# Aplicativo web basado en firmas Yara para la detección de amenazas cibernéticas

Johan Sebastián Pacheco Moreno

Julián David Barinas

Facultad de Ingeniería, Universidad EAN

IFA00648: Proyecto de grado

John Jairo Porras

6 de febrero de 2025

# Índice

**Resumen**

En este documento se presenta un proyecto que desarrolla una plataforma web para el análisis de archivos y detección de amenazas mediante firmas YARA. Una herramienta ampliamente utilizada en ciberseguridad para la identificación de patrones maliciosos en archivos y procesos.

El enfoque del proyecto está en la seguridad informática, proporcionando una herramienta accesible para investigadores de ciberseguridad, analistas forenses y profesionales de TI interesados en la detección proactiva de malware. La plataforma se diseñó con una arquitectura simple y segura, garantizando la integridad de los archivos analizados y la privacidad de los usuarios.

**Palabras clave:** ciberseguridad, YARA, malware, análisis de archivos, detección de amenazas

**Abstract**

This document presents a project that develops a web platform for file analysis and threat detection using YARA signatures. YARA is a widely used tool in cybersecurity for identifying malicious patterns in files and processes.

The project focuses on information security, providing an accessible tool for cybersecurity researchers, forensic analysts, and IT professionals interested in proactive malware detection. The platform is designed with a simple and secure architecture, ensuring the integrity of the analyzed files and the privacy of users.

**Keywords:** cybersecurity, YARA, malware, file analysis, threat detection

## Introducción

Cada día estamos expuestos a todo tipo de amenaza cibernética debido al constante avance de las tecnologías y al incremento de ataques dirigidos a individuos y organizaciones. La proliferación de malware, ransomware, phishing y otras técnicas maliciosas hace imprescindible contar con herramientas que permitan detectar y mitigar riesgos de manera eficiente. Incluso las soluciones más sencillas pueden desempeñar un papel fundamental en la ciberseguridad, ya que proporcionan una primera línea de defensa ante posibles amenazas. (Orvisa Comunicaciones, 2022).

Por eso la importancia de la "ciberseguridad" en nuestro día a día ya que es un aspecto crucial en la protección de sistemas y datos ante amenazas digitales. Sin embargo, muchas herramientas de detección de malware y vulnerabilidades requieren conocimientos avanzados para su implementación y uso. Este proyecto busca desarrollar una plataforma de software que facilite el análisis de vulnerabilidades en archivos y sistemas mediante firmas YARA, para de esta forma permitir a los usuarios ejecutar un diagnóstico mediante reglas predefinidas y editables. Así pues, se ofrece una interfaz accesible y eficiente para identificar riesgos de seguridad, contribuyendo a la protección y prevención de ataques informáticos descrito dentro de un contexto de direccionamiento de red en IPv4 (Rincon, 2021).

De la misma forma, las firmas YARA permiten detectar patrones específicos de malware en archivos y sistemas, para así facilitar su identificación de una manera sencilla. Gracias a su flexibilidad y código abierto, puede ser utilizada por expertos e inexpertos de la ciberseguridad. Es por eso que en este documento se detalla el proceso de desarrollo de la plataforma, incluyendo la planificación, el diseño y la implementación de sus funcionalidades principales. Se describen los requisitos técnicos y funcionales, la selección de tecnologías y las pruebas de

validación que aseguran su efectividad en la detección de amenazas. Además, se aborda la integración con una base de datos para el almacenamiento seguro de reglas YARA y resultados de análisis, así como la implementación de un sistema de registro y autenticación de usuarios, garantizando el acceso seguro a la plataforma y la gestión personalizada de configuraciones.

## Objetivos

### Objetivo general

Desarrollar un aplicativo web que implemente firmas YARA para el análisis y detección de amenazas cibernéticas, permitiendo la identificación temprana de patrones maliciosos en archivos y sistemas.

### Objetivos Específicos

Analizar el funcionamiento de las firmas YARA en el contexto de la detección de amenazas cibernéticas, identificando sus ventajas y limitaciones en comparación con otras técnicas de análisis.

Implementar un módulo de detección basado en YARA dentro de un aplicativo web, permitiendo la carga y escaneo de archivos en busca de patrones maliciosos.

Mejorar la integración de firmas YARA con una interfaz web intuitiva, asegurando que los usuarios puedan gestionar reglas y visualizar resultados de análisis de manera eficiente.

Evaluar la efectividad del aplicativo web en la detección de amenazas, comparándolo con herramientas similares y midiendo su tasa de detección y falsos positivos.

## Definición del problema

El análisis de amenazas cibernéticas requiere herramientas especializadas capaces de identificar patrones maliciosos en archivos y sistemas. Las firmas YARA permiten la detección de amenazas mediante reglas personalizadas, facilitando la identificación de comportamientos sospechosos. Sin embargo, su uso está mayormente limitado a entornos de línea de comandos o integraciones con plataformas de seguridad avanzadas, lo que restringe su accesibilidad a usuarios sin conocimientos técnicos profundos. (Cano & Rocha, 2019)

Existen soluciones comerciales centradas en la detección de amenazas, pero muchas presentan barreras de costo, implementación o complejidad en la configuración (Delgado, 2020). Como resultado, empresas pequeñas, investigadores y analistas independientes carecen de herramientas accesibles para aprovechar esta tecnología sin depender de software propietario o infraestructuras complejas.

Este proyecto aborda la ausencia de una plataforma web que permita la gestión e implementación de firmas YARA en un entorno accesible, proporcionando a los usuarios una interfaz intuitiva para la creación, edición y aplicación de reglas de detección. La investigación se enfocará en desarrollar un aplicativo web que facilite el análisis de archivos en busca de amenazas, optimizando la detección sin comprometer la usabilidad.

**Preguntas de investigación**

¿Cuáles son los principales retos técnicos en la implementación de firmas YARA en un entorno web?

¿Qué estrategias pueden mejorar la eficiencia del análisis de archivos con YARA en una aplicación web?

¿Cómo se compara el rendimiento del aplicativo con herramientas existentes para la detección de amenazas?

Este estudio establecerá los lineamientos técnicos y metodológicos para la integración de firmas YARA en una solución web funcional, evaluando su desempeño y aplicabilidad en distintos escenarios de análisis de amenazas.

## Justificación

Las amenazas cibernéticas continúan en aumento, y aunque las firmas YARA son una herramienta efectiva para la detección de patrones maliciosos, su acceso es limitado debido a su complejidad técnica y a que muchas soluciones requieren altos costos o conocimientos especializados. Esto impide que pequeños negocios, investigadores y usuarios sin formación avanzada en ciberseguridad aprovechen su potencial. (Cano M., 2023)

El desconocimiento sobre estas tecnologías también dificulta su adopción, lo que deja a muchos sistemas vulnerables (Beltran Muñoz, 2024). Para abordar este problema, se desarrollará un aplicativo web que implemente firmas YARA de manera accesible e intuitiva, facilitando la detección de amenazas sin necesidad de herramientas especializadas.

Esta solución permitirá a más usuarios gestionar y aplicar reglas de detección de forma sencilla, mejorando la seguridad informática en distintos entornos. Su diseño priorizará la facilidad de uso y la accesibilidad, con el objetivo de cerrar la brecha en la adopción de este tipo de herramientas.

# Análisis de requerimientos

## 1. Intención del Producto

El propósito del producto es crear una plataforma web accesible y fácil de usar para la detección de amenazas cibernéticas mediante firmas YARA. De forma que esta permita a los usuarios gestionar, aplicar y analizar firmas YARA sobre archivos, identificando patrones o contenido maliciosos con el objetivo de mejorar la seguridad y la respuesta ante incidentes cibernéticos.

### *1.1. Funcionalidades principales:*

- Cargar y escanear archivos: Los usuarios podrán cargar archivos locales o remotos y escanearlos en busca de amenazas, utilizando reglas de YARA personalizadas o prediseñadas.
- Gestión de reglas YARA: Crear, editar, y eliminar reglas YARA de forma fácil y eficiente a través de una interfaz gráfica amigable.
- Visualización de resultados: Presentación clara y concisa de los resultados del análisis, destacando los patrones maliciosos detectados y proporcionando información sobre el archivo afectado.
- Notificaciones: Alertas en tiempo real sobre amenazas detectadas y posibles falsos positivos.

### *1.2. Audiencia objetivo:*

- Pequeñas empresas: Sin recursos suficientes para invertir en soluciones comerciales costosas.
- Investigadores y analistas de ciberseguridad independientes: Necesitan una herramienta accesible y flexible.
- Usuarios no técnicos: Aquellos que buscan proteger sus sistemas de amenazas sin necesidad de conocimientos avanzados en ciberseguridad.

## 2. Verificación de Parámetros de Diseño

El diseño del producto será validado mediante los siguientes parámetros clave que aseguren su efectividad y usabilidad:

### *2.1. UX / UI (Usabilidad e Interfaz de Usuario)*

- La interfaz debe ser intuitiva y fácil de usar, garantizando que usuarios sin experiencia técnica puedan gestionar firmas YARA sin complicaciones.
- Los elementos de la interfaz deben ser claros y bien organizados (por ejemplo, botones para cargar archivos, gestión de reglas, vista de resultados).
- Diseño responsive para adaptarse a diferentes dispositivos (computadoras, tabletas y móviles).

### *2.2. Integración con Firmas YARA*

- El sistema debe permitir que los usuarios elijan de un repositorio de reglas predefinidas cuales quieren implementar.
- Debe ser posible validar la sintaxis de las reglas YARA antes de su aplicación.
- El sistema debe ser capaz de procesar archivos de gran tamaño sin fallar o generar retrasos excesivos.

### *2.4. Compatibilidad con Diferentes Tipos de Archivos*

- El sistema debe ser capaz de analizar diversos tipos de archivos, como documentos, imágenes, ejecutables y archivos comprimidos, entre otros.
- Las firmas YARA deben ser efectivas en la identificación de patrones en distintos tipos de contenido (como bien podrían ser, texto en archivos binarios o scripts en archivos de texto).

## 3. Estimación de características y especificaciones del Producto

### *3.1. Desempeño*

- Tiempo de Respuesta: El sistema debe ofrecer un tiempo de respuesta rápido, con el objetivo de realizar el análisis de archivos con una velocidad consecuente a su tamaño. Si este tarda mucho tiempo debe ejecutar una notificación que alerte la imposibilidad de diagnosticar el archivo
- Capacidad de Análisis Concurrente: El sistema debe soportar al menos 10 usuarios simultáneos realizando análisis de archivos sin degradar significativamente el desempeño.

### *3.2. Potencia y Recursos del Sistema*

- Consumo de CPU y Memoria: El sistema debe estar optimizado para utilizar recursos de manera eficiente. Los análisis deben consumir poca memoria y CPU para no afectar otras operaciones en el servidor.
- Requerimientos de Servidor: Dependiendo del número de usuarios concurrentes, se debe asegurar que el servidor tenga suficiente capacidad de procesamiento y almacenamiento. Se recomienda una arquitectura basada en contenedores (por ejemplo, Docker) para facilitar el despliegue y escalabilidad.

### *3.3. Fiabilidad y Robustez*

- Tasa de Detección: El sistema debe tener una tasa de detección de amenazas superior al 90% en comparación con herramientas existentes de código abierto, tomando en cuenta la precisión de las reglas YARA implementadas.

