

UNIVERSIDADE PAULISTA – UNIP

Ciência da Computação

DESENVOLVIMENTO DE UM SISTEMA DE IDENTIFICAÇÃO E
AUTENTICAÇÃO BIOMÉTRICA

RENAN FELIPE SILVA

RA: C524808

São Paulo

Índice

1. INTRODUÇÃO	4
2. FUNDAMENTOS DAS PRINCIPAIS TÉCNICAS BIOMÉTRICAS.....	4
2.1. Métodos Biométricos.....	4
2.1.1. Impressão Digital.....	4
2.1.2. Reconhecimento de Voz	5
2.1.3. Reconhecimento Facial.....	6
3. PLANO DE DESENVOLVIMENTO DA APLICAÇÃO	8
3.1. Elementos e Ferramentas.....	8
3.2. Bibliotecas utilizadas.....	9
3.3. Comparação de modelos	9
3.3.1. Eigenfaces.....	9
3.3.2. Fisherfaces.....	10
3.3.3. LBPH (Local Binary Patterns Histograms)	10
4. PROJETO.....	12
4.1. Estrutura e módulos	12
5. RELATÓRIO COM AS LINHAS DE CÓDIGO.....	14
6. APRESENTAÇÃO DA APLICAÇÃO EM FUNCIONAMENTO	24
6.1. Sistema	24
6.2. Exceções.....	29
7. BIBLIOGRAFIA.....	30
8. FICHA DE ATIVIDADES PRÁTICAS SUPERVISIONADAS	30

OBJETIVO E MOTIVAÇÃO DO TRABALHO

Este trabalho tem como objetivo a conclusão da Atividade Prática Supervisionada(APS) que propõe a elaboração de um sistema capaz de identificar e autenticar uma entrada de dados através da biometria. A Atividade Prática Supervisionada(APS) é constituída da seguinte forma:

“Pede-se aos alunos que desenvolvam uma ferramenta de identificação e autenticação biométrica que restrinja o acesso a uma rede com banco dados do Ministério do Meio Ambiente. As informações são estratégicas sobre as propriedades rurais que utilizam agrotóxicos proibidos por causarem grandes impactos nos lençõs freáticos, rios e mares. As informações de nível 1 todos podem ter acesso; as de nível 2 são restritas aos diretores de divisões; as de nível 3 somente são acessadas pelo ministro do meio ambiente”.

1. INTRODUÇÃO

A biometria palavra originada da junção de duas palavras gregas(bios= vida, metron = medida), no universo da tecnologia é responsável por indicar as características físicas, biológicas e principalmente únicas de cada um dos seres humanos.

Características essas que devem ser extraídas e processadas através de sistemas de identificação de diversas linguagens de programação de modo que busque garantir a integridade e segurança do acesso a informação.

A biometria já faz parte da rotina de milhares de usuários em diferentes aspectos, desde um simples desbloqueio de tela do smartphone pessoal até permissão de pagamento do sistema do banco, o que se tornou tão simples na verdade possui varias vertentes como por exemplo reconhecimento facial, impressão digital e reconhecimento de voz.

2. FUNDAMENTOS DAS PRINCIPAIS TÉCNICAS BIOMÉTRICAS

Se tratando de biometria, existem diversos meios para se implementar essa tecnologia, tudo isso por conta dos mais variados tipos de biometria que foram criados, que serão desenvolvidos e principalmente aqueles que estão em constante aperfeiçoamento. Dentre as diversas possibilidades de implementação de biometria se faz necessário conhecer alguns dos seus principais tipos e suas informações.

2.1. Métodos Biométricos

2.1.1. Impressão Digital

Um dos métodos mais conhecidos da biometria é o da Impressão Digital, tipo esse considerado o método mais antigo e de melhor custo benefício de biometria e o com um alto nível de confiabilidade. O método de impressão digital é responsável por identificar e autenticar as linhas formadas nas pontas dos dedos das mãos, cada indivíduo possui uma impressão digital única, tornando assim possível indentificá-lo com um alto nível de confiança.

É possível identificar impressões digitais únicas por conta dos pontos característicos e formas que cada uma possui, esses pontos característicos são analisados pelos sistemas AFIS(Automated Fingerprint Identification System),

que também podem ser chamados de Sistemas de identificação Automatizada de Impressão Digital.

É resultado desse método um alto nível de confiabilidade por conta da baixa mutabilidade dos dados de entradas que são as impressões digitais de uma pessoa, ao longo do tempo, não é comum um indivíduo perder suas, assim, as digitais estão menos propensas a alteração.

Como esse método tem um alto nível de confiabilidade, vemos a implementação dele em diversos ambientes e equipamentos, por exemplo:

- Relógio de ponto: empresas utilizam equipamentos que fazem o uso da impressão digital para registrar entrada e saída do funcionário, assim é possível monitorar o horário de cada funcionário.
- Smartphones: não é mais comum smartphones que ao invés de senhas numéricas optam por impressão digital como mecanismo para o desbloqueio do equipamento, assim evitando invasões indesejadas
- Automóveis: conforme avanços desse método de biometria, chegamos num nível onde já é possível ligar o próprio carro com a própria digital, assim evitando furtos.
- Sistemas bancários: para garantir a autenticação do acesso do usuário, bancos implementam essa tecnologia para que seus clientes tenham segurança quando realizar transações, sacar dinheiro e ou simplesmente consultar informações de conta.

2.1.2. Reconhecimento de Voz

O reconhecimento de voz não caminha distante da visão futurística, visão essa que é mostrada principalmente em filmes de espionagem. O fato é que essa ideia não ficou só nos filmes, hoje diversas aplicações realizam essa tarefa de modo simples e rápido.

Uma aplicação “ouve” as palavras e as associa com padrões de pavaras buscando uma alta similaridade, isso só é possível por conta de conversores analógicos que convertem as ondas sonoras para dados digitais:

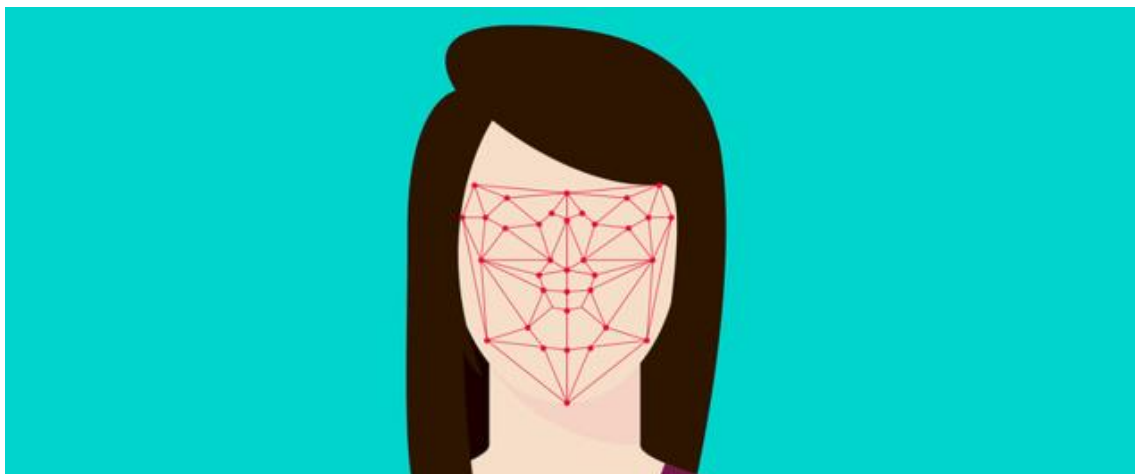
- **Atendente automática:** operadoras de telefonia oferecem esse serviço desde o primeiro contato com um cliente, quando ligamos para qualquer que seja o objetivo somos recepcionados por uma atendente que trilha o nosso caminho naquela chamada.
- **Assistente pessoal:** os smartphones hoje, oferecem um atendente pessoal que possibilita o usuário realizar tarefas como ligar, pesquisar algo na rede com uma simples fala.

2.1.3. Reconhecimento Facial

O Reconhecimento Facial é considerado um método que pode aposentar as carteiras de identificação com foto, muito usadas em carteiras de estudante e bilhetes únicos de transporte, esse método é considerado barato pois se faz necessário basicamente uma câmera e um programa bem definido para começar a implementá-lo.

