



**Curso de Ciência da Computação**

**Atividades Práticas Supervisionadas - APS**

**DESENVOLVIMENTO DE UM SISTEMA DE IDENTIFICAÇÃO E AUTENTICAÇÃO  
BIOMÉTRICA**

**Guilherme Augusto Sbizero Correa RA: F235289**

**Gabriel Macedo Ramos RA: F281060**

São Paulo

2022

## **Resumo**

Biometria é o uso de características biológicas em mecanismos de identificação. Entre essas características tem-se a íris (parte colorida do olho), a retina (membrana interna do globo ocular), a impressão digital, a voz, o formato do rosto e a geometria da mão.

O uso de características biológicas para identificação se mostra como uma ideia viável porque cada pessoa possui as características mencionadas diferentes das outras.

Com a biometria, o problema de roubo de senhas ou cartões com chips é extinto ou, pelo menos amenizado. Embora nada impeça os dispositivos de identificação biométrica de serem enganados, é muito difícil copiar uma característica física e, dependendo do que é usado na identificação, a cópia é impossível (como a íris do olho).

O uso da biometria para identificação de pessoas já é realidade e é pouco provável que outro conceito a substitua.

É certo que a biometria vai ser cada vez mais parte do dia a dia das pessoas. Prova disso é que as tecnológicas envolvidas ganham aprimoramentos constantes.

Palavra-chave: biometria, identificação, reconhecimento facial, autenticação biométrica.

## **SUMÁRIO**

<b>1. OBJETIVO</b>	<b>4</b>
<b>2. INTRODUÇÃO</b>	<b>5</b>
<b>3. FUNDAMENTOS DAS PRINCIPAIS TÉCNICAS BIOMÉTRICAS</b>	<b>7</b>
<b>4. PLANO DE DESENVOLVIMENTO DA APLICAÇÃO</b>	<b>15</b>
<b>5. PROJETO</b>	<b>19</b>
<b>6. RELATÓRIO COM LINHAS DE CÓDIGO DO PROGRAMA</b>	<b>24</b>
<b>7. APRESENTAÇÃO DO PROGRAMA FUNCIONANDO EM UM COMPUTADOR</b>	<b>54</b>
<b>8. BIBLIOGRAFIA</b>	<b>65</b>
<b>9. FICHAS DE ATIVIDADES PRÁTICA SUPERVISIONADAS</b>	<b>67</b>

## 1 OBJETIVO

Desenvolver, utilizando a linguagem de programação Java/Python, um programa de identificação e autenticação biométrica com interface gráfica, para restringir o acesso dos usuários a um determinado banco de dados. Esta restrição deve ser feita levando-se em consideração os níveis de permissão de acesso que os usuários têm em determinadas informações, que serão divididas em 3 níveis.

O banco de dados que deve ser protegido pertence ao Ministério do Meio Ambiente. Contendo informações sigilosas sobre propriedades rurais que fazem uso de agrotóxicos proibidos, causando grandes impactos ambientais.

São objetivos específicos desta atividade:

- Pesquisar e dissertar sobre os conceitos gerais da biometria;
- Definir o tipo de autenticação biométrica que será utilizada;
- Definir estrutura do sistema;
- Criar a interface gráfica;
- Implementar o projeto utilizando a linguagem de programação Java.

## 2 INTRODUÇÃO

Com a cada vez maior informatização da sociedade e a transferência de informações para o mundo digital, é de vital importância conseguir proteger estes dados. Os métodos tradicionais de proteção, que utilizam um nome de usuário e senha para impedir acessos não autorizados, tem demonstrado ser insuficiente nesta tarefa, pois sua segurança é muito baixa. O principal tópico de segurança para proteger serviços de rede ou alguma determinada área do sistema, é a autenticação e identificação dos usuários. Se esta etapa não conseguir impedir acessos indevidos, então dificilmente outro método de segurança implementado terá eficiência nesta tarefa. É neste cenário onde a biometria surge como um dos principais, mais eficazes, seguros e confiáveis métodos de identificação e autenticação indivíduos. Biometria (palavra derivada do grego bio, que significa vida, e metria, que significa medição) é a medição e a análise estatística das características físicas e comportamentais dos indivíduos. A premissa básica da autenticação biométrica é que todos são únicos e que um indivíduo pode ser identificado por suas características físicas ou traços comportamentais. A tecnologia tem sido usada fortemente como o principal meio de controle de acesso e/ou para identificação de indivíduos sob vigilância. O uso da impressão digital para assinar documentos foi pioneiro no conceito de biometria, sendo utilizada por diversos povos em diferentes épocas, desde os antigos babilônios, egípcios, assírios, japoneses e chineses. Eles utilizam a impressão digital para marcar seus produtos e lacrar documentos. A impressão digital ajudava a identificar bons e maus negociantes com os quais os mercadores já haviam realizado algum tipo de negócio. Os chineses costumavam com o uso de papel e tinta, marcar pés e mãos das crianças para diferenciá-las, podendo ser considerado um dos primeiros exemplos de uso da certidão de nascimento. Os avanços comerciais e de automatizações relacionadas à utilização da biometria tiveram início na década de setenta. Nesta época, surgiu um sistema conhecido como Identimat, esse sistema foi instalado em vários locais secretos para controle de acesso. Funcionava medindo a palma da mão e analisando o tamanho dos dedos. O Identimat teve sua produção encerrada na década de oitenta, mas seu pioneirismo foi importante na divulgação e aceitação da tecnologia biométrica 4 perante a população. Junto ao desenvolvimento da tecnologia baseada na mão, a biometria baseada em impressão digital começa a ganhar maior espaço. Durante esse período, algumas empresas estavam trabalhando com a identificação automática das impressões digitais para ajudar as forças policiais. A

comparação das imagens das digitais armazenadas em registros criminais, era, até então, realizada manualmente o que demandava muito tempo e esforço. Outros métodos foram criados e os antigos continuaram a ser melhorados, enquanto a adoção dos baseados impressões digitais avançava. O primeiro sistema a analisar o padrão único da retina iniciou sua utilização na metade dos anos oitenta. No mesmo período na Universidade de Cambridge, foi desenvolvida a tecnologia baseada na íris. No Brasil em 2004 foi inaugurado a AFIS (Automated Fingerprint Identification System, ou, Sistema Automatizado de Identificação de Impressões Digitais), que foi interligado com o sistema de Informações Criminais (Sinic) criando o Sistema de Identificação Nacional (SIN). O SIN foi inaugurado com 800 mil impressões digitais de criminosos. Em 2007 o governo brasileiro iniciou a emissão do passaporte biométrico. Contendo diversos itens de segurança, o novo sistema de passaporte coleta assinatura, foto e dez impressões digitais roladas. Nos últimos anos está sendo implementada a utilização da biometria em eleições, com o intuito de aumentar a segurança do processo eleitoral, buscando evitar fraudes. Muitos países têm desenvolvido grandes inovações nesta área, tanto nos processos de expedição dos documentos eleitorais quanto nos procedimentos da própria votação em si. O maior exemplo são as urnas eletrônicas com autenticação biométrica.

### 3 FUNDAMENTOS DAS PRINCIPAIS TÉCNICAS BIOMÉTRICAS

A autenticação ou identificação de um determinado indivíduo pode ser feita de diversas formas, podemos classificá-las em três grandes grupos: por aquilo que se possui, por aquilo que se sabe que é.

**Por que se sabe:** a autenticação é realizada utilizando algo que o usuário conhece. Exemplos são os nomes de acesso (nomes de usuário), senhas e chaves criptográficas. Em termos de segurança são os níveis mais baixos.

**Por aquilo que se possui:** estes métodos realizam a autenticação baseada em algo que o usuário possui. Geralmente são utilizados dispositivos que contêm memória ou capacidade de processamento. Os exemplos mais conhecidos são os Tokens, Smart Cards (cartões bancários, cartão de identidade pessoal, “chip” de celulares), cartões magnéticos e crachás. Estes métodos possuem nível de segurança média.

**Por aquilo que você é:** a autenticação é baseada naquilo que o usuário é. Ou seja, é nesta categoria que se encaixa a biometria. Podendo ser utilizada uma característica física ou uma característica comportamental única, que permita identificar e distinguir um indivíduo do outro. Exemplos de características biométricas são a impressão digital, a íris, a retina, a voz, a assinatura etc. São considerados os métodos que oferecem o mais alto nível de segurança. O emprego da biométrica em tecnologias de autenticação já faz parte do dia a dia da maior parte da população. Seu uso mais visível são os caixas eletrônicos dos bancos (que estão adotando este método) e as urnas eletrônicas utilizadas nas votações eleitorais. Também é utilizada como método de verificação de funcionários, validação de associados de planos de saúde, controle de acesso em locais de entrada restrita, segurança de redes, comércio eletrônico, acesso virtual, catraca eletrônica etc. A lista de usos somente tende a aumentar conforme a sociedade automatiza e informatiza seus processos. Um sistema de autenticação biométrica é dividido em duas fases: o registro do perfil do usuário e a sua autenticação. Na primeira fase é realizado o cadastramento da amostra biométrica dos usuários. Isto é feito utilizando-se um dispositivo que permita a captura destas características biométricas, como, por exemplo, leitor de digital, microfone, câmera de vídeo etc. 6 Após a captura, a amostra é transformada em um algoritmo matemático que será armazenado em um banco de dados. Toda vez que for preciso autenticar um usuário, será capturada uma nova amostra biométrica que será comparada com o modelo que está armazenado. Com relação aos métodos utilizados na identificação biométrica, eles

podem ser divididos em dois grandes grupos: aqueles que utilizam características físicas e aqueles que utilizam características comportamentais.

### 3.1 Características Físicas

Utilizam a forma ou composição do corpo como identificador único dos indivíduos.

### 3.2 Impressões Digitais

Método automatizado utilizado para identificação ou confirmação de indivíduo baseado em uma comparação entre dois dedos. É o tipo mais conhecido e usado de biometria atualmente. As razões para sua popularidade são sua facilidade de aquisição, uso estabelecido, sua aceitação em comparação com os demais e o fato de que há dez fontes biométricas por indivíduo.

A impressão digital é formada por diversas linhas, criadas a partir das elevações da pele. Ela é única para cada indivíduo, mesmo irmãos gêmeos possuem impressões digitais diferentes. Sendo assim um dos métodos de identificação mais seguros existentes. Tendo sido descoberto e utilizado a mais de mil anos.

Esta característica individual sofre poucas alterações durante a vida de uma pessoa. Alguns fatores externos podem modificá-los, como por exemplo, o uso de produtos químicos ou trabalhos manuais que acabem danificando as linhas da impressão digital.

A identificação biométrica utilizando este, método, funciona através da extração dos pontos de minúcias da impressão digital. Após a extração estes valores são processados realizando-se diversos cálculos a partir dos quais os sistemas computacionais identificam o indivíduo dono da impressão digital.

Minúcias é o nome dado ao conjunto das características de uma impressão digital. Na figura 1 são mostrados exemplos de minúcias.

Figura 1 – Tipos de minúcias



Fonte: GTA – Grupo de Teleinformática e Automação da UFRJ



Existem três padrões básicos da impressão digital: o arco, o laço e a espiral. Algumas imagens de impressões digitais que apresentam estes padrões podem ser vistas na Figura 2.



Fonte: Biometric Solutions (adaptada pelos autores)

Um arco é um padrão onde a linha da digital entra em um dos lados do dedo, cresce ao chegar no meio formando um arco e sai pelo outro lado. No laço, mais famoso padrão de digital, a linha entra por um dos lados, forma uma curva e sai pelo mesmo local pelo qual entrou. Já a espiral é um padrão onde são formados círculos ao redor de um ponto central. Com relação aos dispositivos utilizados na coleta da impressão digital, eles são de três tipos: ótico, capacitivo e ultrassônico. O primeiro, e o mais utilizado, trabalha com uma fonte emissora de luz e utiliza a reflexão da luz sobre o dedo para realizar a leitura da digital. Os dispositivos capacitivos medem o calor emitido pela digital. E o último, o ultrassônico, como o próprio nome indica, envia sinais sonoros e coleta a impressão digital analisando o sinal retornado. Ou seja, funciona como um radar.

### **3.1.2 Reconhecimento de íris**

Método automatizado utilizado para identificação ou confirmação de um indivíduo analisando padrões aleatórios da íris. O reconhecimento por íris é relativamente novo, sendo somente comercialmente desenvolvido na última década principalmente devido a limitações de patente.

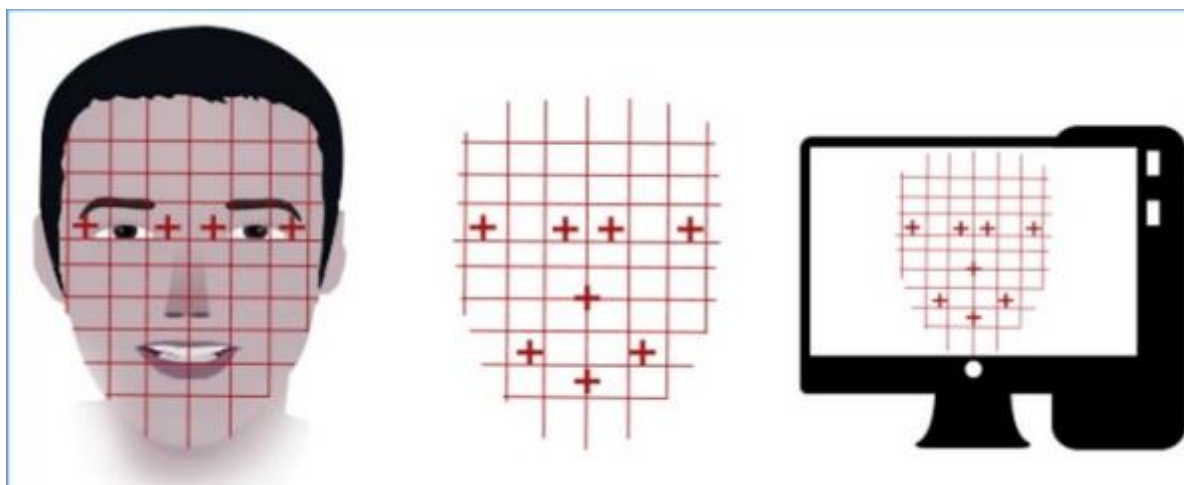
A íris é um músculo dentro do olho que regula o tamanho da pupila, controlando a quantidade de luz que entra no olho. A íris é a parte colorida do olho e cada uma possui sua própria estrutura, o que permite sua utilização na identificação e autenticação de indivíduos.

Esta característica individual não se modifica com o passar dos anos, mantendo-se a mesma por toda a vida.

### 3.1.3 Reconhecimento facial

O reconhecimento facial utiliza a geometria espacial de diferentes características da face (confira a Figura 3). Estas características não são modificadas, mesmo que se realize uma cirurgia plástica. Alguns exemplos de medidas são a distância entre os olhos; a distância entre os olhos, nariz e boca; e a distância entre as linhas dos cabelos, os olhos, a boca e o queixo.

Figura 3 – Reconhecimento facial



Fonte: Tutorials Point

Uma grande vantagem do reconhecimento facial em relação aos demais métodos é que o reconhecimento facial pode ser capturado à distância, como, por exemplo, por uma câmera de segurança, sendo possível sua aplicação sem o conhecimento do sujeito. Sendo adequado para encontrar crianças perdidas e procurar criminosos.

É um dos métodos menos intrusivos, pois não é necessário utilizar informações consideradas extremamente pessoais, como a impressão digital ou a composição da íris.

Outra vantagem é que se pode utilizar, em teoria, quaisquer câmeras digitais como dispositivo de captura da amostra biométrica. Bastando que se tenha um programa de identificação e autenticação biométrica que suporte aquela câmera.

### **3.1.4 Geometria das veias**

Tal como as impressões digitais, o padrão das veias corporais é diferente entre os indivíduos, sendo um excelente critério para a identificação biométrica. O reconhecimento é confirmado a partir dos vasos sanguíneos que fiquem na superfície da pele. Geralmente a parte do corpo utilizada para este método são as mãos. Atualmente, os dois maiores exemplos de instituições que utilizam este método são o FBI e a CIA, ambas dos Estados Unidos da América.

## **3.2 Características Comportamentais**

Utilizam o comportamento como identificador único dos indivíduos.

### **3.2.1 Padrão de digitação**

Método utilizado para a identificação usando como critério os padrões de digitação do indivíduo. As medidas utilizadas por este critério são tempo de permanência (duração que uma tecla é pressionada) e o tempo de não-permanência (duração entre o soltar de uma tecla e o pressionar de outra).

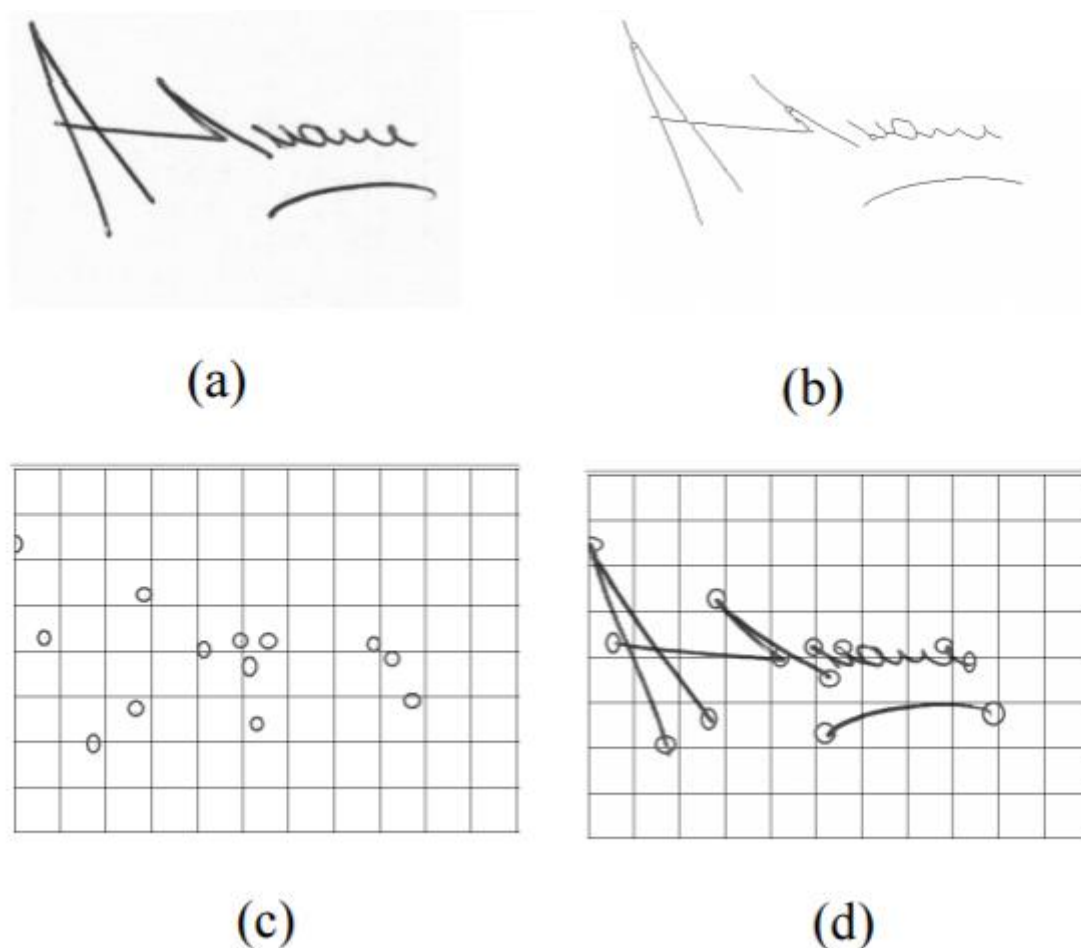
Este padrão foi utilizado na segunda guerra mundial e ficou conhecido como The Fist of the Sender (O punho do remetente). Tendo sido utilizado pelas agências de inteligência militar para distinguir, baseado no ritmo de escrita, se um código morse teria sido enviado por um aliado ou inimigo de guerra.

Como atualmente, na maior parte das residências existe ao menos um teclado, este tipo de método biométrico é o mais fácil de ser implementado, se considerarmos a questão de aquisição do dispositivo de leitura biométrica.

### **3.2.2 Reconhecimento de assinaturas**

Método de baixo nível que leva como critério a assinatura pessoal que cada pessoa opta por utilizar como meio de identificação pessoal. Costuma ser muito confundida com os autógrafos, sendo estes utilizados por artistas e pessoas públicas. Geralmente grandes artistas possuem uma assinatura, onde a mantém privada, e um autógrafo que é utilizado de forma pública.

Figura 4 – Primitivas de “início e fim abruptos”. Imagem original(a), imagem esqueletizada(b), pontos de início e fim do segmento no grid(c), pontos de início e fim na imagem original(d)



Fonte: GTA – Grupo de Teleinformática e Automação da UFRJ (adaptada pelos autores)

O reconhecimento de assinaturas é o método mais utilizado na comprovação de documentos, principalmente em Bancos e cartórios. Ele utiliza diversas características para realizar a análise a assinatura. Entre elas pode-se citar a velocidade, a pressão, a direção e o sentido da escrita. Na Figura 4 está exemplificado parte da análise automatizada de uma digital.

