.addImagem(FuncoesGerais.getImagemPadrao());
            painelHistogram.clear();
            painelNotificacao.clear();
      }
```

```
	private void exportarExcel() {
		try {
			JFileChooser saveFile = new JFileChooser();// new
save dialog
			saveFile.setAcceptAllFileFilterUsed(false);
			saveFile.setDialogType(JFileChooser.SAVE_DIALOG);
			saveFile.setDialogTitle("Exportar usuários com
digitais do Banco de Dados");

			saveFile.addChoosableFileFilter(new FileFilter() {

				String description = "Excel (*.xlsx)";
				String extension = "xlsx";

				public String getDescription() {
					return description;
				}

				public boolean accept(File f) {
					if (f == null)
						return false;
					if (f.isDirectory())
						return true;
					return
f.getName().toLowerCase().endsWith(extension);
				}
			});
			saveFile.setCurrentDirectory(new File("*.srt"));
			int result = saveFile.showSaveDialog(this);

			if (result == JFileChooser.APPROVE_OPTION) {
				String strFileName =
saveFile.getSelectedFile().getName();

				if (!strFileName.isEmpty()) {
					if
(!strFileName.toLowerCase().endsWith("xlsx"))
						throw new IOException(
								"Arquivo deve ter a
extensão xlsx");

					String nomeArquivo =
saveFile.getSelectedFile()

	.getAbsoluteFile().toString();

					controller.exportarPlanilha(nomeArquivo);
				}
			}

		} catch (Exception er) {
			JOptionPane.showMessageDialog(this, er.getMessage(),
"Erro", 0);
		}
	}

	public int getTempoDeBusca() {
		return tempoDeBusca;
	}

}
```

```
package br.com.aps.biometria.controller;

import java.awt.BorderLayout;
import java.awt.Color;
import java.awt.Dimension;

import javax.swing.BorderFactory;
import javax.swing.JPanel;

import org.jfree.chart.ChartFactory;
import org.jfree.chart.ChartPanel;
import org.jfree.chart.JFreeChart;
import org.jfree.chart.plot.PlotOrientation;
import org.jfree.data.category.DefaultCategoryDataset;

import br.com.aps.biometria.tipos.Frequencia;

public class PainelHistograma extends JPanel {

	private static final long serialVersionUID =
7751091147060793494L;
	private JFreeChart grafico;
	private ChartPanel chartPanel;
	private int[][] frequencia = new int[256][3];
	private Frequencia frequenciaImagem;

	public PainelHistograma() {
		loadLayouts();
		loadHistogram();
		customizeComponents();
		addComponents();
	}

	private void loadHistogram() {
		grafico = ChartFactory.createHistogram(null, null, null,
null,
				PlotOrientation.VERTICAL, false, true, true);

		grafico.setBorderVisible(true);
		chartPanel = new ChartPanel(grafico, true);
	}

	public void clear() {
		JFreeChart grafico = ChartFactory.createHistogram(null,
null, null,
				null, PlotOrientation.VERTICAL, false, true,
true);
		grafico.setBorderVisible(true);

		chartPanel.setChart(grafico);
		this.grafico = grafico;
	}

	private void loadLayouts() {
		setLayout(new BorderLayout());
	}

	private void customizeComponents() {
		setBorder(BorderFactory.createTitledBorder("Histograma"));
		grafico.setBorderPaint(Color.lightGray);
		chartPanel.setPreferredSize(new Dimension(500, 0));
```

```
	}

	private void addComponents() {
		add(chartPanel, BorderLayout.CENTER);
	}

	public void setFrequenciaImagem(Frequencia frequenciaImagem) {
		this.frequenciaImagem = frequenciaImagem;
		this.criaChart();
	}

	private void criaChart() {
		this.frequencia = this.frequenciaImagem.getFrequenciaRGB();

		DefaultCategoryDataset dataset = new
DefaultCategoryDataset();

		// insere os valores no dataset para carregar no grafico
		for (int x = 0; x < this.frequencia.length; x++) {
			String v2 = String.valueOf(x);
			// String v2 = x == 0 || 16 % x == 0?
String.valueOf(x) : "";
			dataset.addValue(frequencia[x][0], "R", v2);
			dataset.addValue(frequencia[x][1], "G", v2);
			dataset.addValue(frequencia[x][2], "B", v2);
		}