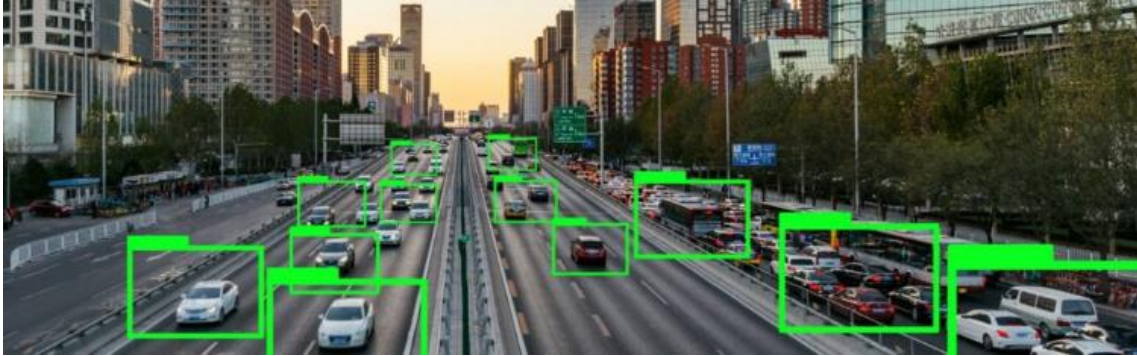
O reconhecimento facial atua no campo da visão computacional que simula a capacidade do olho humano, esse método se baseia em dois campos, detecção e reconhecimento.

No campo de detecção, um sistema de reconhecimento facial apenas identifica faces humanas, a aplicação não irá classificar aquela face humana com nomes ou atributos relacionados à face detectada. E.g:

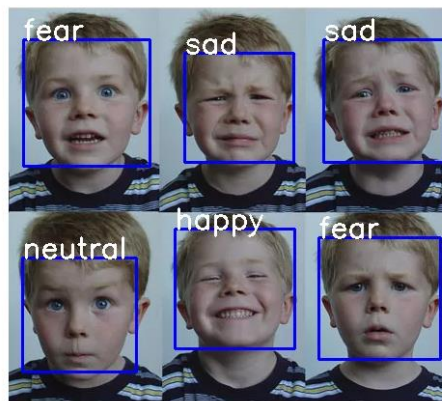


Aplicações que atuam nesse campo dos sistemas de biometria, podemos citar como resultados:

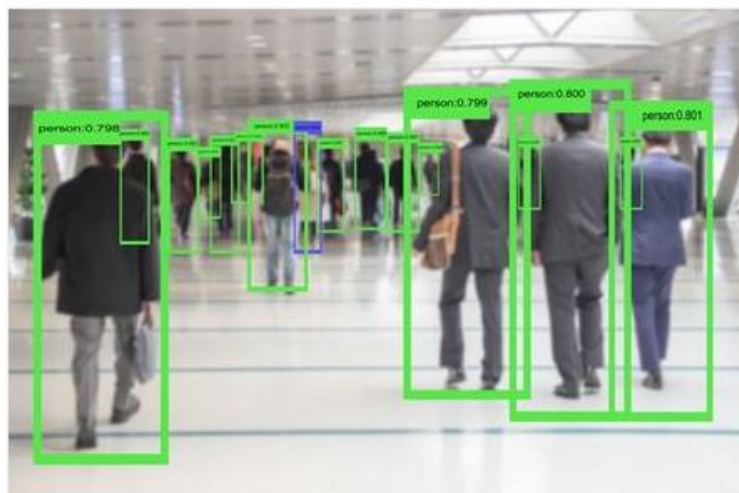
- Contagem de carros numa imagem de rodovia



- Detecção de expressões faciais



- Contagem de pessoas presentes em um determinado ambiente



No campo de Reconhecimento que pode ser considerado o último de um sistema de biometria, o método de reconhecimento facial não só detecta faces humanas mas é capaz de reconhecer aquela face, é possível identificar nome, cor de pele e atributos relacionados aquela face detectada desde que todos os requisitos para implementar um sistema de biometria estejam de acordo.

Para esse tipo de aplicação, podemos analisar como resultado de sistemas de biometria que fazem o uso dessa vertente da biometria:

- Desbloqueio de celular
- Sistemas de segurança
- Check in em aeroporto
- Entrada de condomínios

Para esse projeto, o método de Reconhecimento Facial foi escolhido para implementar os conceitos teóricos adquiridos em sala de aula com o uso da linguagem de programação Python.

3. PLANO DE DESENVOLVIMENTO DA APLICAÇÃO

Para a realização desse projeto, foi escolhido trabalhar com ferramentas open data de modo que acrescento para esse assunto acrescento de referencial teórico/prático de fácil acesso.

3.1. Elementos e Ferramentas

Abaixo a lista das principais ferramentas utilizadas e seu respectivo papel no projeto;

Ferramenta/Linguagem de Programação	Função
Python	Linguagem de programação optada para o desenvolvimento do projeto de Sistema de Biometria.
Visual Studio Code	IDE usada para desenvolvimento do projeto.
Google Keeper	Anotações importantes que impactam o sistema e o trabalho científico do projeto.
Todoist	Sistema de gerenciamento de tarefas para controle do projeto.
GitHub	Repositório/versionamento de código. https://github.com/rfelipesilva

3.2. Bibliotecas utilizadas

Biblioteca	Função
PIL(Python Imaging Library)	Adiciona recursos de processamento de imagem ao seu intérprete Python. Esta biblioteca suporta muitos formatos de arquivo e fornece recursos avançados de processamento de imagens e gráficos.
cv2(OpenCV)	Biblioteca licenciada por BSD de código aberto que inclui várias centenas de algoritmos de visão computacional.
time	Este módulo fornece várias funções relacionadas ao tempo.
Numpy	O NumPy é o pacote fundamental para a computação científica com Python.

3.3. Comparação de modelos

Foram realizados testes com 3 modelos para que o projeto alcance um alto nível de segurança no momento em que fosse necessário classificar qualquer face humana cadastrada.

Os modelos avaliados foram:

3.3.1. Eigenfaces

Eigenfaces refere-se a uma abordagem baseada na aparência para reconhecimento de rosto que busca capturar a variação em uma coleção de imagens de rosto e usa essas informações para codificar e comparar imagens de rostos individuais de maneira holística (em oposição a um baseado em partes ou em recurso). Especificamente, as autofaces são os principais componentes de uma distribuição de faces, ou equivalentemente, os autovetores da matriz de covariância do conjunto de imagens de faces, em que uma imagem com N pixels é considerada um ponto (ou vetor) no espaço N-dimensional.

É importante ressaltar que o Eigenfaces tem uma sensibilidade consideravelmente alta em relação a iluminação do ambiente no momento de treinamento e reconhecimento de faces.

- Avaliacao do modelo

n_components	precision	recall	f1-score	accuracy
10	0.62	0.63	0.6	0.6333
20	0.73	0.7	0.69	0.7
30	0.81	0.73	0.74	0.7333
40	0.8	0.73	0.74	0.7311
50	0.79	0.73	0.74	0.7221

3.3.2. Fisherfaces

O Fisherfaces também funciona com abordagem baseada na aparência para reconhecimento de rosto que busca capturar a variação em uma coleção de imagens de rosto e usa essas informações para codificar e comparar imagens de rostos individuais.

O Fisherfaces tem uma sensibilidade consideravelmente alta em relação a iluminação do ambiente no momento de treinamento e reconhecimento de faces e também requer uma carga computacional maior.

- Avaliacao

n_components	precision	recall	f1-score	accuracy
10	0.98	0.97	0.96	0.9666
20	0.82	0.73	0.75	0.7333
30	0.82	0.73	0.75	0.7333
40	0.82	0.73	0.75	0.7333
50	0.82	0.73	0.75	0.7333

3.3.3. LBPH (Local Binary Patterns Histograms)

O Local Binary Pattern (LBP) é um operador de textura simples, mas muito eficiente, que rotula os pixels de uma imagem limitando a vizinhança de cada pixel e considera o resultado como um número binário. Devido ao seu poder discriminativo e simplicidade computacional, o operador de textura LBP tornou-se uma abordagem popular em várias aplicações.