# Marco teórico

## Ciberseguridad y Amenazas Cibernéticas

La ciberseguridad es una disciplina enfocada en la protección de sistemas informáticos, redes y datos frente a amenazas digitales. Con el auge de la digitalización y la interconectividad global, la superficie de ataque ha crecido significativamente, lo que ha dado lugar a un aumento en la sofisticación y frecuencia de los ciberataques. La seguridad de la información es crucial no solo para las empresas y gobiernos, sino también para los usuarios individuales, quienes diariamente enfrentan riesgos como el robo de identidad, fraudes financieros y espionaje digital (Srinivas, Das & Kumar, 2019) .

Las amenazas cibernéticas pueden clasificarse en diferentes categorías, dependiendo de su propósito y método de ataque. Algunas de las más relevantes incluyen:

**Malware:** programas maliciosos diseñados para infiltrarse, dañar o alterar el funcionamiento de un sistema sin el consentimiento del usuario. Ejemplos incluyen troyanos, spyware y virus informáticos (Kaspersky, 2023) .

**Ransomware:** una variante del malware que cifra los archivos del usuario y exige un pago para restaurar el acceso. En los últimos años, este tipo de ataque ha afectado a instituciones gubernamentales y grandes corporaciones (Trend Micro, 2023) .

**Phishing:** una técnica que utiliza correos electrónicos o sitios web fraudulentos para engañar a los usuarios y obtener información confidencial, como credenciales de acceso o datos bancarios (Symantec, 2022) .

Ataques de denegación de servicio distribuido (DDoS): buscan sobrecargar servidores con tráfico malicioso, provocando la interrupción de sus servicios (Cisco, 2021) .

Dado el crecimiento exponencial de estas amenazas, han surgido múltiples estrategias de defensa, entre ellas la detección basada en firmas y el análisis heurístico, tecnologías fundamentales en la lucha contra el malware (Zhu et al., 2020) .

## Evolución de las Amenazas Cibernéticas y su Impacto en la Seguridad Digital

Las amenazas cibernéticas han evolucionado considerablemente desde la aparición de los primeros virus informáticos en la década de 1980. Mientras que en sus inicios los ataques se realizaban principalmente por curiosidad o con fines de sabotaje, hoy en día son herramientas utilizadas por grupos cibercriminales y actores estatales para realizar espionaje, extorsión y desestabilización de infraestructuras críticas. Esta evolución ha sido impulsada por el desarrollo de nuevas tecnologías, la digitalización de servicios y la interconexión de dispositivos en la llamada era del Internet de las Cosas (IoT) (Singer & Friedman, 2014).

Uno de los factores que ha impulsado el crecimiento de los ataques cibernéticos es la facilidad con la que los atacantes pueden acceder a herramientas automatizadas y a servicios ilegales en la dark web. Actualmente, existen plataformas que ofrecen "malware-as-a-service" (MaaS), lo que permite a ciberdelincuentes sin conocimientos técnicos ejecutar ataques sofisticados sin necesidad de desarrollar su propio software malicioso. Este modelo de negocio ha facilitado la proliferación de ataques dirigidos y campañas de ransomware a nivel mundial (Hutchins, Cloppert & Amin, 2011).

## Inteligencia de Amenazas y su Rol en la Ciberseguridad

En el contexto de la ciberseguridad, la inteligencia de amenazas juega un papel fundamental en la prevención, detección y respuesta a ataques informáticos. Su objetivo principal es anticipar posibles ataques mediante el análisis de información sobre actores maliciosos, tácticas utilizadas y vulnerabilidades explotadas. A través de la inteligencia de amenazas, las organizaciones pueden mejorar su postura de seguridad y minimizar riesgos antes de que un ataque ocurra (Shackleford, 2021).

La inteligencia de amenazas se clasifica en distintos niveles según su alcance y aplicación. La inteligencia estratégica proporciona un análisis de alto nivel sobre tendencias y motivaciones de los atacantes, facilitando la toma de decisiones a largo plazo. La inteligencia táctica, en cambio, se enfoca en las técnicas y procedimientos específicos empleados en ataques cibernéticos, permitiendo el fortalecimiento de los controles de seguridad (Barnum, 2014). En un nivel más operacional, la inteligencia técnica y operativa brindan información detallada sobre indicadores de compromiso (IoC), como direcciones IP sospechosas, hashes de archivos maliciosos y patrones de ataque en tiempo real (Sauerwein et al., 2017).

En términos de aplicación, la inteligencia de amenazas permite a las organizaciones implementar estrategias proactivas de defensa. El uso de sistemas de correlación de eventos (SIEM), plataformas de intercambio de información como STIX/TAXII, y herramientas de detección como YARA, mejora la capacidad de identificación de malware avanzado. YARA, en particular, juega un papel clave al permitir la detección de amenazas mediante reglas de patrones específicas, lo que resulta fundamental para la identificación de nuevas variantes de malware (Sebastián et al., 2016).

El uso eficiente de inteligencia de amenazas en combinación con herramientas avanzadas de detección fortalece significativamente las defensas de ciberseguridad, permitiendo no solo responder a incidentes, sino también anticiparse a ellos. La capacidad de compartir información sobre ataques en tiempo real entre organizaciones ha demostrado ser un factor clave en la mitigación de riesgos a nivel global (Kharraz et al., 2015).

## Técnicas de Detección de Malware

El malware ha evolucionado significativamente, utilizando métodos avanzados de ofuscación y polimorfismo para evadir las soluciones de seguridad tradicionales. Por ello, las estrategias de detección han tenido que adaptarse continuamente. Existen tres enfoques principales para la identificación de amenazas digitales:

Detección basada en firmas: este método identifica malware a partir de patrones predefinidos almacenados en bases de datos. Si bien es eficiente para amenazas conocidas, tiene dificultades para detectar variantes nuevas o malware sin firma registrada (Sikorski & Honig, 2012) .

Análisis heurístico: permite detectar malware desconocido al analizar el comportamiento de un programa en busca de acciones sospechosas, como modificaciones en el registro o intentos de conexión remota (Egele et al., 2012) .

Análisis de comportamiento: monitorea la ejecución de programas en tiempo real, identificando actividades anómalas que puedan indicar la presencia de malware (Moser, Kruegel & Kirda, 2007) .

Dentro de estas estrategias, las firmas YARA han adquirido relevancia al proporcionar un enfoque flexible y personalizable para la detección de amenazas (Sebastián et al., 2016) .

## YARA: Herramienta para la Detección de Malware

YARA es una herramienta de código abierto creada para facilitar la clasificación y detección de malware mediante la aplicación de reglas personalizadas. Fue desarrollada originalmente por VirusTotal, una plataforma de análisis de amenazas adquirida por Google en 2012 (Álvarez, 2013) .

Las reglas de YARA están compuestas por tres elementos clave:

Metadatos: información descriptiva sobre la regla, como el autor y la fecha de creación.

Conjunto de cadenas: patrones de texto o secuencias binarias que se buscan en los archivos analizados.

Condiciones lógicas: expresiones booleanas que determinan si se cumple o no la regla.

YARA ha demostrado ser especialmente útil en la investigación de malware avanzado y en la detección de ataques dirigidos, al permitir la creación de reglas adaptadas a cada amenaza específica (Kharraz et al., 2015) .

## Aplicación de YARA en Entornos Web

A pesar de su potencia, YARA ha sido tradicionalmente utilizada en entornos de línea de comandos o como parte de soluciones avanzadas de ciberseguridad. Sin embargo, su integración en una plataforma web puede democratizar su uso y facilitar su adopción por parte de usuarios sin experiencia técnica.

Las principales ventajas de una aplicación web basada en YARA incluyen:

Interfaz gráfica amigable: permite a los usuarios definir y gestionar reglas sin necesidad de escribir código manualmente.

Escaneo automatizado de archivos: facilita la identificación de malware en documentos y programas subidos a la plataforma.

Generación de reportes detallados: brinda información estructurada sobre las amenazas detectadas y sus posibles impactos.

Implementaciones exitosas de este enfoque han demostrado que la accesibilidad y facilidad de uso mejoran significativamente la eficacia de las herramientas de detección de malware (Ligh et al., 2014) .

## Automatización en la Detección de Amenazas con YARA

La automatización en la detección de malware es un aspecto clave en la ciberseguridad moderna. A medida que las amenazas aumentan en volumen y sofisticación, es inviable depender exclusivamente de la intervención humana para analizar cada posible incidente. Por ello, la

integración de YARA con sistemas automatizados de respuesta a incidentes permite una detección más rápida y eficiente.

Un enfoque común es la implementación de YARA en entornos de análisis dinámico, donde se ejecutan archivos sospechosos en máquinas virtuales para observar su comportamiento. Si una muestra de malware intenta establecer comunicación con servidores externos o modificar archivos del sistema, las reglas YARA pueden detectar estas acciones y activar alertas para los equipos de seguridad. Este tipo de integración es fundamental en sistemas de detección y respuesta extendida (XDR), que combinan múltiples fuentes de datos para identificar ataques de manera temprana (Sebastián et al., 2016).

Además, YARA se ha incorporado en plataformas de análisis de malware como VirusTotal e Intezer, lo que permite a los analistas de seguridad comparar muestras con bases de datos globales y obtener información detallada sobre posibles amenazas. Gracias a estas integraciones, la detección y clasificación de malware se ha vuelto más accesible y precisa (Kharraz et al., 2015).

## Desafíos en la Implementación de Firmas YARA

Si bien YARA es una herramienta poderosa, su efectividad depende de varios factores:

Calidad y actualización de las reglas: las firmas deben mantenerse actualizadas para detectar nuevas variantes de malware.

Falsos positivos y negativos: un equilibrio adecuado en la precisión de las reglas es fundamental para evitar bloqueos innecesarios o la omisión de amenazas reales.

Optimización del rendimiento: el análisis de grandes volúmenes de datos puede ralentizar el sistema, por lo que es necesario optimizar los procesos de búsqueda y detección (Bilge & Dumitras, 2012) .

Para superar estos desafíos, se recomienda la colaboración con comunidades de ciberseguridad y el uso de plataformas de intercambio de información sobre amenazas emergentes (Hutchins, Cloppert & Amin, 2011) .

## Importancia del Intercambio de Información sobre Ciber-amenazas

El intercambio de inteligencia de amenazas se ha convertido en una estrategia clave para fortalecer la seguridad digital. La colaboración entre organizaciones permite detectar patrones de ataque de manera más eficiente y reducir los tiempos de respuesta ante incidentes (Shackleford, 2021) .

Estándares como STIX (Structured Threat Information eXpression) y TAXII (Trusted Automated eXchange of Indicator Information) facilitan la estructuración y el intercambio de indicadores de

compromiso (IoC). Estos protocolos mejoran la detección y mitigación de amenazas en tiempo real (Barnum, 2014) .

La integración de YARA en una plataforma web con capacidad de intercambio de datos no solo fortalecería la seguridad colectiva, sino que también promovería la adopción de estrategias proactivas de ciberseguridad (Sauerwein et al., 2017) .

## YARA y su Aplicación en el Análisis Forense Digital

El análisis forense digital es una disciplina dentro de la ciberseguridad que se enfoca en la recopilación, preservación y análisis de evidencia digital con el propósito de investigar incidentes de seguridad. La importancia de este campo ha crecido exponencialmente debido al aumento en la frecuencia y sofisticación de los ciberataques. En este contexto, YARA se ha convertido en una herramienta esencial para los analistas forenses, ya que permite detectar patrones maliciosos en archivos, procesos en memoria e imágenes de disco sin necesidad de ejecutar el código sospechoso (Ligh et al., 2014).

