Registro Diario de Avances – Modelo basado en IA para detección de somnolencia

**Fecha:** 24/04/2025

**Autor: Tirza Buendia**

**Versión:** 1.0

# 1. Introducción

Este documento detalla el seguimiento diario del Sprint 1 del proyecto modelo basado en IA para detección de somnolencia, centrado en el desarrollo de funcionalidades básicas. Se incluyen actividades diarias, tareas completadas, evidencias de código, un burndown chart y un análisis del rendimiento del equipo.

# 2. Lineamientos de Diseño

Desarrollar una primera versión funcional del modelo basado en IA para detección de somnolencia en conductores que permita a los usuarios:

* Usar detección facial en el modelo.
* Monitorear el cierre ocular.

# 3. Historias de Usuario y Tareas

| Backlog ID | Historia de Usuario | Tiempo Estimado | Fecha de Inicio | Fecha de Finalización |
| --- | --- | --- | --- | --- |
| HU-1 | Detectar rostros con cámara | 8 pts | 24/04/2025 | 29/04/2025 |
| HU-2 | Monitoreo de Cierre Ocular | 5 pts | 30/04/2025 | 23/04/2025 |

# 4. Evidencias de Código

## 4.1 Historia de Usuario 1 — Detectar rostros con cámara

Archivo: Modelo/captura\_video.py

import cv2

import threading

#AQUÍ ESTABA EL BLOQUE A ANTES

captura = cv2.VideoCapture(0) # 0 = cámara web de laptop

#para capturar video

if not captura.isOpened():

print("La cámara no pudo abrirse. Revisa si está en uso o no conectada. captura video")

exit()

else:

print("cámara abierta! captura video")

captura.set(cv2.CAP\_PROP\_FRAME\_WIDTH, 320) #reducción de tamaño de frame para menor latencia

captura.set(cv2.CAP\_PROP\_FRAME\_HEIGHT, 240)

# Variable global para el último frame capturado

frame\_actual = None

lock = threading.Lock() # Para controlar acceso al frame entre hilos

def capturar\_frames\_continuamente():

global frame\_actual

while True:

ret, frame = captura.read()

if ret:

#print("Frame capturado en captura video")

with lock:

frame\_actual = frame.copy() #copia el último frame disponible

else:

print("No se pudo leer el frame en captura video")

# Lanza un hilo que siempre está capturando frames

def iniciar\_hilo\_captura():

hilo = threading.Thread(target=capturar\_frames\_continuamente, daemon=True)

hilo.start()

def obtener\_frame():

with lock:

if frame\_actual is not None:

#print ("se mandó frame de captura video")

return frame\_actual.copy()

else:

print("No se mandó frame de captura video")

return None

Archivo: Modelo/deteccion\_facial.py

import cv2

import mediapipe as mp

from captura\_video import obtener\_frame # Importar la función del otro módulo

#MediaPipe para Landmarks

mp\_puntosFaciales = mp.solutions.face\_mesh

mp\_dibujo = mp.solutions.drawing\_utils

#landmarks

ojo\_derecho = [362,385, 387, 263, 373, 380] # landmarks de ojo derecho

ojo\_izquierdo = [33, 160, 158, 133, 153, 144] #Landmarks de ojo izquierdo

lm\_boca = [0, 13, 14, 17, 61, 291, 314, 317] # Boca (MAR) contorno labios

lm\_cabeza = [1, 10, 152, 151, 234, 454, 200, 423] # Cabeza (ángulo) Línea media y laterales

#para mandar perclos

coordenadas\_ojod=[]

coordenadas\_ojoi=[]

#estilo de landmarks

estilo\_puntos = mp\_dibujo.DrawingSpec(color=(248,243,28), thickness=1)

face\_mesh = mp\_puntosFaciales.FaceMesh(

static\_image\_mode=False,

max\_num\_faces=1,

refine\_landmarks=False,

min\_detection\_confidence=0.5,

min\_tracking\_confidence=0.5

)

def hilo\_procesar\_lm(cola\_frames):

global coordenadas\_ojod, coordenadas\_ojoi

# Continuamente obtener y procesar frames

while True:

frame = obtener\_frame() # Esta es tu función actual que obtiene el frame del video

if frame is None:

continue # Si no hay frame, se omite este ciclo

# Procesar el frame (convertir a RGB y aplicar la detección de landmarks)

frame\_rgb = cv2.cvtColor(frame, cv2.COLOR\_BGR2RGB)

resultados = face\_mesh.process(frame\_rgb)

