

**UNIVERSIDADE PAULISTA
CIÊNCIA DA COMPUTAÇÃO**

**BRUNO AMADOR N416FF-7
GUILHERME GOMES N444BJ-4
THIAGO RAMOS N4893E-8**

PROCESSAMENTO DE IMAGEM E VISÃO COMPUTACIONAL
Atividade Prática Supervisionada

**SANTANA DE PARNAÍBA - SP
2021**

**UNIVERSIDADE PAULISTA
CIÊNCIA DA COMPUTAÇÃO**

**BRUNO AMADOR N416FF-7
GUILHERME GOMES N444BJ-4
THIAGO RAMOS N4893E-8**

PROCESSAMENTO DE IMAGEM E VISÃO COMPUTACIONAL
Atividade Prática Supervisionada

Trabalho solicitado pelo professor Hugo Insua da
matéria de Processamento de imagem e visão
computacional do curso de Ciências da
Computação da Universidade Paulista.

**SANTANA DE PARNAÍBA - SP
2021**

SUMÁRIO

1.	OBJETIVO	3
2.	INTRODUÇÃO	4
3.	FUNDAMENTOS DAS PRINCIPAIS TÉCNICAS BIOMÉTRICAS.....	5
3.1	Detecção Facial	5
3.2	Reconhecimento Facial	5
3.3	Algoritmo: FisherFaces	6
3.4	Linear Discriminant Analysis (LDA).....	7
4.	PLANO DE DESENVOLVIMENTO	8
4.1	Numpy	8
4.2	OpenCV	8
4.3	mysql-python-connector	8
4.4	Cadastro.....	9
4.5	Login.....	11
4.6	Report.....	12
5.	PROJETO	16
6.	RELATÓRIO COM AS LINHAS DE CÓDIGO.....	19
7.	SISTEMA EM FUNCIONAMENTO.....	25
8.	BIBLIOGRAFIA.....	32
9.	FICHA DE ATIVIDADES PRÁTICAS SUPERVISIONADAS	33

1. OBJETIVO

O sistema Agrosafe tem como objetivo servir como ferramenta de cadastro e manipulação de documentos de “*reports*”, esses reports são nada mais nada menos que um documento que contém as informações sobre eventos, tais como empresa, agrotóxico e escala de toxicidade. O sistema funciona através do terminal a partir do momento que executamos o script “*main.py*” o sistema irá ser inicializado, trazendo as opções de login e cadastro de funcionário.

Conforme a necessidade de classificação dos agrotóxicos, utilizaremos a tabela de classificação da *Anvisa* para medir o grau de toxicidade de cada substância. Segue a tabela a abaixo:

Figura 1 – Tabela de Classificação

	CATEGORIA 1	CATEGORIA 2	CATEGORIA 3	CATEGORIA 4	CATEGORIA 5	NÃO CLASSIFICADO
	EXTREMAMENTE TÓXICO	ALTAMENTE TÓXICO	MODERADAMENTE TÓXICO	POUCO TÓXICO	IMPROVÁVEL CAUSAR DANO AGUDO	NÃO CLASSIFICADO
PICTOGRAMA					Sem símbolo	Sem símbolo
PALAVRA DE ADVERTÊNCIA	PERIGO	PERIGO	PERIGO	CUIDADO	CUIDADO	Sem advertência
CLASSE DE PERIGO						
ORAL	Fatal se ingerido	Fatal se ingerido	Tóxico se ingerido	Nocivo se ingerido	Pode ser perigoso se ingerido	-
DÉRMICA	Fatal em contato com a pele	Fatal em contato com a pele	Tóxico em contato com a pele	Nocivo em contato com a pele	Pode ser perigoso em contato com a pele	-
INALATÓRIA	Fatal se inalado	Fatal se inalado	Tóxico se inalado	Nocivo se inalado	Pode ser perigoso se inalado	-

Fonte: <https://www.inca.gov.br/exposicao-no-trabalho-e-no-ambiente/agrotoxicos>

Além do nível de toxicidade o usuário também deverá informar os nomes e a quantidade de agrotóxico para um melhor detalhamento no report.

O gerenciamento de reports é feito a partir do nível do usuário autenticado, os níveis são de 1 a 3, onde o nível 1 pode apenas acessar informações básicas como título do report, nome da empresa e nome do agrotóxico utilizado, conforme maior o nível do usuário mais informações do report e mais interação ele vai conseguir extrair.

2. INTRODUÇÃO

Reconhecimento facial é um método biométrico, sendo conceito bastante aplicado em áreas como segurança pública, educação e saúde. A biometria é uma tecnologia que utiliza a comparação e a checagem de uma característica humana, com o objetivo de liberar acessos a um indivíduo que contenha as características armazenadas.

Com isso, o nosso sistema se baseia em utilizar de reconhecimento facial para realizar a liberação de um usuário devidamente cadastrado para o acesso as informações sobre agrotóxicos, separados por níveis de 1 a 3.

3. FUNDAMENTOS DAS PRINCIPAIS TÉCNICAS BIOMÉTRICAS

Para a autenticação utilizaremos o recurso de reconhecimento facial como técnica de autenticação, para utilizá-lo é necessário antes entender a diferença entre reconhecimento facial e detecção facial e quais são suas funções.

3.1 Detecção Facial

A detecção facial tem como principal objetivo identificar faces em uma foto ou imagem, sendo assim é feita uma busca de padrões que mais se encaixam com os padrões de faces na imagem. A detecção facial tem apenas como objetivo identificar e contabilizar faces em uma determinada imagem, sem funcionalidades para autenticação, mas ela se faz necessária para realizarmos o ato da autenticação.

3.2 Reconhecimento Facial

O reconhecimento facial trabalha através de uma inteligência artificial para realizar ações de captura, treinamento e reconhecimento, essas são as 3 etapas do reconhecimento facial onde cada uma delas trabalha de maneira diferente de algoritmo para algoritmo.

Para trabalharmos com o reconhecimento facial é necessário identificarmos as faces através da detecção facial, onde será feito cortes na imagem gerando uma nova apenas com o rosto, esse corte é necessário para que o algoritmo consiga realizar os três processos de maneira efetiva.

Tendo as faces localizadas e recortadas pelo detector facial podemos dar início na primeira etapa do reconhecimento facial que é a captura de faces.

- **Captura de faces**

No processo de captura de faces é realizado a coleta das fotos que irão servir como material de treinamento no próximo processo. A captura de faces precisa ser mais variada possível para que o reconhecedor tenha um material diverso para o treinamento. Durante o processo de captura o sistema irá recolher ao total de 25 fotos de amostra que é o número recomendado para o algoritmo que iremos utilizar, o máximo de fotos suportado pelo mesmo é um total de 50 fotos, após isso o algoritmo de treinamento irá ignorar as fotos restantes.

- **Treinamento de imagens**

Na etapa de treinamento é onde é feito a extração de características das fotos de amostras que foram coletadas na etapa de captura, as características são definidas de maneira diferentes de algoritmo para algoritmos, mas as características em geral são as características da face.

- **Reconhecimento**

O algoritmo que iremos utilizar para realizar o treinamento e reconhecimento é o FisherFaces

3.3 Algoritmo: FisherFaces

O FisherFaces são variações de aparência na imagem de cada pessoa, observando variações de luz, poses e expressões faciais, no FisherFace os vetores e características possuem um valor para cada. Tendo então projeções bem distantes determinada pela diferença de uma pessoa para outra.

O algoritmo é caracterizado por 8 etapas, sendo elas:

- I. **Cálculo da face média por classe:** É o resultado da soma pixel a pixel de todas as imagens da mesma classe da coleção de treinamento, dividido pelo total de imagens da classe.
- II. **Cálculo de face média geral:** A face média tem como resultado da soma pixel a pixel de todas as imagens da coleção de treinamento dividido pelo total de imagens.
- III. **Transformação das imagens em vetores:** Consiste em concatenar as linhas da imagem, unindo o último pixel de cada linha com o primeiro pixel da linha seguinte.
- IV. **Construção da matriz de dispersão intra-classe:** Calcula o quanto os dados estão espalhados dentro da mesma classe, em resumo, a quantidade de imagens de um mesmo indivíduo diferem umas das outras.
- V. **Construção da matriz de dispersão inter-classe:** Calcula o quanto os dados estão dispersos entre as classes.
- VI. **Cálculo das FisherFaces:** Seus valores associados das FisherFaces do LDA, são respectivamente, os autovetores e autovalores.
- VII. **Cálculo dos vetores de características:** As imagens da face são atribuídas a partir pela soma ponderada das FisherFaces. Os pesos das

combinações foram os vetores de características das imagens de face.

Para calcular o peso na representação de uma imagem de face, multiplica a imagem de face, subtraída da média, para cada FisherFace.

- VIII. **Cálculo da Similaridade:** O reconhecimento pode ser dar pelo uso de um classificador ou de uma métrica de similaridade.

3.4 Linear Discriminant Analysis (LDA)

A Linear Discriminant Analysis em português Análise Discriminante Linear é uma ferramenta utilizada para classificar, reduzir dimensões e visualizar dados, com o objetivo de remover características redundantes e dependentes ao modificar características de um espaço dimensional maior para um menor.

4. PLANO DE DESENVOLVIMENTO

Para o desenvolvimento do projeto escolhemos utilizar o Python como linguagem de programação, por conta de ser uma linguagem simples e direta e ter uma variedade grande de bibliotecas, ele foi a escolha para o nosso. No planejamento do sistema foram identificadas 3 bibliotecas para auxiliar no desenvolvimento, são elas:

4.1 Numpy

Biblioteca responsável por fazer o processamento de grandes de arranjos e matrizes.

4.2 OpenCV

Biblioteca responsável por fazer o processamento de imagem, detecção e reconhecimento facial.

4.3 mysql-python-connector

Biblioteca responsável por realizar a conexão com o banco de dados e fazer a execução de queries.

Após instalar as bibliotecas utilizando o comando “pip install” e as importando nas nossas classes, conseguimos ter acesso aos seus recursos, porém conforme as boas práticas do Python, para auxiliar na execução do sistema em diferentes máquinas, foi criado um *virtual environment* onde é necessário apenas rodar o script de ativação do mesmo para ter acesso as bibliotecas do projeto e suas versões.