Pode ser visto como uma abordagem unificadora dos modelos estatísticos e estruturais tradicionalmente divergentes da análise de textura. Talvez a propriedade mais importante do operador LBP em aplicações do mundo real seja sua robustez às alterações monotônicas da escala de cinza causadas, por exemplo, por variações de iluminação.

Sobre o parâmetro Neighbors:

O número de pontos de amostra para construir o padrão binário local circular. Lembre-se: quanto mais pontos de amostra você incluir, maior será o custo computacional. Geralmente é definido como 8.

- Avaliação

neighbors	precision	recall	f1-score	accuracy
1	0.69	0.63	0.61	0.6333
2	0.69	0.63	0.61	0.6333
3	0.69	0.63	0.61	0.6333
4	0.79	0.67	0.68	0.6666
5	0.78	0.67	0.68	0.6666
6	0.7	0.67	0.65	0.6666
7	0.71	0.63	0.63	0.6333
8	0.74	0.67	0.67	0.6666
9	0.74	0.67	0.67	0.6666
10	0.74	0.67	0.67	0.6666

De acordo com a avaliação de cada modelo e seus parâmetros, podemos afirmar que o modelo FisherFaces é o mais adequado para ser utilizado no sistema de reconhecimento facial pois, sua acurácia é a mais alta.

4. PROJETO

As classes e os módulos que foram desenvolvidas para o funcionamento do sistema de reconhecimento facial serão descritas na presente parte de modo que acrescente no entendimento e construção da base teórica.

4.1. Estrutura e módulos

- main.py

O script desenvolvido em Python contém 4 classes e 1 função, sendo:

CLASS/DEF	FUNÇÃO
ReconecimentoFacial	Classe responsável por concentrar toda a iteração da GUI(Graphic User Interface).
Home	Classe responsável por apresentar a página principal de boas vindas ao usuário.
Entrar	Classe responsável por disponibilizar a página de log in, reconhecendo a face do usuário atual, caso todos os requerimentos estejam de acordo.
Cadastrar	Classe responsável por disponibilizar a página de cadastro ao usuário atual.
inicia_gui	Função responsável por iniciar a GUI(Graphic User Interface).
reinicia_gui	Função responsável por reiniciar a GUI(Graphic User Interface) em caso de inconformidade para o bom funcionamento do sistema.

- CapturaFace.py

O script CapturaFace.py desenvolvido em Python contém 1 classe:

CLASS/DEF	FUNÇÃO
ReferenceFile	Classe responsável por prover para o sistema arquivos fundamentais como referencia de usuário.

- Exceptions.py

O script Exceptions.py desenvolvido em Python contém 1 classe, sendo:

CLASS/DEF	FUNÇÃO
Verify	Classe responsável por verificar se arquivos fundamentais para o funcionamento do sistema estão de acordo como dataset de fotos para treinar o modelo, detectores de faces.

- Treinamento.py

O script Treinamento.py desenvolvido em Python contém 1 classe, sendo:

CLASS/DEF	FUNÇÃO
Treinamento	Classe responsável por treinar a máquina com e gerar o modelo de machine learning com o dataset cadastro pelos usuários.

- Dict.py

O script Dict.py desenvolvido em Python contém 1 classe, sendo:

CLASS/DEF	FUNÇÃO
Dict	Classe responsável por disponibilizar informações de acesso como ID de usuário, nível de acesso permitido e informação correspondente ao nível.

- Configuration.py

O script Configuration.py desenvolvido em Python contém 1 classe, sendo:

CLASS/DEF	FUNÇÃO
Path	Classe responsável por facilitar o acesso a diretórios fundamentais do sistema.

5. RELATÓRIO COM AS LINHAS DE CÓDIGO

- main.py

```
1  #-*- coding: utf-8 -*-
2  #! Python3
3
4  # @author
5  # - Renan Silva
6
7  import PIL
8  from PIL import Image, ImageTk
9
10 import cv2
11 import time
12
13 import numpy as np
14 from fuzzywuzzy import fuzz, process
15
16 import tkinter as tk
17 import tkinter.ttk as ttk
18 from tkinter import filedialog, messagebox
19 from tkinter import font as tkfont
20
21 from Dict import Dictionary
22 from Exceptions import Verify
23 from treinamentoLBPH import Treinamento
24 from CapturaFace import ReferenceFile
25 from Configuration import Path
26
27 class ReconhecimentoFacial(tk.Tk):
28
29     def __init__(self, *args, **kwargs):
30
31         tk.Tk.__init__(self, *args, **kwargs)
32         tk.Tk.wm_title(self, 'Ministério do Meio Ambiente') #title of the window
33         tk.Tk.wm_geometry(self, '700x700') #size of the window
34         self.cabecalho_font = tkfont.Font(family='Calibri', size=10)
35         self.title_font = tkfont.Font(family='Calibri', size=14, weight='bold')
36         self.text_font = tkfont.Font(family='Calibri', size=16)
37         self.small_font = tkfont.Font(family='Calibri', size=11)
38         self.access_name_font = tkfont.Font(family='Calibri', size=21)
39         self.access_nivel_font = tkfont.Font(family='Calibri', size=20)
40
41         container = tk.Frame(self)
42         container.pack(side='top', fill='both', expand=True)
43         container.grid_rowconfigure(0, weight=1)
44         container.grid_columnconfigure(0, weight=1)
45
46         self.frames = {}
47         for F in (Home, Entrar, Cadastrar):
48             page_name = F.__name__
49             frame = F(parent=container, controller=self)
50             self.frames[page_name] = frame
51
52             frame.grid(row=0, column=0, sticky='nsew')
53
54         self.show_frame("Home")
55
56     def show_frame(self, page_name):
57         for frame in self.frames.values():
58             frame.grid_remove()
59         '''Show a frame for the given page name'''
60         frame = self.frames[page_name]
61         frame.grid()
62
63 class Home(tk.Frame):
64
65     def __init__(self, parent, controller):
66         tk.Frame.__init__(self, parent)
67
68         self.controller = controller
69
70         label = tk.Label(self, text="Bem vindo ao Sistema de Identificação Biométrica do\nMinistério do Meio Ambiente", font=controller.title_font)
71         label.pack(side="top", fill="x", pady=10)
72
73         labelText = tk.Label(self)
74         labelText.pack()
75         texto = tk.StringVar()
76         texto.set('Este sistema controla o acesso a uma rede com banco dados do Ministério do Meio Ambiente que contém informações estratégicas sobre
77 | | | | | as propriedades rurais que utilizam agrotóxicos proibidos por causarem grandes impactos nos lenções freáticos, rios e mares.')
78         label = tk.Label(labelText, textvariable=texto, font=controller.text_font, relief="groove", wraplength=700, height=10)
79         label.pack(side="top", pady=100)
80
81         labelButtons = tk.Label(self)
82         labelButtons.pack()
83
84         button1 = ttk.Button(labelButtons, text="Já sou cadastrado", command=lambda: controller.show_frame("Entrar"))
85         button2 = ttk.Button(labelButtons, text="Cadastrar", command=lambda: controller.show_frame("Cadastrar"))
86         button1.pack(side="left", padx=5)
87         button2.pack(side="left", padx=10)
88
89         labelBottom = tk.Label(self)
90         labelBottom.pack(side="bottom")
91
92         labelBottom = tk.Label(labelBottom, text="Ministério do Meio Ambiente", font=controller.cabecalho_font)
93         labelBottom.pack(side="bottom", fill="x", pady=10)
```

```

96 class Entrar(tk.Frame):
97
98     def __init__(self, parent, controller):
99
100         tk.Frame.__init__(self, parent)
101         self.controller = controller
102         label = tk.Label(self, text="Ministério do Meio Ambiente", font=controller.title_font)
103         label.pack(side="top", fill="x", pady=2)
104
105         self.labelText = tk.Label(self)
106         self.labelText.pack()
107         self.textoEntrar = tk.StringVar()
108
109         self.frameFotoEntrar = tk.Frame(self)
110         self.frameFotoEntrar.pack(fill="x", anchor="center")
111
112         self.tirarFotoLabelEntrar = tk.Label(self.frameFotoEntrar)
113         self.tirarFotoLabelEntrar.pack()
114
115         if Verify().verifyFiles() == True:
116
117             self.textoEntrar.set('Detalhes sobre o acesso:\n
118             1 - Informação de nível 1, todos podem ter acesso.{}
119             2 - Informação de nível 2, restritas aos diretores de divisões.{}
120             3 - Informação de nível 3, somente Ministro do Meio Ambiente.{}'.format(' '*40, ' '*24, ' '*18))
121             label = tk.Label(self.labelText, textvariable=self.textoEntrar, font=controller.text_font, relief="groove", wraplength=700, height=10)
122             label.pack(side="top", pady=100)
123
124             self.entrarBtn = tk.Button(self, text="Entrar", command=self.checkFace)
125             self.entrarBtn.pack()
126
127         else:
128
129             self.textoEntrar.set('Parece que algo está errado, garanta que os pontos abaixo estão contemplados.\n
130             1 - Fotos tiradas com sucesso, seu dataset foi gerado corretamente no diretório:{}'.format(' '*40, ' '*24, ' '*18))
131             label = tk.Label(self.labelText, textvariable=self.textoEntrar, font=controller.text_font, relief="groove", wraplength=700, height=10)
132             label.pack(side="top", pady=100)
133
134             self.entrarBtn = tk.Button(self, text="Entrar", command=self.checkFace)
135             self.entrarBtn.pack()
136
137         button = ttk.Button(self, text="Ir para home", command=reinicia_gui)
138         button.pack()
139
140
141

