7º Periodo Noturno – Engenharia da Computação

Nome: João Pedro Cunha – 4200988

Lucas Salaro – 4200978

Luiz Eduardo Marques – 4200974

Otávio Rodrigues da Silva - 4200992

**Documentação do Projeto  
Analisador de Postura com Python**

São Carlos, 01 de Junho de 2025

**Introdução**

O projeto Python Posture Analyzer Software é um software desenvolvido em Python que tem como objetivo analisar a postura de uma pessoa utilizando processamento de imagens e visão computacional. Ele detecta pontos do corpo humano e calcula ângulos que permitem avaliar a postura em tempo real ou a partir de imagens.

O projeto foi estruturado utilizando o padrão de arquitetura MVC (Model-View-Controller), que separa a lógica de dados, a interface e o controle, tornando o código mais organizado, escalável e de fácil manutenção.

Sumário

[1. Arquitetura do Projeto 3](#_Toc199683798)

[1.1. MVC – Model, View e Controller 3](#_Toc199683799)

[2. Classes e Objetos 3](#_Toc199683800)

[3. Funcionalidades 3](#_Toc199683801)

[4. Pré-Requisitos 4](#_Toc199683802)

[5. Instalação 4](#_Toc199683803)

[6. Utilização 4](#_Toc199683804)

[7. Diretrizes para captura de imagem 4](#_Toc199683805)

[8. Estrutura do projeto 5](#_Toc199683806)

[9. Logs 5](#_Toc199683807)

[10. Método Construtor 6](#_Toc199683808)

[11. Métodos de Instância 6](#_Toc199683809)

[12. Variáveis Estáticas 6](#_Toc199683810)

[13. Métodos Estáticos 6](#_Toc199683811)

[14. Encapsulamento 7](#_Toc199683812)

[15. Herança 7](#_Toc199683813)

[16. Polimorfismo 7](#_Toc199683814)

[17. Composição 7](#_Toc199683815)

[18. Conclusão 8](#_Toc199683816)

## Arquitetura do Projeto

## 1.1. MVC – Model, View e Controller

* Model: Responsável pela lógica de processamento dos dados, como cálculos dos ângulos, validações e manipulação das informações obtidas da imagem.
* View: Cuida da interface gráfica do usuário (GUI), construída com Tkinter, permitindo interações simples como carregar imagens, iniciar análises e exibir resultados.
* Controller: Intermedia as ações do usuário na View, chamando os métodos necessários do Model e atualizando a interface.

## Classes e Objetos

O projeto possui as seguintes principais classes:

* Main: Responsável por inicializar e executar o sistema.
* App: Classe que representa a janela principal da aplicação, configurando a interface e seus elementos.
* PostureAnalyzer: Localizada no Model, é responsável pelos cálculos dos ângulos dos pontos do corpo.
* ImageProcessor: Responsável por carregar, tratar e preparar as imagens para análise.
* Controller: Faz a comunicação entre View e Model, recebendo ações da interface e disparando os métodos de processamento.

## Funcionalidades

* Captura de vídeo em tempo real via webcam
* Detecção de postura usando MediaPipe Pose
* Análise de ângulos dos ombros e quadril
* Alertas visuais e sonoros para postura incorreta
* Sugestões de correção de postura
* Estatísticas detalhadas de uso
* Exportação de dados em CSV e JSON
* Interface gráfica moderna e intuitive

## Pré-Requisitos

* Python 3.8 ou superior
* Webcam
* Dependências

## Instalação

* **Clone o repositório:**
  + git clone <https://github.com/JoaoOliveiraaa/python-posture-analyzer-software.git>

cd analise-postura

* **Instale as dependências:**
  + pip install -r requirements.txt
    - **Composto por:**
    - opencv-python==4.8.1.78
    - mediapipe==0.10.8
    - numpy==1.26.2
    - Pillow==10.1.0
    - mysql-connector-python==8.0.33