Para realizar a ativação do *virtual environment* é necessário ter acesso a um terminal e dentro da pasta do sistema acessar o seguinte caminho: `venv\Scripts\` e rodar o script de ativação no caso se estiver uma máquina Windows e utilizando o CMD como terminal o script ao ser executado é o `activate.bat`, caso esteja utilizando o Windows Power Shell como terminal temos o script `activate.ps1`. Assim que estiver ativo a tag (`venv`) irá constar no seu terminal.

Figura 2 - Mostrando Dependencias do Projeto

```
C:\Users\Thiago\Documents\Projetos\Python\aps-2021-2 (main -> origin)
λ cd venv\Scripts\

C:\Users\Thiago\Documents\Projetos\Python\aps-2021-2\venv\Scripts (main -> origin)
λ activate.bat

C:\Users\Thiago\Documents\Projetos\Python\aps-2021-2\venv\Scripts (main -> origin)
(venv) λ pip freeze
mysql-connector-python==8.0.27
numpy==1.21.4
opencv-contrib-python==4.5.4.58
```

Fonte: Thiago Ramos

Assim que executarmos o script *main.py* o sistema irá ser inicializado na tela funcionários, onde o usuário deverá selecionar entre fazer o cadastro de um novo funcionário ou realizar o login.

Figura 3 - Tela Inicial do Programa

```

C:\Users\Thiago\Documents\Projetos\Python\aps-2021-2 (main -> origin)
(venv) λ python main.py
=====
===== AgroSafe =====
=====
...
...
[INFO] --- [1] = Login
[INFO] --- [2] = Cadastro
...
[INPUT] --- Selecionar acao: |
  
```

Fonte: Thiago Ramos

4.4 Cadastro

Na ação do cadastro é a etapa onde o usuário irá realizar o cadastro de um novo funcionário a partir da captura de fotos da amostra em conjunto com treinamento do algoritmo no OpenCV.

Os campos que o usuário deverá preencher na etapa de cadastro de funcionário são os campos de:

- CPF: CPF único para cada funcionário, é feito uma validação assim que usuário inserir o CPF, uma consulta será feita no banco de dados com o CPF como parâmetro de busca, caso já exista algum funcionário com este CPF o sistema informará CPF em uso e reiniciará a ação de Cadastro.
- Nome: Nome do funcionário a ser registrado.
- Número: Número do funcionário a ser registrado.
- E-mail: E-mail do funcionário a ser registrado, é feito uma validação de e-mail, onde caso ele não tenha a estrutura passada na expressão regular(regex): `^[^\\w-\\.]+@([\\w-]+\\.)+[\\w-]{2,4}$` o sistema informará que o e-mail está em formato incorreto e reiniciará a ação de cadastro.
- Cargo: Cargo do funcionário a ser registrado.
- Nível: Informará o nível do funcionário, onde ele é de 1 a 3, caso seja um valor diferente destes será informado que o nível é inválido e reiniciará a ação de cadastro.

Após realizar as validações e montar o objeto do funcionário, nomeado no sistema como `FuncionarioModel`, o sistema irá salvar o usuário utilizando a classe `FuncionarioRepository` que é responsável por realizar a interação com o banco.

Com o funcionário novo cadastrado no sistema, iremos utilizar o id que foi gerado pelo banco de dados ao realizar a inserção para passar como id das fotos de captura de fotos.

Na etapa de captura uma tela com a webcam será aberta, onde iremos realizar a captura das imagens de teste, para realizar a detecção das faces para realizar os cortes, utilizaremos o classificador *haarcascade_frontalface_default.xml*, onde ele irá passar a localização das faces encontradas nas imagens, a partir disso colocamos um quadrado em torno do rosto detectado, para realizar a captura das fotos basta apertar o botão Q enquanto a marcação do quadrado estiver em torno de seu rosto.

Assim que o botão Q é acionado a face que está no quadrado vai ser recordada da imagem e salva na pasta de fotos com o seguinte formato:

`pessoa.(id_funcionario).(numero_face).jpg`

O `id_funcionario` informado é passado pelo banco de dados assim que um funcionário é salvo, já o `numero_face` é o contador de faces salvas. As faces são salvas em uma escala de cinza. As fotos são salvas com a escala de cinza pois o algoritmo trabalha melhor com imagens neste tipo.

Figura 4 - Algoritmo

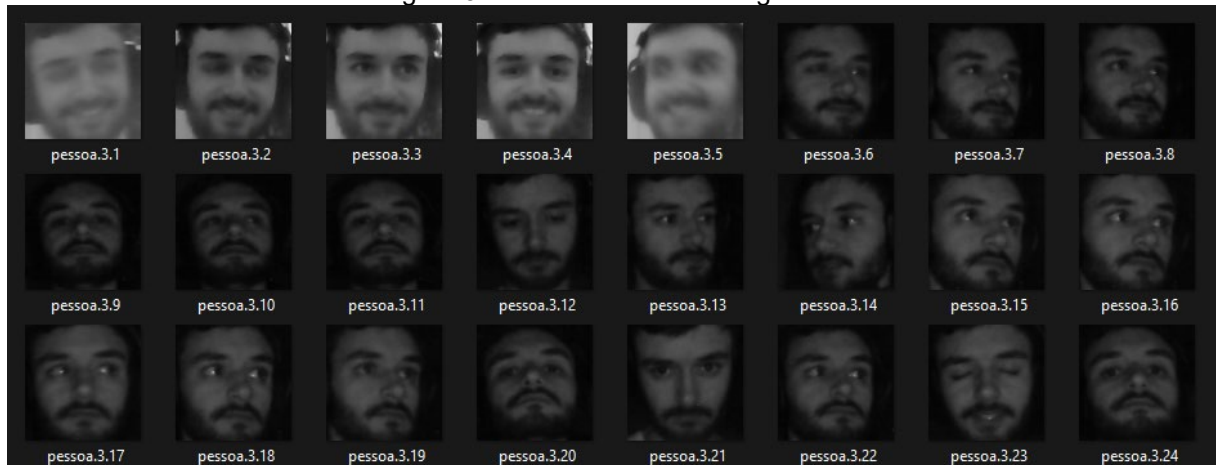
```
[ 94, 94, 95, ..., 189, 188, 186],
[ 95, 95, 97, ..., 190, 189, 188],
[ 95, 96, 98, ..., 189, 189, 189]], dtype=uint8), array([[177, 178, 175, ..., 163, 169, 170],
[177, 178, 175, ..., 164, 170, 171],
[177, 178, 176, ..., 164, 170, 171],
...,
[154, 157, 162, ..., 187, 188, 188],
[167, 166, 166, ..., 187, 188, 189],
[172, 170, 169, ..., 188, 188, 189]], dtype=uint8), array([[178, 178, 178, ..., 155, 158, 161],
[178, 178, 178, ..., 155, 157, 160],
[178, 178, 177, ..., 156, 157, 160],
...,
[171, 170, 171, ..., 185, 187, 188],
[176, 175, 176, ..., 186, 188, 189],
[178, 177, 178, ..., 187, 188, 190]], dtype=uint8), array([[179, 180, 179, ..., 160, 164, 169],
[178, 179, 178, ..., 159, 164, 169],
[179, 180, 179, ..., 158, 163, 169],
...,
```

Fonte: Thiago Ramos

Assim que as 25 fotos forem capturadas o sistema irá dar início ao treinamento do algoritmo FisherFaces, onde será percorrido toda as pastas de fotos e será gerado o arquivo *classificadorFisher.yml*, que é responsável por fazer a detecção facial. A

relação feita é entre faces e o id do funcionário. Após o processo de treinamento o usuário será encaminhado para a tela de Reports.

Figura 5 - Fotos Treinamento Algoritmo



Fonte: Bruno Amador

4.5 Login

Na ação de login é feita a autenticação do sistema e onde será feito a autenticação por biometria facial, nesta tela o usuário deverá informar o CPF a ser consultado onde será feita uma consulta pela entidade de funcionário pelo CPF, caso ele não esteja cadastrado é dada as opções de tentar novamente e realizar o cadastro de funcionário, ao selecionar esta opção o funcionário será direcionado para a tela de cadastro.

Caso o CPF do funcionário esteja cadastrado, o processo de autenticação facial irá ser iniciado com o objeto modelo do funcionário, a tela com a webcam irá ser aberta novamente e caso o id do rosto identificado pelo algoritmo o nome do funcionário irá aparecer abaixo do quadrado de seu rosto, caso contrário nenhum nome será exibido. Ao apertar o botão Q irá ser feita uma validação no nome que é passado no quadro, caso o nome seja igual o nome do funcionário em questão o sistema irá autenticar o usuário e ele irá ser redirecionado para a tela de report. Caso o usuário aperte o botão Q e o nome abaixo do quadrado não seja igual o do funcionário, então a autenticação irá falhar, o usuário poderá escolher entre sair do sistema e tentar novamente. Caso o usuário seja autenticado ele será redirecionado para a tela de reports.

4.6 Report

Ao ser redirecionado para a tela de Reports, a ação de report será inicializada, dando assim as seguintes opções de ações para o usuário:

Figura 6 - Menu de Reports

```

=====
==== Central de Reports: .....
=====
...
[INFO] --- Selecione [0] para listar todos os Reports
[INFO] --- Selecione [1] para cadastrar um novo Report
[INFO] --- Selecione [2] para gerar um relatório de um Report
[INFO] --- Selecione [3] para gerar relatório massivo
[INFO] --- Selecione [4] para gerar relatório massivo por filtro
[INFO] --- Selecione outro botão para sair do sistema
...
[INPUT] --- Selecionar valor:

```

Fonte: Thiago Ramos

- Listar todos os Reports: Lista os campos de id, nome da empresa e nome do agrotóxico de todos os reports.
- Cadastrar um novo Report: Na tela de cadastro de reports o funcionário com o nível maior do que 1 poderá cadastrar um novo Report. Ao selecionar essa opção a ação de cadastro de Report irá ser iniciada, o usuário deverá informar os seguintes campos:
 - Nome da empresa: Nome da empresa a ser reportada.
 - Número da empresa: Número da empresa a ser reportada.
 - Endereço do local: Local do report.
 - Dono da empresa: Dono da empresa a ser reportada.
 - Gravidade: Número inteiro numa escala de 1 a 5.
 - Nome Agro: Nome do agrotóxico.
 - Tipo Agro: Nível da toxicidade do agrotóxico numa escada de 1 a 5 conforme informado acima. O tipo de agrotóxico consta na tabela de domínio do agrotóxico.
 - Quantidade de Agro: Quantidade de agrotóxico identificado.