Uno de los usos más frecuentes de YARA en análisis forense es la identificación de Indicadores de Compromiso (IoC) en sistemas comprometidos. Estos indicadores pueden incluir hashes de archivos maliciosos, direcciones IP sospechosas, nombres de dominio utilizados para el comando y control (C2) y patrones de código malicioso. Al aplicar reglas YARA a la evidencia digital, los investigadores pueden encontrar rápidamente rastros de malware, rootkits o herramientas de ataque utilizadas por los ciberdelincuentes (Kharraz et al., 2015).

Un aspecto clave del análisis forense con YARA es su capacidad para realizar escaneos en memoria (Memory Forensics). Muchas familias de malware avanzadas, como los fileless

malware, no escriben archivos en el disco y residen completamente en la memoria RAM para evitar ser detectadas por soluciones de seguridad tradicionales. En estos casos, herramientas como Volatility o Rekall permiten extraer imágenes de memoria de sistemas comprometidos, sobre las cuales se pueden ejecutar reglas YARA para identificar patrones maliciosos que no serían detectados por métodos convencionales (Bilge & Dumitras, 2012).

Además, YARA es utilizada en el análisis de archivos maliciosos dentro de entornos de sandboxing. En estas plataformas, los archivos sospechosos se ejecutan en un entorno aislado para monitorear su comportamiento. Aplicando reglas YARA a los registros generados por estos análisis, es posible detectar intentos de evasión, modificaciones en el sistema o conexiones a servidores maliciosos. Esta técnica es particularmente útil para analizar malware polimórfico y metamórfico, que cambia su código en cada ejecución para evitar ser detectado por firmas estáticas (Sebastián et al., 2016).

Otro uso importante de YARA en el análisis forense es la detección de ataques persistentes avanzados (APT). Los actores de amenazas que llevan a cabo estos ataques suelen utilizar herramientas personalizadas y técnicas sofisticadas para infiltrarse en redes y mantener acceso a largo plazo sin ser detectados. Al desarrollar reglas YARA específicas basadas en el código, comportamiento y artefactos de ataques previos, los analistas pueden identificar nuevas variantes de amenazas relacionadas con actores conocidos. Esta metodología ha sido ampliamente utilizada en investigaciones de ciber espionaje y ataques dirigidos contra infraestructuras críticas (Hutchins, Cloppert & Amin, 2011).

Finalmente, la integración de YARA con sistemas de inteligencia de amenazas ha permitido una detección más rápida y precisa de nuevas variantes de malware en investigaciones forenses. Plataformas como VirusTotal, Intezer y Threat Intelligence Platforms (TIPs) permiten compartir

y correlacionar reglas YARA con bases de datos globales de amenazas, lo que facilita la identificación de malware emergente y mejora la respuesta a incidentes de seguridad (Barnum, 2014).

En conclusión, la aplicación de YARA en el análisis forense digital ha mejorado significativamente la capacidad de detección y atribución de ciberataques. Su versatilidad para analizar archivos, memoria y tráfico de red la convierte en una herramienta esencial para investigadores de seguridad y equipos de respuesta a incidentes. A medida que las amenazas evolucionan, la mejora continua de reglas YARA y su integración con nuevas tecnologías serán clave para enfrentar los desafíos del cibercrimen en el futuro (Singer & Friedman, 2014).

## Introducción a la Ingeniería de Software y Arquitectura

La ingeniería de software es una disciplina que abarca el diseño, desarrollo, mantenimiento y gestión de sistemas de software de manera eficiente y escalable. Su objetivo principal es la creación de sistemas confiables y robustos que puedan adaptarse a los cambios tecnológicos y a las necesidades de los usuarios. Para lograrlo, se emplean metodologías, estándares y herramientas que permiten optimizar el proceso de desarrollo y reducir la complejidad de los sistemas.

En el contexto de sistemas de detección de amenazas como el desarrollado en este proyecto, la correcta selección de patrones arquitectónicos y buenas prácticas de programación es fundamental para garantizar rendimiento, seguridad y escalabilidad. Una arquitectura de software

bien definida facilita la mantenibilidad y permite una evolución progresiva del sistema, adaptándose a nuevas amenazas y tecnologías emergentes en ciberseguridad.

## Arquitecturas de Software

En el diseño de aplicaciones web, la arquitectura del software juega un papel crucial en el rendimiento, escalabilidad y facilidad de mantenimiento del sistema. Existen diversos patrones arquitectónicos, cada uno con características y ventajas específicas. A continuación, se describen algunos de los más relevantes:

- **Arquitectura Monolítica:** Todas las funcionalidades de la aplicación se encuentran en un solo bloque de software. Es simple de desarrollar y desplegar, pero puede dificultar la escalabilidad y mantenimiento a medida que la aplicación crece en tamaño y complejidad.
- **Arquitectura en Capas:** Separa la aplicación en diferentes niveles lógicos (presentación, negocio y datos), permitiendo mayor modularidad y mantenibilidad. Cada capa tiene una responsabilidad específica, lo que facilita la reutilización de componentes y la reducción de acoplamientos innecesarios entre módulos.
- **Arquitectura Basada en Servicios (SOA):** Divide la aplicación en servicios independientes que se comunican entre sí mediante protocolos estándar como HTTP o mensajes XML/JSON. Esto facilita la interoperabilidad y escalabilidad, permitiendo que diferentes partes del sistema evolucionen de manera independiente.

- **Microservicios:** Modelo distribuido donde cada módulo del sistema es un servicio independiente con su propia base de datos y lógica de negocio. Este enfoque mejora la escalabilidad, flexibilidad y resiliencia del sistema, permitiendo la implementación de nuevos módulos sin afectar el funcionamiento general de la aplicación.

Para este proyecto, se adopta la **arquitectura en capas**, ya que permite una organización clara del sistema y facilita su mantenimiento y escalabilidad. Esta arquitectura segmenta la aplicación en diferentes capas:

- **Capa de Presentación:** Interfaz de usuario que permite la interacción con el sistema, generalmente implementada con tecnologías como HTML, CSS y JavaScript.
- **Capa de Aplicación o Negocio:** Contiene la lógica de negocio y el procesamiento de datos. Es responsable de gestionar las reglas del sistema y coordinar la comunicación entre la capa de presentación y la base de datos.
- **Capa de Datos:** Se encarga de la gestión y almacenamiento de información en una base de datos relacional o NoSQL, garantizando la persistencia y recuperación eficiente de los datos.

La separación en capas favorece la modularidad del software y facilita la implementación de cambios sin afectar la estructura global del sistema. Además, permite la integración de nuevos servicios o tecnologías sin comprometer la estabilidad del sistema.

**Modelo Vista-Controlador (MVC)**

El Modelo Vista-Controlador (MVC) es un patrón de diseño ampliamente utilizado en el desarrollo de aplicaciones web. Se basa en la separación de preocupaciones para mejorar la organización del código y la mantenibilidad del sistema. En el contexto del aplicativo web basado en Django, el MVC se descompone de la siguiente manera:

- **Modelo (Model):** Representa los datos y la lógica de negocio de la aplicación. En Django, se implementa mediante los modelos de base de datos utilizando su ORM (Object-Relational Mapping), lo que permite interactuar con bases de datos sin escribir consultas SQL directamente.
- **Vista (View):** Gestiona la presentación de la información al usuario. En Django, las vistas se encargan de procesar solicitudes HTTP y devolver respuestas adecuadas en forma de HTML, JSON u otros formatos.
- **Controlador (Controller):** Maneja la interacción entre el modelo y la vista. Django adopta el patrón **MTV (Modelo-Template-Vista)**, donde las vistas actúan como controladores al recibir solicitudes y gestionar la lógica de negocio, delegando la presentación en plantillas (Templates).

El uso del modelo MVC/MTV en Django permite una separación clara de responsabilidades, facilitando la escalabilidad y mantenibilidad del sistema. Además, el patrón MVC favorece la reutilización de componentes y la optimización del código, permitiendo una mayor eficiencia en el desarrollo y asegurando una mejor experiencia de usuario.

# Análisis de Restricciones

El desarrollo del aplicativo web basado en firmas YARA para la detección de amenazas cibernéticas enfrenta diversas restricciones que deben ser consideradas para asegurar su viabilidad y éxito. A continuación, se presentan las principales restricciones identificadas:

## Restricciones Ambientales

La implementación del aplicativo web no implica un impacto ambiental directo significativo, dado que se trata de un desarrollo digital. Sin embargo, el consumo de recursos energéticos asociados al funcionamiento de servidores y al almacenamiento de datos en la nube puede representar una carga ambiental si no se gestionan adecuadamente. Es fundamental considerar el uso de servidores eficientes y políticas de sostenibilidad en los centros de datos.

## Restricciones Económicas

El presupuesto para el desarrollo de este aplicativo es limitado, lo que impone restricciones en cuanto a la infraestructura tecnológica y los recursos humanos disponibles. Además, la gratuidad y accesibilidad del aplicativo son objetivos centrales del proyecto, lo que limita el uso de tecnologías costosas o licencias propietarias.

## Restricciones Legales

El desarrollo y uso del aplicativo deben cumplir con las normativas vigentes de protección de datos personales (Ley 1581 de 2012 en Colombia) y ciberseguridad. Además, al manejar potencialmente información sensible durante el análisis de archivos, se deben implementar medidas estrictas de seguridad y privacidad que cumplan con los estándares internacionales, como el RGPD de la Unión Europea.

### Restricciones de Salud y Seguridad

Aunque el proyecto no maneja sustancias peligrosas ni riesgos físicos directos, es necesario garantizar un entorno de desarrollo seguro para los programadores y colaboradores, especialmente considerando que la carga de trabajo mental puede generar estrés. Además, el aplicativo debe ser seguro frente a vulnerabilidades que puedan comprometer la seguridad de los usuarios.

### Restricciones Socioculturales

La aceptación del aplicativo depende de su adaptabilidad a las necesidades y conocimientos de los usuarios objetivo, que incluyen tanto profesionales de TI como usuarios no técnicos. Las barreras culturales y educativas en Colombia, donde el acceso a formación especializada en ciberseguridad puede ser limitado, imponen el reto de diseñar una interfaz intuitiva y de fácil uso.

## 1. Identificación y Evaluación de Soluciones

### *1.1 Soluciones Ilógicas*

Se analizarán las posibles soluciones descartando aquellas que sean inviables o que violen principios fundamentales de la informática y la ciberseguridad. En este contexto, se evitarán enfoques que comprometan el rendimiento del sistema o que sean incompatibles con las restricciones de hardware y software establecidas.

### *1.2 Comparación con Hechos Conocidos y alternativas de solución*

El diseño y desarrollo del aplicativo considerará experiencias previas en herramientas similares de detección de amenazas. Se analizarán estudios de caso, soluciones empresariales y proyectos de código abierto para evaluar la efectividad de distintos enfoques y determinar si la solución propuesta es más eficiente o innovadora en comparación con las ya existentes, información que estará organizada en el siguiente artículo.

Dado que evaluar todas las soluciones posibles sería costoso en términos de tiempo y recursos, se realizará una selección más centralizada hacia el enfoque del proyecto. Las soluciones menos favorables serán descartadas tempranamente para optimizar recursos.

