|  |  |  |  |
| --- | --- | --- | --- |
| ***Interesados en el proyecto*** | | | |
| ***Interesado*** | ***Nombre*** | ***Función*** | ***Teléfono*** |
| Director del proyecto | Yersson Calderon Romero | Supervisar el proyecto y tomar decisiones estratégicas. | 925271857 |
| Gerente del proyecto | Alejandro Humberto Guevara Valdivia | Gestionar las actividades diarias y coordinar al equipo. | 984582724 |
| Coordinador de cronograma y presupuesto | Edwards Andree Ochoa Cutipa | Planificar y controlar el cronograma y presupuesto del proyecto. | 992133112 |
| Docente del curso | Americo Estrada Sanchez | Evaluador del proyecto | - |

**Repositorio:** [**https://github.com/Taller-de-proyectos-I/IDEA-PROYECTO.git**](https://github.com/Taller-de-proyectos-I/IDEA-PROYECTO.git)

[**1. PLANTEAMIENTO DE PROPUESTAS 2**](#_uezntphoj2en)

[1.1. propuesta 1 2](#_z36acm61mvof)

[1.2. propuesta 2 2](#_hm735v2jta3d)

[1.3. propuesta 3 2](#_69qujv6ny6ts)

[1.4. propuesta 4 2](#_8do6rvxrc82o)

[**2. COMPARATIVA ENTRE PROPUESTAS PARA SU ANÁLISIS 2**](#_8hs7arnhc029)

[**3. EVALUACION Y ANALISIS GRUPAL 2**](#_ebxn6hdxld74)

[**4. JUSTIFICACIÓN 2**](#_j4okbkvz8ots)

[**5. DIAGRAMA DE ISHIKAWA BASADO EN ERRORES 2**](#_319ngwgnnwa8)

[**6. DESARROLLO DE LA METODOLOGÍA ÁGIL 2**](#_kn255f6cucu2)

[6.1. Historias de usuario 2](#_xh361z95hw)

[6.2. Estimación de tiempos 2](#_pq2alevlwoho)

[6.3. Estimación de épicas 2](#_sga0vxueo5sj)

[6.4. Product backlog según prioridades y estimaciones 2](#_c3np63o0t6rr)

[6.5. Cronograma: diagrama de gantt 2](#_f2x7ngu3h60u)

# PLANTEAMIENTO DE PROPUESTAS

## Propuesta 1

**Título de la Propuesta:**

Sistema Inteligente de Reconocimiento de Vehículos y Alerta Automática para la Gestión de Seguridad Vial en la Vía de Acceso a la Universidad Continental – Saylla, Cusco 2025

**Propuesto por:** EdwardsAndree Ochoa Cutipa

**Problema:**

En la vía de acceso frente a la Universidad Continental, en Saylla (Cusco), se presentan con frecuencia congestionamientos, incidentes viales y fugas de vehículos que dificultan la respuesta inmediata de las autoridades, como el serenazgo o la Policía Nacional. Esa situación pone en riesgo tanto a peatones como a conductores, especialmente considerando que la infraestructura y señalización de la zona aún son novedosas para muchos.

Según el Observatorio Nacional de Seguridad Vial, en el año 2023 se registraron en todo el país 87 083 siniestros de tránsito, con 58 000 personas heridas y 3 316 fallecidas.

Además, Cusco se posicionó como la segunda región con más siniestros fatales, con 182 casos, solo detrás de Lima (370).

**Descripción de la solución Propuesta:**

Se propone desarrollar un sistema inteligente de monitoreo vehicular, instalado en la vía de acceso a la Universidad Continental – Saylla, que utilice cámaras y visión por computadora (CNN, YOLO) para reconocer en tiempo real las placas, tipo y características de los vehículos. Este sistema utilizará:

1. Cámaras inteligentes junto con modelos de visión artificial (por ejemplo, YOLO para detección de vehículos).
2. Algoritmos de reconocimiento automático de placas (ALPR) y clasificación de tipo de vehículo utilizando CNN u OCR como EasyOCR.
3. Módulo de alerta automática, que enviará información clave (placa, tipo, hora, ubicación, una fotografía del vehículo) al serenazgo o la PNP en caso de detectar:

- Vehículos involucrados en accidentes que intentan fugarse.

- Maniobras sospechosas o incumplimiento de normas viales.

1. Una base de datos centralizada para almacenar registros históricos y permitir análisis posteriores.
2. Canales de notificación en tiempo real (como una aplicación web) para que las autoridades puedan actuar rápidamente.

**Requerimientos del producto de alto nivel:**

1. Detectar y reconocer vehículos en tiempo real mediante cámaras instaladas en la vía de acceso a la universidad.
2. Identificar placas vehiculares automáticamente usando algoritmos OCR e inteligencia artificial.
3. Clasificar el tipo de vehículo (auto, camioneta, moto, bus) a partir de las imágenes captadas.
4. Registrar evidencia (imagen, hora, ubicación, tipo de vehículo y placa) en una base de datos centralizada.
5. Generar alertas automáticas en caso de fuga tras un accidente o comportamiento sospechoso.
6. Enviar notificaciones en tiempo real a serenazgo o PNP a través de un SMS o aplicación web.
7. Acceder a un panel de monitoreo web donde las autoridades puedan visualizar los incidentes y el historial.
8. Almacenar reportes históricos para análisis de tráfico, zonas de riesgo y estadísticas.
9. Soportar integración con sistemas de seguridad existentes (como cámaras municipales o bases de datos policiales).
10. Funcionar de manera continua (24/7) con capacidad de autodiagnóstico básico (alerta si la cámara falla o no hay conexión).

**Requerimientos Tecnológicos:**

**Hardware:**

1. Laptop con las siguientes características mínimas:
   * Procesador: Intel Core i7 (10ma gen o superior).
   * Memoria RAM: 16 GB min.
   * Tarjeta Gráfica (GPU): NVIDIA RTX 3070.
2. Almacenamiento:
   * SSD de 512 GB.
   * HDD/SSD adicional de 1 TB
3. Cámara Web / Cámara IP (mín. 1080p, 30 fps)
4. Router y conexión a internet estable
5. UPS (Sistema de alimentación ininterrumpida) – Para evitar pérdidas de datos en caso de cortes eléctricos.

**Software:**

1. Lenguajes de Programación:
2. Python 3.10+
3. Frameworks de Visión Artificial e IA:
4. OpenCV (procesamiento de imágenes).
5. TensorFlow 2.x / PyTorch (entrenamiento y ejecución de CNN, ej. YOLOv8).
6. EasyOCR (para lectura de placas).
7. Frameworks Web / Backend:
8. Flask (para API REST y microservicios).
9. Base de Datos:
10. PostgreSQL o MySQL (para gestión de registros históricos).
11. Frontend / Dashboard:
12. HTML5, CSS3, JavaScript.
13. Notificaciones y Comunicación:
14. API de Telegram (bot para alertas).