Figura 7 - Gerador de Relatório

```

=====
[Agrosafe] - Relatório: {ID_RELATÓRIO}
Gerado por: {FUNC_NOME}
=====
...
[INFO] ---> Gerais:
=Nome empresa: {NOME_EMPRE}
=Numero empresa: {NUMERO_EMPRE}
=Local afetado: {ENDERECO}
=Dono da empresa: {DONO_EMPRESA}
...
[INFO] ---> Agrotóxico:
=Nome agrotóxico: {NOME_AGRO}
=Tipo agrotóxico: {CD_AGRO}
=Descrição do agrotóxico: {DS_AGRO}
=Efeito oral: {EFEITO_ORAL}
=Efeito dermico: {EFEITO_DERMICO}
=Efeito ao inalar: {EFEITO_INALAR}
=Quantidade de agrotóxico: {QT_AGRO}
...
[INFO] ---> Impacto
=Tipo de dano: {TIPO_DANO}
=Gravidade: {GRAVIDADE}
=====

```

Fonte: Thiago Ramos

- Gerar relatório unitário: Ao selecionar a opção de gerar relatório unitário, será requisitado o id do report, ao realizar uma consulta pelo ID passado pelo usuário será montado a partir do arquivo *REPORT_UNITARIO.txt* o relatório, o arquivo de template irá ser lido pelo sistema, que irá substituir as tags do template pelos seus respectivos campos.

As informações exportadas serão de acordo com o nível do funcionário, onde conforme maior o nível do funcionário, mais informações ele conseguirá exportar. Os níveis disponíveis são armazenados em na tabela de domínio de nível. Caso o usuário não tenha permissão a frase “[INFO] - SEM PERMISSÃO” será exportada no lugar do campo. Os campos exportados por nível são:

- Nível 1:
 - Nome da Empresa
 - Número da empresa
 - Local Afetado
 - Nome Agrotóxico
- Nível 2:
 - Nome da Empresa
 - Número da empresa
 - Local Afetado

- Nome Agrotóxico
- Tipo Agrotóxico
- Descrição Agrotóxico
- Efeito Oral
- Efeito Dérmico
- Efeito ao Inalar

○ Nível 3:

- Nome da Empresa
- Número da empresa
- Dono da empresa
- Local Afetado
- Gravidade
- Nome Agrotóxico
- Tipo Agrotóxico
- Descrição Agrotóxico
- Efeito Oral
- Efeito Dérmico
- Efeito ao Inalar
- Quantidade Agrotóxico

Figura 8 - Gerador de Relatório Massivo

```

===== -> Header
[Agrosafe] - Relatório Massivo
Filtro: {FILTRO}
Gerado por: {FUNC_NOME}
===== -> Body
[INFO] ---> Relatório: {ID_RELATÓRIO}
...
[INFO] ---> Gerais:
=Nome da empresa: {NOME_EMPRE}
=Numero da empresa: {NUMERO_EMPRE}
=Local afetado: {ENDERECO}
=Dono da empresa: {DONO_EMPRESA}
...
[INFO] ---> Agrotóxico:
=Nome do agrotóxico: {NOME_AGRO}
=Tipo do agrotóxico: {CD_AGRO}
=Descrição do agrotóxico: {DS_AGRO}
=Efeito oral: {EFEITO_ORAL}
=Efeito dermico: {EFEITO_DERMICO}
=Efeito ao inalar: {EFEITO_INALAR}
=Quantidade de agrotóxico: {QT_AGRO}
...
[INFO] ---> Impacto
=Tipo de dano: {TIPO_DANO}
=Gravidade: {GRAVIDADE}
=====

```

Fonte: Thiago Ramos

Gerar relatório massivo e gerar relatório massivo com filtro: Seguindo a mesma ideia do report unitário, o sistema irá realizar a leitura do template *REPORT_MASSIVO.txt* para gerar o seu relatório, porém neste template não temos um limite de relatórios, caso seja selecionada a opção de *gerar relatório massivo com filtro*, o usuário irá escolher entre fazer o filtro pelo nome da empresa ou pelo nome do agrotóxico, caso seja selecionado a opção de *gerar relatório massivo* todos os reports serão exportados. Por conta deste template ter um cabeçalho, foi dividido entre header e body, onde o header vai ser exportado no relatório apenas uma vez, enquanto o body vai ser montado de acordo com a quantidade de reports. A exportação de campos por nível segue a mesma estrutura que o report unitário.

5. PROJETO

A estruturação do projeto foi separada em duas etapas, a etapa de estruturação do banco de dados e estruturação do projeto. Para a criação do banco de dados, utilizaremos o arquivo *SCRIPT_MONTA_BASE.sql* que contém todas as queries de criação e de população de tabelas. A linguagem de banco de dados que foi selecionada para o uso foi o MySQL.

Utilizaremos a database aps_agrosafe conforme podemos ver no script e suas tabelas irão seguir o padrão de nomenclatura para identificar melhor as tabelas e suas funcionalidades, o padrão e seus componentes são:

- TAPS_(TIPO_TABELA)_(NOME_TABELA)
 - TAPS: Resumo de Tabela APS, vai ser inserida no início de todas as tabelas criadas no database agrosafe.
 - TIPO_TABELA: Vai ser um identificador do tipo da tabela, no projeto temos dois tipos de tabela, elas são:
 - Tabelas de domínio: Identificadas pelo nome de “DOM” são tabelas estáticas que não serão alteradas pelo usuário.
 - Tabelas quentes: Identificadas pelo nome de “HOT” são tabelas dinâmicas que irão armazenar os dados que o usuário irá inserir.
 - NOME_TABELA: Vai ser o nome da entidade da tabela.

As tabelas seguem um padrão, partindo para o detalhamento e função das tabelas do projeto, temos as seguintes tabelas e suas funcionalidades e campos:

- Tabelas de Domínio:
 - TAPS_DOM_NIVEL: É a tabela responsável por guardar os níveis válidos no sistema. Tem o campo CD_NIVEL como chave primaria.
 - TAPS_DOM_AGRO: É a tabela responsável por guardar os tipos de agrotóxico e suas toxicidades. Tem o campo CD_AGRO como chave primaria.

- Tabelas quente:
 - TAPS_HOT_FUNC: É a tabela responsável por armazenar os dados dos funcionários cadastrados no sistema. Tem o campo CD_FUNC como chave primária e o campo CD_NIVEL como chave estrangeira da tabela TAPS_DOM_NIVEL.
 - TAPS_HOT_REPORT: É a tabela responsável por armazenar os dados dos reports cadastrados no sistema. Tem o campo CD_REPORT como chave primária e o campo CD_AGRO como chave estrangeira da tabela TAPS_DOM_AGRO.

Para a estruturação do projeto se foi utilizado o paradigma de linguagem de programação orientada a objetos, por conta disso cada objeto tem suas classes e suas funções. A separação de pastas é feita inicialmente em 3 pastas e uma classe principal nomeada como *main.py*, as três pastas são:

rsc: É a pasta com a função para armazenar os resources do projeto.

src: É a pasta com a função de armazenar o código do projeto.

venv: É a pasta responsável por armazenar o *virtual environment*.

A pasta que armazena os recursos do projeto armazena os templates do projeto, a pasta de fotos, o script de montagem de base e os classificadores utilizados no projeto.

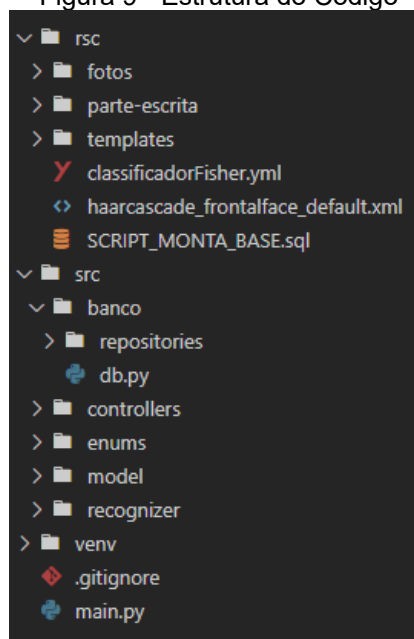
A pasta de rsc abriga 5 subpastas que tem como função fazer a separação e classificação de cada tipo de fase, elas são:

- Banco: É a subpasta que tem como objetivo fazer a conexão com o banco de dados utilizando a biblioteca mysql-python-connector, para isso é feita uma conexão com o banco de dados a partir da classe Banco no arquivo *db.py*, que tem como função estabelecer a conexão para as classes do tipo *Repositories*. Também temos a subpasta repositories que armazena as classes que fazem interação com o banco de dados de cada um dos objetos. A conexão é estabelecida na classe *db.py* e é herdada nas classes *repositories*.
- Controllers: A subpasta de controllers é responsável por armazenar as classes controladoras, responsáveis pelas regras de negócio e consumir as

demais classes. Dentro desta pasta temos as classes responsáveis pelas ações de login, cadastro e reports.

- Enums: A classe de Enums foi feita para manter as boas práticas e não manter campos hardcodes no código. As classes de Enums são classes estáticas que serão acessadas somente para ter acesso a um determinado valor, no projeto cada um dos objetos tem sua classe Enum para ter acesso ao nome das colunas no banco de dados, ou ter acesso as tags do template.
- Model: Está subpasta é responsável por armazenar classes do tipo Model, que tem como principal função servir como abstração dos objetos. As classes de modelo precisam ter os mesmos campos que sua respectiva tabela do banco de dados para que seus dados sejam manipulados de maneira correta. Cada um dos objetos tem sua classe modelo.
- Recognizer: Está subpasta armazena as classes responsáveis por realizar as funções de reconhecimento facial, nela estão as classes responsáveis por realizar a captura de imagens que salva as imagens das faces no caminho rsc/fotos, a classe de treinamento e de reconhecer, que acessa a pasta de resources para ter acesso aos classificadores.

