Universidad Zaragoza

1474

Trabajo Fin de Grado

Grado en Ingeniería Informática

Repositorio de inteligencia de amenazas basada en grafos y fuentes de inteligencia abierta consultable mediante MCP

Graph-Based Threat Intelligence Repository using Open-Source Intelligence and MCP-Enabled Querying

Autores: Carlos Solana Melero

Directores: Alfredo Ramírez / Ricardo Julio Rodríguez

Escuela de Ingeniería y Arquitectura

Junio 2026

AGRADECIMIENTOS

A Fredy por toda su ayuda y por la idea original, así como a Ricardo por todas sus revisiones. A mi familia por su apoyo incondicional y a Javi por su inexplicable interés en ayudarme constantemente a que el proyecto saliera lo mejor posible. A Alex por sus vueltas en Camaro para llegar siempre a tiempo, a Raúl por sus cotilleos en el café e igualmente a Lucas por su experta opinión sobre River y a David por sus eternas ganas de intentar descubrir vulnerabilidades en la interfaz. También a Massimo por su apoyo y por haber confiado en mí para llegar hasta aquí, y a todos mis amigos por haber aguantado mis interminables charlas sobre el “tefegé”. Y por último, a los cocineros de Frampi, porque sin los almuerzos de los viernes no hubiéramos llegado igual de contentos a la oficina.

Resumen

Repositorio de inteligencia de amenazas basada en grafos y fuentes de inteligencia abierta consultable mediante MCP

En los entornos corporativos de ciberseguridad, la investigación de indicadores de compromiso se realiza de forma manual y fragmentada. Los analistas consultan de manera individual numerosas fuentes públicas dispersas, sin persistencia compartida ni correlación automática entre investigaciones. Las soluciones que centralizan este proceso suelen ser comerciales, lo que introduce dependencias de terceros difíciles de asumir cuando se exige control total sobre los datos. La falta de una plataforma autoalojada que cubriera de forma integrada el ciclo completo de la ciberinteligencia motivó la decisión de diseñarla e implementarla desde cero.

Las pocas herramientas de código abierto disponibles ofrecen soluciones parciales: unas gestionan la ingesta, otras el enriquecimiento y otras la visualización, pero ninguna integra el ciclo completo. El primer reto del proyecto fue diseñar una arquitectura distribuida que cubriera todo el proceso, desde la ingesta de fuentes hasta la consulta en lenguaje natural. El segundo fue demostrar su eficacia mediante escenarios representativos de la actividad real de un analista.

La plataforma se concibió como un conjunto de microservicios coordinados y se articuló en torno a tres pilares: Apache Kafka como bus de mensajería asíncrona, Neo4j como grafo de conocimiento modelado según STIX 2.1 e IntelOwl como motor de enriquecimiento con más de 300 analizadores. Sobre ellos se añadieron n8n para la orquestación de flujos, varios microservicios de ingesta (que alimentan el grafo con tres tipos de fuentes: indicadores de compromiso procedentes de fuentes abiertas de inteligencia, el conocimiento estructurado de adversarios de MITRE y las vulnerabilidades explotadas) y el Model Context Protocol como interfaz entre el grafo y los modelos de lenguaje locales. El despliegue se automatizó con Docker Compose y una batería de pruebas continua.

Se diseñaron cuatro escenarios de validación: análisis de direcciones IP, análisis de dominios con resolución automática de sus direcciones asociadas, análisis de hashes con correlación de la familia de malware, y consulta contextual del grafo en lenguaje natural. Todos se resolvieron de forma satisfactoria, generando en cada caso objetos STIX persistidos en Neo4j y correlacionados con entidades previas del grafo. La solución superó los objetivos iniciales: el grafo acumuló más de 993.000 nodos y 1.997.000 relaciones entre actores, técnicas, vulnerabilidades, indicadores y observables.

Sobre este grafo se habilitaron tres capacidades avanzadas. La primera es una capa de consulta en lenguaje natural mediante el Model Context Protocol, accesible desde un cliente local con LM Studio o desde el Bifrost corporativo de la organización. La segunda es una búsqueda semántica sobre el grafo, apoyada en índices vectoriales. La tercera es la publicación automática de la inteligencia enriquecida en la instancia corporativa de OpenCTI, reutilizando los paquetes STIX 2.1 ya generados y etiquetándolos por analizador y veredicto.