**Estado del Arte: Análisis de la competencia:**

|  |
| --- |
| **1. Sistema de Reconocimiento Automático de Placas Vehiculares (ANPR) – ParkPow**   * Nombre: Deep License Plate Recognition * Versión / Año: GitHub Open Source Project (2021, actualizado en 2023) * Funcionalidades:   + Detecta vehículos y extrae automáticamente la matrícula de imágenes o video en tiempo real.   + Compatible con múltiples países y formatos de placas. Permite integración mediante API. * Limitaciones:   + Enfocado principalmente en la lectura de placas.   + No incluye sistema de alertas automáticas ni clasificación avanzada del vehículo (tipo, color). * Ventaja de nuestra propuesta:   + Integramos no solo la detección de placas, sino también la clasificación de tipo de vehículo.   + Incorporamos un módulo de alerta inmediata a serenazgo/PNP, adaptado al contexto local (Saylla – Cusco).   + Implementamos un panel web con historial y notificaciones en tiempo real, no solo extracción de datos.   Referencia: [GitHub – parkpow/deep-license-plate-recognition](https://github.com/parkpow/deep-license-plate-recognition?utm_source=chatgpt.com)  **2. An efficient and layout-independent automatic license plate recognition system based on the YOLO detector” — Laroca et al., 2021 (IET Intelligent Transport Systems)**  Año / versión: 2021, enfoque layout-independent (independiente del diseño de la placa).  Funcionalidades: unifica detección de LP + clasificación del layout y aplica reglas de post-proceso para mejorar OCR; evalúa 8 datasets (China, Europa, Brasil, etc.) con 96.9% de tasa promedio y > 70 FPS en GPU alta, manteniendo tiempo real con múltiples vehículos.  Dónde nuestra propuesta mejora:   * Añadimos clasificación de tipo de vehículo (auto/camión/bus/moto) y atributos (color/marca opcional) para correlacionar con infracciones o eventos. * Entrenamiento fino con datos locales (dominios .pe) y evaluación A/B en la intersección frente a Continental para reducir falsos positivos por obras/señalización nueva.   Referencia: [raysonlaroca.github.io](https://raysonlaroca.github.io/papers/laroca2021efficient-published.pdf) |

## Propuesta 2

**Título de la Propuesta:**

Implementación de sistema de visión por computadora con CNN para diagnóstico de enfermedades en hojas de cultivos andinos en comunidades de Cusco.

**Propuesto por: Yersson Calderon Romero**

**Problema:**

En comunidades agrícolas del Cusco, los agricultores muchas veces no identifican a tiempo las enfermedades o plagas que afectan las hojas de sus cultivos. Esto implica pérdidas económicas, menor productividad y un riesgo latente para la seguridad alimentaria.

La FAO (Organización de las Naciones Unidas para la Alimentación y la Agricultura) estima que hasta el 40 % de los cultivos alimentarios a nivel global se pierden cada año por enfermedades y plagas de las plantas ([FAO, 2021](https://agn.gt/fao-enfermedades-y-plagas-de-las-plantas-causan-la-perdida-del-40-de-los-cultivos/?utm_source=chatgpt.com)).

En el caso del Perú, el SENASA (Servicio Nacional de Sanidad Agraria) reporta que los agricultores de la sierra enfrentan con frecuencia la aparición de enfermedades como tizón tardío en papa, roya amarilla en trigo y manchas foliares en maíz, lo que afecta directamente a pequeños productores que no cuentan con sistemas modernos de monitoreo ([SENASA, 2020](https://www.senasa.gob.pe/senasacontigo/minagri-mantiene-el-control-de-plagas-y-enfermedades-en-la-agricultura/?utm_source=chatgpt.com)).

**Descripción de la solución Propuesta:**

Se propone implementar un sistema de visión por computadora con redes neuronales convolucionales (CNN) que permita diagnosticar de manera temprana enfermedades en hojas de cultivos clave como papa, maíz y tomate.  
  
El sistema permitirá a los agricultores:

* Capturar o subir una foto de la hoja.
* Analizar la imagen mediante un modelo CNN entrenado.
* Obtener un diagnóstico preliminar con precisión superior al 85 %.
* Recibir recomendaciones iniciales para el manejo de la enfermedad detectada.

El enfoque inicial será sobre cultivos andinos de alta relevancia económica, con la posibilidad de ampliar progresivamente a otros como uva, fresa o pimiento.

**Requerimientos del producto de alto nivel:**

* Capturar o cargar imágenes de hojas de cultivo.
* Detectar si la hoja está sana o enferma.
* Identificar el tipo de enfermedad en los cultivos priorizados (papa, maíz, tomate).
* Mostrar recomendaciones iniciales de manejo.
* Guardar historial de diagnósticos por usuario.
* Ofrecer una interfaz accesible en web.
* Permitir actualización del modelo CNN con nuevos datos.
* Generar reportes de enfermedades más frecuentes en cada zona.
* Incluir progresivamente nuevos cultivos según disponibilidad.
* Funcionar en comunidades rurales con baja conectividad (soporte offline).

**Requerimientos Tecnológicos:**

**Hardware:**

* Computadoras de arquitectura x86 o x64.
* GPU NVIDIA (mínimo 8GB VRAM, RTX 3060 superior)
* Memoria RAM ≥16GB.
* Almacenamiento SSD (mínimo 200GB).
* Smartphones
* Servidor en la nube

**Software:**

* Lenguajes de programación: Python (principal), JavaScript
* Frameworks de Deep Learning: TensorFlow/Keras, PyTorch.
* Librerías de procesamiento de imágenes: OpenCV, Pillow, Albumentations (data augmentation).
* Framework web: Flask o FastAPI para API de predicción.
* Control de versiones: GitHub.
* Entornos de ejecución: Google Colab, VS Code, Jupyter Notebook.

**Estado del Arte: Análisis de la competencia:**

|  |
| --- |
| 1. **PlantVillage App (Penn State University, 2019)**    * Detecta enfermedades en diferentes cultivos usando inteligencia artificial.    * Diferencia: nuestra propuesta está enfocada en cultivos andinos y pensada para funcionar en zonas rurales con baja conectividad.    * Fuente oficial: [https://plantvillage.psu.edu/?utm\_source](https://plantvillage.psu.edu/?utm_source=chatgpt.com) 2. **Plantix (App, 2020, Alemania/India)**    * Permite identificar más de 400 enfermedades y deficiencias en cultivos.    * Diferencia: requiere conexión a internet y no está adaptada al contexto rural del Perú.    * Fuente oficial: [https://plantix.net/en/?utm\_source](https://plantix.net/en/?utm_source=chatgpt.com) |

## Propuesta 3

**Título de la Propuesta:**

Aplicación de técnicas de Machine Learning para la detección de vulnerabilidades en redes informáticas de instituciones educativas, Arequipa, 2025.

**Propuesto por:** Alejandro Humberto Guevara Valdivia

**Problema:**

En la ciudad del Arequipa, las instituciones publicas y privadas dependen cada vez mas de sus redes informáticas para garantizar la continuidad de sus servicios y procesos críticos. Sin embargo, estas redes se encuentran constantemente expuestas a ciberataques, como:

* Escaneo de puertos para verificar o identificar servicios vulnerables.
* Ataques de denegación de servicio (DoS/DdoS) que interrumpen operaciones.
* Inyecciones de código y accesos no autorizados a sistemas sensibles.
* ARP spoofing y sniffing, que comprometen la confidencialidad de la información.

Diversas instituciones publicas y privadas han sido victima de ciberataques en los últimos años, lo que evidencia la vulnerabilidad de sus redes informáticas. Un ejemplo a demostrar, fue el fraude cibernético al gobierno regional de Arequipa, la UGEL Caylloma y la municipalidad de Yura, donde se registraron perdidas superiores a los 2 millones de soles debido a técnicas de phishing y suplantación de identidad, ​(Republica, 2023)​. Así mismo, la empresa SEAL sufrió un ataque informático que paralizo temporalmente sus servicios virtuales y de atención al cliente ​(Republica, 2023)​

**Descripción de la solución Propuesta:**

En este contexto, surge la necesidad de aplicar técnicas de machine learning, que permitan mejorar la detección de vulnerabilidades en redes informáticas, contribuyendo a la protección de los servicios y la continuidad se las operaciones.

Entonces la solución propuesta consiste en el diseño e implementación de un sistema inteligente basado en técnicas de Machine Learning, orientado a la detección de vulnerabilidades en redes informáticas de instituciones publicas y privadas en Arequipa. Es decir, el sistema se fundamenta en la utilización de modelos supervisados y no supervisados de aprendizaje automático, que será capaz de analizar grandes volúmenes de trafico de red en tiempo real, con el objetivo de identificar patrones anómalos que puedan asociarse a intentos de intrusión, explotación de vulnerabilidades o ataques cibernéticos.

