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

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 eletrônicos 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 quilo 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 individuo 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.

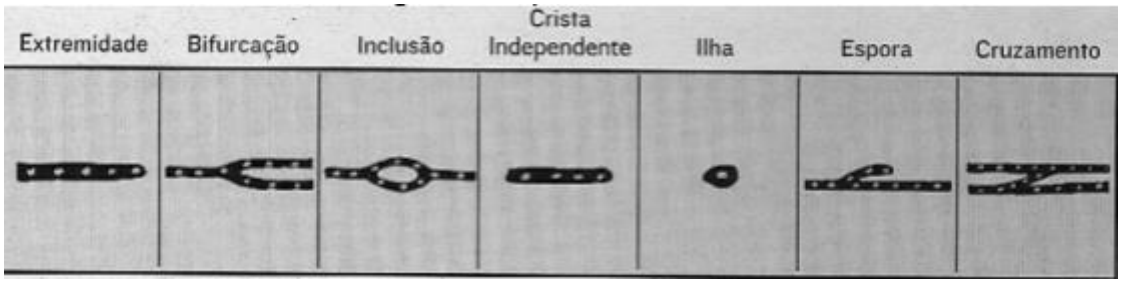
A impressão digital é formada por diversas linhas, criadas a partir das elevações da pele. Ela é única para cada indivíduo, mesmo irmãos gêmeos possuem impressões digitais diferentes. Sendo assim um dos métodos de identificação mais seguros existentes. Tendo sido descoberto e utilizado a mais de mil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

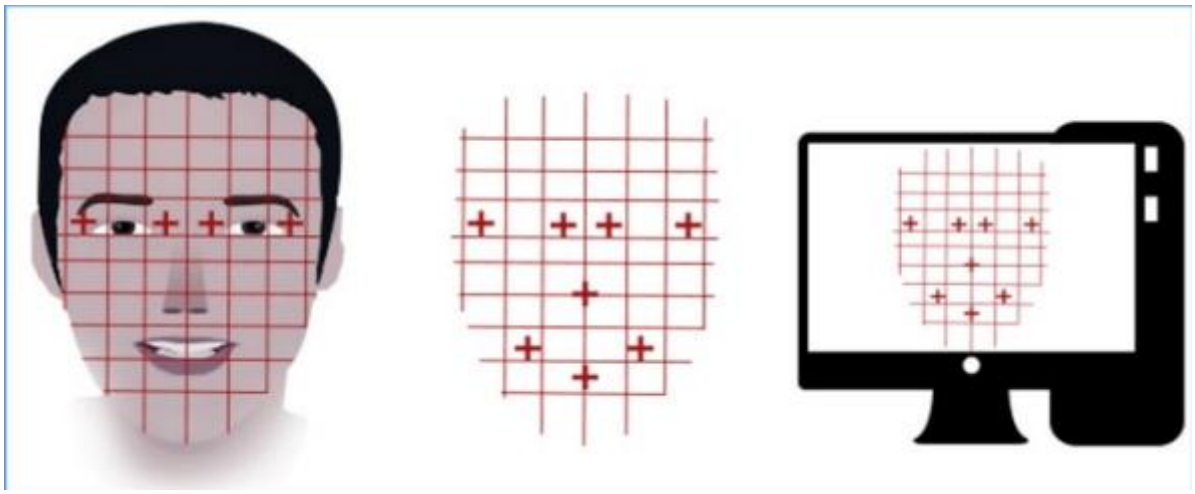
A íris é um músculo dentro do olho que regula o tamanho da pupila, controlando a quantidade de luz que entra no olho. A íris é a parte colorida do olho e cada uma possui sua própria estrutura, o que permite sua utilização na identificação e autenticação de indivíduos.

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 fácil 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 entre reconhecimento facial e os demais métodos é que o reconhecimento facial pode ser capturado à distância, como, por exemplo, por câmera se 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).

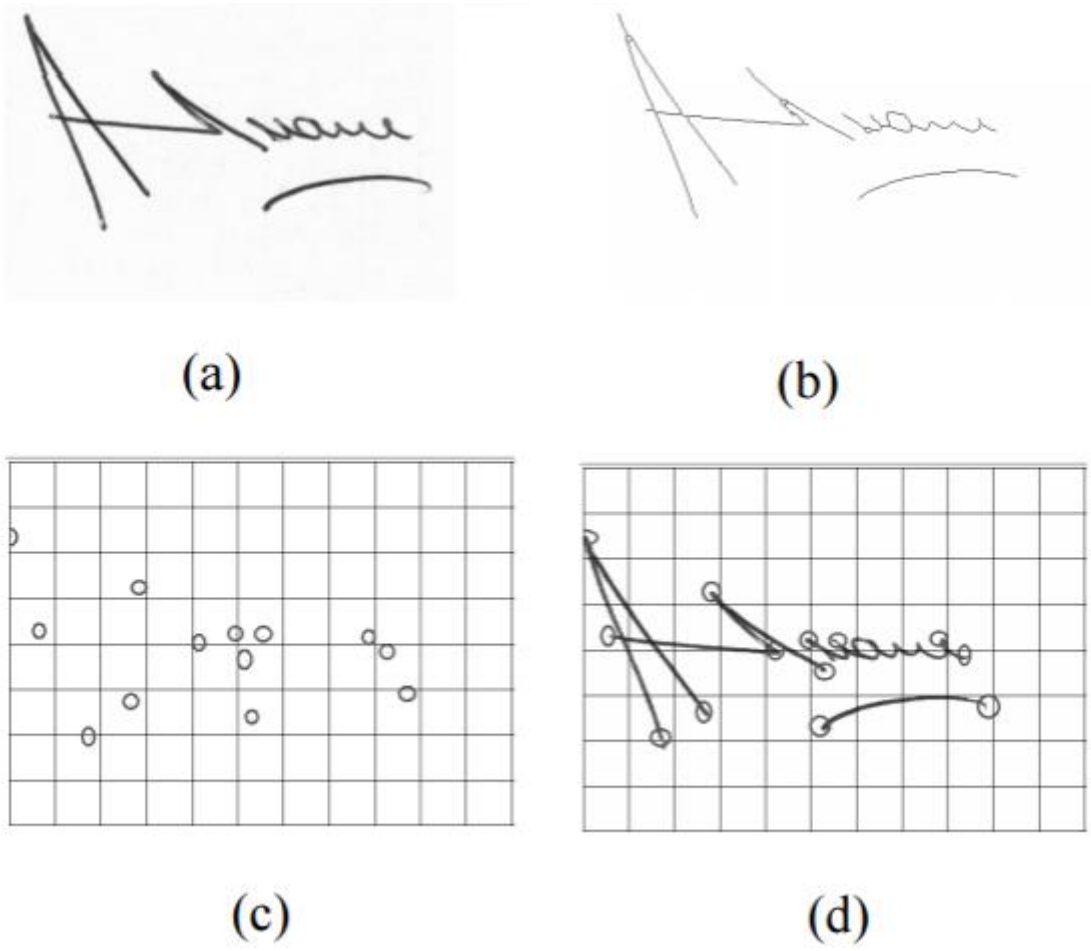
Este padrão foi utilizado na segunda guerra mundial e ficou conhecido como The Fist of the Sender (O punho do remetente). Tendo sido utilizado pelas agências de inteligência militar para distinguir, baseado no ritmo de escrita, se um código morse teria sido enviado por um aliado ou inimigo de guerra.

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 se 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

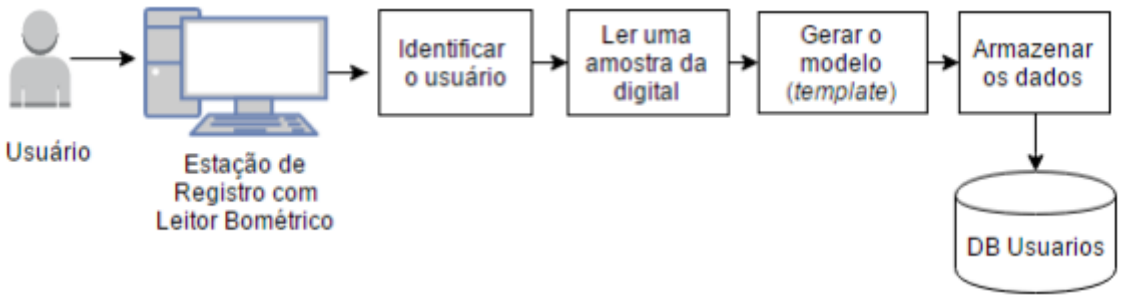
Neste projeto será desenvolvido um sistema de identificação e autenticação biométrica através da impressão digital. Combinando a utilização de nome de usuário com a biometria através da impressão digital. Combinando a utilização do nome de usuário com a biometria, fazendo uma comparação 1:1 na autenticação.

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 está 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 Ministro 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 averificação.

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 (Intergrated Development Environment) foi utilizado o Apache NetBeans IDE versão 12.3 junto com a Java Plataform 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

Notebook Lenovo Ideapad S145	
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 Operancional	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.

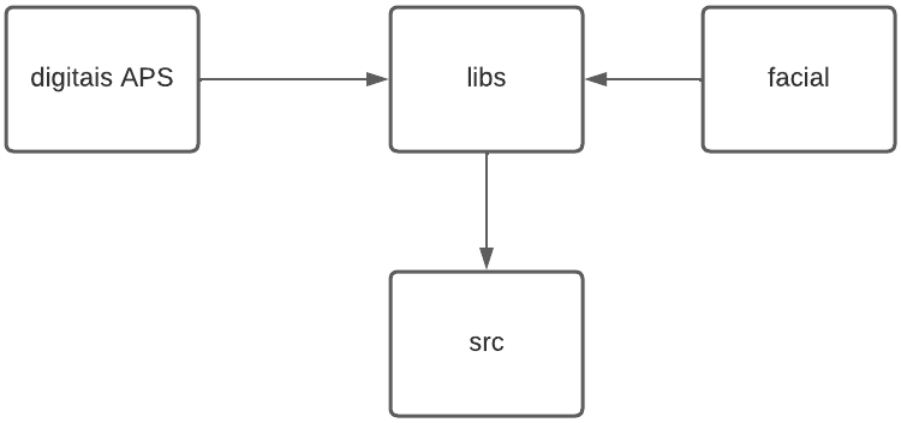
Além dos *templates* será preciso armazenar os dados dos usuários, como seunome, seu nome de usuário e seu nível de acesso. Isto será realizado através dosalvamento do objeto do usuário em um arquivo. Este arquivo será armazenado emformato binário, utilizando a funcionalidade de Serialização disponibilizada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

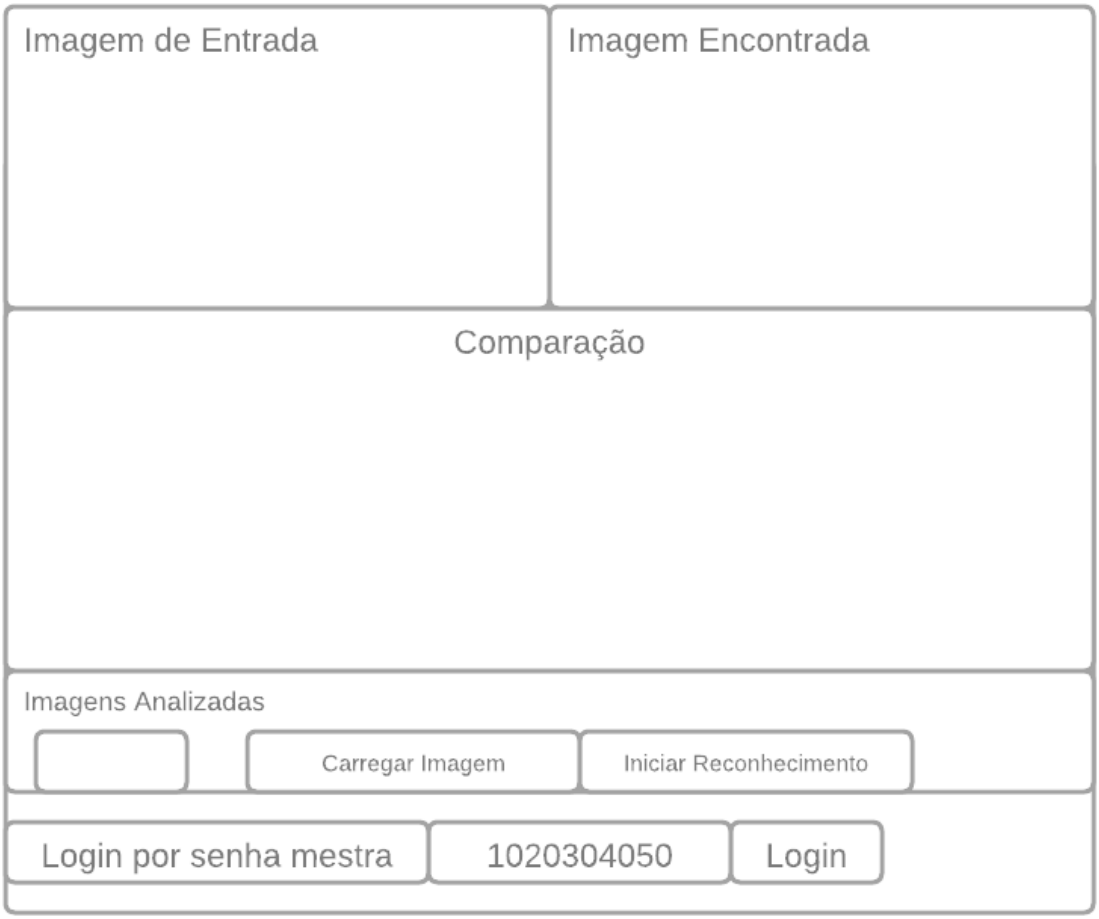
Na segunda parte aparecerá a comparação das imagens, a que foi escolhida e a que foi encontrada.

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



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

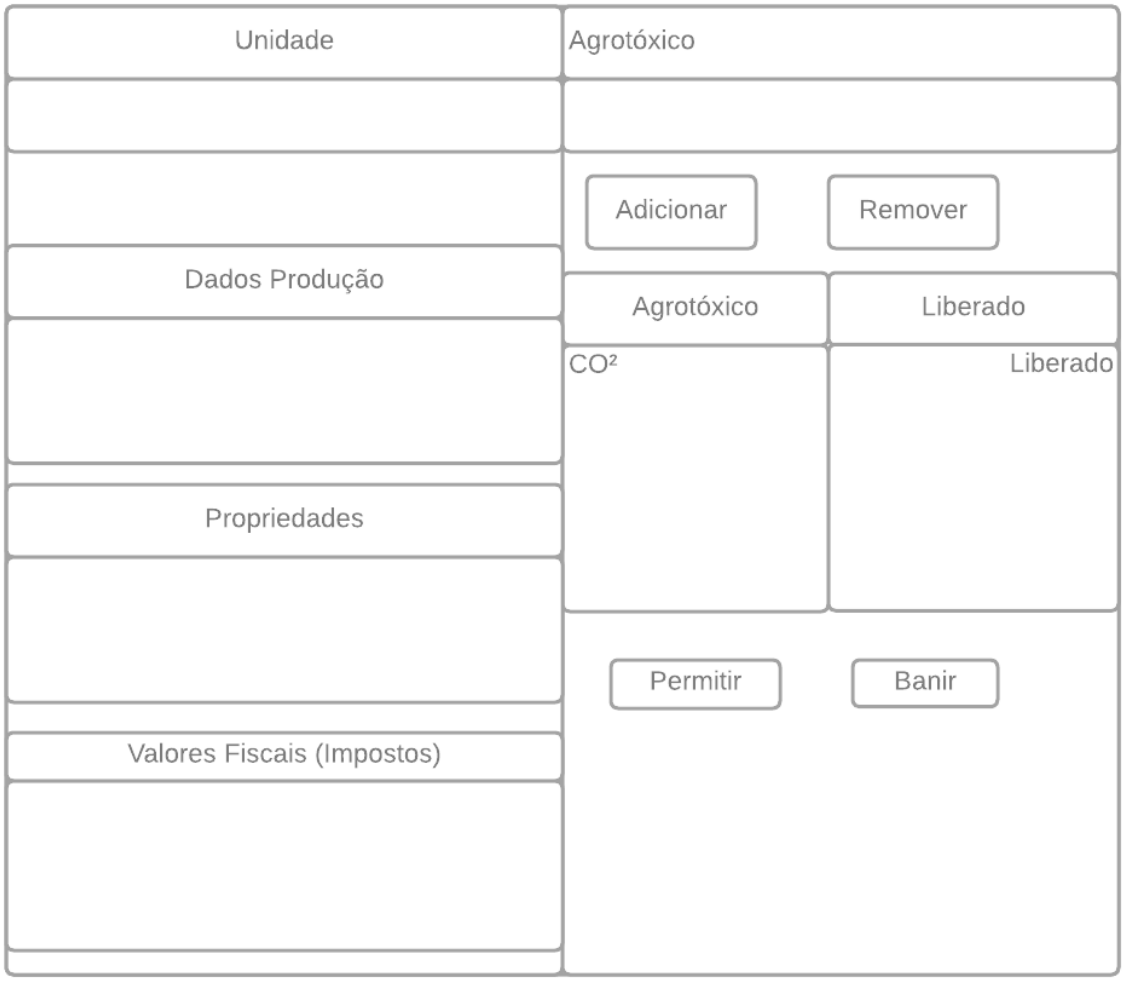
Fonte: Elaborada pelos autores

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



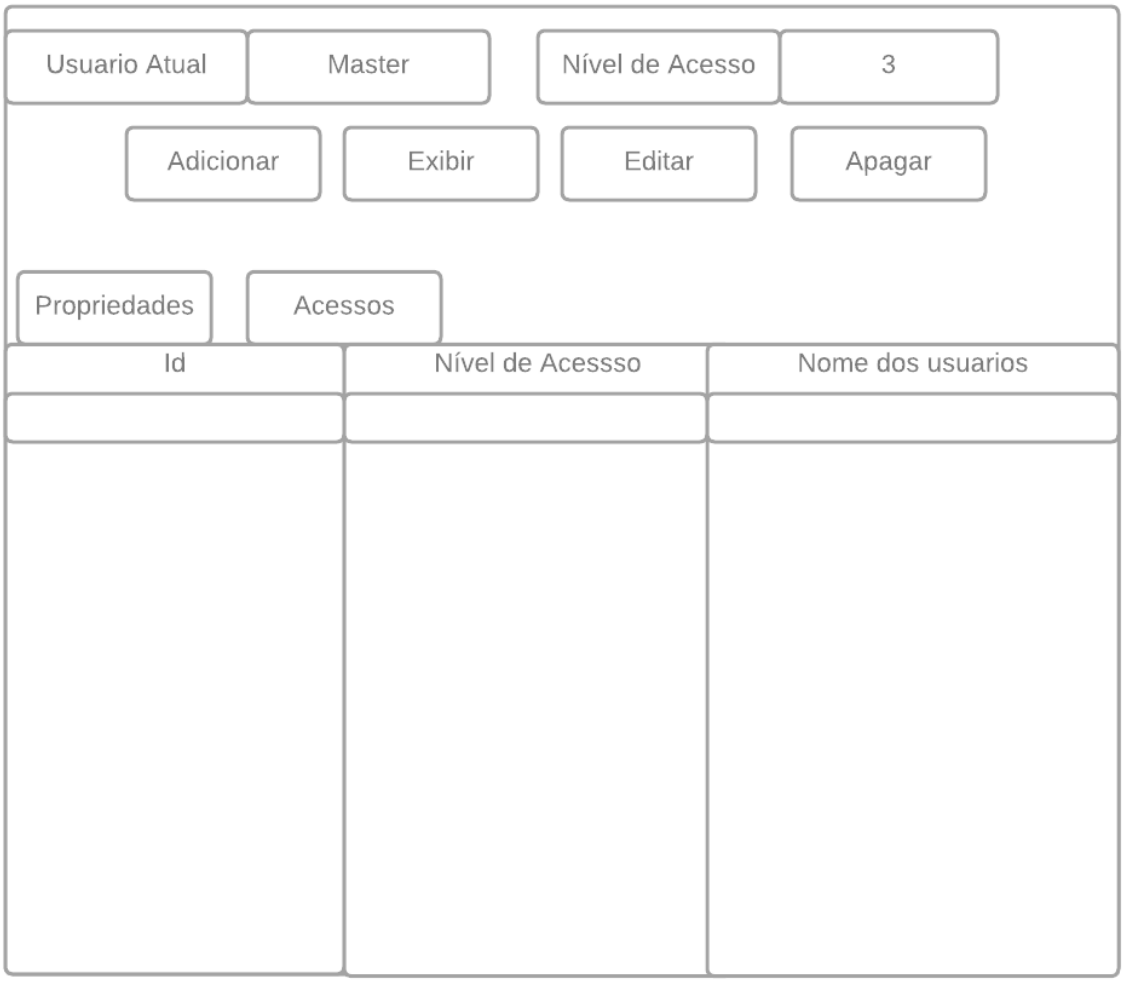
Fonte: Elaborada pelos autores

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 exibe 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



Fonte: Elaborada pelos autores.