```

```

143 self.labelContent = tk.Label(self, text="-")
144 self.labelContent.pack()
145
146 self.textoNome = tk.StringVar()
147 self.textoNome.set('')
148 self.labelNome = tk.Label(self, textvariable=self.textoNome, font=controller.acess_name_font)
149 self.labelNome.pack()
150
151 self.textoNivel = tk.StringVar()
152 self.textoNivel.set('')
153 self.labelNivel = tk.Label(self, textvariable=self.textoNivel, font=controller.acess_nivel_font)
154 self.labelNivel.pack()
155
156 self.scrollBarTexto = tk.Scrollbar(self)
157 self.scrollBarTexto.pack(side="right", fill="y")
158 self.textoInScrollBar = tk.Text(self, wrap= None, yscrollcommand=self.scrollBarTexto.set)
159
160 labelBottom = tk.Label(self)
161 labelBottom.pack(side="bottom")
162
163
164 def restart(self):
165     self.refresh()
166     self.controller.show_frame("Home")
167
168 #USADO PARA LIMPAR E RODAR DE NOVO
169 def refresh(self):
170     self.textoNome.set('')
171     self.textoNivel.set('')
172
173     self.textoInScrollBar.pack_forget()
174     self.textoInScrollBar = tk.Text(self, wrap= None, yscrollcommand=self.scrollBarTexto.set)
175
176

```

```

176 def checkFace(self):
177
178     #USADO PARA LIMPAR O TEXTO DE DESCRICAO
179     self.labelText.destroy()
180
181     detectorFace = cv2.CascadeClassifier("{}haarcascade-frontalface-default.xml".format(Path().reconhecedores))
182     reconhecedor = cv2.face.LBPHFaceRecognizer_create()
183     reconhecedor.read("{}classificadorLBPH.yml".format(Path().models))
184     largura, altura = 228, 228
185     font = cv2.FONT_HERSHEY_COMPLEX_SMALL
186
187     self.cameraEntrar = cv2.VideoCapture(0)
188
189     nomeDict = Dictionary().dictionary_id
190     lvlDict = Dictionary().dictionary_lvl
191
192     checkEntry = []
193
194     ids = []
195     confiancas = []
196
197     temp = 0

```

```

199 while (True):
200
201     conectado, imagem = self.cameraEntrar.read()
202     imagemCinza = cv2.cvtColor(imagem, cv2.COLOR_BGR2GRAY)
203     facesDetectadas = detectorFace.detectMultiScale(imagemCinza,
204                                                       scaleFactor=1.5,
205                                                       minSize=(30,30))
206
207     for (x, y, l ,a) in facesDetectadas:
208
209         imagemFace = cv2.resize(imagemCinza[y:y + a, x:x + l], (largura, altura))
210
211         cv2.rectangle(imagem, (x,y), (x + l, y + a), (0,0,255), 2)
212         id, confianca = reconhecedor.predict(imagemFace)
213         nome = nomeDict[str(id)]
214         lvl = lvlDict[str(id)]
215
216         ids.append(id)
217         confiancas.append(confianca)
218
219         cv2.putText(imagem, str(id), (x,y +(a+30)), font, 2, (0,0,255))
220         cv2.putText(imagem, str(confianca), (x,y +(a+50)), font, 1, (0,0,255))
221         temp += 1
222
223     cv2.imshow("Face", imagem)
224     cv2.waitKey(1)
225     if temp > 100:
226         break
227
228     self.validaFace(ids, confiancas)
229
230     self.cameraEntrar.release()
231     cv2.destroyAllWindows()

```

```

233 def validaFace(self, ids, confiancas):
234
235     nomeDict = Dictionary().dictionary_id
236     lvlDict = Dictionary().dictionary_lvl
237
238     idsAmount = 0
239     idsAmount = len(set(ids))
240
241     id = ''
242     id = ids[0]
243
244     minimo = 0
245     maximo = 0
246     minimo = min(confiancas)
247     maximo = max(confiancas)
248
249     print('Ids amount = {}'.format(idsAmount))
250     print('Min confianca = {}'.format(minimo))
251     print('Max confianca = {}'.format(maximo))
252
253     if idsAmount > 1 and (maximo >= 55 and minimo >= 55):
254         messagebox.showerror("Acesso negado", "Você não tem acesso!")
255     else:
256         self.mostraConteudo(id)

```

```

259 def mostraConteudo(self, id):
260
261     self.entrarBtn.destroy()
262     informations = Dictionary()
263     nomeDict = informations.dictionary_id
264     lvlDict = informations.dictionary_lvl
265
266     nome = ''
267     lvl = ''
268
269     nome = nomeDict[str(id)]
270     lvl = lvlDict[str(id)]
271
272     self.textoName.set('Bem vindo {}'.format(nome))
273     self.textoNivel.set('Você tem acesso à informação de nível {}'.format(lvl))
274
275     self.textoInScrollBar.pack(side="left")
276
277     if '1' in lvl:
278         texto = informations.getText('1')
279         self.textoInScrollBar.insert("1.0", "{}".format(texto))
280     elif '2' in lvl:
281         texto = informations.getText('2')
282         self.textoInScrollBar.insert("1.0", "{}".format(texto))
283     elif '3' in lvl:
284         texto = informations.getText('3')
285         self.textoInScrollBar.insert("1.0", "{}".format(texto))
286     else:
287         messagebox.showerror("Error", "Algo errado ocorreu identificando seu nível de acesso!")
288
289     self.scrollBarTexto.config(command=self.textoInScrollBar.yview)
290

