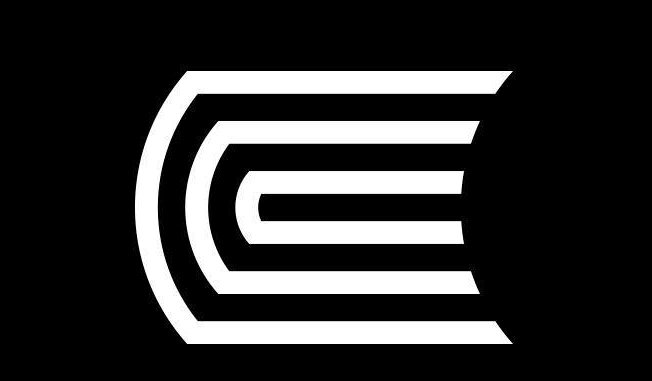
**“Año de la recuperación y consolidación de la economía peruana“**

**U N I V E R S I D A D C O N T I N E N T A L**

****

**FACULTAD DE INGENIERÍA**

**CURSO: TALLER DE PROYECTOS II**

**NRC: 17185**

**DOCENTE: AMERICO ESTRADA SANCHEZ**

**“PROYECTO”**

**ALUMNA: BUENDIA GONZÁLEZ TIRZA NICOL**

**Cusco – Perú**

**2025**

**ÍNDICE**

[Introducción 8](#)

[Problema a Resolver 8](#)

[Objetivos del Proyecto 9](#)

[Objetivo General: 9](#)

[Objetivos Específicos: 9](#)

[Público Objetivo (Usuarios Finales) 9](#)

[Usuarios principales: 9](#)

[Beneficios esperados: 9](#)

[Funcionalidades Principales 9](#)

[Funcionalidades esenciales: 9](#)

[Funcionalidades futuras (Opcionales): 10](#)

[Requisitos Técnicos 10](#)

[Lenguajes y Frameworks: 10](#)

[Compatibilidad: 10](#)

[Riesgos y Limitaciones 10](#)

[Riesgos: 10](#)

[Limitaciones: 10](#)

[Alcance del Proyecto 10](#)

[Lo que incluirá: 11](#)

[Lo que NO incluirá (por ahora): 11](#)

[Especificación de Requisitos de Software 11](#_h3ckawmsvqv6)

[Alcance 11](#)

[Definiciones, Acrónimos y Abreviaturas 11](#)

[Referencias 11](#)

[Descripción General 12](#)

[Descripción General del Producto 12](#)

[Perspectiva del Producto 12](#)

[Funcionalidades Principales 12](#)

[Características de los Usuarios 12](#)

[Restricciones 13](#)

[Suposiciones y Dependencias 13](#)

[Requisitos Específicos 13](#)

[Requisitos Funcionales 13](#)

[Requisitos No Funcionales 13](#)

[Requisitos de Interfaz de Usuario 13](#)

[Requisitos de Hardware y Software 14](#_i9stpes9wasr)

[Backlog del Producto 14](#_gzk6ila0ntur)

[Diseño de Épicas e Historias de Usuario 14](#_3uetegp4qmz3)

[Épicas e Historias de Usuario 14](#_nehg015w24zt)

[Épica 0: Desarrollo del Modelo de Machine Learning (Fase MLOps) 14](#_fz9uq3h7vf4i)

[Épica 1: Detección Facial en Tiempo Real 16](#_xk03anvhvceg)

[Épica 2: Sistema de Alertas 18](#_lwpg3qau8qly)

[Épica 3: Registro de eventos 18](#_e6pekh8sh2jz)

[Priorización del Backlog 19](#_fc05xf8jwy0z)

[Recolección y Etiquetado del Dataset 19](#_yqg0gi7911lx)

[Preprocesamiento de Datos 19](#_gdqqeprdgrhf)

[Entrenamiento del Modelo 19](#_n7bhtlmrq366)

[Integración del Modelo al Flujo Actual 19](#_jd3htvpta2qy)

[Detectar rostros con cámara 19](#_xzgje1ein2pg)

[Monitoreo de Cierre Ocular 19](#_f20kyohi3sgs)

[Alertas Temprana de somnolencia 19](#_tyzr99xgtv48)

[Exportar Datos a CSV 19](#_ozdmmz31hzoq)

[Sprint Backlog con Tareas y Estimaciones 19](#_cuqbuc1x0v8h)

[Criterios de Estimación de Esfuerzo 19](#_i6thu4x74tjr)

[Sprint 0.1: Historias de Usuario Seleccionadas 20](#_ae2zbhb0dgc3)

[Total de Esfuerzo Estimado 20](#_wggng3oaow8j)

[Sprint 0.2: Historias de Usuario Seleccionadas 21](#_ae2zbhb0dgc3)

[Total de Esfuerzo Estimado 22](#_wggng3oaow8j)

[Sprint 1: Historias de Usuario Seleccionadas 22](#_97jxmo1qnkdu)

[Total de Esfuerzo Estimado 23](#_wggng3oaow8j)

[Sprint 2: Historias de Usuario Seleccionadas 24](#_yxq9wf77uvzq)

[Total de Esfuerzo Estimado 24](#_nxtx2yz2xzqk)

[Diseño de Arquitectura 25](#_qlma3acl711w)

[Visión General de la Arquitectura 25](#_8bw3knfw03k4)

[Estilo Arquitectónico 25](#_7gs0z3o0vht2)

[Diagrama General de Arquitectura: 25](#_ind43n1vcbe0)

[Descripción de Componentes Principales 26](#_ljffyisn17sn)

[Capa de presentación 26](#_mahzeaut6siz)

[Capa de Lógica de Negocio 26](#_51yun3xsz7tg)

[Capa de Acceso a Datos 27](#_nl0o10brg0et)

[Capa de Persistencia 27](#_hwmhuxqypazm)

[Integraciones Externas (Opcionales) 28](#_vn0ookk99ve2)

[Seguridad 28](#_kv75hscp9pcr)

[Escalabilidad y Despliegue 28](#_gveeaov22rk9)

[Conclusiones 28](#_fchezkm3vdwm)

[Diseño de Base de Datos 29](#_4gmqn3x59q5g)

[Estructura de Archivos CSV 29](#_90opktbeokjm)

[Registro de Actividad (CSV) 29](#_jtsgokun7vra)

[Registro de Incidentes (Frames) 29](#_hox6sxju1ekf)

[Reglas y Restricciones 30](#_cslgpjcladr7)

[Estructura de Almacenamiento 30](#_xchmd5m4lnv2)

[Diccionario de Datos 31](#_w28ufd6ojvnl)

[Archivo CSV 31](#_h9vlvsxmu2pg)

[Frames Incidentes 31](#_1e6963jygwe0)

[Consideraciones de Escalabilidad 31](#_dnjle46bf63u)

[Conclusiones 32](#_nwc81j9hadvm)

[Wireframes o Prototipos 32](#_tjsfo3cm8hgm)

[Lineamientos de Diseño 32](#_g14fg430inlj)

[Wireframes por Pantalla 32](#_mzqhe6j86u4s)

[Pantalla principal (única) 32](#_9s8fi1nff99p)

[Prototipo de Navegación 33](#_3mdgl843rpk3)

[Guía de Estilo UI (Diseño Visual) 33](#_paf1bxw2puy6)

[Consideraciones Finales 34](#_phaxbmwung0j)

[Registro de Avance 34](#_1k5sxpgpzoun)

[Lineamientos de Diseño 34](#_2y7utcc08f90)

[Sprint 0.1 34](#_b2sn6bhq49a)

[Historias de Usuario y Tareas 34](#_ddlo8eat8505)

[Historia de Usuario 0.2 – Recolección y etiquetado del dataset 35](#_doc4gdn9hsxz)

[Historia de Usuario 0.3 – Entrenamiento del modelo 36](#_hrpv6a62br30)

[Evidencias de Código 40](#_qt9ts0scc80v)

[Historia de Usuario 0.2 — Preprocesamiento de datos 40](#_fz40mm8r40xb)

[Historia de Usuario 0.3 — Entrenamiento del modelo 48](#_nk43y030dcfe)

[Burndown Chart 53](#_ix2dj123a3pa)

[Análisis del Sprint 53](#_e4m8hguhxzzk)

[Conclusiones 54](#_ctvnw11fskza)

[Sprint 0.2 54](#_3q6gfuna5wve)

[Historias de Usuario y Tareas 54](#_ogt23tu7b8nf)

[Evidencias de Código 55](#_4r8n3qeg8o40)

[Historia de Usuario 0.4 — Integración del Modelo al Flujo Actual 55](#_bq3c4wdohnay)

[Burndown Chart 57](#_yr2agg1me25s)

[Análisis del Sprint 58](#_opjzsh5aeh3w)

[Conclusiones 58](#_4gkoenkj6ka4)

[Sprint 1 59](#_idtk5dglcoc7)

[Historias de Usuario y Tareas 59](#_jf8xk6pndp)

[Evidencias de Código 59](#_m0piwa7d3ymj)

[Historia de Usuario 1 — Detectar rostros con cámara 59](#_pcc5j1u8fxvt)

[Historia de Usuario 2 — Monitoreo de Cierre Ocular 67](#_8zg9z818i9n1)

[Burndown Chart 69](#_hbtgfgboqwu9)

[Análisis del Sprint 70](#_w6zlrqrm38ob)

[Conclusiones 70](#_i0f1fdp2cus7)

[Sprint 1-2 71](#_bgskmxfg4mqh)

[Historias de Usuario y Tareas 71](#_mwvu76g704f7)

[Evidencias de Código 72](#_266jzlb3c8gh)

[Historia de Usuario 1.1 — Detectar rostros con cámara 72](#_7ijar1i1o00w)

[Historia de Usuario 1.2 — Monitoreo de Cierre Ocular 73](#_c9n9ismub7g)

[Historia de Usuario 1.3 — Detección de Bostezos 73](#_ma3bn1k1osmz)

[Historia de Usuario 1.4 — Medición de Inclinación de Cabeza 73](#_kz3jk453zqe)

[Historia de Usuario 2.1 — Alerta Temprana de Somnolencia 77](#_xsq26hepiib5)

[Burndown Chart 79](#_yxwtadun7jnc)

[Análisis del Sprint 79](#_uhtjkaar6h5x)

[Conclusiones 80](#_yk59hcbwig66)

[Sprint 2 80](#_5leq1lc0oyza)

[Historias de Usuario y Tareas 80](#_cqjlcj6ry7u6)

[Evidencias de Código 81](#_cdnqilw0091e)

[Historia de Usuario 2.1 — Alerta Temprana de Somnolencia 81](#_3skv476tzub)

[Historia de Usuario 3.1 — Exportar Datos a CSV 82](#_auabs2nfhrqu)

[Burndown Chart 84](#_vqs2qo13bran)

[Análisis del Sprint 85](#_3rd9503icvzs)

[Conclusiones 85](#_vn6chvdl8qi9)

[Manual de Usuario Final 85](#_47ubq7hlou9o)

[Requisitos del Sistema 85](#_jy983u508cpv)

[Acceso al Programa 86](#_w79pv8xlr18a)

[Instalación del programa 86](#_ny22s4kr6vrt)

[Ejecución del programa 86](#_2ok2cm6wy5if)

[Funcionalidades Principales 86](#_evky1usq6aeb)

[Capturar Flujo de Video 86](#_uwp4mealfzhe)

[Monitoreo de Cierre Ocular 86](#_c422ae49fd33)

[Detección de bostezos 86](#_m110y6ms7mfs)

[Medición de Inclinación de Cabeza 86](#_jievifb1ahcy)

[Alertas de Somnolencia 86](#_l4w131rm4r1r)

[Buenas Prácticas de Uso 87](#_yrsd3pucilue)

[Preguntas Frecuentes (FAQ) 87](#_ree7qtln39l0)

[¿Qué requisitos tiene el programa? 87](#_ykr2isrfz28)

[¿Por qué no detecta mi rostro? 87](#_lw7lf29i814)

[Contacto y Soporte 87](#_una424jfx51t)

[Cierre 88](#_xyqkphrtzp7f)

[Manual Técnico 88](#_cgvk8k4gwixx)

[Tecnologías Utilizadas 88](#_lvpj8dja7n15)

[Estructura del Proyecto 89](#_rs2rcvcppw2b)

[Arquitectura General 89](#_63mi4618dn1m)

[Descripción de Componentes 89](#_ui8tamnyloep)

[Interfaz de Usuario (Frontend) 89](#_9bg7y0pg2not)

[Lógica de Negocio (Backend) 90](#_wijl98g0dog2)

[Persistencia y Almacenamiento de Datos 90](#_a9fb94lo33jg)

[Instalación y Ejecución 90](#_pvsq6yuzzjxw)

[Requisitos 90](#_75sk9qdha71u)

[Instalación 90](#_q34zt051aefp)

[Ejecución 90](#_usygokdllxxr)

[Seguridad 91](#_g0cqod6lajvh)

[Buenas Prácticas 91](#_ri4wmm66v28p)

[Futuras Mejoras 91](#_6xnk9vluxigo)

[Contacto 91](#_bns4slfkuiju)

[Manual de Instalación y Configuración 91](#_hh0r1vad7vs5)

[Requisitos del Sistema 92](#_5qwnykaqym4w)

[Requisitos de Software 92](#_4hgpm9f0vnpb)

[Requisitos de Hardware (mínimos) 92](#_3kktpq9tbf7r)

[Estructura del Proyecto 92](#_ch9dnnkef8n0)

[Instalación del Entorno de Desarrollo 92](#_vswl7efkbadb)

[Extracción de Carpeta 92](#_54sm8uam10ku)

[Instalación de Dependencias 92](#_7jk0934cqty)

[Verificación de Recursos 93](#_altz76k60ypg)

[Despliegue en Producción 93](#_e65z21x1f9qa)

[Recomendaciones 93](#_1yh7kyazye0v)

[Configuración sugerida 93](#_15vq2dpkkzhz)

[Configuración Adicional 93](#_hu7y47ynn673)

[Seguridad 93](#_4d7kmcygrkg4)

[Logs 93](#_e5i7bkd3hzpp)

[Verificación de Instalación 93](#_2h0gush3m2lg)

[Contacto Técnico 93](#_a8ns7ru0rjeq)

[Anexos 93](#_mkjzbws4gdjs)

[Manual de Administrador de Sistemas 94](#_9aazxs61lyy2)

[Acceso al Panel de Administración 94](#_gmmjjg20cihy)

[Gestión de Registros 94](#_5o53neooiurr)

[Visualización de Registros 94](#_85zmmtqnmte1)

[Limpieza de Registros 94](#_fo2cqjns0ax1)

[Almacenamiento de Evidencia Visual (frames) 94](#_75ov9mdxpaii)

[Mantenimiento del Modelo y Dataset 94](#_n1o80ve2lnzt)

[Actualización del modelo 94](#_dop2s1e4jenk)

[Respaldo y Recuperación de Base de Datos 94](#_wzhyhmbum1dl)

[Respaldo 94](#_wghcnuag8atq)

[Recuperación Para restaurar 94](#_ltjts7mss0q5)

[Monitoreo y Logs 95](#_ug6eihsx0bt2)

[Supervisión de Funcionamiento 95](#_a5ep07sztnfb)

[Verificación de Componentes 95](#_iyb3strgooid)

[Seguridad Administrativa 95](#_i6vxn5c5kwcc)

[Automatización Recomendada 95](#_dvxf78nxtqks)

[Contacto para Escalamiento 95](#_l68sdfoiasp)

[Cambios Previstos para el Rol de Administrador 95](#_rc87grnb48j6)

[Manual de Mantenimiento 96](#_k7mzh8czfl5y)

[Objetivos del Mantenimiento 96](#_qawutoc8vaoj)

[Mantenimiento Preventivo 96](#_7fdlp0e2wnw5)

[Tareas Semanales 96](#_7il9exxy9uhn)

[Tareas Mensuales 96](#_yd5a6u496ob8)

[Tareas Trimestrales 96](#_1p8i1i709r6f)

[Mantenimiento Correctivo 97](#_htno2c3ulstn)

[Gestión de Errores Comunes 97](#_1bnnsdn5yadt)

[Procedimiento de Corrección 97](#_rxe8732oni2q)

[Respaldo de Información 97](#_q4o4ohej5fh)

[Respaldo Manual 97](#_8gpedijq6zlh)

[Respaldo Programado 97](#_ycj8k5efpicz)

[Actualización del Sistema 97](#_8z97kaiv01mr)

[Dependencias 97](#_8wk49o1lh50n)

[Modelo IA 97](#_1rpwayhu7mpv)

[Documentación de Cambios 97](#_9pegx3vu1c92)

[Herramientas Recomendadas 98](#_tvefq4i3bomo)

[Contacto de Soporte 98](#_o6dk8r4dzga3)

[Conclusión 98](#_7zsm9s11rcd8)

[Manual de Pruebas 98](#_9lgrbaadnodw)

[Objetivos 98](#_8kxzdt7j43cw)

[Tipos de Pruebas Aplicadas 98](#_rzk5kzuht5or)

[Pruebas Unitarias 98](#_gfh6lbct7mia)

[Pruebas de Integración 99](#_g35ohy9bki0j)

[Pruebas Funcionales 99](#_p1fjdnoxvoh1)

[Pruebas Manuales 99](#_7ucr3fng2swo)

[Ambiente de Pruebas 99](#_19lngr2rivk3)

[Casos de Prueba 99](#_635c8nc2wsb9)

[Activación de Alerta 99](#_3bmqh3sjez9r)

[Desactivación de Alerta 100](#_mgpjypm2b7tz)

[Corte de video 100](#_yxiifo4qosbu)

[Corte de video 100](#_qsz5xg8f88xt)

[Resultados 100](#_13b1tz67rtia)

[Criterios de Aceptación 100](#_ifvzejgrtkpt)

[Herramientas Utilizadas 100](#_q1xtaknfq0wd)

[Recomendaciones Futuras 100](#_xyneimmtest1)

[Contacto QA 101](#_hom7impj49fo)

[Registro de Mantenimiento y Plan de Mejoras Futuras 101](#_uqmdjqyn0sqp)

[Registro de Mantenimiento 101](#_ycpgzdv5itap)

[Historial de Mantenimientos Ejecutados 101](#_p9qw0ag4roij)

[Diagnóstico General 101](#_t4r8kmjum0g2)

[Fortalezas 101](#_cdg9yyfyriaf)

[Oportunidades de Mejora 102](#_6ib0qvglwado)

[Plan de Mejoras Futuras 102](#_e1p7gwtlsetb)

[Funcionalidades Prioritarias 102](#_gomeqwol6hd0)

[Acciones de Mantenimiento Programado 102](#_whsd0rbu2iiz)

[Seguimiento y Actualización 102](#_npcq9imbn3rg)

[Contacto Responsable 103](#_4ufmsqr7q2cw)

[Conclusión 103](#_1ar0vfs32pyd)

# Introducción

Conducir con somnolencia es algo que ocurre frecuentemente, especialmente entre los conductores de transporte público e interprovincial en Perú. Estos trabajadores suelen tener largas jornadas laborales, a menudo con poco descanso entre ellos, lo que aumenta el riesgo de fatiga al volante. Se trata de un problema grave porque la somnolencia al volante puede provocar reacciones más lentas, falta de concentración o incluso microsueños, y eso es más que suficiente para provocar un accidente.