Figura 9 - Estrutura do Código



Fonte: Thiago Ramos

6. RELATÓRIO COM AS LINHAS DE CÓDIGO

Figura 10 - Código SCRIPT_MONTA_BASE_1

```

rsc> SCRIPT_MONTA_BASE_1.sql
1  # APS 2021 - 2º Semestre
2  # -> Banco de dados: Thiago Ramos de Oliveira
3
4  #create database
5  create database aps_agrosafe;
6  use aps_agrosafe;
7
8  #create dom tables
9  create table TAPS_DOM_NIVEL(
10     CD_NIVEL int primary key not null,
11     DS_NIVEL varchar(20) not null
12 );
13
14 create table TAPS_DOM_AGRO(
15     CD_AGRO int auto_increment not null,
16     CATEGORIA varchar(20) not null,
17     DS_AGRO varchar(30) not null,
18     EFEITO_ORAL varchar(30) not null,
19     EFEITO_DERMICO varchar(40) not null,
20     EFEITO_INALAR varchar(30) not null,
21
22     primary key(CD_AGRO)
23 );
24
25 #create hot tables
26 create table TAPS_HOT_FUNC(
27     CD_FUNC int auto_increment not null,
28     CPF_FUNC varchar(20) not null,
29     NOME varchar(50) not null,
30     NUMERO varchar(15) not null,
31     EMAIL varchar(30) not null,
32     CARGO varchar(15) not null,
33     CD_NIVEL int not null,
34
35     primary key (CD_FUNC),
36     foreign key (CD_NIVEL) references TAPS_DOM_NIVEL(CD_NIVEL)
37 );
38
39 create table TAPS_HOT_REPORT(
40     CD_REPORT int auto_increment not null,
41     NOME_EMPRE varchar(30) not null,
42     ENDERECO_EMPRE varchar(60) not null,
43     NUMERO_EMPRE varchar(15) not null,
44     DONO_EMPRE varchar(50) not null,
45     GRAVIDADE int not null,
46     TIPO_DANO varchar(15) not null,
47     CD_AGRO int not null,
48     QT_AGRO int not null,
49     NOME_AGRO varchar(15) not null,

```

Fonte: Bruno Amador

Figura 11 - Código SCRIPT_MONTA_BASE_1

```

rsc> SCRIPT_MONTA_BASE_1.sql
50     primary key(CD_REPORT),
51     foreign key(CD_AGRO) references TAPS_DOM_AGRO(CD_AGRO)
52 );
53
54 #popular tabelas dom
55 # Taps_dom_nivel
56 INSERT INTO TAPS_DOM_NIVEL(CD_NIVEL, DS_NIVEL)
57 VALUES(1, "Nivel 1");
58 INSERT INTO TAPS_DOM_NIVEL(CD_NIVEL, DS_NIVEL)
59 VALUES(2, "Nivel 2");
60 INSERT INTO TAPS_DOM_NIVEL(CD_NIVEL, DS_NIVEL)
61 VALUES(3, "Nivel 3");
62
63 # Taps_dom_agro
64 INSERT INTO TAPS_DOM_AGRO(CATEGORIA, DS_AGRO, EFEITO_ORAL, EFEITO_DERMICO, EFEITO_INALAR)
65 VALUES("CATEGORIA 1", "Extremamente toxico", "Fatal se ingerido", "Fatal em contato com a pele", "Fatal se inalado");
66
67 INSERT INTO TAPS_DOM_AGRO(CATEGORIA, DS_AGRO, EFEITO_ORAL, EFEITO_DERMICO, EFEITO_INALAR)
68 VALUES("CATEGORIA 2", "Altamente toxico", "Fatal se ingerido", "Fatal em contato com a pele", "Fatal se inalado");
69
70 INSERT INTO TAPS_DOM_AGRO(CATEGORIA, DS_AGRO, EFEITO_ORAL, EFEITO_DERMICO, EFEITO_INALAR)
71 VALUES("CATEGORIA 3", "Moderadamente toxico", "Toxico se ingerido", "Toxico em contato com a pele", "Toxico se inalado");
72
73 INSERT INTO TAPS_DOM_AGRO(CATEGORIA, DS_AGRO, EFEITO_ORAL, EFEITO_DERMICO, EFEITO_INALAR)
74 VALUES("CATEGORIA 4", "Pouco toxico", "Nocivo se ingerido", "Nocivo em contato com a pele", "Nocivo se inalado");
75
76 INSERT INTO TAPS_DOM_AGRO(CATEGORIA, DS_AGRO, EFEITO_ORAL, EFEITO_DERMICO, EFEITO_INALAR)
77 VALUES("CATEGORIA 5", "Improvavel causar dano agudo", "Pode ser perigoso se ingerido", "Pode ser perigoso em contato com a pele",
78 "Pode ser perigoso se inalado");
79
80 #popular tabelas quentes
81 # Taps_hot_report
82 INSERT INTO TAPS_HOT_REPORT(NOME_EMPRE, ENDERECO_EMPRE, NUMERO_EMPRE,
83     DONO_EMPRE, GRAVIDADE, TIPO_DANO, CD_AGRO, QT_AGRO, NOME_AGRO)
84 VALUES("Jacollaria Fachada", "Endereco 1, numero 1 - SP", "11111111", "Guilherme Gomes David", 1, "Solo",
85     (SELECT CD_AGRO FROM TAPS_DOM_AGRO WHERE CATEGORIA = "CATEGORIA 1"), 1, "Agrotox de solo");
86
87 INSERT INTO TAPS_HOT_REPORT(NOME_EMPRE, ENDERECO_EMPRE, NUMERO_EMPRE,
88     DONO_EMPRE, GRAVIDADE, TIPO_DANO, CD_AGRO, QT_AGRO)
89 VALUES("Oficina Fachada", "Endereco 2, numero 2 - SP", "22222222", "Bruno Amador", 2, "Mar",
90     (SELECT CD_AGRO FROM TAPS_DOM_AGRO WHERE CATEGORIA = "CATEGORIA 2"), 5, "Super Tox");
91
92 INSERT INTO TAPS_HOT_REPORT(NOME_EMPRE, ENDERECO_EMPRE, NUMERO_EMPRE,
93     DONO_EMPRE, GRAVIDADE, TIPO_DANO, CD_AGRO, QT_AGRO)
94 VALUES("Cafeteria Fachada", "Endereco 3, numero 3 - SP", "33333333", "Thiago Ramos de Oliveira", 3, "Fuma",
95     (SELECT CD_AGRO FROM TAPS_DOM_AGRO WHERE CATEGORIA = "CATEGORIA 3"), 8, "Hiper tox");
96

```

Fonte: Bruno Amador

Figura 12 - Código Cadastro

```

src > controllers > cadastro.py > ...
1  import os
2  projectPath = os.path.abspath('main.py').replace('\\main.py', '')
3  import sys
4  sys.path.append(projectPath)
5
6  from src.model.FuncionarioDomModel import FuncionarioDomModel
7  from src.banco.repositories.FuncionarioDomRepository import FuncionarioRepository
8  from src.recognizer.captura import Captura
9  from src.recognizer.treinamento import Treinamento
10 from src.enums.FuncionarioDomEnum import FUNCIONARIO_ENUM
11 from src.controllers.reports import ReportController
12
13 class Cadastro():
14
15     def testeAction(self, ):
16         print(FUNCIONARIO_ENUM.CPF.value)
17
18     def __init__(self, ):
19         self._repository = FuncionarioRepository()
20         self._captura = Captura()
21         self._treinamento = Treinamento()
22         self._reportController = ReportController()
23
24     # -> Realiza a ação de cadastro
25     def cadastroAction(self, ):
26         print('=====')
27         print('=== CADASTRO DE FUNCIONARIO: =====')
28         print('=====')
29         print('.')
30         cpf = str(input('[INPUT]...CPF: '))
31         nome = str(input('[INPUT]...Nome: '))
32         numero = str(input('[INPUT]...Numero: '))
33         email = str(input('[INPUT]...E-mail: '))
34         cargo = str(input('[INPUT]...Cargo: '))
35         nivel = int(input('[INPUT]...Nivel: '))
36
37         model = FuncionarioDomModel()
38
39         model.setCpf(cpf)
40         model.setNome(nome)
41         model.setNumero(numero)
42         model.setEmail(email)
43         model.setCargo(cargo)
44         model.setIdNivel(nivel)
45
46         self._repository.save(model)
47
48     # -> Inicio de Captura e Treinamento Facial
49     print('[INFO] --- Iniciando captura de faces.')
```

Fonte: Bruno Amador

Figura 13 - Código Cadastro

```

50     print('[INFO] --- Favor apertar "Q" para captura. Total de fotos a ser tiradas: 25')
51
52     # -> Necessário realizar consulta pois o ID é gerado pelo banco
53     savedModel:FuncionarioDomModel = self._repository.findByCampo(FUNCIONARIO_ENUM.CPF.value, cpf)
54
55     self._captura.capturaCadastro(savedModel.getId())
56     print('[INFO] --- Captura de faces concluída!')
57
58     print('[INFO] --- Iniciando treinamento de faces.')
59     self._treinamento.treinarFisherFace()
60     print('[INFO] --- Treinamento de faces concluída!')
61
62     print('[SISTEMA] --- Abrindo tela de Reports... ')
63     self._reportController.getReportAction(savedModel)
64
```

Fonte: Bruno Amador

Figura 14 - Código Login

```

src > controllers > login.py > Login > loginAction
1 import os
2 projectPath = os.path.abspath('main.py').replace('main.py', '')
3 import sys
4 sys.path.append(projectPath)
5
6 from src.controllers.cadastro import Cadastro
7 from src.banco.repositorias.FuncionarioDomRepository import FuncionarioRepository as repository
8 from src.model.FuncionarioDomModel import FuncionarioDomModel
9 from src.enums.FuncionarioDomEnum import FUNCIONARIO_ENUM
10 from src.recognizer.regonizer import Recognizer
11 from src.controllers.reports import ReportController
12
13 class Login:
14
15     def __init__(self, ):
16         self._repository = repository()
17         self._cpfList = []
18         self._cadastro = Cadastro()
19         self._recognizer = Recognizer()
20         self._reportController = ReportController()
21         listFunc = self._repository.findAll()
22
23         for func in listFunc:
24             self._cpfList.append(func.getCpf())
25
26     # -> Realiza a ação do login
27     def loginAction(self, ):
28         print('=====')
29         print('==== LOGIN DE FUNCIONARIO: =====')
30         print('=====')
31         print('...')
32         cpf = str(input('[INPUT] --- CPF: '))
33
34         if(cpf in self._cpfList):
35
36             func:FuncionarioDomModel = self._repository.findByCampo(FUNCIONARIO_ENUM.CPF.value, cpf)
37             print('[SISTEMA] --- Iniciado reconhecimento no funcionário: ' + func.getNome())
38             #isAutenticado, confianca = self._recognizer.getAutenticacao(func = func)
39             isAutenticado, confianca = True, 0
40             if(isAutenticado == True):
41                 print('[SISTEMA] --- Bem vindo {}'.format(func.getNome()))
42                 print('[SISTEMA] --- Usuário autenticado com uma confiança de: {}'.format(str(confianca)))
43
44                 # -> Action de reports
45                 self._reportController.getReportAction(func)
46
47             else:
48
49                 while(True):