## 2. Selección de la Solución

La selección final de la solución se basará en la optimización de la seguridad de los usuarios primeramente, el rendimiento del tiempo a la hora de escanear archivos y de la misma manera la usabilidad y escalabilidad de la plataforma web.

- Impacto en la Seguridad: Se medirá la capacidad del sistema para detectar amenazas con un índice de precisión superior al 90% en comparación con herramientas existentes.
- Usabilidad y Accesibilidad: Se garantizará que la interfaz sea intuitiva y accesible para usuarios con distintos niveles de experiencia en ciberseguridad.

### 3. Desarrollo del Aplicativo Web

Se procederá al diseño e implementación del aplicativo siguiendo metodologías ágiles (como scrum), permitiendo iteraciones rápidas y mejora continua durante su construcción y desarrollo.

*Fases del Desarrollo:*

1. Diseño de Arquitectura: Definir la estructura del sistema, integración con bases de datos y compatibilidad con firmas YARA.
2. Implementación del Módulo de Análisis: Desarrollo del motor de detección basado en YARA, optimizando la velocidad de escaneo y consumo de recursos.
3. Desarrollo de la Interfaz Web: Creación de una interfaz intuitiva para la carga de archivos, gestión de reglas y visualización de resultados.
4. Pruebas y Validación: Evaluación del desempeño, tasa de detección y usabilidad mediante pruebas con usuarios y comparación con herramientas existentes.

Siendo así la metodología propuesta, esta permite una selección racional de soluciones y un desarrollo estructurado del aplicativo web, asegurando que el mismo cumpla con los objetivos establecidos en cuanto a detección de amenazas, eficiencia y facilidad de uso. Este enfoque garantiza una herramienta efectiva, competitiva y accesible para mejorar y normalizar la seguridad cibernética.

| HISTORIA DE USUARIO | | |
|---|---|---|
| **Identificador** | **Nombre** | |
| HU001 | Analizador de archivos | |
| **Puntos de historia estimados** | | |
| **Prioridad** | Alta | |
| **Responsable** | Product Manager (Sebastián) | |
| **Descripción** | | |
| Como usuario quiero analizar y diagnosticar que tan seguro es un archivo para evitar ser robos de información o ataques cibernéticos comunes | | |
| **Numero de criterio** | **Criterio de aceptación** | **Atributo de calidad** |
| 1 | Acceso web | Funcionalidad |
| 2 | Acepte diferentes formatos de archivo | Funcionalidad |
| 3 | Que el sistema esté funcionando las 24 horas del día | Disponibilidad |
| 4 | Sea intuitivo de usar | Funcionalidad |

| HISTORIA DE USUARIO | | |
|---|---|---|
| **Identificador** | **Nombre** | |
| HU002 | Sistema de Cuentas | |
| **Puntos de historia estimados** | | |
| **Prioridad** | Media | |
| **Responsable** | Desarrollador frontend y backend (Julián) | |
| **Descripción** | | |
| Como usuario quiero poder manejar una cuenta para tener un workspace y una experiencia de usuario orientada mediante mi interacción con la web | | |
| **Numero de criterio** | **Criterio de aceptación** | **Atributo de calidad** |
| 1 | Base de datos conectada | Funcionalidad |
| 2 | Que el sistema esté funcionando las 24 horas del día | Disponibilidad |
| 3 | Usuario (correo) y contraseña (8 o más caracteres, MAY, min y números) con Sistema de recuperación por correo | Funcionalidad |
| 4 | Sistema de recuperación de contraseña con correo | Funcionalidad |

| HISTORIA DE USUARIO | | |
|---|---|---|
| **Identificador** | **Nombre** | |
| HU003 | Historial de archivos | |
| **Puntos de historia estimados** | | |
| **Prioridad** | Alta | |
| **Responsable** | Desarrolladores | |
| **Descripción** | | |
| Como usuario quiero tener un historial de archivos analizados para poder acceder a los resultados e incluso poder ver una estadística tanto personal como global de seguridad | | |
| **Número de criterio** | **Criterio de aceptación** | **Atributo de calidad** |
| 1 | Base de datos conectada | Funcionalidad |
| 2 | Ventana para acceder a los archivos subidos en la web | Funcionalidad |
| 3 | Diagnóstico de seguridad personal y global | Funcionalidad |
| 4 | Apartado con información global de seguridad | Funcionalidad |

| HISTORIA DE USUARIO | | |
|---|---|---|
| **Identificador** | **Nombre** | |
| HU004 | Criterios de análisis | |
| **Puntos de historia estimados** | | |
| **Prioridad** | Baja | |
| **Responsable** | Product Manager | |
| **Descripción** | | |
| Como usuario quiero cambiar la rigurosidad y los enfoques con los que las firmas analizan mis archivos para personalizar el tipo de diagnóstico que necesito según mi contexto | | |
| **Numero de criterio** | **Criterio de aceptación** | **Atributo de calidad** |
| 1 | Banco de firmas o políticas a aplicar | Funcionalidad |
| 2 | Diseño intuitivo y fácil para personas no técnicas | Funcionalidad |

| HISTORIA DE USUARIO | | |
|---|---|---|
| **Identificador** | **Nombre** | |
| HU005 | Notificaciones | |
| **Puntos de historia estimados** | | |
| **Prioridad** | Media | |
| **Responsable** | Desarrolladores | |
| **Descripción** | | |
| Como usuario quiero que se me notifique cualquier actividad o reporte para estar al tanto de la seguridad de mi información | | |
| **Numero de criterio** | **Criterio de aceptación** | **Atributo de calidad** |
| 1 | Base de datos conectada | Funcionalidad |
| 2 | Ventana con PopUps que sucedan y se muestren mientras hayan notificiones | Funcionalidad |
| 3 | Que las notificaciones esten funcionando las 24 horas del día | Disponibilidad |

| HISTORIA DE USUARIO | | |
|---|---|---|
| **Identificador** | **Nombre** | |
| HU006 | Diseño Responsive | |
| **Puntos de historia estimados** | | |
| **Prioridad** | Alta | |
| **Responsable** | Desarrollador Frontend | |
| **Descripción** | | |
| Como usuario quiero abrir la web en otros dispositivos como mi móvil o la tableta para no necesitar un computador a la hora de analizar mis archivos y que este tenga un diseño orientado al dispositivo | | |
| **Numero de criterio** | **Criterio de aceptación** | **Atributo de calidad** |
| 1 | Diseño orientado a PC | Funcionalidad |
| 2 | Diseño orientado a móvil | Funcionalidad |
| 3 | Diseño orientado a tableta | Funcionalidad |

| HISTORIA DE USUARIO | | |
|---|---|---|
| **Identificador** | **Nombre** | |
| HU007 | Base de datos | |
| **Puntos de historia estimados** | | |
| **Prioridad** | Alta | |
| **Responsable** | Desarrollador backend | |
| **Descripción** | | |
| Como administrador quiero una base de datos para almacenar toda la información de los usuarios y de los banners de la pagina | | |
| **Numero de criterio** | **Criterio de aceptación** | **Atributo de calidad** |
| 1 | Tablas orientadas a objetos | Funcionalidad |
| 2 | Llaves primarias y foraneas | Funcionalidad |
| 3 | Conexión con en LogIn y la pagina en general | Funcionalidad |

| HISTORIA DE USUARIO | | |
|---|---|---|
| **Identificador** | **Nombre** | |
| HU008 | Super Administrador | |
| **Puntos de historia estimados** | | |
| **Prioridad** | Alta | |
| **Responsable** | Desarrollador Frontend | |
| **Descripción** | | |
| Como administrador quiero unas credenciales especiales para acceder a un CRUD que me permita crear, leer, editar, y eliminar información según se determine al interior de la empresa | | |
| **Numero de criterio** | **Criterio de aceptación** | **Atributo de calidad** |
| 1 | Revisar y crear usuarios o información necesaria | Funcionalidad |
| 2 | Enseñar las contraseñas de los usuarios de manera encriptada | Funcionalidad |
| 3 | Acceso inmediato a un portal distinto con las credenciales “CRUD” | Funcionalidad |

## 11. Análisis de Costos

En todo proyecto de ingeniería, es fundamental analizar los costos asociados a su desarrollo y operación para garantizar su viabilidad económica y sostenibilidad a largo plazo. En este caso, la plataforma web para el análisis de archivos y detección de amenazas mediante firmas YARA ha sido diseñada con un enfoque académico, priorizando el uso de herramientas gratuitas y minimizando los costos financieros. A continuación, se presenta un desglose detallado de los costos involucrados.

### 11.1 Costos en los que se incurre para hacer el producto o brindar el servicio

#### *11.1.1 Costos Directos:*

Estos costos están relacionados con los recursos esenciales para la implementación de la plataforma:

- **Infraestructura tecnológica:** Uso de servidores gratuitos o de bajo costo en la nube, como Render para la implementación de la plataforma y base de datos.
- **Software y licencias:** Uso de herramientas de código abierto y gratuitas como FastAPI para el backend, SQLAlchemy para la gestión de la base de datos, PostgreSQL como sistema de almacenamiento, y frameworks de desarrollo web para la interfaz de usuario.
- **Servicios de conectividad:** Gastos asociados al uso de internet personal y recursos en la nube para pruebas y despliegue.

- **Mano de obra:** Desarrollo y programación llevados a cabo por los integrantes del equipo sin costos salariales, asumiendo tareas en backend, frontend, integración de firmas YARA y gestión de bases de datos. Juicio de expertos.

### *11.1.2 Costos Fijos:*

Son aquellos costos que se mantienen constantes independientemente del nivel de producción o uso:

- **Espacio de trabajo:** Uso de instalaciones universitarias y trabajo remoto para minimizar costos operativos.
- **Servicios públicos:** Energía e internet asumidos por los estudiantes.
- **Cumplimiento normativo:** Aplicación de buenas prácticas en ciberseguridad sin necesidad de inversión adicional en esta etapa del proyecto.

### *11.1.3 Gastos Generales (Overhead):*

Incluyen aspectos administrativos y operacionales del proyecto:

- **Gestión del proyecto:** Uso de herramientas gratuitas como Trello y Notion para la organización y planificación del trabajo.
- **Publicidad y difusión:** Creación de documentación técnica, guías de usuario y posibles estrategias de divulgación en plataformas gratuitas.
- **Soporte y documentación:** Acceso a foros, comunidades de desarrolladores y recursos en línea para resolver problemas técnicos sin costos adicionales.

### *11.1.4 Costos de Inversión*

Para la implementación del sistema, se requiere una inversión mínima en los siguientes aspectos:

**Costos Directos:**

- **Desarrollo del software:** Inversión en tiempo y esfuerzo del equipo para la implementación y optimización de la plataforma.
- **Dominio y hosting:** Posible compra de un dominio personalizado (aproximadamente $100 mil COP anuales) si se busca una versión pública y profesional del proyecto con el tiempo.
- **Equipos de desarrollo:** Uso de computadoras personales ya disponibles, evitando la necesidad de adquirir nuevos dispositivos.

### *11.1.5 Costos Indirectos:*

- **Permisos y licencias:** No aplicable en esta fase.
- **Gastos administrativos:** Aún no existen costos asociados debido a la naturaleza educativa del proyecto.
- **Contingencias e imprevistos:** Posible inversión en servidores con mayor capacidad si el proyecto evoluciona a una versión de producción con mayor demanda.