Por eso, para este proyecto, decidí diseñar un sistema que utiliza la inteligencia artificial para detectar somnolencia en tiempo real mediante el análisis de gestos faciales, como la apertura de ojos, apertura de boca y los movimientos de la cabeza. Cuando el sistema detecta niveles peligrosos de somnolencia, envía alertas para que el conductor pueda tomar medidas antes de que ocurra algo malo.

Este documento explica cómo funciona el sistema, la tecnología utilizada y cómo planifiqué su desarrollo mediante historias de usuario y sprints. El objetivo es ofrecer una solución sencilla pero eficaz que pueda ayudar a que la conducción sea más segura, especialmente para quienes trabajan muchas horas en la carretera.

# Problema a Resolver

Es de conocimiento general que un gran porcentaje de la población en Perú hace uso del transporte público día a día, por lo que se puede decir que esta es una de las principales formas de movilidad urbana. Debido a la naturaleza del servicio, es razonable suponer que las jornadas laborales en este sector suelen ser intermitentes, pero lo que pocos nos ponemos a pensar es en cómo esto impacta a trabajadores bajo la profesión de conducción que, en su mayoría, tienen que pasar por largos periodos manejando con cortos descansos de por medio. Asimismo, esto también aplica a los conductores de transporte interprovincial, quienes a menudo enfrentan largos periodos de manejo, incluso durante la noche.

De esto se puede deducir que no es poco común que el sueño y fatiga estén constantemente presentes para los conductores de dichos transportes debido a una rotación desorganizada de horarios con falta de descanso apropiado y trastornos de sueño relacionados con el ambiente. Teniendo en cuenta todo lo expuesto previamente se podría decir que la somnolencia al volante es un factor constantemente presente para los conductores de transporte público debido a la naturaleza intermitente de las jornadas de trabajo.

Es un hecho que el sueño es un factor que influye negativamente en la seguridad vial, puesto que puede ser causante de accidentes de tráfico, sobre todo en trayectos largos y/o durante la noche. La somnolencia y la fatiga en conductores pueden reducir significativamente la capacidad de reacción, la visión y la concentración, inclusive causar lapsos de microsueños que pueden desencadenar malas maniobras detrás del volante y comportamientos de conducción peligrosos. Un conductor que presenta un grado alto de somnolencia y fatiga puede llegar a pestañear, cabecear y hasta quedarse dormido frente al volante dando paso a posibles accidentes automovilísticos que, en el peor de los casos, pueden llegar a tener consecuencias fatales e irreversibles. Es por ello que el sueño al volante representa un riesgo que no se puede pasar por alto.

Para mitigar los riesgos asociados con la conducción bajo condiciones de alta somnolencia, se propone desarrollar un sistema de alarma con inteligencia artificial que monitoree en tiempo real los gestos faciales del conductor con el fin de poder determinar si el estado de fatiga presenta un riesgo en la conducción para así ayudar a mantener al conductor consciente y prevenir accidentes.

# Objetivos del Proyecto

Define lo que se busca lograr con el software.

## Objetivo General:

Mitigar los riesgos asociados a la conducción bajo condiciones altas de somnolencia.

## Objetivos Específicos:

* Identificar patrones faciales indicativos de somnolencia a partir de técnicas de visión por computadora.
* Implementar un sistema de alerta basado en la detección de somnolencia para mejorar la seguridad vial.

# Público Objetivo (Usuarios Finales)

Define quién utilizará el software y qué beneficios obtendrá.

## Usuarios principales:

* Conductores de transporte público.
* Conductores de transporte inter urbano.
* Conductores particulares.

## Beneficios esperados:

* Mayor la seguridad vial.

# Funcionalidades Principales

Lista de características clave de la primera versión del software.

## Funcionalidades esenciales:

* Detección en tiempo real de gestos faciales indicativos de somnolencia.
* Generación de alertas y/o auditivas cuando se detecte somnolencia moderada o grave.
* Registro de eventos de somnolencia para análisis posterior.

## Funcionalidades futuras (Opcionales):

* Predicción del estado del conductor basada en historial de somnolencia.
* Implementación de modelos de aprendizaje profundo para mejorar la precisión.
* Integración con hardware externo, como cámaras infrarrojas de alta resolución.
* Integración con hardware como Raspberry Pi para futuras implementaciones.

# Requisitos Técnicos

Define el stack tecnológico del proyecto.

## Lenguajes y Frameworks:

* Python (para el desarrollo del modelo de IA y visión por computadora).
* OpenCV y TensorFlow/Keras (para el procesamiento de imágenes y entrenamiento del modelo).

## Compatibilidad:

* Windows 10 y versiones superiores.

# Riesgos y Limitaciones

Posibles problemas que pueden surgir durante el desarrollo.

## Riesgos:

* Posible resistencia por parte de los conductores a la adopción del sistema.
* El modelo no presenta respuestas precisas.
* Costos imprevistos (datasets de paga, necesidad de hardware más caro, etc.).

## Limitaciones:

* Falta de acceso a tecnología avanzada para pruebas en entornos reales.
* Necesidad de mayor capacitación en desarrollo de modelos de IA.
* Dependencia de la calidad y cantidad de datos disponibles para el entrenamiento del modelo.

# Alcance del Proyecto

Define qué incluirá y qué quedará fuera en la primera versión.

## Lo que incluirá:

* Herramientas de inteligencia artificial de aprendizaje
* Implementación de criterios de detección de somnolencia moderada.
* Generación de alertas de advertencia en tiempo real.

## Lo que NO incluirá (por ahora):

* Análisis de somnolencia en personas con trastornos del sueño.
* Uso de sensores fisiológicos como frecuencia cardiaca o monitoreo bioeléctrico.
* Implementación en hardware embebido como Raspberry Pi o dispositivos de a bordo en vehículos reales.

# Especificación de Requisitos de Software

Esta sección define los requisitos para el desarrollo de el **sistema basado en Inteligencia Artificial (IA) para detectar somnolencia en conductores**, utilizando visión por computadora. Su objetivo es prevenir accidentes viales mediante el análisis en tiempo real de expresiones faciales como parpadeos, bostezos, cabeceos, etc.

## Alcance

El sistema consistirá en un modelo de inteligencia artificial entrenado con datasets de expresiones faciales, una aplicación de escritorio o móvil que procese video en tiempo real desde una cámara y un sistema de alertas sonoras y visuales para advertir cuando se detecte somnolencia.

## Definiciones, Acrónimos y Abreviaturas

* **IA:** Inteligencia artificial
* **ML:** Machine Learning
* **EAR:** Eye Aspect Ratio.
* **MAR:** Mouth Aspect Ratio
* **Ángulos de navegación:** ángulos de movimiento de objetos en 3 dimensiones. Se basa en el movimiento sobre 3 ejes dimensionales.
* **OpenCV:** Biblioteca para procesamiento de imágenes

## Referencias

* VANNIEUWENHUYZE, Aurélien. *Inteligencia artificial fácil Machine Learning y Deep Learning prácticos*. . ENI EDICIONES, 2020. ISBN 9782409025327.
* PANDEY, Dhruv. *Eye Aspect Ratio ( EAR ) and Drowsiness detector using dlib.* Medium. Online. 2021. [Accessed 3 March 2025]. Available from: <https://medium.com/analytics-vidhya/eye-aspect-ratio-ear-and-drowsiness-detector-using-dlib-a0b2c292d706>
* BELLINO, Alessio. *SEQUENCE: a remote control technique to select objects by matching their rhythm.* Personal and Ubiquitous Computing. Online. 2018. Vol. 22, no. 4, p. 751–770. DOI 10.1007/s00779-018-1129-2.
* GIBSON, Editor-in-chief C Michael and D, M. *Tait-Bryan rotations.* WikiDoc. Online. 2012. [Accessed 2 March 2025]. Available from: <https://www.wikidoc.org/index.php/Tait-Bryan_rotations>

## Descripción General

El objetivo de este proyecto es desarrollar un sistema basado en inteligencia artificial preciso y confiable para la detección de somnolencia en conductores, utilizando visión por computadora. Se busca ofrecer una solución en tiempo real que analice expresiones faciales (como cierre de ojos, bostezos y cabeceos) para alertar al conductor cuando presente signos de fatiga, mejorando así la seguridad vial y reduciendo el riesgo de accidentes. Con este sistema, los conductores podrán recibir advertencias inmediatas que les permitan tomar acciones preventivas, como hacer una pausa o descansar.

Este documento está estructurado en secciones que detallan los requisitos funcionales y no funcionales, las tecnologías empleadas, así como los posibles riesgos y limitaciones del sistema, garantizando un desarrollo robusto y alineado con estándares de calidad.

# Descripción General del Producto

## Perspectiva del Producto

El producto contempla un frontend implementado una interfaz gráfica en Python utilizando Tkinter. En el backend, el sistema integrará un modelo de inteligencia artificial basado en TensorFlow o PyTorch para la detección de somnolencia. En cuanto al hardware, el sistema requerirá una cámara para la captura de imágenes en tiempo real y un computador con una GPU potente, para garantizar un procesamiento eficiente del modelo de IA.

## Funcionalidades Principales

* Detección facial: Identificación de ojos, boca y cabeza en tiempo real.
* Análisis de somnolencia: Cálculo de PERCLOS, frecuencia de bostezos y cabeceos.
* Alertas: Notificaciones sonoras y/o visuales.
* Registro de eventos: Guardar datos de incidentes para análisis posterior.

## Características de los Usuarios

* **Usuarios principales:** Conductores de automóviles.
* **Beneficios esperados:** Reducción de accidentes por somnolencia al volante, cumplimiento de normativas de seguridad vial.

## Restricciones

* Compatible con cámaras estándar.
* Requiere Python 3.9+ y sistemas operativos Windows/Linux.

## Suposiciones y Dependencias

* El sistema operará en entornos con **iluminación adecuada** para el correcto funcionamiento de la cámara convencional.
* Los conductores no usarán **accesorios que obstruyan el rostro** (ej: máscaras, gafas oscuras).
* Dependencias a bibliotecas de IA para el procesamiento de imágenes y detección facial.
* Hardware con GPU para el entrenamiento del modelo.

# Requisitos Específicos

## Requisitos Funcionales

* RF1: El sistema detectará rostros en tiempo real con una cámara.
* RF2: Calculará el porcentaje de cierre ocular (PERCLOS) y la frecuencia de bostezos, al igual que el ángulo de inclinación de la cabeza.
* RF3: Emitirá una alerta sonora y/o visual si: PERCLOS >20% o Frecuencia de cabeceo >3 veces/minuto.
* RF4: Guardará registros de eventos (detectados como somnolencia) en un archivo CSV para análisis posterior.

## Requisitos No Funcionales

* RNF1: Latencia máxima de 1 segundo en la detección en tiempo real.
* RNF2: Precisión mínima del 85% en condiciones de iluminación óptima.
* RNF3: Compatibilidad con cámaras estándar de 720p o superior.

## Requisitos de Interfaz de Usuario

* La interfaz debe ser intuitiva y minimalista.
* Mostrará video en vivo de la cámara.
* Métricas en tiempo real (PERCLOS, bostezos, cabeceos).
* Alertas visuales (cambio de color en la pantalla) y sonoras (tono continuo hasta respuesta del conductor).

## Requisitos de Hardware y Software

* **Cámara web convencional** (resolución mínima: 720p).
* **GPU** para entrenamiento y inferencia
* **Python 3.9+** como lenguaje principal.
* **TensorFlow** o PyTorch para el modelo de IA.
* **OpenCV** para procesamiento de imágenes.

# Backlog del Producto

El backlog está estructurado en **épicas**, **historias de usuario** y sus respectivos **criterios de aceptación**.

## Diseño de Épicas e Historias de Usuario

Para la correcta estructuración del backlog, cada épica y su respectiva historia de usuario estarán diseñadas bajo los siguientes principios:

* **Épicas**: Representan grandes bloques funcionales del sistema, agrupando historias de usuario relacionadas.
* **Historias de Usuario**: Son descripciones cortas y centradas en el usuario sobre una funcionalidad específica del producto.
* **Criterios de Aceptación**: Reglas que deben cumplirse para que la historia de usuario sea considerada completada.
* **Estimación**: Cada historia de usuario incluye una estimación de esfuerzo en puntos de historia.
* **Definición de "Hecho"**: Se especifica cuándo una historia de usuario se considera completada con éxito.

El diseño y priorización de las épicas y las historias de usuario se revisarán periódicamente para adaptarse a nuevas necesidades del negocio y de los usuarios.

## Épicas e Historias de Usuario

### Épica 0: Desarrollo del Modelo de Machine Learning (Fase MLOps)

**Objetivo:** Construir, evaluar e integrar un modelo de aprendizaje automático que detecte somnolencia en conductores utilizando landmarks faciales (MediaPipe) y métricas como EAR/PERCLOS como características.

**Historia de Usuario 0.1: Recolección y Etiquetado del Dataset**

**Como** Data Engineer, **quiero** recolectar y etiquetar un dataset de frames balanceados de rostros somnolientos y alertas, **para** entrenar un modelo supervisado.

**Criterios de Aceptación:**

* 500+ frames por clase (alerta / somnoliento).
* Etiquetado manual por 3 evaluadores independientes.
* Se toma como etiqueta final la mayoría (2/3). En caso de desacuerdo total, se descarta (tiene que haber mayoría de acuerdo para poder aceptar la etiqueta como somnolencia/despierto).
* Diversidad en iluminación, gafas, ángulos (en dataset).

**Estimación:** 9 puntos

**Definición de "Hecho":**

* La detección facial funciona con precisión >90%.
* Se ha probado con al menos 3 usuarios diferentes.
* La latencia es menor a 1 segundo.

**Historia de Usuario 0.2: Preprocesamiento de Datos**

**Como** ML Engineer, **quiero** normalizar y aumentar los datos recolectados, **para** mejorar la generalización del modelo ante distintas condiciones.

**Criterios de Aceptación:**

* Normalización de puntos faciales y escalado de métricas.
* Eliminación de frames corruptos.
* Aumentos con técnicas como RandomBrightnessContrast, rotación, flip horizontal.

**Estimación:** 6 puntos

**Definición de "Hecho":**

* Se generan versiones aumentadas para al menos el 50% del dataset.
* Se ejecutan pruebas visuales de los datos aumentados.
* Se asegura que todos los datos estén en formato limpio, usable y balanceado.

**Historia de Usuario 0.3: Entrenamiento del Modelo**

**Como** Data Scientist, **quiero** entrenar un modelo SVM o Random Forest usando landmarks y métricas, **para** predecir somnolencia con alta precisión.

**Criterios de Aceptación:**

* Precisión >85% con validación cruzada (k=5).
* Falsos negativos <10%.
* Comparación entre SVM y Random Forest con justificación de selección.

**Estimación:** 8 puntos

**Definición de "Hecho":**

* Se genera y guarda el modelo en .pkl o .joblib.
* Se adjunta script con métricas, curva ROC, matriz de confusión (para medición de rendimiento del modelo).
* El modelo se puede cargar desde archivo y ejecutar en menos de 0.5 segundos.

**Historia de Usuario 0.4: Integración del Modelo al Flujo Actual**

**Como** equipo de desarrollo, **necesitamos** integrar el modelo entrenado en el flujo del sistema, **para** reemplazar la lógica de umbrales fijos.

**Criterios de Aceptación:**

* El modelo acepta datos de entrada del pipeline actual (CSV o stream).
* Latencia de predicción <0.5 segundos.
* El sistema genera alertas a partir de la predicción del modelo ML.

**Estimación:** 5 puntos

**Definición de "Hecho":**

* El sistema usa el modelo automáticamente para detectar somnolencia.
* Se prueba en tiempo real con video simulado o en vivo.
* Se valida que las predicciones se reflejan correctamente en los logs (registros csv) o GUI (alarma visual en interfaz).

### Épica 1: Detección Facial en Tiempo Real

**Historia de Usuario 1.1: Detectar rostros con cámara**

**Como** usuario, **quiero** que el sistema detecte mi rostro en tiempo real, **para** realizar el monitoreo de somnolencia.

**Criterios de Aceptación:**

* El sistema inicia la captura de video al encenderse
* Identifica al menos un rostro en el frame
* Funciona con diferentes ángulos de cámara

**Estimación:** 8 puntos

**Definición de "Hecho":**

* La detección facial funciona con precisión >90%
* Se ha probado con al menos 3 usuarios diferentes
* La latencia es menor a 1 segundo

**Historia de Usuario 1.2: Monitoreo de Cierre Ocular**

**Como** usuario, **quiero** que el sistema mida con precisión mi porcentaje de cierre ocular (PERCLOS), **para** detectar fatiga visual.

**Criterios de Aceptación:**

* Calcula PERCLOS cada 30 segundos.
* Ignora parpadeos rápidos (<0.5 segundos).

**Estimación:** 5 puntos

**Definición de "Hecho":**

* Se ha implementado la funcionalidad y probado en un entorno controlado.
* El sistema calcula el PERCLOS con precisión cada 3’ segundos.
* Documentación técnica del algoritmo de cálculo.

**Historia de Usuario 1.3: Detección de Bostezos**

**Como** usuario, **quiero** que el sistema identifique mis bostezos, **para** evaluar mi nivel de somnolencia.

**Criterios de Aceptación:**

* Detecta apertura bucal >70% por más de 2 segundos.
* Filtra falsos positivos (ej: hablar o reír).
* Registra frecuencia por minuto.

**Estimación:** 4 puntos

**Definición de "Hecho":**

* Se ha implementado la funcionalidad y probado en un entorno controlado.
* Se entrena con datasets con imágenes de bostezos reales.

**Historia de Usuario 1.4: Medición de Inclinación de Cabeza**

**Como** usuario, **quiero** que el sistema detecte si mi cabeza se inclina por somnolencia, **para** alertarme.

**Criterios de Aceptación:**

* Distingue entre movimientos voluntarios y cabeceos por fatiga.
* Usar el modelo entrenado para predecir alerta.

**Estimación:** 4 puntos

**Definición de "Hecho":**

* Se ha implementado la funcionalidad y probado en un entorno controlado.
* Se detectan landmarks faciales con OpenCV.

### Épica 2: Sistema de Alertas

**Historia de Usuario 2.1: Alerta Temprana de somnolencia**

**Como** usuario, **quiero** recibir alertas claras cuando el sistema detecte signos de somnolencia, **para** tomar medidas preventivas.

**Criterios de Aceptación:**

* La alerta sonora es suficientemente audible (sin ser molesta).
* La notificación visual aparece en la pantalla con iconos intuitivos.
* Las alertas persisten hasta que las reconozco pulsando un botón.

**Estimación:** 4 puntos

**Definición de "Hecho":**

* Se ha implementado la funcionalidad y probado en un entorno controlado.
* El usuario recibe alertas de somnolencia temprana y reacciona a tiempo.
* El usuario apaga la alerta a través de un botón tras reacción.

### Épica 3: Registro de eventos

**Historia de Usuario 3.1: Exportar Datos a CSV**

**Como** usuario, **quiero** un archivo CSV con eventos de somnolencia, **para** analizar patrones.

**Criterios de Aceptación:**

* Columnas: PERCLOS, bostezos/min, cabeceos/min.
* Compatible con Excel/Pandas.
* Generación automática al finalizar viaje.