# Detectar los landmarks en el rostro

if resultados.multi\_face\_landmarks is not None:

for face\_landmarks in resultados.multi\_face\_landmarks:

h, w, \_ = frame.shape

# Limpia antes de llenar

coordenadas\_ojod.clear()

coordenadas\_ojoi.clear()

for idx in ojo\_derecho + ojo\_izquierdo + lm\_boca + lm\_cabeza:

if idx < len(face\_landmarks.landmark):

landmark = face\_landmarks.landmark[idx]

x, y = int(landmark.x \* w), int(landmark.y \* h)

cv2.circle(frame, (x, y), 1, estilo\_puntos.color, estilo\_puntos.thickness)

if idx in ojo\_derecho:

coordenadas\_ojod.append([x, y])

elif idx in ojo\_izquierdo:

coordenadas\_ojoi.append([x, y])

# Empujar el frame procesado a la cola

if not cola\_frames.full():

cola\_frames.put(frame)

print("deteccion\_landmarks funciona")

Archivo: Modelo/captura\_video.py

import tkinter as tk

from PIL import Image, ImageTk

#from deteccion\_facial import procesar\_frame

from captura\_video import iniciar\_hilo\_captura, captura#, obtener\_frame

import cv2

from deteccion\_landmarks import hilo\_procesar\_lm

from analisis\_somnolencia import mandar\_perclos, mandar\_ear

import time

import threading

import queue

cola\_frames = queue.Queue(maxsize=1)

ultimo\_tiempo = time.time() #para calcular FPS

fps = 0

ventana = tk.Tk() #crea la ventana principal de la interfaz

ventana.configure(bg="#FFF5EE") #configura color de fondo de la ventana

ventana.title("Alarma led") #titulo de ventana

ventana.geometry("800x600") #dimensiones de ventana wxh

#label para mostrar flujo de video

video\_label = tk.Label(ventana,bg="#1C1C1C")

video\_label.place( #configura posición y dimensiones

relx=0.065, # 0% desde la izquierda

rely=0.21, # 0% desde arriba

relwidth=0.43, # 43% del ancho total

relheight=0.58 # 58% del alto total

)

#label para mostrar alarma

fuente =("calibri, 24")

alarma\_label = tk.Label(ventana,text="Hola",bg="#000000",fg="#FFFFFF", font=fuente)

alarma\_label.place(

relx=0.56, # 0% desde la izquierda

rely=0.21, # 0% desde arriba

relwidth=0.375, # 43% del ancho total

relheight=0.58 # 58% del alto total

)

#función para mostrar video (se muestra en frames)

def mostrar\_video():

global ultimo\_tiempo, fps

if not cola\_frames.empty():

frame = cola\_frames.get()

tiempo\_actual = time.time()

fps = 1 / (tiempo\_actual - ultimo\_tiempo)

ultimo\_tiempo = tiempo\_actual

print(f"FPS: {fps:.2f}") # Muestra FPS en consola

print("se recibió frame en alarma")

ear = mandar\_ear()

perclos = mandar\_perclos(ear)

print ("EAR: ", ear)

print(f"PERCLOS: {perclos:.2%}")

frame = cv2.cvtColor(frame,cv2.COLOR\_BGR2RGB) #convierte el frame de formato bgr a rgb

img = Image.fromarray(frame) #convierte el array de pixeles NumPy a objeto Image de Pillow

imgtk = ImageTk.PhotoImage(image=img) #convierte la img en una imagen que tkinter muestra dentro de un label

video\_label.imgtk = imgtk

video\_label.configure(image=imgtk) #muestra imagen en pantallla (secuencias de imagen hacen video)

else:

print("⚠️ Frame aún no disponible")

video\_label.after(10, mostrar\_video) #despues de 10 ms vuelve a llamar a la función

#INICIO DE FLUJO DE CÓDIGO

iniciar\_hilo\_captura()

threading.Thread(target=hilo\_procesar\_lm, args=(cola\_frames,), daemon=True).start() #inicia el hilo

mostrar\_video() #inicia función

print("Iniciando interfaz gráfica")

ventana.mainloop() #bucle principal para que se muestre programa

captura.release() #deja de capturar desde la cámara al cerrar app

## 4.2 Historia de Usuario 2 — Monitoreo de Cierre Ocular

Archivo: Modelo/analisis\_somnolencia.py

import numpy as np

from deteccion\_landmarks import coordenadas\_ojoi, coordenadas\_ojod

# Umbral EAR para detectar ojos cerrados

EAR\_UMBRAL = 0.25

# Para PERCLOS

VENTANA\_FRAMES = 150 # ~15 segundos si FPS ≈ 10 cantidad máxima de frames a considerar para el perclos