6 Relatório com as linhas de código do programa

Classe APS2022_6SemestreApplication (Main):

```java
package com.source;


import com.view.models.SplashScreen;

public class Aps2022_6SemestreApplication {

    private static boolean preloaderFlag = true;
    public static void main(String[] args) {

        new Thread(() -> {
            SplashScreen spl = new SplashScreen();
            while(preloaderFlag) {
                spl.setVisible(true);
            }
            spl.close();
        }).start();

        Application.launch(Aplicacao.class, args); //Inicia a Aplicação
    }

    public static void setPreloaderFlag(boolean flag) {
        preloaderFlag = flag;
    }


}
```

Fonte: Elaborada pelo autor.

A classe Main, é responsável por inicializar a aplicação.

Classe Aplicação:

```java
package com.source;


import java.util.ArrayList;

/**
 * Em conjunto com a classe Aps20212SemestreApplication essas classes são responsaveis por carregar recursos e inicia a aplicação
 * Esta classe contem Stage primaryStage e objetos para operações CRUD com banco de dados EntityManager e EntittyManagerFactory*/

public class Aplicacao extends Application {
    public static List<Image> iconsImg = new ArrayList<Image>();
    public static HashMap<String, FXMLLoader> listFrameRoot = new HashMap<String, FXMLLoader>();
    private static EntityManagerFactory emf = Persistence.createEntityManagerFactory("aps");
    public static EntityManager em = emf.createEntityManager();
    public static Stage primaryStage;

    @Override
    public void init() throws Exception {
        super.init();

    }

    @Override
    public void start(Stage stage){
        Aps2022_6SemestreApplication.setPreloaderFlag(false);
        try {
            loadIcons();
            primaryStage = stage;
            listFrameRoot.put("Login",new FXMLLoader(Thread.currentThread().getContextClassLoader().getResource("com/view/models/Logi
            listFrameRoot.put("Cadastro",new FXMLLoader(Thread.currentThread().getContextClassLoader().getResource("com/view/models/C
            listFrameRoot.put("Registro",new FXMLLoader(Thread.currentThread().getContextClassLoader().getResource("com/view/models/R
            listFrameRoot.put("Acesso",new FXMLLoader(Thread.currentThread().getContextClassLoader().getResource("com/view/models/Ace
            stage.setScene(new Scene(listFrameRoot.get("Login").load(),460,487));
            stage.setTitle("Login");
            stage.setResizable(false);
            stage.getIcons().addAll(iconsImg);
            stage.show();
        } catch (Exception e) {
            e.printStackTrace();
```

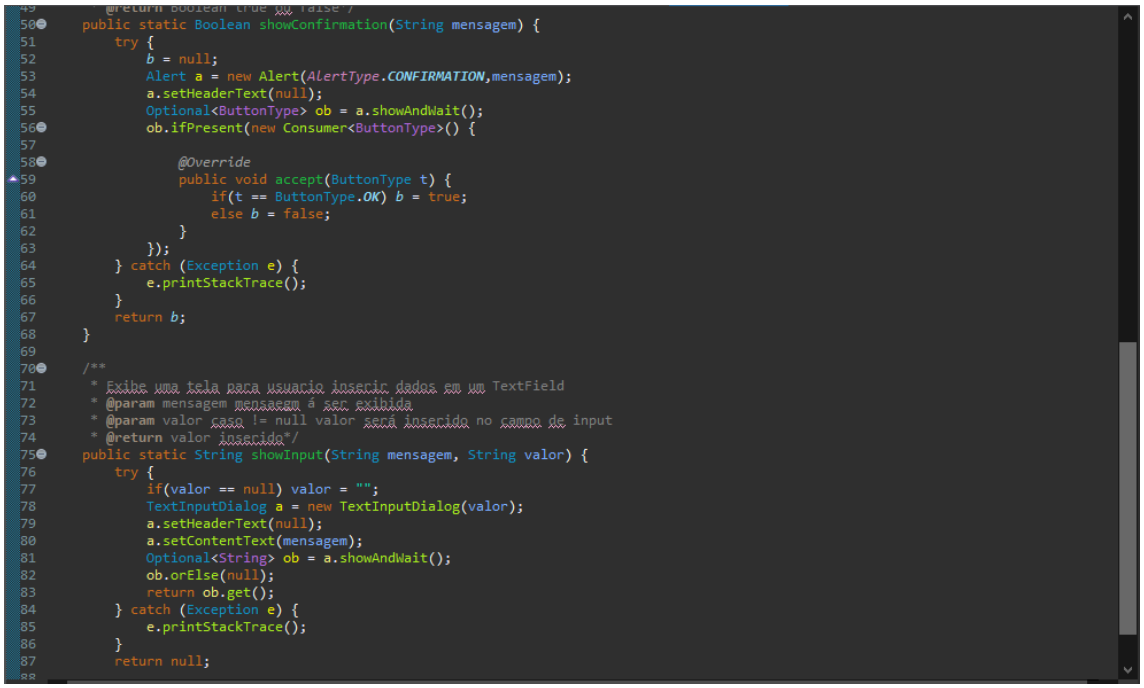
```
54          }
55      }
56
57      public void loadIcons() {
58          Image icon = new Image(getClass().getResource("/Icons/Icone64.png").toString());
59          iconsImg.add(icon);
60      }
61
62      @Override
63      public void stop() throws Exception {
64          super.stop();
65     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

Classe Alerts:

```
1  package com.source;
2
3
4  import java.util.Optional;
11
12 /**
13  * Classe para facilitar a exibição e retorno de informações, todos os metodos são estaticos e podem ser utilizados em qualquer class
14  * desde que o thread responsavel por chamar estes metodos seiam 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 mensaegm 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 mensaem 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
```

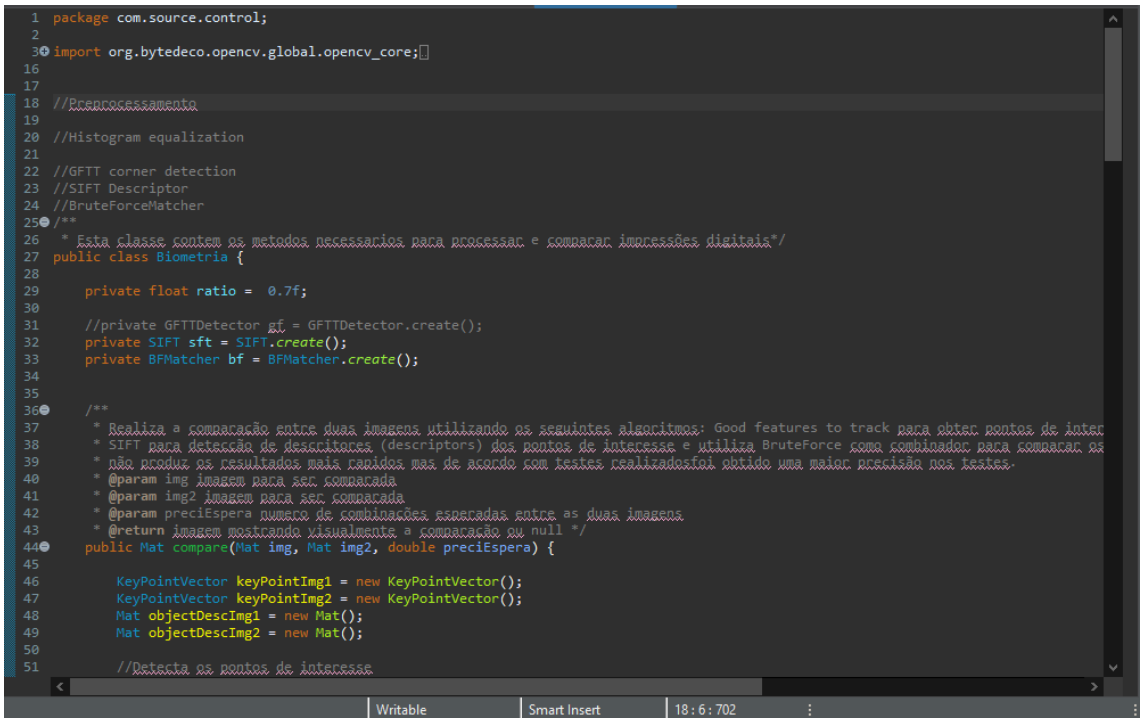


```
    * @return Boolean true ou false */
    public static Boolean showConfirmation(String mensagem) {
        try {
            b = null;
            Alert a = new Alert(AlertType.CONFIRMATION,mensagem);
            a.setHeaderText(null);
            Optional<ButtonType> ob = a.showAndWait();
            ob.ifPresent(new Consumer<ButtonType>() {

                @Override
                public void accept(ButtonType t) {
                    if(t == ButtonType.OK) b = true;
                    else b = false;
                }
            });
        } catch (Exception e) {
            e.printStackTrace();
        }
        return b;
    }

    /**
     * Exibe uma tela para usuario insecir dados em um TextField
     * @param mensagem mensagem a ser exibida
     * @param valor caso != null valor será inserido no campo de input
     * @return valor inserido*/
    public static String showInput(String mensagem, String valor) {
        try {
            if(valor == null) valor = "";
            TextInputDialog a = new TextInputDialog(valor);
            a.setHeaderText(null);
            a.setContentText(mensagem);
            Optional<String> ob = a.showAndWait();
            ob.orElse(null);
            return ob.get();
        } catch (Exception e) {
            e.printStackTrace();
        }
        return null;
```

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:



```
package com.source.control;

import org.bytedeco.opencv.global.opencv_core;


//Preprocessamento

//Histogram equalization

//GFTT corner detection
//SIFT Descriptor
//BruteForceMatcher
/**
 * Esta classe contem os metodos necessarios para processar e comparar impressões digitais*/
public class Biometria {

    private float ratio =  0.7f;

    //private GFTTDetector gf = GFTTDetector.create();
    private SIFT sft = SIFT.create();
    private BFMatcher bf = BFMatcher.create();


    /**
     * Realiza a comparação entre duas imagens utilizando os seguintes algoritmos: Good features to track para obter pontos de inter
     * SIFT para detecção de descritores (descriptors) dos pontos de interesse e utiliza BruteForce como combinador para comparar os
     * não produz os resultados mais rapidos mas de acordo com testes realizadosfoi obtido uma maior precisão nos testes.
     * @param img imagem para ser comparada
     * @param img2 imagem para ser comparada
     * @param preciEspera numero de combinações esperadas entre as duas imagens
     * @return imagem mostrando visualmente a comparação ou null */
    public Mat compare(Mat img, Mat img2, double preciEspera) {

        KeyPointVector keyPointImg1 = new KeyPointVector();
        KeyPointVector keyPointImg2 = new KeyPointVector();
        Mat objectDescImg1 = new Mat();
        Mat objectDescImg2 = new Mat();

        //Detecta os pontos de interesse
```

```
        //Detecta os pontos de interesse
        sft.detect(img, keyPointImg1);
        sft.detect(img2, keyPointImg2);
        //Calcula os descritores de pontos de interesse
        sft.compute(img, keyPointImg1, objectDescImg1);
        sft.compute(img2, keyPointImg2, objectDescImg2);

        DMatchVectorVector knnMatches = new DMatchVectorVector();

        //Analiza as melhores combinações de descritores das duas imagens
        bf.knnMatch(objectDescImg1, objectDescImg2, knnMatches, 2);

        //Realiza um ultimo teste utilizando um valor ratio para filtrar melhores resultados
        DMatchVectorVector listOfGoodMatches = new DMatchVectorVector();
        for (int i = 0; i < knnMatches.size(); i++) {
            if (knnMatches.get(i).size() > 1) {
                DMatch[] matches = knnMatches.get(i).get();
                if (matches[0].distance() < ratio * matches[1].distance()) {
                    listOfGoodMatches.push_back(knnMatches.get(i));
                }
            }
        }
        System.out.println(listOfGoodMatches.size());
        System.out.println(knnMatches.size());

        if(listOfGoodMatches.size() >= preciEspera) {
            Mat imagemCompara  = new Mat();
            opencv_features2d.drawMatchesKnn(img, keyPointImg1, img2, keyPointImg2, listOfGoodMatches,imagemCompara,Scalar.all(-1),S
            listOfGoodMatches.close();
            return imagemCompara;
        }
        listOfGoodMatches.close();
        return null;
    }


    /**
     * Processa a imagem para o padrão utilizado na aplicação
     * @param img imagem para ser processada
```

```
     * @return imagem processada com tamanho de (250,300)*/
    public static Mat processImagem(Mat img) {
        Mat imagem = img.clone();
        if(imagem.type() > 0) {
            opencv_imgproc.cvtColor(imagem, imagem, opencv_imgproc.COLOR_BGR2GRAY);
        }
        //Obtem a as dimensões da impressão digital
        Rect rec = getBigContour(imagem);

        Mat imt = new Mat(imagem.clone(),rec);
        Mat temp2 = new Mat();
        Mat temp3 = new Mat();
        //Equaliza o histograma da imagem
        opencv_imgproc.equalizeHist(imt, imt);

        //Embaça a imagem aplicando um filtro Gaussian, e mistura a imagem original,embaçada e alguns pesos para obter uma imagem nor
        opencv_imgproc.GaussianBlur(imt, temp2, new Size(0,0), 20);
        opencv_core.addWeighted(temp2,  2, imt, -1.1, 1, temp3);


        //Ajusta o tamanho da imagem para um tamanho fixo
        opencv_imgproc.resize(temp3,temp3, new Size(250,300));
        //Equaliza o histograma da imagem
        opencv_imgproc.equalizeHist(temp3.clone(), temp3);
        temp2.close();
        imt.close();
        imagem.close();
        return temp3;
    }
```

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:

```java
package com.source.control;

import java.util.stream.Stream;

/**
 * Classe para operações CRUD generico, esta classe não é Thread safe e possui
 * somente um EntityManager
 */

public class ControllerBd {

    public static EntityManager em = Aplicacao.em;
    private static EntityTransaction trans = em.getTransaction();

    public static void create(Object ob) {
        try {
            checkTrans();
            begin();
            em.persist(ob);
            trans.commit();
        } catch (PersistenceException e) {
            checkTrans();
            e.printStackTrace();
        }
    }

    public static void delete(Object obj) {
        try {
            checkTrans();
            begin();
            em.remove(obj);
            trans.commit();
        } catch (PersistenceException e) {
            checkTrans();
            e.printStackTrace();
        }
    }

    /**
     */
    public static boolean checkPersist(Object obj) {
        try {
            return em.contains(obj);
        } catch (Exception e) {
            e.printStackTrace();
            return false;
        }
    }

    public static Object findById(Class<?> classe, Integer id) throws IllegalArgumentException {
        return em.find(classe, id);
    }

    public static Object findByIdDeatch(Class<?> classe, Integer id) throws IllegalArgumentException {
        Object o = em.find(classe, id);
        em.detach(o);
        return o;
    }

    public static void commit() throws Exception {
        if (trans.isActive()) {
            trans.commit();
        } else {
            throw new Exception();
        }

    }

    @SuppressWarnings("unchecked")
    public static Stream<Acesso> getAcessoAsStream() {
        return em.createQuery("SELECT a FROM ACESSO a").getResultStream();
    }

    public static void begin() throws PersistenceException {
        checkTrans();
        trans.begin();
    }
}
```