Palabras clave: ciberinteligencia, indicadores de compromiso, grafo de conocimiento, STIX, OSINT, MCP

Abstract

Graph-Based Threat Intelligence Repository using Open-Source Intelligence and MCP-Enabled Querying

In corporate cybersecurity environments, the investigation of indicators of compromise is carried out manually and in a fragmented manner. Analysts individually query numerous scattered public sources, with no shared persistence or automatic correlation between investigations. The solutions that centralise this process are usually commercial, which introduces third-party dependencies that are difficult to accept when full control over the data is required. No self-hosted platform covering the complete cyber threat intelligence lifecycle in an integrated way existed, so the need arose to design and implement one from scratch.

The few available open source tools offer only partial solutions: some handle ingestion, others enrichment and others visualisation, but none integrates the full cycle. The first challenge of the project was to design a distributed architecture covering the entire process, from source ingestion to natural language querying. The second was to demonstrate its effectiveness through scenarios representative of an analyst's real activity.

The platform was conceived as a set of coordinated microservices and structured around three pillars: Apache Kafka as the asynchronous messaging bus, Neo4j as the knowledge graph modelled according to STIX 2.1, and IntelOwl as the enrichment engine with more than 300 analysers. On top of them, n8n was added for workflow orchestration, several ingestion microservices —which feed the graph with three types of sources: indicators of compromise from open source feeds, the structured adversary knowledge from MITRE, and exploited vulnerabilities— and the Model Context Protocol as the interface between the graph and local language models. The deployment was automated with Docker Compose and a continuous test suite.

Four validation scenarios were designed: IP address analysis, domain analysis with automatic resolution of their associated addresses, hash analysis with malware family correlation, and contextual querying of the graph in natural language. All of them were resolved satisfactorily, generating in each case STIX objects persisted in Neo4j and correlated with pre-existing graph entities. The solution exceeded the initial objectives: the graph accumulated more than 993,000 nodes and 1,997,000 relationships among actors, techniques, vulnerabilities, indicators and observables.

Three advanced capabilities were enabled on top of this graph. The first is a natural language querying layer through the Model Context Protocol, accessible from a local client with LM Studio or from the organisation's corporate Bifrost. The second is a semantic search over the graph, supported by vector indexes. The third is the automatic publication of the enriched intelligence to the corporate OpenCTI instance, reusing the STIX 2.1 bundles already generated and labelling them by analyser and verdict.

Keywords: cyber threat intelligence, indicators of compromise, knowledge graph, STIX, OSINT, MCP

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

La ciberinteligencia (CTI, del inglés *Cyber Threat Intelligence*) se ha consolidado como disciplina esencial para anticipar, comprender y responder a amenazas digitales [1]. En este contexto, los indicadores de compromiso (IOC, del inglés *Indicator of Compromise*) constituyen evidencias técnicas observables (direcciones IP, dominios, localizadores de recursos uniformes (URL, del inglés *Uniform Resource Locator*) o hashes) a partir de las cuales se identifica actividad potencialmente maliciosa. Su utilidad depende, sin embargo, de la capacidad de correlacionarlos con vulnerabilidades, campañas, actores, técnicas y evidencias históricas.

El problema central que se aborda en este trabajo es la fragmentación y la falta de cohesión de la información de amenazas dentro de infraestructuras corporativas de ciberseguridad [2]. La investigación de un IOC exige la consulta manual e individual de múltiples fuentes públicas dispersas y heterogéneas. Estas fuentes no siempre se encuentran disponibles en local; en ocasiones, se asocian a servicios de pago o a límites estrictos de uso. Esta dependencia de la intervención humana convierte el proceso en una tarea ineficiente, lenta y propensa a errores de interpretación.