frames\_cerrados = 0

frames\_totales = 0

ventana\_ear = [] #acumula los ear de las últimas 150 frames

# Para filtrar parpadeos rápidos

frames\_cierre\_consecutivos = 0

MIN\_FRAMES\_CERRADOS = 5 # ~0.5 segundos con FPS ≈ 10 número de frames seguidos que el ear tiene que estar por debajo del umbral para considerarlo como cierre de ojo y no parpadeo

def eye\_aspect\_ratio(coordenadas):

if len(coordenadas) < 6:

return 0.0 # Protección si aún no hay coordenadas

d\_A = np.linalg.norm(np.array(coordenadas[1]) - np.array(coordenadas[5]))

d\_B = np.linalg.norm(np.array(coordenadas[2]) - np.array(coordenadas[4]))

d\_C = np.linalg.norm(np.array(coordenadas[0]) - np.array(coordenadas[3]))

return (d\_A + d\_B) / (2.0 \* d\_C)

def mandar\_ear():

ear\_ojod = eye\_aspect\_ratio(coordenadas\_ojod)

ear\_ojoi = eye\_aspect\_ratio(coordenadas\_ojoi)

ear = (ear\_ojod + ear\_ojoi) / 2

return ear

def mandar\_perclos(ear):

global frames\_totales, frames\_cerrados, ventana\_ear, frames\_cierre\_consecutivos

ventana\_ear.append(ear)

if len(ventana\_ear) > VENTANA\_FRAMES:

ventana\_ear.pop(0) #elimina el valor más antiguo si es que se guardan más de 150 registros de ear

frames\_totales = len(ventana\_ear) #toma valor máximo de 150 pero puede ser menos a penas se inicia el programa

if ear < EAR\_UMBRAL:

frames\_cierre\_consecutivos += 1

else:

if frames\_cierre\_consecutivos >= MIN\_FRAMES\_CERRADOS:

frames\_cerrados += frames\_cierre\_consecutivos

frames\_cierre\_consecutivos = 0

if frames\_totales == 0:

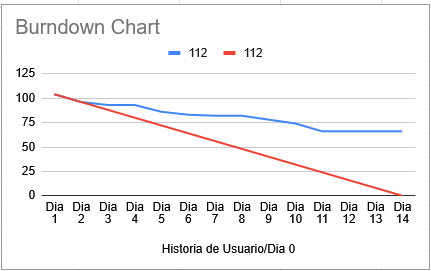
return 0.0

perclos = frames\_cerrados / frames\_totales

return perclos

# 5. Burndown Chart

| Backlog ID | Historia de Usuario | Tiempo Estimado | 24/04 | 25/04 | 26/04 | 27/04 | 28/04 | 29/04 | 30/  04 | 01/  05 | 02/  05 | 03/  05 | 04/  05 | 05/  05 | 06/  05 | 07/  05 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Dia1** | **Dia2** | **Dia3** | **Dia4** | **Dia5** | **Dia6** | **Dia 7** | **Dia 8** | **Dia 9** | **Dia 10** | **Dia 11** | **Dia 12** | **Dia 13** | **Dia 14** |
| HU-1 | Detectar rostros con cámara | 76 | 4 | 2 | 2 | 2 | 2 | 1 |  |  |  |  |  |  |  |  |
| HU-2 | Monitoreo de Cierre Ocular | 36 | 0 | 0 | 0 | 0 | 0 |  | 1 |  | 4 | 4 | 8 |  |  |  |
| Tiempo de trabajo | | 112 | 104 | 96 | 93 | 93 | 86 | 83 | 82 | 82 | 78 | 74 | 66 | 66 | 66 | 66 |



# 6. Análisis del Sprint

* Se cumplió con todas las tareas estimadas.
* El progreso fue constante y sin bloqueos mayores.
* Las decisiones técnicas sobre persistencia con localStorage resultaron efectivas para este primer Sprint.
* Se logró modularizar el código y documentarlo en comentarios.
* Las estimaciones fueron precisas y se logró un buen ritmo de trabajo.

# 7. Conclusiones

El Sprint 1 concluyó exitosamente con todas las funcionalidades básicas implementadas. Se generaron componentes reutilizables, se probaron los flujos de usuario, y el código quedó preparado para futuras funcionalidades como edición y múltiples listas.

El equipo demostró capacidad de entrega puntual y buena gestión de riesgos menores. Las métricas indican que la carga fue bien distribuida y los objetivos del sprint se cumplieron completamente.