### 3.2.3 Reconhecimento de voz

Refere-se ao método automatizado para reconhecimento da identidade de um indivíduo baseado em sua voz. A sua principal desvantagem reside no fato de que nem todo ambiente pode implementar este método. Pois quanto maior a poluição sonora menor será a precisão do reconhecimento da voz. Outra desvantagem é que o estado de saúde do indivíduo, como gripe ou estresse, pode dificultar sua identificação.

A voz possui dois fatores biométricos, fisiológico e comportamental:

- O componente fisiológico da fala é a forma física da voz do sujeito.

- O componente comportamental trata-se do movimento da mandíbula, lábios, língua etc.

Existem dois tipos de reconhecimento de voz:

- Dependente de texto: O sujeito tem que pronunciar uma frase fixa (senha de acesso) que é a mesma tanto para o registro quanto para a verificação.
- Independente de texto: Baseado em qualquer palavra dita pelo sujeito.

O tipo dependente de texto tem uma melhor performance, porém o outro método é mais flexível, podendo ser utilizado até mesmo com indivíduos que não queiram cooperar, como é o caso criminosos.

### 3.3 Classificação dos sistemas biométricos

Os sistemas biométricos podem ser classificados e comparados através das características humanas que eles utilizam e em relação ao próprio conjunto de dispositivos e processos necessários para o seu funcionamento. A seguir estão algumas categorias de classificação.

**Universalidade:** Está relacionada ao fato de todas as pessoas serem dotadas da característica que será utilizada na identificação individual. Nem sempre a 12 característica será universal. A impressão digital, por exemplo, é uma das que mais pontuam neste quesito, pois o número de indivíduos que não possuem nenhum dedo na mão é irrisório. No sentido oposto, a biometria através da forma como uma pessoa anda, não é universal, pois a medição não pode ser realizada em portadores de deficiência física. Além de ser muito dependente do estado de saúde do indivíduo.

**Unicidade:** Mede o quanto a característica se diferencia entre duas pessoas. Deve-se buscar métodos onde a probabilidade de duas pessoas ter a mesma medida da característica seja extremamente baixa.

**Permanência:** Indica se a característica varia com o tempo. Por exemplo, mesmo com o envelhecimento, o DNA é uma medida que dificilmente sofre alterações. Já a voz, é mais suscetível de sofrer alterações durante a vida e por causa de doenças (o mais comum seria o resfriado).

**Coletabilidade:** Está relacionada as etapas do processo biométrico, indicando o quanto de tempo e esforço são necessários para sua execução. A autenticação biométrica baseada em DNA é extremamente demorada, o que é uma grande desvantagem deste método. Já a impressão digital pode ser rapidamente obtida, sem grande esforço.

**Performance:** É a medida do custo do dispositivo, do processamento e da quantidade de tempo utilizada para realizar a autenticação dos indivíduos.

**Precisão:** Com que exatidão o sistema biométrico consegue diferenciar os indivíduos.

**Aceitabilidade:** Refere-se ao quão bem o sistema biométrico foi aceito pelos seus usuários. Diversas características devem ser levadas em consideração para aumentar a aceitação, sendo as principais a privacidade e o conforto dos usuários.

**Proteção:** Define o nível de segurança do sistema. Ou seja, o quão difícil é enganar o processo de autenticação.

#### 4 PLANO DE DESENVOLVIMENTO

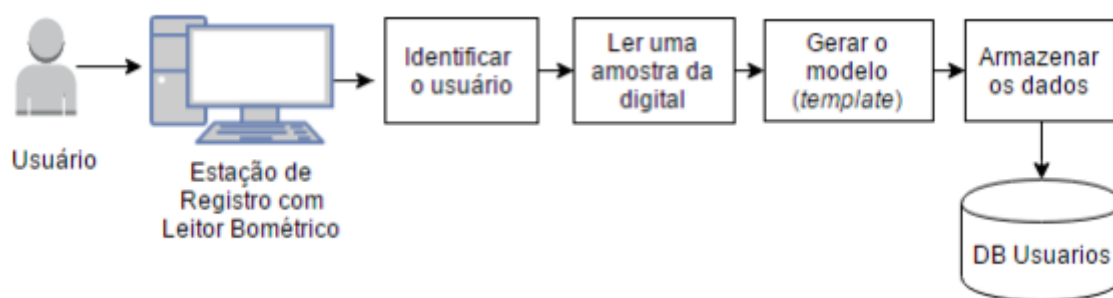
Neste projeto será desenvolvido um sistema de identificação e autenticação biométrica através da impressão digital. Combinando a utilização de nome de usuário com a biometria através da impressão digital. Combinando a utilização do nome de usuário com a biometria, fazendo uma comparação 1:1 na autenticação.

Serão desenvolvidas interfaces gráficas para intermediar todos os processos que dependem da interação dos usuários. Simples e diretas focando nas funções principais da aplicação.

O sistema será formado por dois programas o Gerenciador de Usuários, que cadastrará os usuários e alterar seus dados, e uma aplicação cliente, Banco de Dados, que implementará a autenticação biométrica para restringir o acesso aos dados sigilosos do Ministério do Meio Ambiente.

Para atingir esta finalidade será utilizado uma biblioteca chamada OpenCV desenvolvida pela Intel em 2000, ela é uma biblioteca multiplataforma totalmente livre para o uso acadêmico e comercial, para desenvolvimento de aplicativos para a área de Visão Computacional. Para ler as digitais será feita uma leitura da imagem dela, com isso restringindo o acesso ao banco de dados.

Para o Ministério do Meio Ambiente pode ser utilizado a imagem de sua digital ou a utilização de uma senha, aumentando a segurança das informações restritas. Garantindo que cada usuário possa acessar somente os recursos para quais tem autorização.



Fonte: Elaborada pelos autores

O sistema biométrico trabalha com reconhecimento de padrões, adquirindo as informações biométricas (através da leitura da imagem), gerando um modelo (template) a partir destes dados e armazenando este modelo para posteriormente ser utilizado para comparação e verificação do usuário.

A autenticação dos usuários pode ser realizada de duas formas: 1:1 e 1:N. O

primeiro método utiliza somente a impressão digital. Ele identifica o usuário comparando o modelo da impressão digital coletada com todos os modelos salvos no banco de dados. O que, dependendo da quantidade de usuários cadastrados, consumirá muitos recursos do sistema e demorará mais tempo para concluir a verificação.

O segundo método, 1:1, é o que será utilizado nesta aplicação. Ele realiza a verificação do usuário utilizando também um outro dado armazenado, que neste caso é o *nome de usuário*. Desta forma o modelo da impressão digital coletada será comparado somente com o modelo armazenado para o *nome de usuário* especificado. O que deixará a verificação mais rápida, principalmente quando se tem centenas de funcionários cadastrados, como é o caso Ministério do Meio Ambiente.

#### **4.1 Ambiente de Desenvolvimento**

O sistema será desenvolvido na linguagem de programação Java utilizando o paradigma de Orientação a Objetos. Como IDE (Integrated Development Environment) foi utilizado o Apache NetBeans IDE versão 12.3 junto com a Java Platform Standard Edition Development Kit na versão 15.0.2.

Para realizar o controle das versões do sistema foi empregue o GIT. Permitindo que todos os membros da equipe possam manipular os arquivos do projeto simultaneamente, sem que estas alterações sobrescrevam as de outro membro. Garantido maior produtividade, velocidade e segurança no desenvolvimento do projeto.

O GIMP foi o editor de imagens utilizado na manipulação das figuras deste sistema. Ele é um programa multiplataforma, *open source* e contém as principais funcionalidades e características do Photoshop, além das suas próprias ferramentas. Um de seus diferenciais em relação ao programa da Adobe é ser gratuito.

#### **4.2 Interface Gráfica**

A interface gráfica será desenvolvida fazendo uso da biblioteca gráfica Swing. Todas as JRE, a partir da 1.2, trazem esta biblioteca como o padrão gráfico. Sua principal vantagem reside no fato de ser multiplataforma. Permitindo que as interfaces possam ser utilizadas em qualquer sistema operacional que suporte aJRE, apresentando os componentes da janela com os mesmos atributos (cores,



tamanhos, margens, espaçamento, etc.), independente da plataforma em que está sendo executada.

### 4.3 Leitor Biométrico

Para a realização deste projeto a leitura biométrica será feita através de imagens adquiridas no google, onde serão cadastradas e será feita a leitura e comparação com as que existem no banco de dados.

### 4.4 Computador e Sistema Operacional

As configurações do Notebook utilizado neste projeto, tanto para o desenvolvimento do sistema quanto em seu teste, está na tabela abaixo

<b>Notebook Lenovo Ideapad S145</b>	
Processador	AMD Ryzen 5 3500U with Radeon Vega Mobile Gfx 2.10 GHz
Memória RAM	12,0 GB (utilizável: 9,88 GB)
HD	1 tb
Monitor	15.6 polegadas
Sistema Operacional	Windows 11 Home Single Language 64 bits

Fonte: Elaborada pelos autores

### 4.5 Banco de Dados

Como o SDK utilizado no sistema somente permite a armazenamento dos *templates* no banco de dados proprietário da Neurotechnology, este será o utilizado.

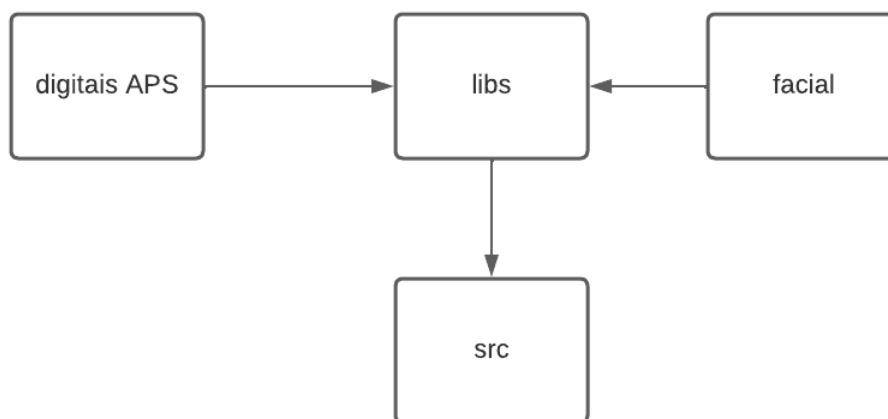
Além dos *templates* será preciso armazenar os dados dos usuários, como seu nome, seu nome de usuário e seu nível de acesso. Isto será realizado através do salvamento do objeto do usuário em um arquivo. Este arquivo será armazenado em formato binário, utilizando a funcionalidade de Serialização disponibilizada pelo Java.

Desta forma o sistema terá dois bancos de dados, um contendo os *templates* e outro os dados dos usuários. Os dois serão arquivos e estarão armazenadas na pasta do sistema. Desta forma torna extremamente fácil mover a aplicação de um computador para outro, ou até mesmo, no futuro, centralizar o banco de dados em

uma única máquina que terá conexão com várias aplicações de cadastro e validação de usuário.

## 5 PROJETO DO PROGRAMA

Os arquivos do sistema foram organizados em 4 diretórios: libs, src, facial e digitais APS. Como visto na figura 11



Fonte: Elaborada pelos autores

O arquivo principal se chama Aps2022\_6SemestreApplication.java que está na pasta source dentro da pasta com, na pasta java, no diretório main e na pasta src. Assim que o abrir no Visual Studio Code click no botão run java para abrir o programa.

### 5.1 Tela de Login

A janela de login é dividida em quatro partes, a primeira é onde ficará a imagem da digital que será utilizada para leitura, ao lado aparecerá a imagem encontrada no banco de dados.

Na segunda parte aparecerá a comparação das imagens, a que foi escolhida e a que foi encontrada.

Na terceira parte aparecerá o número de imagens pesquisadas, ao lado tem um botão chamado “Carregar Imagem” que irá abrir o menu do Explorador de Arquivos para selecionar a imagem da digital, ao lado também tem um botão chamado “Iniciar Reconhecimento” que após selecionar a imagem da digital click nele para iniciar o reconhecimento da digital.

Na quarta e última parte é onde localiza-se uma label para que quando digitada a senha de acesso que é “1020304050” e clicar no botão escrito “Login”

entrará com acesso de nível 3 ao banco de dados do ministério do meio ambiente.

Confira na figura 12 abaixo

Imagem de Entrada	Imagem Encontrada
Comparação	
Imagens Analizadas	
<input type="text"/>	<input type="button" value="Carregar Imagem"/> <input type="button" value="Iniciar Reconhecimento"/>
<input type="button" value="Login por senha mestra"/>	<input type="text" value="1020304050"/> <input type="button" value="Login"/>

Fonte: Elaborada pelos autores

## 5.2 Tela Acesso Banco de Dados

Nesta janela a parte superior é dividida por duas labels, a primeira com o nome do atual usuário e a segunda label indicando o nível de acesso do usuário. Abaixo consiste por quatro botões “Adicionar” que ao clicar nele te encaminha para a janela de Cadastro de Agrotóxicos onde deverão ser preenchidos os dados do agrotóxico, se ele é permitido ou banido (não permitido por causam grandes impactos nos lenções freáticos, rios e mares) e então click no botão “Salvar” para salvar os dados no banco de dados. Veja na figura a seguir

Figura 13

The interface layout includes a header section with the following elements:

- Usuario Atual:** Master
- Nível de Acesso:** 3
- Buttons:** Adicionar, Exibir, Editar, Apagar
- Tabs:** Propriedades, Acessos

Below the tabs is a table with the following columns:

Id	Nome/Razão Social	Cidade	Estado	Status	Destino

Fonte: Elaborada pelos autores

### 5.3 Tela Cadastro Agrotóxico

Clicando no botão “Exibir” após selecionar uma linha da tabela onde irá mostrar os dados da fornecedora dos agrotóxicos, o botão “Editar” os dados cadastrados e o botão “Apagar” após selecionar uma linha irá excluí-la.

Na parte central da janela aparecem dois botões “Propriedades” e “Acessos”, o primeiro é dividido por Id da empresa no sistema, Nome ou Razão Social dela, Cidade onde está localizada a sua sede, o Estado da sede, Destino é para onde os agrotóxicos da empresa serão vendidos, se será no mercado interno ou no mercado externo e por último o Status dos produtos cadastrados se são permitidos ou banidos. Confira a figura a seguir

Figura 14

Unidade	Agrotóxico	
	Adicionar	Remover
Dados Produção	Agrotóxico	Liberado
	CO <sup>2</sup>	Liberado
Propriedades		
	Permitir	Banir
Valores Fiscais (Impostos)		

Fonte: Elaborada pelos autores

## 5.4 Tela Cadastro Usuários

Já na parte de Acessos ela é dividida por Id do Usuário, o Nível de Acesso dele ao sistema e seu Nome, em cima aparecem duas labels com o nome do atual usuário e seu nível de acesso, abaixo existem 4 botões com nomes de “Adicionar”, “Exibir”, “Editar” e “Apagar”.

O Adicionar de leva para uma janela de cadastro de digital junto de seu nome e nível de acesso, o botão Exibir te exhibe as propriedades da digital como nome do usuário dela e nível de acesso, o botão Editar lhe permite modificar o nome de usuário, nível de acesso e mudar a imagem da digital cadastrada e por último o botão Apagar que exclui todos os dados do usuário.

Figura 15

Usuario Atual	Master	Nível de Acesso	3
<div>Adicionar</div> <div>Exibir</div> <div>Editar</div> <div>Apagar</div>			
Propriedades		Acessos	
Id	Nível de Acesso	Nome dos usuarios	

Fonte: Elaborada pelos autores.

## 6 Relatório com as linhas de código do programa

### Classe APS2022\_6SemestreApplication (Main):

```

1 package com.source;
2
3
4 import com.view.models.SplashScreen;
5
6
7
8 public class Aps2022_6SemestreApplication {
9
10     private static boolean preloaderFlag = true;
11     public static void main(String[] args) {
12
13         new Thread(() -> {
14             SplashScreen spl = new SplashScreen();
15             while(preloaderFlag) {
16                 spl.setVisible(true);
17             }
18             spl.close();
19             }).start();
20
21         Application.launch(Aplicacao.class, args); //Inicia a Aplicação
22     }
23
24     public static void setPreloaderFlag(boolean flag) {
25         preloaderFlag = flag;
26     }
27
28
29 }
30

```

Fonte: Elaborada pelo autor.

A classe Main, é responsável por inicializar a aplicação.

### Classe Aplicação:

```

1 package com.source;
2
3
4 import java.util.ArrayList;
5
6
7
8 /**
9  * Em conjunto com a classe Aps2022_6SemestreApplication essas classes são responsáveis por carregar recursos e iniciar a aplicação
10  * Esta classe contém Stage primaryStage e objetos para operações CRUD com banco de dados EntityManager e EntityManagerFactory*/
11
12 public class Aplicacao extends Application {
13     public static List<Image> iconsImg = new ArrayList<Image>();
14     public static HashMap<String, FXMLLoader> listFrameRoot = new HashMap<String, FXMLLoader>();
15     private static EntityManagerFactory emf = Persistence.createEntityManagerFactory("aps");
16     public static EntityManager em = emf.createEntityManager();
17     public static Stage primaryStage;
18
19     @Override
20     public void init() throws Exception {
21         super.init();
22     }
23
24     @Override
25     public void start(Stage stage){
26         Aps2022_6SemestreApplication.setPreloaderFlag(false);
27         try {
28             loadIcons();
29             primaryStage = stage;
30             listFrameRoot.put("Login", new FXMLLoader(Thread.currentThread().getContextClassLoader().getResource("com/view/models/Login.fxml")));
31             listFrameRoot.put("Cadastro", new FXMLLoader(Thread.currentThread().getContextClassLoader().getResource("com/view/models/Cadastro.fxml")));
32             listFrameRoot.put("Registro", new FXMLLoader(Thread.currentThread().getContextClassLoader().getResource("com/view/models/Registro.fxml")));
33             listFrameRoot.put("Acesso", new FXMLLoader(Thread.currentThread().getContextClassLoader().getResource("com/view/models/Acesso.fxml")));
34             stage.setScene(new Scene(listFrameRoot.get("Login").load(), 460, 487));
35             stage.setTitle("Login");
36             stage.setResizable(false);
37             stage.getIcons().addAll(iconsImg);
38             stage.show();
39         } catch (Exception e) {
40             e.printStackTrace();
41         }
42     }
43
44 }
45

```



```

54     }
55 }
56
57 public void loadIcons() {
58     Image icon = new Image(getClass().getResource("/Icons/Icone64.png").toString());
59     iconsImg.add(icon);
60 }
61
62 @Override
63 public void stop() throws Exception {
64     super.stop();
65     Platform.exit();
66     System.exit(0);
67 }

```

Fonte: Elaborada pelo autor.

Em conjunto com a classe Main essas classes são responsáveis por carregar recursos e iniciar a aplicação. Esta classe contém Stage primaryStage e objetos para operações CRUD com banco de dados EntityManager e EntityManagerFactory.

### Classe Alerts:

```

1 package com.source;
2
3
4 import java.util.Optional;
5
6
7
8
9
10
11
12 /**
13  * Classe para facilitar a exibição e retorno de informações. todos os metodos são estaticos e podem ser utilizados em qualquer classe
14  * desde que o thread responsável por chamar estes metodos sejam threads de aplicação JavaFX*/
15
16 public class Alerts {
17
18     private static Boolean b;
19
20     /**
21      * Exibe na tela uma informação de erro
22      * @param mensagem mensagem a ser exibida */
23     public static void showError(String mensagem) {
24         try {
25             Alert a = new Alert(AlertType.ERROR, mensagem);
26             a.setHeaderText(null);
27             a.showAndWait();
28         } catch (Exception e) {
29             e.printStackTrace();
30         }
31     }
32
33     /**
34      * Exibe na tela uma informação
35      * @param mensagem mensagem a ser exibida*/
36     public static void showInformation(String mensagem) {
37         try {
38             Alert a = new Alert(AlertType.INFORMATION, mensagem);
39             a.setHeaderText(null);
40             a.showAndWait();
41         } catch (Exception e) {
42             e.printStackTrace();
43         }
44     }
45 }

```

```

49  @return boolean true ou false
50  public static Boolean showConfirmation(String mensagem) {
51      try {
52          b = null;
53          Alert a = new Alert(AlertType.CONFIRMATION, mensagem);
54          a.setHeaderText(null);
55          Optional<ButtonType> ob = a.showAndWait();
56          ob.ifPresent(new Consumer<ButtonType>() {
57
58              @Override
59              public void accept(ButtonType t) {
60                  if(t == ButtonType.OK) b = true;
61                  else b = false;
62              }
63          });
64      } catch (Exception e) {
65          e.printStackTrace();
66      }
67      return b;
68  }
69
70  /**
71   * Exibe uma tela para usuario inserir dados em um TextField
72   * @param mensagem mensagem a ser exibida
73   * @param valor caso != null valor sera inserido no campo de input
74   * @return valor inserido*/
75  public static String showInput(String mensagem, String valor) {
76      try {
77          if(valor == null) valor = "";
78          TextInputDialog a = new TextInputDialog(valor);
79          a.setHeaderText(null);
80          a.setContentText(mensagem);
81          Optional<String> ob = a.showAndWait();
82          ob.orElse(null);
83          return ob.get();
84      } catch (Exception e) {
85          e.printStackTrace();
86      }
87      return null;
88  }

```

Fonte: Elaborada pelo autor.