```
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 Agrotoxico findAgroByName(String agro) {
110         try {
111             return (Agrotoxico) em.createQuery("select a from AGROTOXICO a where a.agrotoxico = ?1").setParameter(1, agro).getSingle
112         } catch (Exception e) {
113         }
114         return null;
115     }
116 }
117
```

Fonte: Elaborada pelo autor.

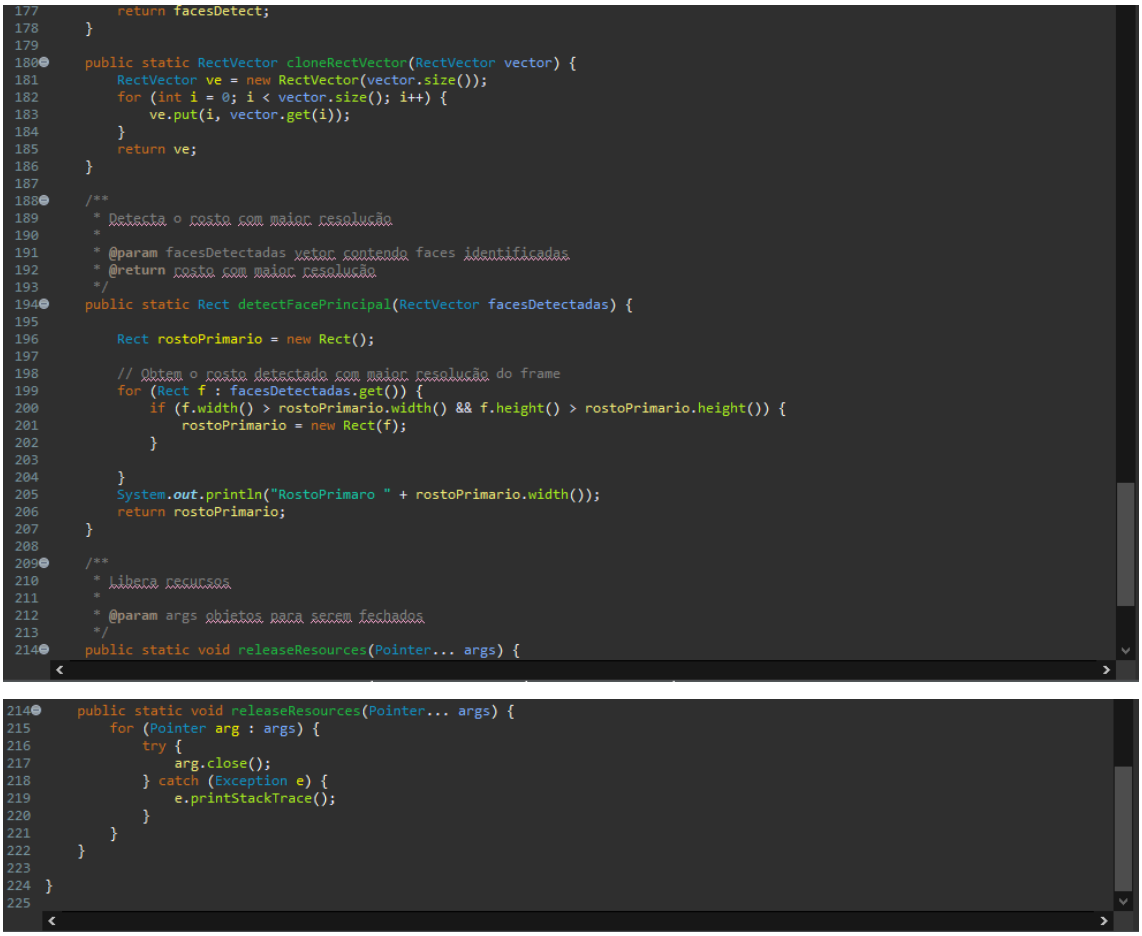
Essa Classe é utilizada para operações CRUD genéricas, esta classe não é Thread safe e possui somente um EntityManager.

Classe Utilitários:

```java
/* Esta classe contem diversos metodos estaticos necessarios para detecção de rostos e outros metodos auxiliares*/
package com.source.control;

import org.bytedeco.javacpp.Pointer;

public class Utilitarios {

    private static JavaFXFrameConverter fxConverter = new JavaFXFrameConverter();
    private static OpenCVFrameConverter.ToMat converter = new OpenCVFrameConverter.ToMat();

    private JavaFXFrameConverter fxConver = new JavaFXFrameConverter();
    private  OpenCVFrameConverter.ToMat conver = new OpenCVFrameConverter.ToMat();
    /**
     * Converte um vetor de pixels Mat para um objeto javafx image
     *
     * @param Mat imagem para ser convertida
     * @return imagem javafx
     */
    public synchronized static Image convertMatToImage(org.bytedeco.opencv.opencv_core.Mat grabbedImage) {
        Frame fram = converter.convert(grabbedImage.clone());
        System.out.println("Fram" + fram.imageWidth);
        Image img = fxConverter.convert(fram);
        fram.close();
        return img;
    }

    public synchronized Image convertMatToImg(org.bytedeco.opencv.opencv_core.Mat grabbedImage) {
        Frame fram = conver.convert(grabbedImage.clone());
        System.out.println("Fram" + fram.imageWidth);
        Image img = fxConver.convert(fram);
        fram.close();
        return img;
    }
    public static void showImage(Image img) {
        Stage stage = new Stage();
        StackPane pa = new StackPane();
        ImageView vi = new ImageView();
        pa.getChildren().add(vi);
        stage.setScene(new Scene(pa, 500, 600));
        vi.setFitHeight(img.getHeight());
        vi.setFitWidth(img.getWidth());
        vi.setImage(img);
        stage.show();
    }

    public static byte[] getImagemAsByteArr(Mat img) {
        // Aloca um array de bytes de acordo com o tamanho da imagem
        byte[] bytes = new byte[((int) (img.total() * img.elemSize()))];
        img.data().get(bytes);
        return bytes;
    }

    /**
     * Reconstroi a imagem a partir de um array de bytes
     *
     * @param rows quantidade de linhas contida na imagem original
     * @param cols quantidade de colunas contida na imagem original
     * @param type tipo da imagem original
     * @param class1  array de bytes
     * @return
     */
    public static Mat createMatByByteArr(int rows, int cols, int type, byte[] bytes) {
        Mat imagemMat = new Mat(rows, cols, type);
        imagemMat.data().put(bytes);
        return imagemMat;
    }
    public static void showImage(Mat img) {
        try {
            if (img.type() == 0) {
                Mat img2 = new Mat();
                opencv_imgproc.cvtColor(img, img2, opencv_imgproc.COLOR_GRAY2BGR);
                showImage(convertMatToImage(img2));
            } else {
                showImage(convertMatToImage(img));
            }
```

```
99          } catch (Exception e) {
100             e.printStackTrace();
101         }
102     }
103
104     /**
105      * */
106     /**
107      * @param cas           CascadeClassfier com classificadores
108      * @param grabbedImage Imagem para ser processada
109      * @return Image retorna uma imagem com rostos contornados por retangulos
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             rect.close();
121         }
122         facesDetect.close();
123         return Utilitarios.convertMatToImage(grabbedImage);
124
125     }
126
127     /**
128      * Detecta rostos em uma imagem grabbedImage detecta qual o contorno com maio
129      * tamanho e retorna uma imagem tipo JavaFx
130      *
131      * @param cas           Cascade Classifier com classificadores
132      * @param grabbedImage imagem para ser processada
133      * @return Image retorna uma imagem com rosto contornado por um retangulo
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
148         img.close();
149         rect.close();
150         facesDetect.close();
151
152         return fxImg;
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();
```



```
        return facesDetect;
    }

    public static RectVector cloneRectVector(RectVector vector) {
        RectVector ve = new RectVector(vector.size());
        for (int i = 0; i < vector.size(); i++) {
            ve.put(i, vector.get(i));
        }
        return ve;
    }

    /**
     * Detecta o rosto com maior resolução
     *
     * @param facesDetectadas vetor contendo faces identificadas
     * @return rosto com maior resolução
     */
    public static Rect detectFacePrincipal(RectVector facesDetectadas) {

        Rect rostoPrimario = new Rect();

        // Obtem o rosto detectado com maior resolução do frame
        for (Rect f : facesDetectadas.get()) {
            if (f.width() > rostoPrimario.width() && f.height() > rostoPrimario.height()) {
                rostoPrimario = new Rect(f);
            }

        }
        System.out.println("RostoPrimaro " + rostoPrimario.width());
        return rostoPrimario;
    }

    /**
     * Libera recursos
     *
     * @param args objetos para serem fechados
     */
    public static void releaseResources(Pointer... args) {
```



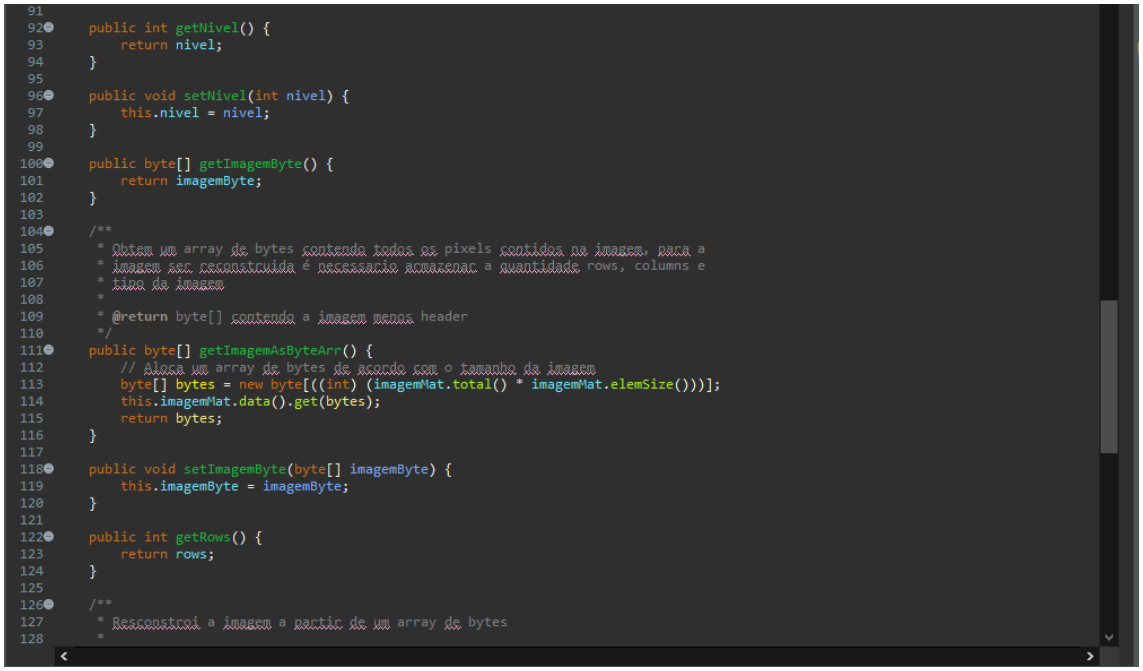
```
    public static void releaseResources(Pointer... args) {
        for (Pointer arg : args) {
            try {
                arg.close();
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
    }

}
```

Fonte: Elaborada pelo autor.

Esta classe contém diversos metódos estáticos necessarios para detecção de rostos e outros metódos auxiliares. Nessa classe, podemos observar metódos responsável por alocar um vetor de bytes e rescontruir a imagem a partir do vetor de bytes, temos metodo 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 Acessos:

```java
package com.source.model;

import java.io.Serializable;

/**
 * Modelo para objecto Acesso que é utilizado para representar dados na TableView CRegistro e este modelo esta mapeado com a tabela
 * do banco de dados ACESSO*/
@Entity(name = "ACESSO")
public class Acesso implements Serializable {
    private static final long serialVersionUID = 1L;

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "ID")
    private Integer id;
    @Column(name = "NOME", length = 100)
    private String nome;
    @Column(name = "NIVEL")
    private int nivel;
    @Lob
    @Basic(fetch = FetchType.LAZY)
    @Column(name = "IMAGEM")
    private byte[] imagemByte;
    @Column(name = "ROWS")
    private int rows;
    @Column(name = "COLUMNS")
    private int col;
    @Column(name = "TYPE")
    private int type;
    @Transient
    private transient Mat imagemMat = new Mat();

    public Acesso() {

    }

    public Acesso(Integer id, String nome, int nivel, byte[] imagemByte, int rows, int col, int type, Mat imagemMat) {
        super();
        this.id = id;
        this.nome = nome;
        this.nivel = nivel;
        this.imagemByte = imagemByte;
        this.rows = rows;
        this.col = col;
        this.type = type;
        if (imagemMat != null) {
            this.imagemMat = imagemMat.clone();
        }

    }


    public Acesso(Integer id, String nome, int nivel) {
        super();
        this.id = id;
        this.nome = nome;
        this.nivel = nivel;
    }

    public void cloneImagem(Mat img) {
        this.imagemMat = img.clone();
    }

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public String getNome() {
        return nome;
    }

    public void setNome(String nome) {
        this.nome = nome;
    }
```



```
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 todos os pixels contidos na imagem, para a
106      * imagem ser reconstruida é necessario armazenar a quantidade rows, columns e
107      * tipo da imagem
108      *
109      * @return byte[] contendo a imagem menos header
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      * Resconstroi a imagem a partic 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 setImagemByteArr(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 reconstroi a imagem, realizando a liberação do acesso.

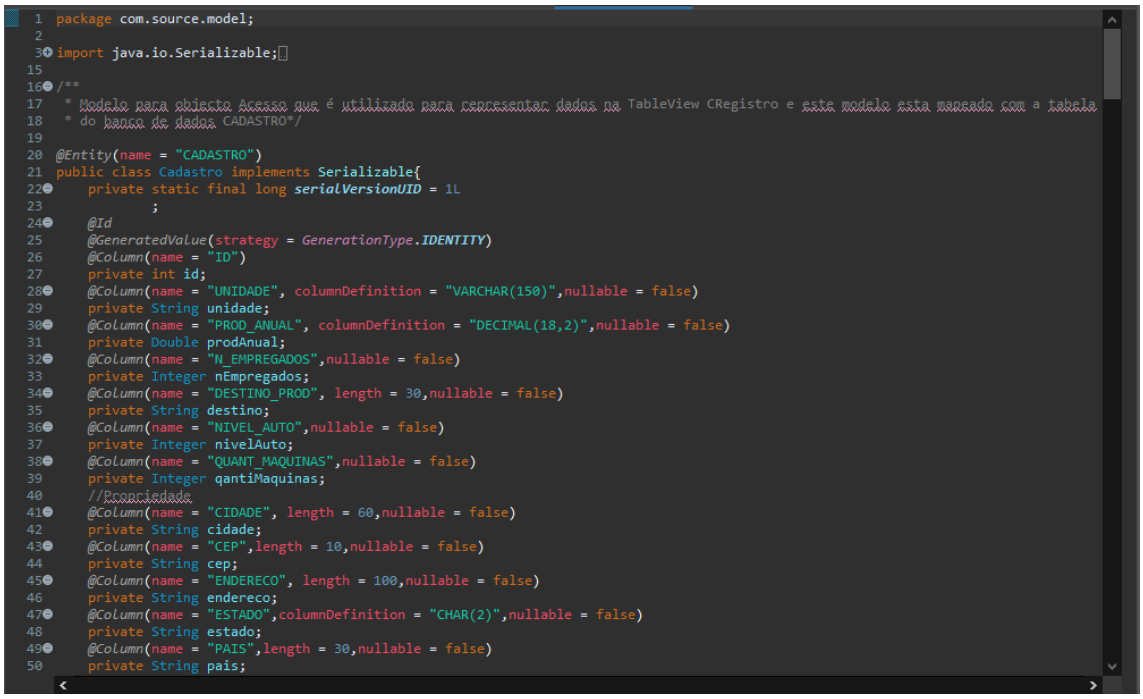
Classe Agrotoxico:

```java
package com.source.model;

import java.io.Serializable;

/**
 * Modelo para objecto Acesso que é utilizado para representar uma lista de agrotoxicos e este modelo esta mapeado com a tabela
 * do banco de dados AGROTOXICO*/

@Entity(name = "AGROTOXICO")
public class Agrotoxico implements Serializable{
    private static final long serialVersionUID = 1L;

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "ID")
    private Integer id;
    @Column(name = "AGROTOXICO",columnDefinition = "VARCHAR(150)",nullable = false)
    private String agrotoxico;
    @Column(name = "PROIBIDO",columnDefinition = "BOOLEAN",nullable = false)
    private Boolean proibido;


    @ManyToMany(mappedBy = "agrotoxicos")
    private List<Cadastro> cadastro = new ArrayList<Cadastro>();
    public Agrotoxico() {

    }



    public Agrotoxico(Integer id, String agrotoxico, Boolean proibido) {
        super();
        this.id = id;
        this.agrotoxico = agrotoxico;
        this.proibido = proibido;

    }


    public Integer getId() {
        return id;
    }

    public void setId(Integer id) {
        this.id = id;
    }

    public String getAgrotoxico() {
        return agrotoxico;
    }

    public void setAgrotoxico(String agrotoxico) {
        this.agrotoxico = agrotoxico;
    }

    public List<Cadastro> getCadastro() {
        return cadastro;
    }

    public void setCadastro(List<Cadastro> cadastro) {
        this.cadastro = cadastro;
    }

    public Boolean getProibido() {
        return proibido;
    }

    public void setProibido(Boolean proibido) {
        this.proibido = proibido;
    }


}
```

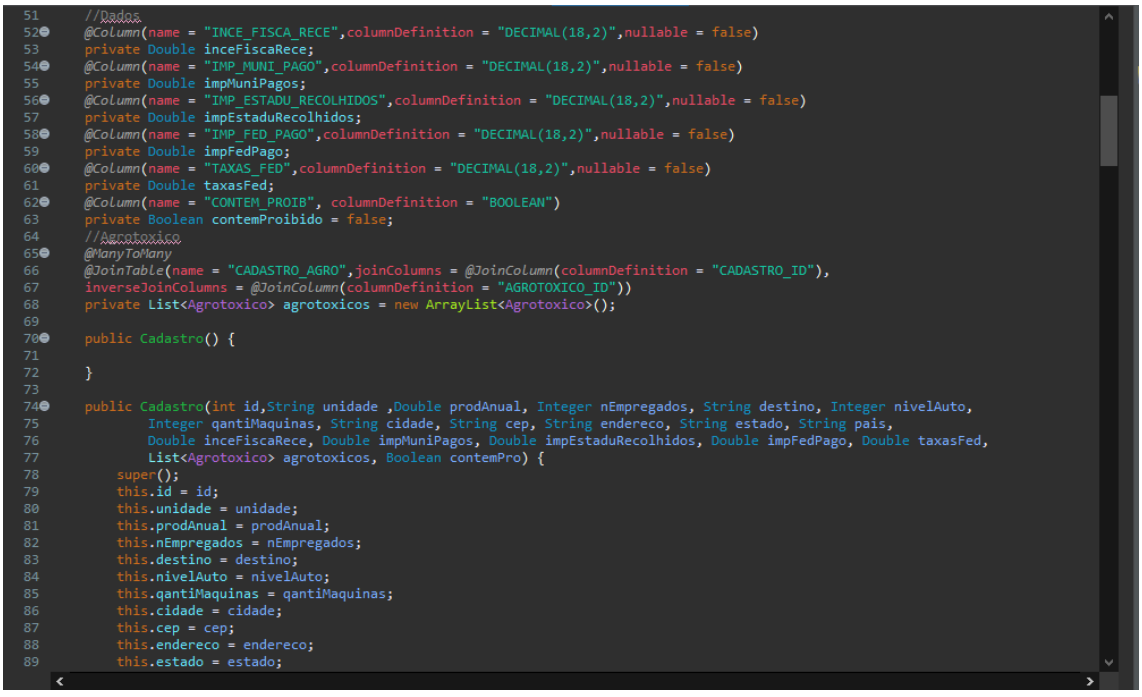
Fonte: Elaborada pelo autor.