**Estimación:** 3 puntos

**Definición de "Hecho":**

* Los reportes en archivos CSV son generados según los criterios de aceptación.
* Se tiene ejemplo de CSV en documentación

## Priorización del Backlog

| Prioridad | Historia de Usuario | Estado | Estimación (Puntos) |
| --- | --- | --- | --- |
| Alta | Recolección y Etiquetado del Dataset | Pendiente | 9 |
| Alta | Preprocesamiento de Datos | Pendiente | 6 |
| Alta | Entrenamiento del Modelo | Pendiente | 8 |
| Alta | Integración del Modelo al Flujo Actual | Pendiente | 5 |
| Alta | Detectar rostros con cámara | Pendiente | 8 |
| Alta | Monitoreo de Cierre Ocular | Pendiente | 5 |
| Media | Detección de Bostezos | Pendiente | 5 |
| Media | Medición de Inclinación de Cabeza | Pendiente | 4 |
| Alta | Alertas Temprana de somnolencia | Pendiente | 4 |
| Baja | Exportar Datos a CSV | Pendiente | 3 |

# Sprint Backlog con Tareas y Estimaciones

En esta sección se detalla el Sprint Backlog del equipo de desarrollo para el Sprint actual de un modelo **basado en Inteligencia Artificial (IA) para detectar somnolencia en conductores**. Se desglosan las historias de usuario seleccionadas, sus tareas específicas y la estimación de esfuerzo en horas.

## Criterios de Estimación de Esfuerzo

Para estimar el esfuerzo de cada tarea, se utilizó la técnica **Planning Poker**, donde el equipo de desarrollo discutió cada tarea y asignó una cantidad de horas basada en:

* Complejidad técnica
* Dependencias con otras tareas
* Experiencia previa en tareas similares
* Posibles riesgos o bloqueos

Cada tarea se estimó en consenso utilizando la escala de **horas-persona**

## Sprint 0.1: Historias de Usuario Seleccionadas

**Historia de Usuario 0.1: Recolección y Etiquetado del Dataset**

**Descripción:** Como Data Engineer, quiero recolectar y etiquetar un dataset de frames balanceados de rostros somnolientos y alertas, para entrenar un modelo supervisado.

**Historia de Usuario 0.2: Preprocesamiento de Datos**

**Descripción:** Como ML Engineer, quiero normalizar y aumentar los datos recolectados, para mejorar la generalización del modelo ante distintas condiciones.

**Historia de Usuario 0.3: Entrenamiento del Modelo**

**Descripción:** Como Data Scientist, quiero entrenar un modelo SVM o Random Forest usando landmarks y métricas, para predecir somnolencia con alta precisión.

**Tareas:**

| ID | Tarea | Responsable | Estimación (Horas) | Estado |
| --- | --- | --- | --- | --- |
| 0.1.1 | Recolectar frames (datasets públicos) | Data Engineer | 15 | Pendiente |
| 0.1.2 | Etiquetar frames | Data Engineer | 25 | Pendiente |
| 0.2.1 | Normalizar datos y escalar métricas | ML Engineer | 15 | Pendiente |
| 0.2.2 | Aumentar datos (brillo/contraste, rotación leve) | ML Engineer | 15 | Pendiente |
| 0.3.1 | Seleccionar algoritmo y definir features (EAR, MAR, ángulos) | Data Scientist | 10 | Pendiente |
| 0.3.2 | Desarrollar modelo ML base (SVM / RF configurado) | Data Scientist | 20 | Pendiente |

### Total de Esfuerzo Estimado

| Función | Responsable | Horas Totales |
| --- | --- | --- |
| Computer Vision Development | Computer Vision Engineer | 0 |
| ML/Data Science | Data Scientist/ML Engineer | 100 |
| Development Team | Developers Team | 0 |
|
| UX/UI Design | UX/UI Designer | 0 |
| Quality Assurance (QA) | QA Engineer | 0 |
| Technical Writing | Technical Writer | 0 |
| Total General |  | 100 |

Este Sprint Backlog será actualizado diariamente durante las reuniones de seguimiento del equipo reflejará el avance del desarrollo y posibles ajustes en la planificación.

## Sprint 0.2: Historias de Usuario Seleccionadas

**Historia de Usuario 0.3: Entrenamiento del Modelo**

**Descripción:** Como Data Scientist, quiero entrenar un modelo SVM o Random Forest usando landmarks y métricas, para predecir somnolencia con alta precisión.

**Historia de Usuario 0.4: Integración del Modelo al Flujo Actual**

**Descripción: Como** equipo de desarrollo, **necesitamos** integrar el modelo entrenado en el flujo del sistema, **para** reemplazar la lógica de umbrales fijos.

**Tareas:**

| ID | Tarea | Responsable | Estimación (Horas) | Estado |
| --- | --- | --- | --- | --- |
| 0.3.3 | Entrenar modelo (SVM / Random Forest) | Data Scientist | 30 | Pendiente |
| 0.3.4 | Evaluar con validación cruzada + métricas | Data Scientist | 15 | Pendiente |
| 0.3.3 | Documentar proceso de entrenamiento | Data Scientist | 10 | Pendiente |
| 0.4.1 | Exportar modelo en formato .pkl | ML Engineer | 10 | Pendiente |
| 0.4.2 | Integrar modelo en entorno de prueba/programa final. | QA Engineer | 10 | Pendiente |

### Total de Esfuerzo Estimado

| Función | Responsable | Horas Totales |
| --- | --- | --- |
| Computer Vision Development | Computer Vision Engineer | 0 |
| ML/Data Science | Data Scientist/ML Engineer | 65 |
| Development Team | Developers Team | 0 |
|
| UX/UI Design | UX/UI Designer | 0 |
| Quality Assurance (QA) | QA Engineer | 10 |
| Technical Writing | Technical Writer | 0 |
| Total General |  | 75 |

Este Sprint Backlog será actualizado diariamente durante las reuniones de seguimiento del equipo reflejará el avance del desarrollo y posibles ajustes en la planificación.

## Sprint 1: Historias de Usuario Seleccionadas

**Historia de Usuario 1.1: Detectar rostros con cámara**

**Descripción:** Como usuario, quiero que el sistema detecte mi rostro en tiempo real, para realizar el monitoreo de somnolencia.

**Historia de Usuario 1.2: Monitoreo de Cierre Ocular**

**Descripción:** Como usuario, quiero que el sistema mida con precisión mi porcentaje de cierre ocular (PERCLOS), para detectar fatiga visual.

**Historia de Usuario 1.3: Detección de Bostezos**

**Descripción:** Como usuario, quiero que el sistema identifique mis bostezos, para evaluar mi nivel de somnolencia.

**Historia de Usuario 1.4: Medición de Inclinación de Cabeza**

**Descripción:** Como usuario, quiero que el sistema detecte si mi cabeza se inclina por somnolencia, para alertarme.

**Historia de Usuario 2.1 : Alerta Temprana de Somnolencia**

**Descripción:** Como usuario, quiero recibir alertas claras cuando el sistema detecte signos de somnolencia, para tomar medidas preventivas.

Tareas:

| ID | Tarea | Responsable | Estimación (Horas) | Estado |
| --- | --- | --- | --- | --- |
| 1.1.1 | Configurar cámara y flujo de video en tiempo real | Developers Team | 3 | Pendiente |
| 1.1.2 | Implementar detección facial con OpenCV/MediaPipe | Computer Vision Engineer | 3 | Pendiente |
| 1.1.3 | Optimizar latencia (<1 segundo por frame) | Developers Team | 3 | Pendiente |
| 1.1.4 | Pruebas con diferentes ángulos y condiciones de luz | QA Engineer | 2 | Pendiente |
| 1.1.5 | Documentar configuración y requisitos de hw | Technical Writer | 2 | Pendiente |
| 1.2.1 | Implementar cálculo de EAR con landmarks oculares | Computer Vision Engineer | 3 | Pendiente |
| 1.2.2 | Documentar algoritmo y métricas de validación | Technical Writer | 1 | Pendiente |
| 1.3.1 | Implementar Mouth Aspect Ratio (MAR) para bostezos | Computer Vision Engineer | 3 | Pendiente |
| 1.4.1 | Implementar detección de ángulo de cabeza (pitch/yaw/pitch) | Computer Vision Engineer | 3 | Pendiente |
| 1.4.2 | Pruebas de precisión | QA Engineer | 8 | Pendiente |
| 1.4.3 | Pruebas de función | QA Engineer | 8 | Pendiente |
| 2.1.1 | Diseñar interfaz de alerta visual | UX/UI Designer | 8 | Pendiente |
| 2.1.2 | Implementar alarma sonora (85 dB, no intrusiva) | Developers Team | 8 | Pendiente |

### Total de Esfuerzo Estimado

| Función | Responsable | Horas Totales |
| --- | --- | --- |
| Computer Vision Development | Computer Vision Engineer | 12 |
| ML/Data Science | Data Scientist/ML Engineer | 0 |
| Development Team | Developers Team | 14 |
|
| UX/UI Design | UX/UI Designer | 8 |
| Quality Assurance (QA) | QA Engineer | 18 |
| Technical Writing | Technical Writer | 3 |
| Total General |  | 55 |

Este Sprint Backlog será actualizado diariamente durante las reuniones de seguimiento del equipo reflejará el avance del desarrollo y posibles ajustes en la planificación

## Sprint 2: Historias de Usuario Seleccionadas

**Historia de Usuario 2.1: Alerta Temprana de Somnolencia**

**Descripción:** Como usuario, quiero recibir alertas claras cuando el sistema detecte signos de somnolencia, para tomar medidas preventivas.

**Historia de Usuario 3.1: Exportar Datos a CSV**

**Descripción:** Como usuario, quiero un archivo CSV e imágen con eventos de somnolencia, para analizar patrones.

Tareas:

| ID | Tarea | Responsable | Estimación (Horas) | Estado |
| --- | --- | --- | --- | --- |
| 2.1.3 | Configurar persistencia de alerta hasta confirmación | Developers Team | 5 | Pendiente |
| 2.1.4 | Pruebas de usabilidad | QA Engineer | 8 | Pendiente |
| 3.1.1 | Definir estructura del CSV (timestamp, PERCLOS, etc.) | Data Scientist | 2 | Pendiente |
| 3.1.2 | Implementar generación automática de CSV | Developers Team | 16 | Pendiente |
| 3.1.3 | Validar compatibilidad con Excel/Pandas | QA Engineer | 5 | Pendiente |
| 3.1.4 | Documentar formato y ejemplos de CSV | Technical Writer | 3 | Pendiente |
| 3.1.5 | Implementar generación automática de frames cuando se active la alarma | Developers Team | 8 | Pendiente |
| 3.1.6 | Documentar formato y ejemplos de frames | Technical Writer | 3 | Pendiente |

### Total de Esfuerzo Estimado

| Función | Responsable | Horas Totales |
| --- | --- | --- |
| Computer Vision Development | Computer Vision Engineer | 0 |
| ML/Data Science | Data Scientist/ML Engineer | 2 |
| Development Team | Developers Team | 29 |
|
| UX/UI Design | UX/UI Designer | 0 |
| Quality Assurance (QA) | QA Engineer | 13 |
| Technical Writing | Technical Writer | 6 |
| Total General |  | 50 |

Este Sprint Backlog será actualizado diariamente durante las reuniones de seguimiento del equipo reflejará el avance del desarrollo y posibles ajustes en la planificación.

# Diseño de Arquitectura

Esta sección describe de manera detallada la arquitectura del sistema para el modelo basado en IA, cuyo propósito es detectar la somnolencia al volante y generar una alerta temprana y precisa. Se define la estructura general del sistema, los componentes que lo conforman, sus interacciones y las tecnologías empleadas.

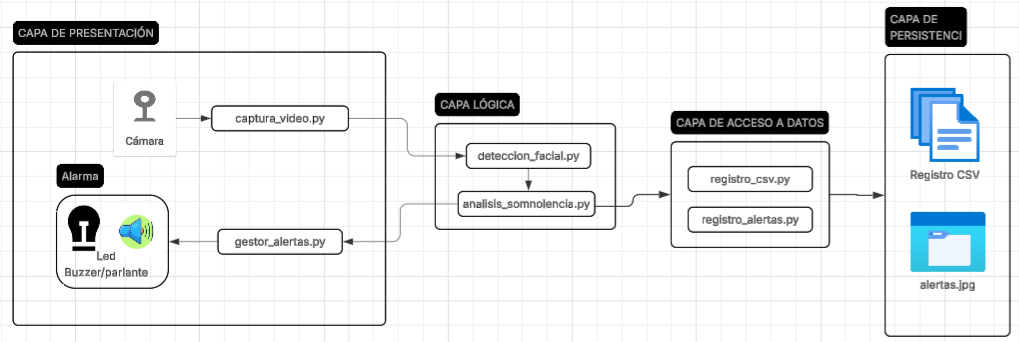
## Visión General de la Arquitectura

### Estilo Arquitectónico

El sistema adopta una arquitectura en capas monolítica, ya que no cuenta con capas que estén distribuidas y no utiliza servicios externos (todo se comunica dentro del mismo entorno). Se organiza en las capas principales:

* Capa de Presentación: Se encarga de la emisión de la alarma en caso que se identifique somnolencia.
* Capa de Lógica de Negocio: Procesamiento de IA, gestión de alertas y archivos CSV.
* Capa de Acceso a Datos: Gestiona los archivos que serán almacenados de manera local.
* Capa de Persistencia de Datos: Almacenamiento local de eventos de somnolencia, frames (datasets de entrenamiento), y el modelo .

### Diagrama General de Arquitectura:



## 

## Descripción de Componentes Principales

### Capa de presentación

La capa de presentación cuenta con una arquitectura monolítica, contiene componentes de HW que son la cámara y los dispositivos que conforman la alarma (led y buzzer) que serán simulados en una interfaz. Asimismo, se cuenta con los módulos encargados de la captura de video en tiempo real para pasar los datos a la capa lógica y el módulo encargado de gestionar la activación y desactivación de las alertas.

* Framework/Librería: Python (Tkinter para simulación de alarma en interfaz).
* Responsabilidades:
  + Capturar video en tiempo real de la cámara.
  + Alertas: Generación de alertas sonoras y notificaciones visuales (simulados en interfaz) para alertar al conductor si el sistema detecta signos de somnolencia.
  + Desactivar las alertas tras reacción del usuario.

**Módulos lógicos:**

* captura\_video.py: Se encarga de capturar el video en tiempo real de la cámara para ser procesado.
* gestor\_alertas.py: Gestiona la creación y disparo de alertas, tanto sonoras como visuales (simuladas en interfaz).

### Capa de Lógica de Negocio

La capa lógica presentará una arquitectura monolítica con los módulos encargados de la detección facial y el análisis de nivel de somnolencia según los parámetros establecidos (para activar alarma o no).

* Lenguaje y Frameworks: Python con OpenCV, TensorFlow/PyTorch.
* Responsabilidades:
  + Detección facial: Uso de OpenCV o MediaPipe para identificar los landmarks faciales y procesar la imagen.
  + Modelo de IA: Implementación de machine learning para analizar la somnolencia mediante métricas como PERCLOS, bostezos y la inclinación de la cabeza.

**Módulos lógicos:**

* deteccion\_facial.py: Módulo encargado de la captura y procesamiento de la imagen para identificar el rostro del conductor y sus características faciales (ear, mar, yaw, pitch, roll).
* analisis\_somnolencia.py: Módulo donde se usa el modelo entrenado para predicción de somnolencia o no somnolencia .

### Capa de Acceso a Datos

Esta capa también sigue una arquitectura monolítica y cuenta con los módulos encargados de la gestión de la capa de persistencia, es decir, el registro de los archivos CSV y frames de incidentes.

* Responsabilidades:
  + Gestión de registros: Genera archivos CSV para mantener registro del uso del modelo.
  + Gestión de alertas: Registra las frames con métricas tras la activación de la alarma.

**Módulos lógicos:**

* registro\_csv.py: Registra los eventos y resultados del sistema en un archivo CSV para seguimiento y análisis posterior (por ejemplo, hora de la alerta, PERCLOS, etc.).
* registro\_alertas.py: Registra frames de incidentes de somnolencia para poder analizar el rendimiento del sistema.

### Capa de Persistencia

La arquitectura de la capa de datos es una arquitectura monolítica con archivos planos, los registros de actividad serán almacenados en archivos CSV y los incidentes registrados en imágenes jpg también de manera local.

* Sistema Gestor: Archivos CSV. Archivos .jpg
* Esquema de Datos:
  + timestamp | PERCLOS | bostezos\_por\_minuto | inclinación\_cabeza | estado\_somnolencia.

## Integraciones Externas (Opcionales)

* Integración futura con un módulo de administración para una mejor gestión de los registros de alarmas y visualización de frames.

## Seguridad

Teniendo en cuenta que el desarrollo del sistema se realizará en un entorno controlado (sin acceso externo) y haciendo uso de datos públicos (datasets de entrenamiento libres). Las medidas de seguridad planteadas son las siguientes:

* Acceso Local Restringido: Los archivos (CSV, imágenes) solo pueden ser leídos/modificados por el sistema (la aplicación completa), no por usuarios u otros programas. Se configuran permisos de sistema operativo para evitar accesos no autorizados.
* Validación de Datos: Se verifica que los registros en CSV no contengan caracteres extraños o maliciosos (ej: scripts o comandos).

Ejemplo: Si el timestamp incluye un símbolo raro (\* o !), se rechaza.

* Seguridad Física (Cámara): La cámara solo transmite video en tiempo real al sistema. No se guardan videos completos, evitando riesgos de vigilancia no deseada.
* Logging Seguro: Los archivos de registro (registros.csv) guardan solo eventos técnicos (ej: "Alerta: PERCLOS alto a las 10:30"). No almacenan datos personales ni imágenes.

## Escalabilidad y Despliegue

Posibles problemas que pueden surgir durante el desarrollo.

* Simulación de interfaz: Desplegado en Tkinter
* Módulos: Python
* Almacenamiento: Archivos planos
* La arquitectura monolítica por capas del sistema facilita su mantenimiento y comprensión en entornos locales.
* La separación lógica del sistema permite desacoplar progresivamente sus componentes, facilitando que funciones como la gestión de alertas o el análisis de somnolencia puedan convertirse en servicios independientes en el futuro.
* La persistencia con archivos planos puede migrar a una base de datos estructurada, permitiendo separar la capa de datos en una instancia dedicada.

## Conclusiones

La arquitectura del sistema combina eficiencia, modularidad y potencial de escalabilidad, alineándose con los objetivos del proyecto. Su diseño monolítico por capas (presentación, lógica de negocio y persistencia) claramente diferenciadas. La modularidad de la capa lógica, dividida en componentes funcionales, facilita el mantenimiento y futuras actualizaciones. Además, aunque inicialmente el sistema se ejecutará de forma local, la separación lógica de sus bloques permite planificar una evolución progresiva hacia una arquitectura más escalable. Esto incluye la posibilidad de desacoplar funciones críticas como el análisis de somnolencia o la gestión de alertas, y migrar la capa de datos a una base de datos estructurada en servidores dedicados o en la nube.

# Diseño de Base de Datos

En esta sección se describe el modelo de archivos almacenados localmente para el sistema de detección de somnolencia basado en IA. Dado que el sistema no utiliza una base de datos tradicional sino archivos planos, se detallan las estructuras de los archivos CSV para registros de actividad y el almacenamiento de frames de incidentes (alertas) para depuración y evaluación del sistema.

## Estructura de Archivos CSV

### Registro de Actividad (CSV)

El sistema generará un archivo CSV principal que registrará todas las métricas relevantes y eventos de somnolencia detectados.