Classe para facilitar a exibição e retorno de informações, todos os métodos são estáticos e podem ser utilizados em qualquer classe, desde que o thread responsável por chamar estes métodos sejam threads de aplicação. Nessa classe, temos o tratamento das exceções, retornando os respectivos erros ao usuário no caso de falha.

### Classe Biometria:

```

1  package com.source.control;
2
3  import org.bytedeco.opencv.global.opencv_core;
4
16
17
18  //Preprocessamento
19
20  //Histogram equalization
21
22  //GFTT corner detection
23  //SIFT Descriptor
24  //BruteForceMatcher
25  /**
26   * Esta classe contém os métodos necessários para processar e comparar impressões digitais*/
27  public class Biometria {
28
29      private float ratio = 0.7f;
30
31      //private GFTTDetector gf = GFTTDetector.create();
32      private SIFT sft = SIFT.create();
33      private BFMatcher bf = BFMatcher.create();
34
35
36  /**
37   * Realiza a comparação entre duas imagens utilizando os seguintes algoritmos: Good features to track para obter pontos de inter
38   * SIFT para detecção de descritores (descriptors) dos pontos de interesse e utiliza BruteForce como combinator para comparar os
39   * Não produz os resultados mais rápidos mas de acordo com testes realizados obtido uma maior precisão nos testes.
40   * @param img imagem para ser comparada
41   * @param img2 imagem para ser comparada
42   * @param preciEspera numero de combinações esperadas entre as duas imagens
43   * @return imagem mostrando visualmente a comparação ou null */
44  public Mat compare(Mat img, Mat img2, double preciEspera) {
45
46      KeyPointVector keyPointImg1 = new KeyPointVector();
47      KeyPointVector keyPointImg2 = new KeyPointVector();
48      Mat objectDescImg1 = new Mat();
49      Mat objectDescImg2 = new Mat();
50
51      //Detecta os pontos de interesse

```

```

51 //Detecta os pontos de interesse
52 sft.detect(img, keyPointImg1);
53 sft.detect(img2, keyPointImg2);
54 //Calcula os descritores de pontos de interesse
55 sft.compute(img, keyPointImg1, objectDescImg1);
56 sft.compute(img2, keyPointImg2, objectDescImg2);
57
58 DMatchVectorVector knnMatches = new DMatchVectorVector();
59
60 //Analiza as melhores combinações de descritores das duas imagens
61 bf.knnMatch(objectDescImg1, objectDescImg2, knnMatches, 2);
62
63 //Realiza um utilimo teste utilizando um valor ratio para filtrar melhores resultados
64 DMatchVectorVector listOfGoodMatches = new DMatchVectorVector();
65 for (int i = 0; i < knnMatches.size(); i++) {
66     if (knnMatches.get(i).size() > 1) {
67         DMatch[] matches = knnMatches.get(i).get();
68         if (matches[0].distance() < ratio * matches[1].distance()) {
69             listOfGoodMatches.push_back(knnMatches.get(i));
70         }
71     }
72 }
73 System.out.println(listOfGoodMatches.size());
74 System.out.println(knnMatches.size());
75
76 if(listOfGoodMatches.size() >= preciEspera) {
77     Mat imagemCompara = new Mat();
78     opencv_features2d.drawMatchesKnn(img, keyPointImg1, img2, keyPointImg2, listOfGoodMatches, imagemCompara, Scalar.all(-1), S
79     listOfGoodMatches.close();
80     return imagemCompara;
81 }
82 listOfGoodMatches.close();
83 return null;
84 }
85
86
87 /**
88  * Processa a imagem para o padrão utilizado na aplicação
89  * @param img imagem para ser processada

```

```

90 * @return imagem processada com tamanho de (250,300)*/
91 public static Mat processImagem(Mat img) {
92     Mat imagem = img.clone();
93     if(imagem.type() > 0) {
94         opencv_imgproc.cvtColor(imagem, imagem, opencv_imgproc.COLOR_BGR2GRAY);
95     }
96     //Obtem as dimensões da imagem digital
97     Rect rec = getBigContour(imagem);
98
99     Mat imt = new Mat(imagem.clone(), rec);
100     Mat temp2 = new Mat();
101     Mat temp3 = new Mat();
102     //Equaliza o histograma da imagem
103     opencv_imgproc.equalizeHist(imt, imt);
104
105     //Embraca a imagem aplicando um filtro Gaussian, e mistura a imagem original embacada e alguns pesos para obter uma imagem men
106     opencv_imgproc.GaussianBlur(imt, temp2, new Size(0,0), 20);
107     opencv_core.addWeighted(temp2, 2, imt, -1.1, 1, temp3);
108
109
110     //Ajusta o tamanho da imagem para um tamanho fixo
111     opencv_imgproc.resize(temp3, temp3, new Size(250,300));
112     //Equaliza o histograma da imagem
113     opencv_imgproc.equalizeHist(temp3.clone(), temp3);
114     temp2.close();
115     imt.close();
116     imagem.close();
117     return temp3;
118 }

```

Fonte: Elaborada pelo autor.

Esta classe contém os métodos necessários para processar e comparar impressões digitais. Realiza a comparação entre duas imagens utilizando os seguintes algoritmos: Good features to track para obter pontos de interesse(keypoints). SIFT para detecção de descritores (descriptors) dos pontos de interesse e utiliza BruteForce como combinador para comparar os descritores das duas imagens, utilizar um método BruteForce não produz os resultados mais rápidos, mas de acordo com testes realizados foi obtido uma maior precisão nos testes.

## Classe ControllerBD:

```

1 package com.source.control;
2
3 import java.util.stream.Stream;
4
5 /**
6  * Classe para operações CRUD genéricas, esta classe não é Thread safe e possui
7  * somente um EntityManager
8  */
9
10 public class ControllerBd {
11
12     public static EntityManager em = Aplicacao.em;
13     private static EntityTransaction trans = em.getTransaction();
14
15     public static void create(Object obj) {
16         try {
17             checkTrans();
18             begin();
19             em.persist(obj);
20             trans.commit();
21         } catch (PersistenceException e) {
22             checkTrans();
23             e.printStackTrace();
24         }
25     }
26
27     public static void delete(Object obj) {
28         try {
29             checkTrans();
30             begin();
31             em.remove(obj);
32             trans.commit();
33         } catch (PersistenceException e) {
34             checkTrans();
35             e.printStackTrace();
36         }
37     }
38
39     /**

```

```

53     */
54     public static boolean checkPersist(Object obj) {
55         try {
56             return em.contains(obj);
57         } catch (Exception e) {
58             e.printStackTrace();
59             return false;
60         }
61     }
62
63     public static Object findById(Class<?> classe, Integer id) throws IllegalArgumentException {
64         return em.find(classe, id);
65     }
66
67     public static Object findByIdDeatch(Class<?> classe, Integer id) throws IllegalArgumentException {
68         Object o = em.find(classe, id);
69         em.detach(o);
70         return o;
71     }
72
73     public static void commit() throws Exception {
74         if (trans.isActive()) {
75             trans.commit();
76         } else {
77             throw new Exception();
78         }
79     }
80
81     @SuppressWarnings("unchecked")
82     public static Stream<Acesso> getAcessoAsStream() {
83         return em.createQuery("SELECT a FROM ACESSO a").getResultStream();
84     }
85
86     public static void begin() throws PersistenceException {
87         checkTrans();
88         trans.begin();
89     }
90

```

```

91
92● public static void checkTrans() throws PersistenceException {
93     if (trans.isActive()) {
94         trans.rollback();
95     }
96 }
97
98
99● public static Boolean checkIfProibido(String agro) {
100     try {
101         em.createQuery("select 1 from AGROTOXICO a where a.agrotoxico = ?1 and a.proibido = true").setParameter(1, agro)
102             .getSingleResult();
103         return true;
104     } catch (Exception e) {
105         return false;
106     }
107 }
108
109● public static Agrototoxic findAgroByName(String agro) {
110     try {
111         return (Agrototoxic) em.createQuery("select a from AGROTOXICO a where a.agrotoxico = ?1").setParameter(1, agro).getSingleResult();
112     } catch (Exception e) {
113     }
114     return null;
115 }
116 }
117

```

Fonte: Elaborada pelo autor.

Essa Classe é utilizada para operações CRUD genéricas, esta classe não é Thread safe e possui somente um EntityManager.

### Classe Utilitários:

```

30 | * Esta classe contém diversos métodos estáticos necessários para detecção de rostos = outros métodos auxiliares
7  package com.source.control;
8
9  import org.bytedeco.javacpp.Pointer;
27
28  public class Utilitarios {
29
30      private static JavaFXFrameConverter fxConverter = new JavaFXFrameConverter();
31      private static OpenCVFrameConverter.ToMat converter = new OpenCVFrameConverter.ToMat();
32
33      private JavaFXFrameConverter fxConver = new JavaFXFrameConverter();
34      private OpenCVFrameConverter.ToMat conver = new OpenCVFrameConverter.ToMat();
35      /**
36       * Converte um vetor de pixels Mat para um objeto JavaFX Image
37       *
38       * @param Mat imagem para ser convertida
39       * @return imagem JavaFX
40       */
41      public synchronized static Image convertMatToImage(org.bytedeco.opencv.opencv_core.Mat grabbedImage) {
42          Frame fram = converter.convert(grabbedImage.clone());
43          System.out.println("Fram" + fram.imageWidth);
44          Image img = fxConverter.convert(fram);
45          fram.close();
46          return img;
47      }
48
49      public synchronized Image convertMatToImg(org.bytedeco.opencv.opencv_core.Mat grabbedImage) {
50          Frame fram = converter.convert(grabbedImage.clone());
51          System.out.println("Fram" + fram.imageWidth);
52          Image img = fxConver.convert(fram);
53          fram.close();
54          return img;
55      }
56      public static void showImage(Image img) {
57          Stage stage = new Stage();
58          StackPane pa = new StackPane();
59          ImageView vi = new ImageView();
60          pa.getChildren().add(vi);
61          stage.setScene(new Scene(pa, 500, 600));

```

```

62      vi.setFitHeight(img.getHeight());
63      vi.setFitWidth(img.getWidth());
64      vi.setImage(img);
65      stage.show();
66  }
67
68      public static byte[] getImagemAsByteArr(Mat img) {
69          // Aloca um array de bytes de acordo com o tamanho da imagem
70          byte[] bytes = new byte[((int) (img.total() * img.elemSize()))];
71          img.data().get(bytes);
72          return bytes;
73      }
74
75      /**
76       * Reconstrói a imagem a partir de um array de bytes
77       *
78       * @param rows quantidade de linhas contida na imagem original
79       * @param cols quantidade de colunas contida na imagem original
80       * @param type tipo da imagem original
81       * @param class1 array de bytes
82       * @return
83       */
84      public static Mat createMatByByteArr(int rows, int cols, int type, byte[] bytes) {
85          Mat imagemMat = new Mat(rows, cols, type);
86          imagemMat.data().put(bytes);
87          return imagemMat;
88      }
89      public static void showImage(Mat img) {
90          try {
91              if (img.type() == 0) {
92                  Mat img2 = new Mat();
93                  opencv_imgproc.cvtColor(img, img2, opencv_imgproc.COLOR_GRAY2BGR);
94                  showImage(convertMatToImage(img2));
95              } else {
96                  showImage(convertMatToImage(img));
97              }

```

```

99     } catch (Exception e) {
100         e.printStackTrace();
101     }
102 }
103
104 /**
105  * */
106 /**
107  * @param cas CascadeClassifier com classificadores
108  * @param grabbedImage imagem para ser processada
109  * @return Image retorna uma imagem com rostos contornados por retângulos
110  * @throws Exception
111  */
112 public static Image detectFaceRect(CascadeClassifier cas, Mat grabbedImage) throws Exception {
113     RectVector facesDetect = detectFaces(cas, grabbedImage);
114     System.out.println(facesDetect.get().length);
115     for (Rect rect : facesDetect.get()) {
116         System.out.println(rect.x() + " " + rect.y() + " " + rect.width() + " " + rect.height());
117         opencv_imgproc.rectangle(grabbedImage, new Point(rect.x(), rect.y()),
118             new Point(rect.x() + rect.width(), rect.y() + rect.height()), Scalar.GREEN, 1,
119             opencv_imgproc.LINE_AA, 0);
120     }
121     rect.close();
122     facesDetect.close();
123     return Utilitarios.convertMatToImage(grabbedImage);
124 }
125
126 /**
127  * Detecta rostos em uma imagem grabbedImage detecta qual o contorno com maior
128  * tamanho e retorna uma imagem tipo JavaFx
129  * */
130
131 * @param cas CascadeClassifier com classificadores
132 * @param grabbedImage imagem para ser processada
133 * @return Image retorna uma imagem com rosto contornado por um retângulo
134 * @throws Exception caso a imagem não contenha rostos ou imagem não estiver no
135 * formato correto
136 */

```

```

137 public static Image detectFacePrincipalRect(CascadeClassifier cas, Mat grabbedImage) throws Exception {
138     RectVector facesDetect = detectFaces(cas, grabbedImage);
139     System.out.println(facesDetect.get().length);
140     Rect rect = detectFacePrincipal(facesDetect);
141     Mat img = grabbedImage.clone();
142     System.out.println(rect.x() + " " + rect.y() + " " + rect.width() + " " + rect.height());
143     opencv_imgproc.rectangle(img, new Point(rect.x(), rect.y()),
144         new Point(rect.x() + rect.width(), rect.y() + rect.height()), Scalar.GREEN, 1, opencv_imgproc.LINE_AA,
145         0);
146     Image fxImg = convertMatToImage(img);
147     img.close();
148     rect.close();
149     facesDetect.close();
150     return fxImg;
151 }
152
153
154 }
155
156 /**
157  * Detecta a posição de rostos na imagem
158  * */
159 * @param cas Classificador cascade
160 * @param grabbedImage imagem original
161 * @return retorna um vetor de posição de todas as faces do frame
162 */
163 public static RectVector detectFaces(CascadeClassifier cas, Mat grabbedImage) {
164     System.out.println("Metodo detectFaces");
165     Mat imgGray = grabbedImage.clone();
166
167     System.out.println(grabbedImage.rows() + " rows");
168     System.out.println(grabbedImage.channels() + " channels");
169     if (grabbedImage.type() > 0) {
170         opencv_imgproc.cvtColor(grabbedImage, imgGray, opencv_imgproc.COLOR_BGR2GRAY);
171     }
172     RectVector facesDetect = new RectVector();
173     cas.detectMultiScale(imgGray, facesDetect);
174     System.out.println(facesDetect.size() + " size");
175     imgGray.close();

```

```

177         return facesDetect;
178     }
179
180     public static RectVector cloneRectVector(RectVector vector) {
181         RectVector ve = new RectVector(vector.size());
182         for (int i = 0; i < vector.size(); i++) {
183             ve.put(i, vector.get(i));
184         }
185         return ve;
186     }
187
188     /**
189      * Detecta o rosto com maior resolução
190      *
191      * @param facesDetectadas vetor contendo faces identificadas
192      * @return rosto com maior resolução
193      */
194     public static Rect detectFacePrincipal(RectVector facesDetectadas) {
195         Rect rostoPrimario = new Rect();
196
197         // Obtém o rosto detectado com maior resolução do frame
198         for (Rect f : facesDetectadas.get()) {
199             if (f.width() > rostoPrimario.width() && f.height() > rostoPrimario.height()) {
200                 rostoPrimario = new Rect(f);
201             }
202         }
203
204         System.out.println("RostoPrimario " + rostoPrimario.width());
205         return rostoPrimario;
206     }
207
208     /**
209      * Libera recursos
210      *
211      * @param args objetos para serem fechados
212      */
213     public static void releaseResources(Pointer... args) {
214
215         for (Pointer arg : args) {
216             try {
217                 arg.close();
218             } catch (Exception e) {
219                 e.printStackTrace();
220             }
221         }
222     }
223
224 }
225

```

Fonte: Elaborada pelo autor.

Esta classe contém diversos métodos estáticos necessários para detecção de rostos e outros métodos auxiliares. Nessa classe, podemos observar métodos responsáveis por alocar um vetor de bytes e reconstruir a imagem a partir do vetor de bytes, temos método conversor de pixels para um objeto JavaFX Image e outros utilitários, como detectar qual o contorno com maior tamanho, detectar a posição do rosto na imagem e detectar o rosto com maior resolução.



## Classe Acesso:

```

1 package com.source.model;
2
3 import java.io.Serializable;
4
5 /**
6  * Modelo para objeto Acesso que é utilizada para representar dados na TableView CRegistro e este modelo está mapeado com a tabela
7  * do banco de dados ACESSO*/
8 @Entity(name = "ACESSO")
9 public class Acesso implements Serializable {
10     private static final long serialVersionUID = 1L;
11
12     @Id
13     @GeneratedValue(strategy = GenerationType.IDENTITY)
14     @Column(name = "ID")
15     private Integer id;
16     @Column(name = "NOME", length = 100)
17     private String nome;
18     @Column(name = "NIVEL")
19     private int nivel;
20
21     @Lob
22     @Basic(fetch = FetchType.LAZY)
23     @Column(name = "IMAGEM")
24     private byte[] imagemByte;
25     @Column(name = "ROWS")
26     private int rows;
27     @Column(name = "COLUMNS")
28     private int col;
29     @Column(name = "TYPE")
30     private int type;
31     @Transient
32     private transient Mat imagemMat = new Mat();
33
34     public Acesso() {
35     }
36
37     public Acesso(Integer id, String nome, int nivel, byte[] imagemByte, int rows, int col, int type, Mat imagemMat) {
38         super();
39         this.id = id;
40         this.nome = nome;
41         this.nivel = nivel;
42         this.imagemByte = imagemByte;
43         this.rows = rows;
44         this.col = col;
45         this.type = type;
46         if (imagemMat != null) {
47             this.imagemMat = imagemMat.clone();
48         }
49     }
50
51     public void cloneImagem(Mat img) {
52         this.imagemMat = img.clone();
53     }
54
55     public int getId() {
56         return id;
57     }
58
59     public void setId(int id) {
60         this.id = id;
61     }
62
63     public String getNome() {
64         return nome;
65     }
66
67     public void setNome(String nome) {
68         this.nome = nome;
69     }
70
71     public int getNivel() {
72         return nivel;
73     }
74
75     public void setNivel(int nivel) {
76         this.nivel = nivel;
77     }
78
79     public byte[] getImagemByte() {
80         return imagemByte;
81     }
82
83     public void setImagemByte(byte[] imagemByte) {
84         this.imagemByte = imagemByte;
85     }
86
87     public int getRows() {
88         return rows;
89     }
90
91     public void setRows(int rows) {
92         this.rows = rows;
93     }
94
95     public int getCol() {
96         return col;
97     }
98
99     public void setCol(int col) {
100         this.col = col;
101     }
102
103     public int getType() {
104         return type;
105     }
106
107     public void setType(int type) {
108         this.type = type;
109     }
110
111     public Mat getImagemMat() {
112         return imagemMat;
113     }
114
115     public void setImagemMat(Mat imagemMat) {
116         this.imagemMat = imagemMat;
117     }
118 }

```

```

91
92 public int getNivel() {
93     return nivel;
94 }
95
96 public void setNivel(int nivel) {
97     this.nivel = nivel;
98 }
99
100 public byte[] getImagemByte() {
101     return imagemByte;
102 }
103
104 /**
105  * Obtem um array de bytes contendo todas as pixels contidas na imagem, para a
106  * imagem ser reconstruida é necessario armazenar a quantidade rows, columns e
107  * tipo da imagem
108  * @return byte[] contendo a imagem menos header
109  */
110
111 public byte[] getImagemAsByteArr() {
112     // Aloca um array de bytes de acordo com o tamanho da imagem
113     byte[] bytes = new byte[(((int) (imagemMat.total() * imagemMat.elemSize()))];
114     this.imagemMat.data().get(bytes);
115     return bytes;
116 }
117
118 public void setImagemByte(byte[] imagemByte) {
119     this.imagemByte = imagemByte;
120 }
121
122 public int getRows() {
123     return rows;
124 }
125
126 /**
127  * Reconstrói a imagem a partir de um array de bytes
128  */

```

```

129  * @param rows quantidade de linhas contida na imagem original
130  * @param cols quantidade de colunas contida na imagem original
131  * @param type tipo da imagem original
132  * @param arr array de bytes
133  */
134 public void setImagemByByteArr(int rows, int cols, int type, byte[] arr) {
135     imagemMat = new Mat(rows, cols, type);
136     this.imagemMat.data().put(arr);
137 }
138
139 public void setRows(int rows) {
140     this.rows = rows;
141 }
142
143 public int getCol() {
144     return col;
145 }
146
147 public void setCol(int col) {
148     this.col = col;
149 }
150
151 public int getType() {
152     return type;
153 }
154
155 public void setType(int type) {
156     this.type = type;
157 }
158
159 public Mat getImagemMat() {
160     return imagemMat;
161 }
162
163 public void setImagemMat(Mat imagemMat) {
164     this.imagemMat = imagemMat;
165     if (imagemMat.empty()) {
166         this.rows = imagemMat.rows();
167         this.col = imagemMat.cols();

```

Fonte: Elaborada pelo autor.

Essa classe é um modelo para objeto Acesso que é utilizado para representar dados na TableView CRegistro e este modelo esta mapeado com a tabela do banco de dados. Obtém um array de bytes contendo todos os pixels contidos na imagem, para a imagem ser reconstruida é necessário armazenar a quantidade de linhas, colunas e tipo de imagem. A partir dessa informações reconstrói a imagem, realizando a liberação do acesso.