#### *11.1.6 Capital de Trabajo:*

- **Presupuesto para mantenimiento y actualizaciones:** Uso de plataformas gratuitas en la fase inicial con posibilidad de migración a versiones pagas si el proyecto se expande.
- **Fondo de respaldo:** No se requiere en esta etapa, pero podría considerarse en una futura escalabilidad del proyecto.

### 11.2 Evaluación de Rentabilidad

Si bien el proyecto se desarrolla en un contexto académico, se han evaluado posibles estrategias para su monetización en caso de expandirse a un entorno comercial. Entre estas estrategias se encuentran:

- Modelos de monetización: Implementación de servicios freemium, consultorías en ciberseguridad, integración con otras plataformas o venta de reglas YARA personalizadas.
- Costo-beneficio: Análisis del impacto económico de una posible escalabilidad del proyecto considerando costos de infraestructura y mantenimiento.
- Escalabilidad: Evaluación de modelos en la nube que permitan ofrecer el servicio a terceros de manera rentable.

Con este análisis de costos, se garantiza que el desarrollo de la plataforma sea viable dentro del contexto académico, optimizando recursos gratuitos y sentando las bases para una posible expansión en el futuro.

### 11.3 Conclusión de costos

Aunque se utilizó software libre y servicios gratuitos para minimizar los costos, el desarrollo del proyecto implicó una inversión importante en tiempo y trabajo técnico. Estimando la dedicación de cada integrante (alrededor de 180 horas por persona durante el semestre) y considerando un valor promedio de $20.000 COP por hora para trabajo de desarrollo, el costo estimado de la mano de obra asciende a aproximadamente $10.800.000 COP.

Sumando este valor al posible costo de dominio y hosting básico ($100.000 COP anuales), se estima que el costo total del proyecto, en su versión actual, ronda los $11 millones de pesos colombianos.

Esta cifra permite dimensionar el esfuerzo invertido y constituye una base realista para evaluar la viabilidad del proyecto en contextos productivos.

## 12. **Product Backlog**

El Product Backlog es una lista priorizada de funcionalidades que el sistema debe implementar, basada en las necesidades del usuario y los objetivos del proyecto. Esta lista está compuesta por historias de usuario que describen, desde la perspectiva del usuario final, qué desea hacer con el sistema, con el fin de lograr un determinado valor. A continuación, se detallan las historias de usuario identificadas para el desarrollo del aplicativo web basado en firmas YARA.

| ID | Historia de Usuario | Rol | ¿Qué necesita? | ¿Para qué? | Criterios de aceptación |
|---|---|---|---|---|---|
| HU1 | Crear reglas YARA | Admin | Ingresar nuevas firmas YARA en el sistema | Para detectar patrones maliciosos en archivos | La firma debe guardarse y ser visible en la base de datos |
| HU2 | Escanear archivos | Usuario | Subir archivos a través de la interfaz web | Para que el sistema determine si contienen amenazas | El sistema debe devolver un resultado con la evaluación YARA |
| HU3 | Consultar historial | Usuario | Ver los análisis realizados previamente | Para revisar amenazas encontradas en el pasado | Se debe mostrar una lista con fechas y resultados previos |
| HU4 | Autenticación | Usuario | Iniciar sesión o registrarse | Para acceder al sistema y ver funcionalidades protegidas | Debe poder iniciar sesión y ver su información |
| HU5 | Gestión de usuarios | Admin | Ver, editar o eliminar usuarios registrados | Para mantener control sobre el acceso al sistema | Las acciones deben estar disponibles solo para administradores |
| HU6 | Editar reglas YARA | Admin | Modificar firmas previamente almacenadas | Para actualizar reglas frente a nuevas amenazas | La regla modificada debe reemplazar a la anterior |

### 12.2 Diagrama de arquitectura del sistema

La arquitectura del sistema sigue un enfoque modular y por capas, permitiendo una clara separación entre la interfaz de usuario, la lógica del negocio y el acceso a datos. La aplicación está compuesta por tres componentes principales:

*Frontend (Cliente web):*

Desarrollado en HTML, CSS y JavaScript, este módulo permite la interacción del usuario con la plataforma. Desde aquí se pueden cargar archivos, crear reglas YARA y visualizar resultados. Todas las acciones generan peticiones HTTP hacia el backend.

*Backend (API REST con FastAPI):*

Es el núcleo del sistema. Gestiona autenticación, creación de reglas, procesamiento de archivos y ejecución del motor YARA. Se implementó en Python usando FastAPI, incluyendo control de sesiones con JWT, manejo de errores, validación de datos y enrutamiento claro por módulos.

*Base de datos (PostgreSQL + SQLAlchemy):*

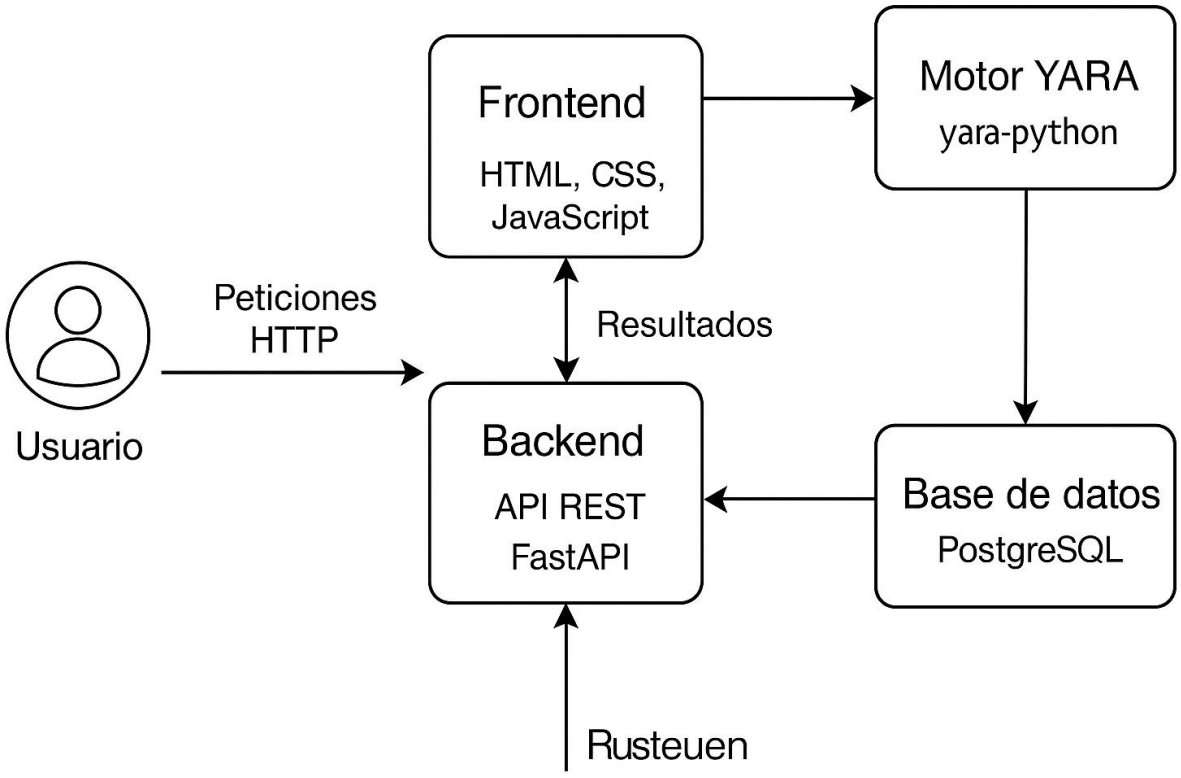
Contiene la información persistente: usuarios, reglas, archivos y resultados. La conexión y manipulación se hace mediante SQLAlchemy como ORM, permitiendo operaciones CRUD y relaciones entre entidades de manera segura y eficiente.

*Motor YARA:*

Integrado directamente en el backend usando la librería yara-python, permite analizar archivos cargados por los usuarios en tiempo real, utilizando las reglas almacenadas en la base de datos.

*Resumen del flujo funcional:*

El usuario accede a través del navegador → envía peticiones al backend vía formularios y botones → el backend procesa la lógica → ejecuta el análisis con YARA si corresponde → y finalmente guarda y devuelve los resultados al usuario desde la base de datos.



**Descripción de casos uso:**

En este apartado se mostrará una tabla, con la descripción de los casos expuestos en el punto anterior, con la intención de mejorar la comprensión.

| Caso de uso | Actor | Breve descripción |
|---|---|---|
| Registrarse | Usuario | El usuario crea una cuenta para acceder al sistema. |
| Iniciar sesión | Usuario/Admin | El actor ingresa al sistema con sus credenciales. |
| Subir archivo | Usuario | El usuario sube un archivo para ser analizado con las reglas YARA. |
| Consultar historial | Usuario | El usuario puede ver los resultados anteriores de análisis. |
| Crear/editar/eliminar reglas | Administrador | El administrador puede gestionar las firmas YARA almacenadas en la base de datos. |
| Ver y administrar usuarios | Administrador | El administrador puede ver, editar o eliminar cuentas de usuarios registrados. |

***12.3*** Arquitectura de la solución

La arquitectura del aplicativo web se ha diseñado siguiendo el modelo C4, que permite representar de forma estructurada los diferentes niveles de abstracción del sistema. A continuación, se presentan los tres diagramas fundamentales (Contexto, Contenedores y Componentes) que

describen la solución implementada, facilitando la comprensión de su estructura interna y sus interacciones externas.

### 12.3.1 **Diagrama de Contexto (Nivel 1 - C4)**

Este diagrama representa una vista de alto nivel del sistema, identificando los actores externos (usuarios y administradores) y su interacción con el aplicativo web. También se muestran los sistemas de soporte, como la base de datos y el repositorio de reglas YARA, que complementan el funcionamiento del sistema.

**Elementos clave:**

- **Usuario final:** Interactúa a través del navegador web para subir archivos, consultar resultados y gestionar su cuenta.
- **Administrador:** Accede a funciones especiales como el CRUD de reglas YARA y usuarios.
- **Aplicativo Web de Detección YARA (Sistema principal):** Plataforma central del sistema.
- **Repositorio YARA (opcional):** Fuente de reglas externas (manual o automatizado en versiones futuras).

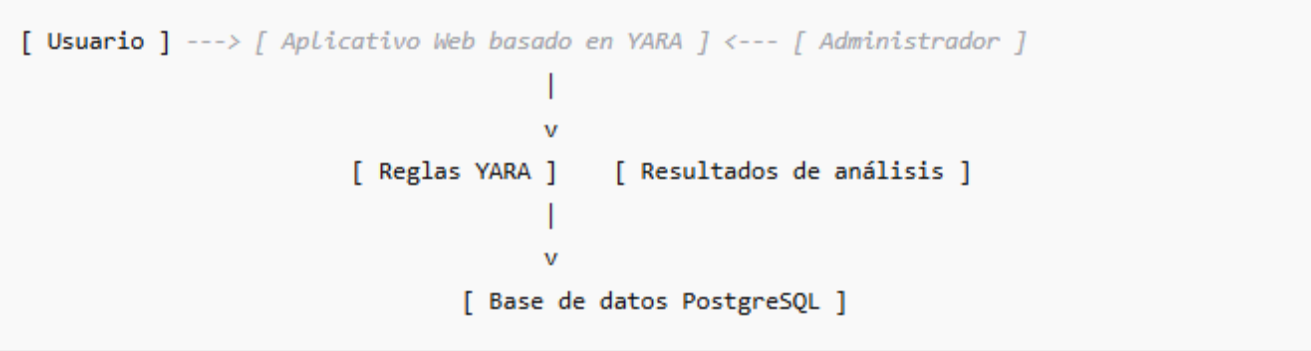



imagen 2. Diagrama de contexto

12.3.2 Diagrama de Contenedores

En este nivel, se detallan los principales contenedores que conforman el sistema, es decir, las aplicaciones o servicios ejecutables que interactúan entre sí. Se incluye la separación entre el frontend (interfaz de usuario), el backend (API y lógica de negocio) y los servicios auxiliares como el motor de análisis YARA y la base de datos. Este diagrama permite visualizar cómo los distintos módulos tecnológicos se comunican dentro de la solución.