**Nombre del archivo:**  registros\_somnolencia\_YYYY-MM-DD HH:MM:SS.csv

**Ubicación:** /registros/csv/

**Estructura:** Hora | EAR | MAR | yaw | pitch | roll | nombre\_frame\_incidente

**Descripción de campos:**

* Hora: Hora del evento (formato: HH:MM:SS)
* EAR: tasa de apertura de ojos.
* MAR: tasa de apertura de boca
* yaw: ángulo de rotación en el eje vertical
* pitch: ángulo de rotación en el eje horizontal
* roll: ángulo de rotación en el eje longitudinal
* nombre\_frame\_incidente: Nombre del frame guardado si hubo alerta
* ejemplo de entrada: 14:30:45, 0.25770, 0.01799, 2.188950516, 173.6060827, 2.579939681, incidente\_20250410\_143045.png

### Registro de Incidentes (Frames)

El sistema almacenará imágenes JPEG de los frames donde se detectaron incidentes de somnolencia, etiquetados con metadatos para facilitar la depuración.

* Estructura de directorios:

/registros/frames/

├── 2025-04/

│ ├── 10/

│ │ ├── frame\_20250410\_143045.png

│ │ ├── frame\_20250410\_150122.png

│ ├── 11/

│ │ ├── frame\_20250411\_082315.png

* Convención de nombres: incidente\_AAAAMMDD\_HHMMSS.png
* Metadatos incluidos en el nombre: Fecha (AAAAMMDD) y Hora (HHMMSS)
* Características: Cada imagen incluirá superpuestas landmarks relevantes que generaron la alerta.

### Reglas y Restricciones

* Los datos deben seguir la estructura establecida para un correcto funcionamiento del modelo.

## Estructura de Almacenamiento

Sistema de Almacenamiento (Registros)

│

├── CSV

│ └── registros\_somnolencia.csv

│ ├── Hora

│ ├── EAR

│ ├── MAR

│ ├── Yaw

│ ├── Pitch

│ ├── Roll

│ └── nombre\_frame\_incidente

│

└── frames

└── incidente\_AAAAMMDD\_HHMMSS.png

└── Metadatos en nombre

## Diccionario de Datos

### Archivo CSV

| **Campo** | **Tipo de Dato** | **Descripción** | **Ejemplo** |
| --- | --- | --- | --- |
| **timestamp** | String | Fecha y hora del evento | "2025-04-10 14:30:45" |
| **EAR** | Float | Tasa de cierre ocular al saltar la alarma. | 0.03 |
| **MAR** | Foat | Tasa de apertura de boca al saltar la alarma | 0.01799 |
| **Yaw, Pitch, Roll** | Float | Ángulo de inclinación de cabeza en grados en 3 ejes. | 173.6060827 |
| **nombre\_frame\_incidente** | String | Nombre del frame guardado | "incidente\_20250410\_143045.png" |

### Frames Incidentes

| **Atributo** | **Descripción** |
| --- | --- |
| **Formato** | JPEG |
| **Resolución** | 640x480 |
| **Metadatos** | Fecha y hora (incluidos en el nombre del archivo) |
| **Métricas visibles** | Perclos, bostezos, inclinación y timestamp superpuestos en la imagen |
| **Organización** | Directorios por mes y día (/YYYY-MM/DD/) |

## Consideraciones de Escalabilidad

**Escalabilidad:**

* Rotación automática: CSV mensuales (registros\_YYYYMM.csv) .
* Migración futura: Estructura normalizada (nombres de archivos estandarizados, campos compatibles con bases de datos).
* Optimización: Compresión de imágenes (PNG al 85%) y CSV minimalistas (solo alertas).

**Seguridad:**

* Acceso: Permisos restrictivos (solo escritura para el sistema, solo lectura para admins).
* Integridad: Validación de datos (rangos numéricos, sanitización de timestamps).
* Privacidad: Sin videos completos ni datos personales, solo frames técnicos con métricas.

## Conclusiones

Este modelo de almacenamiento basado en archivos planos garantiza la flexibilidad necesaria para gestionar alertas de somnolencia, métricas técnicas y frames de incidentes, optimizando el espacio mediante registros CSV minimalistas (solo eventos relevantes) y compresión de imágenes (PNG al 85%). La estructura organizativa asegura escalabilidad en entornos locales, mientras que la normalización de campos y nombres estandarizados facilita una futura migración a bases de datos. Las medidas de seguridad (permisos restrictivos, validación de datos y privacidad en metadatos) protegen la integridad de la información sin almacenar datos sensibles, permitiendo escalar el sistema sin comprometer su eficacia ni robustez.

# Wireframes o Prototipos

Esta sección presenta los wireframes y prototipos de alta fidelidad para el modelo basado en IA para detección de somnolencia. Se busca representar de forma visual la interacción del usuario con el sistema y la disposición de los elementos clave en cada pantalla.

## Lineamientos de Diseño

El wireframe presentado corresponde a una simulación de la alarma visual que se activará cuando el sistema detecte somnolencia. En este entorno de pruebas, la pantalla de la laptop se utilizará para emular el funcionamiento de un LED que, en un entorno implementado físicamente, actuaría como señal visual de alerta. Del mismo modo, el buzzer será representado por el parlante del computador, cumpliendo la función de alarma sonora. Esta simulación se ha diseñado con las siguientes características:

* Diseño centrado en el usuario: Interfaz minimalista.
* Consistencia visual: Uso de estilos unificados (colores, tipografías).

## Wireframes por Pantalla

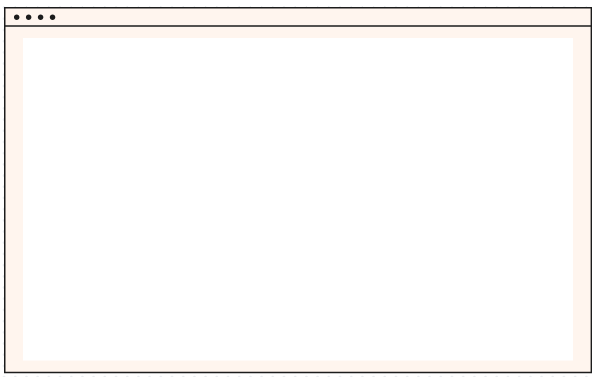
### Pantalla principal (única)

* Componentes: Alarma visual.

Se detecta somnolencia:



Sin detección de somnolencia:



## Prototipo de Navegación

No se requiere ningúna intervención por parte del usuario ni administrador.

## Guía de Estilo UI (Diseño Visual)

Tipografía

* Primaria: Calibri
* Títulos: Bold, tamaños desde 24px (h1) a 16px (h4)
* Texto: Regular 14px

Paleta de Colores

* Principal: #1C1C1C (negro grafito)
* Secundario: #FFF5EE (crema)
* Fondo: #FFF5EE (crema)
* Texto Principal: #FFFFFF (blanco)
* Alerta: #E62E1B (rojo bermellón)

## Consideraciones Finales

* Los wireframes están sujetos a revisión con usuarios reales para mejorar la experiencia.
* Se recomienda realizar pruebas de usabilidad antes del desarrollo final del frontend.

# Registro de Avance

Esta sección 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.

## 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.

## Sprint 0.1

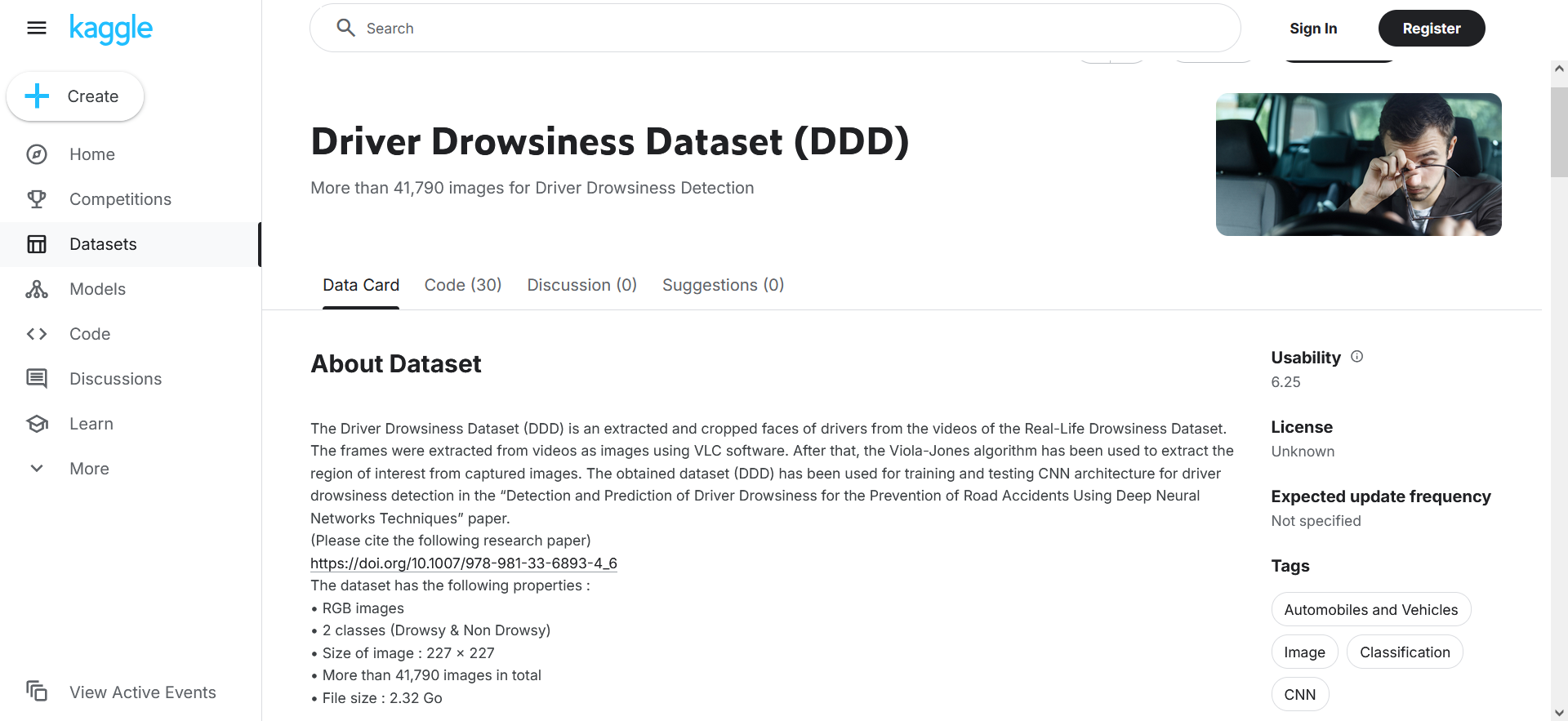
### Historias de Usuario y Tareas

| Backlog ID | Historia de Usuario | Tiempo Estimado | Fecha de Inicio | Fecha de Finalización |
| --- | --- | --- | --- | --- |
| HU-0.1 | Recolección y Etiquetado del dataset | 9 | 08/05/2025 | 10/05/2025 |
| HU-0.2 | Preprocesamiento de datos | 6 | 11/05/2025 | 14/05/2025 |
| HU-0.3 | Entrenamiento del Modelo | 8 | 15/05 | 20/05/2025 |

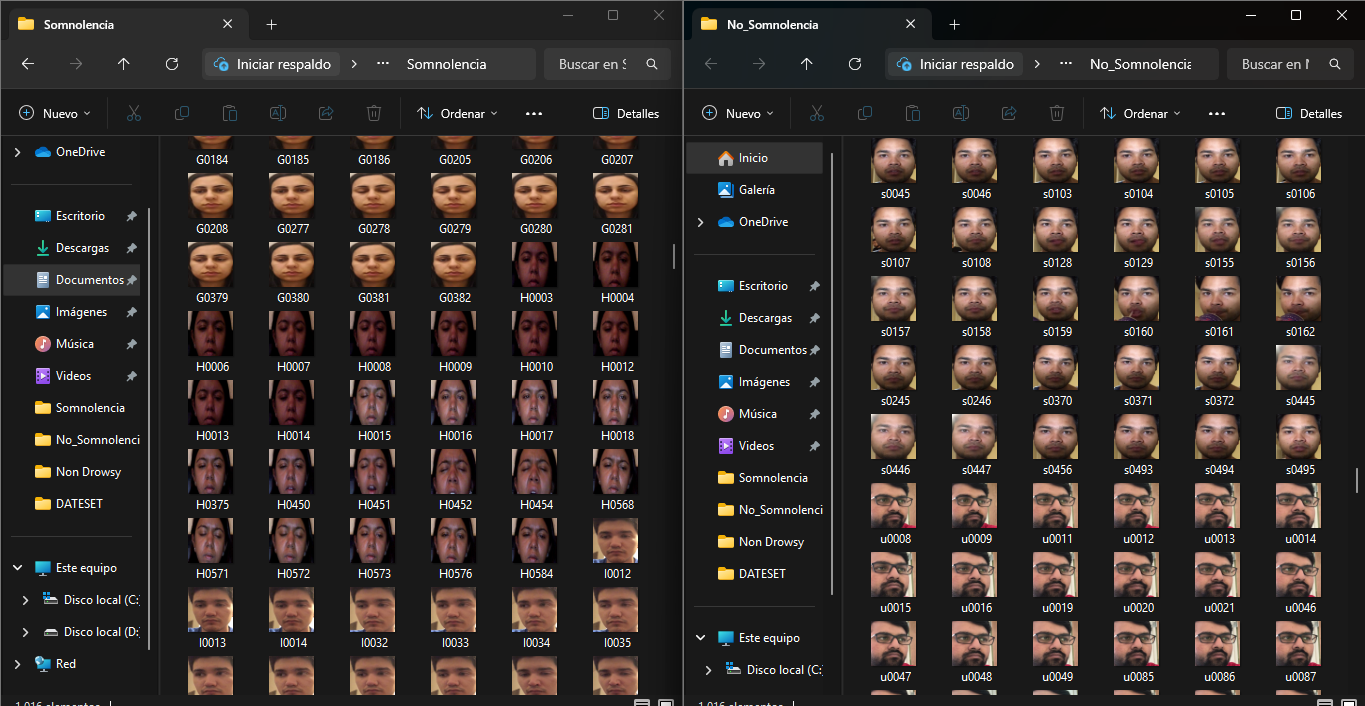
| ID | Tarea | Responsable | Estimación (Horas) | Estado |
| --- | --- | --- | --- | --- |
| 0.1.1 | Recolectar frames (datasets públicos) | Data Engineer | 15 | Pendiente |
| 0.1.2 | Etiquetar frames | Data Engineer | 25 | Pendiente |
| 0.2.1 | Normalizar datos y escalar métricas | ML Engineer | 15 | Pendiente |
| 0.2.2 | Aumentar datos (brillo/contraste, rotación leve) | ML Engineer | 15 | Pendiente |
| 0.3.1 | Seleccionar algoritmo y definir features (EAR, MAR, ángulos) | Data Scientist | 10 | Pendiente |
| 0.3.2 | Desarrollar modelo ML base (SVM / RF configurado) | Data Scientist | 20 | Pendiente |

#### Historia de Usuario 0.2 – Recolección y etiquetado del dataset

* Actividad 0.1.1 Recolectar Frames: se usó una dataset pública de Kaggle, Driver Drowsiness Dataset (DDD). <https://www.kaggle.com/datasets/ismailnasri20/driver-drowsiness-dataset-ddd>



* Actividad 0.1.2 Etiquetar Frames: Se separó un grupo de 1013 frames para la clase somnolencia y 1013 frames para la clase de no somnolencia a partir de la dataset original que cuenta con +42000 frames.



#### Historia de Usuario 0.3 – Entrenamiento del modelo

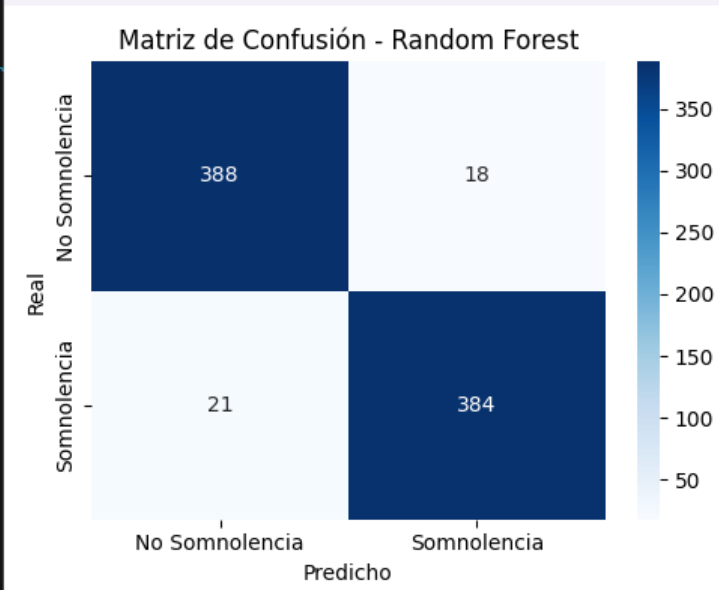
* Actividad 0.3.1 Seleccionar algoritmo y definir features: Se definieron los features en base a las métricas recolectadas en los frames del dataset y la selección del algoritmo se hizo con ayuda del análisis de un cuadro comparativo.
  + Features: EAR (Eye aspect ratio), MAR (Mouth aspect ratio), Ángulos de Tait-Bryan (Pitch, Yaw, Roll).
  + Potenciales algoritmos a usar: Se decidió tomar 2 algoritmos (RF y XGBoost debido a su buen rendimiento bajo las consideraciones del tipo de datos y hardware con el que se cuenta para el proyecto) (documento completo: <https://docs.google.com/document/d/13DHo6f-SFOGolDbunmPCI4tlgLuc9-XXutow92xk-nI/edit?usp=sharing>)

| **Característica** | **Random Forest (RF)** | **SVM** | **XGBoost** | **KNN** | **Regresión Logística** |
| --- | --- | --- | --- | --- | --- |
| **Facilidad de aplicación** | Media | Media | Baja | Alta | Alta |
| **Datos no lineales** | **Media** | **Alta (con kernel)** | **Alta** | **Baja** | **Baja** |
| **Tolerancia a datos ruidosos** | Alta | Baja | Media | Baja | Media |
| **Esfuerzo computacional** | Medio | Alto | Alto | Alto | Bajo |
| **Grandes medianas** | Medio | Bajo | Alta | Bajo | Alta |
| **Tareas de clasificación** | Sí | Sí | Sí | Sí | Sí |
| **Requiere datos estandarizados** | No | Sí | No | Sí | Sí |
| **Escalabilidad** | Media | Baja | Alta | Baja | Alta |
| **Riesgo de sobreajuste** | Bajo | Bajo | Medio-alto (en datasets pequeños) | Alto | Bajo |

* Actividad 0.3.4 Evaluar con validación cruzada + test final: Para la evaluación de ambos modelos se usó una metodología híbrida (nested cross validation), mezcla de k-fold con train-test split (80/20). Se pudo analizar la evaluación según las métricas de rendimiento establecidas .