Essa classe é um Modelo para o objeto Acesso que é utilizado para representar uma lista de agrotóxicos e este modelo esta 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:



```java
package com.source.model;

import java.io.Serializable;

/**
 * Modelo para objecto Acesso que é utilizado para representar dados na TableView CRegistro e este modelo esta mapeado com a tabela
 * do banco de dados CADASTRO*/

@Entity(name = "CADASTRO")
public class Cadastro implements Serializable{
    private static final long serialVersionUID = 1L
            ;
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "ID")
    private int id;
    @Column(name = "UNIDADE", columnDefinition = "VARCHAR(150)",nullable = false)
    private String unidade;
    @Column(name = "PROD_ANUAL", columnDefinition = "DECIMAL(18,2)",nullable = false)
    private Double prodAnual;
    @Column(name = "N_EMPREGADOS",nullable = false)
    private Integer nEmpregados;
    @Column(name = "DESTINO_PROD", length = 30,nullable = false)
    private String destino;
    @Column(name = "NIVEL_AUTO",nullable = false)
    private Integer nivelAuto;
    @Column(name = "QUANT_MAQUINAS",nullable = false)
    private Integer qantiMaquinas;
    //Propriedade
    @Column(name = "CIDADE", length = 60,nullable = false)
    private String cidade;
    @Column(name = "CEP",length = 10,nullable = false)
    private String cep;
    @Column(name = "ENDERECO", length = 100,nullable = false)
    private String endereco;
    @Column(name = "ESTADO",columnDefinition = "CHAR(2)",nullable = false)
    private String estado;
    @Column(name = "PAIS",length = 30,nullable = false)
    private String pais;
```



```java
    //Dados
    @Column(name = "INCE_FISCA_RECE",columnDefinition = "DECIMAL(18,2)",nullable = false)
    private Double inceFiscaRece;
    @Column(name = "IMP_MUNI_PAGO",columnDefinition = "DECIMAL(18,2)",nullable = false)
    private Double impMuniPagos;
    @Column(name = "IMP_ESTADU_RECOLHIDOS",columnDefinition = "DECIMAL(18,2)",nullable = false)
    private Double impEstaduRecolhidos;
    @Column(name = "IMP_FED_PAGO",columnDefinition = "DECIMAL(18,2)",nullable = false)
    private Double impFedPago;
    @Column(name = "TAXAS_FED",columnDefinition = "DECIMAL(18,2)",nullable = false)
    private Double taxasFed;
    @Column(name = "CONTEM_PROIB", columnDefinition = "BOOLEAN")
    private Boolean contemProibido = false;
    //Agrotoxico
    @ManyToMany
    @JoinTable(name = "CADASTRO_AGRO",joinColumns = @JoinColumn(columnDefinition = "CADASTRO_ID"),
    inverseJoinColumns = @JoinColumn(columnDefinition = "AGROTOXICO_ID"))
    private List<Agrotoxico> agrotoxicos = new ArrayList<Agrotoxico>();

    public Cadastro() {

    }

    public Cadastro(int id,String unidade ,Double prodAnual, Integer nEmpregados, String destino, Integer nivelAuto,
            Integer qantiMaquinas, String cidade, String cep, String endereco, String estado, String pais,
            Double inceFiscaRece, Double impMuniPagos, Double impEstaduRecolhidos, Double impFedPago, Double taxasFed,
            List<Agrotoxico> agrotoxicos, Boolean contemPro) {
        super();
        this.id = id;
        this.unidade = unidade;
        this.prodAnual = prodAnual;
        this.nEmpregados = nEmpregados;
        this.destino = destino;
        this.nivelAuto = nivelAuto;
        this.qantiMaquinas = qantiMaquinas;
        this.cidade = cidade;
        this.cep = cep;
        this.endereco = endereco;
        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.taxasFed = taxasFed;
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 
```

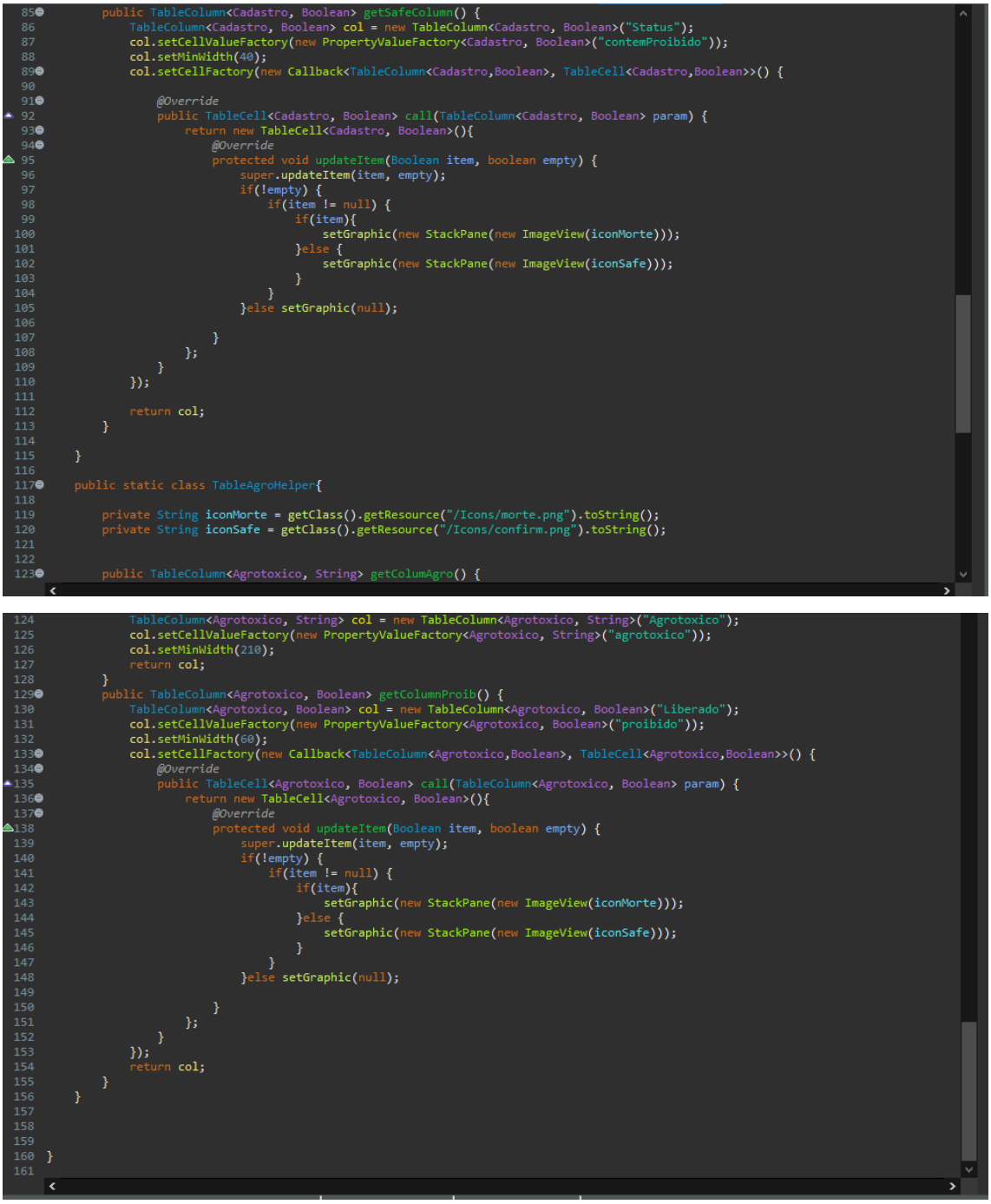
```
    public List<Agrotoxico> getAgrotoxicos() {
        return agrotoxicos;
    }

    public void setAgrotoxicos(List<Agrotoxico> agrotoxicos) {
        this.agrotoxicos = agrotoxicos;
    }

    public void addAgrotoxico(Agrotoxico agro) {
        agrotoxicos.add(agro);
        agro.getCadastro().add(this);
    }
    public void removeAgrotoxico(Agrotoxico agro) {
        agrotoxicos.remove(agro);
        agro.getCadastro().remove(this);
    }

    public void checkContemProibido() {
        boolean bol = false;
        for(Agrotoxico agro : agrotoxicos) {
            if(agro.getProibido()) bol = true;
        }
        contemProibido = bol;
    }
    public Boolean getContemProibido() {
        return contemProibido;
    }

    public void setContemProibido(Boolean contemProibido) {
        this.contemProibido = contemProibido;
    }



}

```

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:

```java
package com.source.model;

import javafx.scene.control.Control;

/**
 * Classes auxiliar na criação das tabelas CRegistro.tablePro e CRegistro.tableAce
 */
public class TableHelpers {

    public static class TableAceHelper {

        public TableColumn<Acesso, Integer> getIdColumn() {
            TableColumn<Acesso, Integer> col = new TableColumn<Acesso, Integer>("Id");
            col.setCellValueFactory(new PropertyValueFactory<Acesso, Integer>("id"));
            col.setMinWidth(60);
            return col;
        }

        public TableColumn<Acesso, String> getNomeColumn() {
            TableColumn<Acesso, String> col = new TableColumn<Acesso, String>("Nome");
            col.setCellValueFactory(new PropertyValueFactory<Acesso, String>("nome"));
            col.setMinWidth(100);
            col.setPrefWidth(Control.USE_COMPUTED_SIZE);
            col.setPrefWidth(20000);
            return col;
        }

        public TableColumn<Acesso, String> getNivelColumn() {
            TableColumn<Acesso, String> col = new TableColumn<Acesso, String>("Nível de Acesso");
            col.setCellValueFactory(new PropertyValueFactory<Acesso, String>("nivel"));
            col.setMinWidth(70);
            return col;
        }

    }

    public static class TableProHelper {

        private String iconMorte = getClass().getResource("/Icons/morte.png").toString();
        private String iconSafe = getClass().getResource("/Icons/confirm.png").toString();

        public TableColumn<Cadastro, Integer> getIdColumn() {
            TableColumn<Cadastro, Integer> col = new TableColumn<Cadastro, Integer>("Id");
            col.setCellValueFactory(new PropertyValueFactory<Cadastro, Integer>("id"));
            col.setMinWidth(60);
            return col;
        }

        public TableColumn<Cadastro, String> getRazaoColumn() {
            TableColumn<Cadastro, String> col = new TableColumn<Cadastro, String>("Nome/Razão Social");
            col.setCellValueFactory(new PropertyValueFactory<Cadastro, String>("unidade"));
            col.setMinWidth(100);
            col.setPrefWidth(Control.USE_COMPUTED_SIZE);
            col.setMaxWidth(20000);

            return col;
        }

        public TableColumn<Cadastro, String> getEstadoColumn() {
            TableColumn<Cadastro, String> col = new TableColumn<Cadastro, String>("Cidade");
            col.setCellValueFactory(new PropertyValueFactory<Cadastro, String>("cidade"));
            col.setMinWidth(60);
            return col;
        }

        public TableColumn<Cadastro, String> getNivelColumn() {
            TableColumn<Cadastro, String> col = new TableColumn<Cadastro, String>("Estado");
            col.setCellValueFactory(new PropertyValueFactory<Cadastro, String>("estado"));
            col.setMinWidth(50);
            return col;
        }

        public TableColumn<Cadastro, String> getRamoColumn() {
            TableColumn<Cadastro, String> col = new TableColumn<Cadastro, String>("Destino");
            col.setCellValueFactory(new PropertyValueFactory<Cadastro, String>("destino"));
            col.setMinWidth(150);
            return col;
        }
```



```
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                  @Override
92                  public TableCell<Cadastro, Boolean> call(TableColumn<Cadastro, Boolean> param) {
93                      return new TableCell<Cadastro, Boolean>(){
94                          @Override
95                          protected void updateItem(Boolean item, boolean empty) {
96                              super.updateItem(item, empty);
97                              if(!empty) {
98                                  if(item != null) {
99                                      if(item){
100                                         setGraphic(new StackPane(new ImageView(iconMorte)));
101                                     }else {
102                                         setGraphic(new StackPane(new ImageView(iconSafe)));
103                                     }
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> getColumAgro() {
```

```
124             TableColumn<Agrotoxico, String> col = new TableColumn<Agrotoxico, String>("Agrotoxico");
125             col.setCellValueFactory(new PropertyValueFactory<Agrotoxico, String>("agrotoxico"));
126             col.setMinWidth(210);
127             return col;
128         }
129         public TableColumn<Agrotoxico, Boolean> getColumnProib() {
130             TableColumn<Agrotoxico, Boolean> col = new TableColumn<Agrotoxico, Boolean>("Liberado");
131             col.setCellValueFactory(new PropertyValueFactory<Agrotoxico, Boolean>("proibido"));
132             col.setMinWidth(60);
133             col.setCellFactory(new Callback<TableColumn<Agrotoxico,Boolean>, TableCell<Agrotoxico,Boolean>>() {
134                 @Override
135                 public TableCell<Agrotoxico, Boolean> call(TableColumn<Agrotoxico, Boolean> param) {
136                     return new TableCell<Agrotoxico, Boolean>(){
137                         @Override
138                         protected void updateItem(Boolean item, boolean empty) {
139                             super.updateItem(item, empty);
140                             if(!empty) {
141                                 if(item != null) {
142                                     if(item){
143                                         setGraphic(new StackPane(new ImageView(iconMorte)));
144                                     }else {
145                                         setGraphic(new StackPane(new ImageView(iconSafe)));
146                                     }
147                                 }
148                             }else setGraphic(null);
149
150                         }
151                     };
152                 }
153             });
154             return col;
155         }
156     }
157
158
159
160 }
161
```

Fonte: Elaborada pelo autor.

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

Classe CAcesso:

```java
package com.view.controllers;

import java.net.URL;

/**
 * Classe que é utilizada como controlador de listeners para interface GUI
 * Acesso.fxml
 */
public class CAcesso implements Initializable{

    @FXML
    private AnchorPane parentPane;
    @FXML
    private TextField txtUsuario;
    @FXML
    private ComboBox<String> comboNivel;
    @FXML
    private ImageView img;
    @FXML
    private Button btnCancelar;

    @FXML
    private Button btnSalvar;
    @FXML
    private Button btnSalvarImg;

    private Mat imagem;

    private Acesso acesso;

    private Border bordaVerm = new Border(new BorderStroke(Color.RED, BorderStrokeStyle.SOLID, new CornerRadii(3),
            new BorderWidths(2), new Insets(-2)));
    private Border bordaDef;

    @Override
    public void initialize(URL location, ResourceBundle resources) {
        bordaDef = txtUsuario.getBorder();
        txtUsuario.textProperty().addListener(new StringList(txtUsuario));
        comboNivel.setItems(FXCollections.observableArrayList(new String[] {"1","2","3"}));
```

```java
    public void construtor(Acesso acesso) {
        this.acesso = acesso;
        if(this.acesso != null) {
            this.acesso = (Acesso) ControllerBd.findById(Acesso.class, acesso.getId());
            setValues();
        }
    }

    @FXML
    private void actBtnCancelar() {
        resetTxts();
        ((Stage) parentPane.getScene().getWindow()).close();
    }

    @FXML
    private void actBtnSalvar() {
        try {
            boolean flag = true;

            if (acesso == null) {
                acesso = new Acesso();
                flag = false;
            }

            ControllerBd.begin();

            if (txtUsuario.getText() == null && txtUsuario.getText().isEmpty() && txtUsuario.getText().isBlank()) {
                Alerts.showError("Campo Nome não pode estar vazio");
                throw new Exception();
            }

            acesso.setNome(txtUsuario.getText());
            acesso.setNivel(Integer.parseInt(comboNivel.getSelectionModel().getSelectedItem()));
            if(img.getImage() == null) {
                Alerts.showError("Campo Imagem não pode estar vazio");
                throw new Exception();
            }
            if(imagem != null) {
                acesso.setImagemMat(imagem);
```

```
                acesso.setCol(imagem.cols());
                acesso.setRows(imagem.rows());
                acesso.setType(imagem.type());
                acesso.setImagemByte(acesso.getImagemAsByteArr());
            }

            System.out.println(acesso.getNivel());
            if(!flag) ControllerBd.em.persist(acesso);
            ControllerBd.commit();

            CRegistro.refreshTableAce();
            resetTxts();
            ((Stage) parentPane.getScene().getWindow()).close();

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

        } catch (Exception e) {
            Alerts.showError("Falha ao cadastrar Acesso");
            e.printStackTrace();
        }

    }

    @FXML
    private void actBtnSalvarImg() {
        try {
            imagem = opencv_imgcodecs.imread(new FileChooser().showOpenDialog(null).getAbsolutePath(),
                    opencv_imgcodecs.IMREAD_GRAYSCALE);
            Mat a = Biometria.processImagem(imagem);
            imagem.close();
            imagem = a.clone();
            Mat temp = new Mat();
            opencv_imgproc.cvtColor(a, temp, opencv_imgproc.COLOR_GRAY2BGR);
            img.setImage(Utilitarios.convertMatToImage(temp));
            temp.close();
            a.close();
        } catch (Exception e) {
            Alerts.showError("Falha ao abrir imagem");
            if (imagem != null || !imagem.empty())
                imagem.close();
            imagem = null;
            e.printStackTrace();
        }
    }
    private void setValues() {
        txtUsuario.setText(acesso.getNome());
        comboNivel.getSelectionModel().select(Integer.toString(acesso.getNivel()));
        acesso.setImagemByteArr(acesso.getRows(), acesso.getCol(), acesso.getType(), acesso.getImagemByte());
        Mat temp = new Mat();
        opencv_imgproc.cvtColor(acesso.getImagemMat(), temp, opencv_imgproc.COLOR_GRAY2BGR);
        img.setImage(Utilitarios.convertMatToImage(temp));
        temp.close();
    }
    public void setEditavel(boolean bol) {
        txtUsuario.setEditable(bol);
        btnSalvar.setDisable(!bol);
        btnCancelar.setDisable(!bol);
        btnSalvarImg.setDisable(!bol);

    }

    public void resetTxts() {
        acesso = null;
        txtUsuario.setText("");
        comboNivel.getSelectionModel().select(0);
        if (imagem != null && imagem.empty())
            imagem.close();
        imagem = null;
        img.setImage(null);
    }

    public void setAcesso(Acesso acesso) {
        this.acesso = acesso;
        txtUsuario.setText(acesso.getNome());
        comboNivel.getSelectionModel().select(Integer.toString(acesso.getNivel()));
    }

    protected class StringList implements ChangeListener<String>{
        private TextField f;

        public StringList(TextField f) {
            this.f = f;
        }
        @SuppressWarnings("null")
        @Override
        public void changed(ObservableValue<? extends String> observable, String oldValue, String newValue) {
            if(newValue == null && newValue.isEmpty()) {
                f.setBorder(bordaVerm);
            }else {
                f.setBorder(bordaDef);
            }
        }
    }
}
```

Fonte: Elaborada pelo autor.