		JFreeChart grafico =
ChartFactory.createLineChart("Histograma", // chart

                                // title
                        "Pixels", // domain axis label
                        "Quantidade", // range axis label
                        dataset, // data
                        PlotOrientation.VERTICAL, // orientation
                        true, // include legend
                        true, // tooltips?
                        false // URLs?
                        );

		chartPanel.setChart(grafico);

		this.grafico = grafico;
	}// fim do criaChart

}// fim da classe

package br.com.aps.biometria.controller;

import java.awt.BorderLayout;
import java.awt.Color;
import java.awt.Dimension;
import java.awt.GridLayout;
import java.awt.event.MouseAdapter;
import java.awt.event.MouseEvent;
import java.io.File;
import java.io.IOException;

import javax.swing.BorderFactory;
import javax.swing.ImageIcon;
import javax.swing.JLabel;
```

```
import javax.swing.JPanel;
import javax.swing.JScrollPane;

import br.com.aps.biometria.file.FileTransfer;

public class PainelImagemAutenticacao extends JPanel {

	private static final long serialVersionUID = 1L;
	private JPanel panelName;
	private JLabel labelImage, labelName;
	private JScrollPane scrollImage;
	private FileTransfer fileTransfer;
	private Controller controller;

	public PainelImagemAutenticacao() {
		loadComponents();
		loadLayouts();
		customizeComponents();
		addComponents();
		addListeners();
	}

	public void setController(Controller controller) {
		this.controller = controller;
	}

	private void loadComponents() {
		panelName = new JPanel();
		labelImage = new JLabel("", JLabel.CENTER);
		labelName = new JLabel(" Arrastar aqui a imagem");
		scrollImage = new JScrollPane();
		fileTransfer = new FileTransfer();
	}

	private void loadLayouts() {
		setLayout(new BorderLayout());
		labelImage.setLayout(new GridLayout());
		panelName.setLayout(new GridLayout());
	}

	private void customizeComponents() {
		setBorder(BorderFactory.createTitledBorder("Autenticação de
Imagem"));
		panelName.setBackground(Color.white);
		labelImage.setBackground(Color.white);
		labelImage.setOpaque(true);
		scrollImage.setPreferredSize(new Dimension(200, 0));
		labelName.setPreferredSize(new Dimension(200, 15));
		labelImage.setTransferHandler(fileTransfer);
	}

	private void addComponents() {
		panelName.add(labelName);
		scrollImage.setViewportView(labelImage);
		add(scrollImage, BorderLayout.CENTER);
		add(panelName, BorderLayout.SOUTH);
	}

	private void addImagem(File file) throws IOException {
		addImagem(file.getPath());
		labelName.setText(" " + file.getName());
```

```
			fileTransfer.setFileOnDrag(false);
			controller.autenticarImagem(file);
		}

		public void addImagem(String arquivo) {
			ImageIcon imageIcon = new ImageIcon(arquivo);
			labelImage.setIcon(imageIcon);
		}

		public void removeImagem() {
			fileTransfer.removeFile();
			labelImage.setIcon(null);
			labelName.setText(" Arrastar aqui a imagem");
		}

		public void setDragEnabled(boolean option) {
			fileTransfer.setDragEnabled(option);
		}

		public void addListeners() {
			labelImage.addMouseListener(new MouseAdapter() {
				@Override
				public void mouseEntered(MouseEvent e) {
					if (fileTransfer.isFileOnDrag())
						if (fileTransfer.getFile() != null) {

		controller.getPainelNotificacao().addText(
                                                "Carregando a imagem do
arquivo: "
                                                                    +
fileTransfer.getFile());
                                    try {

		addImagem(fileTransfer.getFile());
                                    } catch (IOException e1) {
                                          e1.printStackTrace();
                                    }
                               }
                    }
               });
          }

}
package br.com.aps.biometria.controller;

import java.awt.BorderLayout;
import java.awt.Color;
import java.awt.Dimension;
import java.awt.GridLayout;

import javax.swing.BorderFactory;
import javax.swing.ImageIcon;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.JScrollPane;

public class PainelImagemDB extends JPanel {

	private static final long serialVersionUID = 1L;
	private JPanel panelName;
	private JScrollPane scrollImage;
```