## Utilização

* Execute o programa principal:
  + python main.py
* Na interface gráfica:
  + Clique em "Iniciar Monitoramento" para começar
  + Ajuste as configurações da câmera conforme necessário
  + Monitore sua postura em tempo real
  + Consulte as estatísticas e exporte os dados quando desejar

## Diretrizes para captura de imagem

Para obter resultados precisos na análise postural, siga as seguintes recomendações ao capturar imagens:

* Uma Pessoa por Imagem: Certifique-se de que apenas uma pessoa esteja presente na imagem.
* Formato da Imagem: Utilize imagens no formato JPEG (.jpg).
* Altura da Câmera: Posicione a câmera na altura do quadril do paciente para uma visão lateral adequada.
* Orientação da Câmera: Garanta que a câmera esteja posicionada perpendicularmente ao chão para capturar uma imagem lateral precisa.

## Estrutura do projeto

├── controllers/

│ └── controller.py # Lógica de controle e processamento

├── models/

│ └── model.py # Gerenciamento de dados e banco de dados

├── views/

│ └── view.py # Interface gráfica

├── main.py # Ponto de entrada do programa

├── requirements.txt # Dependências do projeto

└── README.md # Documentação

## Logs

O sistema mantém logs detalhados de:

* Operações do banco de dados
* Erros e exceções
* Eventos importantes
* Estatísticas de uso

## Método Construtor

Todas as classes possuem um método construtor \_\_init\_\_, que inicializa seus atributos principais.

Exemplo da classe App:  
class App(tk.Tk):  
 def \_\_init\_\_(self):  
 super().\_\_init\_\_()  
 self.title('Posture Analyzer')  
 self.geometry('800x600')

## Métodos de Instância

Os métodos de instância são utilizados para executar ações específicas dos objetos, como:

* Carregar imagem
* Processar pontos do corpo
* Calcular ângulos
* Exibir resultados na interface

Exemplo:

def calculate\_angle(self, point1, point2, point3):  
 # Cálculo do ângulo entre três pontos

## Variáveis Estáticas

O projeto não faz uso direto de variáveis estáticas, mas poderia ser aplicado, por exemplo, para manter o número de análises feitas ou uma configuração compartilhada entre objetos.

## Métodos Estáticos

Utiliza métodos estáticos principalmente para funções matemáticas e auxiliares que não dependem do estado do objeto.

Exemplo:

@staticmethod  
def distance(point1, point2):  
 return ((point1.x - point2.x) \*\* 2 + (point1.y - point2.y) \*\* 2) \*\* 0.5

## Encapsulamento

Os atributos das classes são protegidos utilizando a convenção de underscore (\_) para indicar que são privados ou protegidos.

Exemplo:  
class PostureAnalyzer:  
 def \_\_init\_\_(self):  
 self.\_landmarks = None

## Herança

O projeto faz uso de herança, especialmente na classe App, que herda de tk.Tk para construir a interface gráfica.

Exemplo:

class App(tk.Tk):  
 ...

## Polimorfismo

O polimorfismo é aplicado quando diferentes métodos ou objetos podem executar a mesma função de formas distintas. No contexto do projeto, isso ocorre na manipulação dos eventos e métodos da interface, onde diferentes botões disparam funções com comportamentos variados.

## Composição

A composição está presente na integração entre as classes:

* A classe App possui uma instância de Controller.
* Controller instancia PostureAnalyzer e ImageProcessor, conectando a interface (View) com a lógica (Model).

Exemplo:  
class Controller:  
 def \_\_init\_\_(self, app):  
 self.app = app  
 self.analyzer = PostureAnalyzer()

## Conclusão

O projeto Posture Analyzer é uma aplicação que demonstra conceitos fundamentais de Programação Orientada a Objetos (POO) e arquitetura MVC. Seu código é organizado em módulos que separam interface, lógica de processamento e controle. Além disso, aplica conceitos como encapsulamento, composição, herança e polimorfismo, alinhados às boas práticas de desenvolvimento em Python.