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

Classe CCadastro:

```java
package com.view.controllers;

import java.net.URL;

/* Classe que é utilizada como controlador de listeners para interface GUI Cadastro.fxml*/
public class CCadastro implements Initializable {

    private GridPane parentPane;
    private TextField txtUnidade;
    private TextField txtProdAnual;
    private TextField txtNEmpregados;
    private ComboBox<String> comboDestino;
    private TextField txtNivelAuto;
    private TextField txtQMaquinas;
    private TextField txtCidade;
    private TextField txtCep;
    private TextField txtEndereco;
    private TextField txtPais;
    private TextField txtEstado;
    private Button btnSalvar;
    private Button btnCancelar;
    private Button btnBanir;
    private Button btnPermitir;
    private TextField txtFiscaisRece;
    private TextField txtMuniPagos;
    private TextField txtEstaReco;
    private TextField txtFedePagos;
    private TextField txtTaxasFed;
    private TextField txtAgro;
    private Button btnRemover;
    private Button btnAdicionar;
    private TableView<Agrotoxico> listAgro;
    private ObservableList<Agrotoxico> agroModel = FXCollections.observableArrayList();

    //TODO agrotoxicos

    private Cadastro c;
    private Integer nivel;
```

```java
    private Border bordaVerm = new Border(new BorderStroke(Color.RED, BorderStrokeStyle.SOLID, new CornerRadii(3),
            new BorderWidths(2), new Insets(-2)));
    private Border bordaDef;

    @SuppressWarnings("unchecked")
    @Override
    public void initialize(URL location, ResourceBundle resources) {
        bordaDef = txtUnidade.getBorder();
        setBordas();
        listAgro.setItems(agroModel);
        //Adiciona colunas na tabela
        TableAgroHelper h = new TableHelpers.TableAgroHelper();
        listAgro.getColumns().addAll(h.getColumAgro(),h.getColumnProib());


    }
    public void construtor(Cadastro cadastro, Integer nivel) {
        this.c = cadastro;
        this.nivel = nivel;
        comboDestino.setItems(FXCollections.observableArrayList(new String[] {"Externo", "Interno"}));


        ControllerBd.begin();
        if(c != null) {
            c = (Cadastro) ControllerBd.findById(Cadastro.class, c.getId());
            setValues();
        }else {
            c = new Cadastro();
        }
    }


    @FXML
    public void actBtnSalvar() {
        try {
            boolean flag = true;

            if(c.getId() == 0) { flag = false;}
```

```
            c.setUnidade(txtUnidade.getText());
            c.setProdAnual(Double.parseDouble(txtProdAnual.getText()));
            c.setnEmpregados(Integer.parseInt(txtNEmpregados.getText()));
            c.setDestino(comboDestino.getValue());
            c.setNivelAuto(Integer.parseInt(txtNivelAuto.getText()));
            c.setQantiMaquinas(Integer.parseInt(txtQMaquinas.getText()));
            c.setCidade(txtCidade.getText());
            c.setCep(txtCep.getText());
            c.setEndereco(txtEndereco.getText());
            c.setPais(txtPais.getText());
            c.setEstado(txtEstado.getText());
            if(nivel >= 2) {
                c.setInceFiscaRece(Double.parseDouble(txtFiscaisRece.getText()));
                c.setImpMuniPagos(Double.parseDouble(txtMuniPagos.getText()));
                c.setImpEstaduRecolhidos(Double.parseDouble(txtEstaReco.getText()));
                c.setImpFedPago(Double.parseDouble(txtFedePagos.getText()));
                c.setTaxasFed(Double.parseDouble(txtTaxasFed.getText()));
            }else {
                c.setInceFiscaRece(0.0);
                c.setImpMuniPagos(0.0);
                c.setImpEstaduRecolhidos(0.0);
                c.setImpFedPago(0.0);
                c.setTaxasFed(0.0);
            }

            if(!flag) ControllerBd.em.persist(c);
            else ControllerBd.em.merge(c);

            ControllerBd.commit();


            CRegistro.refreshTablePro();
            c = null;
            resetTxt();
            ((Stage) parentPane.getScene().getWindow()).close();
        } catch (Exception e) {
            e.printStackTrace();
            Alerts.showError("Falha ao inserir dados, cheque os campos");
        }
```

```
    @FXML
    public void actBtnCancelar() {
        resetTxt();
        ControllerBd.checkTrans();
        ((Stage) parentPane.getScene().getWindow()).close();

    }

    @FXML
    public void actBtnRemover() {
        int index = listAgro.getSelectionModel().getSelectedIndex();
        if(index > -1) {
            Agrotoxico agro = agroModel.get(index);
            c.removeAgrotoxico(agro);
            agroModel.remove(agro);
        }
    }

    @FXML
    public void actBtnBanir() {
        int row = listAgro.getSelectionModel().getSelectedIndex();
        if(row > -1) {
            Agrotoxico agro = agroModel.get(row);
            agro.setProibido(true);
            c.checkContemProibido();
            listAgro.refresh();
        }
    }
    @FXML
    public void actBtnPermitir() {
        int row = listAgro.getSelectionModel().getSelectedIndex();
        if(row > -1) {
            Agrotoxico agro = agroModel.get(row);
            agro.setProibido(false);
            c.checkContemProibido();
            listAgro.refresh();
        }
    }
```

```
213     @FXML
214     public void actBtnAdicionar() {
215         try {
216             String agroS = txtAgro.getText();
217             if(!agroS.isEmpty()) {
218                 Agrotoxico agro = ControllerBd.findAgroByName(agroS);
219                 if(agro == null) agro = new Agrotoxico(null, agroS, ControllerBd.checkIfProibido(agroS));
220                 ControllerBd.em.persist(agro);
221                 c.addAgrotoxico(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.getAgrotoxicos());
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);
```

```
            txtMuniPagos.setEditable(bol);
            txtEstaReco.setEditable(bol);
            txtFedePagos.setEditable(bol);
            txtTaxasFed.setEditable(bol);
        }else {
            txtFiscaisRece.setEditable(false);
            txtMuniPagos.setEditable(false);
            txtEstaReco.setEditable(false);
            txtFedePagos.setEditable(false);
            txtTaxasFed.setEditable(false);
        }
        if(bol && nivel >= 3) {
            btnRemover.setDisable(false);
            btnAdicionar.setDisable(false);
        }else {
            btnRemover.setDisable(true);
            btnAdicionar.setDisable(true);
        }
        btnCancelar.setDisable(!bol);
        btnSalvar.setDisable(!bol);
    }

    private void setBordas() {
        // Bordas customizadas para checar valores que não são double ou int ou string
        txtUnidade.textProperty().addListener(new StringList(txtUnidade));
        txtCidade.textProperty().addListener(new StringList(txtCidade));
        txtCep.textProperty().addListener(new StringList(txtCep));
        txtEndereco.textProperty().addListener(new StringList(txtEndereco));
        txtPais.textProperty().addListener(new StringList(txtPais));
        txtEstado.textProperty().addListener(new StringList(txtEstado));
        txtProdAnual.textProperty().addListener(new DoubleList(txtProdAnual));
        txtNEmpregados.textProperty().addListener(new IntList(txtNEmpregados));
        txtNivelAuto.textProperty().addListener(new IntList(txtNivelAuto));
        txtQMaquinas.textProperty().addListener(new IntList(txtQMaquinas));
        txtFiscaisRece.textProperty().addListener(new DoubleList(txtFiscaisRece));
        txtMuniPagos.textProperty().addListener(new DoubleList(txtMuniPagos));
        txtEstaReco.textProperty().addListener(new DoubleList(txtEstaReco));
        txtFedePagos.textProperty().addListener(new DoubleList(txtFedePagos));
        txtTaxasFed.textProperty().addListener(new DoubleList(txtTaxasFed));
```

```
    }

    protected class StringList implements ChangeListener<String>{
        private TextField f;

        public StringList(TextField f) {
            this.f = f;
        }
        @Override
        public void changed(ObservableValue<? extends String> observable, String oldValue, String newValue) {
            if(newValue == null || newValue.isEmpty()) {
                f.setBorder(bordaVerm);
            }else {
                f.setBorder(bordaDef);
            }
        }
    }
    protected class DoubleList implements ChangeListener<String> {

        private TextField f;

        public DoubleList(TextField f) {
            this.f = f;
        }

        @Override
        public void changed(ObservableValue<? extends String> observable, String oldValue, String newValue) {
            try {
                Double.parseDouble(newValue);
                f.setBorder(bordaDef);
            } catch (NumberFormatException e) {
                f.setBorder(bordaVerm);
            }
        }

    }

    protected class IntList implements ChangeListener<String> {
```

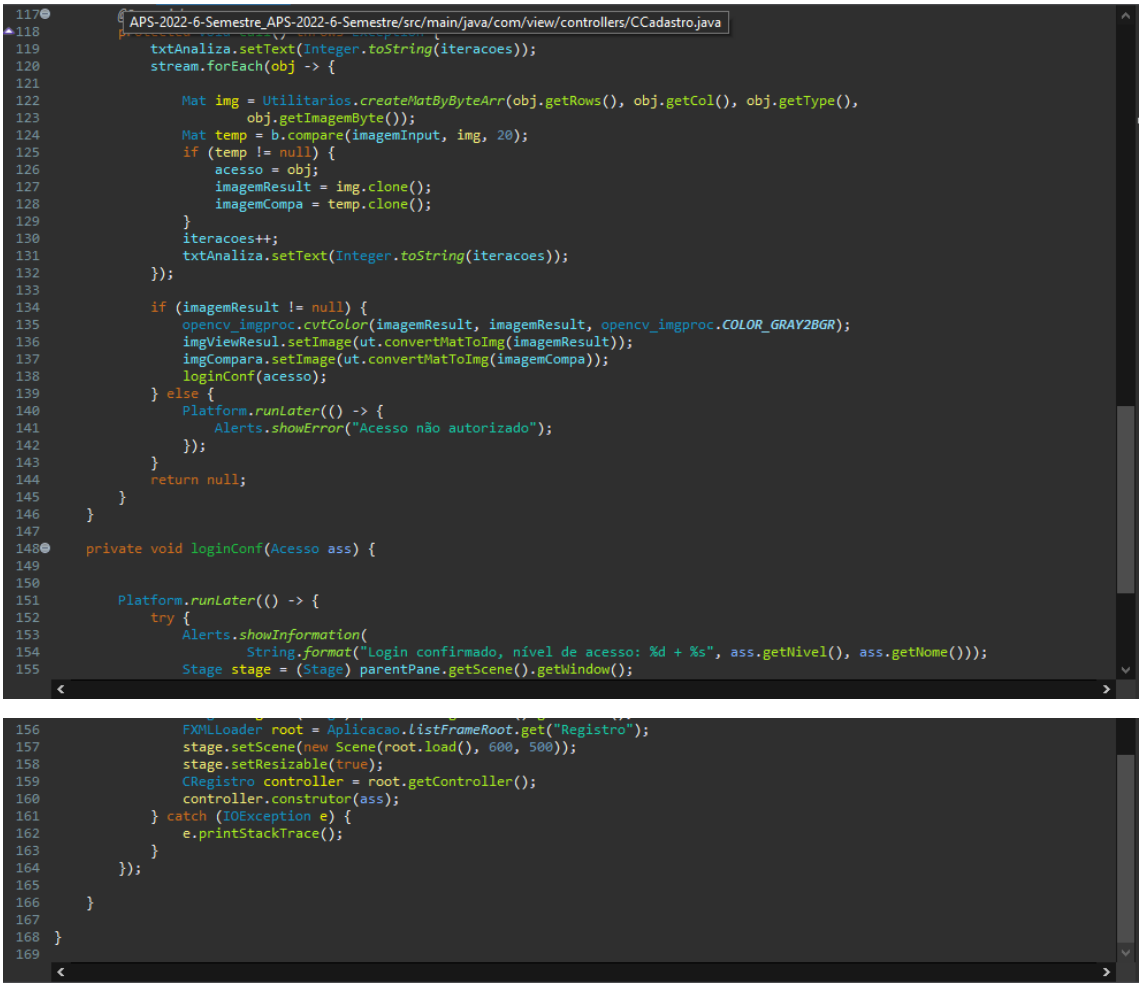
```
        private TextField f;

        public IntList(TextField f) {
            this.f = f;
        }

        @Override
        public void changed(ObservableValue<? extends String> observable, String oldValue, String newValue) {
            try {
                if(newValue.matches("\\d+")) f.setBorder(bordaDef);
                else {
                    f.setBorder(bordaVerm);
                }
            } catch (NumberFormatException e) {
                f.setBorder(bordaVerm);
            }
        }

    }

}
```

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 inteface do usuário.

Classe CLogin:

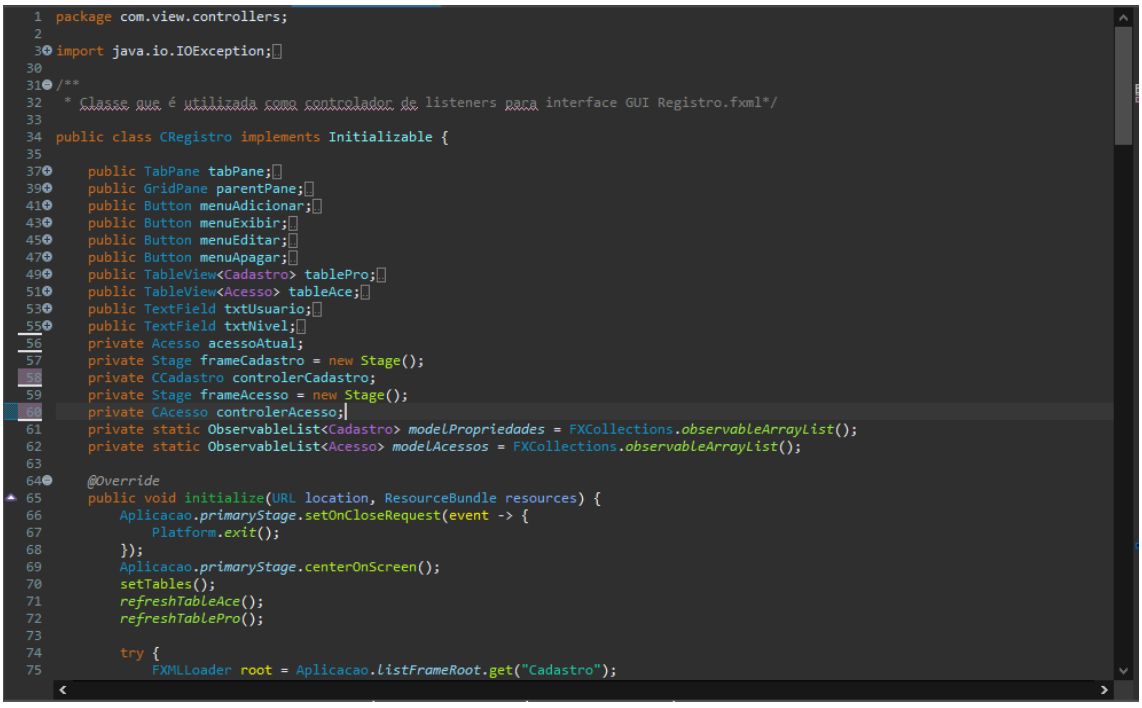
```java
package com.view.controllers;

import java.io.File;

/**
 * Classe que é utilizada como controlador de listeners para interface GUI
 * Login.fxml
 */
public class CLogin {

    private TextField txtAnaliza;
    private BorderPane parentPane;
    private TextField txtSenhaMestra;
    public Button btnLogin;
    private ImageView imgViewInput;
    private ImageView imgCompara;
    private ImageView imgViewResul;
    private Button btnCarregar;
    private Button btnIniciarRe;
    private Acesso acesso;
    private Mat imagemInput;

    public CLogin() {

    }

    @FXML
    private void actBtnCarregar() {
        try {

            FileChooser cho = new FileChooser();
            File f = cho.showOpenDialog(null);
            Mat img = opencv_imgcodecs.imread(f.getAbsolutePath());
            Mat show = img.clone();
            imagemInput = Biometria.processImagem(show);
            opencv_imgproc.cvtColor(imagemInput, show, opencv_imgproc.COLOR_GRAY2BGR);

            imgViewInput.setImage(Utilitarios.convertMatToImage(show));

            btnIniciarRe.setDisable(false);
        } catch (Exception e) {
            Alert a = new Alert(AlertType.ERROR, "Formato de arquivo inválido");
            a.setHeaderText(null);
            a.showAndWait();
            e.printStackTrace();
        }
    }

    @FXML
    private void actBtnIniciarRe() {
        Stream<Acesso> stre = ControllerBd.getAcessoAsStream();
        ThreadCompara t = new ThreadCompara(stre);
        new Thread(t).start();
    }

    @FXML
    private void actBtnLogin() {
        String senha = txtSenhaMestra.getText();
        if(senha != null && !senha.isEmpty() && senha.contentEquals("1020304050")) {
            Acesso ace = new Acesso(null, "Master", 3, null, 0, 0, 0, null);
            loginConf(ace);
        }else {
            Alerts.showError("Senha Incorreta");
        }

    }

    private class ThreadCompara extends Task<Void> {

        private Stream<Acesso> stream;
        private Integer iteracoes = 0;
        private Biometria b = new Biometria();
        private Mat imagemResult;
        private Mat imagemCompa;
        private Utilitarios ut = new Utilitarios();

        public ThreadCompara(Stream<Acesso> stream) {
            this.stream = stream;
```



Fonte: Elaborada pelo autor.