```



```

291 class Cadastrar(tk.Frame):
292
293     def __init__(self, parent, controller):
294
295         tk.Frame.__init__(self, parent)
296         self.controller = controller
297
298         #-----
299         frameTitle = tk.Frame(self)
300         frameTitle.pack(fill='both')
301
302         labelTitle = tk.Label(frameTitle, text="Página de Cadastro", font=controller.title_font)
303         labelTitle.pack(side="top", fill="x", pady=10)
304
305         #-----
306         #nome
307         frameName = tk.Frame(self)
308         frameName.pack(fill='x')
309
310
311         nomeLabel = tk.Label(frameName, text="Digite seu nome:", bd=10, width=12, anchor="w")
312         nomeLabel.pack(side='left', padx=5, pady=5)
313
314         self.nomeEntry = tk.Entry(frameName)
315         self.nomeEntry.pack(fill='x', padx=5, expand=True)
316
317         #-----
318         #níveis
319         frameNiveis = tk.Frame(self)
320         frameNiveis.pack(fill='x')
321
322         niveisLabel = tk.Label(frameNiveis, text="Selecione seu nível de acesso:", bd=10, width=23, anchor="w")
323         niveisLabel.pack(side='left', padx=5, pady=5)
324
325         #to do -----
326         self.lv1 = tk.IntVar()
327         self.lv1Um = tk.Radiobutton(frameNiveis, text="Nível 1", variable=self.lv1, value=1)
328         self.lv1Um.pack(side="left")
329
330         self.lv1Dois = tk.Radiobutton(frameNiveis, text="Nível 2", variable=self.lv1, value=2)
331         self.lv1Dois.pack(side="left")
332
333         self.lv1Tres = tk.Radiobutton(frameNiveis, text="Nível 3", variable=self.lv1, value=3)
334         self.lv1Tres.pack(side="left")
335
336         #-----
337         #progress bar
338         self.frameProgressBar = tk.Frame(self)
339         self.frameProgressBar.pack(fill='both')
340
341         barProgressTitle = tk.Label(self.frameProgressBar, text="Barra de progresso", anchor='w')
342         barProgressTitle.pack(fill='x')
343
344         self.barProgress = ttk.Progressbar(self.frameProgressBar, length=250, orient='horizontal', mode='determinate')
345         self.barProgress.pack(fill='x')
346
347         #-----
348         #tirar foto
349         self.frameFotoCadastrar = tk.Frame(self)
350         self.frameFotoCadastrar.pack(fill='x', anchor='center')
351
352         self.tirarFotoLabelCadastrar = tk.Label(self.frameFotoCadastrar)
353
354         self.tirarFotoLabelCadastrar.pack()
355
356         frameBottom = tk.Frame(self)
357         frameBottom.pack(fill='x', anchor='center')
358
359         tirarFotoBtn = ttk.Button(frameBottom, text="Tirar fotos", width=15, command=self.tirarFotos)
360         tirarFotoBtn.pack(side='left', fill='x', padx=100)
361
362         button = ttk.Button(frameBottom, text="Ir para home", width=15, command=reinicia_gui)
363         button.pack(side='right', fill='x', padx=100)
364
365         labelBottom = tk.Label(self)
366         labelBottom.pack(side='bottom')
367
368         labelBottom = tk.Label(labelBottom, text="Ministério do Meio Ambiente", font=controller.cabecalho_font)
369         labelBottom.pack(side="bottom", fill="x", pady=10)

```

```

379 #USADO PARA LIMPAR E RODAR DE NOVO
380 def refresh(self):
381
382     self.nomeEntry.delete(0, "end")
383     self.lv1.set(0)
384
385     self.barProgress.pack_forget()
386     self.barProgress = ttk.Progressbar(self.frameProgressBar, length=250, orient='horizontal', mode='determinate')
387     self.barProgress.pack(fill='x')
388
389     self.frameFotoCadastrar.pack_forget()
390     self.frameFotoCadastrar = tk.Frame(self)
391     self.frameFotoCadastrar.pack(fill='x', anchor='center')
392
393     self.tirarFotoLabelCadastrar = tk.Label(self.frameFotoCadastrar)
394     self.tirarFotoLabelCadastrar.pack()
395
396 def tirarFotos(self):
397
398     checkName = self.checkName()
399     checkNiveis = self.checkNiveis()
400
401     if checkName == None:
402         #print("Por favor informe seu nome!")
403         messagebox.showerror("Error", "Por favor, informe seu nome!")
404     elif checkName == False:
405         #print("Por favor informe seu nome, apenas letras!")
406         messagebox.showerror("Error", "Por favor, apenas letras são permitidas no campo nome!")
407     else:
408
409         if checkNiveis == None:
410             #print("Por favor informe seu nível!")
411             messagebox.showerror("Error", "Por favor, informe seu nível!")
412         elif checkNiveis == False:
413             #print("Por favor informe seu nível de acesso corretamente!")
414             messagebox.showerror("Error", "Por favor, informe seu nível de acesso corretamente!")
415         else:
416             print('ok')
417             # initBarProgress(self.nomeEntry.get(), self.niveisEntry.get())
418             # self.capturarFace(self.nomeEntry.get(), self.niveisEntry.get())
419             self.capturarFace(self.nomeEntry.get(), self.lv1.get())

```

```

421 def checkName(self):
422
423     checkName = str(self.nomeEntry.get())
424     checkFlag = None
425
426     if len(checkName) == 0:
427         return checkFlag
428     else:
429         for eachLetter in checkName:
430             if eachLetter.isalpha() == True or ord(eachLetter) == 32:
431                 checkFlag = True
432             else:
433                 checkFlag = False
434                 self.nomeEntry.delete(0, 'end')
435
436         return checkFlag
437
438 def checkNiveis(self):
439
440     checkNivel = str(self.lv1.get())
441     checkFlag = None
442
443     if len(checkNivel) == 0:
444         return checkFlag
445     else:
446         for eachLetter in checkNivel:
447             if eachLetter.isnumeric() == True and (eachLetter == '1' or eachLetter == '2' or eachLetter == '3') and len(eachLetter) <= 3:
448                 checkFlag = True
449             else:
450                 checkFlag = False
451                 # self.niveisEntry.delete(0, 'end')
452                 break
453
454         return checkFlag

```

```

456 def capturarFace(self, _name, _niveis):
457
458     classificador = cv2.CascadeClassifier("{}\haarcascade-frontalface-default.xml".format(Path().reconhecedores))
459     classificadorOlho = cv2.CascadeClassifier("{}\haarcascade-eye.xml".format(Path().reconhecedores))
460
461     amostra = 1
462     numeroAmostras = 30
463
464     name = _name
465     niveis = _niveis
466     largura, altura = 200, 200
467     print("Capturando...")
468
469     messagebox.showwarning("Aviso", "Pronto?")
470     id = ReferenceFile().writeReference(name, niveis)
471
472     messagebox.showwarning("Mensagem de atenção", "Por favor, olhe para a camera e espere a barra de progresso terminar.")
473     self.barProgress.start()
474     self.cameraCadastrar = cv2.VideoCapture(0)
475
476     while (True):
477         time.sleep(1)
478         conectado, imagem = self.cameraCadastrar.read()
479         imagemCinza = cv2.cvtColor(imagem, cv2.COLOR_BGR2GRAY)
480         facesDetectadas = classificador.detectMultiScale(imagemCinza,
481                                                         scaleFactor=1.5,
482                                                         minSize=(150, 150))
483

```

```

485 =         for (x, y, l, a) in facesDetectadas:
486 =             cv2.rectangle(imagem, (x, y), (x + l, y + a), (0, 0, 255), 2)
487 =             regiao = imagem[y:y + a, x:x + l]
488 =             regiaoCinzaOlho = cv2.cvtColor(regiao, cv2.COLOR_BGR2GRAY)
489 =             img = PIL.Image.fromarray(regiaoCinzaOlho)
490 =             imgtk = ImageTk.PhotoImage(image=img)
491 =             self.tirarFotoLabelCadastrar.imgtk = imgtk
492 =             self.tirarFotoLabelCadastrar.configure(image=imgtk)
493 =
494 =
495 =             olhosDetectados = classificadorOlho.detectMultiScale(regiaoCinzaOlho)
496 =             for (ox, oy, ol, oa) in olhosDetectados:
497 =                 cv2.rectangle(regiao, (ox, oy), (ox + ol, oy + oa), (0, 255, 0), 2)
498 =                 if np.average(imagemCinza) > 110:
499 =                     self.barProgress.update()
500 =                     self.barProgress["maximum"] = numeroAmostras
501 =                     self.barProgress["value"] = amostra
502 =                     imagemFace = cv2.resize(imagemCinza[y:y + a, x:x + l], (largura, altura))
503 =                     cv2.imwrite("{} / pessoa-{}-{}.jpg".format(Path().dataset, str(id), str(amostra)), imagemFace)
504 =                     print("Foto {} capturada".format(str(amostra)))
505 =                     amostra += 1
506 =
507 =             cv2.waitKey(1)
508 =             if (amostra >= numeroAmostras + 1):
509 =                 break
510 =
511 =         print("Fases Capturadas!!")
512 =         self.cameraCadastrar.release()
513 =         cv2.destroyAllWindows()
514 =         self.barProgress.stop()
515 =         self.barProgress.destroy()
516 =         messagebox.showinfo("Pronto!", "Obrigado {}, suas fotos foram tiradas!\nVolte para home e entre no sistema.".format(self.nomeEntry.get()))
517 =         Treinamento().runTreinamento()
518 =
519 =
520 =
521 = def inicia_gui():
522 =
523 =     global root
524 =     root = ReconhecimentoFacial()
525 =     root.mainloop()
526 =
527 =
528 = if __name__ == '__main__':
529 =     def reinicia_gui():
530 =         root.destroy()
531 =         inicia_gui()
532 =
533 =     inicia_gui()