```

Fonte: Bruno Amador

Figura 15 - Código Login

```

50 print('[SISTEMA] --- A autenticação facial falhou. Deseja tentar novamente ?')
51 selectValue = input('[SISTEMA] --- Informe [1] para tentar novamente e [2] para sair do sistema')
52
53 if(int(selectValue) == 1):
54     print('[SISTEMA] --- Iniciado reconhecimento no funcionário: ' + func.getNome())
55     isAutenticado, confianca = self._recognizer.getAutenticacao(func = func)
56
57     if(isAutenticado == True):
58         print('[SISTEMA] --- Bem vindo {}'.format(func.getNome()))
59         print('[SISTEMA] --- Usuário autenticado com uma confiança de: {}'.format(str(confianca)))
60
61         # -> Action de Reports
62         self._reportController.getReportAction(func)
63
64     elif(int(selectValue) == 2):
65         print('[INFO] --- Fechando sistema...')
66         exit
67     else:
68         print('[ERRO] --- Comando invalido. Fechando sistema...')
69         exit
70
71 else:
72     print('[ERRO] --- CPF não cadastrado ')
73     value = int(input('[SISTEMA] --- Aperte [1] para cadastrar e [2] para sair do sistema: '))
74
75     if(value == 1):
76         self._cadastro.cadastroAction()
77     else:
78         exit

```

Fonte: Bruno Amador

Figura 16 - Código ReportAction

```

src > controllers > reports.py > ReportController > getReportAction
1 import os
2 projectPath = os.path.abspath('main.py').replace('main.py', '')
3 import sys
4 sys.path.append(projectPath)
5
6 from src.enums.AgroToxicoHotEnum import AGROTOX_ENUM
7 from src.model.FuncionarioDomModel import FuncionarioDomModel
8 from src.model.ReportHotModel import ReportHotModel
9 from src.banco.repositories.ReportHotRepository import ReportRepository
10 from src.enums.ReportHotEnum import REPORT_ENUM
11 from src.banco.repositories.AgroToxicoHotRepository import AgroToxRepository
12
13 class ReportController():
14
15     def __init__(self, ):
16         self._repository = ReportRepository()
17         self._agroRepository = AgroToxRepository()
18         home = os.path.expanduser('~')
19         self._documentPath = os.path.join(home, 'Documents')
20
21     def getReportAction(self, func:FuncionarioDomModel):
22         self._func = func
23
24         print('=====')
25         print('=== Central de Reports: =====')
26         print('=====')
27
28         print('...')
29         print('[INFO] --- Selecione [0] para listar todos os Reports')
30         print('[INFO] --- Selecione [1] para cadastrar um novo Report')
31         print('[INFO] --- Selecione [2] para gerar um relatório de um Report')
32         print('[INFO] --- Selecione [3] para gerar export massivo')
33         print('[INFO] --- Selecione [4] para gerar export massivo por filtro')
34         print('[INFO] --- Selecione outro botão para sair do sistema')
35         print('...')
36         selectedValue = input('[INPUT] --- Selecionar valor: ')
37
38         if(int(selectedValue) == 0):
39             self._listReports()
40         elif(int(selectedValue) == 1):
41             self._cadastroReport()
42         elif(int(selectedValue) == 2):
43             self._gerarRelatorio(func.getIdNivel())
44         elif(int(selectedValue) == 3):
45             self._gerarRelatorioMassivo(int(selectedValue))
46         elif(int(selectedValue) == 4):
47             self._gerarRelatorioMassivo(int(selectedValue))
48

```

Fonte: Bruno Amador

Figura 17 - Código CapturaCadastro

```

src > recognizer > captura.py > Captura > capturaCadastro
1 import os
2 projectPath = os.path.abspath('main.py').replace('main.py', '')
3 import sys
4 sys.path.append(projectPath)
5 import cv2 as cv
6
7 class Captura:
8     classificador = cv.CascadeClassifier('rsc/haarcascade_frontalface_default.xml')
9     camera = cv.VideoCapture(0)
10     amostra = 1
11     numeroDeAmostras = 25
12     largura, altura = 220, 220
13
14     # -> Realiza captura e salva as imagens das faces na pasta de fotos
15     def capturaCadastro(self, id):
16         while(True):
17             conectado, imagem = self.camera.read()
18             imagemCinza = cv.cvtColor(imagem, cv.COLOR_BGR2GRAY)
19             facesDetect = self.classificador.detectMultiScale(imagemCinza,
20                                                         scaleFactor = 1.5,
21                                                         minSize = (150, 150))
22             for(x, y, l, a) in facesDetect:
23                 cv.rectangle(imagem, (x, y), (x + l, y + a), (0, 255, 0), 2)
24
25                 if (cv.waitKey(1) & 0xFF == ord('q')):
26                     imagemFace = cv.resize(imagemCinza[y:y + a, x:x + l], (self.largura, self.altura))
27                     imgPath = 'rsc/fotos/pessoa.' + str(id) + '.' + str(self.amostra) + '.jpg'
28                     cv.imwrite(imgPath, imagemFace)
29                     print('[SISTEMA] --- Foto salva no caminho:...' + str(imgPath))
30                     self.amostra += 1
31
32             cv.imshow("Face Capture", imagem)
33             cv.waitKey(1)
34
35             if((self.amostra >= self.numeroDeAmostras) | (cv.waitKey(1) & 0xFF == ord('e'))):
36                 break
37
38         print('[INFO] --- Faces capturadas com sucesso!')
39         self.camera.release()
40         cv.destroyAllWindows()
41         return imgPath
42

```

Fonte: Bruno Amador

Figura 18 - Código getAutenticacao

```

src > recognizer > regonizer.py > Recognizer > getAutenticacao
1 import os
2 projectPath = os.path.abspath('main.py').replace('\\main.py', '')
3 import sys
4 sys.path.append(projectPath)
5 import cv2 as cv
6 from src.model.funcionarioDomModel import FuncionarioDomModel
7
8 class Recognizer:
9     def __init__(self, ):
10         self._camera = cv.VideoCapture(0)
11         self._detectorFace = cv.CascadeClassifier('rsc/haarcascade_frontalface_default.xml')
12         self._reconhecedor = cv.face.FisherFaceRecognizer_create()
13
14
15 # -> Realiza autenticação por facial ao apertar o botão "Q"
16 def getAutenticacao(self, func:FuncionarioDomModel):
17     self._reconhecedor.read(r'rsc/classificadorFisher.yml')
18     largura, altura = 220, 220
19     font = cv.FONT_HERSHEY_DUPLEX
20     nome = ''
21
22     while(True):
23         conectado, imagem = self._camera.read()
24         imagemCinza = cv.cvtColor(imagem, cv.COLOR_BGR2GRAY)
25         facesDetectadas = self._detectorFace.detectMultiScale(imagemCinza,
26                                                                 scaleFactor=1.5,
27                                                                 minSize=(150,150))
28
29         for (x, y, l, a) in facesDetectadas:
30
31             imagemFace = cv.resize(imagemCinza[y: y + a, x: x + l], (largura, altura))
32             cv.rectangle(imagem, (x, y), (x + l, y + a), (0, 255, 0), 2)
33
34             id, confinca = self._reconhecedor.predict(imagemFace)
35
36             if(int(id) == int(func.getId())):
37                 nome = func.getNome()
38
39                 cv.putText(imagem, nome, (x,y + (a + 50)), font, 1, (0, 255, 0))
40
41             if (cv.waitKey(1) & 0xFF == ord('q')):
42                 if(nome == func.getNome()):
43                     return True, confinca
44
45             else:
46                 return False, 0
47
48     cv.imshow("FaceEigen", imagem)
49     cv.waitKey(1)

```

Fonte: Bruno Amador

Figura 19 - Código treinarFisherFace

```

src > recognizer > treinamento.py > Treinamento > treinarFisherFace
1 import os
2 projectPath = os.path.abspath('main.py').replace('\\main.py', '')
3 import sys
4 sys.path.append(projectPath)
5
6 import cv2 as cv
7 import numpy as np
8 import os
9
10 class Treinamento():
11     fisherFace = cv.face.FisherFaceRecognizer_create()
12
13     def teste(self,):
14         print('ok')
15
16     def getImagemComId(self, ):
17         paths = [os.path.join('rsc/fotos', f) for f in os.listdir('rsc/fotos')]
18         faces = []
19         ids = []
20
21         for path in paths:
22             imageFace = cv.cvtColor(cv.imread(path), cv.COLOR_RGB2GRAY)
23             id = int(os.path.splitext(path)[-1].split('.')[1])
24             ids.append(id)
25             faces.append(imageFace)
26
27         return faces, np.array(ids)
28
29 # -> Realiza o treinamento do Algoritmo: FisherFaces
30 def treinarFisherFace(self, ):
31     faces, ids = self.getImagemComId()
32     print('[SISTEMA] --- Iniciando treinamento de algiritmo: [FisherFace]...')
33     self.fisherFace.train(faces, ids)
34     self.fisherFace.write('rsc/classificadorFisher.yml')
35     print('[SISTEMA] --- Finalizado treinamento de algiritmo: [FisherFace]')
36
37
38