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

Classe CRegistro:

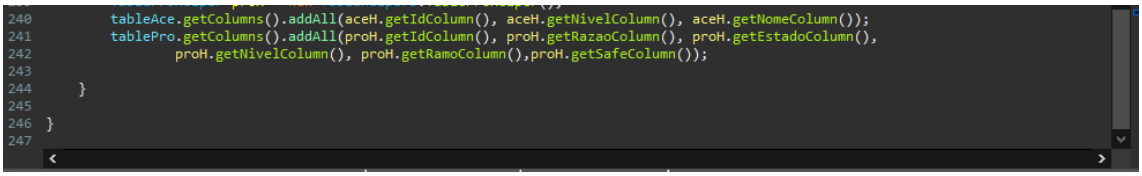


```
            frameCadastro.setScene(new Scene(root.load(), 650, 600));
            frameCadastro.setResizable(false);
            frameCadastro.getIcons().addAll(Aplicacao.iconsImg);
            controlerCadastro = root.getController();
            frameCadastro.setTitle("Cadastro");

            root = Aplicacao.listFrameRoot.get("Acesso");
            frameAcesso.setScene(new Scene(root.load(), 360, 360));
            frameAcesso.setResizable(false);
            frameAcesso.getIcons().addAll(Aplicacao.iconsImg);
            controlerAcesso = root.getController();
            frameAcesso.setTitle("Cadastro Acesso");

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

    }
    public void construtor(Acesso acesso) {
        this.acessoAtual = acesso;
        txtUsuario.setText(acesso.getNome());
        txtNivel.setText(Integer.toString(acesso.getNivel()));
    }

    @FXML
    public void actMenuAdicionar() {
        if (tabPane.getSelectionModel().getSelectedIndex() == 0) {
            controlerCadastro.resetTxt();
            controlerCadastro.construtor(null, acessoAtual.getNivel());
            controlerCadastro.setEditavel(true);
            frameCadastro.show();
            frameCadastro.centerOnScreen();
        } else {
            if(acessoAtual.getNivel() >= 3) {
                controlerAcesso.resetTxts();
                controlerAcesso.construtor(null);
                controlerAcesso.setEditavel(true);
                frameAcesso.show();
                frameAcesso.centerOnScreen();
```

```
            }else Alerts.showError("Apenas usuarios nível 3 podem cadastrar acessos");
        }

    }

    @FXML
    public void actMenuExibir() {
        if (tabPane.getSelectionModel().getSelectedIndex() == 0) {
            controlerCadastro.resetTxt();
            controlerCadastro.construtor(tablePro.getSelectionModel().getSelectedItem(), acessoAtual.getNivel());
            controlerCadastro.setEditavel(false);
            frameCadastro.show();
            frameCadastro.centerOnScreen();
        } else {
            if(acessoAtual.getNivel() >= 3) {
                controlerAcesso.resetTxts();
                controlerAcesso.construtor(tableAce.getSelectionModel().getSelectedItem());
                controlerAcesso.setEditavel(false);
                frameAcesso.show();
                frameAcesso.centerOnScreen();
            }else Alerts.showError("Apenas usuarios nível 3 podem Exibir acessos");
        }

    }

    @FXML
    public void actMenuEditar() {
        if (tabPane.getSelectionModel().getSelectedIndex() == 0) {
            controlerCadastro.resetTxt();
            controlerCadastro.construtor(tablePro.getSelectionModel().getSelectedItem(), acessoAtual.getNivel());
            controlerCadastro.setEditavel(true);

            frameCadastro.show();
        } else {
            if(acessoAtual.getNivel() >= 3) {
                controlerAcesso.resetTxts();
                controlerAcesso.construtor(tableAce.getSelectionModel().getSelectedItem());
                controlerAcesso.setEditavel(true);
                frameAcesso.show();
```

```
                frameAcesso.centerOnScreen();
            }else Alerts.showError("Apenas usuarios nível 3 podem Editar acessos");
        }
    }

    @FXML
    public void actMenuApagar() {
        try {
            if(Alerts.showConfirmation("Deseja apagar linha selecionada")) {

                if (tabPane.getSelectionModel().getSelectedIndex() == 0) {
                    if(tablePro.getSelectionModel().getSelectedIndex() > -1) {
                        Cadastro obj = tablePro.getSelectionModel().getSelectedItem();
                        if(ControllerBd.checkPersist(obj)) {
                            ControllerBd.delete(obj);
                        }else {
                            obj = (Cadastro) ControllerBd.findById(Cadastro.class, obj.getId());
                            ControllerBd.delete(obj);

                        }
                        refreshTablePro();
                    }else Alerts.showError("Nenhuma linha selecionada");


                } else {
                    if(tableAce.getSelectionModel().getSelectedIndex() > -1) {
                        Acesso obj = tableAce.getSelectionModel().getSelectedItem();
                        if(ControllerBd.checkPersist(obj)) {
                            ControllerBd.delete(obj);
                        }else {
                            obj = (Acesso) ControllerBd.findById(Acesso.class, obj.getId());
                            ControllerBd.delete(obj);

                        }
                        refreshTableAce();
                    }else Alerts.showError("Nenhuma linha selecionada");

                }
            }
```

```
    public static void refreshTablePro() {
        modelPropriedades.clear();
        ControllerBd.checkTrans();
        ControllerBd.em.clear();
        @SuppressWarnings("unchecked")
        List<Cadastro> l = Aplicacao.em.createQuery("select a from CADASTRO a ").getResultList();
        if(l.size() > 0) {
            for(Cadastro cad : l) {
                //TODO Atualiar get contem proibido quando atualiar tabela
                cad.checkContemProibido();
                System.out.println(cad.getContemProibido());
                modelPropriedades.add(cad);
            }
        }

    }

    public static void refreshTableAce() {
        modelAcessos.clear();
        ControllerBd.checkTrans();
        @SuppressWarnings("unchecked")
        List<Object[]> l = Aplicacao.em.createQuery("select a.id,a.nivel,a.nome from ACESSO a").getResultList();
        if(l.size() > 0) {
            for(Object[] a : l) {
                modelAcessos.add(new Acesso((int)a[0], (String)a[2], (int)a[1]));
            }

        }



    }

    @SuppressWarnings("unchecked")
    private void setTables() {
        tableAce.setItems(modelAcessos);
        tablePro.setItems(modelPropriedades);
        TableAceHelper aceH = new TableHelpers.TableAceHelper();
        TableProHelper proH = new TableHelpers.TableProHelper();
```



```
        tableAce.getColumns().addAll(aceH.getIdColumn(), aceH.getNivelColumn(), aceH.getNomeColumn());
        tablePro.getColumns().addAll(proH.getIdColumn(), proH.getRazaoColumn(), proH.getEstadoColumn(),
                proH.getNivelColumn(), proH.getRamoColumn(),proH.getSafeColumn());

    }
}
```

Fonte: Elaborada pelo autor.

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

Classe SplashScreen:

```java
package com.view.models;

import javax.swing.JFrame;

public class SplashScreen extends JFrame{

    private static final long serialVersionUID = 1L;

    public SplashScreen() {
        super("Aguarde");
        setSize(256,117);
        setLocationRelativeTo(null);
        setDefaultCloseOperation(DO_NOTHING_ON_CLOSE);
        setResizable(false);
        JLabel lblNewLabel = new JLabel("Iniciando aplicação, Aguarde");
        lblNewLabel.setFont(new Font("Tahoma", Font.PLAIN, 16));
        lblNewLabel.setHorizontalAlignment(SwingConstants.CENTER);
        getContentPane().add(lblNewLabel, BorderLayout.CENTER);
        setVisible(true);
    }

    public void close() {
        dispose();
    }
}

```

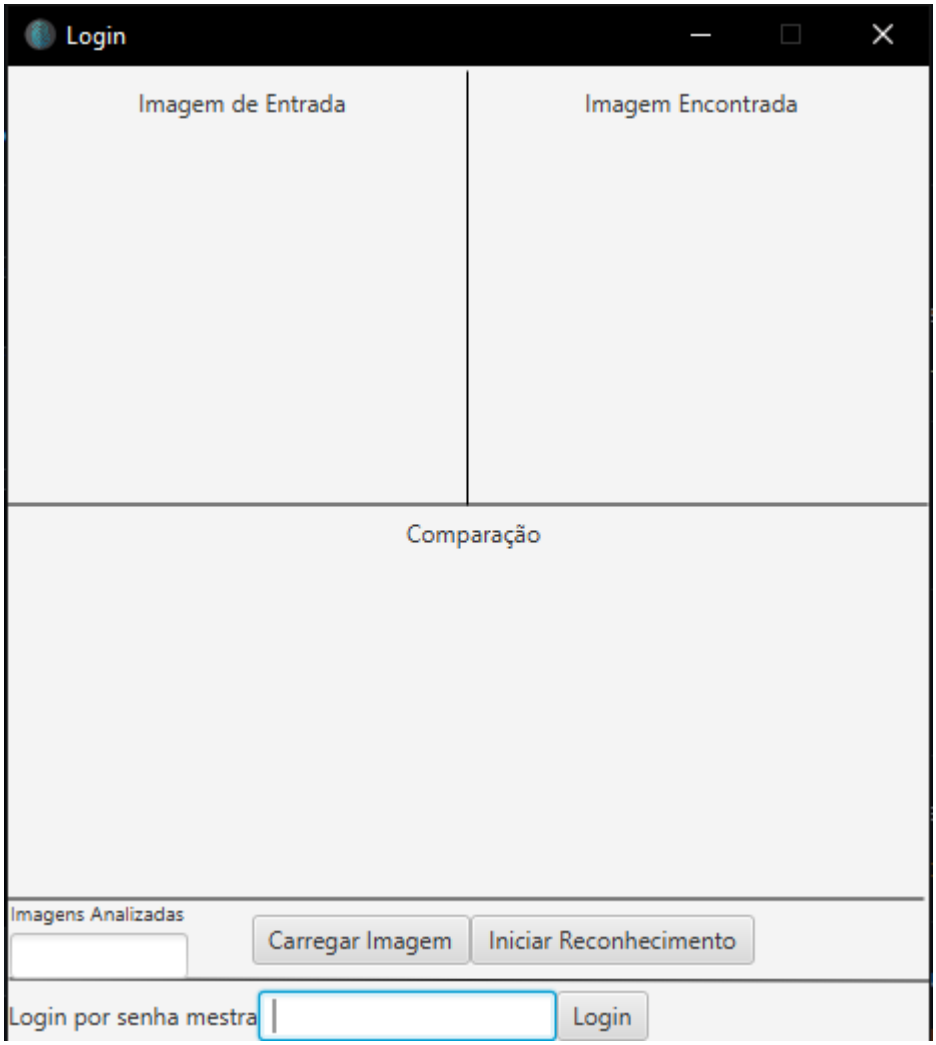
Fonte: Elaborada pelo autor.

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



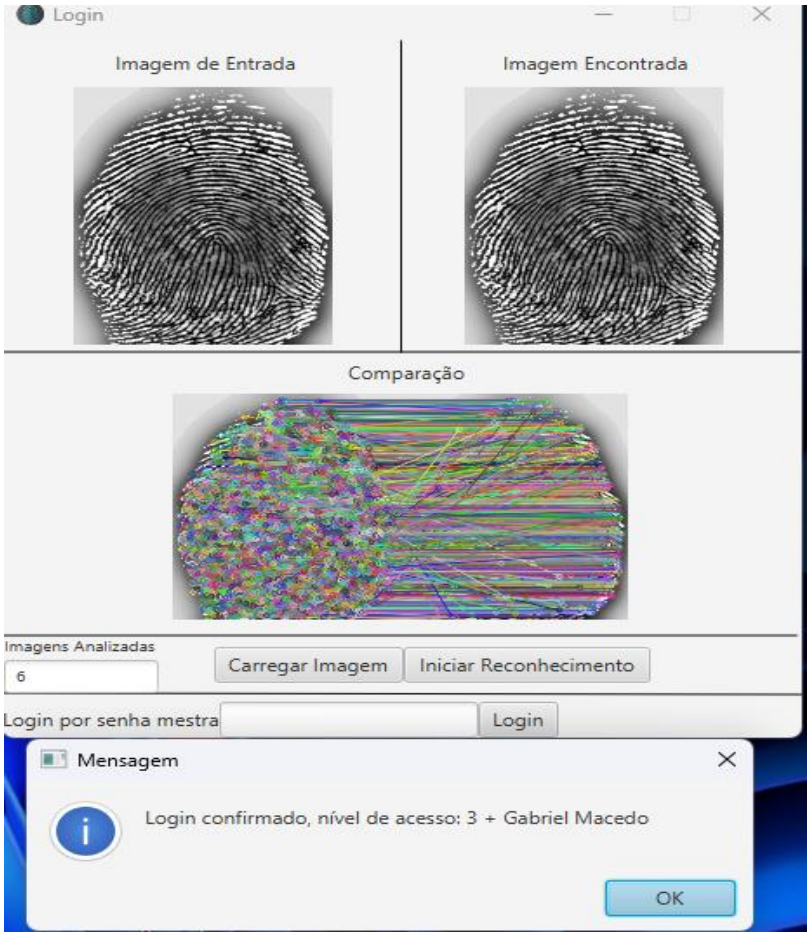
Fonte: Elaborada pelo Autor

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 respetivo nível concedido aos dados armazenados.

ACESSO AUTORIZADO UTILIZANDO DIGITAL

Figura 17

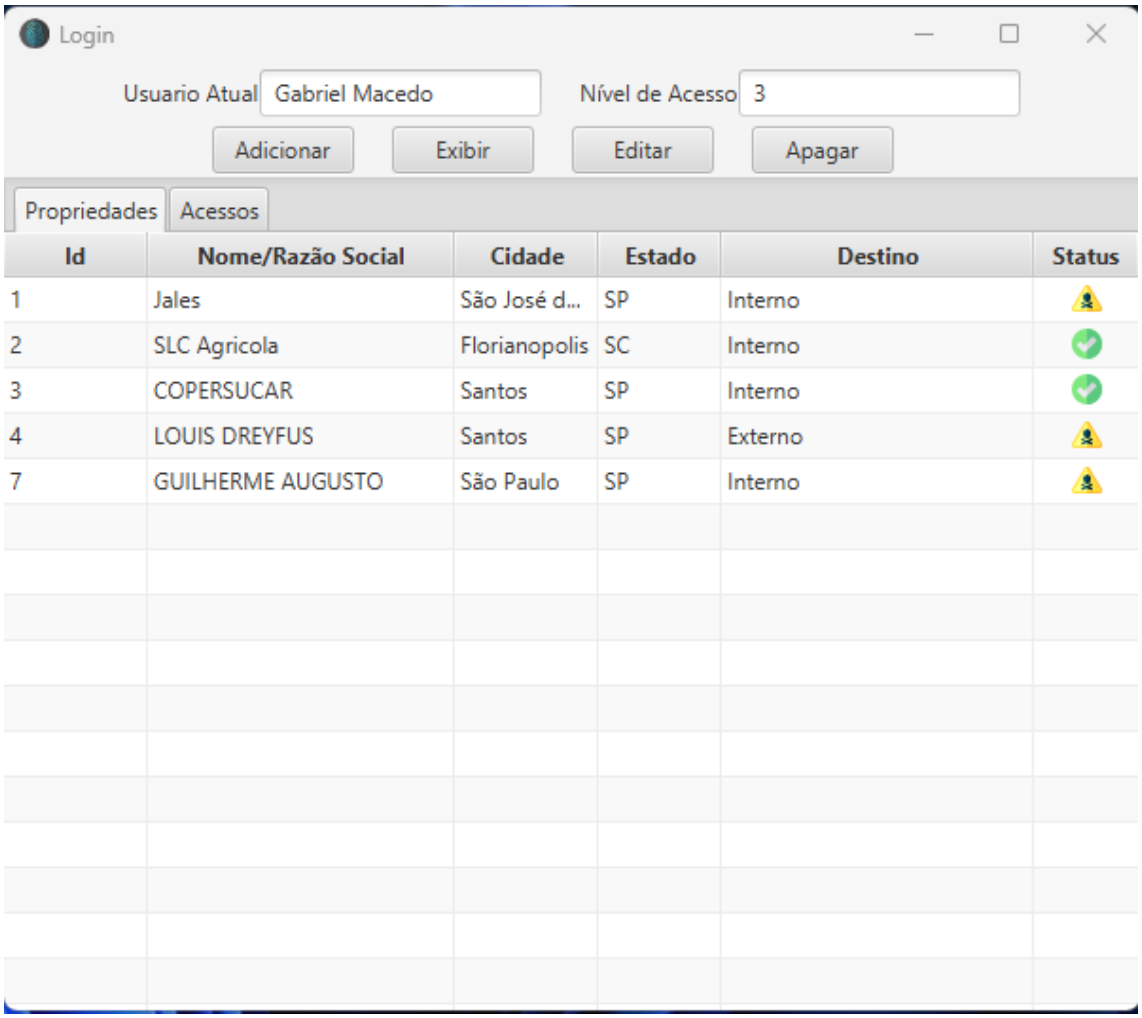


Fonte: Elaborada pelo Autor

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 respetivo nível concedido aos dados armazenados.

INTERFACE DE PROPRIEDADES REGISTRADAS

Figura 18



Fonte: Elaborada pelo Autor

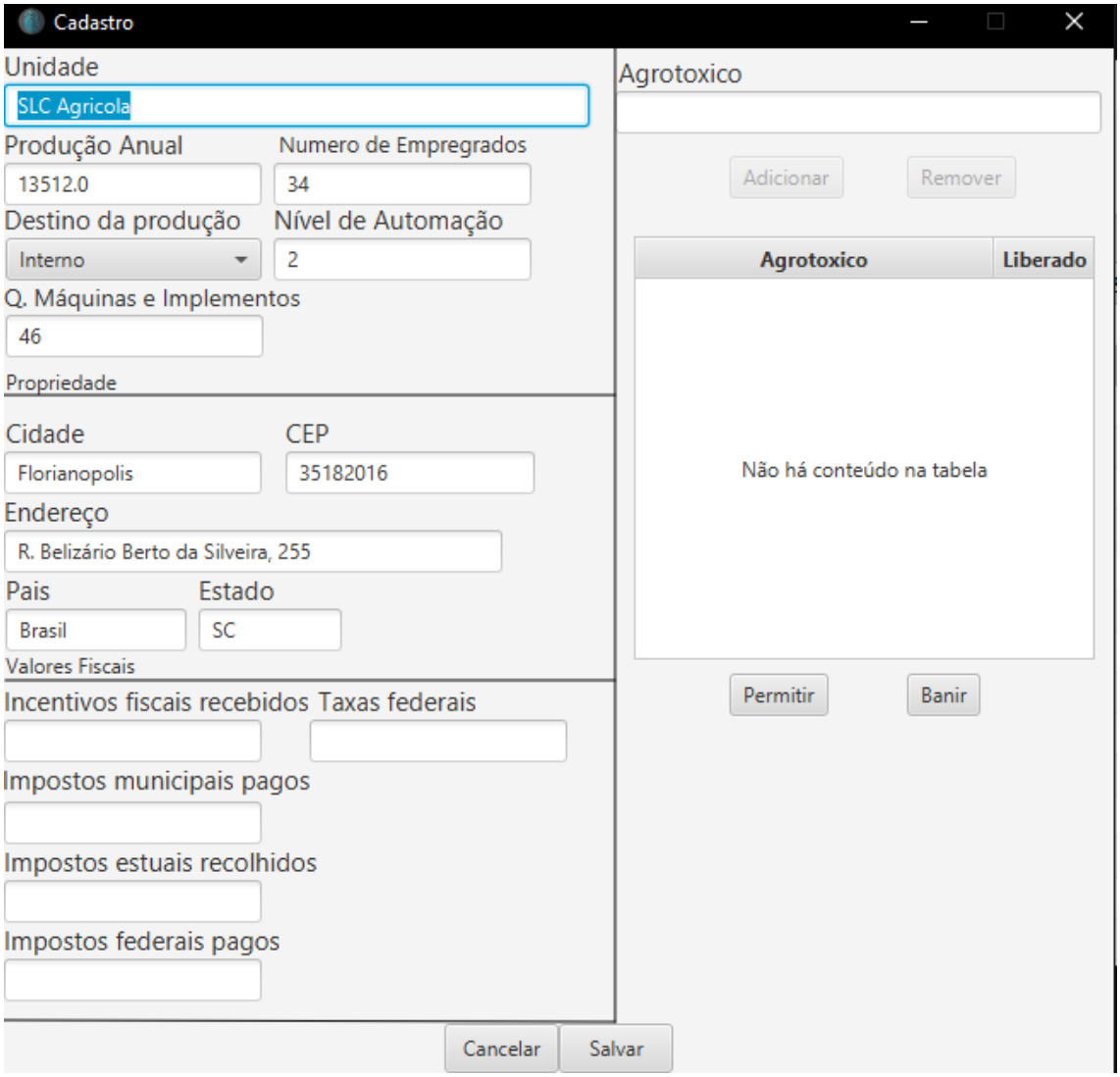
A imagem, é seguido pelo login efetuado anteriormente. Temos informações do usuário conectado e a disposição dos dados armazenados. Temos informações das empresas cadastradas e seus respectivos endereços comerciais. Sendo possível adicionar, alterar, exibir ou apagar os dados, dependendo do nível de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