```

- CapturaFace.py

```

19 - class ReferenceFile:
20 -
21 -     def __init__(self):
22 -
23 -         self.location = Path().data_store
24 -         self.fileCreated = None
25 -
26 -     def writeReference(self, name, niveis):
27 -
28 -         self.name = name
29 -         self.niveis = niveis
30 -
31 -         print("{} - {}".format(self.name, str(self.niveis)))
32 -         currentId = self.getId()
33 -         print(currentId)
34 -
35 -         file = open("{}\\id_with_reference.csv".format(self.location), mode="a", encoding="utf-8")
36 -         file.write("{};{};\n".format(self.name, currentId, self.niveis))
37 -         file.close()
38 -         return currentId
39 -
40 -
41 -     def getId(self):
42 -
43 -         self.getFile()
44 -         nextId = 0
45 -         file = open("{}\\id_with_reference.csv".format(self.location), mode="r", encoding="utf-8")
46 -         registros = file.readlines()
47 -
48 -         if len(registros) == 1:
49 -             print("só cabeçalho")
50 -             nextId = 1
51 -             return nextId
52 -             print(nextId)
53 -         else:
54 -             lastId = int(registros[-1].split(";")[1])+1
55 -             nextId = lastId
56 -             return nextId
57 -             print(nextId)
58 -
59 -     def getFile(self):
60 -
61 -         for eachFile in os.listdir(self.location):
62 -             if "id_with_reference.csv" in eachFile:
63 -                 self.fileCreated = True
64 -             else: pass
65 -
66 -         if self.fileCreated == True:
67 -             print("Ja existe")
68 -             pass
69 -
70 -         elif self.fileCreated == None:
71 -             with open("{}\\id_with_reference.csv".format(self.location), mode="w", encoding="utf-8") as file:
72 -                 file.write("nome;id;niveis\n")
73 -                 print("Arquivo criado")
74 -         else:
75 -             print("arquivo com problema")

```

- Exceptions.py

```
11 #classe criada para verificar arquivos mandatórios
12 class Verify:
13
14     def __init__(self):
15
16         self.picturesFolder = Path().dataset
17         self.filesFolder = Path().reconhecedores
18         self.model = Path().models
19
20     #verifica se existe fotos capturadas para treinar o modelo
21     def verifyDataset(self):
22
23         datasetAmount = os.listdir(self.picturesFolder)
24         if len(datasetAmount) > 10: return True
25         else: return False
26
27     #verifica se arquivo do modelo foi criado
28     def verifyClassifier(self):
29
30         files = os.listdir(self.model)
31         checkClassifier = False
32         for eachFile in files:
33             if 'classificadorLBPH.yml' in eachFile:
34                 checkClassifier = True
35             else: pass
36
37         return checkClassifier
38
39     #verifica se arquivos de deteccao foram criados
40     def verifyDetectors(self):
41
42         files = os.listdir(self.filesFolder)
43         if 'deteccaooface.xml' and 'haarcascade-frontalface-default.xml' and 'haarcascade-frontalface-default.xml' in files:
44             return True
45         else: return False
46
47     def verifyFiles(self):
48         check = False
49         while check == False:
50             if self.verifyDataset() == False or self.verifyClassifier() == False or self.verifyDetectors == False:
51                 check = False
52                 return False
53             else:
54                 check = True
55                 return True
```

- Treinamento.py

```
13 class Treinamento:
14
15     def __init__(self):
16
17         self.dataset = Path().dataset
18         self.model = Path().models
19         self.verifyFiles = Verify()
20         self.lbph = cv2.face.LBPHFaceRecognizer_create()
21
22     def getImagemComId(self):
23
24         caminhos = [os.path.join('{}'.format(self.dataset), f) for f in os.listdir('{}'.format(self.dataset))]
25         faces = []
26         ids = []
27
28         for caminhoImagem in caminhos:
29             imagemFace = cv2.cvtColor(cv2.imread(caminhoImagem), cv2.COLOR_BGR2GRAY)
30             id = int(os.path.splitext(caminhoImagem)[-1].split('.')[1])
31             ids.append(id)
32             faces.append(imagemFace)
33         return np.array(ids), faces
34
35     def treinarMaquina(self):
36
37         ids, faces = self.getImagemComId()
38
39         checkFile = True
40
41         for each in os.listdir(self.model):
42             if "classificadorLBPH.yml" in each:
43                 os.remove('{}\classificadorLBPH.yml'.format(self.model))
44             else:
45                 pass
46
47         try:
48             print("Criando treinamento")
49             self.lbph.train(faces, ids)
50             self.lbph.write("{}\classificadorLBPH.yml".format(self.model))
51
52             print("Treinamento concluido")
53         except:
54             print("Erro no treinamento do modelo")
55
56     def runTreinamento(self):
57
58         if self.verifyFiles.verifyDataset() == True:
59
60             self.treinarMaquina()
61             return True
62
63         else:
64             print('Voce nao tem um dataset de treino! Por favor, cadastre primeiro')
65             return False
```

- Dict.py

```

7 from Configuration import Path
8
9 class Dictionary:
10
11     def __init__(self):
12
13         self.path = Path().data_store+"\\id_with_reference.csv"
14         self.textFilePath = Path().data_store
15         self.dictionary_id = self.getId()
16         self.dictionary_lvl = self.getLvl()
17
18     def getId(self):
19
20         """This function returna dictionary of ids
21
22         Returns:
23         | (str) -- A dictionary of ids
24         """
25
26         file = open("{}".format(self.path), mode="r", encoding="utf-8")
27         registros = file.readlines()
28
29         dict_id_with_reference = {}
30
31         for each in registros:
32             if "nome" in each: pass
33             else:
34                 registroSplited = each.split(";")
35                 dict_id_with_reference[registroSplited[1].replace('\n','')] = registroSplited[0]
36
37         return dict_id_with_reference

```

```

39     def getLvl(self):
40
41         """This function returna dictionary of levels
42
43         Returns:
44         | (str) -- A dictionary of levels
45         """
46
47         file = open("{}".format(self.path), mode="r", encoding="utf-8")
48         registros = file.readlines()
49
50         dict_lvl_with_reference = {}
51
52         for each in registros:
53             if "nome" in each: pass
54             else:
55                 registroSplited = each.split(";")
56                 dict_lvl_with_reference[registroSplited[1].replace('\n','')] = registroSplited[2]
57
58         return dict_lvl_with_reference
59
60     def getText(self, lvl):
61
62         """This function will provide the text for specific lvl informed
63
64         Arguments:
65         | lvl (str) -- A string
66
67         Returns:
68         | (str) -- A string that corresponds a specific text file
69         """
70
71         listaOfLines = open("{}\texto{}.txt".format(self.textFilePath, lvl), mode="r", encoding="utf-8")
72
73         return listaOfLines.read()