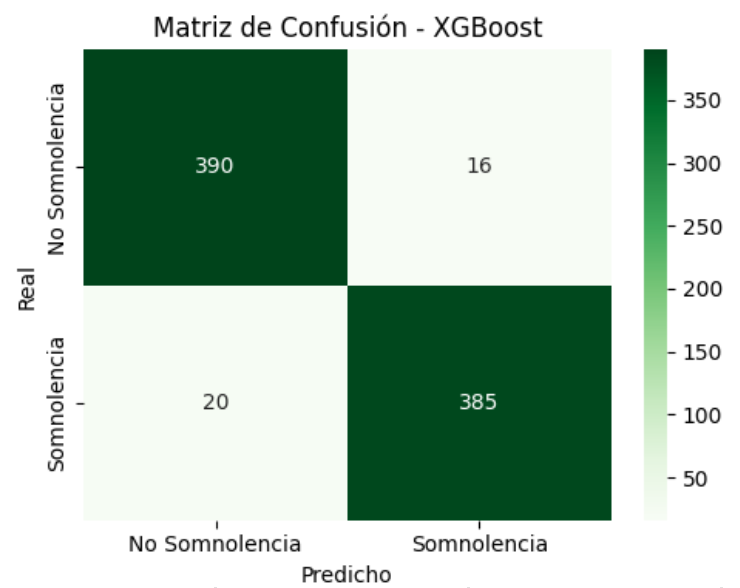
| **RF - Entrenamiento con validación cruzada** | | | |
| --- | --- | --- | --- |
| **Métrica** | **No Somnolencia** | **Somnolencia** | **Accuracy** |
| Precision | 0.93 | 0.95 | 0.94 |
| Recall | 0.95 | 0.93 |  |
| F1-Score | 0.94 | 0.94 |  |
| Support | 1620 | 1621 | 3241 |

| **RF - Test Final** | | | |
| --- | --- | --- | --- |
| **Métrica** | **No Somnolencia** | **Somnolencia** | **Accuracy** |
| Precision | 0.95 | 0.96 | 0.95 |
| Recall | 0.96 | 0.95 |  |
| F1-Score | 0.95 | 0.95 |  |
| Support | 406 | 405 | 811 |



| **XGB - Entrenamiento con validación cruzada** | | | |
| --- | --- | --- | --- |
| **Métrica** | **No Somnolencia** | **Somnolencia** | **Accuracy** |
| Precision | 0.93 | 0.94 | 0.94 |
| Recall | 0.94 | 0.93 |  |
| F1-Score | 0.94 | 0.93 |  |
| Support | 1620 | 1621 | 3241 |

| **XGB - TEST FINAL** | | | |
| --- | --- | --- | --- |
| **Métrica** | **No Somnolencia** | **Somnolencia** | **Accuracy** |
| Precision | 0.95 | 0.96 | 0.96 |
| Recall | 0.96 | 0.95 |  |
| F1-Score | 0.96 | 0.96 |  |
| Support | 406 | 405 | 811 |



| **Comparativa de test final** | | | |
| --- | --- | --- | --- |
| **Métrica** | **Clase** | **Random Forest (RF)** | **XGBoost (XGB)** |
| Accuracy | Global | 0.95 | **0.96** |
| Precision | No Somnolencia | 0.95 | 0.95 |
| Precision | Somnolencia | 0.96 | 0.96 |
| Recall | No Somnolencia | 0.96 | 0.96 |
| Recall | Somnolencia | 0.95 | 0.95 |
| F1-Score | No Somnolencia | 0.95 | **0.96** |
| F1-Score | Somnolencia | 0.95 | **0.96** |

En base a los resultados obtenidos, ambos algoritmos muestran un buen desempeño en la tarea de clasificación. Sin embargo, al analizar las métricas del conjunto de prueba (compuesto por datos no vistos durante el entrenamiento) se observa que el algoritmo XGBoost presenta una ligera ventaja sobre Random Forest. En las matrices de confusión, XGBoost reduce tanto los falsos positivos como los falsos negativos, lo cual es crítico en la detección de somnolencia. Además, en el test final, ambos modelos obtuvieron un Recall del 96% y una Precision del 95%, pero XGBoost alcanzó un F1-Score de 96% y Accuracy (predicciones acertadas) del 96%, superando al modelo RF por 1%. Si bien la validación cruzada fue fundamental para comparar modelos de forma objetiva y prevenir sobreajuste, la selección final se basó en el rendimiento con datos completamente nuevos. Por estas razones, se eligió XGBoost como el modelo definitivo para la implementación.

### Evidencias de Código

#### Historia de Usuario 0.2 — Preprocesamiento de datos

Archivo: Modelo/extraccion\_metricas.py

import cv2

import mediapipe as mp

import pandas as pd

import os

import math

import numpy as np

# Función para calcular distancia euclidiana

def euclidean(p1, p2):

return ((p1[0]-p2[0])\*\*2 + (p1[1]-p2[1])\*\*2)\*\*0.5

# Función para calcular EAR (ambos ojos)

def compute\_ear(landmarks):

# Ojo izquierdo

left = [landmarks[i] for i in [33, 160, 158, 133, 153, 144]]

left\_ear = (euclidean(left[1], left[5]) + euclidean(left[2], left[4])) / (2.0 \* euclidean(left[0], left[3]))

# Ojo derecho

right = [landmarks[i] for i in [362, 385, 387, 263, 373, 380]]

right\_ear = (euclidean(right[1], right[5]) + euclidean(right[2], right[4])) / (2.0 \* euclidean(right[0], right[3]))

return (left\_ear + right\_ear) / 2.0

# Función para calcular MAR

def compute\_mar(landmarks):

mouth = [landmarks[i] for i in [61, 291, 81, 178, 13, 14]]

mar = (euclidean(mouth[2], mouth[3]) + euclidean(mouth[4], mouth[5])) / (2.0 \* euclidean(mouth[0], mouth[1]))

return mar

# Función para calcular ángulos de orientación de la cabeza

def compute\_tait\_bryan\_angles(landmarks\_3d, img\_shape):

ref\_indices = [1, 152, 33, 263, 61, 291] # nariz, barbilla, ojos, boca

image\_points = np.array([(landmarks\_3d[i].x \* img\_shape[1], landmarks\_3d[i].y \* img\_shape[0]) for i in ref\_indices], dtype="double")

model\_points = np.array([

(0.0, 0.0, 0.0), # nariz

(0.0, -63.0, -12.0), # barbilla

(-43.0, 32.0, -26.0), # ojo izq

(43.0, 32.0, -26.0), # ojo der

(-28.0, -28.0, -24.0), # boca izq

(28.0, -28.0, -24.0) # boca der

])

focal\_length = img\_shape[1]

center = (img\_shape[1] / 2, img\_shape[0] / 2)

camera\_matrix = np.array([[focal\_length, 0, center[0]],

[0, focal\_length, center[1]],

[0, 0, 1]], dtype="double")

dist\_coeffs = np.zeros((4, 1)) # Sin distorsión

success, rotation\_vector, \_ = cv2.solvePnP(model\_points, image\_points, camera\_matrix, dist\_coeffs)

if not success:

return None, None, None

rotation\_mat, \_ = cv2.Rodrigues(rotation\_vector)

sy = math.sqrt(rotation\_mat[0, 0] \*\* 2 + rotation\_mat[1, 0] \*\* 2)

singular = sy < 1e-6

if not singular:

pitch = math.degrees(math.atan2(rotation\_mat[2, 1], rotation\_mat[2, 2]))

yaw = math.degrees(math.atan2(-rotation\_mat[2, 0], sy))

roll = math.degrees(math.atan2(rotation\_mat[1, 0], rotation\_mat[0, 0]))

else:

pitch = math.degrees(math.atan2(-rotation\_mat[1, 2], rotation\_mat[1, 1]))

yaw = math.degrees(math.atan2(-rotation\_mat[2, 0], sy))

roll = 0

return pitch, yaw, roll

# Configurar FaceMesh

mp\_face\_mesh = mp.solutions.face\_mesh

face\_mesh = mp\_face\_mesh.FaceMesh(static\_image\_mode=True)

# Función para procesar imágenes de un directorio

def procesar\_frames(directorio, clase):

datos = []

for filename in os.listdir(directorio):

if filename.lower().endswith(('.jpg', '.png')):

ruta = os.path.join(directorio, filename)

img = cv2.imread(ruta)

if img is None:

continue

rgb = cv2.cvtColor(img, cv2.COLOR\_BGR2RGB)

results = face\_mesh.process(rgb)

if results.multi\_face\_landmarks:

for face in results.multi\_face\_landmarks:

h, w, \_ = img.shape

puntos = [(int(p.x \* w), int(p.y \* h)) for p in face.landmark]

ear = compute\_ear(puntos)

mar = compute\_mar(puntos)

pitch, yaw, roll = compute\_tait\_bryan\_angles(face.landmark, img.shape)

if pitch is None:

continue

datos.append([filename, ear, mar, pitch, yaw, roll, clase])

return datos

# Directorios de entrada y salida

ruta\_somnolencia = r"C:\ruta\a\frames\_sin\_metricas\somnolencia"

ruta\_no\_somnolencia = r"C:\ruta\a\frames\_sin\_metricas\no\_somnolencia"

ruta\_salida = r"C:\ruta\a\frames\_sin\_metricas\metricas\_somnolenciaf.csv"

# Procesamiento y guardado

datos\_somnolencia = procesar\_frames(ruta\_somnolencia, 1)

datos\_no\_somnolencia = procesar\_frames(ruta\_no\_somnolencia, 0)

df = pd.DataFrame(datos\_somnolencia + datos\_no\_somnolencia,

columns=["frame", "EAR", "MAR", "Pitch", "Yaw", "Roll", "Clase"])

df.to\_csv(ruta\_salida, index=False)

print("¡CSV guardado correctamente en:", ruta\_salida)

Archivo: Modelo/escalado\_caracteristicas.py

#sin rutas absolutas, cuando los archivos se manejan dentro de la carpeta del programa

from sklearn.preprocessing import MinMaxScaler, StandardScaler

import pandas as pd

try:

df = pd.read\_csv("metricas\_somnolencia.csv")

# Columnas por tipo

minmax\_cols = ['EAR', 'MAR']

standard\_cols = ['Pitch', 'Yaw', 'Roll']

# Aplicar escalado

df[minmax\_cols] = MinMaxScaler().fit\_transform(df[minmax\_cols])

df[standard\_cols] = StandardScaler().fit\_transform(df[standard\_cols])

# Exportar si deseas probar con el modelo

df.to\_csv("datos\_escalados\_mixto.csv", index=False)

except Exception as e:

print(f"Error durante el procesamiento: {str(e)}")

Archivo: Modelo/aumentacion\_datos.py

import cv2

import os

import random

import numpy as np

# --- Configuración ---

INPUT\_BASE = r"C:\Users\USER\Documents\DATESET\DS\_filtrada" #Usa doble barra o raw string (r"...") es la dirección de donde salen las frames

OUTPUT\_BASE = r"C:\Users\USER\Documents\DATESET\DS\_filtrada\frames\_aumentadass" #es la dirección donde se guardan las frames

os.makedirs(OUTPUT\_BASE, exist\_ok=True) #Crea la carpeta si no existe

# --- Funciones de aumento ---

# Ajuste de brillo y contraste (TU COMENTARIO ORIGINAL)

def ajustar\_brillo(img):

"""Brillo aleatorio suave (-20 a +20)""" #Rango de brillo más realista y seguro

brillo = random.randint(-30, 30)

return cv2.convertScaleAbs(img, beta=brillo)

# Rotación de imagen (TU COMENTARIO ORIGINAL)

def rotar\_imagen(img, angulo):

"""Rotación con bordes reflejados (para no perder información)""" #BORDER\_REFLECT evita artefactos

(h, w) = img.shape[:2]

centro = (w // 2, h // 2)

M = cv2.getRotationMatrix2D(centro, angulo, 1.0)

return cv2.warpAffine(img, M, (w, h), borderMode=cv2.BORDER\_REFLECT)

# Aumentar con aleatoriedad controlada (TU COMENTARIO ORIGINAL + MEJORA)

def aplicar\_aumento(img, clase):

"""Aplicar 1 transformación aleatoria, controlada por clase"""

# Opciones ponderadas (50% brillo, 50% rotación) # weights asegura balance

opcion = random.choices(['brillo', 'rotacion'], weights=[0.5, 0.5], k=1)[0]

if opcion == 'brillo':

return ajustar\_brillo(img)

else:

# Rotación más amplia para somnolencia (-15° a +15° vs ±5°)

angulo = random.uniform(-15, 15) if clase == "somnolencia" else random.uniform(-5, 5)

return rotar\_imagen(img, angulo)

# --- Procesamiento por clase --- (TU COMENTARIO ORIGINAL + MANEJO DE ERRORES)

def aumentar\_clase(clase):

input\_dir = os.path.join(INPUT\_BASE, clase)

output\_dir = os.path.join(OUTPUT\_BASE, clase)

os.makedirs(output\_dir, exist\_ok=True) #Asegura que exista la subcarpeta

total\_ok = 0

total\_error = 0

for filename in os.listdir(input\_dir):

if filename.endswith((".jpg", ".png")): #Soporta ambos formatos

try:

path = os.path.join(input\_dir, filename)

img = cv2.imread(path)

if img is None:

raise ValueError(f"Error al leer {filename}")

# Aplicar aumento y guardar (TU LÓGICA ORIGINAL)

img\_aug = aplicar\_aumento(img, clase)

nombre\_salida = f"aug\_{filename}"

cv2.imwrite(os.path.join(output\_dir, nombre\_salida), img\_aug)

total\_ok += 1

except Exception as e:

print(f"⚠️ Error en {filename}: {str(e)}")

total\_error += 1

continue

# Mensaje detallado por clase

print(f"\nResultados para clase '{clase}':")

print(f" - Frames procesados: {total\_ok}")

print(f" - Errores: {total\_error}")

return total\_ok > 0 # Retorna True si al menos 1 frame se procesó

# --- Ejecución --- (CON REPORTE FINAL MEJORADO)

if \_\_name\_\_ == "\_\_main\_\_":

print(" Iniciando data augmentation...\n")

# Procesar ambas clases

resultado\_somnolencia = aumentar\_clase("somnolencia")

resultado\_no\_somnolencia = aumentar\_clase("no\_somnolencia")

# ✨ Reporte consolidado

if resultado\_somnolencia and resultado\_no\_somnolencia:

print("\n✅ ¡Proceso completado con éxito!")

print(f" - Frames aumentados guardados en: {OUTPUT\_BASE}")

else:

print("\n❌ ¡Proceso completado con errores!")

print(" Verifica los mensajes anteriores.")

#### Historia de Usuario 0.3 — Entrenamiento del modelo

Archivo: Modelo/modelo\_rf.py

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.ensemble import RandomForestClassifier

from sklearn.model\_selection import cross\_val\_predict, StratifiedKFold

from sklearn.metrics import classification\_report, confusion\_matrix

# Cargar train y test

train = pd.read\_csv("train.csv")

test = pd.read\_csv("test.csv")

X\_train = train[["EAR", "MAR", "Pitch", "Yaw", "Roll"]]

y\_train = train["Clase"]

X\_test = test[["EAR", "MAR", "Pitch", "Yaw", "Roll"]]

y\_test = test["Clase"]

# Configurar validación cruzada

kfold = StratifiedKFold(n\_splits=5, shuffle=True, random\_state=42)

# Crear el modelo

modelo\_rf = RandomForestClassifier(random\_state=42)

# Validación cruzada con los datos de entrenamiento

y\_pred\_cv = cross\_val\_predict(modelo\_rf, X\_train, y\_train, cv=kfold)

print("\n=== Random Forest - Validación Cruzada ===")

print(classification\_report(y\_train, y\_pred\_cv, target\_names=["No Somnolencia", "Somnolencia"]))

# Entrenamiento final

modelo\_rf.fit(X\_train, y\_train)

# Evaluación final en test

y\_pred\_test = modelo\_rf.predict(X\_test)

print("\n=== Random Forest - Test Final ===")

print(classification\_report(y\_test, y\_pred\_test, target\_names=["No Somnolencia", "Somnolencia"]))

# Matriz de confusión

cm = confusion\_matrix(y\_test, y\_pred\_test)

plt.figure(figsize=(5, 4))

sns.heatmap(cm, annot=True, fmt="d", cmap="Blues", xticklabels=["No Somnolencia", "Somnolencia"], yticklabels=["No Somnolencia", "Somnolencia"])

plt.title("Matriz de Confusión - Random Forest")

plt.xlabel("Predicho")

plt.ylabel("Real")

plt.tight\_layout()

plt.show()

Archivo: Modelo/modelo\_xgb.py

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

from xgboost import XGBClassifier

from sklearn.model\_selection import cross\_val\_predict, StratifiedKFold

from sklearn.metrics import classification\_report, confusion\_matrix

# Cargar train y test

train = pd.read\_csv("train.csv")

test = pd.read\_csv("test.csv")

X\_train = train[["EAR", "MAR", "Pitch", "Yaw", "Roll"]]

y\_train = train["Clase"]

X\_test = test[["EAR", "MAR", "Pitch", "Yaw", "Roll"]]

y\_test = test["Clase"]

# Configurar validación cruzada

kfold = StratifiedKFold(n\_splits=5, shuffle=True, random\_state=42)

# Crear el modelo

modelo\_xgb = XGBClassifier(random\_state=42, use\_label\_encoder=False, eval\_metric='logloss')

# Validación cruzada

y\_pred\_cv = cross\_val\_predict(modelo\_xgb, X\_train, y\_train, cv=kfold)

print("\n=== XGBoost - Validación Cruzada ===")

print(classification\_report(y\_train, y\_pred\_cv, target\_names=["No Somnolencia", "Somnolencia"]))

# Entrenamiento final

modelo\_xgb.fit(X\_train, y\_train)

# Evaluación final en test

y\_pred\_test = modelo\_xgb.predict(X\_test)

print("\n=== XGBoost - Test Final ===")

print(classification\_report(y\_test, y\_pred\_test, target\_names=["No Somnolencia", "Somnolencia"]))

# Matriz de confusión

cm = confusion\_matrix(y\_test, y\_pred\_test)

plt.figure(figsize=(5, 4))

sns.heatmap(cm, annot=True, fmt="d", cmap="Greens", xticklabels=["No Somnolencia", "Somnolencia"], yticklabels=["No Somnolencia", "Somnolencia"])

plt.title("Matriz de Confusión - XGBoost")

plt.xlabel("Predicho")

plt.ylabel("Real")

plt.tight\_layout()

plt.show()

Archivo: Modelo/limpieza\_split\_datos.py

import pandas as pd

from sklearn.model\_selection import train\_test\_split

# Cargar el CSV original

df = pd.read\_csv("TOTALmetricas\_somnolencia.csv") # <-- Reemplaza con el path correcto

# Verificar si hay valores vacíos o nulos

print("Valores nulos por columna:\n", df.isnull().sum())

# Separar variables predictoras y etiqueta

X = df[['EAR', 'MAR', 'Pitch', 'Yaw', 'Roll']]

y = df['Clase']

# División 80/20

X\_train, X\_test, y\_train, y\_test = train\_test\_split(

X, y, test\_size=0.2, random\_state=42, stratify=y

)

# Unir X e y antes de guardar

train = pd.concat([X\_train, y\_train], axis=1)

test = pd.concat([X\_test, y\_test], axis=1)

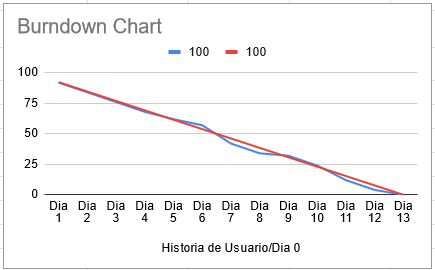
train.to\_csv("train.csv", index=False)

test.to\_csv("test.csv", index=False)

print("Split completado y archivos guardados como train.csv y test.csv")

### Burndown Chart

| Backlog ID | Historia de Usuario | Tiempo Estimado | 08/05 | 09/05 | 10/05 | 11/05 | 12/05 | 13/05 | 14/05 | 15/  05 | 16/  05 | 17/  05 | 18/  05 | 19/  05 | 20/  05 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Dia1** | **Dia2** | **Dia3** | **Dia4** | **Dia5** | **Dia6** | **Dia 7** | **Dia 8** | **Dia 9** | **Dia 10** | **Dia 11** | **Dia 12** | **Dia 13** |
| HU-0.1 | Recolección y Etiquetado del dataset | 8 | 8 | 8 | 4 |  |  |  |  |  |  |  |  |  |  |
| HU-0.2 | Preprocesamiento de datos |  |  |  | 4 | 6 | 5 | **15** |  |  |  |  |  |  |  |
| HU-0.3 | Entrenamiento del Modelo |  |  |  |  |  |  |  |  | 8 | 2 | 8 | **12** | 8 | 4 |
| Tiempo de trabajo | | 100 | 92 | 84 | 76 | 68 | 62 | 57 | 42 | 34 | 32 | 24 | 12 | 4 | 0 |



### Análisis del Sprint

* Se cumplió con todas las tareas estimadas.
* El progreso fue constante y sin bloqueos mayores.
* Se logró un buen ritmo de trabajo.
* Algunas actividades tomaron menos tiempo de lo estimado por lo que se adelantaron 2 actividades del Sprint 0.2. para mantener los días de avance del sprint 0.1.