En este Trabajo Fin de Grado (TFG) se presenta Skyfall-CTI, un repositorio local de inteligencia de amenazas basado en grafos y alimentado con fuentes de inteligencia en fuentes abiertas (OSINT, del inglés *Open Source Intelligence*). A través de él, se posibilita la consulta textual de conocimiento relacionado con vulnerabilidades y exposiciones comunes (CVE, del inglés *Common Vulnerabilities and Exposures*), IOCs y amenazas persistentes avanzadas (APT, del inglés *Advanced Persistent Threat*); la investigación de IOCs de forma centralizada mediante fuentes públicas de ciberinteligencia; y la preservación de cada investigación para futuras consultas. Se integran ingesta automática de fuentes externas, enriquecimiento de observables mediante IntelOwl [3] (plataforma de código abierto para el análisis centralizado de observables a escala), persistencia en un grafo Neo4j conforme al estándar STIX (del inglés, *Structured Threat Information Expression*) 2.1, consulta mediante interfaz de programación de aplicaciones (API, del inglés *Application Programming Interface*) e interfaz web, y publicación de resultados en OpenCTI [4] (plataforma de código abierto para la gestión de inteligencia de amenazas, desarrollada por Filigran). Además, se incorpora un servidor MCP (del inglés, *Model Context Protocol*) [5] que conecta el grafo con modelos extensos de lenguaje (LLMs, del inglés *Large Language Models*) locales para consultas contextuales e investigaciones autónomas.

1.1 Entorno de trabajo y contexto

Este trabajo se desarrolló en el marco de las prácticas extracurriculares en Nologin Consulting SL, empresa española especializada en ciberseguridad, dentro de su equipo de seguridad. La empresa empleaba Cortex XSIAM (del inglés, *Extended Security Intelligence and Automation Management*) [6] como plataforma de detección y respuesta extendida para la gestión de eventos de seguridad. Disponía además de una instancia corporativa de OpenCTI (del inglés, *Open Cyber Threat Intelligence*) [4] como repositorio central de conocimiento de CTI, sobre la que el equipo consolidaba la información de amenazas.

El problema central que se abordó es la fragmentación y la falta de cohesión de la información de amenazas en la infraestructura de ciberseguridad de la organización. La investigación de un IOC exigía una recopilación de datos eminentemente manual. Los analistas de seguridad debían consultar de forma individual múltiples fuentes de información pública dispersas y heterogéneas, ninguna de ellas accesible en local. Cada verificación generaba numerosas consultas externas, muchas de ellas sujetas a límites de uso o a planes de pago. Esta dependencia de la intervención humana para agregar datos de proveedores externos convertía el proceso en una tarea ineficiente, lenta y altamente propensa a errores de interpretación [1, 7]. La necesidad de una plataforma que centralice esta consulta y persista los resultados en un repositorio reutilizable está documentada como requisito recurrente en la literatura especializada [2].

Un segundo problema se detectó en la falta de trazabilidad de los bloqueos. Los IOCs se bloqueaban en XSIAM, pero no existía ningún mecanismo para generar documentación estructurada de cada investigación. El analista carecía de un informe que recogiera las capturas obtenidas, los indicadores consultados y la razón técnica que justificaba el bloqueo. Este vacío dificultaba la transferencia de conocimiento entre analistas y la revisión posterior de decisiones de bloqueo.

La investigación de IOCs se realizaba de forma descentralizada, sin ningún punto central de agregación. Se consultaban servicios como VirusTotal, AbuseIPDB, Shodan o MalwareBazaar de manera individual y sin persistencia de los resultados. Se disponía de una instancia de OpenCTI, pero su acceso estaba restringido a determinados equipos y su interfaz no se adecuaba a la realización de consultas rápidas sobre un observable concreto. Ninguna investigación quedaba almacenada para su reutilización futura.

En este Trabajo de Fin de Grado (TFG) se construye Skyfall-CTI como respuesta a estas carencias: una plataforma que permite realizar consultas locales en lenguaje natural sobre CVEs, IOCs y grupos APT, investigar observables de forma centralizada contra múltiples fuentes públicas de ciberinteligencia, y conservar el resultado de cada investigación para consultas posteriores. Técnicamente, la herramienta ocupa el espacio entre las plataformas CTI, las herramientas OSINT y los sistemas de orquestación de seguridad, y su aportación concreta es reunir en un único despliegue autoalojado el ciclo completo: ingesta, enriquecimiento, persistencia en grafo y consulta en lenguaje natural.