El sistema incluirá las siguientes fases:

* Recolección de datos: Captura de tráfico de red real y generación de datasets mediante entornos de simulación, considerando tanto comportamientos normales como ataques conocidos (DoS/DDoS, escaneo de puertos, ARP spoofing, inyección de código, etc.)
* Procesamiento y Análisis de datos: Limpieza, normalización y extracción de características relevantes para facilitar el entrenamiento del modelo de Machine Learning.
* Entrenamiento de modelos de ML/IA: Uso de algoritmos como (Random Forest, Suport vector Machines, Redes Neuronales Artificiales y K- Means) para clasificar el tráfico entre normal y malicioso.
* Implementación de un sistema de detección: Desarrollo de una plataforma que integre el modelo entrenado, con capacidad para procesar tráfico en tiempo real y generar alertas de seguridad cuando se identifiquen posibles vulnerabilidades.
* Validación y pruebas: Evaluación de la precisión, sensibilidad y especificidad del sistema, comparando su desempeño frente a herramientas tradicionales de detección basadas en firmas y reglas.

**Requerimientos del producto de alto nivel:**

* **Captura de tráfico de red:**
* El sistema deberá ser capaz de recolectar tráfico de red en tiempo real desde diferentes dispositivos y segmentos de red.
* **Analizar paquetes y flujos para detectar anomalías**
* **Clasificar ataques (ej. DoS, port scanning, malware, instrusiones)**
* **Generar reportes automáticos de incidentes**
* **Debe enviar alertas (Notificaciones por correo electrónico)**
* **Almacenar registros históricos de tráfico y detecciones**
* **Ofrecer un panel web interactivo para la visualización de vulnerabilidades y métricas.**

**Requerimientos Tecnológicos:**

**Hardware:**

1. **Servidor o estación de trabajo**

* Procesador: Intel Core i7 (10ma gen o superior) o AMD Ryzen 7 equivalente.
* Memoria RAM: mínimo 16 GB (recomendado 32 GB para entrenamiento de modelos más grandes).
* Almacenamiento: mínimo 512 GB SSD (recomendado 1 TB para datasets y registros de tráfico).
* Tarjeta de Red: compatible con captura de paquetes en tiempo real (ej. Gigabit Ethernet).
* GPU (opcional pero recomendable): NVIDIA con soporte CUDA (ej. RTX 3060 o superior) para acelerar el entrenamiento de modelos de IA.

1. **Equipos de prueba en red**

* PCs/VMs configuradas como clientes y servidores.
* Switches y routers virtualizados (ej. GNS3, Cisco Packet Tracer) o físicos para pruebas controladas.

**Software:**

1. **Sistema Operativo**

* Linux (Ubuntu Server 22.04 LTS recomendado) para servidor de entrenamiento y despliegue.
* Windows 10/11 para estaciones de prueba y uso de herramientas de simulación.

1. **Entornos y librerías de IA / Machine Learning**

* Python 3.10+
* Librerías: TensorFlow, Keras, Scikit-learn, PyTorch, Pandas, NumPy, Matplotlib, Seaborn.

1. **Herramientas de captura y análisis de tráfico de red**

* Wireshark / Tshark.
* Tcpdump.
* Zeek (antes Bro IDS) para detección de intrusiones.
* Snort/Suricata para comparación con sistemas tradicionales.

1. **Simuladores y datasets de ciberseguridad**

* NSL-KDD, CICIDS2017 o UNSW-NB15 (datasets públicos de ataques de red).
* Cisco Packet Tracer o GNS3 para generar tráfico controlado en entornos simulados.

1. **Entornos de desarrollo y despliegue**

* Visual Studio Code / PyCharm.
* Docker para contenedorización del sistema.
* Git/GitHub para control de versiones.

1. **Base de datos y almacenamiento de registros**

* PostgreSQL o MySQL para almacenar registros de tráfico y alertas.
* Elasticsearch + Kibana para visualización avanzada de datos de red.

**Estado del Arte: Análisis de la competencia:**

1. **Perú: Análisis integral de IDS y algoritmos asociados (Universidad Nacional de Trujillo, 2023)**

* Estudia diferentes IDS y algoritmos de detección de intrusos, evaluando rendimiento en seguridad de la información, su limitación seria que no se implemento un sistema completo en un entorno real.
* Fuente oficial:<https://revistas.upt.edu.pe/ojs/index.php/ingenieria/article/view/840?utm_source=chatgpt.com>

1. **Perú: Rendimiento de IDS en redes SDN (PUCP, 2021)**

* Evalua el rendimiento de IDS (Snort, Suricata, Ml) en redes definidas por software, lo cual son escenarios modernos de redes virtualizadas, por ello es su limitación, es decir, se centra en SDN, no redes empresariales convencionales.
* Fuente oficial: <https://tesis.pucp.edu.pe/items/4fa47595-d598-4713-bf51-2359dd34914d>

1. **Argentina: IDS en redes industriales o detección de intrusiones en redes industriales (UNPL, CACIC, 2022)**

* Evalúa 9 algoritmos ML en tráfico Modbus, que es usado en redes industriales, lo cual su enfoque es su seguridad en infraestructuras criticas y no es aplicable directamente a redes corporativas comunes.
* Fuente oficial: <https://sedici.unlp.edu.ar/handle/10915/149402?utm_source=chatgpt.com>

1. **España: IDS con ML para trafico real (UPM, 2021)**

* Implementa un IDS usando Random Forest, entrenado con dataset UNSW-NB15, aplicado en entornos reales, lo cual lo limita porque depende de datasets internacionales, no locales
* Fuente oficial: <https://oa.upm.es/69370/?utm_source=chatgpt.com>