### Conclusiones

El Sprint 0.1 concluyó exitosamente con todas las historias de usuario. Se hizo una búsqueda dataset para el preprocesamiento de datos, luego se hizo una selección y balanceo de frames por clase, se aumentaron dichos datos, se hizo una selección de algoritmos para luego entrenarlos y validar los resultados para finalmente hacer una selección del algoritmo final.

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.

También se pudo ver que la actividad “estandarización y normalización de datos” no era necesaria, como recomendación, es mejor hacerlos después de la elección del algoritmos para ver si este lo requiere o no. Las actividades “Aumentar datos” (id 0.2.2) y “Desarrollar modelo ML base” (id 0.3.2) se terminaron en menos tiempo del programado, por lo que se agregaron 2 actividades más al Sprint 0.1, la actividad “Entrenar modelo” (id 0.3.3) y “Evaluar con validación cruzada más métricas” (id 0.3.4).

## Sprint 0.2

### Historias de Usuario y Tareas

| Backlog ID | Historia de Usuario | Tiempo Estimado | Fecha de Inicio | Fecha de Finalización |
| --- | --- | --- | --- | --- |
| HU-0.3 | Entrenamiento del Modelo | 8 | 23/05 | 25/05/2025 |
| HU-0.4 | Integración del Modelo al Flujo Actual | 5 | 26/05 | 28/05/2025 |

| ID | Tarea | Responsable | Estimación (Horas) | Estado |
| --- | --- | --- | --- | --- |
| 0.3.3 | Entrenar modelo (SVM / Random Forest) | Data Scientist | 30 | Pendiente |
| 0.3.4 | Evaluar con validación cruzada + métricas | Data Scientist | 15 | Pendiente |
| 0.3.3 | Documentar proceso de entrenamiento | Data Scientist | 10 | Pendiente |
| 0.4.1 | Exportar modelo en formato .pkl | ML Engineer | 10 | Pendiente |
| 0.4.2 | Integrar modelo en entorno de prueba/programa final. | QA Engineer | 10 | Pendiente |

### Evidencias de Código

#### Historia de Usuario 0.4 — Integración del Modelo al Flujo Actual

Archivo: Modelo/modelo\_xgb.py

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

from xgboost import XGBClassifier

from sklearn.model\_selection import cross\_val\_predict, StratifiedKFold

from sklearn.metrics import classification\_report, confusion\_matrix

import pickle #para guardar modelo entrenado

# Cargar train y test

train = pd.read\_csv("train.csv")

test = pd.read\_csv("test.csv")

X\_train = train[["EAR", "MAR", "Pitch", "Yaw", "Roll"]]

y\_train = train["Clase"]

X\_test = test[["EAR", "MAR", "Pitch", "Yaw", "Roll"]]

y\_test = test["Clase"]

# Configurar validación cruzada

kfold = StratifiedKFold(n\_splits=5, shuffle=True, random\_state=42)

# Crear el modelo

modelo\_xgb = XGBClassifier(random\_state=42, use\_label\_encoder=False, eval\_metric='logloss')

# Validación cruzada

y\_pred\_cv = cross\_val\_predict(modelo\_xgb, X\_train, y\_train, cv=kfold)

print("\n=== XGBoost - Validación Cruzada ===")

print(classification\_report(y\_train, y\_pred\_cv, target\_names=["No Somnolencia", "Somnolencia"]))

# Entrenamiento final

modelo\_xgb.fit(X\_train, y\_train)

# Evaluación final en test

y\_pred\_test = modelo\_xgb.predict(X\_test)

print("\n=== XGBoost - Test Final ===")

print(classification\_report(y\_test, y\_pred\_test, target\_names=["No Somnolencia", "Somnolencia"]))

# Matriz de confusión

cm = confusion\_matrix(y\_test, y\_pred\_test)

plt.figure(figsize=(5, 4))

sns.heatmap(cm, annot=True, fmt="d", cmap="Greens", xticklabels=["No Somnolencia", "Somnolencia"], yticklabels=["No Somnolencia", "Somnolencia"])

plt.title("Matriz de Confusión - XGBoost")

plt.xlabel("Predicho")

plt.ylabel("Real")

plt.tight\_layout()

plt.show()

# Guardar el modelo entrenado como archivo .pkl

with open("modelo\_xgboost.pkl", "wb") as f:

pickle.dump(modelo\_xgb, f)

print("Modelo XGBoost guardado como modelo\_xgboost.pkl")

Archivo: Modelo/analisis\_somnolencia.py

import pickle

import numpy as np

from deteccion\_facial import lectura\_datos

# Cargar modelo previamente entrenado

with open("modelo\_xgboost.pkl", "rb") as f:

modelo = pickle.load(f)

# Supón que ya tienes los datos de un frame:

EAR, MAR, Pitch, Yaw, Roll = lectura\_datos()

# Crear input para el modelo

X\_nuevo = np.array([[EAR, MAR, Pitch, Yaw, Roll]])

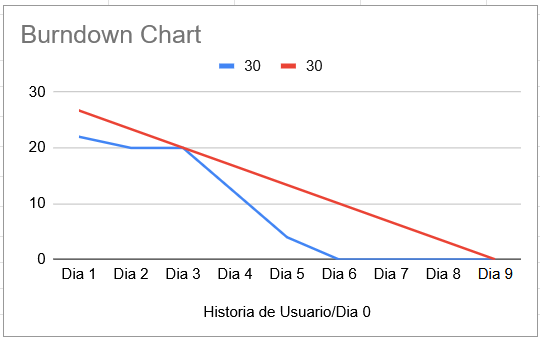
# Hacer predicción

prediccion = modelo.predict(X\_nuevo)

### Burndown Chart

| Backlog ID | Historia de Usuario | Tiempo Estimado | 23/05 | 24/05 | 25/05 | 26/05 | 27/05 | 28/05 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Dia1** | **Dia2** | **Dia3** | **Dia4** | **Dia5** | **Dia6** |
| HU-0.3 | Entrenamiento del Modelo | 10 | 8 | 2 |  |  |  |  |
| HU-0.4 | Integración del modelo al flujo actual | 20 |  |  |  | 8 | 8 | 4 |
| Tiempo de trabajo | | 30 | 22 | 20 | 20 | 12 | 4 | 0 |

### 



### Análisis del Sprint

* Se cumplió con todas las tareas estimadas.
* El progreso fue constante y sin bloqueos mayores.
* Se logró un buen ritmo de trabajo.
* El sprint se terminó en menos días por las actividades que se adelantaron en el sprint anterior.
* Se mantuvieron los días en los que se avanzó el sprint (6 días) ya que está más alineado a la duración de un sprint regular (1 semana).
* Se cambió la actividad 0.4.2 “Probar predicción sobre CSV actual” a “Integrar modelo en entorno de prueba/programa final”, porque el equipo encontró que la actividad antigua era redundante e innecesaria ya que el algoritmo fue entrenado con el CSV mencionado.

### Conclusiones

El Sprint se concluyó exitosamente con todas las historias de usuario. Se finalizó la documentación del avance, para luego guardar el modelo elegido y entrenado, y finalmente se integró dicho modelo en el flujo del programa del Sprint 1.

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.

El sprint duró menos días de lo planeado debido a que se adelantaron 2 actividades del mismo en un sprint anterior. No se agregaron más actividades para llenar el hueco de días ya que se vió que con 6 días de duración el sprint 0.2 estaba mejor alineado a una duración de sprint regular (1 semana) en comparación con una duración de 9 días (estimación original).

## Sprint 1

### 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 |

| ID | Tarea | Responsable | Estimación (Horas) | Estado |
| --- | --- | --- | --- | --- |
| 1.1.1 | Configurar cámara y flujo de video en tiempo real | Backend Dev | 18 | Pendiente |
| 1.1.2 | Implementar detección facial con OpenCV/MediaPipe | Computer Vision Engineer | 24 | Pendiente |
| 1.1.3 | Optimizar latencia (<1 segundo por frame) | Backend Dev | 18 | Pendiente |
| 1.1.4 | Pruebas con diferentes ángulos y condiciones de luz | QA Engineer | 10 | Pendiente |
| 1.1.5 | Documentar configuración y requisitos de hw | Technical Writer | 6 | Pendiente |
| 1.2.1 | Implementar cálculo de PERCLOS con landmarks oculares | Computer Vision Engineer | 22 | Pendiente |
| 1.2.2 | Filtrar parpadeos rápidos (<0.5 segundos) | Data Scientist | 14 | Pendiente |

### Evidencias de Código

#### 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/alarma\_led.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

from analisis\_somnolencia import calibrar\_usuario

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,EAR\_UMBRAL)

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()

# Realizar calibración al iniciar

EAR\_UMBRAL = calibrar\_usuario()

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

#### 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

import time

# 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,umbral\_ear): #usa el ear detectado y el umbral calculado en la calibración

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 < umbral\_ear:

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

def calibrar\_usuario(duracion\_segundos=5, fps\_estimado=10):

#Realiza calibración EAR para adaptar el umbral según el usuario.

muestras\_ear = []

total\_frames = duracion\_segundos \* fps\_estimado

print("🔧 Calibración: mantén los ojos abiertos...")

time.sleep(2)

for \_ in range(total\_frames):

ear = mandar\_ear()

if ear > 0:

muestras\_ear.append(ear)

time.sleep(1 / fps\_estimado)

if len(muestras\_ear) >= fps\_estimado \* 2:

promedio\_ear\_abierto = np.mean(muestras\_ear)

nuevo\_umbral = promedio\_ear\_abierto \* 0.7

print(f"✅ EAR calibrado: promedio = {promedio\_ear\_abierto:.3f}, nuevo umbral = {nuevo\_umbral:.3f}")

return nuevo\_umbral

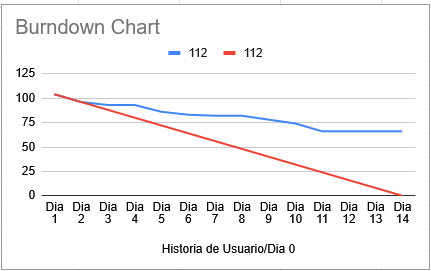
else:

print("⚠️ Calibración insuficiente, usando umbral por defecto.")

return 0.25 # Umbral fijo si falla

### 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 | 8 | 8 | 3 |  | 7 | 3 |  |  |  |  |  |  |  |  |
| HU-2 | Monitoreo de Cierre Ocular | 54 |  |  |  |  |  |  | 1 |  | 4 | 4 | 8 | 8 |  |  |
| Tiempo de trabajo | | 112 | 104 | 96 | 93 | 93 | 86 | 83 | 82 | 82 | 78 | 74 | 66 | 58 | 50 | 50 |



### 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.

### 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.

## Sprint 1-2

### Historias de Usuario y Tareas

| Backlog ID | Historia de Usuario | Tiempo Estimado | Fecha de Inicio | Fecha de Finalización |
| --- | --- | --- | --- | --- |
| HU-1.1 | Detectar rostros con cámara | 8 pts | 01/06/2025 | 03/06/2025 |
| HU-1.2 | Monitoreo de Cierre Ocular | 5 pts | 04/06/2025 | 04/06/2025 |
| HU-1.3 | Detección de Bostezos | 4 pts | 04/06/2025 | 04/06/2025 |
| HU-1.4 | Medición de Inclinación de Cabeza | 4 pts | 04/05/2025 | 09/06/2025 |
| HU-2.1 | Alerta Temprana de Somnolencia | 4 pts | 10/06/2025 | 14/06/2025 |

| ID | Tarea | Responsable | Estimación (Horas) | Estado |
| --- | --- | --- | --- | --- |
| 1.1.1 | Configurar cámara y flujo de video en tiempo real | Developers Team | 3 | Pendiente |
| 1.1.2 | Implementar detección facial con OpenCV/MediaPipe | Computer Vision Engineer | 3 | Pendiente |
| 1.1.3 | Optimizar latencia (<1 segundo por frame) | Developers Team | 3 | Pendiente |
| 1.1.4 | Pruebas con diferentes ángulos y condiciones de luz | QA Engineer | 2 | Pendiente |
| 1.1.5 | Documentar configuración y requisitos de hw | Technical Writer | 2 | Pendiente |
| 1.2.1 | Implementar cálculo de EAR con landmarks oculares | Computer Vision Engineer | 3 | Pendiente |
| 1.2.2 | Documentar algoritmo y métricas de validación | Technical Writer | 1 | Pendiente |
| 1.3.1 | Implementar Mouth Aspect Ratio (MAR) para bostezos | Computer Vision Engineer | 3 | Pendiente |
| 1.4.1 | Implementar detección de ángulo de cabeza (pitch/yaw/pitch) | Computer Vision Engineer | 3 | Pendiente |
| 1.4.2 | Pruebas de precisión | QA Engineer | 8 | Pendiente |
| 1.4.3 | Pruebas de función | QA Engineer | 8 | Pendiente |
| 2.1.1 | Diseñar interfaz de alerta visual | UX/UI Designer | 8 | Pendiente |
| 2.1.2 | Implementar alarma sonora (85 dB, no intrusiva) | Developers Team | 8 | Pendiente |

### Evidencias de Código

#### Historia de Usuario 1.1 — Detectar rostros con cámara

Archivo: Modelo/captura\_video.py

import cv2

import threading

#Se inicia la captura de video

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

#### Historia de Usuario 1.2 — Monitoreo de Cierre Ocular

#### Historia de Usuario 1.3 — Detección de Bostezos

#### Historia de Usuario 1.4 — Medición de Inclinación de Cabeza

Archivo: Modelo/deteccion\_facial.py

#FUNCIONA para flujo de video con lm específicos

import cv2

import mediapipe as mp

import numpy as np

import math

from capturar\_video import obtener\_frame, iniciar\_hilo\_captura

# Landmarks específicos

landmarks\_faciales = [33, 160, 158, 133, 153, 144, 362, 385, 387, 263, 373, 380, 61, 291, 81, 178, 13, 14, 1, 152, 33, 263, 61, 291]

# Función para calcular distancia euclidiana (para mar y ear)

def euclidean(p1, p2):

return ((p1[0]-p2[0])\*\*2 + (p1[1]-p2[1])\*\*2)\*\*0.5

# Función para calcular EAR (ambos ojos)

def compute\_ear(landmarks):

# Ojo izquierdo

left = [landmarks[i] for i in [33, 160, 158, 133, 153, 144]]

left\_ear = (euclidean(left[1], left[5]) + euclidean(left[2], left[4])) / (2.0 \* euclidean(left[0], left[3]))

# Ojo derecho

right = [landmarks[i] for i in [362, 385, 387, 263, 373, 380]]

right\_ear = (euclidean(right[1], right[5]) + euclidean(right[2], right[4])) / (2.0 \* euclidean(right[0], right[3]))

ear = (left\_ear + right\_ear) / 2.0

return ear

# Función para calcular MAR

def compute\_mar(landmarks):

mouth = [landmarks[i] for i in [61, 291, 81, 178, 13, 14]]

mar = (euclidean(mouth[2], mouth[3]) + euclidean(mouth[4], mouth[5])) / (2.0 \* euclidean(mouth[0], mouth[1]))

return mar

# Función para calcular ángulos de orientación de la cabeza

def compute\_tait\_bryan\_angles(landmarks\_3d, img\_shape):

ref\_indices = [1, 152, 33, 263, 61, 291] # nariz, barbilla, ojos, boca

image\_points = np.array([(landmarks\_3d[i].x \* img\_shape[1], landmarks\_3d[i].y \* img\_shape[0]) for i in ref\_indices], dtype="double")

model\_points = np.array([

(0.0, 0.0, 0.0), # nariz

(0.0, -63.0, -12.0), # barbilla

(-43.0, 32.0, -26.0), # ojo izq

(43.0, 32.0, -26.0), # ojo der

(-28.0, -28.0, -24.0), # boca izq

(28.0, -28.0, -24.0) # boca der

])

focal\_length = img\_shape[1]

center = (img\_shape[1] / 2, img\_shape[0] / 2)

camera\_matrix = np.array([[focal\_length, 0, center[0]],

[0, focal\_length, center[1]],

[0, 0, 1]], dtype="double")

dist\_coeffs = np.zeros((4, 1)) #Cámara sin distorsión

success, rotation\_vector, \_ = cv2.solvePnP(model\_points, image\_points, camera\_matrix, dist\_coeffs)

if not success:

return None, None, None

rotation\_mat, \_ = cv2.Rodrigues(rotation\_vector)

sy = math.sqrt(rotation\_mat[0, 0] \*\* 2 + rotation\_mat[1, 0] \*\* 2)

singular = sy < 1e-6

if not singular:

pitch = math.degrees(math.atan2(rotation\_mat[2, 1], rotation\_mat[2, 2]))

yaw = math.degrees(math.atan2(-rotation\_mat[2, 0], sy))

roll = math.degrees(math.atan2(rotation\_mat[1, 0], rotation\_mat[0, 0]))

else:

pitch = math.degrees(math.atan2(-rotation\_mat[1, 2], rotation\_mat[1, 1]))

yaw = math.degrees(math.atan2(-rotation\_mat[2, 0], sy))

roll = 0

return pitch, yaw, roll

# Configuración de MediaPipe

mp\_drawing = mp.solutions.drawing\_utils

mp\_face\_mesh = mp.solutions.face\_mesh

face\_mesh = mp\_face\_mesh.FaceMesh(

static\_image\_mode=False,

max\_num\_faces=1,

min\_detection\_confidence=0.5,

min\_tracking\_confidence=0.5

)

# Estilo de dibujo

estilo\_puntos = mp\_drawing.DrawingSpec(color=(255, 0, 255), thickness=2)

def procesar\_video():

iniciar\_hilo\_captura()

while True:

frame = obtener\_frame()

if frame is None:

continue

# Procesar frame

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

results = face\_mesh.process(rgb)

if results.multi\_face\_landmarks:

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

h, w, \_ = frame.shape

# Obtener landmarks

todos\_lm = [(int(lm.x \* w), int(lm.y \* h)) for lm in face\_landmarks.landmark]

puntos = [todos\_lm[idx] for idx in landmarks\_faciales]

# Dibujar landmarks

for (x, y) in puntos:

cv2.circle(frame, (x, y), 3, (255, 0, 255), -1)