## Classe Agrotóxico:

```

1 package com.source.model;
2
3 import java.io.Serializable;
4
14 /**
15  * Modelo para o objeto Acesso que é utilizado para representar uma lista de agrotóxicos e este modelo está mapeado com a tabela
16  * do banco de dados AGROTOXICO*/
17
18 @Entity(name = "AGROTOXICO")
19 public class Agrotoxico implements Serializable{
20     private static final long serialVersionUID = 1L;
21
22     @Id
23     @GeneratedValue(strategy = GenerationType.IDENTITY)
24     @Column(name = "ID")
25     private Integer id;
26     @Column(name = "AGROTOXICO",columnDefinition = "VARCHAR(150)",nullable = false)
27     private String agrotoxico;
28     @Column(name = "PROIBIDO",columnDefinition = "BOOLEAN",nullable = false)
29     private Boolean proibido;
30
31     @ManyToMany(mappedBy = "agrototoxicos")
32     private List<Cadastro> cadastro = new ArrayList<Cadastro>();
33     public Agrotoxico() {
34     }
35
36
37
38
39
40     public Agrotoxico(Integer id, String agrotoxico, Boolean proibido) {
41         super();
42         this.id = id;
43         this.agrotoxico = agrotoxico;
44         this.proibido = proibido;
45     }
46
47
48

```

```

50     public Integer getId() {
51         return id;
52     }
53
54     public void setId(Integer id) {
55         this.id = id;
56     }
57
58     public String getAgrotoxico() {
59         return agrotoxico;
60     }
61
62     public void setAgrotoxico(String agrotoxico) {
63         this.agrotoxico = agrotoxico;
64     }
65
66     public List<Cadastro> getCadastro() {
67         return cadastro;
68     }
69
70     public void setCadastro(List<Cadastro> cadastro) {
71         this.cadastro = cadastro;
72     }
73
74     public Boolean getProibido() {
75         return proibido;
76     }
77
78     public void setProibido(Boolean proibido) {
79         this.proibido = proibido;
80     }
81
82
83 }
84

```

Fonte: Elaborada pelo autor.

Essa classe é um Modelo para o objeto Acesso que é utilizado para representar uma lista de agrotóxicos e este modelo está mapeado com a tabela do banco de dados AGROTOXICO. É através dessa classe que os dados referentes aos agrotóxicos são armazenados na base de dados da aplicação.

## Classe Cadastro:

```

1 package com.source.model;
2
3 import java.io.Serializable;
4
5 /**
6  * Modelo para objecto Acesso que é utilizada para representar dados na TableView (Registro e esta modelo esta mapeado com a tabela
7  * do banco de dados CADASTRO)
8  */
9
10 @Entity(name = "CADASTRO")
11 public class Cadastro implements Serializable{
12     private static final long serialVersionUID = 1L
13     ;
14
15     @Id
16     @GeneratedValue(strategy = GenerationType.IDENTITY)
17     @Column(name = "ID")
18     private int id;
19
20     @Column(name = "UNIDADE", columnDefinition = "VARCHAR(150)", nullable = false)
21     private String unidade;
22
23     @Column(name = "PROD_ANUAL", columnDefinition = "DECIMAL(18,2)", nullable = false)
24     private Double prodAnual;
25
26     @Column(name = "N_EMPREGADOS", nullable = false)
27     private Integer nEmpregados;
28
29     @Column(name = "DESTINO_PROD", length = 30, nullable = false)
30     private String destino;
31
32     @Column(name = "NIVEL_AUTO", nullable = false)
33     private Integer nivelAuto;
34
35     @Column(name = "QUANT_MAQUINAS", nullable = false)
36     private Integer quantiMaquinas;
37
38     //Propriedade
39     @Column(name = "CIDADE", length = 60, nullable = false)
40     private String cidade;
41
42     @Column(name = "CEP", length = 10, nullable = false)
43     private String cep;
44
45     @Column(name = "ENDERECO", length = 100, nullable = false)
46     private String endereco;
47
48     @Column(name = "ESTADO", columnDefinition = "CHAR(2)", nullable = false)
49     private String estado;
50
51     @Column(name = "PAIS", length = 30, nullable = false)
52     private String pais;

```

```

51 //Dados
52 @Column(name = "INCE_FISCA_RECE", columnDefinition = "DECIMAL(18,2)", nullable = false)
53 private Double inceFiscaRece;
54 @Column(name = "IMP_MUNI_PAGO", columnDefinition = "DECIMAL(18,2)", nullable = false)
55 private Double impMuniPagos;
56 @Column(name = "IMP_ESTADU_RECOLHIDOS", columnDefinition = "DECIMAL(18,2)", nullable = false)
57 private Double impEstaduRecolhidos;
58 @Column(name = "IMP_FED_PAGO", columnDefinition = "DECIMAL(18,2)", nullable = false)
59 private Double impFedPago;
60 @Column(name = "TAXAS_FED", columnDefinition = "DECIMAL(18,2)", nullable = false)
61 private Double taxasFed;
62 @Column(name = "CONTEM_PROIB", columnDefinition = "BOOLEAN")
63 private Boolean contemProibido = false;
64 //Agrotóxico
65 @ManyToMany
66 @JoinTable(name = "CADASTRO_AGRO", joinColumns = @JoinColumn(columnDefinition = "CADASTRO_ID"),
67 inverseJoinColumns = @JoinColumn(columnDefinition = "AGROTOXICO_ID"))
68 private List<Agrotoxico> agrototoxicos = new ArrayList<Agrotoxico>();
69
70 public Cadastro() {
71
72 }
73
74 public Cadastro(int id,String unidade ,Double prodAnual, Integer nEmpregados, String destino, Integer nivelAuto,
75 Integer quantiMaquinas, String cidade, String cep, String endereco, String estado, String pais,
76 Double inceFiscaRece, Double impMuniPagos, Double impEstaduRecolhidos, Double impFedPago, Double taxasFed,
77 List<Agrotoxico> agrototoxicos, Boolean contemPro) {
78     super();
79     this.id = id;
80     this.unidade = unidade;
81     this.prodAnual = prodAnual;
82     this.nEmpregados = nEmpregados;
83     this.destino = destino;
84     this.nivelAuto = nivelAuto;
85     this.quantiMaquinas = quantiMaquinas;
86     this.cidade = cidade;
87     this.cep = cep;
88     this.endereco = endereco;
89     this.estado = estado;

```

```

90     this.pais = pais;
91     this.inceFiscaRece = inceFiscaRece;
92     this.impMuniPagos = impMuniPagos;
93     this.impEstaduRecolhidos = impEstaduRecolhidos;
94     this.impFedPago = impFedPago;
95     this.texasFed = texasFed;
96     this.agrotoxicos = agrotoxicos;
97     this.contemProibido = contemPro;
98 }
99
100
101 public Cadastro(int id, String unidade, String destino, String cidade, String estado) {
102     super();
103     this.id = id;
104     this.unidade = unidade;
105     this.destino = destino;
106     this.cidade = cidade;
107     this.estado = estado;
108 }
109
110 public int getId() {
111     return id;
112 }
113
114 public String getUnidade() {
115     return unidade;
116 }
117
118 public void setUnidade(String unidade) {
119     this.unidade = unidade;
120 }
121
122 public void setId(int id) {
123     this.id = id;
124 }
125
126 public Double getProdAnual() {
127     return prodAnual;
128 }

```

```

129 public String getProdAnualStr() {
130     try {
131         return Double.toString(prodAnual);
132     } catch (Exception e) {
133         e.printStackTrace();
134     }
135     return null;
136 }
137
138 public void setProdAnual(Double prodAtual) {
139     this.prodAnual = prodAtual;
140 }
141
142 public Integer getnEmpregados() {
143     return nEmpregados;
144 }
145
146 public String getnEmpregadosStr() {
147     try {
148         return Integer.toString(nEmpregados);
149     } catch (Exception e) {
150         e.printStackTrace();
151     }
152     return null;
153 }
154
155 public void setnEmpregados(Integer nEmpregados) {
156     this.nEmpregados = nEmpregados;
157 }
158
159 public String getDestino() {
160     return destino;
161 }
162
163 public void setDestino(String destino) {
164     this.destino = destino;
165 }
166
167 public Integer getNivelAuto() {

```

```
168         return nivelAuto;
169     }
170     public String getNivelAutoStr() {
171         try {
172             return Integer.toString(nivelAuto);
173         } catch (Exception e) {
174             e.printStackTrace();
175         }
176         return null;
177     }
178
179
180
181     public void setNivelAuto(Integer nivelAuto) {
182         this.nivelAuto = nivelAuto;
183     }
184
185     public Integer getQantiMaquinas() {
186         return qantiMaquinas;
187     }
188     public String getQantiMaquinasStr() {
189         try {
190             return Integer.toString(qantiMaquinas);
191         } catch (Exception e) {
192             e.printStackTrace();
193         }
194         return null;
195     }
196
197     public void setQantiMaquinas(Integer qantiMaquinas) {
198         this.qantiMaquinas = qantiMaquinas;
199     }
200
201
202     public String getCidade() {
203         return cidade;
204     }
205
206     public void setCidade(String cidade) {
```

```
207         this.cidade = cidade;
208     }
209
210     public String getCep() {
211         return cep;
212     }
213
214     public void setCep(String cep) {
215         this.cep = cep;
216     }
217
218     public String getEndereco() {
219         return endereco;
220     }
221
222     public void setEndereco(String endereco) {
223         this.endereco = endereco;
224     }
225
226     public String getEstado() {
227         return estado;
228     }
229
230     public void setEstado(String estado) {
231         this.estado = estado;
232     }
233
234     public String getPais() {
235         return pais;
236     }
237
238     public void setPais(String pais) {
239         this.pais = pais;
240     }
241
242     public Double getInceFiscaRece() {
243         return inceFiscaRece;
244     }
245     public String getInceFiscaReceStr() {
```

```

246         try {
247             return Double.toString(inceFiscaRece);
248         } catch (Exception e) {
249             e.printStackTrace();
250         }
251         return null;
252     }
253
254
255     public void setInceFiscaRece(Double inceFiscaRece) {
256         this.inceFiscaRece = inceFiscaRece;
257     }
258
259     public Double getImpMuniPagos() {
260         return impMuniPagos;
261     }
262     public String getImpMuniPagosStr() {
263         try {
264             return Double.toString(impMuniPagos);
265         } catch (Exception e) {
266             e.printStackTrace();
267         }
268         return null;
269     }
270
271     public void setImpMuniPagos(Double impMuniPagos) {
272         this.impMuniPagos = impMuniPagos;
273     }
274
275     public Double getImpEstaduRecolhidos() {
276         return impEstaduRecolhidos;
277     }
278     public String getImpEstaduRecolhidosStr() {
279         try {
280             return Double.toString(impEstaduRecolhidos);
281         } catch (Exception e) {
282             e.printStackTrace();
283         }
284         return null;

```

```

285     }
286
287     public void setImpEstaduRecolhidos(Double impEstaduRecolhidos) {
288         this.impEstaduRecolhidos = impEstaduRecolhidos;
289     }
290
291     public Double getImpFedPago() {
292         return impFedPago;
293     }
294     public String getImpFedPagoStr() {
295         try {
296             return Double.toString(impFedPago);
297         } catch (Exception e) {
298             e.printStackTrace();
299         }
300         return null;
301     }
302
303     public void setImpFedPago(Double impFedPago) {
304         this.impFedPago = impFedPago;
305     }
306
307     public Double getTaxasFed() {
308         return taxasFed;
309     }
310
311     public String getTaxasFedStr() {
312         try {
313             return Double.toString(taxasFed);
314         } catch (Exception e) {
315             e.printStackTrace();
316         }
317         return null;
318     }
319
320     public void setTaxasFed(Double taxasFed) {
321         this.taxasFed = taxasFed;
322     }
323

```

```

324 public List<Agrotoxico> getAgrotoxicos() {
325     return agrotoxicos;
326 }
327
328 public void setAgrotoxicos(List<Agrotoxico> agrotoxicos) {
329     this.agrotoxicos = agrotoxicos;
330 }
331
332 public void addAgrotoxica(Agrotoxico agro) {
333     agrotoxicos.add(agro);
334     agro.getCadastro().add(this);
335 }
336 public void removeAgrotoxica(Agrotoxico agro) {
337     agrotoxicos.remove(agro);
338     agro.getCadastro().remove(this);
339 }
340
341 public void checkContemProibido() {
342     boolean bol = false;
343     for(Agrotoxico agro : agrotoxicos) {
344         if(agro.getProibido()) bol = true;
345     }
346     contemProibido = bol;
347 }
348 public Boolean getContemProibido() {
349     return contemProibido;
350 }
351
352 public void setContemProibido(Boolean contemProibido) {
353     this.contemProibido = contemProibido;
354 }
355
356
357
358 }
359

```

Fonte: Elaborada pelo autor.

A classe Cadastro é responsável por obter os dados referente ao formulário de preenchimento. Modelo para o objeto Acesso que é utilizado para representar dados na TableView CRegistro e este modelo esta mapeado com a tabela do banco de dados CADASTRO.



## Classe TableHelpers:

```

1 package com.source.model;
2
3 import javafx.scene.control.Control;
4
5 /**
6  * Classe auxiliar na criação das tabelas CRegistro.tablePro e CRegistro.tableAce
7  */
8 public class TableHelpers {
9
10     public static class TableAceHelper {
11
12         public TableColumn<Acesso, Integer> getIdColumn() {
13             TableColumn<Acesso, Integer> col = new TableColumn<Acesso, Integer>("Id");
14             col.setCellValueFactory(new PropertyValueFactory<Acesso, Integer>("id"));
15             col.setMinWidth(60);
16             return col;
17         }
18
19         public TableColumn<Acesso, String> getNomeColumn() {
20             TableColumn<Acesso, String> col = new TableColumn<Acesso, String>("Nome");
21             col.setCellValueFactory(new PropertyValueFactory<Acesso, String>("nome"));
22             col.setMinWidth(100);
23             col.setPrefWidth(Control.USE_COMPUTED_SIZE);
24             col.setPrefWidth(20000);
25             return col;
26         }
27
28         public TableColumn<Acesso, String> getNivelColumn() {
29             TableColumn<Acesso, String> col = new TableColumn<Acesso, String>("Nível de Acesso");
30             col.setCellValueFactory(new PropertyValueFactory<Acesso, String>("nivel"));
31             col.setMinWidth(70);
32             return col;
33         }
34     }
35
36     public static class TableProHelper {
37
38         private String iconMorte = getClass().getResource("/Icons/morte.png").toString();
39
40     }
41
42     }
43
44     private String iconSafe = getClass().getResource("/Icons/confirm.png").toString();
45
46     public TableColumn<Cadastro, Integer> getIdColumn() {
47         TableColumn<Cadastro, Integer> col = new TableColumn<Cadastro, Integer>("Id");
48         col.setCellValueFactory(new PropertyValueFactory<Cadastro, Integer>("id"));
49         col.setMinWidth(60);
50         return col;
51     }
52
53     public TableColumn<Cadastro, String> getRazaoColumn() {
54         TableColumn<Cadastro, String> col = new TableColumn<Cadastro, String>("Nome/Razão Social");
55         col.setCellValueFactory(new PropertyValueFactory<Cadastro, String>("unidade"));
56         col.setMinWidth(100);
57         col.setPrefWidth(Control.USE_COMPUTED_SIZE);
58         col.setMaxWidth(20000);
59         return col;
60     }
61
62     public TableColumn<Cadastro, String> getEstadoColumn() {
63         TableColumn<Cadastro, String> col = new TableColumn<Cadastro, String>("Cidade");
64         col.setCellValueFactory(new PropertyValueFactory<Cadastro, String>("cidade"));
65         col.setMinWidth(60);
66         return col;
67     }
68
69     public TableColumn<Cadastro, String> getNivelColumn() {
70         TableColumn<Cadastro, String> col = new TableColumn<Cadastro, String>("Estado");
71         col.setCellValueFactory(new PropertyValueFactory<Cadastro, String>("estado"));
72         col.setMinWidth(50);
73         return col;
74     }
75
76     public TableColumn<Cadastro, String> getRamoColumn() {
77         TableColumn<Cadastro, String> col = new TableColumn<Cadastro, String>("Destino");
78         col.setCellValueFactory(new PropertyValueFactory<Cadastro, String>("destino"));
79         col.setMinWidth(150);
80         return col;
81     }
82
83     }
84

```

```

85● public TableColumn<Cadastro, Boolean> getSafeColumn() {
86     TableColumn<Cadastro, Boolean> col = new TableColumn<Cadastro, Boolean>("Status");
87     col.setCellValueFactory(new PropertyValueFactory<Cadastro, Boolean>("contemProibido"));
88     col.setMinWidth(40);
89● col.setCellFactory(new Callback<TableColumn<Cadastro, Boolean>, TableCell<Cadastro, Boolean>>() {
90
91●
92     @Override
93     public TableCell<Cadastro, Boolean> call(TableColumn<Cadastro, Boolean> param) {
94●         return new TableCell<Cadastro, Boolean>(){
95             @Override
96             protected void updateItem(Boolean item, boolean empty) {
97                 super.updateItem(item, empty);
98                 if(!empty) {
99                     if(item != null) {
100                         if(item){
101                             setGraphic(new StackPane(new ImageView(iconMorte)));
102                         }else {
103                             setGraphic(new StackPane(new ImageView(iconSafe)));
104                         }
105                     }else setGraphic(null);
106                 }
107             }
108         };
109     });
110 }
111
112 return col;
113 }
114
115 }
116
117● public static class TableAgroHelper{
118
119     private String iconMorte = getClass().getResource("/Icons/morte.png").toString();
120     private String iconSafe = getClass().getResource("/Icons/confirm.png").toString();
121
122
123● public TableColumn<Agrotoxico, String> getColumnAgro() {
124
125     TableColumn<Agrotoxico, String> col = new TableColumn<Agrotoxico, String>("Agrotoxico");
126     col.setCellValueFactory(new PropertyValueFactory<Agrotoxico, String>("agrotoxico"));
127     col.setMinWidth(210);
128     return col;
129 }
130
131● public TableColumn<Agrotoxico, Boolean> getColumnProib() {
132     TableColumn<Agrotoxico, Boolean> col = new TableColumn<Agrotoxico, Boolean>("Liberado");
133     col.setCellValueFactory(new PropertyValueFactory<Agrotoxico, Boolean>("proibido"));
134     col.setMinWidth(60);
135● col.setCellFactory(new Callback<TableColumn<Agrotoxico, Boolean>, TableCell<Agrotoxico, Boolean>>() {
136
137●
138     @Override
139     public TableCell<Agrotoxico, Boolean> call(TableColumn<Agrotoxico, Boolean> param) {
140●         return new TableCell<Agrotoxico, Boolean>(){
141             @Override
142             protected void updateItem(Boolean item, boolean empty) {
143                 super.updateItem(item, empty);
144                 if(!empty) {
145                     if(item != null) {
146                         if(item){
147                             setGraphic(new StackPane(new ImageView(iconMorte)));
148                         }else {
149                             setGraphic(new StackPane(new ImageView(iconSafe)));
150                         }
151                     }else setGraphic(null);
152                 }
153             }
154         };
155     });
156     return col;
157 }
158
159 }
160
161 }

```

Fonte: Elaborada pelo autor.

Classe auxiliar na criação das tabelas CRegistro.tablePro e CRegistro.tableAce.