# 

# COMPARATIVA DE LAS PROPUESTAS

|  |  |  |  |
| --- | --- | --- | --- |
| **Aspecto** | Propuesta 1: “Sistema Inteligente de Reconocimiento de Vehículos y Alerta Automática para la Gestión de Seguridad Vial en la Vía de Acceso a la Universidad Continental – Saylla, Cusco 2025” | Propuesta 2 “Implementación de sistema de visión por computadora con CNN para diagnóstico de enfermedades en hojas de cultivos andinos en comunidades de Cusco.” | Propuesta 3: “Aplicación de técnicas de Machine Learning para la detección de vulnerabilidades en redes informáticas de instituciones educativas, Arequipa, 2025” |
| **Problema Identificado** | Congestionamientos, incidentes viales y fugas de vehículos en la vía de acceso a la Universidad Continental (Saylla), dificultan la respuesta rápida de autoridades. En Cusco hubo 182 siniestros fatales en 2023 (2da región más alta del país). | En comunidades agrícolas del Cusco, los agricultores no identifican a tiempo las enfermedades en las hojas de cultivos, ocasionando pérdidas económicas, menor productividad y riesgo a la seguridad alimentaria. | En la ciudad de Arequipa, muchas organizaciones (Universidades, entidades públicas) carecen de sistemas avanzados de detección de intrusiones y vulnerabilidades. Lo cual esto incrementa el riesgo operativo, financiero y reputacional, limitando la capacidad de respuesta y afectando la confianza en la seguridad digital local. |
| **Oportunidad** | Uso de visión por computadora y reconocimiento automático de placas para apoyar a serenazgo/PNP en seguridad vial, con alertas en tiempo real y almacenamiento de evidencia. | Aplicar visión por computadora para ofrecer a pequeños agricultores una herramienta tecnológica accesible, que permita diagnósticos rápidos y confiables, incluso en zonas rurales con baja conectividad. | Contribuir a la ciberseguridad en Arequipa, siendo pionero en aplicar IA en este campo a nivel regional y poder ofrecer un sistema adaptable tanto a locales como a instituciones académicas, fomentando una cultura de seguridad digital. |
| **Solución Propuesta** | |  | | --- | | Sistema de cámaras + CNN/YOLO + OCR (EasyOCR) para reconocer placas y vehículos, generar alertas automáticas y enviar notificaciones a autoridades. Incluye panel de monitoreo web y base de datos histórica. | | Un sistema basado en redes neuronales convolucionales (CNN) que analice imágenes de hojas (papa, maíz, tomate), determine si están sanas o enfermas, identifique la enfermedad y brinde recomendaciones iniciales con precisión ≥ 85%. | Desarrollo de un **IDS inteligente** basado en **Machine Learning** que:   * Capture tráfico de red (con Wireshark/Tcpdump/Zeek). * Clasifique comportamientos normales y anómalos mediante algoritmos de ML/DL (Random Forest, CNN, LSTM). * Genere alertas automáticas con un dashboardvisual. * Permita integración con sistemas existentes (ej. Snort/Suricata) para mejorar la detección. * Escalable y adaptable a diferentes organizaciones de Arequipa |
| **Requerimientos Funcionales** | - Detectar y reconocer vehículos en tiempo real.  - Identificar placas automáticamente.  - Clasificar tipo de vehículo (auto, moto, bus, camión).  - Registrar evidencia (foto, hora, ubicación, placa).  - Generar alertas automáticas en caso de fuga o sospecha.  - Enviar notificaciones en tiempo real (SMS/app web).  - Acceso a panel de monitoreo.  - Guardar reportes históricos.  - Integración con sistemas de seguridad existentes.  - Funcionar 24/7 con autodiagnóstico básico. | - Capturar o cargar imágenes de hojas.  - Detectar si está sana o enferma.  - Identificar tipo de enfermedad.  - Mostrar recomendaciones iniciales.  - Guardar historial de diagnósticos.  - Interfaz web accesible.  - Permitir actualización del modelo.  - Generar reportes por zona.  - Incluir nuevos cultivos.  - Funcionar offline. | - Captura de tráfico de red:  - Analizar paquetes y flujos para detectar anomalías  - Clasificar ataques (ej. DoS, port scanning, malware, intrusiones)  - Generar reportes automáticos de incidentes  - Debe enviar alertas (Notificaciones por correo electrónico)  - Almacenar registros históricos de tráfico y detecciones  - Ofrecer un panel web interactivo para la visualización de vulnerabilidades y métricas. |
| **Requerimientos Tecnológicos** | **Hardware:**  - Laptop i7 10ma gen+, RAM 16GB, GPU RTX 3070, SSD 512GB + 1TB extra.  - Cámaras IP (1080p, 30fps), router, UPS.  **Software:**  - Python 3.10+.  - OpenCV, TensorFlow/PyTorch, YOLOv8, EasyOCR.  - Flask (API REST).  - PostgreSQL/MySQL.  - HTML, CSS, JS (frontend).  - API Telegram, Twilio/SMTP (alertas). | - **Hardware:** PC x86/x64, GPU ≥ RTX 3060 (8GB VRAM), RAM ≥16GB, SSD ≥200GB, smartphones, servidor en la nube.  - **Software:** Python, JavaScript, TensorFlow/Keras, PyTorch, OpenCV, Pillow, Albumentations, Flask/FastAPI, GitHub, Colab, VS Code, Jupyter. | **Hardware:**  **Servidor o estación de trabajo**   * Procesador: Intel Core i7 (10ma gen o superior) o AMD Ryzen 7 equivalente. * Memoria RAM: mínimo 16 GB (recomendado 32 GB para entrenamiento de modelos más grandes). * Almacenamiento: mínimo 512 GB SSD (recomendado 1 TB para datasets y registros de tráfico). * Tarjeta de Red: compatible con captura de paquetes en tiempo real (ej. Gigabit Ethernet). * GPU (opcional pero recomendable): NVIDIA con soporte CUDA (ej. RTX 3060 o superior) para acelerar el entrenamiento de modelos de IA.   **Equipos de prueba en red**   * PCs/VMs configuradas como clientes y servidores. * Switches y routers virtualizados (ej. GNS3, Cisco Packet Tracer) o físicos para pruebas controladas.   **Software:**  **Sistema Operativo**   * Linux (Ubuntu Server 22.04 LTS recomendado) para servidor de entrenamiento y despliegue. * Windows 10/11 para estaciones de prueba y uso de herramientas de simulación.   **Entornos y librerías de IA / Machine Learning**   * Python 3.10+ * Librerías: TensorFlow, Keras, Scikit-learn, PyTorch, Pandas, NumPy, Matplotlib, Seaborn.   **Herramientas de captura y análisis de tráfico de red**   * Wireshark / Tshark. * Tcpdump. * Zeek (antes Bro IDS) para detección de intrusiones. * Snort/Suricata para comparación con sistemas tradicionales.   **Simuladores y datasets de ciberseguridad**   * NSL-KDD, CICIDS2017 o UNSW-NB15 (datasets públicos de ataques de red). * Cisco Packet Tracer o GNS3 para generar tráfico controlado en entornos simulados.   **Entornos de desarrollo y despliegue**   * Visual Studio Code / PyCharm. * Docker para contenedorización del sistema. * Git/GitHub para control de versiones.   **Base de datos y almacenamiento de registros**  PostgreSQL o MySQL para almacenar registros de tráfico y alertas.   * Elasticsearch + Kibana para visualización avanzada de datos de red. |
| **Plataforma Backend** | Flask para API REST y microservicios, integrando panel de monitoreo y notificaciones. | Flask para la API de predicción y gestión de diagnósticos. | Backend principal:  -Django REST Framework o FastAPI (Python)  -API REST para integrar el modulo de IA con el frontend y la base de datos.  Base de Datos:  -PostgreSQL para registros y Elasticsearch para búsquedas rápidas. |
| **Tecnologías de Mapa** | |  | | --- | | Integración con mapas de calor en panel web para mostrar zonas de mayor frecuencia de incidentes. (LeafletJS / Google Maps API). | | Uso de mapas interactivos (ej. LeafletJS o Google Maps API) para mostrar distribución geográfica de enfermedades detectadas en cada zona. | -Mapa de red interactivo con D3.js o Cytoscape.js (visualización de nodos, hosts y tráfico).  -Integración con Kibana dashboards para ver anomalías en tiempo real.  -Posibilidad de usar Grafana para métricas y correlación de eventos. |
| **Gamificación** | |  | | --- | | Puede implementarse como incentivos a conductores responsables (ej. puntos por cumplimiento de normas en campañas universitarias). | | No aplica de manera directa, pero podría añadirse un módulo opcional con insignias o logros para agricultores que registren diagnósticos frecuentes o adopten prácticas recomendadas. | Para aumentar el interés de usuarios (administradores de red, estudiantes en pruebas):   * Sistema de **logros o badges** (“Red segura 30 días sin ataques detectados”). * **Ranking de detecciones** por severidad o velocidad de respuesta. * **Simulaciones de ataques controlados** para entrenar a los administradores y medir tiempos de reacción. * Retroalimentación visual tipo “semáforo de seguridad” (verde/amarillo/rojo). |
| **Impacto Social** | * Incrementa la seguridad vial en la zona universitaria, protege a peatones y conductores, y fortalece la capacidad de respuesta de autoridades locales. | Mejora la productividad agrícola, reduce pérdidas económicas, contribuye a la seguridad alimentaria y democratiza el acceso a tecnologías de IA en comunidades rurales del Cusco. | -**Educativo**: capacita a futuros ingenieros de sistemas en el uso de IA aplicada a ciberseguridad.  -**Económico**: ayuda a pymes de Arequipa a reducir costos de ataques informáticos.  -**Social**: genera confianza en los servicios digitales locales, fortaleciendo la seguridad de la información.  -**Científico**: aporta un estudio pionero en la región sobre aplicación de IA en detección de vulnerabilidades en redes reales.  **-Regional**: contribuye a la visión de Arequipa como un polo de innovación en tecnología y seguridad informática en el sur del Perú. |
| **Competencia y Avances Similares** | - **Deep License Plate Recognition (ParkPow, 2021-2023):** reconocimiento de placas en tiempo real. Diferencia: nuestra propuesta integra alertas inmediatas y clasificación de vehículos.  - **Laroca et al. (2021, YOLO-based ANPR):** alta precisión en detección de placas, pero sin conexión a sistemas locales. Diferencia: adaptado al contexto de Cusco, con panel y notificaciones en tiempo real. | |  | | --- | | - **PlantVillage (2019):** Diagnóstico de cultivos con IA, pero requiere internet y no está enfocado en cultivos andinos.  - **Plantix (2020):** Detecta más de 400 enfermedades, pero no está adaptada al Perú ni funciona offline.  Diferencia: propuesta enfocada en cultivos andinos, soporte offline y adaptación local. | | **Perú: Análisis integral de IDS y algoritmos asociados (Universidad Nacional de Trujillo, 2023)**  -Estudia diferentes IDS y algoritmos de detección de intrusos, evaluando rendimiento en seguridad de la información, su limitación seria que no se implemento un sistema completo en un entorno real.    **Perú: Rendimiento de IDS en redes SDN (PUCP, 2021)**  Evalua el rendimiento de IDS (Snort, Suricata, Ml) en redes definidas por software, lo cual son escenarios modernos de redes virtualizadas, por ello es su limitación, es decir, se centra en SDN, no redes empresariales convencionales.  **Argentina: IDS en redes industriales o detección de intrusiones en redes industriales (UNPL, CACIC, 2022)**  Evalúa 9 algoritmos ML en tráfico Modbus, que es usado en redes industriales, lo cual su enfoque es su seguridad en infraestructuras criticas y no es aplicable directamente a redes corporativas comunes.    **España: IDS con ML para trafico real (UPM, 2021)**  Implementa un IDS usando Random Forest, entrenado con dataset UNSW-NB15, aplicado en entornos reales, lo cual lo limita porque depende de datasets internacionales, no locales. |