```

Fonte: Bruno Amador

Figura 20 - Código main

```

main.py > ...
1  from src.controllers.cadastro import Cadastro
2  from src.controllers.login import Login
3  from src.recognizer.treinamento import Treinamento
4
5  def main():
6      cadastro = Cadastro()
7      login = Login()
8
9      def spaces(count, char):
10         for x in range(0, int(count)):
11             print(char)
12
13     print('=====')
14     print('=====      AgroSafe      =====')
15     print('=====')
16
17     spaces(3, '...')
18
19     print('[INFO] --- [1] = Login')
20     print('[INFO] --- [2] = Cadastro')
21
22     spaces(1, '...')
23
24     valueSelected = int(input('[INPUT] --- Selecionar acao: '))
25
26     if(valueSelected == 1):
27         print('[SISTEMA] --- Iniciando Login')
28         #Action Login
29         login.loginAction()
30
31     elif(valueSelected == 2):
32         print('[SISTEMA] --- Iniciando Cadastro')
33         #Action Cadastro
34         cadastro.cadastroAction()
35
36     else:
37         print('[ERRO] --- Valor inválido: ' + str(valueSelected))
38         exit
39
40 main()

```

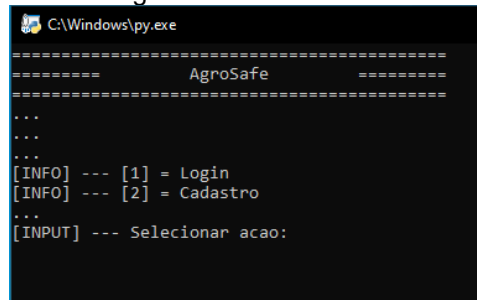
Fonte: Bruno Amador

Para maior detalhamento do código, link do github:

<https://github.com/Venedertti/aps-2021-2>

7. SISTEMA EM FUNCIONAMENTO

Figura 21 - Menu Inicial



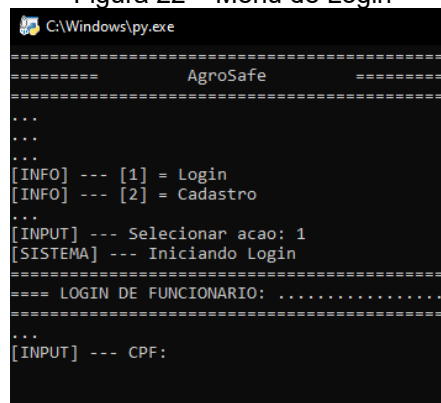
```

C:\Windows\py.exe
=====
                        AgroSafe
=====
...
...
[INFO] --- [1] = Login
[INFO] --- [2] = Cadastro
...
[INPUT] --- Selecionar acao:
  
```

Fonte: Bruno Amador

Logo ao abrir o programa este é o menu apresentado para o usuário, dando a possibilidade de duas escolhas, para login ou cadastro.

Figura 22 – Menu de Login



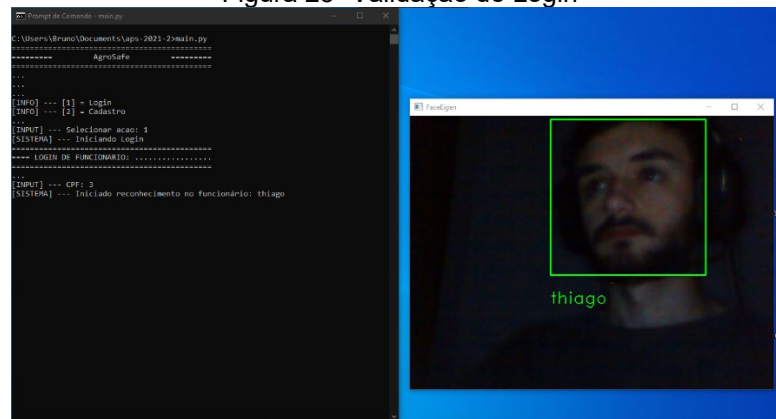
```

C:\Windows\py.exe
=====
                        AgroSafe
=====
...
...
[INFO] --- [1] = Login
[INFO] --- [2] = Cadastro
...
[INPUT] --- Selecionar acao: 1
[SISTEMA] --- Iniciando Login
=====
=== LOGIN DE FUNCIONARIO: .....
=====
...
[INPUT] --- CPF:
  
```

Fonte: Bruno Amador

Escolhendo a ação de Login será necessário estar cadastrado no sistema, e logo pedirá o CPF do usuário cadastrado.

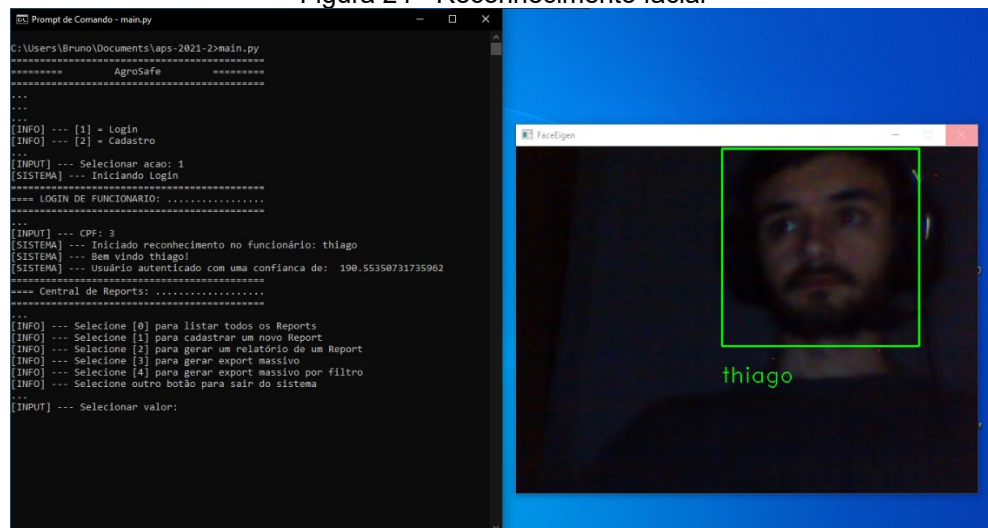
Figura 23 -Validação de Login



Fonte: Bruno Amador

Após digitar o CPF, O algoritmo fará o reconhecimento facial do usuário cadastrado no sistema.

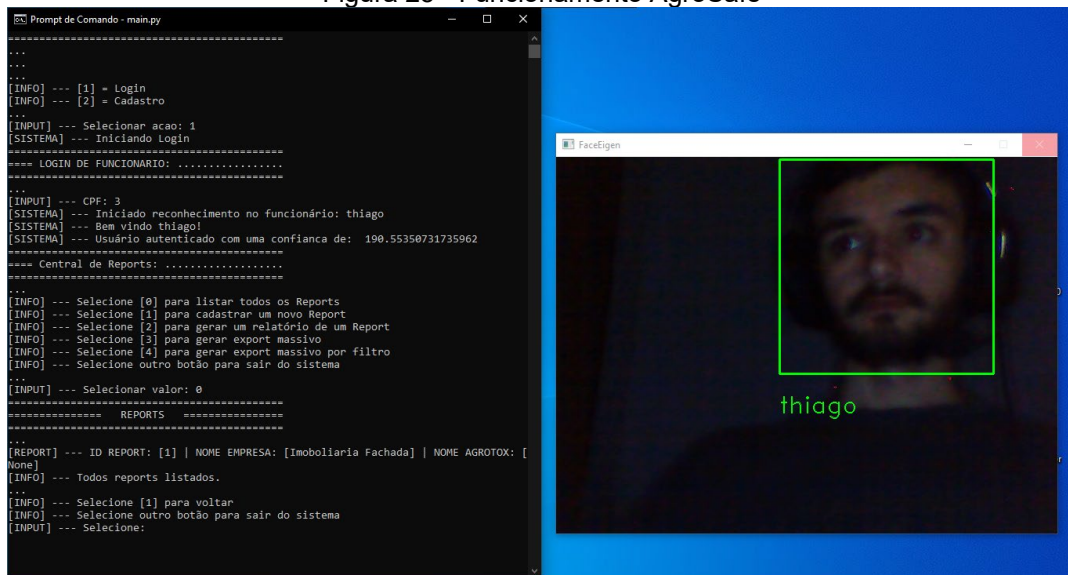
Figura 24 - Reconhecimento facial



Fonte: Bruno Amador

O sistema reconhecendo o usuário, irá aparecer uma mensagem de boas-vindas, e apresentando ao usuário a Central de Reports.

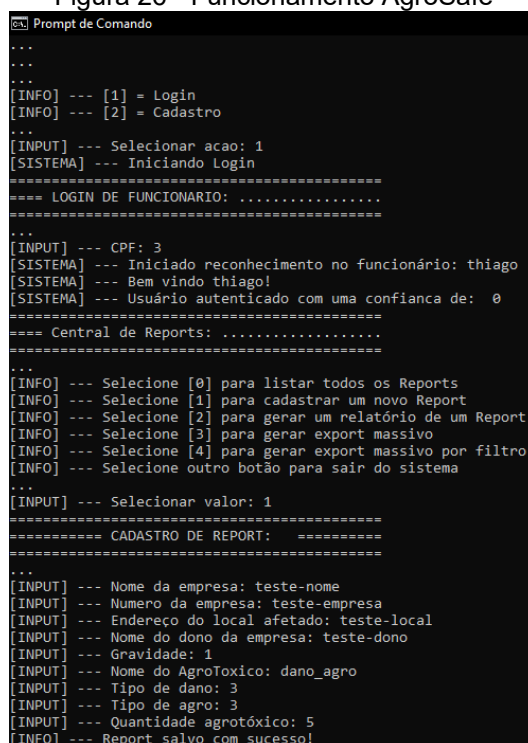
Figura 25 - Funcionamento AgroSafe



Fonte: Bruno Amador

Selecionando a opção 0, o sistema listará todos os reports armazenados no sistema.

Figura 26 - Funcionamento AgroSafe



Fonte: Bruno Amador

Selecionando a ação 1, o usuário irá cadastrar um novo Report, sendo assim será necessário dados como nome da empresa, local afetado, nome do Agrotóxico, quantidade etc.