**Contenedores:**

- **Frontend (HTML + JS + CSS):** UI web donde el usuario interactúa.
- **Backend (FastAPI):** Maneja la lógica de negocio, autenticación, análisis con YARA, etc.
- **YARA Engine:** Componente interno que procesa los archivos usando reglas YARA.
- **Base de datos PostgreSQL:** Almacena usuarios, reglas YARA y resultados de análisis.

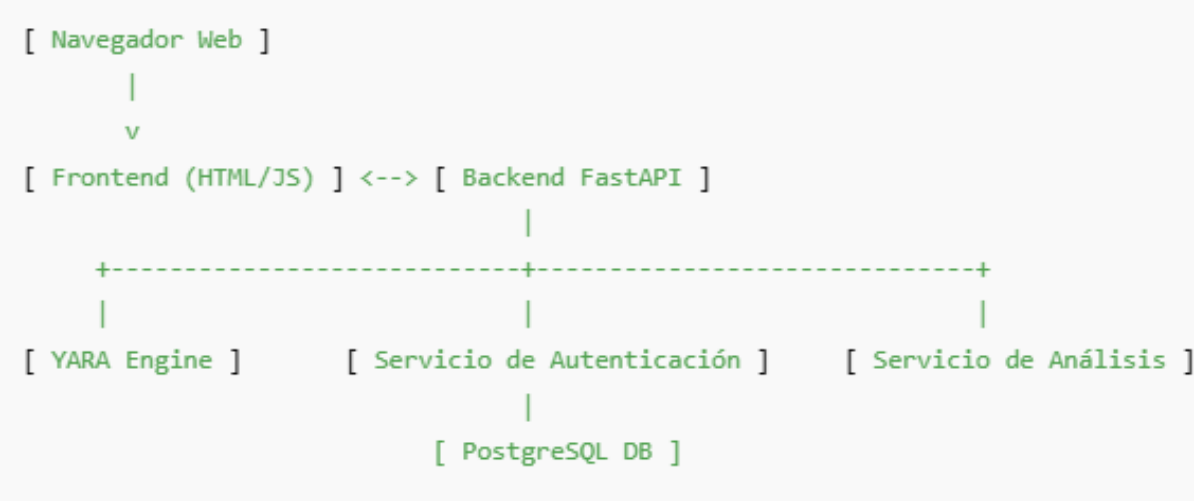



imagen 3. Diagrama de contenedores

### 12.3.3 **Diagrama de Componentes**

Este diagrama muestra el desglose interno del backend desarrollado con FastAPI, identificando los componentes de software que implementan la funcionalidad del sistema. Se incluyen las APIs específicas (autenticación, reglas YARA, análisis de archivos), el controlador del motor YARA, y el ORM encargado de la interacción con la base de datos. Este nivel de detalle es útil para comprender cómo se organizan y relacionan los módulos de código dentro del contenedor de backend.

**Componentes Backend:**

- **API de autenticación:** Registro, login y gestión de sesiones.
- **API de archivos:** Subida, escaneo y diagnóstico.
- **API de reglas YARA:** Crear, editar y aplicar reglas.
- **Controlador de análisis:** Invoca al motor YARA y recibe los resultados.

- **ORM (SQLAlchemy):** Conecta los modelos Python con PostgreSQL.


imagen 3. diagrama de componentes

### 12.4 Modelo de datos

El modelo de datos del sistema fue diseñado con enfoque relacional, buscando una estructura clara, escalable y fácilmente consultable. Para ello, se implementó una base de datos PostgreSQL, gestionada mediante el ORM SQLAlchemy.

El sistema se compone de cinco entidades principales que reflejan la lógica de funcionamiento de la plataforma:

1. **usuarios**: almacena la información de autenticación y rol de cada usuario registrado (normal o administrador).
2. **reglas_yara**: contiene las firmas YARA creadas o registradas en el sistema, con su contenido, fecha de creación y autor.
3. **archivos**: representa los archivos cargados por los usuarios para ser analizados.

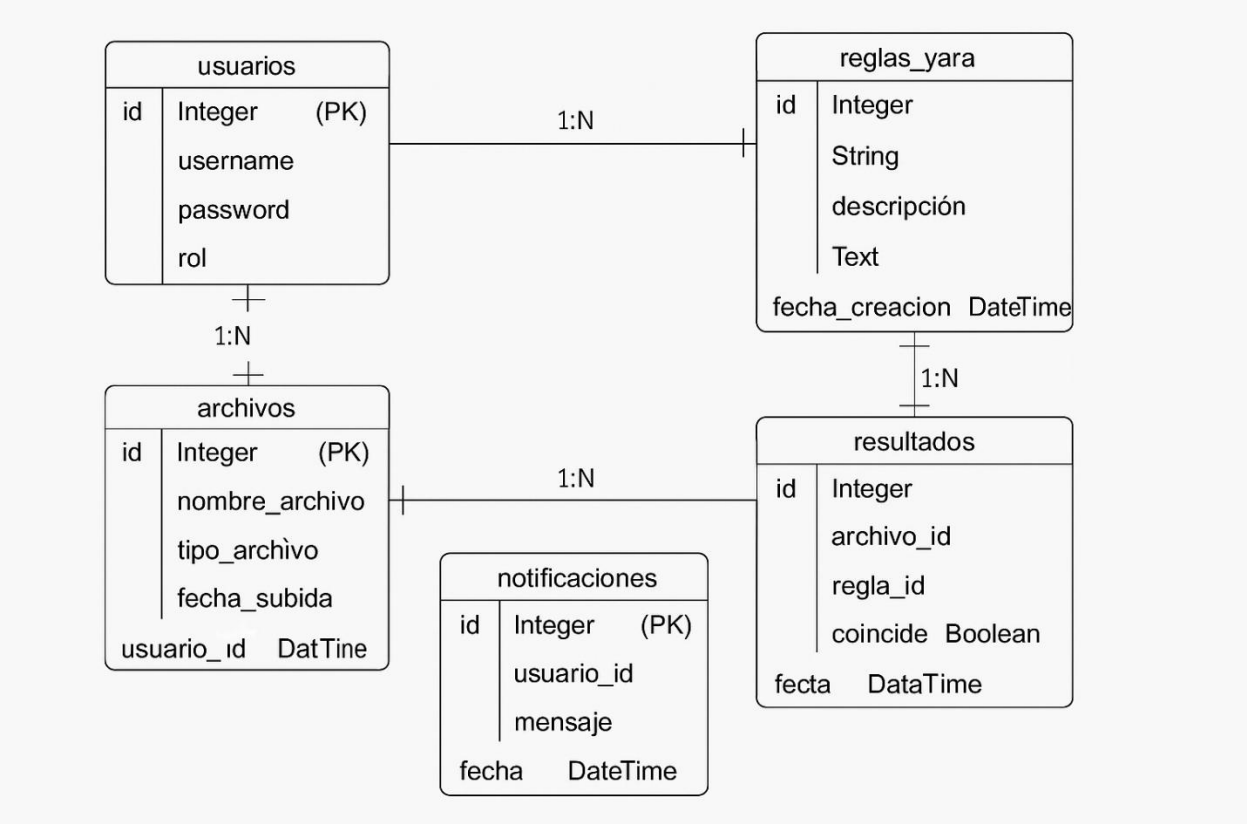
4. **resultados**: almacena los resultados del análisis de cada archivo con respecto a cada regla YARA.

5. **notificaciones**: gestiona los mensajes automáticos que recibe el usuario sobre sus análisis o actividades.

   Cada entidad está normalizada y tiene relaciones 1:N bien definidas, por ejemplo:

- Un usuario puede subir múltiples archivos → relación 1:N entre usuarios y archivos
- Un archivo puede tener múltiples resultados → relación 1:N entre archivos y resultados
- Una regla YARA puede aplicarse en múltiples resultados → relación 1:N entre reglas_yara y resultados

A continuación se presenta el diagrama entidad-relación del sistema



Este diagrama permite visualizar gráficamente las entidades principales del sistema, sus atributos y las relaciones entre ellas, facilitando la comprensión del modelo implementado en la base de datos PostgreSQL.

- Un **usuario** puede subir múltiples **archivos**.
- Cada **archivo** puede generar múltiples **resultados**, uno por cada regla aplicada.
- Una **regla YARA** puede estar asociada a múltiples **resultados**.
- Un **usuario** puede recibir múltiples **notificaciones**.

DIAGRAMA ENTIDAD RELACION...

Estas relaciones permiten una trazabilidad completa del proceso de análisis y una gestión estructurada de las reglas y resultados.

| Tabla | Campo | Tipo de dato | Descripción |
|---|---|---|---|
| **usuarios** | id | Integer (PK) | Identificador único del usuario |
| | correo | String | Correo electrónico del usuario |
| | contraseña_hash | String | Contraseña en formato cifrado |
| | rol | String | Rol del usuario (admin o usuario) |

BLOQUES DE TABLAS

- | ***reglas_yara*** | *id* | *Integer (PK)* | *Identificador único de la regla* |
  | | *nombre* | *String* | *Nombre de la regla YARA* |
  | | *descripcion* | *Text* | *Breve explicación de la regla* |
  | | *contenido* | *Text* | *Código de la regla YARA* |
  | | *fecha_creacion* | *DateTime* | *Fecha de creación de la regla* |

- | ***archivos*** | *id* | *Integer (PK)* | *Identificador único del archivo* |
  | | *nombre_archivo* | *String* | *Nombre original del archivo cargado* |
  | | *tipo_archivo* | *String* | *Tipo MIME del archivo* |
  | | *fecha_subida* | *DateTime* | *Fecha de subida del archivo* |
  | | *usuario_id* | *Integer (FK)* | *Identificador del usuario que subió el archivo* |

- | ***resultados*** | *id* | *Integer (PK)* | *Identificador único del resultado* |
  | | *archivo_id* | *Integer (FK)* | *Archivo al que corresponde el resultado* |
  | | *regla_id* | *Integer (FK)* | *Regla YARA utilizada* |
  | | *coincide* | *Boolean* | *Indica si hubo coincidencia con la regla* |
  | | *detalle* | *Text* | *Detalles del resultado del escaneo* |

- | ***notificaciones***| *id* | *Integer (PK)* | *Identificador único de la notificación* |
  | | *usuario_id* | *Integer (FK)* | *Usuario al que va dirigida la notificación* |
  | | *mensaje* | *Text* | *Mensaje enviado al usuario* |
  | | *fecha* | *DateTime* | *Fecha en que se generó la notificación* |