1.2 Objetivos y alcance del proyecto

El objetivo general del proyecto es construir un repositorio local de inteligencia de amenazas, basado en grafos de conocimiento y en fuentes de información de acceso público, que sea consultable mediante MCP y que resuelva la fragmentación con la que hoy se investiga un IOC. La plataforma debe centralizar en un único punto la investigación de indicadores, vulnerabilidades y grupos adversarios, automatizando la consulta de múltiples fuentes externas que el analista realiza hoy de forma manual y dispersa. El resultado perseguido es una herramienta autoalojada, construida íntegramente con software libre, que cubra el ciclo completo de la ciberinteligencia (ingesta, enriquecimiento, persistencia, correlación y consulta en lenguaje natural) sin depender de servicios comerciales ni ceder el control de los datos a terceros.

Los objetivos específicos que articulan el camino hacia ese propósito son los siguientes:

- Analizar el estado del arte de plataformas CTI como OpenCTI y MISP (del inglés, *Malware Information Sharing Platform*), del estándar STIX 2.1, de las principales fuentes OSINT y de las herramientas de inteligencia de amenazas existentes.
- Diseñar y desplegar una infraestructura distribuida y contenedorizada, con un bus de mensajería asíncrona que desacople los productores de inteligencia de los consumidores del grafo.

- Implementar mecanismos de recopilación de información mediante rastreo OSINT y conectores hacia fuentes públicas de ciberinteligencia.
- Integrar una plataforma de enriquecimiento de observables que centralice el análisis mediante fuentes externas y perfiles de análisis especializados.
- Construir grafos de conocimiento que almacenen y correlacionen la información obtenida conforme al estándar STIX 2.1.
- Desarrollar una interfaz web que permita al analista consultar, visualizar y explorar el grafo de conocimiento de forma interactiva.
- Habilitar la interacción agéntica con el repositorio de inteligencia mediante MCP, facilitando consultas textuales e investigaciones autónomas apoyadas en LLMs locales.
- Integrar la plataforma con una instancia corporativa de OpenCTI, publicando de forma automática la inteligencia enriquecida como paquetes STIX 2.1 para su difusión en el ecosistema de la organización.
- Habilitar la creación manual de entidades STIX 2.1 por parte del analista, permitiendo incorporar al grafo hallazgos propios de una investigación sin necesidad de conocer la sintaxis del estándar.

1.3 Estructura de la memoria

La memoria se organiza en seis capítulos. El capítulo 1 introduce el problema, el contexto operativo, los objetivos y la estructura del documento. El capítulo 2 describe el estado del arte: las soluciones existentes para la gestión de la ciberinteligencia (plataformas CTI, herramientas de intercambio comunitario y motores de enriquecimiento) y sus limitaciones frente al objetivo de este trabajo. El capítulo 3 expone el análisis y diseño del sistema: formaliza los requisitos funcionales y no funcionales, identifica las capacidades necesarias para satisfacerlos y presenta la arquitectura que los materializa, en términos abstractos e independientes de tecnologías concretas. El capítulo 4 describe cinco escenarios representativos de la actividad del analista, detallando en términos funcionales qué hace el sistema y qué obtiene el usuario en cada caso. El capítulo 5 detalla la implementación: justifica la elección de cada tecnología frente a sus alternativas y describe la arquitectura de contenedores, el modelo de datos STIX 2.1, el flujo de ingesta, el enriquecimiento de observables, los microservicios desarrollados, las integraciones externas y la infraestructura de integración continua. El capítulo 6 presenta las conclusiones, las limitaciones del trabajo, las aportaciones propias y las líneas de trabajo futuro. Por último, se incluyen el anexo A, que detalla la planificación y dedicación, y el anexo B, que recoge la formulación matemática del sistema de puntuación de riesgo usada por Skyfall-CTI.