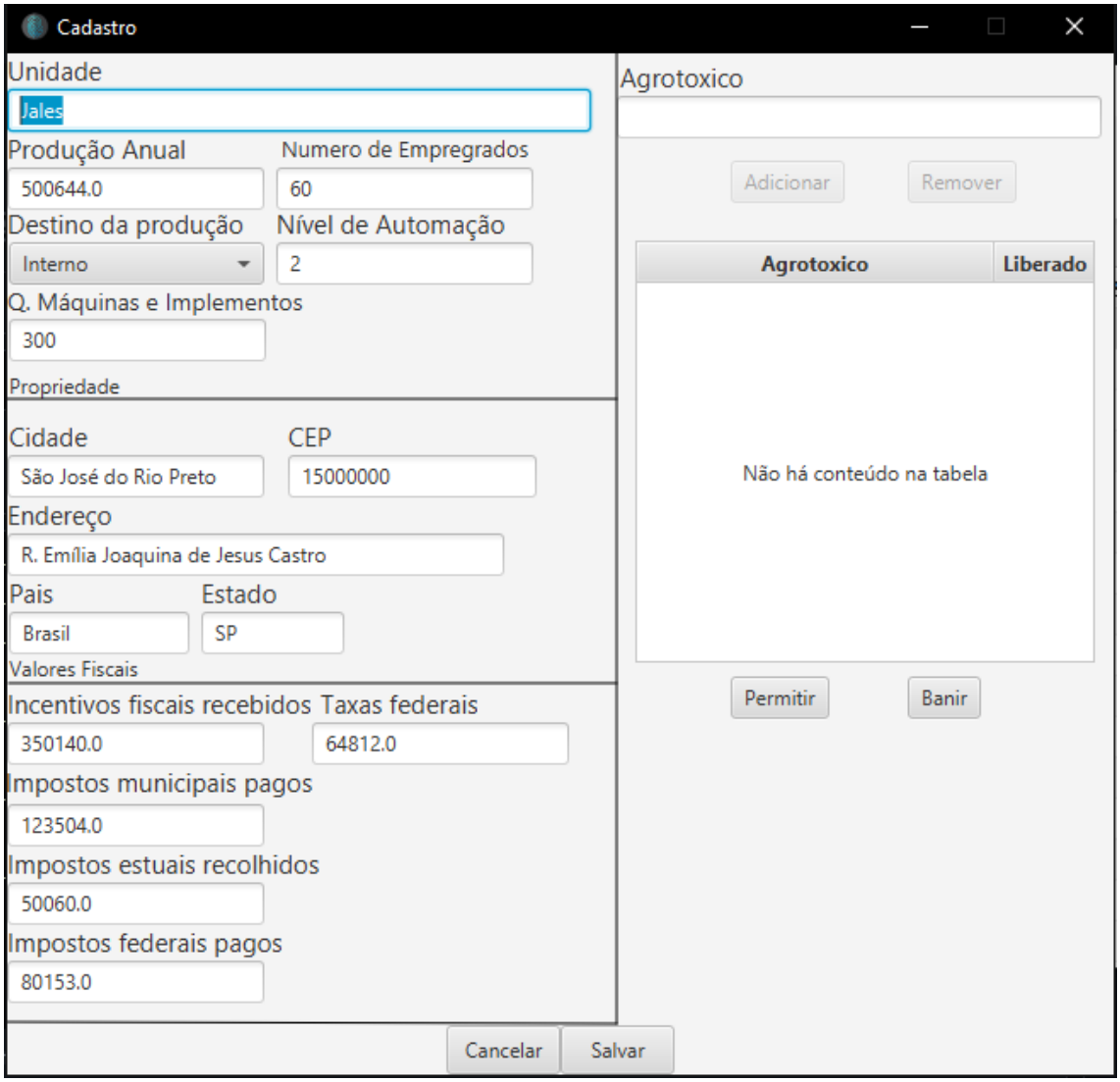
Figura 19



Fonte: Elaborada pelo Autor

INTERFACE DE CADASTRO NÍVEL 2

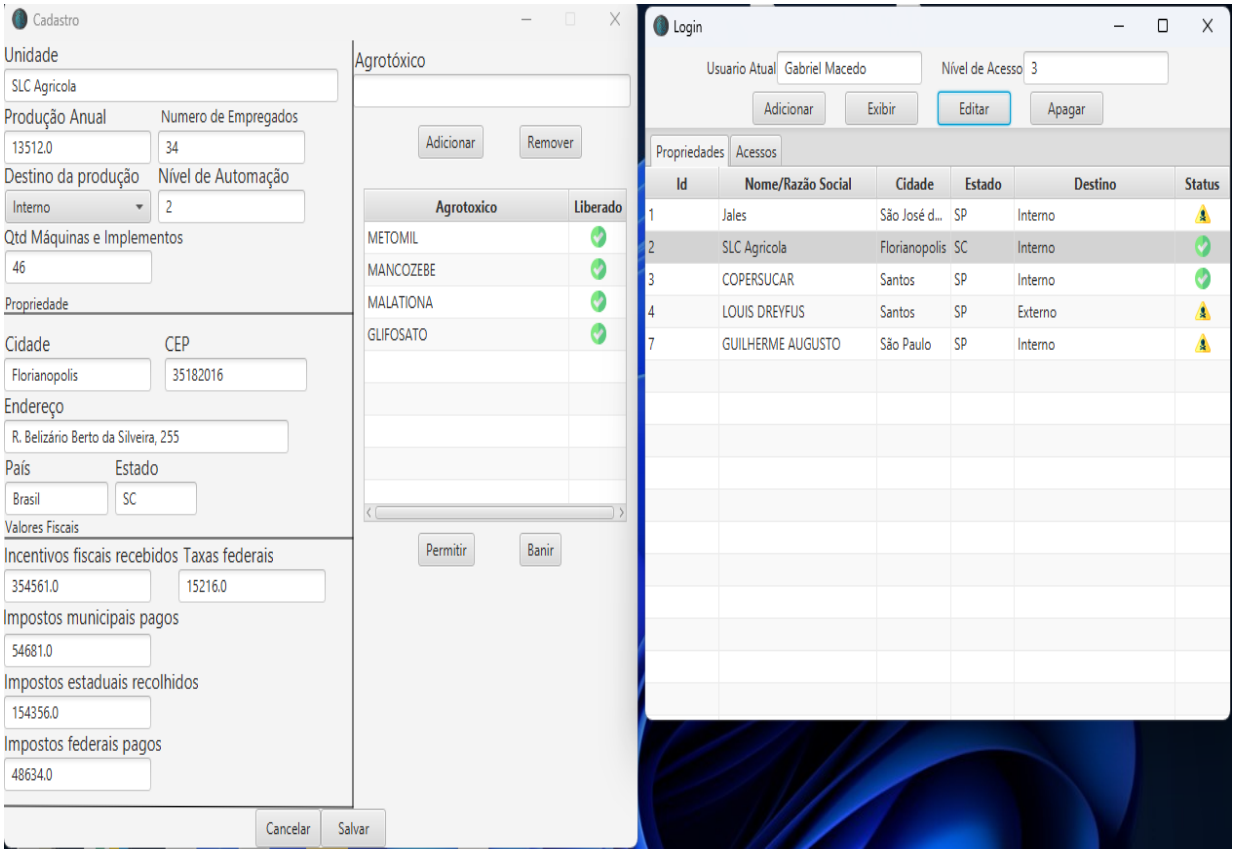
Figura 20



Fonte: Elaborada pelo Autor

INTERFACE DE CADASTRO NÍVEL 3

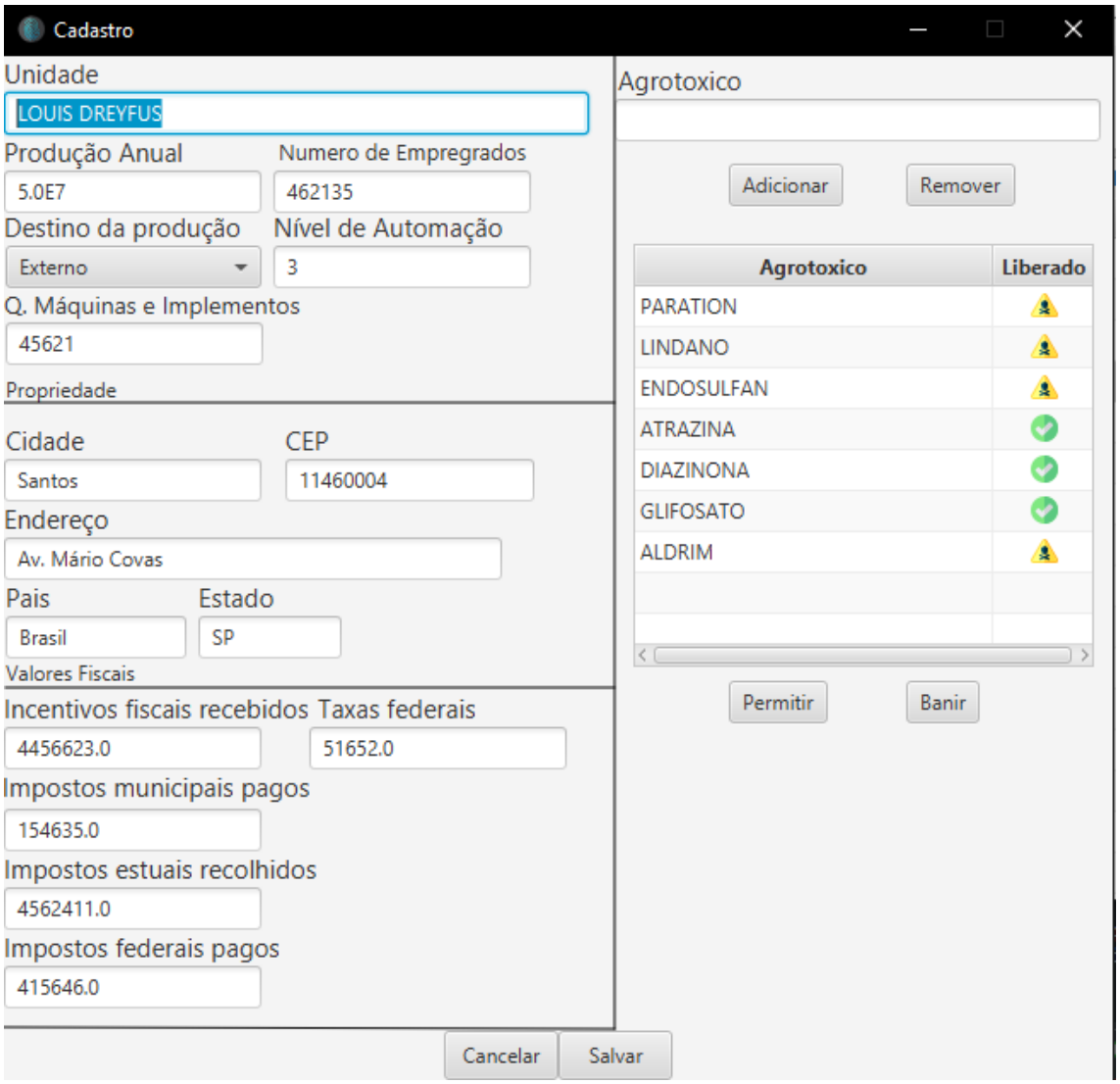
Figura 20



Fonte: Elaborada pelo Autor

Interface Cadastro Agrotóxico

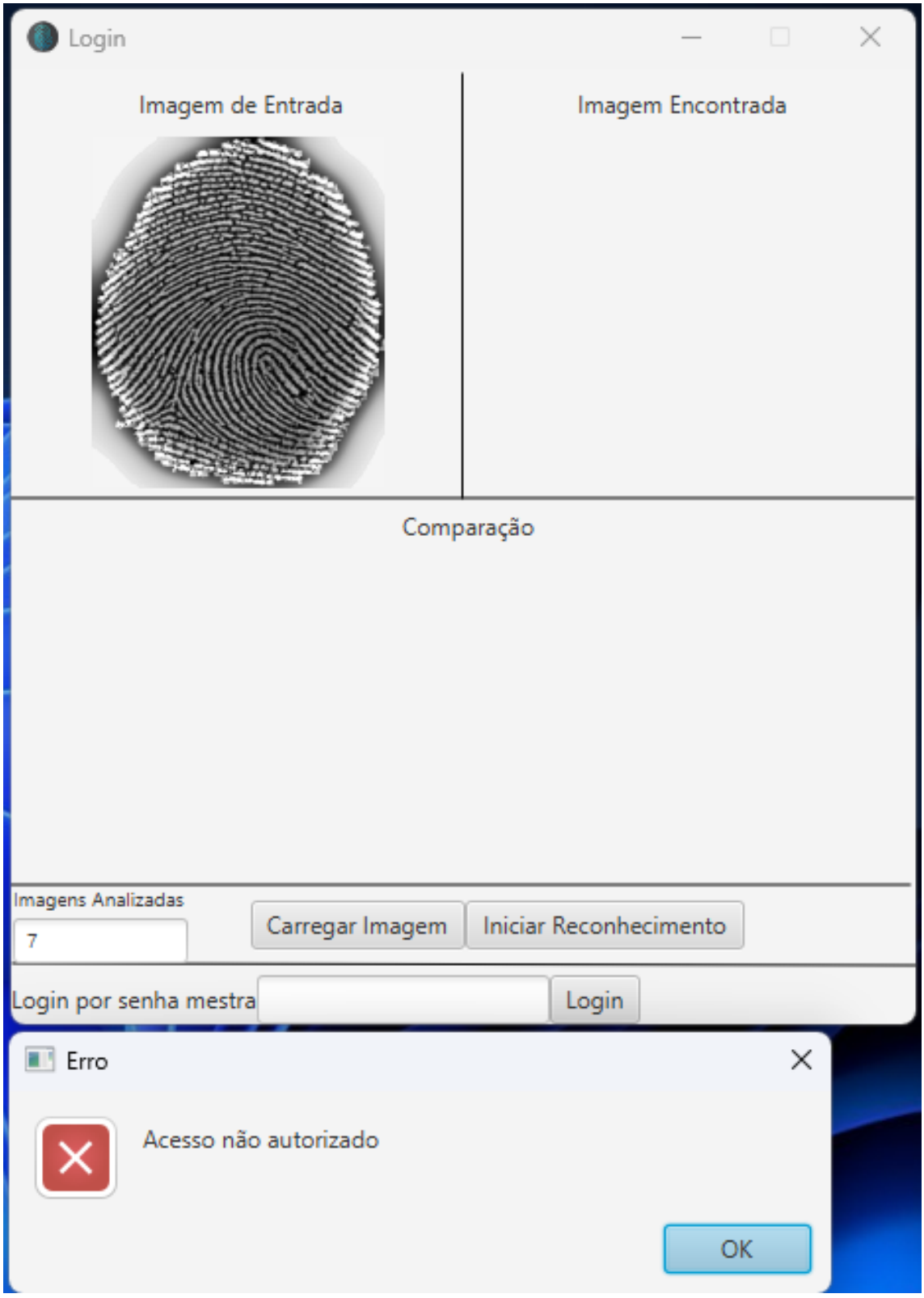
Figura 21



Fonte: Elaborada pelo Autor

DIGITAL NÃO CADASTRADA

Figura 22

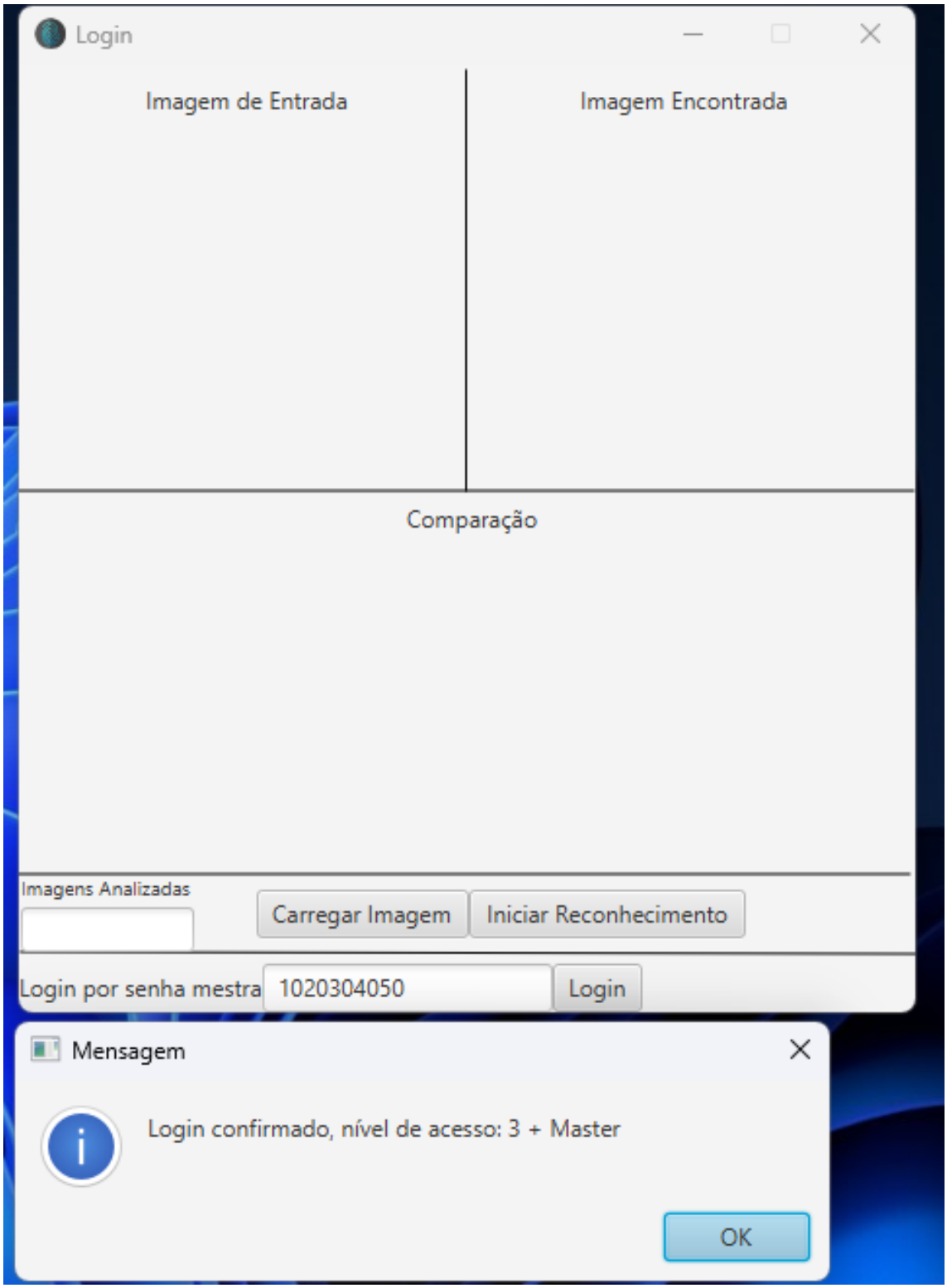


Fonte: Elaborada pelo Autor

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

LOGIN ATRÁVES DE SENHA

Figura 23

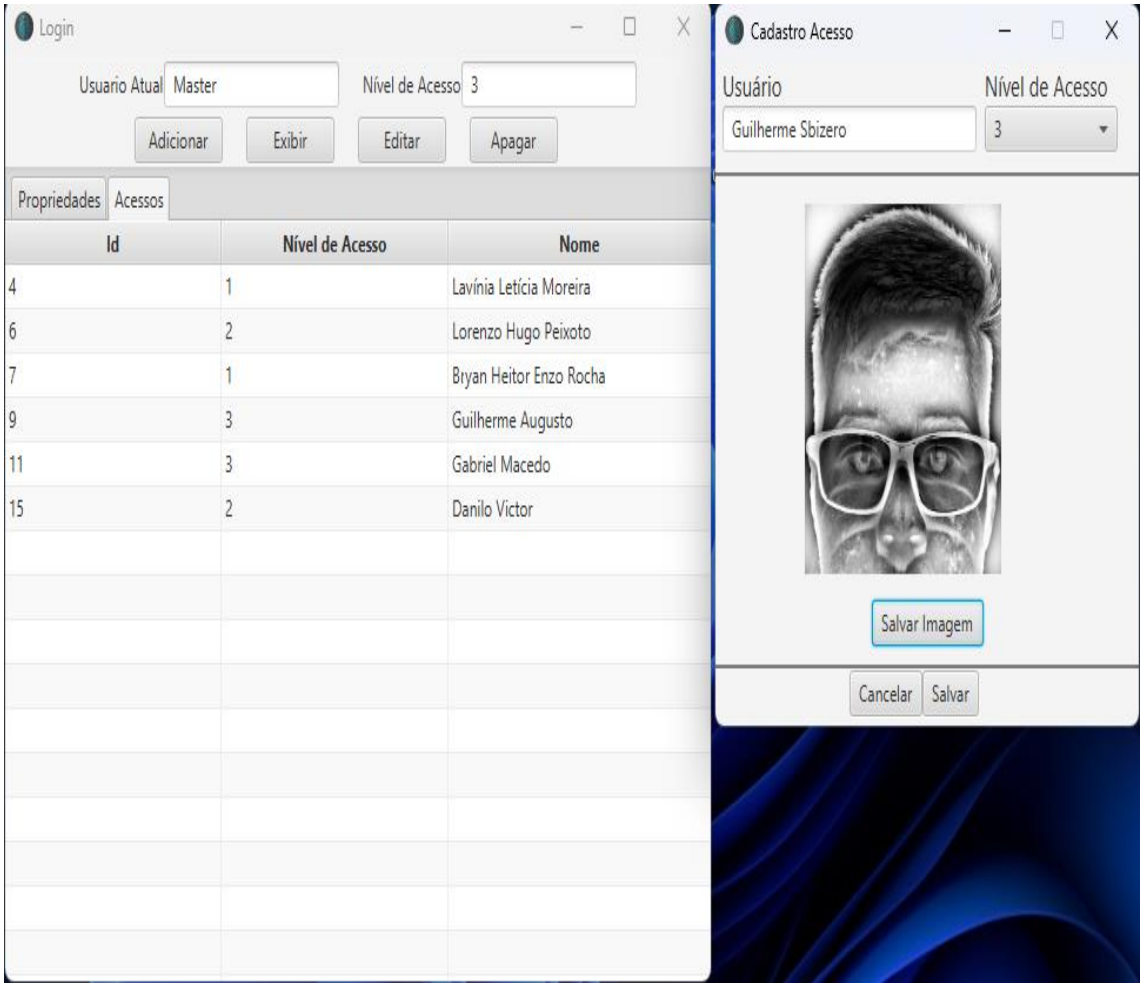


Fonte: Elaborada pelo Autor

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

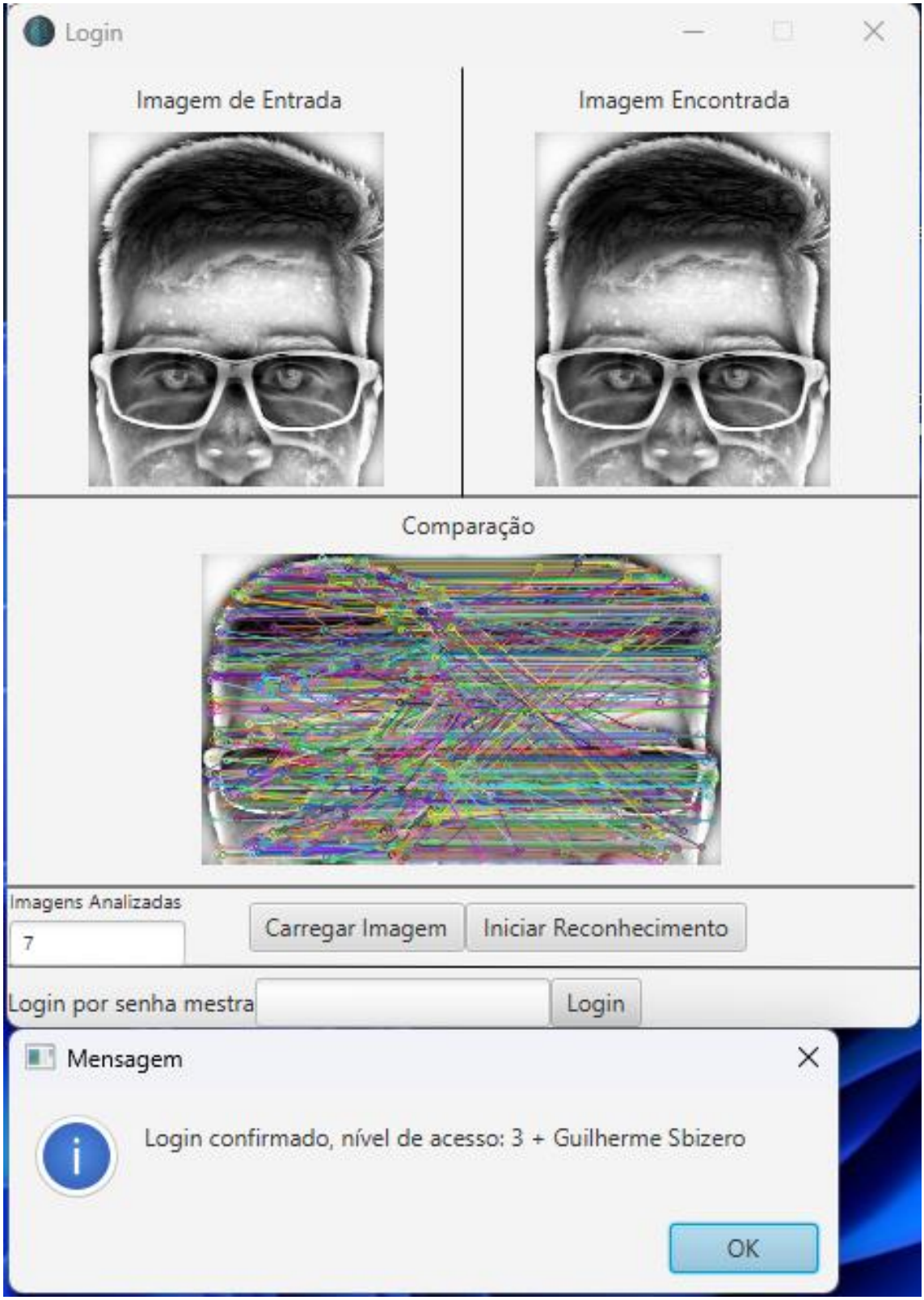


Fonte: Elaborada pelo Autor

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**. Disponívelem: <https://www.profissionaisti.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. **Biométria: Análise de Assinaturas**. Disponível em:

<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>. Acesso em:20 nov. 2016.

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

<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:

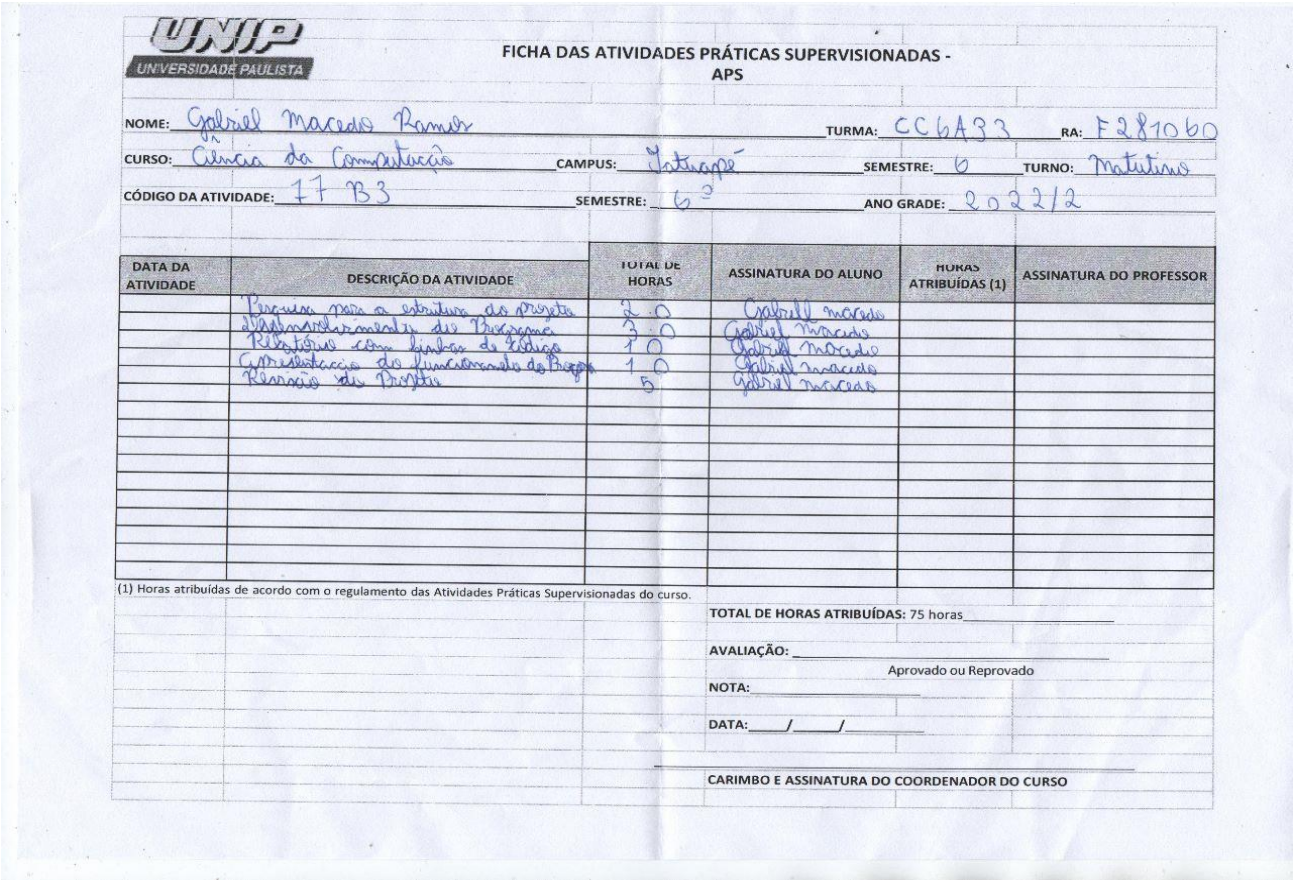
<https://www.tutorialspoint.com/biometrics/physiological_modalities.htm>. Acessoem: 20 nov. 2016.

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



UNIP
UNIVERSIDADE PAULISTA

FICHA DAS ATIVIDADES PRÁTICAS SUPERVISIONADAS - APS

NOME: Gabriel Macedo Ramos TURMA: CC6A33 RA: F281060

CURSO: Ciência da Computação CAMPUS: Jaguariúna SEMESTRE: 6 TURNO: Matutino