Figura 27 - Funcionamento AgroSafe

```

Prompt de Comando
...
[INFO] --- [1] = Login
[INFO] --- [2] = Cadastro
...
[INPUT] --- Selecionar acao: 1
[SISTEMA] --- Iniciando Login
===== LOGIN DE FUNCIONARIO: =====
...
[INPUT] --- CPF: 3
[SISTEMA] --- Iniciado reconhecimento no funcionário: thiago
[SISTEMA] --- Bem vindo thiago!
[SISTEMA] --- Usuário autenticado com uma confiança de: 0
===== Central de Reports: =====
...
[INFO] --- Selecione [0] para listar todos os Reports
[INFO] --- Selecione [1] para cadastrar um novo Report
[INFO] --- Selecione [2] para gerar um relatório de um Report
[INFO] --- Selecione [3] para gerar export massivo
[INFO] --- Selecione [4] para gerar export massivo por filtro
[INFO] --- Selecione outro botão para sair do sistema
...
[INPUT] --- Selecionar valor: 2
[INPUT] --- Insira o código do REPORT: 1
[INFO] --- Relatório salvo no path: C:\Users\Bruno\Documents\REPORT-UNITARIO-1.txt
[WARN:0] global D:\a\opencv-python\opencv-python\opencv\modules\videoio\src\cap_msm
F.cpp (438) 'anonymous-namespace':::SourceReaderCB::~SourceReaderCB terminating async
callback
[WARN:0] global D:\a\opencv-python\opencv-python\opencv\modules\videoio\src\cap_msm
F.cpp (438) 'anonymous-namespace':::SourceReaderCB::~SourceReaderCB terminating async
callback
C:\Users\Bruno\Documents\aps-2021-2>

```

Fonte: Bruno Amador

Selecionando a ação 2, o sistema irá gerar um relatório de um report digitando o seu código, armazenado na pasta documentos do usuário.

Figura 28 - Relatório do Report

```

REPORT-UNITARIO-1.txt - Bloco de Notas
Arquivo  Editar  Formatar  Exibir  Ajuda
=====
[Agrosafe] - Relatório: 1
Gerado por: thiago
=====
...
[INFO] ---> Gerais:
=Nome empresa: Imobiliaria Fachada
=Numero empresa: Endereco 1, numero 1 - SP
=Local afetado: ENDERECO
=Dono da empresa: Guilherme Gomes David
...
[INFO] ---> Agrotóxico:
=Nome agrotóxico: Agrotox de solo
=Tipo agrotóxico: CATEGORIA 1
=Descrição do agrotóxico: Extremamente toxico
=Efeito oral: Fatal se ingerido
=Efeito dermico: Fatal em contato com a pele
=Efeito ao inalar: Fatal se inalado
=Quantidade de agrotóxico: 1
...
[INFO] ---> Impacto
=Tipo de dano: 1
=Gravidade: 1
=====

```

Fonte: Bruno Amador

Figura 29 - Funcionamento AgroSafe

```

Prompt de Comando
...
[INFO] --- [1] = Login
[INFO] --- [2] = Cadastro
...
[INPUT] --- Selecionar acao: 1
[SISTEMA] --- Iniciando Login
==== LOGIN DE FUNCIONARIO: .....
...
[INPUT] --- CPF: 3
[SISTEMA] --- Iniciado reconhecimento no funcionário: thiago
[SISTEMA] --- Bem vindo thiago!
[SISTEMA] --- Usuário autenticado com uma confiança de: 0
==== Central de Reports: .....
...
[INFO] --- Selecione [0] para listar todos os Reports
[INFO] --- Selecione [1] para cadastrar um novo Report
[INFO] --- Selecione [2] para gerar um relatório de um Report
[INFO] --- Selecione [3] para gerar export massivo
[INFO] --- Selecione [4] para gerar export massivo por filtro
[INFO] --- Selecione outro botão para sair do sistema
...
[INPUT] --- Selecionar valor: 3
[INFO] --- Relatório salvo no path: C:\Users\Bruno\Documents\REPORT-Massivo.txt
[WARN:0] global D:\a\opencv-python\opencv-python\opencv\modules\videoio\src\cap_msm
F.cpp (438) 'anonymous-namespace':::SourceReaderCB::~SourceReaderCB terminating async
callback
[WARN:0] global D:\a\opencv-python\opencv-python\opencv\modules\videoio\src\cap_msm
F.cpp (438) 'anonymous-namespace':::SourceReaderCB::~SourceReaderCB terminating async
callback
C:\Users\Bruno\Documents>

```

Fonte: Bruno Amador

Selecionando a ação 3, o sistema irá exportar um Report Massivo.

Figura 30 – Relatório Massivo

```

REPORT-Massivo.txt - Bloco de Notas
Arquivo Editar Formatar Exibir Ajuda
=====
[Agrosafe] - Relatório Massivo
Filtro: ALL
Gerado por: thiago
=====
...
[INFO] ----> Gerais:
=Nome da empresa: {Imobiliaria Fachada}
=Numero da empresa: {Endereco 1, numero 1 - SP}
=Local afetado: 111111111
=Dono da empresa: Guilherme Gomes David
...
[INFO] ----> Agrotóxico:
=Nome do agrotóxico: Agrotox de solo
=Tipo do agrotóxico: CATEGORIA 1
=Descrição do agrotóxico: Extremamente toxico
=Efeito oral: Fatal se ingerido
=Efeito dermico: Fatal em contato com a pele
=Efeito ao inalar: Fatal se inalado
=Quantidade de agrotóxico: 1
...
[INFO] ----> Impacto
=Tipo de dano: 1
=Gravidade: 1
=====...
[INFO] ----> Gerais:
=Nome da empresa: {Imobiliaria Fachada}
=Numero da empresa: {Endereco 1, numero 1 - SP}
=Local afetado: 111111111
=Dono da empresa: Guilherme Gomes David
...
[INFO] ----> Agrotóxico:
=Nome do agrotóxico: Agrotox de solo
=Tipo do agrotóxico: CATEGORIA 1
=Descrição do agrotóxico: Extremamente toxico
=Efeito oral: Fatal se ingerido
=Efeito dermico: Fatal em contato com a pele
=Efeito ao inalar: Fatal se inalado
=Quantidade de agrotóxico: 1
...
[INFO] ----> Impacto
=Tipo de dano: 1
=Gravidade: 1

```

Fonte: Bruno Amador

Figura 31 - Funcionamento AgroSafe

```

C:\ Prompt de Comando
...
[INFO] --- [1] = Login
[INFO] --- [2] = Cadastro
[INPUT] --- Selecionar acao: 1
[SISTEMA] --- Iniciando Login
=====
=== LOGIN DE FUNCIONARIO: .....
=====
[INPUT] --- CPF: 3
[SISTEMA] --- Iniciado reconhecimento no funcionário: thiago
[SISTEMA] --- Bem vindo thiago!
[SISTEMA] --- Usuário autenticado com uma confiança de: 0
=====
=== Central de Reports: .....
=====
[INFO] --- Selecione [0] para listar todos os Reports
[INFO] --- Selecione [1] para cadastrar um novo Report
[INFO] --- Selecione [2] para gerar um relatório de um Report
[INFO] --- Selecione [3] para gerar export massivo
[INFO] --- Selecione [4] para gerar export massivo por filtro
[INFO] --- Selecione outro botão para sair do sistema
[INPUT] --- Selecionar valor: 4
[INFO] --- Selecione [1] para fazer o filtro pelo nome da [Empresa]
[INFO] --- Selecione [2] para fazer o filtro pelo nome do [AgroTóxico]
[INPUT] --- Selecione o filtro: 2
[INPUT] --- Insira o valor para o filtro: nome
[INFO] --- Relatório salvo no path: C:\Users\Bruno\Documents\REPORT-MASSIVO-COM-FILTRO-NOME_Agro.txt

[ WARN:0] global D:\opencv-python\opencv-python\opencv\modules\videoio\src\cap_msmf.cpp (438) `anony
callback
[ WARN:0] global D:\opencv-python\opencv-python\opencv\modules\videoio\src\cap_msmf.cpp (438) `anony
callback

C:\Users\Bruno\Documents\aps-2021-2>

```

Fonte: Bruno Amador

Selecionando a ação 4, o sistema irá exportar um Report Massivo por filtro pelo nome da empresa ou do agrotóxico, cujo inserido pelo usuário.

Figura 32 – Relatório Massivo com Filtro Empresa

```

REPORT-MASSIVO-COM-FILTRO-NOME_EMPRE.txt - Bloco de Notas
Arquivo  Editar  Formatar  Exibir  Ajuda
=====
[Agrosafe] - Relatório Massivo
Filtro: NOME_EMPRE
Gerado por: thiago
=====
...
[INFO] ---> Gerais:
=Nome da empresa: teste-nome
=Numero da empresa: teste-empresa
=Local afetado: teste-local
=Dono da empresa: teste-dono
...
[INFO] ---> Agrotóxico:
=Nome do agrotóxico: nome_agro
=Tipo do agrotóxico: CATEGORIA 3
=Descrição do agrotóxico: Moderadamente toxico
=Efeito oral: Toxico se ingerido
=Efeito dermico: Toxico em contato com a pele
=Efeito ao inalar: Toxico se inalado
=Quantidade de agrotóxico: 3
...
[INFO] ---> Impacto
=Tipo de dano: 3
=Gravidade: 1
=====