# 

# EVALUACION Y ANALISIS GRUPAL

| **Nombre del Proyecto** | **Conocimientos Teóricos y Técnicos Referentes a lo Planteado** | **Impacto de la Solución a Implementar** | **Preferencia Personal por el Tema** | **Factibilidad Técnica de la Solución** | **Total** |
| --- | --- | --- | --- | --- | --- |
| **Propuesta 1:** | Alejandro 2  Yersson 5  total:  7 | Alejandro 4  Yersson 4  total: 8 | Alejandro 2  Yersson 4  total: 6 | Alejandro 3  Yersson 2  total: 5 | 26 |
| **Propuesta 2:** | Alejandro 3  Andree 3  total: 6 | Alejandro 5  Andree 5  total: 10 | Alejandro 5  Andree 5  total: 10 | Alejandro 4  Andree 4  total: 8 | 34 |
| **Propuesta 3:** | Andree 2  Yersson 3  total: 5 | Andree 3  Yersson 2  total: 5 | Andree 2  Yersson 1  total: 3 | Andree 2  Yersson 3  total: 5 | 18 |

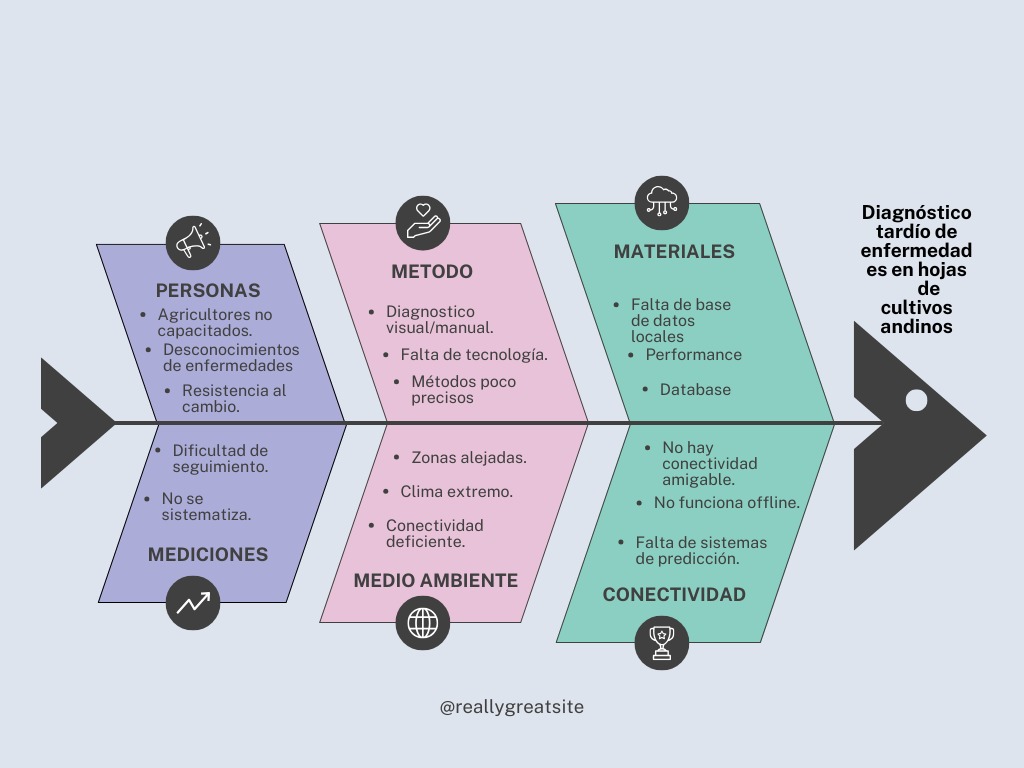
# JUSTIFICACIÓN E IMPORTANCIA

Desde el punto de vista de la Gestión de Proyectos, la selección de esta propuesta responde a su viabilidad, impacto social y alineación con los Objetivos de Desarrollo Sostenible (ODS). El proyecto de detección temprana de enfermedades en cultivos mediante redes neuronales convolucionales se ejecutará considerando buenas prácticas de planificación, control y gestión de riesgos, lo que permitirá optimizar recursos limitados (tiempo, presupuesto y alcance) y garantizar entregables en menos de 10 semanas. Además, la relevancia del problema asegura un alto valor agregado para las comunidades rurales, donde la reducción de pérdidas agrícolas se traduce directamente en mejoras económicas y de seguridad alimentaria.

Desde el punto de vista de la Ingeniería de Software, se desarrollará un modelo de inteligencia artificial basado en CNN implementado en frameworks como PyTorch y TensorFlow, utilizando técnicas de preprocesamiento y aumento de datos para garantizar la precisión del sistema. El proyecto implica fases propias de la disciplina: análisis de requerimientos, diseño, construcción, pruebas y despliegue de un prototipo funcional. Se busca no solo generar un producto mínimo viable (MVP) con resultados cuantitativos (precisión, recall y exactitud del modelo), sino también sentar las bases para futuras ampliaciones, como la integración en aplicaciones móviles accesibles a los agricultores.

Desde nuestro punto de vista como equipo, seleccionamos esta idea porque une innovación tecnológica con un impacto social real. Entre las alternativas evaluadas, esta propuesta nos motiva por su contribución directa a la solución de una problemática que afecta tanto a comunidades andinas como al país en general. Además, nos permite aplicar de manera práctica los conocimientos adquiridos en inteligencia artificial, visión por computadora y gestión de proyectos. Creemos que, más allá de la calificación académica, este trabajo puede ser un punto de partida hacia soluciones más amplias que apoyen al agricultor peruano frente a los retos de las plagas y el cambio climático.