CÓDIGO DA ATIVIDADE: 17 B3 SEMESTRE: 6º ANO GRADE: 2022/2

DATA DA ATIVIDADE	DESCRIÇÃO DA ATIVIDADE	TOTAL DE HORAS	ASSINATURA DO ALUNO	HORAS ATRIBUÍDAS (1)	ASSINATURA DO PROFESSOR
	Pesquisa para a estrutura do projeto	2.0	Gabriel Macedo		
	Desenvolvimento do Programa	3.0	Gabriel Macedo		
	Relatório com linhas de código	1.0	Gabriel Macedo		
	Apresentação do funcionamento do Programa	1.0	Gabriel Macedo		
	Revisão do Projeto	5	Gabriel Macedo		

(1) Horas atribuídas de acordo com o regulamento das Atividades Práticas Supervisionadas do curso.

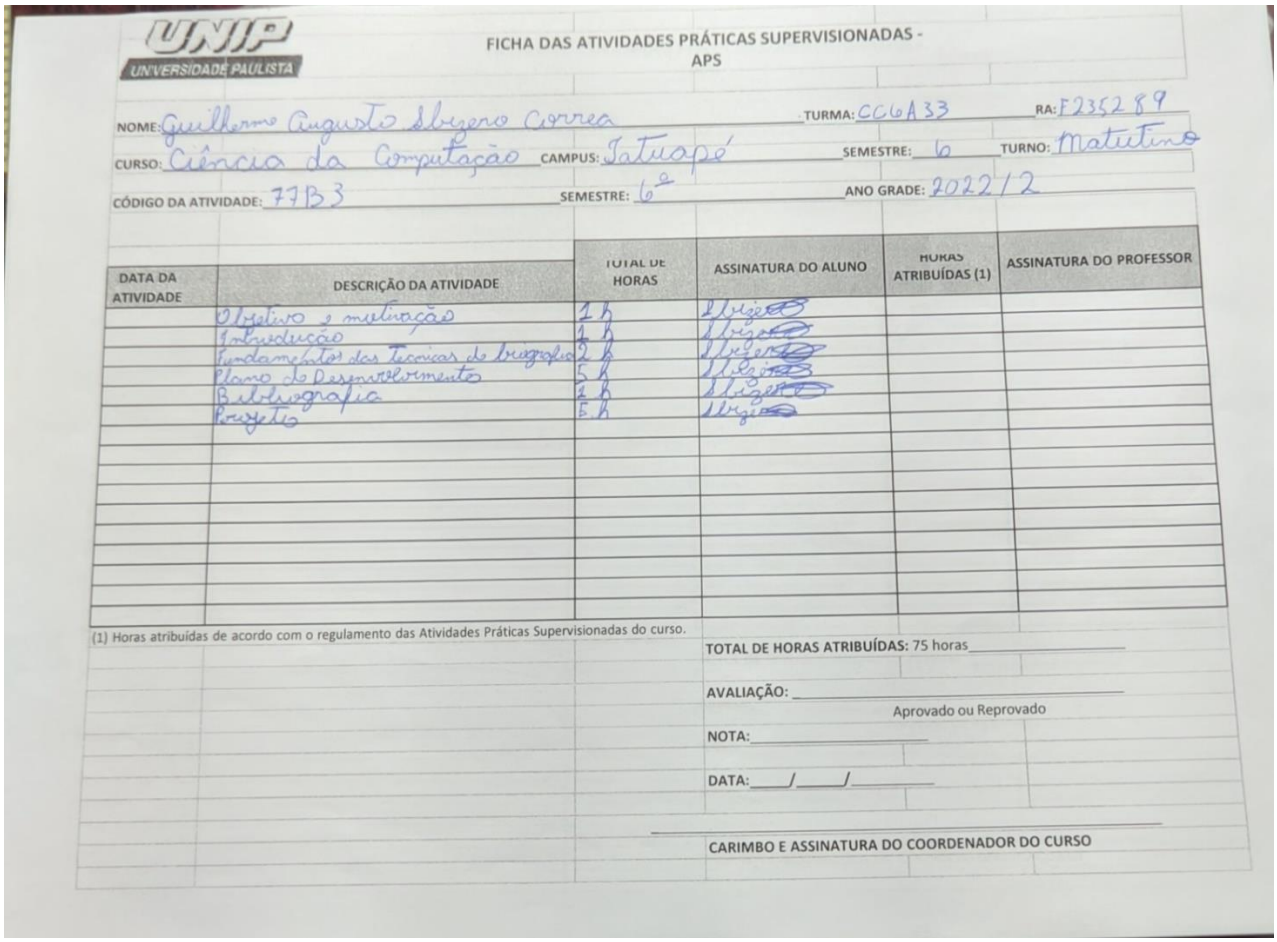
TOTAL DE HORAS ATRIBUÍDAS: 75 horas

AVALIAÇÃO: _____
Aprovado ou Reprovado

NOTA: _____

DATA: ____/____/_____

_____
CARIMBO E ASSINATURA DO COORDENADOR DO CURSO



UNIP
UNIVERSIDADE PAULISTA

FICHA DAS ATIVIDADES PRÁTICAS SUPERVISIONADAS - APS

NOME: Guilherme Augusto Ibrizero Correa TURMA: CC6A33 RA: F235289

CURSO: Ciência da Computação CAMPUS: Jatuapé SEMESTRE: 6 TURNO: Matutino

CÓDIGO DA ATIVIDADE: 77B3 SEMESTRE: 6º ANO GRADE: 2022/2

DATA DA ATIVIDADE	DESCRIÇÃO DA ATIVIDADE	TOTAL DE HORAS	ASSINATURA DO ALUNO	HORAS ATRIBUÍDAS (1)	ASSINATURA DO PROFESSOR
	Objetivo e motivação	1 h			
	Introdução	1 h			
	Fundamentos das Técnicas de Integração	2 h			
	Plano de Desenvolvimento	5 h			
	Bibliografia	1 h			
	Projeto	5 h			

(1) Horas atribuídas de acordo com o regulamento das Atividades Práticas Supervisionadas do curso.

TOTAL DE HORAS ATRIBUÍDAS: 75 horas

AVALIAÇÃO: _____
Aprovado ou Reprovado

NOTA: _____

DATA: ____/____/_____

_____
CARIMBO E ASSINATURA DO COORDENADOR DO CURSO