## Classe CAcesso:

```

1 package com.view.controllers;
2
3 import java.net.URL;
4
5 /**
6  * Classe que é utilizada como controlador de listeners para interface GUI
7  * Acesso.fxml
8  */
9
10 public class CAcesso implements Initializable{
11
12     @FXML
13     private AnchorPane parentPane;
14     @FXML
15     private TextField txtUsuario;
16     @FXML
17     private ComboBox<String> comboNivel;
18     @FXML
19     private ImageView img;
20     @FXML
21     private Button btnCancelar;
22
23     @FXML
24     private Button btnSalvar;
25     @FXML
26     private Button btnSalvarImg;
27
28     private Mat imagem;
29
30     private Acesso acesso;
31
32     private Border bordaVerm = new Border(new BorderStroke(Color.RED, BorderStrokeStyle.SOLID, new CornerRadii(3),
33         new BorderWidths(2), new Insets(-2)));
34     private Border bordaDef;
35
36     @Override
37     public void initialize(URL location, ResourceBundle resources) {
38         bordaDef = txtUsuario.getBorder();
39         txtUsuario.textProperty().addListener(new StringList(txtUsuario));
40         comboNivel.setItems(FXCollections.observableArrayList(new String[] { "1", "2", "3" }));
41     }
42

```

```

43     public void construtor(Acesso acesso) {
44         this.acesso = acesso;
45         if(this.acesso != null) {
46             this.acesso = (Acesso) ControllerBd.findById(Acesso.class, acesso.getId());
47             setValues();
48         }
49     }
50
51     @FXML
52     private void actBtnCancelar() {
53         resetTxts();
54         ((Stage) parentPane.getScene().getWindow()).close();
55     }
56
57     @FXML
58     private void actBtnSalvar() {
59         try {
60             boolean flag = true;
61
62             if (acesso == null) {
63                 acesso = new Acesso();
64                 flag = false;
65             }
66
67             ControllerBd.begin();
68
69             if (txtUsuario.getText() == null && txtUsuario.getText().isEmpty() && txtUsuario.getText().isBlank()) {
70                 Alerts.showError("Campo Nome não pode estar vazio");
71                 throw new Exception();
72             }
73
74             acesso.setName(txtUsuario.getText());
75             acesso.setNivel(Integer.parseInt(comboNivel.getSelectionModel().getSelectedItem()));
76             if(img.getImage() == null) {
77                 Alerts.showError("Campo Imagem não pode estar vazio");
78                 throw new Exception();
79             }
80             if(imagem != null) {
81                 acesso.setImagemMat(imagem);
82             }
83         } catch (Exception e) {
84             e.printStackTrace();
85         }
86     }
87

```

```

112         acesso.setCol(imagem.cols());
113         acesso.setRows(imagem.rows());
114         acesso.setType(imagem.type());
115         acesso.setImagemByte(acesso.getImagemAsByteArr());
116     }
117
118     System.out.println(acesso.getNivel());
119     if(!flag) ControllerBd.em.persist(acesso);
120     ControllerBd.commit();
121
122     CRegistro.refreshTableAce();
123     resetTxts();
124     ((Stage) parentPane.getScene().getWindow()).close();
125
126 } catch (NumberFormatException e) {
127     e.printStackTrace();
128
129 } catch (Exception e) {
130     Alerts.showError("Falha ao cadastrar Acesso");
131     e.printStackTrace();
132 }
133
134 }
135
136 @FXML
137 private void actBtnSalvarImg() {
138     try {
139         imagem = opencv_imgcodecs.imread(new FileChooser().showOpenDialog(null).getAbsolutePath(),
140             opencv_imgcodecs.IMREAD_GRAYSCALE);
141         Mat a = Biometria.processImagem(imagem);
142         imagem.close();
143         imagem = a.clone();
144         Mat temp = new Mat();
145         opencv_imgproc.cvtColor(a, temp, opencv_imgproc.COLOR_GRAY2BGR);
146         img.setImage(Utilitarios.convertMatToImage(temp));
147         temp.close();
148         a.close();
149     } catch (Exception e) {
150         Alerts.showError("Falha ao abrir imagem");
151
152         if (imagem != null || !imagem.empty())
153             imagem.close();
154         imagem = null;
155         e.printStackTrace();
156     }
157
158 private void setValues() {
159     txtUsuario.setText(acesso.getNome());
160     comboNivel.getSelectionModel().select(Integer.toString(acesso.getNivel()));
161     acesso.setImagemByByteArr(acesso.getRows(), acesso.getCol(), acesso.getType(), acesso.getImagemByte());
162     Mat temp = new Mat();
163     opencv_imgproc.cvtColor(acesso.getImagemMat(), temp, opencv_imgproc.COLOR_GRAY2BGR);
164     img.setImage(Utilitarios.convertMatToImage(temp));
165     temp.close();
166
167 public void setEditavel(boolean bol) {
168     txtUsuario.setEditable(bol);
169     btnSalvar.setDisable(!bol);
170     btnCancel.setDisable(!bol);
171     btnSalvarImg.setDisable(!bol);
172 }
173
174 public void resetTxts() {
175     acesso = null;
176     txtUsuario.setText("");
177     comboNivel.getSelectionModel().select(0);
178     if (imagem != null && imagem.empty())
179         imagem.close();
180     imagem = null;
181     img.setImage(null);
182 }
183
184 public void setAcesso(Acesso acesso) {
185     this.acesso = acesso;
186     txtUsuario.setText(acesso.getNome());
187     comboNivel.getSelectionModel().select(Integer.toString(acesso.getNivel()));
188 }
189
190 protected class StringList implements ChangeListener<String>{
191     private TextField f;
192
193     public StringList(TextField f) {
194         this.f = f;
195     }
196     @SuppressWarnings("null")
197     @Override
198     public void changed(ObservableValue<? extends String> observable, String oldValue, String newValue) {
199         if(newValue == null && newValue.isEmpty()) {
200             f.setBorder(bordaVerM);
201         }else {
202             f.setBorder(bordaDef);
203         }
204     }
205 }
206
207

```

Fonte: Elaborada pelo autor.

Classe que é utilizada como controlador de listeners para interface GUI(Inteface Gráfica do Usuário).

## Classe CCadastro:

```

1 package com.view.controllers;
2
3 import java.net.URL;
4
5 * Classe que é utilizada como controlador de listeners para interface GUI Cadastro.fxml */
6 public class CCadastro implements Initializable {
7
8     private GridPane parentPane;
9     private TextField txtUnidade;
10    private TextField txtProdAnual;
11    private TextField txtNEmpregados;
12    private ComboBox<String> comboDestino;
13    private TextField txtNivelAuto;
14    private TextField txtQMaquinas;
15    private TextField txtCidade;
16    private TextField txtCep;
17    private TextField txtEndereco;
18    private TextField txtPais;
19    private TextField txtEstado;
20    private Button btnSalvar;
21    private Button btnCancel;
22    private Button btnBanir;
23    private Button btnPermitir;
24    private TextField txtFiscaisRece;
25    private TextField txtMuniPagos;
26    private TextField txtEstaRece;
27    private TextField txtFedePagos;
28    private TextField txtTaxasFed;
29    private TextField txtAgro;
30    private Button btnRemover;
31    private Button btnAdicionar;
32    private TableView<Agrotoxico> listAgro;
33    private ObservableList<Agrotoxico> agroModel = FXCollections.observableArrayList();
34
35    //TODO agrotoxicos
36
37    private Cadastro c;
38    private Integer nivel;
39

```

```

94    private Border bordaVerm = new Border(new BorderStroke(Color.RED, BorderStrokeStyle.SOLID, new CornerRadii(3),
95        new BorderWidths(2), new Insets(-2)));
96    private Border bordaDef;
97
98    @SuppressWarnings("unchecked")
99    @Override
100    public void initialize(URL location, ResourceBundle resources) {
101        bordaDef = txtUnidade.getBorder();
102        setBordas();
103        listAgro.setItems(agroModel);
104        //Adiciona colunas na tabela
105        TableAgroHelper h = new TableAgroHelper();
106        listAgro.getColumns().addAll(h.getColumnAgro(), h.getColumnProib());
107
108    }
109
110    public void construtor(Cadastro cadastro, Integer nivel) {
111        this.c = cadastro;
112        this.nivel = nivel;
113        comboDestino.setItems(FXCollections.observableArrayList(new String[] { "Externo", "Interno" }));
114
115        ControllerBd.begin();
116        if (c != null) {
117            c = (Cadastro) ControllerBd.findById(Cadastro.class, c.getId());
118            setValues();
119        } else {
120            c = new Cadastro();
121        }
122    }
123
124
125
126    @FXML
127    public void actBtnSalvar() {
128        try {
129            boolean flag = true;
130
131            if (c.getId() == 0) { flag = false; }
132

```

```

133      c.setUnidade(txtUnidade.getText());
134      c.setProdAnual(Double.parseDouble(txtProdAnual.getText()));
135      c.setnEmpregados(Integer.parseInt(txtNEmpregados.getText()));
136      c.setDestino(comboDestino.getValue());
137      c.setNivelAuto(Integer.parseInt(txtNivelAuto.getText()));
138      c.setQntdMaquinas(Integer.parseInt(txtQMaquinas.getText()));
139      c.setCidade(txtCidade.getText());
140      c.setCep(txtCep.getText());
141      c.setEndereco(txtEndereco.getText());
142      c.setPais(txtPais.getText());
143      c.setEstado(txtEstado.getText());
144      if(nivel >= 2) {
145          c.setInceFiscaRece(Double.parseDouble(txtFiscaisRece.getText()));
146          c.setImpMuniPagos(Double.parseDouble(txtMuniPagos.getText()));
147          c.setImpEstaduRecolhidos(Double.parseDouble(txtEstaReco.getText()));
148          c.setImpFedPago(Double.parseDouble(txtFedePagos.getText()));
149          c.setTaxasFed(Double.parseDouble(txtTaxasFed.getText()));
150      } else {
151          c.setInceFiscaRece(0.0);
152          c.setImpMuniPagos(0.0);
153          c.setImpEstaduRecolhidos(0.0);
154          c.setImpFedPago(0.0);
155          c.setTaxasFed(0.0);
156      }
157
158      if(!flag) ControllerBd.em.persist(c);
159      else ControllerBd.em.merge(c);
160
161      ControllerBd.commit();
162
163      CRegistro.refreshTablePro();
164      c = null;
165      resetTxt();
166      ((Stage) parentPane.getScene().getWindow()).close();
167  } catch (Exception e) {
168      e.printStackTrace();
169      Alerts.showError("Falha ao inserir dados, cheque os campos");
170  }
171  }

```

```

174  @FXML
175  public void actBtnCancelar() {
176      resetTxt();
177      ControllerBd.checkTrans();
178      ((Stage) parentPane.getScene().getWindow()).close();
179  }
180
181
182  @FXML
183  public void actBtnRemover() {
184      int index = listAgro.getSelectionModel().getSelectedIndex();
185      if(index > -1) {
186          Agrotoxico agro = agroModel.get(index);
187          c.removeAgrotoxico(agro);
188          agroModel.remove(agro);
189      }
190  }
191
192  @FXML
193  public void actBtnBanir() {
194      int row = listAgro.getSelectionModel().getSelectedIndex();
195      if(row > -1) {
196          Agrotoxico agro = agroModel.get(row);
197          agro.setProibido(true);
198          c.checkContemProibido();
199          listAgro.refresh();
200      }
201  }
202  @FXML
203  public void actBtnPermitir() {
204      int row = listAgro.getSelectionModel().getSelectedIndex();
205      if(row > -1) {
206          Agrotoxico agro = agroModel.get(row);
207          agro.setProibido(false);
208          c.checkContemProibido();
209          listAgro.refresh();
210      }
211  }
212

```

```

213 @FXML
214 public void actBtnAdicionar() {
215     try {
216         String agroS = txtAgro.getText();
217         if (!agroS.isEmpty()) {
218             Agrototoxic agro = ControllerBd.findAgroByName(agroS);
219             if (agro == null) agro = new Agrototoxic(null, agroS, ControllerBd.checkIfProibido(agroS));
220             ControllerBd.em.persist(agro);
221             c.addAgrototoxic(agro);
222             c.checkContemProibido();
223             agroModel.add(agro);
224         }
225     } catch (Exception e) {
226         e.printStackTrace();
227     }
228 }
229 public void resetTxt() {
230     txtUnidade.setText("");
231     txtProdAnual.setText("");
232     txtNEmpregados.setText("");
233     comboDestino.getSelectionModel().select(0);
234     txtNivelAuto.setText("");
235     txtQMaquinas.setText("");
236     txtCidade.setText("");
237     txtCep.setText("");
238     txtEndereco.setText("");
239     txtPais.setText("");
240     txtEstado.setText("");
241     txtFiscaisRece.setText("");
242     txtMuniPagos.setText("");
243     txtEstaReco.setText("");
244     txtFedePagos.setText("");
245     txtTaxasFed.setText("");
246     nivel = null;
247     c = null;
248     agroModel.clear();
249     txtAgro.clear();
250     listAgro.refresh();
251 }

```

```

253 private void setValues() {
254     txtUnidade.setText(c.getUnidade());
255     txtProdAnual.setText(c.getProdAnualStr());
256     txtNEmpregados.setText(c.getNEmpregadosStr());
257     comboDestino.getSelectionModel().select(c.getDestino());
258     txtNivelAuto.setText(c.getNivelAutoStr());
259     txtQMaquinas.setText(c.getQantiMaquinasStr());
260     txtCidade.setText(c.getCidade());
261     txtCep.setText(c.getCep());
262     txtEndereco.setText(c.getEndereco());
263     txtPais.setText(c.getPais());
264     txtEstado.setText(c.getEstado());
265     if (nivel >= 2) {
266         txtFiscaisRece.setText(c.getInceFiscaReceStr());
267         txtMuniPagos.setText(c.getImpMuniPagosStr());
268         txtEstaReco.setText(c.getImpEstaduRecolhidosStr());
269         txtFedePagos.setText(c.getImpFedPagoStr());
270         txtTaxasFed.setText(c.getTaxasFedStr());
271     }
272     if (nivel >= 3) {
273         agroModel.addAll(c.getAgrototoxicos());
274     }
275 }
276
277
278
279 public void setEditavel(boolean bol) {
280     txtUnidade.setEditable(bol);
281     txtProdAnual.setEditable(bol);
282     txtNEmpregados.setEditable(bol);
283     txtNivelAuto.setEditable(bol);
284     txtQMaquinas.setEditable(bol);
285     txtCidade.setEditable(bol);
286     txtCep.setEditable(bol);
287     txtEndereco.setEditable(bol);
288     txtPais.setEditable(bol);
289     txtEstado.setEditable(bol);
290     if (bol && nivel >= 2) {
291         txtFiscaisRece.setEditable(bol);

```

```

292     txtMuniPagos.setEditable(bol);
293     txtEstaReco.setEditable(bol);
294     txtFedePagos.setEditable(bol);
295     txtTaxasFed.setEditable(bol);
296 }else {
297     txtFiscaisRece.setEditable(false);
298     txtMuniPagos.setEditable(false);
299     txtEstaReco.setEditable(false);
300     txtFedePagos.setEditable(false);
301     txtTaxasFed.setEditable(false);
302 }
303 if(bol && nivel >= 3) {
304     btnRemover.setDisable(false);
305     btnAdicionar.setDisable(false);
306 }else {
307     btnRemover.setDisable(true);
308     btnAdicionar.setDisable(true);
309 }
310 btnCancelar.setDisable(!bol);
311 btnSalvar.setDisable(!bol);
312 }
313
314 private void setBordas() {
315     // Bordas customizadas para campos valores que são double ou int ou string
316     txtUnidade.textProperty().addListener(new StringList(txtUnidade));
317     txtCidade.textProperty().addListener(new StringList(txtCidade));
318     txtCep.textProperty().addListener(new StringList(txtCep));
319     txtEndereco.textProperty().addListener(new StringList(txtEndereco));
320     txtPais.textProperty().addListener(new StringList(txtPais));
321     txtEstado.textProperty().addListener(new StringList(txtEstado));
322     txtProdAnual.textProperty().addListener(new DoubleList(txtProdAnual));
323     txtNEmpregados.textProperty().addListener(new IntList(txtNEmpregados));
324     txtNivelAuto.textProperty().addListener(new IntList(txtNivelAuto));
325     txtQMaquinas.textProperty().addListener(new IntList(txtQMaquinas));
326     txtFiscaisRece.textProperty().addListener(new DoubleList(txtFiscaisRece));
327     txtMuniPagos.textProperty().addListener(new DoubleList(txtMuniPagos));
328     txtEstaReco.textProperty().addListener(new DoubleList(txtEstaReco));
329     txtFedePagos.textProperty().addListener(new DoubleList(txtFedePagos));
330     txtTaxasFed.textProperty().addListener(new DoubleList(txtTaxasFed));

```

```

331 }
332
333 protected class StringList implements ChangeListener<String>{
334     private TextField f;
335
336     public StringList(TextField f) {
337         this.f = f;
338     }
339     @Override
340     public void changed(ObservableValue<? extends String> observable, String oldValue, String newValue) {
341         if(newValue == null || newValue.isEmpty()) {
342             f.setBorder(bordaVerm);
343         }else {
344             f.setBorder(bordaDef);
345         }
346     }
347 }
348 protected class DoubleList implements ChangeListener<String> {
349     private TextField f;
350
351     public DoubleList(TextField f) {
352         this.f = f;
353     }
354
355     @Override
356     public void changed(ObservableValue<? extends String> observable, String oldValue, String newValue) {
357         try {
358             Double.parseDouble(newValue);
359             f.setBorder(bordaDef);
360         } catch (NumberFormatException e) {
361             f.setBorder(bordaVerm);
362         }
363     }
364 }
365
366 }
367
368 protected class IntList implements ChangeListener<String> {
369

```

```

370     private TextField f;
371
372     public IntList(TextField f) {
373         this.f = f;
374     }
375
376     @Override
377     public void changed(ObservableValue<? extends String> observable, String oldValue, String newValue) {
378         try {
379             if(newValue.matches("\\d+")) f.setBorder(bordaDef);
380             else {
381                 f.setBorder(bordaVerm);
382             }
383         } catch (NumberFormatException e) {
384             f.setBorder(bordaVerm);
385         }
386     }
387 }
388
389 }
390 }
391

```

Fonte: Elaborada pelo autor.



Classe que é utilizada como controlador de listeners para interface GUI Cadastro.fxml. Nessa classe são declarados todas as variaveis do formulario, bem como também, as propriedades do design da interface do usuário.