# DIAGRAMA DE ISHIKAWA BASADO EN ERRORES



# 

# DESARROLLO DE LA METODOLOGÍA ÁGIL

## Historias de usuario

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ID** | **Historia de Usuario** | **Criterios de Aceptación** | **Prioridad** | **Estimación (horas)** |
| HU-1 | Como desarrollador, quiero configurar el entorno de desarrollo con PyTorch y librerías necesarias, para comenzar a trabajar en el proyecto. | Entorno funcionando con PyTorch, OpenCV y Pandas. | Alta | 8h |
| HU-2 | Como equipo de datos, queremos recolectar y organizar un conjunto inicial de imágenes de hojas sanas y enfermas, para tener una base de trabajo. | Dataset cargado y estructurado por carpetas/clases. | Alta | 12h |
| HU-3 | Como desarrollador, quiero implementar preprocesamiento de imágenes (resize, normalización, aumentos), para preparar los datos para el modelo. | Código ejecuta aumentos básicos correctamente. | Alta | 8h |
| HU-4 | Como científico de datos, quiero diseñar la arquitectura base de la CNN en PyTorch, para establecer una línea base de rendimiento. | Red CNN inicial entrenable y validada. | Alta | 12h |
| HU-5 | Como usuario, quiero subir una imagen de una hoja y recibir un diagnóstico preliminar (sana/enferma), para tener una primera evaluación. | Sistema diferencia ≥80% precisión. | Alta | 8h |
| HU-6 | Como usuario, quiero que el sistema identifique automáticamente el tipo de cultivo (papa, maíz, tomate), para mayor precisión en el diagnóstico. | Sistema identifica cultivo correctamente. | Alta | 12h |
| HU-7 | Como usuario, quiero conocer la enfermedad específica que afecta a la planta, para aplicar el tratamiento correcto. | Precisión ≥85% en 10 clases iniciales. | Alta | 16h |
| HU-8 | Como usuario, quiero ver el nivel de confianza del diagnóstico, para entender la fiabilidad del resultado. | Sistema devuelve probabilidad (%) en salida. | Alta | 4h |
| HU-9 | Como usuario, quiero una interfaz web sencilla para subir imágenes y ver el resultado, para acceder sin necesidad de instalar nada. | Navegador muestra diagnóstico sin errores. | Alta | 12h |
| HU-10 | Como usuario, quiero una interfaz simple para subir imágenes (drag & drop, botón), para obtener un diagnóstico rápido. | Imagen cargada y procesada sin fallos. | Alta | 4h |
| HU-11 | Como científico de datos, quiero ampliar el dataset con imágenes de diferentes condiciones, para mejorar la generalización del modelo. | Precisión mejora ≥3% con dataset ampliado. | Media | 8h |
| HU-12 | Como científico de datos, quiero aplicar transfer learning en PyTorch usando modelos preentrenados (ResNet, VGG, AlexNet), para mejorar la precisión del diagnóstico. | Modelo preentrenado ajustado en PyTorch. | Alta | 16h |
| HU-13 | Como científico de datos, quiero generar métricas de validación (precisión, recall, F1), para evaluar el desempeño del modelo. | Métricas generadas después del entrenamiento. | Alta | 8h |
| HU-14 | Como usuario, quiero acceder a un historial de diagnósticos anteriores, para dar seguimiento a mis cultivos. | Lista básica de diagnósticos previos accesible. | Media | 8h |
| HU-15 | Como usuario, quiero ver ejemplos visuales de hojas sanas y enfermas, para comparar con mis cultivos. | Página muestra imágenes de referencia. | Media | 4h |
| HU-16 | Como usuario, quiero recibir recomendaciones básicas según el diagnóstico, para tomar acciones rápidas. | Texto sugerido para cada enfermedad. | Alta | 8h |
| HU-17 | Como usuario, quiero que el sistema clasifique la severidad de la enfermedad (leve, moderada, grave), para priorizar acciones. | Diagnóstico incluye nivel de gravedad. | Media | 8h |
| HU-18 | Como usuario, quiero que el sistema valide la calidad de la imagen (borrosa/incompleta), para repetir la captura si es necesario. | Advertencia en caso de mala calidad. | Media | 8h |
| HU-19 | Como usuario, quiero generar reportes en PDF con mis diagnósticos, para compartirlos con técnicos agrícolas. | Botón descarga diagnóstico en PDF. | Media | 8h |
| HU-20 | Como investigador, quiero exportar datos anonimizados (CSV), para analizar patrones de propagación de enfermedades. | Datos exportables sin info personal. | Media | 8h |
| HU-21 | Como usuario, quiero cambiar el idioma de la interfaz (Español/Quechua), para usar la app en mi idioma preferido. | Menú de configuración cambia idioma. | Media | 8h |
| HU-22 | Como usuario, quiero poder dar retroalimentación sobre el diagnóstico (correcto/incorrecto), para mejorar el sistema. | Registro de feedback disponible. | Media | 4h |
| HU-23 | Como investigador, quiero visualizar métricas gráficas (matriz de confusión), para validar los resultados del modelo. | Página muestra gráfico generado. | Alta | 8h |
| HU-24 | Como usuario, quiero añadir notas personales a cada diagnóstico, para llevar un registro detallado de mis cultivos. | Campo de notas guardado junto al diagnóstico. | Media | 4h |
| HU-25 | Como desarrollador, quiero optimizar el modelo para despliegue web, para que sea rápido y eficiente. | Modelo reducido y funcional en servidor. | Alta | 12h |
| HU-26 | Como usuario, quiero recibir alertas tempranas en la interfaz web, para actuar rápido ante enfermedades críticas. | Alertas visibles en el sistema. | Media | 8h |
| HU-27 | Como científico de datos, quiero validar el modelo con imágenes de prueba externas, para asegurar robustez. | Diagnósticos probados con set externo. | Alta | 8h |
| HU-28 | Como desarrollador, quiero implementar backup automático del modelo entrenado, para no perder avances. | Modelo guardado periódicamente. | Media | 4h |
| HU-29 | Como desarrollador, quiero documentar técnicamente el sistema, para que pueda ser replicado. | Documento técnico con instalación y uso. | Alta | 12h |
| HU-30 | Como usuario, quiero acceder a un manual de usuario sencillo, para entender cómo usar la interfaz web. | PDF disponible con pasos básicos. | Media | 8h |

## Estimación de tiempos

1. **Estimación inicial:**

* Estimación total (suma de todas las historias de usuario): **256 horas**.
* Este total representa el esfuerzo combinado del equipo, ya distribuido entre las 30 historias.

1. **Capacidad del Equipo por Semana:**

Los integrantes indicaron su disponibilidad diaria:

* **Alejandro:** 2.5 horas/día
* **André:** 2.0 horas/día
* **Yersson (tú):** 4.0 horas/día

Promedio por persona (día):

(Se redondea a **2.83 h/día por persona** — este será el valor de referencia de disponibilidad diaria).

**Capacidad semanal por persona (7 días):**

**Capacidad total del equipo por semana (3 integrantes):**

1. **Verificación para ≤ 10 Semanas:**

* Total, de horas requeridas: 256 h
* Capacidad total del equipo en 10 semanas:

**Resultado:** La capacidad disponible (595 h) es ampliamente superior a las 256 h requeridas. El equipo tiene **holgura** para completar el proyecto dentro de 10 semanas y absorber imprevistos, pruebas adicionales o correcciones.