Capítulo 2

Contexto y estado del arte

Este capítulo sitúa Skyfall-CTI en su entorno tecnológico. En primer lugar, se examina cómo se gestiona la ciberinteligencia en organizaciones corporativas y qué papel desempeña el estándar STIX 2.1 como modelo común de representación. Después, se analizan las principales soluciones existentes (plataformas de gestión de conocimiento, herramientas de intercambio comunitario y motores de enriquecimiento) y se identifican las carencias que motivan el diseño de una solución propia. Por último, se sintetiza la comparativa y se justifica el enfoque adoptado.

2.1 El estándar STIX 2.1

La gestión moderna de la CTI ha evolucionado desde una práctica eminentemente reactiva, basada en la recopilación de IOC aislados, hacia la construcción de repositorios de conocimiento conectados, estructurados y reutilizables [1]. En los entornos corporativos tradicionales el problema central no suele ser la ausencia total de datos, sino su dispersión: el analista debe consultar manualmente fuentes públicas, plataformas comerciales, informes de fabricantes, fuentes comunitarias, bases de vulnerabilidades y resultados de analizadores externos. Esta fragmentación introduce retrasos, inconsistencias y dependencia de servicios de terceros, especialmente cuando las fuentes aplican límites estrictos de uso o requieren licencias de pago.

Para superar las limitaciones de los almacenes tabulares o documentales planos, la industria y la academia han adoptado STIX 2.1 [2] como lenguaje común para representar CTI. STIX define objetos de dominio (como `indicator`, `malware`, `vulnerability`, `attack-pattern`, `threat-actor` o `campaign`) y objetos de relación que expresan vínculos semánticos dirigidos (`indicates`, `uses`, `targets`, `attributed-to`). Esta estructura permite distinguir entre el dato técnico, la interpretación analítica y la relación que conecta ambos elementos.

La naturaleza intrínsecamente conectada de STIX encaja con las bases de datos de grafos nativas y, en particular, con el modelo de grafo de propiedades etiquetadas implementado por Neo4j [8, 9]. En un grafo, los indicadores se materializan como nodos con propiedades, mientras que las relaciones STIX se convierten en aristas dirigidas con metadatos. Esta representación facilita consultas como: qué vulnerabilidades aparecen relacionadas con un dominio, qué técnicas utiliza un actor asociado a una campaña, o qué caminos conectan un IOC reciente con evidencias históricas almacenadas.

Herramientas como StixToNeoDB e iniciativas como el *Structured Threat Intelligence Graph* (STIG) del Idaho National Laboratory muestran la viabilidad de transformar colecciones STIX en nodos y relaciones consultables mediante Cypher [10-12].

2.2 Exploración de alternativas

El panorama de soluciones existentes se articula en torno a tres familias de herramientas, que se examinan a continuación: las plataformas de gestión de conocimiento, las plataformas de intercambio comunitario y las herramientas de triaje y enriquecimiento.

OpenCTI

OpenCTI, desarrollada por Filigran, se posiciona como una de las plataformas abiertas más relevantes para la gestión de conocimiento CTI a escala corporativa [4]. Su propuesta consiste en unificar inteligencia táctica, técnica y estratégica sobre un modelo compatible con STIX 2.1, incorporando gestión de entidades, visualización de relaciones, conectores, automatización y funciones de colaboración [13, 14].

Su despliegue típico combina una aplicación principal, procesos de trabajo, conectores, Elasticsearch u OpenSearch para indexación, Redis para caché, RabbitMQ para colas y almacenamiento de objetos mediante MinIO o S3 [4, 13]. La tabla 2.1 resume los componentes habituales. Esta separación de responsabilidades proporciona escalabilidad y resiliencia, pero también eleva la huella de recursos, la complejidad de operación y el esfuerzo de mantenimiento.

Tabla 2.1: Componentes habituales en un despliegue OpenCTI

Componente	**Función principal**
Aplicación OpenCTI	Interfaz, API, modelo de conocimiento y coordinación de operaciones.
Workers	Procesamiento asíncrono de tareas internas, importaciones y enriquecimientos.
Conectores	Integración con fuentes externas, MITRE ATT&CK, OTX, Shodan u otras APIs.
Elasticsearch/OpenSearch	Indexación y búsqueda de entidades, relaciones y eventos.
Redis	Caché y apoyo a operaciones de alto rendimiento.
RabbitMQ	Mensajería interna entre servicios y conectores.
MinIO/S3	Almacenamiento de objetos, adjuntos y artefactos.