```

Fonte: Bruno Amador

Figura 33 - Relatório Massivo com Filtro Agrotóxico

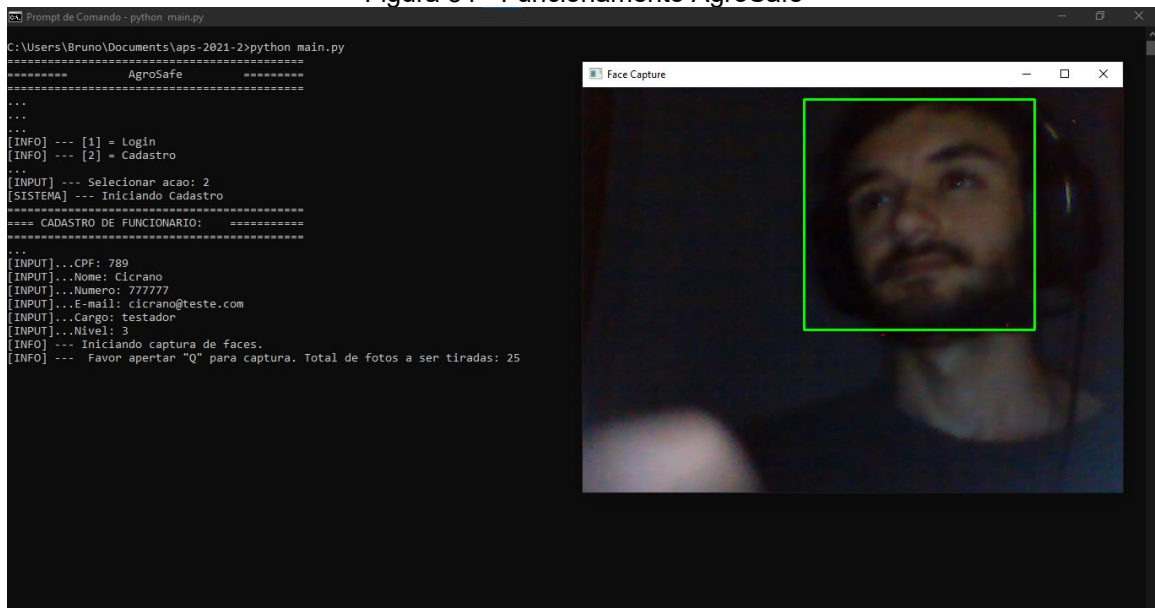
```

REPORT-MASSIVO-COM-FILTRO-NOME_AGRO.txt - Bloco de Notas
Arquivo  Editar  Formatar  Exibir  Ajuda
=====
[Agrosafe] - Relatório Massivo
Filtro: NOME_AGRO
Gerado por: thiago
=====
...
[INFO] ---> Gerais:
=Nome da empresa: teste-nome
=Numero da empresa: teste-empresa
=Local afetado: teste-local
=Dono da empresa: teste-dono
...
[INFO] ---> Agrotóxico:
=Nome do agrotóxico: nome_agro
=Tipo do agrotóxico: CATEGORIA 3
=Descrição do agrotóxico: Moderadamente toxico
=Efeito oral: Toxico se ingerido
=Efeito dermico: Toxico em contato com a pele
=Efeito ao inalar: Toxico se inalado
=Quantidade de agrotóxico: 3
...
[INFO] ---> Impacto
=Tipo de dano: 3
=Gravidade: 1
=====

```

Fonte: Bruno Amador

Figura 34 - Funcionamento AgroSafe



Fonte: Bruno Amador

Outra opção na tela inicial é a de cadastro, ao selecionar opção, o sistema irá abrir a câmera configurada na máquina do usuário, para que faça o treinamento do algoritmo tirando 25 fotos em diferentes ângulos e expressões, para que haja o melhor desempenho no reconhecimento.

8. BIBLIOGRAFIA

AGROTÓXICO. **INCA - INSTITUTO NACIONAL DE CÂNCER**. Disponível em: <https://www.inca.gov.br/exposicao-no-trabalho-e-no-ambiente/agrotoxicos#:~:text=Agrot%C3%B3xicos%20s%C3%A3o%20produtos%20qu%C3%ADmicos%20sint%C3%A9ticos,2002%3B%20INCA%2C%202021>. Acesso em: 2021.

GRANATYR, J.; ALVES, G. Reconhecimento Facial com Python e OpenCV. **Udemy**. Disponível em: <https://www.udemy.com/course/reconhecimento-facial-com-python-e-opencv/>. Acesso em: 2021.

HOSTGATOR. Conheça os 7 princípios do Clean Code. **Hostgator**. Disponível em: <https://www.hostgator.com.br/blog/clean-code-o-que-e/>. Acesso em: 2021.

JESUS, L. *et al.* Análise de Métodos de Processamento de Imagens para Reconhecimento Facial utilizando Fisherfaces em Imagens sob Condições Desfavoráveis. Salvador, Bahia, Brasil: Núcleo de Arquitetura de Computadores e Sistemas Operacionais (ACSO) - Universidade do Estado da Bahia (UNEB), p. 10.

OPENCV. **OpenCV - Open Source Computer Vision**. Disponível em: https://docs.opencv.org/4.x/d6/d00/tutorial_py_root.html. Acesso em: 2021.

RASCHKA, S. Linear Discriminant Analysis. **sebastianraschka**. Disponível em: https://sebastianraschka.com/Articles/2014_python_lda.html. Acesso em: 2021.

9. FICHA DE ATIVIDADES PRÁTICAS SUPERVISIONADAS



FICHA DAS ATIVIDADES PRÁTICAS SUPERVISIONADAS - APS

NOME: Bruno Amador Rodrigues TURMA: CC6P06 RA: 11416FF7
 CURSO: Ciência da Computação CAMPUS: Alphaville SEMESTRE: 6º TURNO: Noturno
 CÓDIGO DA ATIVIDADE: 77B3 SEMESTRE: _____ ANO GRADE: _____

DATA DA ATIVIDADE	DESCRIÇÃO DA ATIVIDADE	TOTAL DE HORAS	ASSINATURA DO ALUNO	HORAS ATRIBUÍDAS (1)	ASSINATURA DO PROFESSOR
18/09	pesquisa sobre reconhecimento facial	5	Bruno Amador		
19/09	Reunião em grupo	3	Bruno Amador		
24/09	Definição de atividades	2	Bruno Amador		
29/09	Curso em python com foco em reconhecimento facial	30	Bruno Amador		
08/10	Início da parte escrita	5	Bruno Amador		
09/10	Desenvolvimento da aplicação	15	Bruno Amador		
22/10	Revisão do código	5	Bruno Amador		
06/11	Revisão da parte escrita	5	Bruno Amador		
13/11	Verificação de possíveis erros	5	Bruno Amador		

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

TOTAL DE HORAS ATRIBUÍDAS: 75

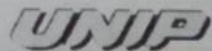
AValiação: _____

Aprovado ou Reprovado

NOTA: _____

DATA: ____/____/____

CARIMBO E ASSINATURA DO COORDENADOR DO CURSO

**FICHA DAS ATIVIDADES PRÁTICAS SUPERVISIONADAS - APS**

NOME: Guilherme Gomes David TURMA: CC696 RA: N44461-4
CURSO: Ciências da Computação CAMPUS: Alphaville SEMESTRE: 6º TURNO: Nocturno
CÓDIGO DA ATIVIDADE: SEMESTRE: ANO GRADE:

[illegible]

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

TOTAL DE HORAS ATRIBUÍDAS: 75

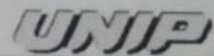
AVALIAÇÃO: _____

Aprovado ou Reprovado

NOTA: _____

DATA: / /

CARIMBO E ASSINATURA DO COORDENADOR DO CURSO



UNIVERSIDADE PAULISTA

FICHA DAS ATIVIDADES PRÁTICAS SUPERVISIONADAS - APS

NOME: Thiago Gomes de Oliveira TURMA: CC6006 RA: N4893E8
 CURSO: Ciências da Computação CAMPUS: Alfândega SEMESTRE: 6º TURNO: Noite
 CÓDIGO DA ATIVIDADE: _____ SEMESTRE: _____ ANO GRADE: _____

DATA DA ATIVIDADE	DESCRIÇÃO DA ATIVIDADE	TOTAL DE HORAS	ASSINATURA DO ALUNO	HORAS ATRIBUÍDAS (1)	ASSINATURA DO PROFESSOR
08/09	Atividade prática supervisionada	5	Thiago Gomes		
09/09	Atividade prática supervisionada	5	Thiago Gomes		
10/09	Atividade prática supervisionada	5	Thiago Gomes		
11/09	Atividade prática supervisionada	5	Thiago Gomes		
12/09	Atividade prática supervisionada	5	Thiago Gomes		
01/10	Atividade prática supervisionada	5	Thiago Gomes		
02/10	Atividade prática supervisionada	5	Thiago Gomes		
03/10	Atividade prática supervisionada	5	Thiago Gomes		
04/10	Atividade prática supervisionada	5	Thiago Gomes		
05/10	Atividade prática supervisionada	5	Thiago Gomes		
06/10	Atividade prática supervisionada	5	Thiago Gomes		
07/10	Atividade prática supervisionada	5	Thiago Gomes		
08/10	Atividade prática supervisionada	5	Thiago Gomes		
09/10	Atividade prática supervisionada	5	Thiago Gomes		
10/10	Atividade prática supervisionada	5	Thiago Gomes		
11/10	Atividade prática supervisionada	5	Thiago Gomes		
12/10	Atividade prática supervisionada	5	Thiago Gomes		
01/11	Atividade prática supervisionada	5	Thiago Gomes		
02/11	Atividade prática supervisionada	5	Thiago Gomes		
03/11	Atividade prática supervisionada	5	Thiago Gomes		
04/11	Atividade prática supervisionada	5	Thiago Gomes		
05/11	Atividade prática supervisionada	5	Thiago Gomes		
06/11	Atividade prática supervisionada	5	Thiago Gomes		
07/11	Atividade prática supervisionada	5	Thiago Gomes		
08/11	Atividade prática supervisionada	5	Thiago Gomes		
09/11	Atividade prática supervisionada	5	Thiago Gomes		
10/11	Atividade prática supervisionada	5	Thiago Gomes		
11/11	Atividade prática supervisionada	5	Thiago Gomes		
12/11	Atividade prática supervisionada	5	Thiago Gomes		
01/12	Atividade prática supervisionada	5	Thiago Gomes		
02/12	Atividade prática supervisionada	5	Thiago Gomes		
03/12	Atividade prática supervisionada	5	Thiago Gomes		
04/12	Atividade prática supervisionada	5	Thiago Gomes		
05/12	Atividade prática supervisionada	5	Thiago Gomes		
06/12	Atividade prática supervisionada	5	Thiago Gomes		
07/12	Atividade prática supervisionada	5	Thiago Gomes		
08/12	Atividade prática supervisionada	5	Thiago Gomes		
09/12	Atividade prática supervisionada	5	Thiago Gomes		
10/12	Atividade prática supervisionada	5	Thiago Gomes		
11/12	Atividade prática supervisionada	5	Thiago Gomes		
12/12	Atividade prática supervisionada	5	Thiago Gomes		

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

TOTAL DE HORAS ATRIBUÍDAS: 79

AVALIAÇÃO: _____

Aprovado ou Reprovado

NOTA: _____

DATA: ____/____/____

CARIMBO E ASSINATURA DO COORDENADOR DO CURSO