### Classe CLogin:

```

1 package com.view.controllers;
2
3 import java.io.File;
4
5 /**
6  * Classe que é utilizada como controlador de listeners para interface GUI
7  * login.fxml
8  */
9 public class CLogin {
10
11     private TextField txtAnaliza;
12     private BorderPane parentPane;
13     private TextField txtSenhaMestra;
14     private Button btnLogin;
15     private ImageView imgViewInput;
16     private ImageView imgCompara;
17     private ImageView imgViewResul;
18     private Button btnCarregar;
19     private Button btnIniciarRe;
20     private Acesso acesso;
21     private Mat imagemInput;
22
23     public CLogin() {
24
25     }
26
27     @FXML
28     private void actBtnCarregar() {
29         try {
30
31             FileChooser cho = new FileChooser();
32             File f = cho.showOpenDialog(null);
33             Mat img = opencv_imgcodecs.imread(f.getAbsolutePath());
34             Mat show = img.clone();
35             imagemInput = Biometria.processImagem(show);
36             opencv_imgproc.cvtColor(imagemInput, show, opencv_imgproc.COLOR_GRAY2BGR);
37
38             imgViewInput.setImage(Utilitarios.convertMatToImage(show));
39
40         } catch (Exception e) {
41             Alert a = new Alert(AlertType.ERROR, "Formato de arquivo inválido");
42             a.setHeaderText(null);
43             a.showAndWait();
44             e.printStackTrace();
45         }
46     }
47
48     @FXML
49     private void actBtnIniciarRe() {
50         Stream<Acesso> stre = ControllerBd.getAcessoAsStream();
51         ThreadCompara t = new ThreadCompara(stre);
52         new Thread(t).start();
53     }
54
55     @FXML
56     private void actBtnLogin() {
57         String senha = txtSenhaMestra.getText();
58         if (senha != null && !senha.isEmpty() && senha.contentEquals("1020304050")) {
59             Acesso ace = new Acesso(null, "Master", 3, null, 0, 0, 0, null);
60             loginConf(ace);
61         } else {
62             Alerts.showError("Senha incorreta");
63         }
64     }
65
66     private class ThreadCompara extends Task<Void> {
67
68         private Stream<Acesso> stream;
69         private Integer iteracoes = 0;
70         private Biometria b = new Biometria();
71         private Mat imagemResult;
72         private Mat imagemCompa;
73         private Utilitarios ut = new Utilitarios();
74
75         public ThreadCompara(Stream<Acesso> stream) {
76             this.stream = stream;
77         }
78     }
79
80     private void loginConf(Acesso ace) {
81         // ...
82     }
83
84     private void showError(String msg) {
85         Alerts.showError(msg);
86     }
87
88     private void processImagem(Mat img) {
89         // ...
90     }
91
92     private void convertMatToImage(Mat img) {
93         // ...
94     }
95
96     private void showOpenDialog() {
97         // ...
98     }
99
100     private void imread(String path) {
101         // ...
102     }
103
104     private void clone(Mat img) {
105         // ...
106     }
107
108     private void processImagem(Mat img) {
109         // ...
110     }
111
112     private void.cvtColor(Mat img, Mat show) {
113         // ...
114     }
115
116     private void.setImage(Mat img) {
117         // ...
118     }
119
120     private void.setHeaderText(String msg) {
121         // ...
122     }
123
124     private void.showAndWait() {
125         // ...
126     }
127
128     private void.printStackTrace() {
129         // ...
130     }
131
132     private void.getAcessoAsStream() {
133         // ...
134     }
135
136     private void.getAcesso() {
137         // ...
138     }
139
140     private void.getSenhaMestra() {
141         // ...
142     }
143
144     private void.getAnaliza() {
145         // ...
146     }
147
148     private void.getparentPane() {
149         // ...
150     }
151
152     private void.getbtnLogin() {
153         // ...
154     }
155
156     private void.getimgViewInput() {
157         // ...
158     }
159
160     private void.getimgCompara() {
161         // ...
162     }
163
164     private void.getimgViewResul() {
165         // ...
166     }
167
168     private void.getbtnCarregar() {
169         // ...
170     }
171
172     private void.getbtnIniciarRe() {
173         // ...
174     }
175
176     private void.getacesso() {
177         // ...
178     }
179
180     private void.getimagemInput() {
181         // ...
182     }
183
184     private void.getiteracoes() {
185         // ...
186     }
187
188     private void.getb() {
189         // ...
190     }
191
192     private void.getimagemResult() {
193         // ...
194     }
195
196     private void.getimagemCompa() {
197         // ...
198     }
199
200     private void.getut() {
201         // ...
202     }
203
204     private void.getstream() {
205         // ...
206     }
207
208     private void.getthis() {
209         // ...
210     }
211
212     private void.getmsg() {
213         // ...
214     }
215
216     private void.getstream() {
217         // ...
218     }
219
220     private void.getpath() {
221         // ...
222     }
223
224     private void.getimg() {
225         // ...
226     }
227
228     private void.getshow() {
229         // ...
230     }
231
232     private void.getmat() {
233         // ...
234     }
235
236     private void.getshow() {
237         // ...
238     }
239
240     private void.getshow() {
241         // ...
242     }
243
244     private void.getshow() {
245         // ...
246     }
247
248     private void.getshow() {
249         // ...
250     }
251
252     private void.getshow() {
253         // ...
254     }
255
256     private void.getshow() {
257         // ...
258     }
259
260     private void.getshow() {
261         // ...
262     }
263
264     private void.getshow() {
265         // ...
266     }
267
268     private void.getshow() {
269         // ...
270     }
271
272     private void.getshow() {
273         // ...
274     }
275
276     private void.getshow() {
277         // ...
278     }
279
280     private void.getshow() {
281         // ...
282     }
283
284     private void.getshow() {
285         // ...
286     }
287
288     private void.getshow() {
289         // ...
290     }
291
292     private void.getshow() {
293         // ...
294     }
295
296     private void.getshow() {
297         // ...
298     }
299
300     private void.getshow() {
301         // ...
302     }
303
304     private void.getshow() {
305         // ...
306     }
307
308     private void.getshow() {
309         // ...
310     }
311
312     private void.getshow() {
313         // ...
314     }
315
316     private void.getshow() {
317         // ...
318     }
319
320     private void.getshow() {
321         // ...
322     }
323
324     private void.getshow() {
325         // ...
326     }
327
328     private void.getshow() {
329         // ...
330     }
331
332     private void.getshow() {
333         // ...
334     }
335
336     private void.getshow() {
337         // ...
338     }
339
340     private void.getshow() {
341         // ...
342     }
343
344     private void.getshow() {
345         // ...
346     }
347
348     private void.getshow() {
349         // ...
350     }
351
352     private void.getshow() {
353         // ...
354     }
355
356     private void.getshow() {
357         // ...
358     }
359
360     private void.getshow() {
361         // ...
362     }
363
364     private void.getshow() {
365         // ...
366     }
367
368     private void.getshow() {
369         // ...
370     }
371
372     private void.getshow() {
373         // ...
374     }
375
376     private void.getshow() {
377         // ...
378     }
379
380     private void.getshow() {
381         // ...
382     }
383
384     private void.getshow() {
385         // ...
386     }
387
388     private void.getshow() {
389         // ...
390     }
391
392     private void.getshow() {
393         // ...
394     }
395
396     private void.getshow() {
397         // ...
398     }
399
400     private void.getshow() {
401         // ...
402     }
403
404     private void.getshow() {
405         // ...
406     }
407
408     private void.getshow() {
409         // ...
410     }
411
412     private void.getshow() {
413         // ...
414     }
415
416     private void.getshow() {
417         // ...
418     }
419
420     private void.getshow() {
421         // ...
422     }
423
424     private void.getshow() {
425         // ...
426     }
427
428     private void.getshow() {
429         // ...
430     }
431
432     private void.getshow() {
433         // ...
434     }
435
436     private void.getshow() {
437         // ...
438     }
439
440     private void.getshow() {
441         // ...
442     }
443
444     private void.getshow() {
445         // ...
446     }
447
448     private void.getshow() {
449         // ...
450     }
451
452     private void.getshow() {
453         // ...
454     }
455
456     private void.getshow() {
457         // ...
458     }
459
460     private void.getshow() {
461         // ...
462     }
463
464     private void.getshow() {
465         // ...
466     }
467
468     private void.getshow() {
469         // ...
470     }
471
472     private void.getshow() {
473         // ...
474     }
475
476     private void.getshow() {
477         // ...
478     }
479
480     private void.getshow() {
481         // ...
482     }
483
484     private void.getshow() {
485         // ...
486     }
487
488     private void.getshow() {
489         // ...
490     }
491
492     private void.getshow() {
493         // ...
494     }
495
496     private void.getshow() {
497         // ...
498     }
499
500     private void.getshow() {
501         // ...
502     }
503
504     private void.getshow() {
505         // ...
506     }
507
508     private void.getshow() {
509         // ...
510     }
511
512     private void.getshow() {
513         // ...
514     }
515
516     private void.getshow() {
517         // ...
518     }
519
520     private void.getshow() {
521         // ...
522     }
523
524     private void.getshow() {
525         // ...
526     }
527
528     private void.getshow() {
529         // ...
530     }
531
532     private void.getshow() {
533         // ...
534     }
535
536     private void.getshow() {
537         // ...
538     }
539
540     private void.getshow() {
541         // ...
542     }
543
544     private void.getshow() {
545         // ...
546     }
547
548     private void.getshow() {
549         // ...
550     }
551
552     private void.getshow() {
553         // ...
554     }
555
556     private void.getshow() {
557         // ...
558     }
559
560     private void.getshow() {
561         // ...
562     }
563
564     private void.getshow() {
565         // ...
566     }
567
568     private void.getshow() {
569         // ...
570     }
571
572     private void.getshow() {
573         // ...
574     }
575
576     private void.getshow() {
577         // ...
578     }
579
580     private void.getshow() {
581         // ...
582     }
583
584     private void.getshow() {
585         // ...
586     }
587
588     private void.getshow() {
589         // ...
590     }
591
592     private void.getshow() {
593         // ...
594     }
595
596     private void.getshow() {
597         // ...
598     }
599
600     private void.getshow() {
601         // ...
602     }
603
604     private void.getshow() {
605         // ...
606     }
607
608     private void.getshow() {
609         // ...
610     }
611
612     private void.getshow() {
613         // ...
614     }
615
616     private void.getshow() {
617         // ...
618     }
619
620     private void.getshow() {
621         // ...
622     }
623
624     private void.getshow() {
625         // ...
626     }
627
628     private void.getshow() {
629         // ...
630     }
631
632     private void.getshow() {
633         // ...
634     }
635
636     private void.getshow() {
637         // ...
638     }
639
640     private void.getshow() {
641         // ...
642     }
643
644     private void.getshow() {
645         // ...
646     }
647
648     private void.getshow() {
649         // ...
650     }
651
652     private void.getshow() {
653         // ...
654     }
655
656     private void.getshow() {
657         // ...
658     }
659
660     private void.getshow() {
661         // ...
662     }
663
664     private void.getshow() {
665         // ...
666     }
667
668     private void.getshow() {
669         // ...
670     }
671
672     private void.getshow() {
673         // ...
674     }
675
676     private void.getshow() {
677         // ...
678     }
679
680     private void.getshow() {
681         // ...
682     }
683
684     private void.getshow() {
685         // ...
686     }
687
688     private void.getshow() {
689         // ...
690     }
691
692     private void.getshow() {
693         // ...
694     }
695
696     private void.getshow() {
697         // ...
698     }
699
700     private void.getshow() {
701         // ...
702     }
703
704     private void.getshow() {
705         // ...
706     }
707
708     private void.getshow() {
709         // ...
710     }
711
712     private void.getshow() {
713         // ...
714     }
715
716     private void.getshow() {
717         // ...
718     }
719
720     private void.getshow() {
721         // ...
722     }
723
724     private void.getshow() {
725         // ...
726     }
727
728     private void.getshow() {
729         // ...
730     }
731
732     private void.getshow() {
733         // ...
734     }
735
736     private void.getshow() {
737         // ...
738     }
739
740     private void.getshow() {
741         // ...
742     }
743
744     private void.getshow() {
745         // ...
746     }
747
748     private void.getshow() {
749         // ...
750     }
751
752     private void.getshow() {
753         // ...
754     }
755
756     private void.getshow() {
757         // ...
758     }
759
760     private void.getshow() {
761         // ...
762     }
763
764     private void.getshow() {
765         // ...
766     }
767
768     private void.getshow() {
769         // ...
770     }
771
772     private void.getshow() {
773         // ...
774     }
775
776     private void.getshow() {
777         // ...
778     }
779
780     private void.getshow() {
781         // ...
782     }
783
784     private void.getshow() {
785         // ...
786     }
787
788     private void.getshow() {
789         // ...
790     }
791
792     private void.getshow() {
793         // ...
794     }
795
796     private void.getshow() {
797         // ...
798     }
799
800     private void.getshow() {
801         // ...
802     }
803
804     private void.getshow() {
805         // ...
806     }
807
808     private void.getshow() {
809         // ...
810     }
811
812     private void.getshow() {
813         // ...
814     }
815
816     private void.getshow() {
817         // ...
818     }
819
820     private void.getshow() {
821         // ...
822     }
823
824     private void.getshow() {
825         // ...
826     }
827
828     private void.getshow() {
829         // ...
830     }
831
832     private void.getshow() {
833         // ...
834     }
835
836     private void.getshow() {
837         // ...
838     }
839
840     private void.getshow() {
841         // ...
842     }
843
844     private void.getshow() {
845         // ...
846     }
847
848     private void.getshow() {
849         // ...
850     }
851
852     private void.getshow() {
853         // ...
854     }
855
856     private void.getshow() {
857         // ...
858     }
859
860     private void.getshow() {
861         // ...
862     }
863
864     private void.getshow() {
865         // ...
866     }
867
868     private void.getshow() {
869         // ...
870     }
871
872     private void.getshow() {
873         // ...
874     }
875
876     private void.getshow() {
877         // ...
878     }
879
880     private void.getshow() {
881         // ...
882     }
883
884     private void.getshow() {
885         // ...
886     }
887
888     private void.getshow() {
889         // ...
890     }
891
892     private void.getshow() {
893         // ...
894     }
895
896     private void.getshow() {
897         // ...
898     }
899
900     private void.getshow() {
901         // ...
902     }
903
904     private void.getshow() {
905         // ...
906     }
907
908     private void.getshow() {
909         // ...
910     }
911
912     private void.getshow() {
913         // ...
914     }
915
916     private void.getshow() {
917         // ...
918     }
919
920     private void.getshow() {
921         // ...
922     }
923
924     private void.getshow() {
925         // ...
926     }
927
928     private void.getshow() {
929         // ...
930     }
931
932     private void.getshow() {
933         // ...
934     }
935
936     private void.getshow() {
937         // ...
938     }
939
940     private void.getshow() {
941         // ...
942     }
943
944     private void.getshow() {
945         // ...
946     }
947
948     private void.getshow() {
949         // ...
950     }
951
952     private void.getshow() {
953         // ...
954     }
955
956     private void.getshow() {
957         // ...
958     }
959
960     private void.getshow() {
961         // ...
962     }
963
964     private void.getshow() {
965         // ...
966     }
967
968     private void.getshow() {
969         // ...
970     }
971
972     private void.getshow() {
973         // ...
974     }
975
976     private void.getshow() {
977         // ...
978     }
979
980     private void.getshow() {
981         // ...
982     }
983
984     private void.getshow() {
985         // ...
986     }
987
988     private void.getshow() {
989         // ...
990     }
991
992     private void.getshow() {
993         // ...
994     }
995
996     private void.getshow() {
997         // ...
998     }
999
1000    private void.getshow() {
1001        // ...
1002    }
1003
1004    private void.getshow() {
1005        // ...
1006    }
1007
1008    private void.getshow() {
1009        // ...
1010    }
1011
1012    private void.getshow() {
1013        // ...
1014    }
1015
1016    private void.getshow() {
1017        // ...
1018    }
1019
1020    private void.getshow() {
1021        // ...
1022    }
1023
1024    private void.getshow() {
1025        // ...
1026    }
1027
1028    private void.getshow() {
1029        // ...
1030    }
1031
1032    private void.getshow() {
1033        // ...
1034    }
1035
1036    private void.getshow() {
1037        // ...
1038    }
1039
1040    private void.getshow() {
1041        // ...
1042    }
1043
1044    private void.getshow() {
1045        // ...
1046    }
1047
1048    private void.getshow() {
1049        // ...
1050    }
1051
1052    private void.getshow() {
1053        // ...
1054    }
1055
1056    private void.getshow() {
1057        // ...
1058    }
1059
1060    private void.getshow() {
1061        // ...
1062    }
1063
1064    private void.getshow() {
1065        // ...
1066    }
1067
1068    private void.getshow() {
1069        // ...
1070    }
1071
1072    private void.getshow() {
1073        // ...
1074    }
1075
1076    private void.getshow() {
1077        // ...
1078    }
1079
1080    private void.getshow() {
1081        // ...
1082    }
1083
1084    private void.getshow() {
1085        // ...
1086    }
1087
1088    private void.getshow() {
1089        // ...
1090    }
1091
1092    private void.getshow() {
1093        // ...
1094    }
1095
1096    private void.getshow() {
1097        // ...
1098    }
1099
1100    private void.getshow() {
1101        // ...
1102    }
1103
1104    private void.getshow() {
1105        // ...
1106    }
1107
1108    private void.getshow() {
1109        // ...
1110    }
1111
1112    private void.getshow() {
1113        // ...
1114    }
1115
1116    private void.getshow() {
1117        // ...
1118    }
1119
1120    private void.getshow() {
1121        // ...
1122    }
1123
1124    private void.getshow() {
1125        // ...
1126    }
1127
1128    private void.getshow() {
1129        // ...
1130    }
1131
1132    private void.getshow() {
1133        // ...
1134    }
1135
1136    private void.getshow() {
1137        // ...
1138    }
1139
1140    private void.getshow() {
1141        // ...
1142    }
1143
1144    private void.getshow() {
1145        // ...
1146    }
1147
1148    private void.getshow() {
1149        // ...
1150    }
1151
1152    private void.getshow() {
1153        // ...
1154    }
1155
1156    private void.getshow() {
1157        // ...
1158    }
1159
1160    private void.getshow() {
1161        // ...
1162    }
1163
1164    private void.getshow() {
1165        // ...
1166    }
1167
1168    private void.getshow() {
1169        // ...
1170    }
1171
1172    private void.getshow() {
1173        // ...
1174    }
1175
1176    private void.getshow() {
1177        // ...
1178    }
1179
1180    private void.getshow() {
1181        // ...
1182    }
1183
1184    private void.getshow() {
1185        // ...
1186    }
1187
1188    private void.getshow() {
1189        // ...
1190    }
1191
1192    private void.getshow() {
1193        // ...
1194    }
1195
1196    private void.getshow() {
1197        // ...
1198    }
1199
1200    private void.getshow() {
1201        // ...
1202    }
1203
1204    private void.getshow() {
1205        // ...
1206    }
1207
1208    private void.getshow() {
1209        // ...
1210    }
1211
1212    private void.getshow() {
1213        // ...
1214    }
1215
1216    private void.getshow() {
1217        // ...
1218    }
1219
1220    private void.getshow() {
1221        // ...
1222    }
1223
1224    private void.getshow() {
1225        // ...
1226    }
1227
1228    private void.getshow() {
1229        // ...
1230    }
1231
1232    private void.getshow() {
1233        // ...
1234    }
1235
1236    private void.getshow() {
1237        // ...
1238    }
1239
1240    private void.getshow() {
1241        // ...
1242    }
1243
1244    private void.getshow() {
1245        // ...
1246    }
1247
1248    private void.getshow() {
1249        // ...
1250    }
1251
1252    private void.getshow() {
1253        // ...
1254    }
1255
1256    private void.getshow() {
1257        // ...
1258    }
1259
1260    private void.getshow() {
1261        // ...
1262    }
1263
1264    private void.getshow() {
1265        // ...
1266    }
1267
1268    private void.getshow() {
1269        // ...
1270    }
1271
1272    private void.getshow() {
1273        // ...
1274    }
1275
1276    private void.getshow() {
1277        // ...
1278    }
1279
1280    private void.getshow() {
1281        // ...
1282    }
1283
1284    private void.getshow() {
1285        // ...
1286    }
1287
1288    private void.getshow() {
1289        // ...
1290    }
1291
1292    private void.getshow() {
1293        // ...
1294    }
1295
1296    private void.getshow() {
1297        // ...
1298    }
1299
1300    private void.getshow() {
1301        // ...
1302    }
1303
1304    private void.getshow() {
1305        // ...
1306    }
1307
1308    private void.getshow() {
1309        // ...
1310    }
1311
1312    private void.getshow() {
1313        // ...
1314    }
1315
1316    private void.getshow() {
1317        // ...
1318    }
1319
1320    private void.getshow() {
1321        // ...
1322    }
1323
1324    private void.getshow() {
1325        // ...
1326    }
1327
1328    private void.getshow() {
1329        // ...
1330    }
1331
1332    private void.getshow() {
1333        // ...
1334    }
1335
1336    private void.getshow() {
1337        // ...
1338    }
1339
1340    private void.getshow() {
1341        // ...
1342    }
1343
1344    private void.getshow() {
1345        // ...
1346    }
1347
1348    private void.getshow() {
1349        // ...
1350    }
1351
1352    private void.getshow() {
1353        // ...
1354    }
1355
1356    private void.getshow() {
1357        // ...
1358    }
1359
1360    private void.getshow() {
1361        // ...
1362    }
1363
1364    private void.getshow() {
1365        // ...
1366    }
1367
1368    private void.getshow() {
1369        // ...
1370    }
1371
1372    private void.getshow() {
1373        // ...
1374    }
1375
1376    private void.getshow() {
1377        // ...
1378    }
1379
1380    private void.getshow() {
1381        // ...
1382    }
1383
1384    private void.getshow() {
1385        // ...
1386    }
1387
1388    private void.getshow() {
1389        // ...
1390    }
1391
1392    private void.getshow() {
1393        // ...
1394    }
1395
1396    private void.getshow() {
1397        // ...
1398    }
1399
1400    private void.getshow() {
1401        // ...
1402    }
1403
1404    private void.getshow() {
1405        // ...
1406    }
1407
1408    private void.getshow() {
1409        // ...
1410    }
1411
1412    private void.getshow() {
1413        // ...
1414    }
1415
1416    private void.getshow() {
1417        // ...
1418    }
1419
1420    private void.getshow() {
1421        // ...
1422    }
1423
1424    private void.getshow() {
1425        // ...
1426    }
1427
1428    private void.getshow() {
1429        // ...
1430    }
1431
1432    private void.getshow() {
1433        // ...
1434    }
1435
1436    private void.getshow() {
1437        // ...
1438    }
1439
1440    private void.getshow() {
1441        // ...
1442    }
1443
1444    private void.getshow() {
1445        // ...
1446    }
1447
1448    private void.getshow() {
1449        // ...
1450    }
1451
1452    private void.getshow() {
1453        // ...
1454    }
1455
1456    private void.getshow() {
1457        // ...
1458    }
1459
1460    private void.getshow() {
1461        // ...
1462    }
1463
1464    private void.getshow() {
1465        // ...
1466    }
1467
1468    private void.getshow() {
1469        // ...
1470    }
1471
1472    private void.getshow() {
1473        // ...
1474    }
1475
1476    private void.getshow() {
1477        // ...
1478    }
1479
1480    private void.getshow() {
1481        // ...
1482    }
1483
1484    private void.getshow() {
1485        // ...
1486    }
1487
1488    private void.getshow() {
1489        // ...
1490    }
1491
1492    private void.getshow() {
1493        // ...
1494    }
1495
1496    private void.getshow() {
1497        // ...
1498    }
1499
1500    private void.getshow() {
1501        // ...
1502    }
1503
1504    private void.getshow() {
1505        // ...
1506    }
1507
1508    private void.getshow() {
1509        // ...
1510    }
1511
1512    private void.getshow() {
1513        // ...
1514    }
1515
1516    private void.getshow() {
1517        // ...
1518    }
1519
1520    private void.getshow() {
1521        // ...
1522    }
1523
1524    private void.getshow() {
1525        // ...
1526    }
1527
1528    private void.getshow() {
1529        // ...
1530    }
1531
1532    private void.getshow() {
1533        // ...
1534    }
1535
1536    private void.getshow() {
1537        // ...
1538    }
1539
1540    private void.getshow() {
1541        // ...
1542    }
1543
1544    private void.getshow() {
1545        // ...
1546    }
1547
1548    private void.getshow() {
1549        // ...
1550    }
1551
1552    private void.getshow() {
1553        // ...
1554    }
1555
1556    private void.getshow() {
1557        // ...
1558    }
1559
1560    private void.getshow() {
1561        // ...
1562    }
1563
1564    private void.getshow() {
1565        // ...
1566    }
1567
1568    private void.getshow() {
1569        // ...
1570    }
1571
1572    private void.getshow() {
1573        // ...
1574    }
1575
1576    private void.getshow() {
1577        // ...
1578    }
1579
1580    private void.getshow() {
1581        // ...
1582    }
1583
1584    private void.getshow() {
1585        // ...
1586    }
1587
1588    private void.getshow() {
1589        // ...
1590    }
1591
1592    private void.getshow() {
1593        // ...
1594    }
1595
1596    private void.getshow() {
1597        // ...
1598    }
1599
1600    private void.getshow() {
1601        // ...
1602    }
1603
1604    private void.getshow() {
1605        // ...
1606    }
1607
1608    private void.getshow() {
1609        // ...
1610    }
1611
1612    private void.getshow() {
1613        // ...
1614    }
1615
1616    private void.getshow() {
1617        // ...
1618    }
1619
1620    private void.getshow() {
1621        // ...