MISP

Frente al modelo de gestión de conocimiento de OpenCTI, MISP representa la alternativa comunitaria dominante para el intercambio automatizado de indicadores técnicos y eventos entre organizaciones de confianza [14-16]. Su filosofía se basa en la reciprocidad analítica: una comunidad comparte eventos, atributos, objetos, taxonomías y galaxias para que otros miembros puedan reutilizarlos en detección, bloqueo o investigación.

La arquitectura interna de MISP es más ligera que la de OpenCTI, apoyándose en una aplicación web, una base de datos relacional, Redis para colas y caché, y módulos de expansión en Python. MISP-STIX mantiene la conversión bidireccional entre el formato nativo de MISP y STIX 1.x, 2.0 y 2.1 [17], lo que facilita la interoperabilidad con plataformas basadas en grafos.

La principal limitación de MISP radica en que sus vistas están optimizadas para eventos y atributos, no para la exploración profunda de caminos ni para la investigación agéntica sobre un grafo persistente.

2.3 Análisis comparativo de soluciones CTI

La tabla 2.2 resume la posición relativa de las principales alternativas frente al objetivo de construir un repositorio local, modular, persistente y consultable mediante interacción agéntica.

Tabla 2.2: Comparativa cualitativa con herramientas relacionadas

Criterio	**OpenCTI**	**MISP**
Enfoque analítico	Gestión integral de conocimiento táctico, técnico y estratégico.	Intercambio rápido de IOCs y eventos entre comunidades.
Persistencia	Stack híbrido con índices, colas y almacenamiento distribuido.	Base relacional orientada a eventos y atributos.
Complejidad operativa	Elevada, con múltiples servicios, conectores y procesos de trabajo.	Baja o moderada, adecuada para comunidades y despliegues ligeros.
Enriquecimiento	Conectores independientes por fuente o integración.	Módulos, API REST y correlación comunitaria.
IA y consulta	Integraciones propias en suites empresariales.	Sin soporte agéntico nativo avanzado.
Limitación principal	Complejidad y huella de infraestructura.	Menor capacidad de exploración relacional profunda.

Del análisis se desprende que cada plataforma cubre un subconjunto del problema. OpenCTI ofrece la mayor profundidad de conocimiento táctico, pero a costa de una infraestructura compleja. MISP destaca en el intercambio rápido entre comunidades, aunque con menor capacidad de exploración relacional profunda.

Queda, por tanto, un espacio sin cubrir por completo: el de un repositorio local, explorable y autoalojado, con soporte agéntico, orientado al analista que necesita investigar y conservar resultados sin depender de una infraestructura empresarial compleja. Este hueco es el que trata de cubrir el sistema desarrollado en el marco de este Trabajo de Fin de Grado.

Capítulo 3

Análisis y diseño

En este capítulo se expone el análisis que precedió al diseño de Skyfall-CTI. Se presentan los requisitos funcionales y no funcionales derivados del contexto operativo y del estado del arte, las capacidades que el sistema debe incorporar para satisfacerlos y la arquitectura que los materializa. El capítulo adopta una perspectiva abstracta: describe qué debe hacer la plataforma y cómo se organiza estructuralmente.

3.1 Requisitos del sistema

Los requisitos se clasificaron en dos grupos: funcionales (RF), que describen capacidades que la plataforma debe ofrecer, y no funcionales (RNF), que establecen restricciones de calidad, rendimiento y operación. Cada requisito se derivó del análisis del contexto operativo de la organización y de la revisión del estado del arte del capítulo 2. Las tablas 3.1 y 3.2 recogen, respectivamente, los requisitos funcionales y los no funcionales.