```
	private JLabel labelImage, labelName;

	public PainelImagemDB() {
		loadComponents();
		loadLayouts();
		customizeComponents();
		addComponents();
	}

	private void loadComponents() {
		panelName = new JPanel();
		labelImage = new JLabel("", JLabel.CENTER);
		labelName = new JLabel(" Imagem não encontrada..");
		scrollImage = new JScrollPane();
	}

	private void loadLayouts() {
		setLayout(new BorderLayout());
		labelImage.setLayout(new GridLayout());
		panelName.setLayout(new GridLayout());
	}

	private void customizeComponents() {
		setBorder(BorderFactory.createTitledBorder("Imagem BD"));
		panelName.setBackground(Color.white);
		labelImage.setBackground(Color.white);
		labelImage.setOpaque(true);
		scrollImage.setPreferredSize(new Dimension(200, 0));
		labelName.setPreferredSize(new Dimension(200, 15));
	}

	private void addComponents() {
		panelName.add(labelName);
		scrollImage.setViewportView(labelImage);
		add(scrollImage, BorderLayout.CENTER);
		add(panelName, BorderLayout.SOUTH);
	}

	public void addImagem(String path) {
		ImageIcon imageIcon = new ImageIcon(path);
		labelName.setText(" Nome");
		labelImage.setIcon(imageIcon);
	}

	public void removeImage() {
		labelName.setText(null);
		labelImage.setIcon(null);
	}

	public void setNome(String nome) {
		labelName.setText(nome);
	}
}package br.com.aps.biometria.controller;

import java.awt.BorderLayout;
import java.awt.Dimension;

import javax.swing.BorderFactory;
import javax.swing.JPanel;
import javax.swing.JProgressBar;
import javax.swing.JScrollPane;
```

```
import javax.swing.JTextArea;

import org.joda.time.DateTime;
import org.joda.time.format.DateTimeFormat;
import org.joda.time.format.DateTimeFormatter;

public class PainelNotificacao extends JPanel {

	private static final long serialVersionUID = 1L;
	private JTextArea textNotification;
	private JProgressBar loading;
	private JScrollPane scroll;
	private DateTimeFormatter timeFormatter;

	public PainelNotificacao() {
		loadComponents();
		loadLayouts();
		customizeComponents();
		addComponents();
	}

	private void loadComponents() {
		textNotification = new JTextArea();
		loading = new JProgressBar();
		scroll = new JScrollPane();
		timeFormatter = DateTimeFormat.forPattern("HH:mm:ss: ");
	}

	private void loadLayouts() {
		setLayout(new BorderLayout());
	}

	private void customizeComponents() {

	setBorder(BorderFactory.createTitledBorder("Notificações"));
		scroll.setPreferredSize(new Dimension(500, 0));
		loading.setPreferredSize(new Dimension(500, 15));
		textNotification.setEditable(false);
	}

	private void addComponents() {
		scroll.setViewportView(textNotification);
		add(scroll, BorderLayout.CENTER);
		add(loading, BorderLayout.SOUTH);
	}
	public void addText(String text) {
		int length = textNotification.getText().length();
		String time = timeFormatter.print(DateTime.now());
		textNotification.append(time + text + "\n");
		textNotification.setCaretPosition(length);
		update(getGraphics());
	}
	public void clear() {
		textNotification.setText("");
		loading.setValue(0);
	}
	public void setLoadingValue(int value) {
		loading.setValue(value);
	}

}
```

6. Referências bibliográficas

6.1 Livros

Use a cabeça Java, segunda edição, Kathy Sierra, Bert Bates, Editora Alta Books, Ano 2010, ISBN 9788576081739.

6.2 Acessos de links via Internet

<http://www.pergamum.udesc.br/dadosbu/000000/000000000005/0000052A.pdf> Acesso em 30/09/2014.

<http://pt.wikipedia.org/wiki/Biometria#Tipos_de_biometria> Acesso em 05/10/2014.

<http://www.ibiometrica.com.br/biometria_sistemas.asp > Acesso em 06/10/2014.

<http://www.infowester.com/biometria.php> Acesso em 10/10/2014.

<http://www.smartsec.com.br/biometria.htm> Acesso em 12/10/2014.

<http://pt.wingwit.com/P/java-programming/90718.html> Acesso em 17/10/2014.

<http://www.ufrgs.br/engcart/PDASR/hist.html> Acesso em 18/10/2014.

<http://www.cambridgeincolour.com/pt-br/tutorials/histograms1.htm> Acesso em 19/10/2014.

<http://www.dpi.inpe.br/spring/teoria/realce/realce.htm> Acesso em 23/10/2014.