**Logotipo

El contenido generado por IA puede ser incorrecto.DOCUMENTACIÓN TÉCNICA**

**Proyecto**: WorkAware

**Versión**: 1.0

**Autor**: Juan David Rivaldo Diaz Sierra

**Correo**: juandavidrivaldo1@gmail.com

**Año**: 2025

1**. Propósito y Alcance**

WorkAware es una aplicación de escritorio multiplataforma desarrollada en Python, orientada a la gestión eficiente de tareas y al monitoreo de la postura del usuario frente al computador. Su objetivo es mejorar la productividad y la salud digital, integrando notificaciones inteligentes y análisis en tiempo real de la postura corporal.

2. **Arquitectura General**

WorkAware está estructurada en módulos independientes que interactúan entre sí a través de interfaces bien definidas.

La arquitectura sigue el patrón MVC (Modelo-Vista-Controlador) adaptado a aplicaciones de escritorio con Flet.

2.1 **Diagrama de Arquitectura**

* Interfaz de Usuario (Flet): Pantallas, navegación, controles y visualización.
* Lógica de Negocio: Gestión de tareas, monitoreo de postura, notificaciones.
* Persistencia: Base de datos SQLite gestionada con SQLAlchemy.
* Recursos: Imágenes, sonidos, iconos.

3. **Estructura de Carpetas y Archivos**

workaware/

│

├── app/

│ ├── db\_actions.py

│ ├── notifier.py

│ ├── posture\_monitor.py

│ ├── session\_logger.py

│ └── ...

│

├── detector/

│ └── detector.py

│

├── Interface/

│ ├── main\_screen.py

│ ├── add\_update\_screen.py

│ ├── expand\_task\_screen.py

│ └── ...

│

├── models/

│ └── task.py

│

├── assets/

│ └── icon.png

│

├── sounds/

│ └── ...

│

├── database.py

├── devices.db

├── main.py

├── README.md

├── LICENSE

├── requirements.txt / pyproject.toml

└── ...

4. **Descripción de Módulos Principales**

4.1 **Módulo de Gestión de Tareas**

* *Propósito*: Permite crear, editar, eliminar y visualizar tareas.
* *Campos de tarea:* Título, descripción, prioridad, estado, tipo, fecha de vencimiento, duración estimada.
* *Funcionalidades:* 
  + CRUD de tareas.
  + Filtros avanzados por prioridad, estado y tipo.
  + Visualización en tarjetas con iconos y colores distintivos.
* *Flujo:* 
  + El usuario accede a la pantalla principal.
  + Puede añadir, editar o eliminar tareas.
  + Puede filtrar tareas según criterios seleccionados.

4.2 **Módulo de Monitoreo de Postura**

* *Propósito*: Detecta posturas inadecuadas usando visión por computadora.
* T*ecnologías*:
  + MediaPipe para landmarks.
  + OpenCV para captura y visualización.
* *Flujo*:
  1. El usuario activa el switch de monitoreo.
  2. Se inicia la captura de video y el análisis en tiempo real.
  3. Si se detecta mala postura, se envía una notificación y alerta sonora.
  4. El usuario puede detener el monitoreo en cualquier momento.
  5. Criterios de mala postura:
  6. Desplazamiento hacia adelante de los hombros respecto a la boca.
  7. Altura relativa de los hombros por debajo de un umbral.

4.3 **Módulo de Notificaciones y Alertas**

* *Propósito*: Informar al usuario sobre eventos importantes (mala postura, tareas, etc.).
* *Tecnologías:* 
  + Plyer para notificaciones de escritorio.
  + Pygame para alertas sonoras.
* *Características:* 
  + Notificaciones visuales y sonoras.
  + Control de frecuencia para evitar saturación.
  + Personalización de mensajes y sonidos.

4.4 **Interfaz de Usuario**

* *Framework:* Flet.
* *Pantallas:* 
  + Splash screen (pantalla de carga con logo).
  + Pantalla principal (gestión de tareas, filtros, switch de postura).
  + Pantalla de añadir/editar tarea.
  + Pantalla de detalles de tarea.
* *Elementos visuales:* 
  + Iconos personalizados.
  + Diseño responsive.
  + Uso de colores y tipografía para mejorar la experiencia.

4.5 **Persistencia y Base de Datos**

* *Motor:* SQLite.
* *ORM:* SQLAlchemy.
* *Tablas:* 
  + tasks: almacena todas las tareas con sus atributos.
* *Migración:* 
  + Creación automática de tablas si no existen.

**5. Flujos de Usuario**

5.1 **Flujo de Gestión de Tareas**

1. El usuario abre la app y ve la pantalla principal.
2. Puede añadir una nueva tarea, editar o eliminar tareas existentes.
3. Puede filtrar tareas por prioridad, estado o tipo.
4. Puede expandir una tarea para ver detalles completos.

5.2 **Flujo de Monitoreo de Postura**

1. El usuario activa el switch de monitoreo.
2. El sistema inicia la cámara y comienza el análisis.
3. Si se detecta mala postura, se muestra una notificación y se reproduce un sonido.
4. El usuario puede detener el monitoreo en cualquier momento.

**6. Dependencias y Requisitos Técnicos**

* Python >= 3.12
* Flet >= 0.28.3
* SQLAlchemy >= 2.0.41
* Plyer >= 2.1.0
* Pygame >= 2.6.1
* Mediapipe >= 0.10.21
* OpenCV-Python >= 4.11.0.86
* certifi >= 2025.4.26

**7. Consideraciones Técnicas y de Seguridad**

* El icono de la ventana solo puede personalizarse en el ejecutable, no en modo desarrollo.
* Las notificaciones dependen del sistema operativo y pueden variar en comportamiento.
* El monitoreo de postura usa hilos y debe sincronizarse correctamente al navegar entre pantallas.
* El splash screen se muestra solo al inicio, antes de la pantalla principal.
* El acceso a la cámara requiere permisos del sistema.
* El ejecutable puede ser firmado digitalmente para mayor seguridad y autoría.

**8. Autoría y Licenciamiento**

* Todos los archivos fuente incluyen encabezados con nombre, año y derechos.
* Archivo LICENSE con licencia MIT y nombre del autor.
* README y pyproject.toml con datos de autoría.
* Icono y nombre personalizados en la app y ejecutable.
* Repositorio público en GitHub como evidencia de autoría.

**9. Anexos**

9.1 Ejemplo de Código: Detección de Postura

python

import cv2

import mediapipe as mp

class PostureDetector:

def \_\_init\_\_(self):

self.mp\_holistic = mp.solutions.holistic

self.holistic = self.mp\_holistic.Holistic()

# ...

def process\_frame(self, frame):

# Procesamiento y lógica de detección

pass

9.2 Diagrama de Flujo (puedes agregar imágenes o diagramas en Word)

**10. Contacto**

**Autor**: Juan David Rivaldo Diaz Sierra

**Correo**: [juandavidrivaldo1@gmail.com](mailto:juandavidrivaldo1@gmail.com)

**LinkedIn**: <https://www.linkedin.com/in/juan-david-rivaldo-diaz-sierra-72aa99222>

**11. Próximos pasos del desarrollo**

* Incorporación de sugerencias de descansos activos por tiempo de trabajo (técnica pomodoro, descanso acumulativo (max 30 minutos))
* Adicion de notas y/o comentarios a cada tarea para documentación del proceso
* Vinculación con herramientas como Google calendar para sincronizar tareas y reuniones
* Envio de notificaciones al correo electrónico según tareas por vencer o ya vencidas