# Calcular métricas

ear = compute\_ear(todos\_lm)

mar = compute\_mar(todos\_lm)

pitch, yaw, roll = compute\_tait\_bryan\_angles(face\_landmarks.landmark, frame.shape)

# Mostrar métricas en pantalla

'''cv2.putText(frame, f"EAR: {ear:.2f}", (10, 30), cv2.FONT\_HERSHEY\_SIMPLEX, 0.7, (0, 255, 0), 2)

cv2.putText(frame, f"MAR: {mar:.2f}", (10, 60), cv2.FONT\_HERSHEY\_SIMPLEX, 0.7, (0, 255, 0), 2)'''

# Mostrar frame

#cv2.imshow("Detección Facial", frame)

# Salir con 'q'

#if cv2.waitKey(1) & 0xFF == ord('q'):

break

cv2.destroyAllWindows()

return ear, mar, pitch, yaw, roll #yo aumenté

# Ejecutar

if \_\_name\_\_ == "\_\_main\_\_":

procesar\_video()

#### Historia de Usuario 2.1 — Alerta Temprana de Somnolencia

Archivo: Modelo/gestor\_alertas.py

import tkinter as tk

import threading

import winsound

ventana = None

ventana\_abierta = False

buzzer\_thread = None

stop\_buzzer = False

def buzzer():

global stop\_buzzer

while not stop\_buzzer:

try:

winsound.Beep(1000, 1000)

except RuntimeError:

break

def iniciar\_ventana():

global ventana, ventana\_abierta, buzzer\_thread, stop\_buzzer

if ventana\_abierta:

return

ventana\_abierta = True

stop\_buzzer = False

def crear\_ventana():

global ventana, ventana\_abierta, stop\_buzzer

def on\_closing():

global ventana, ventana\_abierta, stop\_buzzer

stop\_buzzer = True

ventana\_abierta = False

if ventana:

ventana.destroy()

ventana = None

ventana = tk.Tk()

ventana.title("ALERTA DE SOMNOLENCIA")

ventana.geometry("400x150")

ventana.configure(bg="red")

label = tk.Label(

ventana,

text="¡ALERTA DE SOMNOLENCIA DETECTADA!",

font=("Helvetica", 14),

bg="red",

fg="white"

)

label.pack(pady=30)

ventana.protocol("WM\_DELETE\_WINDOW", on\_closing)

ventana.mainloop()

threading.Thread(target=crear\_ventana, daemon=True).start()

buzzer\_thread = threading.Thread(target=buzzer)

buzzer\_thread.start()

def cerrar\_ventana():

global ventana, ventana\_abierta, stop\_buzzer

if not ventana\_abierta:

return

stop\_buzzer = True

ventana\_abierta = False

def cerrar():

global ventana

if ventana:

ventana.destroy()

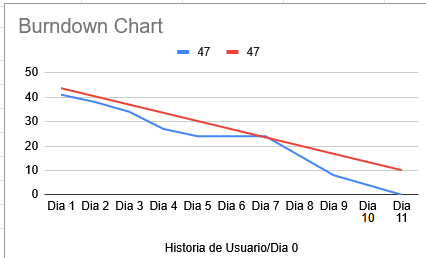
ventana = None

if ventana:

ventana.after(0, cerrar)

### Burndown Chart

| Backlog ID | Historia de Usuario | Tiempo Estimado | 01/06 | 02/06 | 03/06 | 04/06 | 05/06 | 06/06 | 07/06 | 08/06 | 09/06 | 10/06 | 11/06 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Dia1** | **Dia2** | **Dia3** | **Dia4** | **Dia5** | **Dia6** | **Dia 7** | **Dia 8** | **Dia 9** | **Dia 10** | **Dia 11** |
| HU-1 | Detectar rostros con cámara | 13 | 6 | 3 | 4 |  |  |  |  |  |  |  |  |
| HU-2 | Monitoreo de Cierre Ocular | 4 |  |  |  | 4 |  |  |  |  |  |  |  |
| HU-3 | Detección de Bostezos | 3 |  |  |  | 3 |  |  |  |  |  |  |  |
| HU-4 | Medición de Inclinación de Cabeza | 19 |  |  |  |  | 3 |  |  | 8 | 8 |  |  |
| HU-5 | Alerta Temmprana de Somnolencia | 8 |  |  |  |  |  |  |  |  |  | 4 | 4 |
| Tiempo de trabajo | | 47 | 41 | 38 | 34 | 27 | 24 | 24 | 24 | 16 | 8 | 4 | 0 |



### Análisis del Sprint

* Se realizó una reestructuración de los sprints tras el avance del Sprint 1 (después de la realización del Sprint 0)
* Se cumplió con todas las tareas estimadas.
* El progreso fue constante y sin bloqueos mayores.
* Se logró modularizar el código y documentarlo en comentarios.
* Las estimaciones fueron precisas y se logró un buen ritmo de trabajo.

### Conclusiones

El Sprint 1 concluyó exitosamente con todas las funcionalidades básicas implementadas. Algunas tareas tomaron menos tiempo que el inicial planteado en el sprint 1 original (ya que se tuvo que cambiar la estructura tomando en cuenta el sprint 0). Asimismo, se hizo una reestructuración de la distribución de las tareas en los sprints debido a que el avance inicial permitió terminar algunas tareas en menos tiempo y volvió alguna tareas obsoletas (se quitaron algunas tareas), lo que nos dejó con 3 sprints en total (0, 1 y 2).

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.

## Sprint 2

### Historias de Usuario y Tareas

| Backlog ID | Historia de Usuario | Tiempo Estimado | Fecha de Inicio | Fecha de Finalización |
| --- | --- | --- | --- | --- |
| HU-2.1 | Alerta temprana de Somnolencia | 4 pts | 16/06/2025 | 17/06/2025 |
| HU-3.1 | Exportar Datos a CSV | 3 pts | 18/06/2025 | 22/06/2025 |

| ID | Tarea | Responsable | Estimación (Horas) | Estado |
| --- | --- | --- | --- | --- |
| 2.1.3 | Configurar persistencia de alerta hasta confirmación | Developers Team | 5 | Pendiente |
| 2.1.4 | Pruebas de usabilidad | QA Engineer | 8 | Pendiente |
| 3.1.1 | Definir estructura del CSV (timestamp, PERCLOS, etc.) | Data Scientist | 2 | Pendiente |
| 3.1.2 | Implementar generación automática de CSV | Developers Team | 16 | Pendiente |
| 3.1.3 | Validar compatibilidad con Excel/Pandas | QA Engineer | 5 | Pendiente |
| 3.1.4 | Documentar formato y ejemplos de CSV | Technical Writer | 3 | Pendiente |
| 3.1.5 | Implementar generación automática de frames cuando se active la alarma | Developers Team | 8 | Pendiente |
| 3.1.6 | Documentar formato y ejemplos de frames | Technical Writer | 3 | Pendiente |

### Evidencias de Código

#### Historia de Usuario 2.1 — Alerta Temprana de Somnolencia

Archivo: Modelo/analisis\_somnolencia.py

import pickle

import numpy as np

from deteccion\_facial import procesar\_video

from gestor\_alertas import iniciar\_ventana, cerrar\_ventana

import keyboard

from capturar\_video import captura, iniciar\_hilo\_captura

import cv2

import time

from registro\_cvs import registrarcsv, guardarcsv

from registro\_alertas import guardar\_frame

somnolencia=0

registro = 0

# Cargar modelo previamente entrenado

with open("modelo\_xgboost.pkl", "rb") as f:

modelo = pickle.load(f)

iniciar\_hilo\_captura()

time.sleep(0.5) #Para que el hilo tenga tiempo de empezaar

while keyboard.is\_pressed('x') is False:

try:

# Se leen las métricas

EAR, MAR, Pitch, Yaw, Roll,frame = procesar\_video()

if None in (EAR, MAR, Pitch, Yaw, Roll): # Si no se detecta cara, omitir

continue #vuelve al inicio del blucle sin hacer nada más

# Crear input para el modelo

X\_nuevo =[[EAR, MAR, Pitch, Yaw, Roll]]

# Hacer predicciónx

prediccion = modelo.predict(X\_nuevo)

#print(EAR, MAR, Pitch, Yaw, Roll,prediccion[0])

if prediccion[0] == 1:

somnolencia += 1

if somnolencia >= 15: # Evita múltiples aperturas

if registro==0:

nombre\_frame = guardar\_frame(frame)

registrarcsv(EAR,MAR,Pitch,Yaw,Roll, nombre\_frame)

registro = 1

iniciar\_ventana()

else:

somnolencia = 0

registro = 0

cerrar\_ventana()

time.sleep(0.03) # 30 ms entre frames

except Exception as e:

print(f"Excepción en el bucle principal: {e}")

break

cv2.imshow("Detección Facial", frame)

if cv2.waitKey(1) & 0xFF == ord('x'): # Permite salir con 'x'

break

cerrar\_ventana()

captura.release()

guardarcsv()

print("Programa finalizado correctamente")

#### Historia de Usuario 3.1 — Exportar Datos a CSV

Archivo: Modelo/registro\_csv.py

import pandas as pd

from datetime import datetime

import os

# Lista global para ir acumulando los registros durante la ejecución

registros = []

# Función para registrar un incidente (se llama solo cuando hay alarma)

def registrarcsv(ear, mar, pitch, yaw, roll, nombre\_frame):

ahora = datetime.now()

hora = ahora.strftime("%H:%M:%S")

registros.append([hora, ear, mar, pitch, yaw, roll, nombre\_frame]) # ← Agrega al registro global

# Función para guardar todo al final

def guardarcsv():

ahora = datetime.now()

timestamp = ahora.strftime("%Y-%m-%d\_%H%M%S")

nombre = f"registro\_somnolencia\_{timestamp}.csv"

directorio = r"C:\Users\USER\Documents\modelo\registros\CSV"

os.makedirs(directorio, exist\_ok=True)

ruta\_completa = os.path.join(directorio, nombre)

# Verificar si hay registros antes de guardar

if not registros:

print("No hubo incidentes registrados.")

return

# Crear DataFrame y guardar

df = pd.DataFrame(registros, columns=["Hora", "EAR", "MAR", "Pitch", "Yaw", "Roll", "Frame"])

df.to\_csv(ruta\_completa, index=False)

print("¡Se guardó el CSV en:", ruta\_completa, "!")

Archivo: Modelo/registro\_alertas.py

import cv2

import os

from datetime import datetime

def guardar\_frame(frame):

# Obtener timestamp

ahora = datetime.now()

timestamp = ahora.strftime("%Y-%m-%d\_%H%M%S")

nombre\_archivo = f"frame\_{timestamp}.png"

directorio= r"C:\Users\USER\Documents\modelo\registros\frames"

# Crear el directorio si no existe

os.makedirs(directorio, exist\_ok=True)

# Ruta completa

ruta\_completa = os.path.join(directorio, nombre\_archivo)

# Guardar el frame como imagen PNG

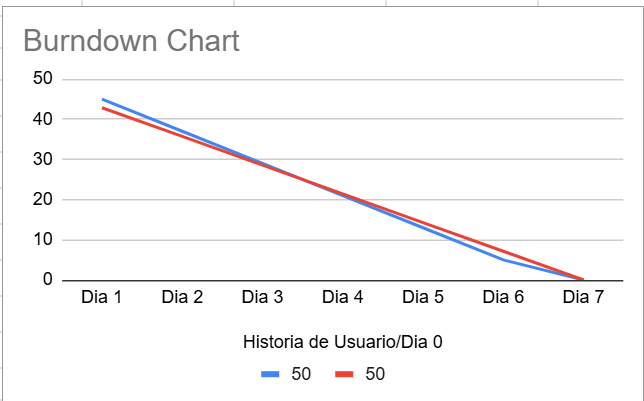
cv2.imwrite(ruta\_completa, frame)

return nombre\_archivo

### 

### Burndown Chart

| Backlog ID | Historia de Usuario | Tiempo Estimado | 16/06 | 17/06 | 18/06 | 19/06 | 20/06 | 21/06 | 22/06 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Dia1** | **Dia2** | **Dia3** | **Dia4** | **Dia5** | **Dia6** | **Dia 7** |
| 2.1 | Alerta Temprana de Somnolencia | 13 | 5 | 8 |  |  |  |  |  |
| 3.1 | Exportar Datos a CSV | 37 |  |  | 8 | 8 | 8 | 8 | 8 |
| Tiempo de trabajado | | 50 | 45 | 37 | 29 | 21 | 13 | 5 | 0 |



### Análisis del Sprint

* Se cumplió con todas las tareas estimadas.
* El progreso fue constante y sin bloqueos mayores.
* Las estimaciones fueron precisas y se logró un buen ritmo de trabajo.
* Los ajustes del Sprint fueron adecuados

### Conclusiones

El Sprint 2 concluyó exitosamente con todas las funcionalidades básicas implementadas. Se generaron las alarmas de interfaz y auditivas adecuadamente, se probaron las funcionalidades, y el código quedó preparado para evaluación.

El equipo demostró capacidad de entrega puntual y buena gestión de riesgos menores. Los objetivos del sprint se cumplieron completamente.

# Manual de Usuario Final

Bienvenido al modelo basado en IA para detección de somnolencia, un programa que busca mantener la seguridad vial ayudando a los conductores a mantenerse despiertos durante tramos o rutas de manejo muy largas. Este manual tiene como objetivo guiarte paso a paso en el uso del programa, explicando cada una de sus funcionalidades.

## Requisitos del Sistema

Para acceder y utilizar modelo basado en IA para detección de somnolencia correctamente, necesitas:

* Una laptop portátil.
* Una cámara web (externa o de la laptop).
* Parlantes (internos de la laptop).

## Acceso al Programa

### Instalación del programa

1. Descargar Python v. 3.9.x
2. Descargar librerías necesarias.
3. Descargar Visual studio code.

### Ejecución del programa

1. Ejecutar el módulo “analisis\_somonlencia.py”, es el módulo principal que inicia el programa. Luego el programa correrá como corresponde.

## Funcionalidades Principales

Más allá del inicio de ejecución del programa el usuario no necesita intervenir para que este cumpla alguna función.

### Capturar Flujo de Video

* Una vez iniciado, el programa empezará a capturar el video a través de la cámara reconocida.

### Monitoreo de Cierre Ocular

* Con el flujo de video capturado por la cámara, el programa puede identificar los puntos de referencia faciales en los ojos para así calcular la relación de aspecto del ojo (EAR), para determinar su amplitud.

### Detección de bostezos

* Con el flujo de video capturado por la cámara, el programa puede identificar los puntos de referencia faciales en la boca para así calcular la relación de aspecto de la boca (MAR), para determinar su amplitud.

### Medición de Inclinación de Cabeza

* Con el flujo de video capturado por la cámara, el programa puede identificar los puntos de referencia faciales para así calcular los ángulos de inclinación vertical, horizontal y rotacional de la cabeza para determinar si hay fatiga o somonlencia.

### Alertas de Somnolencia

* Después de que el programa captura todas las métricas relacionadas al estado del conductor, si es que el modelo predice somnolencia se generará una alarma visual y auditiva para despertar o prevenir al conductor.
* Se registrarán las métricas relacionadas a la alarma y la imágen.

## Buenas Prácticas de Uso

* Evitar usar lentes de sol o artículos que tapen la cara.
* Tratar de tener una iluminación adecuada para el óptimo funcionamiento de la captura de video.

## Preguntas Frecuentes (FAQ)

#### ¿Qué requisitos tiene el programa?

* Sistema operativo: Windows 10/11.
* Hardware: Cámara web (720p mínimo), 4GB RAM, GPU recomendada para mejor rendimiento.
* Software: Python 3.9.x (si se ejecuta desde código), o el instalador auto-contenido (.exe/.dmg).

#### ¿Por qué no detecta mi rostro?

* Asegúrate de:
  + Tener buena iluminación frontal.
  + Posicionar el rostro dentro del área marcada en la pantalla.
  + Actualizar los drivers de la cámara.
* Si persiste, revisa los logs (app.log) para ver errores de MediaPipe.

**¿La ventana de alarma se cierra automáticamente?**

La ventana se cierra solo cuando el sistema detecta que el conductor reaccionó o que ya no presenta somnolencia.

**¿Cómo cierro el programa?**

El programa dejará de correr si es que se terminó el trayecto o si ya no se usará, al presionar cualquier tecla del portátil.

## Contacto y Soporte

Para dudas o sugerencias sobre el sistema, puedes escribir al correo de soporte técnico: 71417350@continental.edu.pe

## Cierre

Gracias por utilizar el **modelo basado en IA para detección de somnolencia**. Esperamos que esta herramienta te ayude a mantenerte alerta durante jornadas de conducción largas o muy demandantes.

# Manual Técnico

Este manual técnico documenta los aspectos internos del desarrollo del modelo basado en IA para la detección de somnolencia en conductores. El documento está dirigido a desarrolladores, técnicos de soporte y cualquier persona interesada en comprender la estructura, tecnologías y funcionamiento interno del sistema.

## Tecnologías Utilizadas

| Componente | Tecnología o Herramienta |
| --- | --- |
| Interfaz simulada | Python - Tkinter |
| Lógica del sistema | Python (programación estructurada y modular) |
| Modelo de IA | XGBoost (clasificación binaria) |
| Dataset | CSV (con métricas faciales: EAR, MAR, pitch, yaw, roll) |
| Detección facial | MediaPipe (Face Mesh de Google) |
| Preprocesamiento de video | OpenCV |
| Captura en tiempo real | OpenCV + threading (multihilo en Python) |
| Alertas visuales | Tkinter (ventana emergente con mensaje) |
| Alertas auditivas | winsound (pitido de alerta en Windows) |

## Estructura del Proyecto

/Modelo

│

├── captura\_video.py # Módulo para capturar frames desde la cámara en tiempo real

├── deteccion\_facial.py # Procesamiento de landmarks faciales con MediaPipe

├── analisis\_somnolencia.py # Script principal: predicción y activación de alertas

├── gestor\_alertas.py # Interfaz de alerta (ventana + buzzer)

├── modelo\_xgboost.pkl # Modelo entrenado (XGBoost serializado con pickle)

├── registro\_csv.py

├── registro\_alertas.py

├── data/

│ └── dataset\_somnolencia.csv # Dataset con métricas faciales y etiquetas

├── Registros

└──registro.csv

└── incidentes\_frames.png/jpg

## Arquitectura General

Arquitectura en capas:

* Capa de Presentación: Python - Tkinter (interfaz gráfica emergente para alerta visual)
* Capa de Lógica de Negocio: Python - OpenCV, MediaPipe, XGBoost. Se encarga del procesamiento de video, extracción de características, predicción de somnolencia y control de alertas.
* Capa de Acceso a Datos: Python - Lectura y escritura de archivos .csv para análisis y almacenamiento de métricas.
* Capa de Persistencia: Python - Almacenamiento local del modelo entrenado (.pkl), datasets .csv y capturas (.jpg).

## Descripción de Componentes

### Interfaz de Usuario (Frontend)

Lenguaje: Python Librería: Tkinter Estilo visual: Interfaz básica con colores de alerta (fondo rojo, texto blanco). Interacción: Se activa automáticamente al detectar somnolencia y muestra un mensaje visual junto con un sonido (buzzer). Componentes clave:

* Ventana de Alerta: Se muestra solo cuando se detecta somnolencia mantenida.
* Sonido de Buzzer: Se activa en paralelo a la ventana, con un tono intermitente.