```

```

117
118 APS-2022-6-Semestre_APS-2022-6-Semestre/src/main/java/com/view/controllers/CCadastro.java
119 txtAnaliza.setText(Integer.toString(iteracoes));
120 stream.forEach(obj -> {
121
122     Mat img = Utilitarios.createMatByByteArr(obj.getRows(), obj.getCol(), obj.getType(),
123     obj.getImageByte());
124     Mat temp = b.compare(imagemInput, img, 20);
125     if (temp != null) {
126         acesso = obj;
127         imagemResult = img.clone();
128         imagemCompa = temp.clone();
129     }
130     iteracoes++;
131     txtAnaliza.setText(Integer.toString(iteracoes));
132 });
133
134 if (imagemResult != null) {
135     opencv_imgproc.cvtColor(imagemResult, imagemResult, opencv_imgproc.COLOR_GRAY2BGR);
136     imgViewResul.setImage(ut.convertMatToImg(imagemResult));
137     imgCompara.setImage(ut.convertMatToImg(imagemCompa));
138     loginConf(acesso);
139 } else {
140     Platform.runLater(() -> {
141         Alerts.showError("Acesso não autorizado");
142     });
143 }
144 return null;
145 }
146 }
147
148 private void loginConf(Acesso ass) {
149
150     Platform.runLater(() -> {
151         try {
152             Alerts.showInformation(
153                 String.format("Login confirmado, nível de acesso: %d + %s", ass.getNivel(), ass.getNome());
154             Stage stage = (Stage) parentPane.getScene().getWindow();
155
156             FXMLLoader root = Aplicacao.listFrameRoot.get("Registro");
157             stage.setScene(new Scene(root.load(), 600, 500));
158             stage.setResizable(true);
159             CRegistro controller = root.getController();
160             controller.construtor(ass);
161         } catch (IOException e) {
162             e.printStackTrace();
163         }
164     });
165 }
166 }
167
168 }
169

```

Fonte: Elaborada pelo autor.

Classe que é utilizada como controlador de listeners para interface GUI Login.fxml.

### Classe CRegistro:

```

1 package com.view.controllers;
2
3 import java.io.IOException;
4
5 /**
6  * Classe que é utilizada como controlador de listeners para interface GUI Registro.fxml*/
7
8 public class CRegistro implements Initializable {
9
10     public TabPane tabPane;
11     public GridPane parentPane;
12     public Button menuAdicionar;
13     public Button menuExibir;
14     public Button menuEditar;
15     public Button menuApagar;
16     public TableView<Cadastro> tablePro;
17     public TableView<Acesso> tableAce;
18     public TextField txtUsuario;
19     public TextField txtNivel;
20     private Acesso acessoAtual;
21     private Stage frameCadastro = new Stage();
22     private CCadastro controllerCadastro;
23     private Stage frameAcesso = new Stage();
24     private CAcesso controllerAcesso;
25     private static ObservableList<Cadastro> modelPropriedades = FXCollections.observableArrayList();
26     private static ObservableList<Acesso> modelAcessos = FXCollections.observableArrayList();
27
28     @Override
29     public void initialize(URL location, ResourceBundle resources) {
30         Aplicacao.primaryStage.setOnCloseRequest(event -> {
31             Platform.exit();
32         });
33         Aplicacao.primaryStage.centerOnScreen();
34         setTables();
35         refreshTableAce();
36         refreshTablePro();
37
38         try {
39             FXMLLoader root = Aplicacao.listFrameRoot.get("Cadastro");
40

```

```

76     frameCadastro.setScene(new Scene(root.load(), 650, 600));
77     frameCadastro.setResizable(false);
78     frameCadastro.getIcons().addAll(Aplicacao.iconsImg);
79     controlerCadastro = root.getController();
80     frameCadastro.setTitle("Cadastro");
81
82     root = Aplicacao.listFrameRoot.get("Acesso");
83     frameAcesso.setScene(new Scene(root.load(), 360, 360));
84     frameAcesso.setResizable(false);
85     frameAcesso.getIcons().addAll(Aplicacao.iconsImg);
86     controlerAcesso = root.getController();
87     frameAcesso.setTitle("Cadastro Acesso");
88
89     } catch (IOException e) {
90         e.printStackTrace();
91     }
92
93     }
94     public void construtor(Acesso acesso) {
95         this.acessoAtual = acesso;
96         txtUsuario.setText(acesso.getNome());
97         txtNivel.setText(Integer.toString(acesso.getNivel()));
98     }
99
100     @FXML
101     public void actMenuAdicionar() {
102         if (tabPane.getSelectionModel().getSelectedItem() == 0) {
103             controlerCadastro.resetTxt();
104             controlerCadastro.construtor(null, acessoAtual.getNivel());
105             controlerCadastro.setEditavel(true);
106             frameCadastro.show();
107             frameCadastro.centerOnScreen();
108         } else {
109             if (acessoAtual.getNivel() >= 3) {
110                 controlerAcesso.resetTxts();
111                 controlerAcesso.construtor(null);
112                 controlerAcesso.setEditavel(true);
113                 frameAcesso.show();
114                 frameAcesso.centerOnScreen();

```

```

115             } else Alerts.showError("Apenas usuarios nível 3 podem cadastrar acessos");
116         }
117     }
118 }
119
120 @FXML
121 public void actMenuExibir() {
122     if (tabPane.getSelectionModel().getSelectedItem() == 0) {
123         controlerCadastro.resetTxt();
124         controlerCadastro.construtor(tablePro.getSelectionModel().getSelectedItem(), acessoAtual.getNivel());
125         controlerCadastro.setEditavel(false);
126         frameCadastro.show();
127         frameCadastro.centerOnScreen();
128     } else {
129         if (acessoAtual.getNivel() >= 3) {
130             controlerAcesso.resetTxts();
131             controlerAcesso.construtor(tableAce.getSelectionModel().getSelectedItem());
132             controlerAcesso.setEditavel(false);
133             frameAcesso.show();
134             frameAcesso.centerOnScreen();
135         } else Alerts.showError("Apenas usuarios nível 3 podem Exibir acessos");
136     }
137 }
138
139
140 @FXML
141 public void actMenuEditar() {
142     if (tabPane.getSelectionModel().getSelectedItem() == 0) {
143         controlerCadastro.resetTxt();
144         controlerCadastro.construtor(tablePro.getSelectionModel().getSelectedItem(), acessoAtual.getNivel());
145         controlerCadastro.setEditavel(true);
146
147         frameCadastro.show();
148     } else {
149         if (acessoAtual.getNivel() >= 3) {
150             controlerAcesso.resetTxts();
151             controlerAcesso.construtor(tableAce.getSelectionModel().getSelectedItem());
152             controlerAcesso.setEditavel(true);
153             frameAcesso.show();

```

```

154         frameAcesso.centerOnScreen();
155     }else Alerts.showError("Apenas usuarios nivel 3 podem Editar acessos");
156 }
157 }
158
159 @FXML
160 public void actMenuApagar() {
161     try {
162         if(Alerts.showConfirmation("Deseja apagar linha selecionada")) {
163
164             if (tabPane.getSelectionModel().getSelectedIndex() == 0) {
165                 if(tablePro.getSelectionModel().getSelectedIndex() > -1) {
166                     Cadastro obj = tablePro.getSelectionModel().getSelectedItem();
167                     if(ControllerBd.checkPersist(obj)) {
168                         ControllerBd.delete(obj);
169                     }else {
170                         obj = (Cadastro) ControllerBd.findById(Cadastro.class, obj.getId());
171                         ControllerBd.delete(obj);
172                     }
173                 }
174                 refreshTablePro();
175             }else Alerts.showError("Nenhuma linha selecionada");
176
177         } else {
178             if(tableAce.getSelectionModel().getSelectedIndex() > -1) {
179                 Acesso obj = tableAce.getSelectionModel().getSelectedItem();
180                 if(ControllerBd.checkPersist(obj)) {
181                     ControllerBd.delete(obj);
182                 }else {
183                     obj = (Acesso) ControllerBd.findById(Acesso.class, obj.getId());
184                     ControllerBd.delete(obj);
185                 }
186             }
187             refreshTableAce();
188         }else Alerts.showError("Nenhuma linha selecionada");
189     }
190 }
191 }
192

```

```

201 public static void refreshTablePro() {
202     modelPropriedades.clear();
203     ControllerBd.checkTrans();
204     ControllerBd.em.clear();
205     @SuppressWarnings("unchecked")
206     List<Cadastro> l = Aplicacao.em.createQuery("select a from CADASTRO a ").getResultList();
207     if(l.size() > 0) {
208         for(Cadastro cad : l) {
209             //TODO Atualizar get contem proibido quando atualizar tabela
210             cad.checkContemProibido();
211             System.out.println(cad.getContemProibido());
212             modelPropriedades.add(cad);
213         }
214     }
215 }
216
217
218 public static void refreshTableAce() {
219     modelAcessos.clear();
220     ControllerBd.checkTrans();
221     @SuppressWarnings("unchecked")
222     List<Object[]> l = Aplicacao.em.createQuery("select a.id,a.nivel,a.nome from ACESSO a").getResultList();
223     if(l.size() > 0) {
224         for(Object[] a : l) {
225             modelAcessos.add(new Acesso((int)a[0], (String)a[2], (int)a[1]));
226         }
227     }
228 }
229
230
231 }
232
233
234 @SuppressWarnings("unchecked")
235 private void setTables() {
236     tableAce.setItems(modelAcessos);
237     tablePro.setItems(modelPropriedades);
238     TableAceHelper aceH = new TableHelpers.TableAceHelper();
239     TableProHelper proH = new TableHelpers.TableProHelper();

```

```

240     tableAce.getColumns().addAll(aceH.getIdColumn(), aceH.getNivelColumn(), aceH.getNomeColumn());
241     tablePro.getColumns().addAll(proH.getIdColumn(), proH.getRazaoColumn(), proH.getEstadoColumn(),
242     proH.getNivelColumn(), proH.getRamoColumn(),proH.getSafeColumn());
243 }
244 }
245 }
246 }
247

```

Fonte: Elaborada pelo autor.

Classe que é utilizada como controlador de listeners para interface GUI Registro.fxml.

### Classe SplashScreen:

```
1 package com.view.models;
2
3 import javax.swing.JFrame;
4
5 public class SplashScreen extends JFrame{
6
7     private static final long serialVersionUID = 1L;
8
9     public SplashScreen() {
10         super("Aguarde");
11         setSize(256,117);
12         setLocationRelativeTo(null);
13         setDefaultCloseOperation(DO_NOTHING_ON_CLOSE);
14         setResizable(false);
15         JLabel lblNewLabel = new JLabel("Iniciando aplicação. Aguarde");
16         lblNewLabel.setFont(new Font("Tahoma", Font.PLAIN, 16));
17         lblNewLabel.setHorizontalAlignment(SwingConstants.CENTER);
18         getContentPane().add(lblNewLabel, BorderLayout.CENTER);
19         setVisible(true);
20     }
21
22     public void close() {
23         dispose();
24     }
25 }
26
27
28
29
30
```

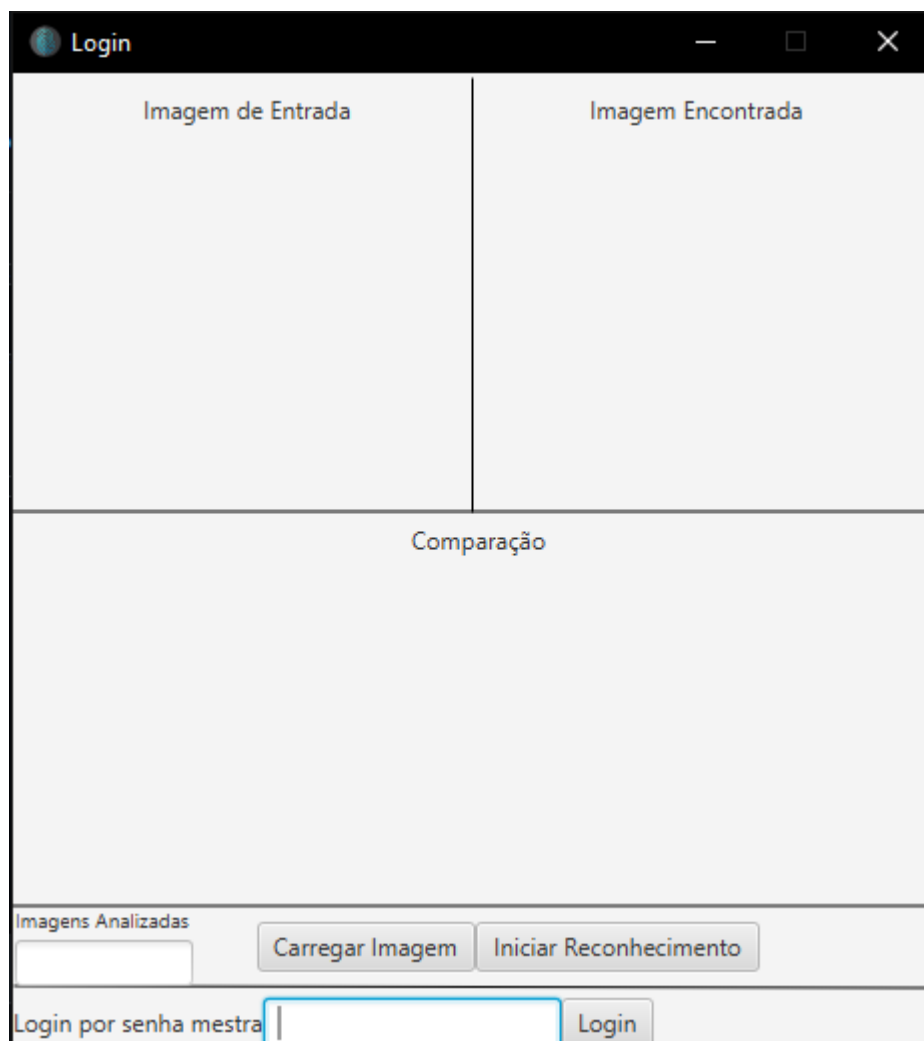
Fonte: Elaborada pelo autor.

Essa classe é responsável pela interface que é exibida ao iniciar a aplicação enquanto está sendo carregado as informações necessárias para o funcionamento.

**7 Apresentação do programa em funcionamento em um computador, apresentando todas as funcionalidades pedidas e extras.**

## TELA DE LOGIN

Figura 16



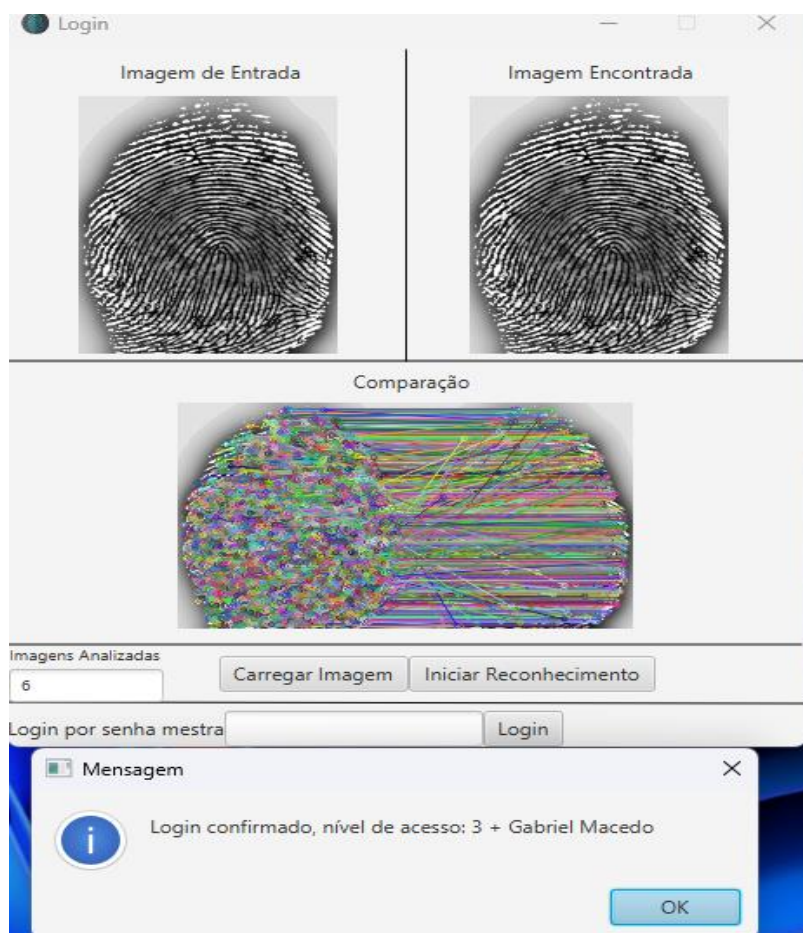
Fonte: Elaborada pelo Autor

Essa seria a tela inicial da aplicação, nessa tela temos algumas possibilidades. Temos a possibilidade de fazer o login com o usuário master, utilizando-se uma senha pré-definida, realizando o acesso as informações de todos os usuários e dados cadastrados. Temos a possibilidade de carregar uma imagem biometria e iniciar o reconhecimento da biometria para definir os níveis de acesso aos dados.

Nessa imagem, temos um exemplo de acesso via biometria cadastrada. É possível visualizar os pontos de comparação da biometria e assim garantindo o acesso ao respectivo nível concedido aos dados armazenados.

## ACESSO AUTORIZADO UTILIZANDO DIGITAL

Figura 17



Fonte: Elaborada pelo Autor

Nessa imagem, temos um exemplo de acesso via biometria cadastrada. É possível visualizar os pontos de comparação da biometria e assim garantindo o acesso ao respectivo nível concedido aos dados armazenados.

## INTERFACE DE PROPRIEDADES REGISTRADAS

Figura 18

Id	Nome/Razão Social	Cidade	Estado	Destino	Status
1	Jales	São José d...	SP	Interno	
2	SLC Agrícola	Florianopolis	SC	Interno	
3	COPERSUCAR	Santos	SP	Interno	
4	LOUIS DREYFUS	Santos	SP	Externo	
7	GUILHERME AUGUSTO	São Paulo	SP	Interno	

Fonte: Elaborada pelo Autor

A imagem, é seguido pelo login efetuado anteriormente. Temos informações do usuário conectado e a disposição dos dados armazenados. Temos informações das empresas cadastradas e seus respectivos endereços comerciais. Sendo possível adicionar, alterar, exibir ou apagar os dados, dependendo do nível de permissão do usuário.

## INTERFACE DE CADASTRO NÍVEL 1

Nos próximos prints, temos as telas onde é possível o cadastramento das informações. Temos diversos campos que devem ser preenchidos com as



informações relevantes dos produtos que serão administrados pelo Ministério do Meio Ambiente.

