**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:

Interface gráfica do usuário, Aplicativo

Descrição gerada automaticamente

Figura - 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.

**Introdução**

**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.

* **Detecção Facial**

A detecção facial tem como principal objetivo é 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.

* **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 onde é feito 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

* **Algoritmo: FisherFaces**

**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 identificados 3 libs para auxiliar no desenvolvimento, são elas:

* **Numpy:** Lib responsável por fazer o processamento de grandes de arranjos e matrizes.
* **OpenCV:** Lib responsável por fazer o processamento de imagem, detecção e reconhecimento facial.
* **mysql-python-connector:** Lib responsável por realizar a conexão com o banco de dados e fazer a execução de querys.

Após instalar as libs 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 libs 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 scipt *activate.ps1.* Assim que estiver ativo a tag *(venv)* irá constar no seu terminal.

Texto

Descrição gerada automaticamente

Assim que executarmos o scritp *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.

Texto

Descrição gerada automaticamente

* **Cadastro**

Na ação do cadastro é a etapa onde o usuário irá realizar o cadastro de um novo funcionário, realizar a captura de fotos de amostra e 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 insira 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 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.

Ne 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 na escala pois o algoritmo trabalha melhor com imagens nesta escala.

Texto

Descrição gerada automaticamente

Assim que as 25 fotos forem capturas 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.

* **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 está 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.

* **Report**

Na tela de report o usuário irá ter acesso e interação com os reports, podendo gerar relatório, cadastrar report e consultar outros reports de acordo com o nível do funcionário.

**Projeto**