Tabla 3.1: Requisitos funcionales de Skyfall-CTI

ID	Nombre	Descripción
RF1	Análisis de observables	El sistema debe analizar IPs, dominios, URL y hashes consultando analizadores externos y devolver el resultado de forma estructurada.
RF2	Persistencia en grafo	El sistema debe persistir cada análisis como un paquete STIX 2.1 en la base de datos de grafos.
RF3	Correlación CTI	El sistema debe correlacionar automáticamente los IOCs con vulnerabilidades, malware, campañas, actores y técnicas presentes en el grafo.
RF4	Ingesta de fuentes externas	El sistema debe ingerir, de forma periódica y automática, fuentes públicas de inteligencia de amenazas: técnicas de adversarios, vulnerabilidades, indicadores y canales de mensajería.
RF5	Interfaz web	El sistema debe proporcionar una interfaz web desde la que el analista pueda lanzar análisis, visualizar resultados y explorar el grafo.
RF6	Consulta en lenguaje natural	El sistema debe permitir consultas textuales sobre el grafo mediante MCP, sin requerir conocimiento de Cypher.
RF7	Puntuación de riesgo	El sistema debe asignar a cada observable analizado una puntuación de riesgo compuesta [0–100] ponderando señales de múltiples fuentes.
RF8	Generación de informes	El sistema debe generar informes *Portable Document Format* (PDF) descargables con el resumen de una investigación.
RF9	Búsqueda semántica	El sistema debe permitir consultar los nodos CTI del grafo mediante similitud semántica.
RF10	Integración con OpenCTI corporativo	El sistema debe publicar los resultados del enriquecimiento en la instancia OpenCTI corporativa como paquetes STIX 2.1, conservando el marcado TLP, la autoría y una etiqueta por analizador ejecutado.
RF11	Exploración visual del grafo	El sistema debe permitir al analista expandir nodos y navegar por sus relaciones de forma interactiva en el navegador.

Tabla 3.2: Requisitos no funcionales de Skyfall-CTI

ID	Nombre	Descripción
RNF1	Autoalojado	La plataforma debe ejecutarse íntegramente en local, sin depender de servicios externos de cómputo.
RNF2	Modularidad	Cada componente debe poder actualizarse, escalarse o reiniciarse de forma independiente sin afectar al resto.
RNF3	Idempotencia	La ingesta de paquetes STIX debe ser idempotente: reingestar el mismo paquete no debe crear duplicados en la base de datos.
RNF4	Aislamiento de red	Los servicios de bases de datos y mensajería no deben exponerse directamente al exterior del entorno de despliegue.
RNF5	Acceso de solo lectura	La interfaz web debe acceder al grafo exclusivamente en modo lectura; las credenciales de escritura no deben estar disponibles en el cliente.
RNF6	Gestión de secretos	Las credenciales y claves de API externas deben gestionarse exclusivamente mediante variables de entorno, sin aparecer en el código fuente.
RNF7	Tolerancia a fallos	El flujo de enriquecimiento debe tolerar fallos individuales de analizadores sin interrumpir el análisis global del observable.

3.2 Capacidades requeridas

Cada requisito de las tablas 3.1 y 3.2 se traduce en una capacidad que el sistema debe incorporar. Estas capacidades se describen aquí en términos de función, no de producto.