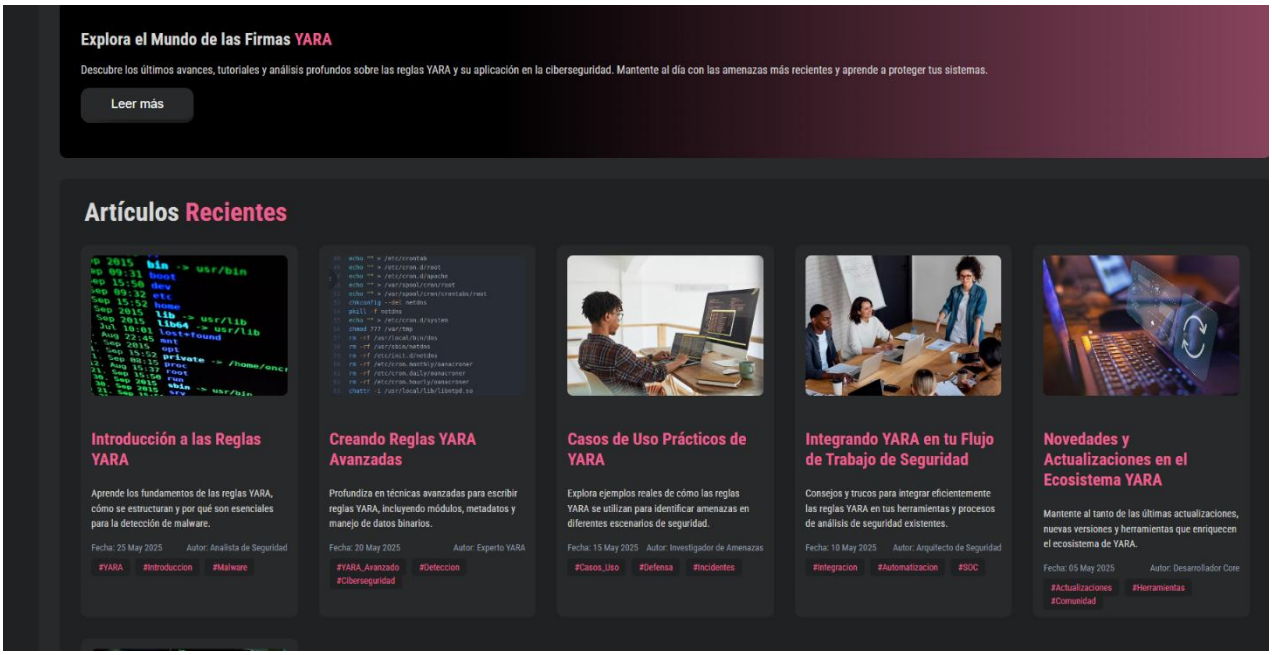
## 12.5 **Prototipo no funcional**

El diseño del aplicativo web se desarrolló considerando principios de **usabilidad, accesibilidad y experiencia de usuario (UX/UI)**. Antes de la implementación definitiva, se construyó un prototipo no funcional que permitió validar la estructura de navegación, disposición de elementos y claridad visual de las funcionalidades clave.

El diseño del sistema consideró la experiencia del usuario (UX) y la claridad visual. A continuación, se presentan capturas de pantalla que reflejan la estructura de la interfaz gráfica desarrollada.

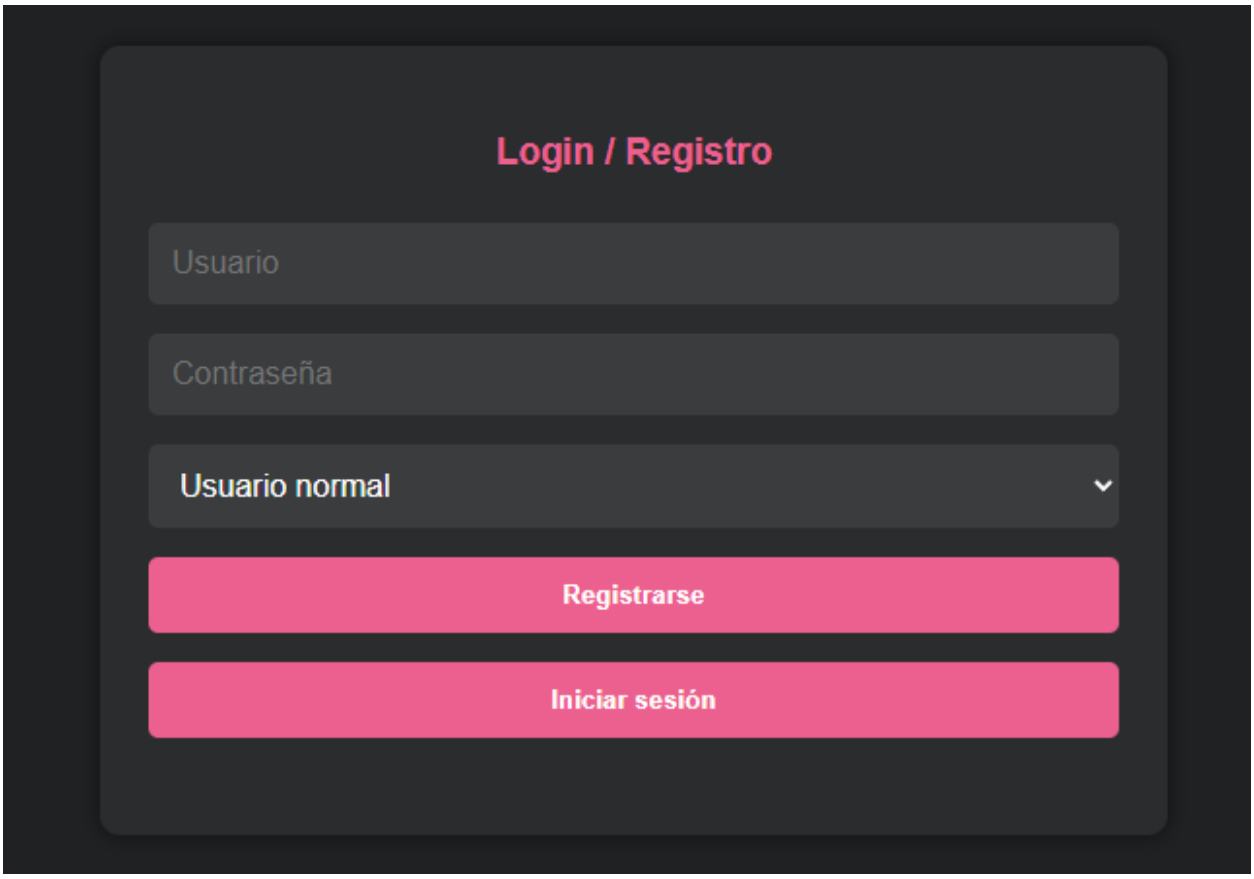
**Pantalla de inicio – Vista del portal principal**

Esta pantalla representa el acceso principal del usuario al sistema. En ella se muestran artículos relacionados con YARA, novedades y casos de uso, permitiendo al usuario estar informado y familiarizarse con el entorno. La interfaz fue diseñada para ser responsiva, moderna y clara, facilitando la navegación entre secciones como creación de reglas, visualización de resultados y gestión de archivos.



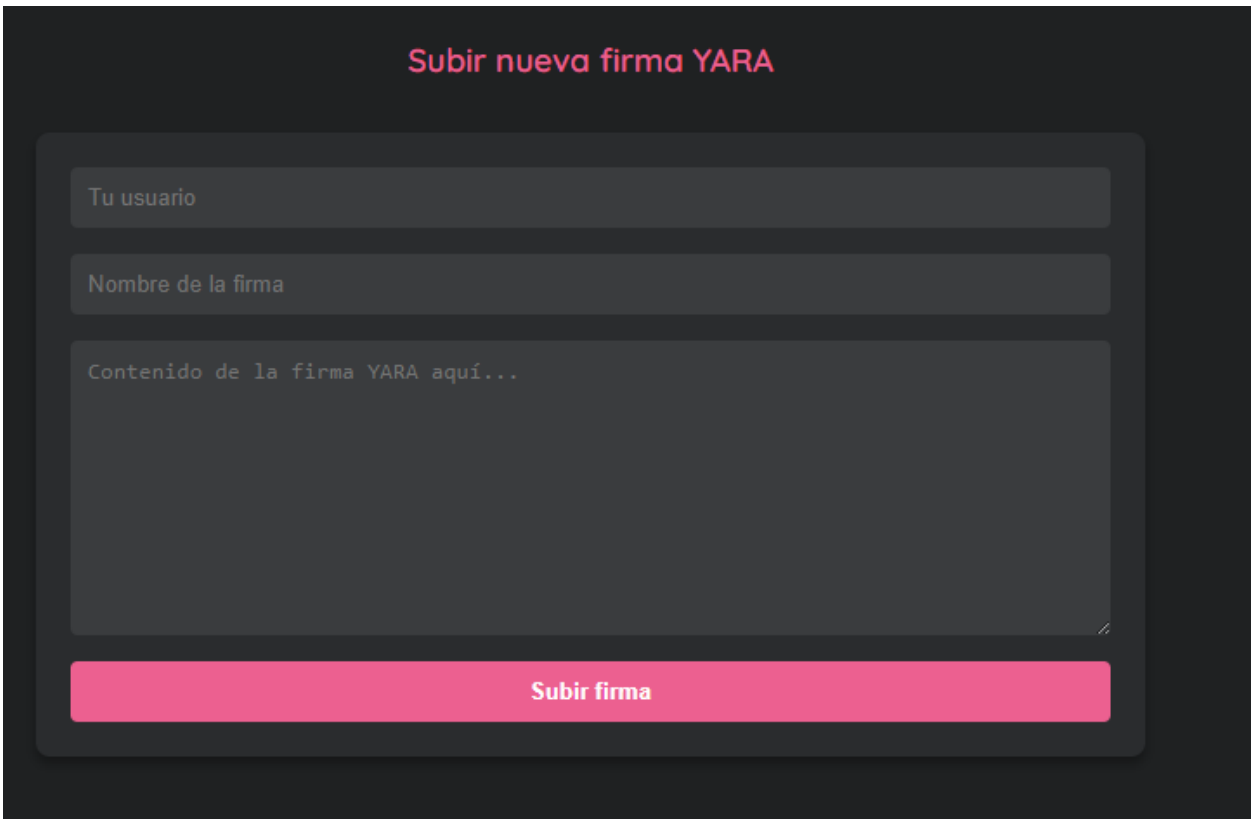
**Pantalla de Login / Registro**

Esta vista permite al usuario registrarse o iniciar sesión en la plataforma. Incluye campos para ingresar nombre de usuario, contraseña y seleccionar el tipo de cuenta (usuario normal o administrador). El diseño fue pensado para ser claro y funcional, priorizando la simplicidad en el acceso y la seguridad mediante roles diferenciados. Esta pantalla constituye el punto de entrada a las funcionalidades protegidas del sistema.