```

- Configuration.py

```

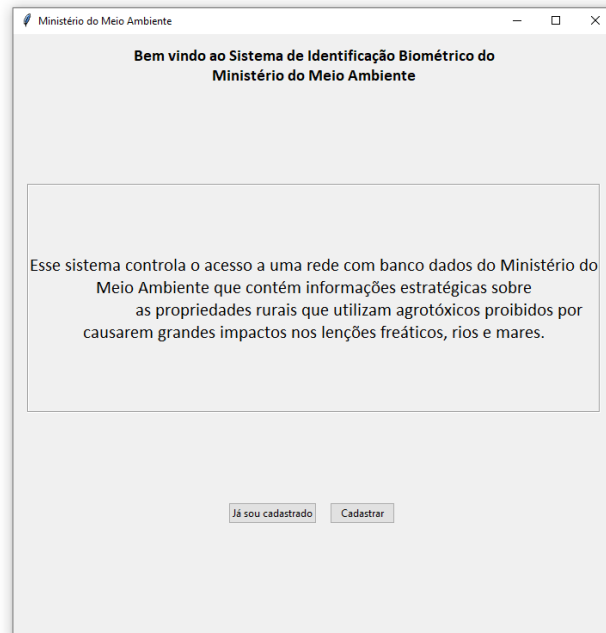
15 import requests
16 import inspect
17 import argparse
18 import time
19
20 class Path:
21
22     def __init__(self):
23
24         #-- ACESSO RÁPIDO A DIRETORIOS USADOS
25         self.data_store = os.path.dirname(os.path.abspath(inspect.getfile(inspect.currentframe()).replace("source_code", "config")))
26         self.dataset = os.path.dirname(os.path.abspath(inspect.getfile(inspect.currentframe()).replace("source_code", "fotos")))
27         self.reconhecedores = os.path.dirname(os.path.abspath(inspect.getfile(inspect.currentframe()).replace("source_code", "reconhecedores")))
28         self.models = os.path.dirname(os.path.abspath(inspect.getfile(inspect.currentframe()).replace("source_code", "models")))
29         self.source_code = os.path.dirname(os.path.abspath(inspect.getfile(inspect.currentframe())))

```

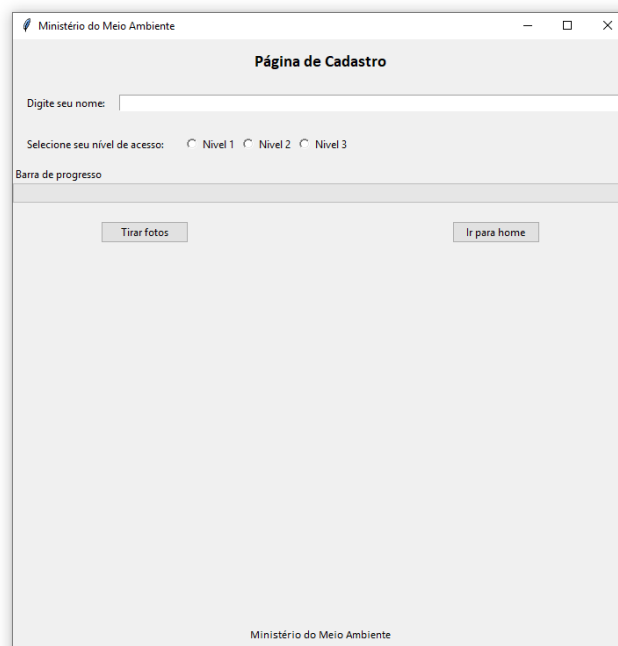
6. APRESENTAÇÃO DA APLICAÇÃO EM FUNCIONAMENTO

6.1. Sistema

Para iniciar o sistema, basta somente executar o script main.py e a seguinte janela será aberta:



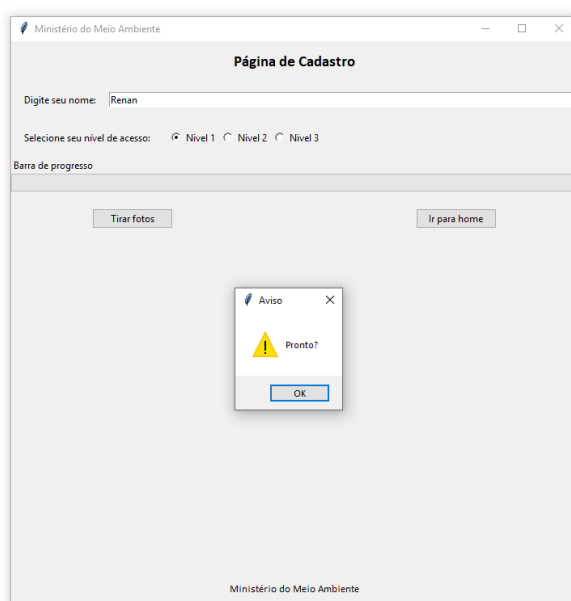
Antes de entrar no sistema é necessário efetuar um cadastro, clicando em “Cadastrar” o sistema apresentará a seguinte tela:



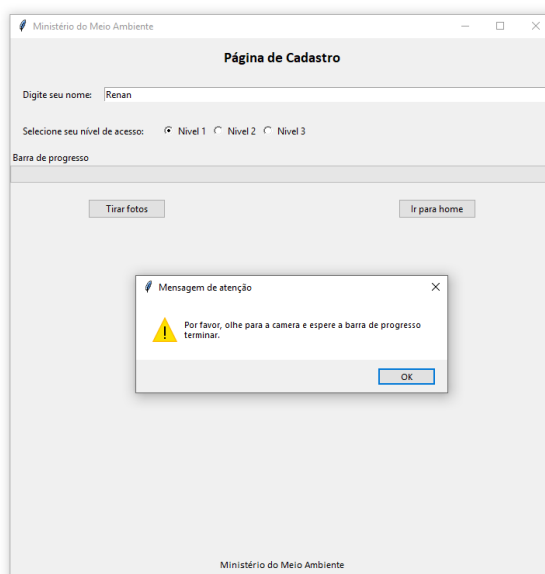
Na “Página de Cadastro” é solicitado algumas informações obrigatórias para o usuário como:

- **Digite seu nome** – podendo digitar somente letras e não permite cadastrar com esse campo em branco
- **Selecione seu nível de acesso** – permitido selecionar somente um nível, não permite cadastrar com esse campo em branco

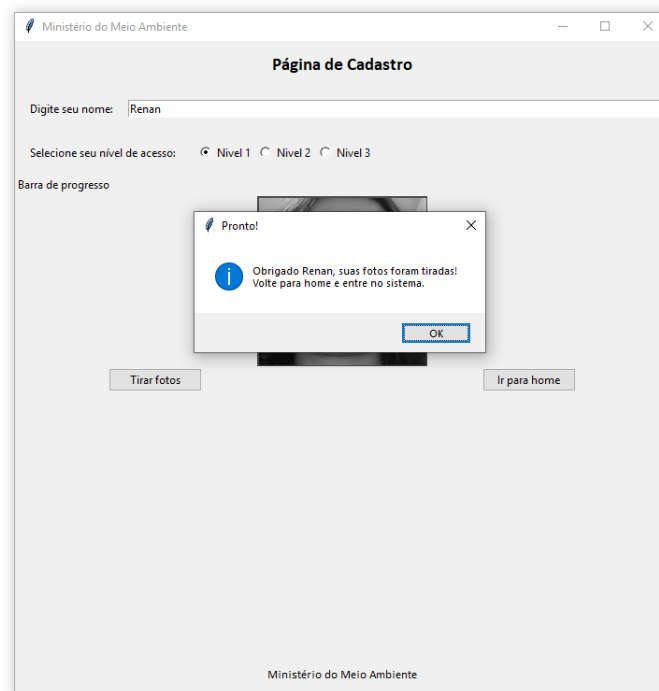
Após preencher todas as informações necessárias, é possível clicar no botão “**Tirar fotos**” e a seguinte tela aparecerá:



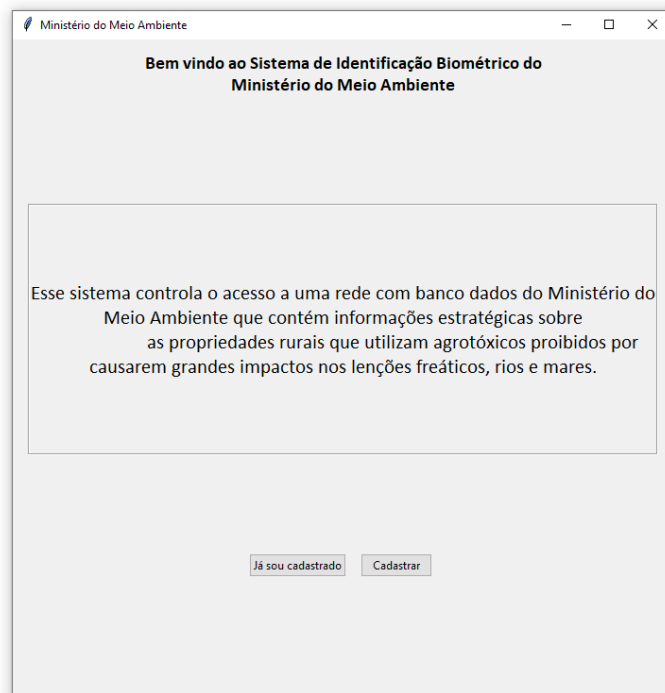
Ao clicar em “**ok**” o sistema mostrará uma mensagem de atenção:



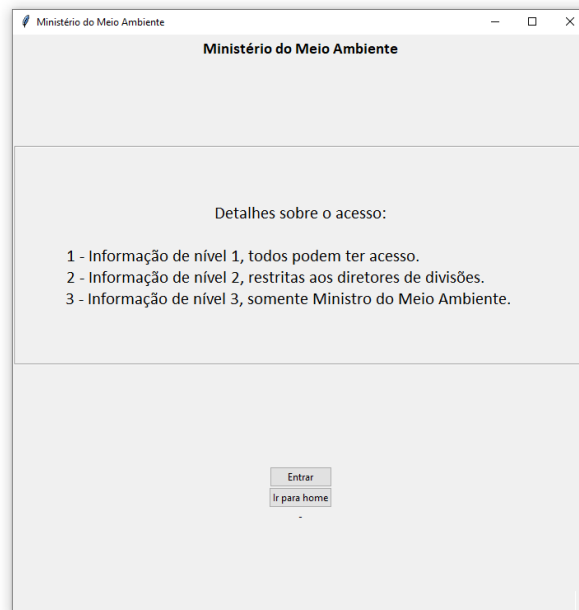
O sistema irá tirar 30 fotos e mostra-lás na aplicação e logo em seguida mostrará a seguinte mensagem:



Conforme a message, o usuário deve retornar a pagina “**Home**” clicando em “Ir para a home” logo em seguida de clicar em “**OK**”.

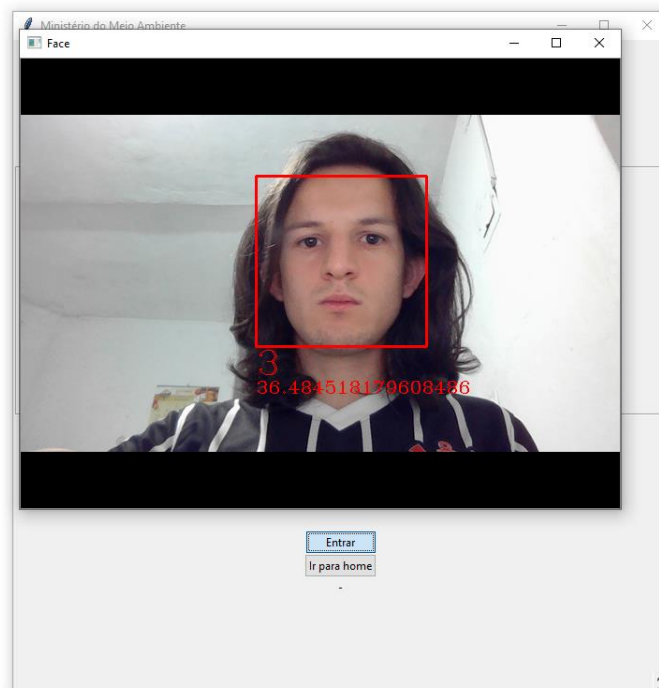


Uma vez que o usuário tenha se cadastrado com sucesso, basta clicar em “**Já sou cadastrado**” e a seguinte tela aparecerá:

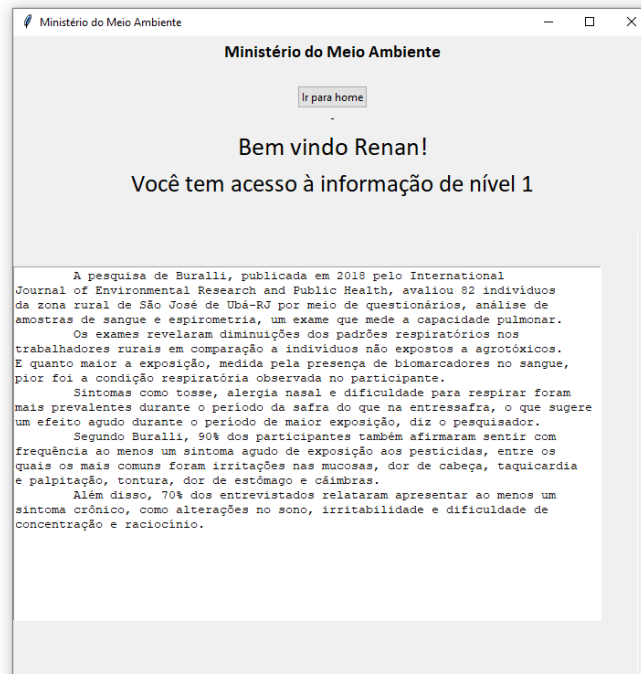


Algumas informações relacionadas ao nível de acesso e respectivas informações aparecerão, uma vez que todos os requerimentos estejam de acordo, o usuário pode clicar em “**Entrar**” e então entrar no sistema.

Ao clicar no botão “**Entrar**”, uma câmera aparecerá para reconhecer a face do usuário:



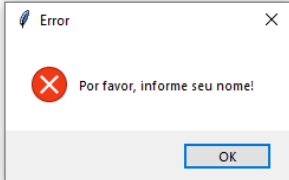
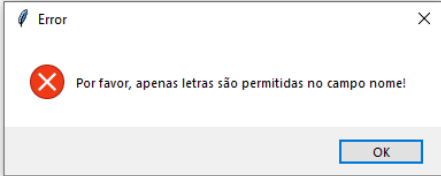
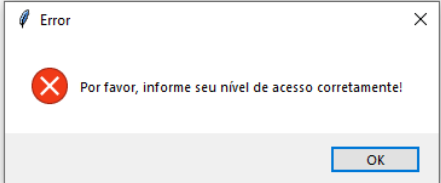
Uma vez que o usuário tenha sido reconhecido corretamente, informações referentes ao nível de acesso do usuário reconhecido estarão disponíveis:



Desse momento em diante, o usuário pode retornar a “**Home**” clicando em “**Ir para home**” e cadastrar novos usuários ou até mesmo clicando no “**X**” para sair do sistema.

6.2. Exceções

Exceções foram criadas para garantir o bom funcionamento do sistema, abaixo mensagens de erro no mento de cadastro do usuário:

IMAGEM	FUNÇÃO
	Exceção criada para garantir que o nome do usuário seja informado no momento do cadastro.
	Exceção criada para garantir que somente letras sejam informadas no momento de cadastro do usuário.
	Exceção criada para garantir que o nível de acesso seja informado no momento do cadastro do usuário.

7. BIBLIOGRAFIA

Dicionario Informal, Biometria, 1. Biometria.

Disponível em: <<https://www.dicionarioinformal.com.br/biometria/>>.

TecMundo, Como funciona o reconhecimento de voz? Disponível em:

<<https://www.tecmundo.com.br/curiosidade/3144-como-funciona-o-reconhecimento-de-voz-.htm>>.

Noginfo, ATIVIDADE PRATICA SUPERVISIONADA, PROPOSTA DO TRABALHO. Disponível em:

<http://noginfo.salaead.com.br/pluginfile.php/6171/mod_resource/content/0/AP_S%20-%205o%20e%206o%20CC%20-%202019.pdf>.

Notícias R7, Tecnologia de reconhecimento facial se populariza e levanta debate.

Disponível em: <<https://noticias.r7.com/tecnologia-e-ciencia/tecnologia-de-reconhecimento-facial-se-populariza-e-levanta-debate-21072018>>.

Sick, Sensores de Cor. Disponível em:

<http://www.sick.com/group/en/home/products/product_portfolio/registration_sensors

/pages/colour_sensors.aspx>. Acesso em: 10 out. 2012.

8. FICHA DE ATIVIDADES PRÁTICAS SUPERVISIONADAS