### Lógica de Negocio (Backend)

Lenguaje: Python Frameworks / Bibliotecas:

* OpenCV: Captura y procesamiento de video.
* MediaPipe: Extracción de métricas faciales (landmarks).
* XGBoost: Modelo de detección de somnolencia entrenado previamente.
* threading, keyboard, time: Control de ejecución y concurrencia.

Flujo principal:

* Captura de video y extracción de métricas (EAR, MAR, Pitch, Yaw, Roll).
* Predicción usando un modelo XGBoost entrenado con dataset externo.
* Activación o cierre de alerta según detección sostenida de somnolencia.

### Persistencia y Almacenamiento de Datos

Formato: CSV Ubicación: Local (en la carpeta del proyecto) Datos almacenados:

* Dataset de entrenamiento (frames y métricas).
* Registros de eventos de alerta (timestamp y condición).
* Modelo entrenado (formato .pkl generado con pickle).

## Instalación y Ejecución

### Requisitos

* Python 3.9+
* Bibliotecas necesarias: opencv-python, mediapipe, xgboost, tkinter, keyboard
* Sistema operativo: Windows 10+ (uso de winsound)

### Instalación

* Clonar o copiar el repositorio del proyecto.
* Instalar dependencias con pip:
* pip install opencv-python mediapipe xgboost keyboard

## Ejecución

* Ejecutar el script principal: python analisis\_somnolencia.py
* Presionar 'x' para detener el sistema.

## Seguridad

* Acceso Local Restringido: Los archivos del sistema (CSV, modelo, imágenes) solo pueden ser accedidos por la aplicación. Se recomienda configurar permisos de solo lectura para otros usuarios.
* Validación de Datos: Se inspeccionan los registros CSV para evitar entradas maliciosas o caracteres inválidos.
* Seguridad Física (Cámara): La cámara transmite video en tiempo real, sin almacenar grabaciones completas, previniendo vigilancia indebida.
* Logging Seguro: Los logs generados solo contienen información técnica como timestamps y estados de alerta. No se almacenan datos sensibles ni personales.

## Buenas Prácticas

* Modularización del código: Separación clara entre módulos de captura, análisis, alertas y lógica principal.
* Código legible y comentado: Cada módulo y función cuenta con comentarios que explican su propósito.
* Validación previa: Se verifica la existencia y el formato correcto de los datos antes de procesarlos.
* Uso de hilos (threads): Para mantener la interfaz responsiva y la ejecución paralela entre captura, predicción y alerta.
* Control de errores: Se encapsula el bucle principal en bloques try/except para evitar caídas inesperadas.
* Separación de responsabilidades: La lógica de detección, alerta y captura están implementadas en archivos distintos según su rol.

## Futuras Mejoras

* Generación de reportes con métricas de somnolencia (para ser analizados).
* Uso de cámara infrarroja para mejor detección facial en condiciones de baja luz.
* Sistema embebido para una fácil aplicación en escenarios de conducción.

## Contacto

Para soporte técnico o colaboración: 71417350@continental.edu.pe

# Manual de Instalación y Configuración

Este documento proporciona las instrucciones detalladas para la instalación, configuración y despliegue del modelo de detección de somnolencia en un entorno de desarrollo local. Se dirige a administradores del sistema, personal técnico o desarrolladores que deban implementar la solución.

## Requisitos del Sistema

### Requisitos de Software

* Python 3.9.x o superiores (compatibles con mediapipe)
* pip (administrador de paquetes de Python)
* Git
* Visual Studio Code (o algún editor de texto alternativo compatible con python e IA)
* Windows 10+

### Requisitos de Hardware (mínimos)

* CPU: 2 núcleos
* RAM: 4 GB
* Almacenamiento: 1GB libre
* Cámara web funcional

## Estructura del Proyecto

/Modelo

├── captura\_video.py

├── deteccion\_facial.py

├── gestor\_alertas.py

├── analisis\_somnolencia.py

├── modelo\_xgboost.pkl

├── registro\_csv.py

├── registro\_alertas.py

├── Registros

└──registro.csv

└── incidentes\_frames.png/jpg

## Instalación del Entorno de Desarrollo

### Extracción de Carpeta

Se necesita extraer la carpeta con todos los archivos.

### Instalación de Dependencias

pip install opencv-python mediapipe xgboost keyboard

### Verificación de Recursos

* Asegúrese de contar con una cámara web funcional.
* Verifique que el sistema tenga permisos para acceder a ella.

## Despliegue en Producción

El sistema está diseñado para ejecución local en equipos individuales

### Recomendaciones

* Usar equipos con acceso restringido.
* Evitar ejecución en servidores compartidos.

### Configuración sugerida

* Ejecutar el sistema como servicio de Windows para uso constante.
* Redirigir logs a un archivo con protección de escritura.

## Configuración Adicional

### Seguridad

* Limitar el acceso a los archivos CSV y modelos (.pkl) usando permisos del sistema operativo.

### Logs

* El sistema genera eventos en registros que pueden ser usados para auditoría técnica.
* Evite modificar manualmente este archivo.

## Verificación de Instalación

Verifique que:

* La cámara transmite correctamente.
* Se extraen las métricas faciales.
* Se activa la alerta (ventana y sonido) ante somnolencia mantenida.

## Contacto Técnico

Para asistencia técnica: soporte@taskmanager.app

## Anexos

* Código fuente comentado.
* Dataset de entrenamiento utilizado (.csv)

# Manual de Administrador de Sistemas

Este manual está destinado a los administradores del sistema de detección de somnolencia basado en inteligencia artificial. Su propósito es describir las funciones administrativas disponibles, incluyendo la gestión de registros, respaldo de datos, monitoreo del sistema, y consideraciones de seguridad para el entorno local en el que se ejecuta la aplicación.

## Acceso al Panel de Administración

Nota: En esta versión, el sistema no cuenta con una interfaz visual dedicada para administradores. Toda la gestión se realiza mediante acceso a archivos locales (CSV, modelos, capturas de eventos) y ejecución de scripts en Python.

## Gestión de Registros

### Visualización de Registros

Los eventos de detección se almacenan en el archivo registros.csv. Este archivo puede abrirse con cualquier editor de texto o programa de hoja de cálculo.

### Limpieza de Registros

Para limpiar el historial se puede eliminar directamente los registros relacionados a las fechas que se quieren eliminar.

### Almacenamiento de Evidencia Visual (frames)

Los frames de eventos se guardan en la carpeta Registros/ como archivos .jpg o .png. Pueden ser revisados manualmente o eliminados periódicamente si el almacenamiento es un problema.

## Mantenimiento del Modelo y Dataset

### Actualización del modelo

Los archivos del modelo completo se actualizarán desde github, solo es necesario descargar las actualizaciones.

## Respaldo y Recuperación de Base de Datos

### Respaldo

Copiar manualmente los archivos críticos y almacenarlos en un entorno seguro.

cp registros.csv backup/registros\_backup.csv

cp modelo\_xgboost.pkl backup/modelo\_backup.pkl

cp -r Registros/ backup/frames\_backup/

### Recuperación Para restaurar

Copiar desde la carpeta backup a a la ubicación original:

cp backup/registros\_backup.csv registros.csv

## Monitoreo y Logs

### Supervisión de Funcionamiento

El archivo registros.csv refleja el comportamiento del sistema. La ausencia de registros por largos períodos puede indicar que el sistema no está funcionando correctamente.

### Verificación de Componentes

* Confirmar que la cámara esté funcionando y accesible.
* Verificar que los módulos captura\_video.py y analisis\_somnolencia.py estén en ejecución.

## Seguridad Administrativa

* Proteger el archivo modelo\_xgboost.pkl contra escritura no autorizada.
* Limitar el acceso a los archivos CSV y a la carpeta Registros/ mediante permisos del sistema operativo.
* Asegurar que el sistema se ejecute en un entorno local cerrado, sin conexión a internet si no es necesario.

## Automatización Recomendada

* Crear una tarea programada para respaldar registros.csv diariamente.
* Establecer scripts para limpieza mensual de capturas visuales.

## Contacto para Escalamiento

En caso de incidencias técnicas graves, contactar con: 71417350@continental.edu.pe

## Cambios Previstos para el Rol de Administrador

* Panel visual en Tkinter para acceder a los registros.
* Gráficos automáticos de tendencias de somnolencia.
* Opciones para reentrenar el modelo desde la interfaz.

El modelo es un sistema en crecimiento. Este manual será actualizado conforme se integren nuevas funciones administrativas.

# Manual de Mantenimiento

Este manual describe los procedimientos de mantenimiento preventivo y correctivo del sistema de detección de somnolencia basado en inteligencia artificial. Su objetivo es asegurar el funcionamiento continuo del sistema, mantener un rendimiento óptimo y preservar la seguridad de los datos generados. Está dirigido al equipo técnico responsable del soporte y operación local del sistema.

## Objetivos del Mantenimiento

* Prevenir fallos antes de que ocurran (mantenimiento preventivo).
* Corregir errores y fallas del sistema en tiempo real (mantenimiento correctivo).
* Mantener la integridad de los registros generados y asegurar la operatividad de la detección en tiempo real.

## Mantenimiento Preventivo

### Tareas Semanales

* Verificación del archivo registros.csv para asegurar formato correcto y sin datos corruptos.
* Revisión del estado de la cámara (disponibilidad y acceso desde el sistema).
* Limpieza de registros antiguos si el volumen de datos lo requiere.
* Prueba funcional del sistema (ejecución de analisis\_somnolencia.py y confirmación de alerta visual/sonora).

### Tareas Mensuales

* Revisión de espacio en disco disponible (para evitar problemas de almacenamiento).
* Actualización de dependencias del sistema:

pip list --outdated

pip install --upgrade <paquete>

* Revalidar que el modelo modelo\_xgboost.pkl sigue funcionando correctamente con nuevos registros.

### Tareas Trimestrales

* Evaluación del rendimiento del sistema (tiempos de respuesta de la alerta).
* Reentrenamiento del modelo (opcional, si se cuenta con nuevos datos etiquetados).
* Auditoría de seguridad básica (acceso a archivos, permisos del sistema operativo).

## Mantenimiento Correctivo

### Gestión de Errores Comunes

* No se abre ventana de alerta: Revisar gestor\_alertas.py, asegurarse que no haya hilos bloqueados o errores en la GUI.
* No se detecta la cámara: Confirmar que esté conectada y que OpenCV la esté reconociendo.
* No se generan registros: Verificar el módulo registro\_csv.py y los permisos de escritura.

### Procedimiento de Corrección

1. Identificar el error mediante los mensajes de consola o revisión de archivos.
2. Replicar el fallo en entorno controlado.
3. Corregir el código, configuración o dependencias.
4. Probar nuevamente el sistema.
5. Documentar el incidente y la solución aplicada.

## Respaldo de Información

### Respaldo Manual

Realizar copia de seguridad del archivo registros.csv y de las imágenes de eventos (si se guardan):

cp registros.csv backups/registros\_$(date +%Y%m%d).csv

### Respaldo Programado

Con el programador de tareas de Windows:

@echo off

set FECHA=%date:~10,4%%date:~4,2%%date:~7,2%

copy registros.csv backups\registros\_%FECHA%.csv

## Actualización del Sistema

### Dependencias

pip install -r requirements.txt --upgrade

### Modelo IA

Para actualizar el modelo modelo\_xgboost.pkl, asegúrese de entrenarlo previamente y reemplazar el archivo con la nueva versión.

## Documentación de Cambios

Registrar cada cambio importante en el archivo CHANGELOG.md bajo formato:

## [1.1.0] - 2025-06-26

### Agregado

- Validación de registros con caracteres especiales.

### Corregido

- Error al cerrar ventana de alerta desde GUI.

## Herramientas Recomendadas

* Python 3.9+
* Git para control de versiones
* Visual Studio Code o PyCharm para edición de código
* Explorador de archivos con permisos avanzados (para respaldos y seguridad)

## Contacto de Soporte

Para reportar errores graves o caídas del sistema: 71417350@continenta.edu.pe

## Conclusión

Este manual proporciona un conjunto de prácticas esenciales para mantener el sistema de detección de somnolencia operativo, seguro y actualizado. Su aplicación periódica es fundamental para la sostenibilidad del proyecto.

# Manual de Pruebas

Este manual describe el proceso de verificación y validación del sistema de detección de somnolencia para conductores. Incluye los tipos de pruebas realizadas, casos de prueba definidos, criterios de aceptación y herramientas utilizadas. Está dirigido al equipo técnico encargado de asegurar el correcto funcionamiento del sistema antes de su despliegue final.

## Objetivos

* Verificar que cada componente del sistema funcione de forma adecuada.
* Validar el comportamiento del sistema ante condiciones reales de uso.
* Detectar errores o comportamientos inesperados antes del despliegue.
* Garantizar que el sistema sea confiable, usable y seguro.

## Tipos de Pruebas Aplicadas

### Pruebas Unitarias

* Validan funciones individuales de cada módulo del sistema.

### Pruebas de Integración

* Verificación de comunicación entre los módulos: captura de video, análisis facial, predicción del modelo y alerta visual/sonora.

### Pruebas Funcionales

* Validación de la lógica de detección de somnolencia.
* Activación de alerta cuando se detecta somnolencia mantenida.
* Desactivación de alerta cuando el estado vuelve a normal.

### Pruebas Manuales

* Pruebas realizadas por el usuario simulando diferentes condiciones faciales para comprobar el comportamiento general del sistema.

## Ambiente de Pruebas

| Componente | Versión / Requisito |
| --- | --- |
| Sistema Operativo | Windows 10 / 11 |
| Python | 3.9+ |
| Bibliotecas | OpenCV, MediaPipe, XGBoost, keyboard |
| Cámara Web | Cámara HD integrada o USB |
| Hardware | Intel i7 / 12 GB RAM |

## Casos de Prueba

### Activación de Alerta

| ID | Descripción | Entrada simulada | Resultado Esperado | Estado |
| --- | --- | --- | --- | --- |
| TC01 | Activar alerta al detectar somnolencia mantenida | EAR bajo, mirada hacia abajo, MAR alto (simulado 10 veces) | Se muestra ventana de alerta y buzzer activo | Aceptado |

### Desactivación de Alerta

| ID | Descripción | Entrada simulada | Resultado Esperado | Estado |
| --- | --- | --- | --- | --- |
| TC02 | Cerrar alerta al cesar la somnolencia | EAR y MAR normales durante 3 segundos | La ventana se cierra, el sonido se detiene | Aceptado |

### Corte de video

| ID | Descripción | Entrada simulada | Resultado Esperado | Estado |
| --- | --- | --- | --- | --- |
| TC03 | No hay rostro en el video | Usuario se retira del encuadre | El sistema omite el frame, sin error | Aceptado |

### Corte de video

| ID | Descripción | Entrada | Resultado Esperado | Estado |
| --- | --- | --- | --- | --- |
| TC04 | Cierre manual del sistema | Presionar la tecla ‘x’ | El sistema se detiene y cierra limpiamente | Aceptado |

## Resultados

Todas las pruebas fueron ejecutadas exitosamente en entorno local. El sistema reaccionó de forma correcta ante condiciones normales y de somnolencia en pruebas manuales.

## Criterios de Aceptación

* El sistema muestra una alerta visual y sonora al detectar somnolencia persistente.
* El sistema desactiva la alerta al recuperar el estado de atención.
* El sistema tolera la pérdida de frames sin bloquearse.
* El sistema se cierra de manera controlada al presionar la tecla designada.

## Herramientas Utilizadas

* **Cámara Web**: Para simulación de condiciones reales.
* **Python (print/logs)**: Verificación de métricas y predicciones en consola.  
  **Keyboard**: Para simulación de evento de salida.
* **Observación manual**: Validación visual de comportamiento de GUI y sonido.

## Recomendaciones Futuras

* Automatizar pruebas de video con secuencias simuladas.
* Implementar pruebas de estrés en hardware de menor capacidad.
* Incluir métricas de rendimiento (uso de CPU/RAM) en pruebas extendidas.
* Considerar un módulo de autodiagnóstico para la cámara y el modelo.

## Contacto QA

Para informes de errores o coordinación de pruebas: [71417350@continental.edu.pe](mailto:71417350@continental.edu.pe)

# Registro de Mantenimiento y Plan de Mejoras Futuras

Este documento consolida el historial de mantenimiento del sistema de detección de somnolencia basado en inteligencia artificial y presenta un plan estructurado de mejoras futuras. Su propósito es garantizar la sostenibilidad del sistema, documentar acciones técnicas relevantes y trazar la hoja de ruta para su evolución tecnológica y funcional.

## Registro de Mantenimiento

### Historial de Mantenimientos Ejecutados

| Fecha | Tipo | Descripción | Responsable |
| --- | --- | --- | --- |
| 2025-06-15 | Preventivo | Limpieza de archivos CSV obsoletos y revisión de espacio de almacenamiento | Desarrollador |
| 2025-06-17 | Correctivo | Ajuste en la lógica del contador de somnolencia (detección por frames) | Desarrollador |
| 2025-06-19 | Preventivo | Validación de integridad de archivos de registro y modelo entrenado | QA / Técnico |
| 2025-06-21 | Correctivo | Corrección de error en el módulo de alerta (buzzer no se detenía) | Backend Developer |
| 2025-06-25 | Preventivo | Revisión del rendimiento general y pruebas de estabilidad | QA / DevOps |

## Diagnóstico General

Tras la primera iteración del sistema, se han identificado fortalezas y áreas de oportunidad:

### Fortalezas

* Arquitectura modular en capas (presentación, lógica, persistencia).
* Integración efectiva de modelo XGBoost para predicción.
* Interfaz visual simple y clara en caso de detección de somnolencia.
* Código robusto con manejo de errores y validaciones básicas.

### Oportunidades de Mejora

* Falta de capacidad para operar en entornos con poca luz.
* Sistema no portable aún a hardware embebido.
* No cuenta con una interfaz administrativa para monitoreo remoto o multiusuario.
* Registro de eventos básico sin visualización gráfica o remota.

## Plan de Mejoras Futuras

### Funcionalidades Prioritarias

| Mejora | Descripción | Prioridad | Versión Estimada |
| --- | --- | --- | --- |
| Visión nocturna con cámara IR | Integrar cámaras infrarrojas para mejorar detección en condiciones de baja luz | Alta | v1.1 |
| Sistema embebido | Adaptar el sistema a hardware embebido (ej. Raspberry Pi) para mayor portabilidad | Alta | v1.2 |
| Panel de administración remota | Crear interfaz web para monitoreo de múltiples conductores y alertas remotas | Alta | v1.3 |
| Notificaciones remotas | Enviar alertas al celular o dashboard cuando se detecte somnolencia | Media | v1.4 |
| Visualización de registros históricos | Mostrar gráficos y métricas desde los archivos CSV o logs | Media | v1.4 |

### Acciones de Mantenimiento Programado

* Revisión quincenal de los registros CSV para evitar acumulación innecesaria.
* Verificación mensual del estado del modelo (.pkl) y su capacidad predictiva.
* Validación de compatibilidad con futuras versiones de bibliotecas (OpenCV, MediaPipe, XGBoost).
* Limpieza y respaldo de frames antiguos de entrenamiento cada 2 meses (si se usa).
* Pruebas de rendimiento y estabilidad cada 3 meses.

## Seguimiento y Actualización

* El presente documento será revisado y actualizado tras cada sprint o versión liberada.

## Contacto Responsable

71417350@continental.edu.pe

## Conclusión

El mantenimiento constante y la planificación estructurada de mejoras permitirán que el sistema de detección de somnolencia evolucione hacia una solución más precisa, portable y adaptable a distintos contextos. Este documento sirve como guía para el desarrollo futuro del proyecto.