1. **Distribución del Trabajo:**

Horas por semana requeridas (equipo):

Horas por persona por semana:

Horas por persona por día (7 días):

## **Estimación de épicas**

**Épica 1 – Preparación y Configuración Técnica**

**Descripción:** Configuración del entorno de desarrollo, dataset y arquitectura inicial del modelo CNN en PyTorch.

* HU-1: Configuración del entorno (8h, Prioridad: Alta)
* HU-2: Recolección y organización del dataset (12h, Prioridad: Alta)
* HU-3: Preprocesamiento de imágenes (8h, Prioridad: Alta)
* HU-4: Diseño de arquitectura base CNN (12h, Prioridad: Alta)

**Total, estimado Épica 1: 40h**

**Épica 2 – Funcionalidades Principales (Diagnóstico Web)**

**Descripción:** Funcionalidades base para diagnóstico de hojas mediante CNN, interfaz web y experiencia mínima viable.

* HU-5: Clasificación básica (8h, Prioridad: Alta)
* HU-6: Detección de cultivo (12h, Prioridad: Alta)
* HU-7: Diagnóstico específico por enfermedad (16h, Prioridad: Alta)
* HU-8: Nivel de confianza (4h, Prioridad: Alta)
* HU-9: Interfaz web (12h, Prioridad: Alta)
* HU-10: Subida intuitiva de imágenes (4h, Prioridad: Alta)

**Total, estimado Épica 2: 56h**

**Épica 3 – Optimización y Validación del Modelo**

**Descripción:** Mejoras de rendimiento y precisión del modelo usando técnicas avanzadas.

* HU-11: Entrenamiento con dataset ampliado (8h, Prioridad: Media)
* HU-12: Transfer learning con modelos preentrenados (16h, Prioridad: Alta)
* HU-13: Métricas de validación (8h, Prioridad: Alta)
* HU-23: Gráficos de métricas (8h, Prioridad: Alta)
* HU-25: Optimización del modelo para despliegue web (12h, Prioridad: Alta)
* HU-27: Validación con dataset externo (8h, Prioridad: Alta)

**Total, estimado Épica 3: 60h**

**Épica 4 – Funcionalidades de Usuario y Experiencia**

**Descripción:** Funcionalidades adicionales para enriquecer la experiencia del agricultor en el uso de la aplicación web.

* HU-14: Historial de diagnósticos (8h, Prioridad: Media)
* HU-15: Ejemplos visuales (4h, Prioridad: Media)
* HU-16: Recomendaciones básicas (8h, Prioridad: Alta)
* HU-17: Clasificación de severidad (8h, Prioridad: Media)
* HU-18: Validación de calidad de imagen (8h, Prioridad: Media)
* HU-24: Guardar notas junto al diagnóstico (4h, Prioridad: Media)
* HU-26: Alertas tempranas en interfaz (8h, Prioridad: Media)

**Total, estimado Épica 4: 48h**

**Épica 5 – Reportes, Exportaciones e Inclusión**

**Descripción:** Generación de reportes, exportación de datos, idiomas y soporte para la toma de decisiones.

* HU-19: Reportes PDF (8h, Prioridad: Media)
* HU-20: Exportación de datos anonimizados (8h, Prioridad: Media)
* HU-21: Multilenguaje Español/Quechua (8h, Prioridad: Media)
* HU-22: Retroalimentación del usuario (4h, Prioridad: Media)
* HU-30: Manual de usuario (8h, Prioridad: Media)

**Total, estimado Épica 5: 36h**

**Épica 6 – Soporte, Seguridad y Documentación**

**Descripción:** Mantenimiento del sistema, seguridad de datos y documentación del proyecto.

* HU-28: Backup automático del modelo (4h, Prioridad: Media)
* HU-29: Documentación técnica (12h, Prioridad: Alta)

**Total, estimado Épica 6: 16h**

**Resumen de horas por épica**

* **Épica 1 – Preparación y Configuración Técnica:** 40h
* **Épica 2 – Funcionalidades Principales (Diagnóstico Web):** 56h
* **Épica 3 – Optimización y Validación del Modelo:** 60h
* **Épica 4 – Funcionalidades de Usuario y Experiencia:** 48h
* **Épica 5 – Reportes, Exportaciones e Inclusión:** 36h
* **Épica 6 – Soporte, Seguridad y Documentación:** 16h

**Total, general:** **256h**.

## **Product backlog según prioridades y estimaciones**

**Ordenado de mayor a menor prioridad.**

**Épicas de Prioridad Alta:**

**Épica 2 – Funcionalidades Principales (Diagnóstico Web) (56h, Prioridad: Alta)**  
Incluye las funcionalidades esenciales para el diagnóstico en la interfaz web.

* HU-5: Clasificación básica (8h, Alta)
* HU-6: Detección de cultivo (12h, Alta)
* HU-7: Diagnóstico específico por enfermedad (16h, Alta)
* HU-8: Nivel de confianza (4h, Alta)
* HU-9: Interfaz web (12h, Alta)
* HU-10: Subida intuitiva de imágenes (4h, Alta)

**Épica 3 – Optimización y Validación del Modelo (60h, Prioridad: Alta)**  
Mejoras de rendimiento y validación del modelo CNN en PyTorch.

* HU-11: Entrenamiento con dataset ampliado (8h, Media)
* HU-12: Transfer learning (16h, Alta)
* HU-13: Métricas de validación (8h, Alta)
* HU-23: Gráficos de métricas (8h, Alta)
* HU-25: Optimización del modelo para despliegue web (12h, Alta)
* HU-27: Validación con dataset externo (8h, Alta)

**Épica 1 – Preparación y Configuración Técnica (40h, Prioridad: Alta)**  
Etapas iniciales para preparar el entorno y la arquitectura base.

* HU-1: Configuración del entorno (8h, Alta)
* HU-2: Recolección y organización del dataset (12h, Alta)
* HU-3: Preprocesamiento de imágenes (8h, Alta)
* HU-4: Diseño de arquitectura base CNN (12h, Alta)

**Épicas de Prioridad Media:**

**Épica 4 – Funcionalidades de Usuario y Experiencia (48h, Prioridad: Media)**  
Aporta valor agregado para la interacción y usabilidad del sistema.

* HU-14: Historial de diagnósticos (8h, Media)
* HU-15: Ejemplos visuales (4h, Media)
* HU-16: Recomendaciones básicas (8h, Alta)
* HU-17: Clasificación de severidad (8h, Media)
* HU-18: Validación de calidad de imagen (8h, Media)
* HU-24: Guardar notas junto al diagnóstico (4h, Media)
* HU-26: Alertas tempranas en interfaz (8h, Media)

**Épica 5 – Reportes, Exportaciones e Inclusión (36h, Prioridad: Media)**  
Facilita el análisis y la inclusión de agricultores quechua-hablantes.

* HU-19: Reportes PDF (8h, Media)
* HU-20: Exportación de datos anonimizados (8h, Media)
* HU-21: Multilenguaje Español/Quechua (8h, Media)
* HU-22: Retroalimentación del usuario (4h, Media)
* HU-30: Manual de usuario (8h, Media)

**Épica de Prioridad Baja:**

**Épica 6 – Soporte, Seguridad y Documentación (16h, Prioridad: Baja)**  
Garantiza la seguridad y documentación del sistema.

* HU-28: Backup automático del modelo (4h, Media)
* HU-29: Documentación técnica (12h, Alta)

## Cronograma: diagrama de Gantt

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Nombre Actividad** | **Actividades** | **Inicio** | **Días** | **Final** | **Épicas/HU** |
| **Preparación & Configuración** | Configuración del entorno; Recolección y organización del dataset; Preprocesamiento de imágenes; Diseño de la CNN base | 01/09/2025 | 7 | 07/09/2025 | Épica 1 (HU-1 a HU-4) |
| **Funcionalidades Básicas** | Clasificación sana/enferma; Detección de cultivo; Diagnóstico específico; Nivel de confianza; Interfaz web; Subida de imágenes | 08/09/2025 | 7 | 14/09/2025 | Épica 2 (HU-5 a HU-10) |
| **Optimización Inicial del Modelo** | Entrenamiento con dataset ampliado; Transfer learning; Métricas de validación; Gráficos de métricas; Optimización para web; Validación externa | 15/09/2025 | 7 | 21/09/2025 | Épica 3 (HU-11 a HU-13, HU-23, HU-25, HU-27) |
| **Experiencia de Usuario I** | Historial de diagnósticos; Ejemplos visuales; Recomendaciones básicas | 22/09/2025 | 7 | 28/09/2025 | Épica 4 (HU-14, HU-15, HU-16) |
| **Experiencia de Usuario II** | Clasificación de severidad; Validación de calidad de imagen; Notas en diagnósticos; Alertas tempranas | 29/09/2025 | 7 | 05/10/2025 | Épica 4 (HU-17, HU-18, HU-24, HU-26) |
| **Reportes y Exportaciones** | Reportes PDF; Exportación de datos anonimizados; Soporte multilenguaje; Retroalimentación; Manual de usuario | 06/10/2025 | 7 | 12/10/2025 | Épica 5 (HU-19 a HU-22, HU-30) |
| **Mantenimiento y Documentación** | Backup automático del modelo; Documentación técnica | 13/10/2025 | 7 | 19/10/2025 | Épica 6 (HU-28, HU-29) |

**Diagrama de Gantt:**

## **Definición de Sprints**

**Capacidad real del sprint (equipo)**

* Promedio diario por persona: 2.83 h/día
* Días del sprint: 7 días
* Horas disponibles por persona/sprint:
* Equipo de 3 personas:

Cada sprint tiene una capacidad de ~59 horas.