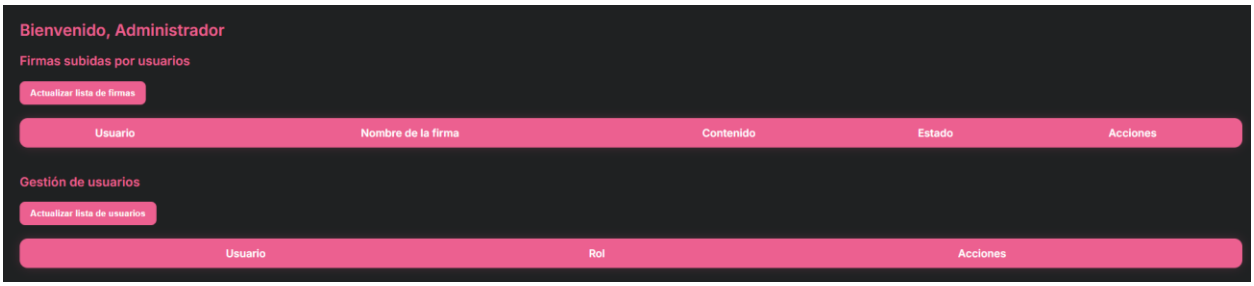
**Pantalla de creación de reglas YARA**

Esta interfaz permite al usuario registrar nuevas reglas YARA en la plataforma. Se compone de tres campos: nombre de usuario, nombre de la firma y el contenido de la regla. La información ingresada se almacena en la base de datos y puede ser utilizada luego en el análisis de archivos. Esta funcionalidad es clave en el sistema, ya que permite adaptar la detección a nuevas amenazas de manera personalizada y colaborativa.



**Pantalla de gestión administrativa**

Esta interfaz está diseñada para usuarios con rol de administrador. Desde aquí se puede consultar y actualizar la lista de firmas YARA subidas por los usuarios, revisar su contenido y estado, así como administrar los perfiles de los usuarios registrados en el sistema. El objetivo de esta vista es ofrecer un entorno centralizado de control para validar el uso adecuado de la plataforma y mantener la integridad de la información gestionada.




El prototipo fue diseñado con los siguientes criterios:

- **Interfaz intuitiva:** Se priorizó una navegación sencilla, con botones y menús claramente identificados. Esto permite que tanto usuarios técnicos como no técnicos puedan interactuar sin dificultades.
- **Diseño responsivo:** La interfaz se adaptó a diferentes dispositivos (computadores, tablets y móviles) garantizando una buena experiencia en cualquier resolución.
- **Jerarquía visual clara:** Se utilizaron colores y tipografías diferenciadas para separar títulos, botones de acción y campos de entrada.
- **Retroalimentación visual:** Cada acción del usuario (como subir un archivo, iniciar sesión o crear una regla YARA) proporciona una respuesta visual clara (por ejemplo, mensajes de éxito o error).

- **Accesibilidad:** El prototipo fue diseñado considerando contrastes adecuados y textos descriptivos, en línea con buenas prácticas de accesibilidad digital.

Estas decisiones buscaron maximizar la **eficiencia, satisfacción y aprendizaje del usuario final**, de acuerdo con los principios de diseño centrado en el usuario.

### 12.5.2 **Mockups y Wireframes**

Para validar la estructura del sistema y las funcionalidades principales, se crearon los siguientes wireframes:

- **Pantalla de inicio:** Acceso a funcionalidades básicas como login y registro.

IMAGENES DE VISTAS

- **Panel principal de usurio:** Visualización de reglas YARA disponibles, opción de escanear archivos y consultar resultados.

IMAGENES DE VISTAS

- **Gestión de reglas YARA:** Interfaz para crear, editar y eliminar firmas.
- **Historial de análisis:** Tabla con los archivos analizados, fecha, y resultados obtenidos.
- **Vista de administrador:** Módulo exclusivo con CRUD de usuarios y reglas YARA.

Estos mockups permitieron realizar pruebas tempranas de usabilidad con usuarios, asegurando que las funcionalidades fueran comprensibles y que la navegación fuera fluida.

## 12.6 Implementación

La implementación del aplicativo web se llevó a cabo de forma modular, integrando distintos componentes de software que interactúan entre sí para ofrecer una solución completa, funcional y segura para el análisis de archivos mediante firmas YARA. A continuación, se describe cada uno de los elementos del sistema, su arquitectura asociada y su funcionalidad.

| Componente de software | Arquitectura | Funcionalidad |
|---|---|---|
| **Frontend (HTML, CSS, JavaScript)** | Capa de presentación (cliente) | Interfaz gráfica que permite a los usuarios interactuar con el sistema. Incluye pantallas para iniciar sesión, cargar archivos, gestionar reglas YARA, consultar resultados y recibir notificaciones. Diseño responsive adaptado a PC, tabletas y móviles. |
| **Backend (FastAPI)** | Capa de negocio (servidor) | API REST que gestiona la lógica del sistema. Maneja la autenticación, la creación/edición de reglas YARA, la gestión de archivos y la interacción con el motor de escaneo. Contiene endpoints protegidos por JWT para garantizar la seguridad. |
| **Motor YARA** | Servicio interno embebido en el backend | Ejecuta las reglas YARA sobre los archivos subidos. Utiliza la librería `yara-python` para compilar y aplicar reglas. Devuelve resultados al backend para su almacenamiento y visualización. |
| **Base de datos PostgreSQL** | Capa de persistencia de datos | Almacena de forma estructurada los usuarios, reglas, archivos, resultados y notificaciones. Implementada mediante el ORM SQLAlchemy que facilita las |

| | | |
|---|---|---|
| | | operaciones CRUD y mantiene la integridad referencial. |
| **ORM SQLAlchemy** | Abstracción entre lógica y datos | Traduce modelos Python a tablas relacionales de la base de datos. Facilita la validación, relaciones y consultas complejas sin escribir SQL directamente. |
| **Sistema de autenticación JWT** | Middleware de seguridad | Permite controlar el acceso a los endpoints protegidos mediante tokens. Asegura que solo usuarios autenticados puedan cargar archivos o modificar reglas. |
| **Controlador de reglas YARA** | Backend - Módulo específico | Gestiona la creación, edición y validación de reglas. Antes de guardar una regla, verifica su sintaxis para evitar errores durante la ejecución. |
| **Controlador de archivos** | Backend - Módulo específico | Administra la carga de archivos, su almacenamiento temporal y el paso al motor YARA. Luego envía los resultados al frontend y a la base de datos. |
| **Panel de administración (frontend + backend)** | Interfaz exclusiva + endpoints protegidos | Permite a los administradores ver, crear, editar y eliminar usuarios, así como gestionar todas las reglas YARA del sistema. |

## 12.7 Pruebas y aseguramiento de calidad del software

Durante el desarrollo del sistema se realizaron diferentes tipos de pruebas para verificar el correcto funcionamiento de cada componente y garantizar la calidad de la solución entregada.

Se aplicaron los siguientes tipos de pruebas:

**1. Pruebas funcionales (manuales):**

- **Registro e inicio de sesión de usuarios**

  Se verificó que los usuarios pudieran registrarse, autenticarse correctamente y acceder según su rol (normal o administrador).

- **Carga de archivos**

  Se comprobó que el sistema aceptara distintos tipos de archivos y los almacenara correctamente en la base de datos.

- **Creación y edición de reglas YARA**

  Se validó que los usuarios pudieran ingresar reglas válidas, visualizarlas y modificarlas si era necesario.

- **Análisis de archivos con reglas YARA**

  Se realizaron pruebas con archivos reales y reglas de ejemplo para verificar que el motor YARA funcionara adecuadamente y devolviera resultados correctos.

- **Consulta del historial de resultados**

  Se evaluó la visualización de los análisis previos por parte de los usuarios, con acceso seguro y segmentado.

**2. Pruebas de seguridad básica:**

- Autenticación con **JSON Web Tokens (JWT)**

Se comprobó que solo usuarios autenticados pudieran acceder a funciones protegidas.

- Validaciones de entrada

Se implementaron validaciones en formularios para prevenir envíos vacíos o malformados.

**3. Resultados generales:**

- Todas las funcionalidades principales del sistema fueron probadas y confirmadas como operativas.
- No se detectaron errores críticos en las funcionalidades implementadas.
- La experiencia del usuario fue validada mediante pruebas directas con el equipo de desarrollo, cumpliendo criterios básicos de usabilidad.

## 13. **Conclusiones**

El presente proyecto ha permitido desarrollar e implementar firmas YARA en un aplicativo web para el análisis de amenazas cibernéticas, logrando una solución efectiva para la detección y clasificación de malware. A lo largo del proceso, se ha demostrado la viabilidad de esta técnica dentro de un entorno de análisis automatizado, facilitando la identificación de patrones maliciosos en archivos sospechosos y optimizando el tiempo de respuesta ante posibles incidentes de seguridad.

Entre los aspectos novedosos desarrollados en este trabajo, se destaca la integración de firmas YARA con un sistema web, lo que permite su utilización de manera accesible y eficiente en entornos empresariales. Además, se exploraron diferentes metodologías para la generación y prueba de reglas, asegurando su precisión y minimizando falsos positivos.

En cuanto al cumplimiento de los objetivos planteados, se alcanzaron de manera satisfactoria, permitiendo validar la efectividad del sistema desarrollado. Se logró implementar una herramienta funcional, capaz de integrarse con otras soluciones de seguridad y mejorar los procesos de análisis de amenazas en un entorno real.

La metodología utilizada en los análisis realizados se basó en pruebas experimentales, evaluación de desempeño y comparaciones con herramientas similares. Esto permitió garantizar la validez de los resultados obtenidos y establecer puntos de mejora para futuras optimizaciones del sistema.

Dentro de las limitaciones del proyecto, se identificó la dependencia de la actualización continua de firmas YARA para mantener la efectividad del sistema frente a nuevas amenazas.

Asimismo, la necesidad de recursos computacionales adecuados para el procesamiento eficiente de grandes volúmenes de archivos representa un desafío en entornos con infraestructura limitada.

En términos de proyecciones futuras, se plantea la posibilidad de mejorar el rendimiento del sistema mediante el uso de técnicas de inteligencia artificial para la generación y actualización automática de reglas YARA. También se sugiere la integración con otras plataformas de seguridad para potenciar su capacidad de detección y respuesta ante amenazas cibernéticas.

En conclusión, este trabajo ha demostrado la importancia del uso de firmas YARA en la detección de amenazas y su integración en un entorno web. Se ha brindado una solución práctica y escalable que puede ser adoptada en múltiples escenarios, contribuyendo significativamente a la seguridad informática y a la mitigación de riesgos cibernéticos en organizaciones y empresas.

## Referencias


Beltran Muñoz, A. (2024). Educar y proteger: análisis de la educación en ciberseguridad para combatir la ciberdelincuencia. *Revista de Educación y Derecho*, *30*. https://doi.org/10.1344/REYD2024.30.44082

Cano, J. J. M., & Rocha, A. (2019). Ciberseguridad y ciberdefensa. Retos y perspectivas en un mundo digital. *Revista Ibérica de Sistemas e Tecnologias de Informação*, *32*, VII–IX. https://doi.org/https://doi.org/10.17013/risti.32.0

Cano M., J. J. (2023). *Ciberseguridad empresarial : reflexiones y retos para los ejecutivos del siglo XXI* (2a edición.). Ediciones de la U.

Delgado, M. (2020, November 1). El valor de la ciberseguridad: *CE Noticias Financieras*. https://login.bdbiblioteca.universidadean.edu.co/login?url=https://www.proquest.com/wire-feeds/el-valor-de-la-ciberseguridad/docview/2456713592/se-2?accountid=34925

Orvisa Comunicaciones, editor. (2022). Ciberseguridad. *Mundo Eléctrico*.

Rincon, L. (2021). TEST DE PENETRACIÓN PARA EL ESTUDIO DE VULNERABILIDADES A LOS CIBERATAQUES MEDIANTE TÉCNICAS DE HACKING ÉTICO EN REDES IPV4. *Télématique*, *20*(2), 70–85.