Figura 19

**Cadastro**

**Unidade**

**Produção Anual**  **Numero de Empregados**

**Destino da produção**  **Nível de Automação**

**Q. Máquinas e Implementos**

**Propriedade**

**Cidade**  **CEP**

**Endereço**

**Pais**  **Estado**

**Valores Fiscais**

**Incentivos fiscais recebidos**  **Taxas federais**

**Impostos municipais pagos**

**Impostos estuais recolhidos**

**Impostos federais pagos**

**Agrotoxico**

Agrotoxico	Liberado
Não há conteúdo na tabela	

Fonte: Elaborada pelo Autor

## INTERFACE DE CADASTRO NÍVEL 2

Figura 20

**Cadastro**

**Unidade**

Unidade:

Produção Anual:  Número de Empregados:

Destino da produção:  Nível de Automação:

Q. Máquinas e Implementos:

**Propriedade**

Cidade:  CEP:

Endereço:

País:  Estado:

**Valores Fiscais**

Incentivos fiscais recebidos:  Taxas federais:

Impostos municipais pagos:

Impostos estaduais recolhidos:

Impostos federais pagos:

**Agrotoxico**

Agrotoxico	Liberado
Não há conteúdo na tabela	

Fonte: Elaborada pelo Autor

## INTERFACE DE CADASTRO NÍVEL 3

Figura 20

**Cadastro**

Unidade: SLC Agrícola

Produção Anual: 13512.0    Numero de Empregados: 34

Destino da produção: Interno    Nível de Automação: 2

Qtd Máquinas e Implementos: 46

Propriedade:

Cidade: Florianópolis    CEP: 35182016

Endereço: R. Belizário Berto da Silveira, 255

País: Brasil    Estado: SC

Valores Fiscais

Incentivos fiscais recebidos: 354561.0    Taxas federais: 15216.0

Impostos municipais pagos: 54681.0

Impostos estaduais recolhidos: 154356.0

Impostos federais pagos: 48634.0

Agrotóxico

Adicionar    Remover

Agrotóxico	Liberado
METOMIL	✓
MANCOZEBE	✓
MALATIONA	✓
GLIFOSATO	✓

Permitir    Banir

Cancelar    Salvar

**Login**

Usuario Atual: Gabriel Macedo    Nível de Acesso: 3

Adicionar    Exibir    Editar    Apagar

Propriedades    Acessos

Id	Nome/Razão Social	Cidade	Estado	Destino	Status
1	Jales	São José d...	SP	Interno	⚠
2	SLC Agrícola	Florianópolis	SC	Interno	✓
3	COPERSUCAR	Santos	SP	Interno	✓
4	LOUIS DREYFUS	Santos	SP	Externo	⚠
7	GUILHERME AUGUSTO	São Paulo	SP	Interno	⚠

Fonte: Elaborada pelo Autor

## Interface Cadastro Agrotóxico

Figura 21

Cadastro

Unidade

LOUIS DREYFUS

Produção Anual

5.0E7

Numero de Empregados

462135

Destino da produção

Externo

Nível de Automação

3

Q. Máquinas e Implementos

45621

Propriedade

Cidade

Santos

CEP

11460004

Endereço

Av. Mário Covas

Pais

Brasil

Estado

SP

Valores Fiscais

Incentivos fiscais recebidos

4456623.0

Taxas federais

51652.0

Impostos municipais pagos

154635.0

Impostos estaduais recolhidos

4562411.0

Impostos federais pagos

415646.0

Cancelar

Salvar

Agrotóxico

Adicionar

Remover

Agrotóxico	Liberado
PARATION	
LINDANO	
ENDOSULFAN	
ATRAZINA	
DIAZINONA	
GLIFOSATO	
ALDRIM	

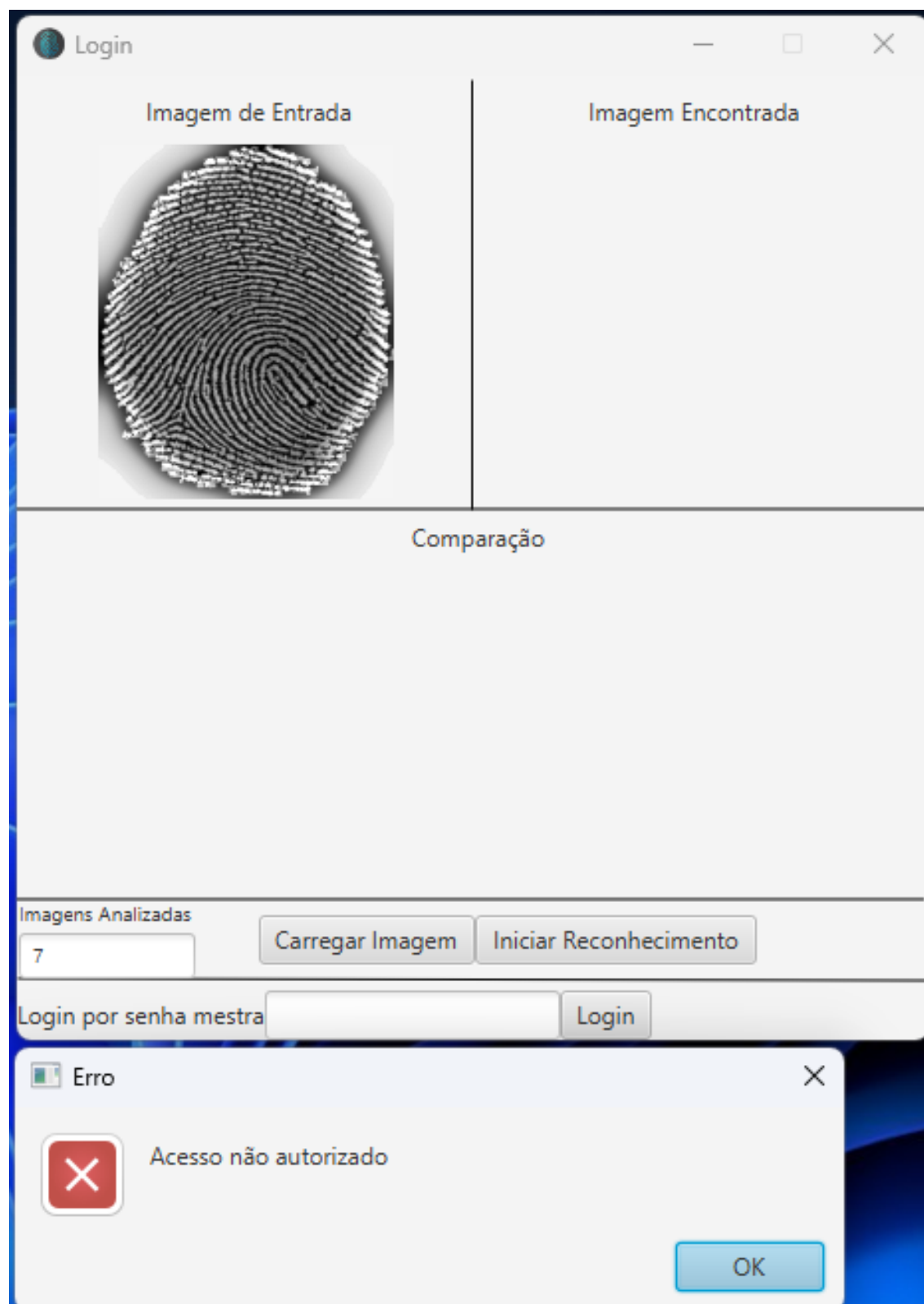
Permitir

Banir

Fonte: Elaborada pelo Autor

## DIGITAL NÃO CADASTRADA

Figura 22

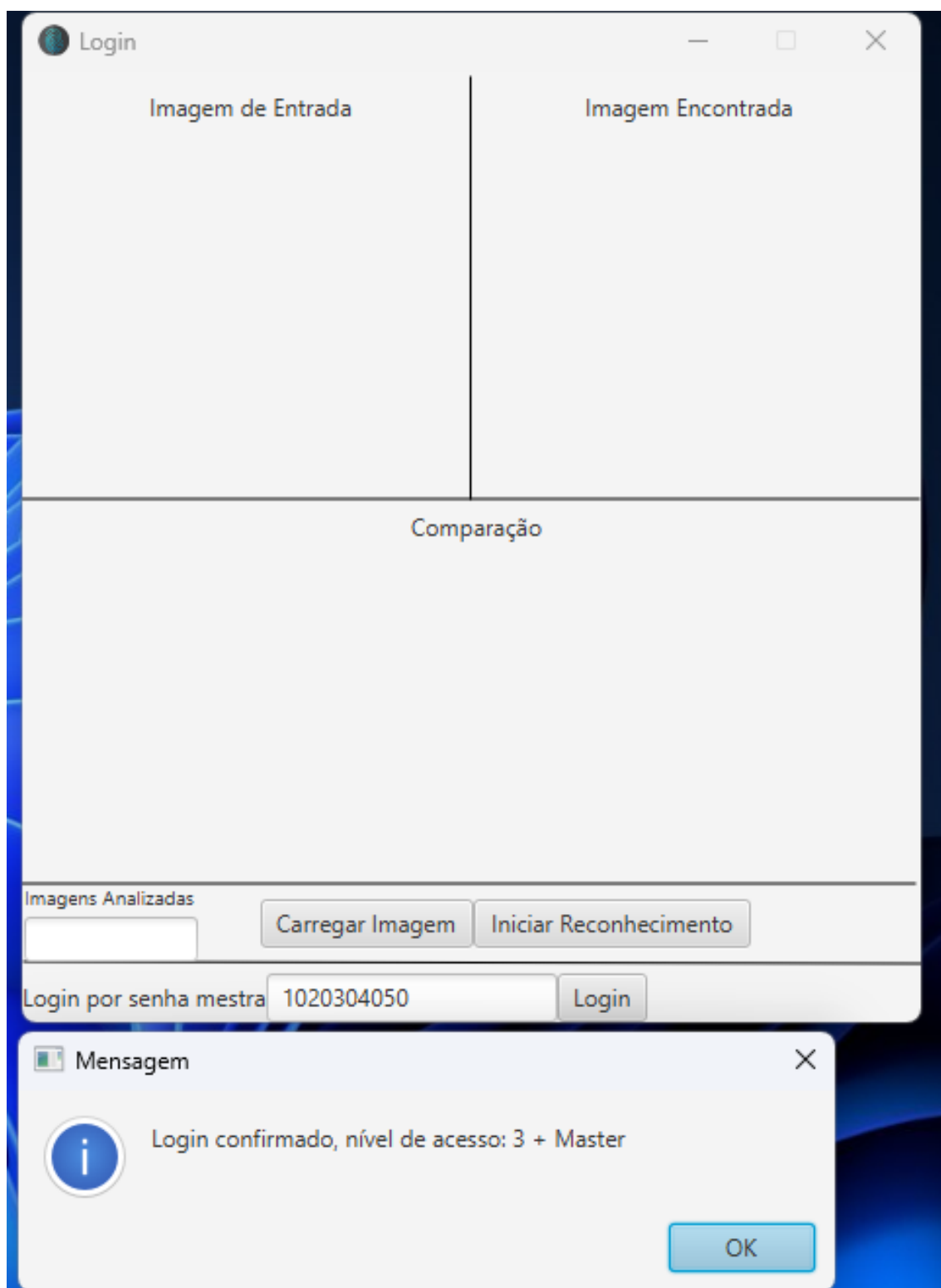


Fonte: Elaborada pelo Autor

Segue um exemplo de erro, no caso de a biometria não estar cadastrada na base de dados.

## LOGIN ATRÁVES DE SENHA

Figura 23

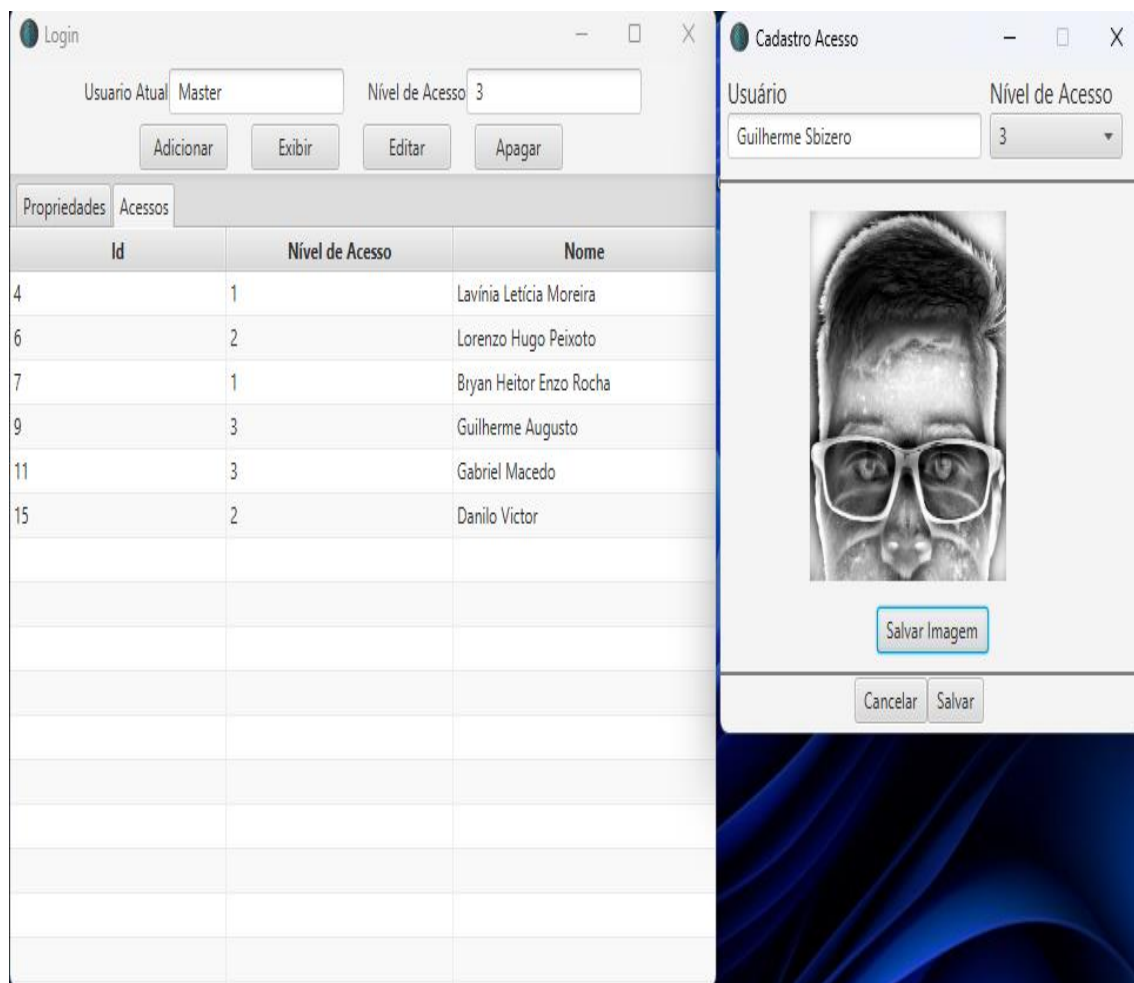


Fonte: Elaborada pelo Autor

O usuário Master, tem todos os níveis de acesso e permissões dentro da aplicação. É um usuário administrador.

## INTERFACE DE CADASTRO DE ACESSO COM CADASTRO FACIAL

Figura 24

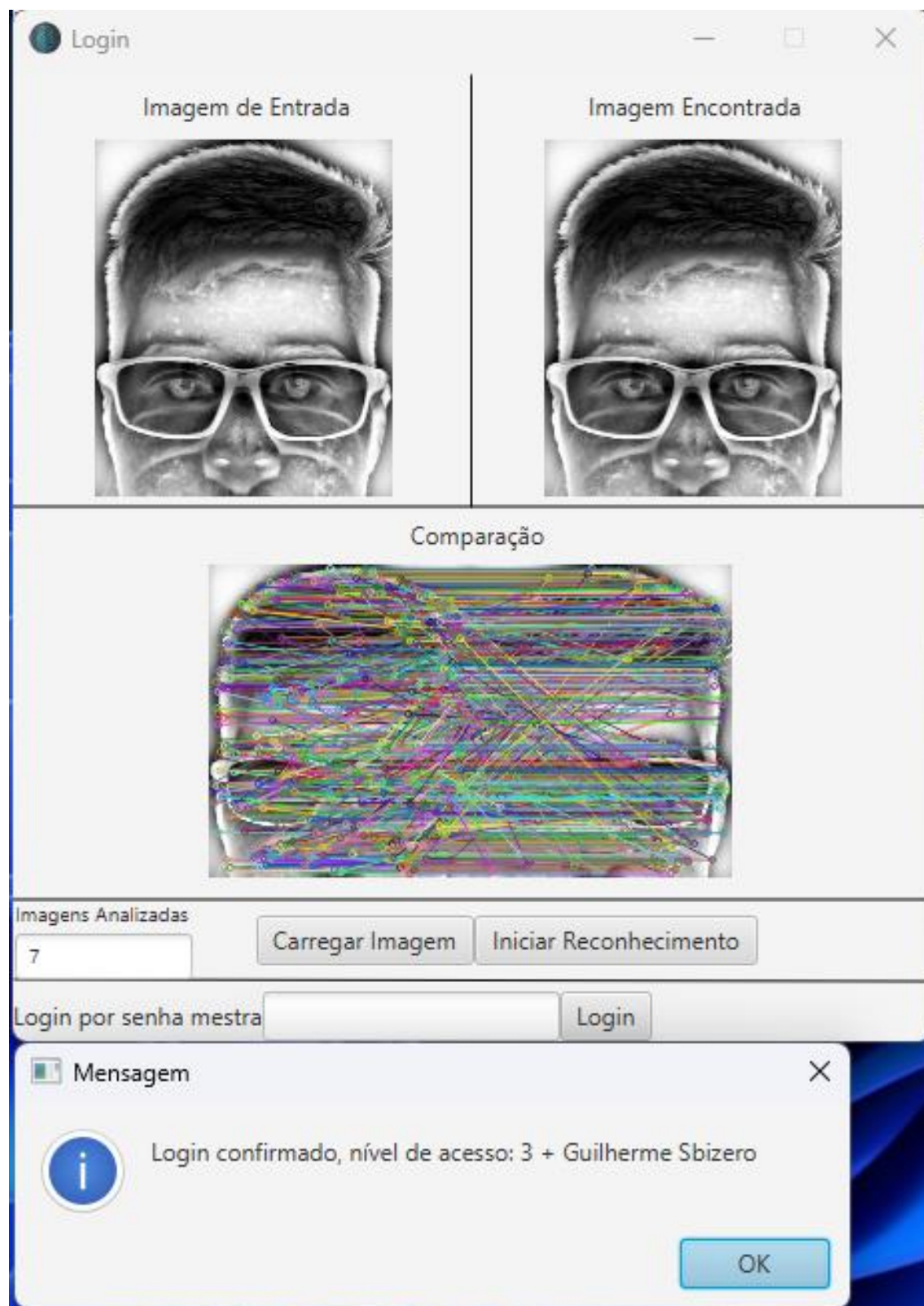


Fonte: Elaborada pelo Autor

Na aba “Acesso”, é onde podemos definir os níveis de acessos e as permissões que o usuário poderá utilizar dentro da aplicação. Seja na visualização dos dados cadastrados ou mesmo na alteração ou exclusão desses dados.

## TELA DE LOGIN COM RECONHECIMENTO FACIAL

Figura 25



Fonte: Elaborada pelo Autor

Temos a possibilidade de realizar o reconhecimento facial e cadastrar a face para futuros acessos. Sendo possível visualizar a comparação entre as faces.



## 8 BIBLIOGRAFIA

AGÊNCIA BRASIL. **Polícia Federal inaugura amanhã moderno sistema de identificação digital**. Disponível em: <http://memoria.ebc.com.br/agenciabrasil/noticia/2004-08-02/policia-federal-inaugura-amanha-moderno-sistema-de-identificacao-digital>>. Acesso em: 20 nov. 2016.

ARAÚJO, Paulo Gabriel Ribacionka Góes de. **Sistema de controle de acesso via smart card com autenticação biométrica da impressão digital**. 2010. 105 f.

Trabalho de Conclusão de Curso (Graduação em Engenharia de Computação) –Centro Universitário de Brasília (UniCEUB), Brasília, 2010. Disponível em: <http://www.repositorio.uniceub.br/bitstream/123456789/3382/3/20516507.pdf>>. Acesso em: 20 nov. 2016.

BIER, Carlos Eduardo. **Bioengine SDK: Identificação Biométrica 1:N**. Disponível em: <https://www.professionaisti.com.br/2011/06/bioengine-sdk-identificacao-biometrica-1n/>>. Acesso em: 20 nov. 2016.

BIOMETRIC SOLUTIONS. **Face Recognition**. Disponível em: <http://www.biometric-solutions.com/face-recognition.html>>. Acesso em: 20 nov. 2016.

BIOMETRIC SOLUTIONS. **Fingerprint Recognition**. Disponível em: <http://www.biometric-solutions.com/fingerprint-recognition.html>>. Acesso em: 20 nov. 2016.

BIOMETRIC SOLUTIONS. **Iris Recognition**. Disponível em: <http://www.biometric-solutions.com/iris-recognition.html>>. Acesso em: 20 nov. 2016.

BIOMETRIC SOLUTIONS. **Keystroke Dynamic**. Disponível em: <http://www.biometric-solutions.com/keystroke-dynamics.html>>. Acesso em: 20 nov. 2016.

BIOMETRIC SOLUTIONS. **Speaker Recognition**. Disponível em: <http://www.biometric-solutions.com/speaker-recognition.html>>. Acesso em: 20 nov. 2016.

BURSZTYN, Victor Soares. **Biométrie: Análise de Assinaturas**. Disponível em: [http://www.gta.ufrj.br/grad/08\\_1/assinat/](http://www.gta.ufrj.br/grad/08_1/assinat/)>. Acesso em: 20 nov. 2016.

FARIA, Alessandro de Oliveira. **Biometria: Processamento de imagens capturadas em leitores de impressão digital**. Disponível em:

<<http://www.linhadecodigo.com.br/artigo/1162/biometria-processamento-de-imagens-capturadas-em-leitores-de-impressao-digital.aspx>>. Acesso em: 20 nov. 2016.

GTA – UFRJ. **Biometria – Assinatura**. Disponível em:

<[http://www.gta.ufrj.br/grad/10\\_1/1a-versao/assinatura/historico.html](http://www.gta.ufrj.br/grad/10_1/1a-versao/assinatura/historico.html)>. Acesso em: 20 nov. 2016.

GTA – UFRJ. **Impressão Digital: Constituição**. Disponível em:

<[http://www.gta.ufrj.br/grad/07\\_2/leonardo/Constituio.html](http://www.gta.ufrj.br/grad/07_2/leonardo/Constituio.html)>. Acesso em: 20 nov. 2016.

NEUROTECHNOLOGY. **Free Fingerprint Verification SDK**. Disponível em:

<<http://www.neurotechnology.com/free-fingerprint-verification-sdk.html>>. Acesso em: 20 nov. 2016.

ROUSE, Margaret. **Biometrics**. Disponível em:

<<http://searchsecurity.techtarget.com/definition/biometrics>>. Acesso em: 20 nov. 2016.

TUTORIALS POINT. **Biometrics: Physiological Modalities**. Disponível em:

<[https://www.tutorialspoint.com/biometrics/physiological\\_modalities.htm](https://www.tutorialspoint.com/biometrics/physiological_modalities.htm)>. Acesso em: 20 nov. 2016.

VIVA O LINUX. **Como funcionam os sistemas de biometria: um estudo geral**.

Disponível em: <<https://www.vivaolinux.com.br/artigo/Como-funcionam-os-sistemas-de-biometria-um-estudo-geral>>. Acesso em: 20 nov. 2016.

WILSON, Tracy V. **How Biometrics Works**. Disponível em:

<<http://science.howstuffworks.com/biometrics.htm>>. Acesso em: 20 nov. 2016.

## 9 FICHAS DE ATIVIDADES PRÁTICA SUPERVISIONADAS

[illegible]

[illegible]