**Total de esfuerzo a cubrir**

* Total del backlog: 256 h
* Número de sprints necesarios:

Se necesitarán 5 sprints (los últimos con menos carga).

**Asignación de historias de usuario a cada sprint**

Agruparemos HU en orden de prioridad (alta → media → baja) hasta llenar ~59h por sprint.

**Sprint 1 (59h máx – Total: 56h)**

Enfoque: Preparación técnica y base del sistema.

* HU-1: Configuración del entorno (8h)
* HU-2: Recolección y organización del dataset (12h)
* HU-3: Preprocesamiento de imágenes (8h)
* HU-4: Diseño de arquitectura base CNN (12h)
* HU-5: Clasificación básica (8h)
* HU-8: Nivel de confianza (4h)  
  **Total Sprint 1 = 56h**

**Sprint 2 (59h máx – Total: 56h)**

Enfoque: Funcionalidades principales en la interfaz web.

* HU-6: Detección de cultivo (12h)
* HU-7: Diagnóstico específico por enfermedad (16h)
* HU-9: Interfaz web (12h)
* HU-10: Subida intuitiva de imágenes (4h)
* HU-16: Recomendaciones básicas (8h)
* HU-15: Ejemplos visuales (4h)  
  **Total Sprint 2 = 56h**

**Sprint 3 (59h máx – Total: 60h)**

Enfoque: Optimización y validación del modelo.

* HU-12: Transfer learning (16h)
* HU-13: Métricas de validación (8h)
* HU-23: Gráficos de métricas (8h)
* HU-25: Optimización del modelo para web (12h)
* HU-27: Validación con dataset externo (8h)
* HU-11: Dataset ampliado (8h)  
  **Total Sprint 3 = 60h**

**Sprint 4 (59h máx – Total: 52h)**

Enfoque: Funcionalidades de usuario y reportes básicos.

* HU-14: Historial de diagnósticos (8h)
* HU-17: Clasificación de severidad (8h)
* HU-18: Validación de calidad de imagen (8h)
* HU-24: Notas en diagnóstico (4h)
* HU-26: Alertas tempranas (8h)
* HU-19: Reportes PDF (8h)
* HU-22: Retroalimentación (4h)
* HU-28: Backup automático del modelo (4h)  
  **Total Sprint 4 = 52h**

**Sprint 5 (59h máx – Total: 32h)**

Enfoque: Inclusión, documentación y cierre del proyecto.

* HU-20: Exportación de datos anonimizados (8h)
* HU-21: Multilenguaje (8h)
* HU-30: Manual de usuario (8h)
* HU-29: Documentación técnica (12h)  
  **Total Sprint 5 = 32h**

**Tabla de los Sprint:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Sprint** | **Historias de Usuario** | **Actividades** | **Horas** | **Total Sprint** |
| **Sprint 1** | HU-1, HU-2, HU-3, HU-4, HU-5, HU-8 | Configuración del entorno; Recolección y organización del dataset; Preprocesamiento de imágenes; Diseño de CNN base; Clasificación básica; Mostrar nivel de confianza | 8 + 12 + 8 + 12 + 8 + 4 | **56h** |
| **Sprint 2** | HU-6, HU-7, HU-9, HU-10, HU-16, HU-15 | Detección de cultivo; Diagnóstico específico por enfermedad; Interfaz web; Subida intuitiva de imágenes; Recomendaciones básicas; Ejemplos visuales | 12 + 16 + 12 + 4 + 8 + 4 | **56h** |
| **Sprint 3** | HU-12, HU-13, HU-23, HU-25, HU-27, HU-11 | Transfer learning; Generar métricas; Visualización de métricas; Optimización del modelo para web; Validación con dataset externo; Entrenamiento con dataset ampliado | 16 + 8 + 8 + 12 + 8 + 8 | **60h** |
| **Sprint 4** | HU-14, HU-17, HU-18, HU-24, HU-26, HU-19, HU-22, HU-28 | Historial de diagnósticos; Clasificación de severidad; Validación de calidad de imagen; Notas en diagnóstico; Alertas tempranas; Reportes PDF; Retroalimentación; Backup automático | 8 + 8 + 8 + 4 + 8 + 8 + 4 + 4 | **52h** |
| **Sprint 5** | HU-20, HU-21, HU-30, HU-29 | Exportación de datos anonimizados; Multilenguaje; Manual de usuario; Documentación técnica | 8 + 8 + 8 + 12 | **32h** |

REFERENCIAS:

1. LOZADA-PORTILLA, H., et al. Clasificación de enfermedades en hojas de papa utilizando técnicas de aprendizaje profundo con CNN y modelos híbridos. 2025. Disponible en: <http://scielo.senescyt.gob.ec/scielo.php?script=sci_arttext&pid=S2631-26542025000100142>
2. FERNANDEZ FERNANDEZ, K.D. y PINGLO CABEZAS, W.R. Redes neuronales convolucionales para la detección de enfermedades en cultivos de arroz: una revisión sistemática. 2025. Disponible en: <https://repositorio.uss.edu.pe/bitstream/handle/20.500.12802/14298/Fernandez%20Fernandez%20Kenedy%20David%20&%20Pinglo%20Cabezas%20Williams%20Rafael.pdf?sequence=1&isAllowed=y>
3. DIALNET. Detección de enfermedades en cultivos de papa usando procesamiento digital de imágenes. Disponible en: <https://dialnet.unirioja.es/descarga/articulo/8228807.pdf>
4. FLYPIX.AI. Detección de enfermedades de cultivos impulsada por IA. Disponible en: <https://flypix.ai/es/blog/crop-disease-detection/>
5. UNIVERSIDAD DE LOS ANDES. Sistema de visión por computadora para la monitorización y gestión de cultivos con visión artificial. Disponible en: <https://repositorio.uniandes.edu.co/entities/publication/05a1c5a2-b619-447b-9883-24fb71ebdf27>
6. MIOTI. Agricultura Inteligente: El poder del Computer Vision. 2024. Disponible en: <https://mioti.es/es/blog-agricultura-inteligente-el-poder-del-computer-vision/>
7. ACOFI. Sistemas de visión por computadora aplicados a cultivos tropicales con UAV. Disponible en: <https://acofipapers.org/index.php/eiei/article/download/2270/1807/5809>
8. PRATEC. Agricultura y cultura andina: tecnologías campesinas aplicadas a la agricultura andina. Disponible en: <https://www.pratec.org/wpress/pdfs-pratec/agricultura-y-cultura-andina-final.pdf>
9. FLORES MENDOZA, J.C. y MEJÍA CARHUAJULCA, G. Agricultura y cultura andina. Universidad Nacional Pedro Ruiz Gallo. Disponible en: <https://repositorio.unprg.edu.pe/bitstream/handle/20.500.12893/11478/Flores_Mendoza_Juan_Carlos%20y%20Mejia_Carhuajulca_Gianfranco.pdf?sequence=1&isAllowed=y>
10. BANCO INTERAMERICANO DE DESARROLLO (BID). Agricultura de precisión: tecnología para enfrentar el cambio climático. 2024. Disponible en: <https://blogs.iadb.org/innovacion/es/agricultura-de-precision-tecnologia-para-enfrentar-el-cambio-climatico/>