- Mensajería asíncrona desacoplada. Para satisfacer la modularidad (RNF2) y la tolerancia a fallos (RNF7), los productores de inteligencia y los consumidores del grafo deben comunicarse a través de un bus de mensajería persistente que permita reiniciar, escalar o reprocesar cada componente de forma independiente. Esta capacidad sostiene además la ingesta continua de fuentes (RF4).
- Persistencia orientada a grafos. La correlación de IOCs con vulnerabilidades, malware, campañas, actores y técnicas (RF3) y la persistencia de cada análisis (RF2) exigen un almacén que represente entidades y relaciones de forma nativa y que permita navegarlas con consultas expresivas. La búsqueda semántica (RF9) y la exploración visual (RF11) requieren que ese mismo almacén soporte índices de similitud vectorial.
- Enriquecimiento multi-fuente. El análisis de observables (RF1) necesita una capa capaz de consultar en paralelo múltiples fuentes externas de reputación y devolver un resultado homogéneo, tolerando el fallo individual de cualquier fuente (RNF7).
- Recopilación OSINT automatizada. La ingesta periódica de fuentes públicas (RF4) requiere mecanismos de rastreo y conectores programables, coordinados por un planificador de flujos.
- Modelo de datos estandarizado. La persistencia (RF2) y la correlación (RF3) deben apoyarse en un modelo de representación común e interoperable, que distinga el dato técnico de su interpretación analítica y de las relaciones entre ambos, garantizando además ingesta idempotente (RNF3).
- Consulta agéntica en lenguaje natural. La consulta sin conocer el lenguaje del grafo (RF6) requiere una capa de interfaz que conecte modelos de lenguaje con el repositorio, exponiendo operaciones de alto nivel verificables sobre los datos locales.
- Interfaz web de analista. El lanzamiento de análisis, la visualización de resultados y la exploración interactiva del grafo (RF5, RF11) precisan una aplicación web que acceda al grafo exclusivamente en modo lectura (RNF5).
- Servicios de valor analítico. La puntuación de riesgo compuesta (RF7) y la generación de informes (RF8) se resuelven mediante lógica de servidor que agrega las señales de las distintas fuentes.
- Interoperabilidad con plataformas CTI corporativas. La publicación de los resultados en una instancia OpenCTI corporativa (RF10) obliga a tener en cuenta varias condiciones de diseño. El sistema debe reutilizar el paquete STIX 2.1 que ya se genera en el enriquecimiento, sin construir un modelo de datos paralelo. La ingesta en OpenCTI ha de ser idempotente por identificador STIX, de modo que reenviar un mismo análisis no duplique entidades (RNF3). El protocolo de clasificación de la información (TLP, del inglés *Traffic Light Protocol*) y la autoría deben preservarse para respetar la política de diseminación de la información. Por último, la integración debe ser opcional y estar desacoplada del flujo principal: la indisponibilidad o un fallo de OpenCTI no puede interrumpir el análisis ni la persistencia local en el grafo propio (RNF7).
- Despliegue autoalojado y aislado. El conjunto debe ejecutarse íntegramente en local (RNF1), con los servicios de datos y mensajería sin exposición directa al exterior (RNF4) y los secretos gestionados fuera del código (RNF6).

3.3 Arquitectura del sistema

La arquitectura de Skyfall-CTI se documenta mediante el modelo C4 [18], que organiza la descripción de un sistema software en cuatro niveles de abstracción progresiva: contexto (qué hace el sistema y con quién se comunica), contenedores (qué procesos o servicios lo componen), componentes (cómo se estructura cada servicio internamente) y código (implementación de cada componente). En este capítulo se emplean los dos primeros niveles, que son suficientes para comunicar las decisiones de diseño relevantes sin descender a detalles de implementación, tratados en el capítulo 5. El diagrama de contexto (nivel 1; véase la sección 3.3.1) sitúa la plataforma en su entorno y muestra los sistemas externos con los que se comunica. El diagrama de contenedores (nivel 2; véase la sección 3.3.2) desglosa los servicios internos de la plataforma, los volúmenes que persisten su estado y las relaciones de comunicación entre ellos.

3.3.1 Diagrama de contexto: integraciones externas

La Figura 3.1 muestra el diagrama de contexto de Skyfall-CTI, es decir, los sistemas externos con los que la plataforma se comunica. El diagrama agrupa las integraciones en tres zonas cromáticas según su papel en el ciclo de inteligencia. El fondo naranja (zona superior) reúne las fuentes de ingesta, que alimentan el grafo de forma autónoma y periódica con conocimiento estructurado: técnicas y grupos adversarios de MITRE ATT&CK, vulnerabilidades explotadas del catálogo CISA KEV, pulsos de OTX, indicadores de fuentes OSINT y síntesis de amenazas activas vía Grok. El fondo oscuro (zona inferior) agrupa las fuentes de enriquecimiento, consultadas a través de IntelOwl cuando el analista solicita el análisis de un observable; aportan reputación, geolocalización, análisis de malware y detección multimotor. Esta separación facilita ampliar el sistema sin rediseñar el núcleo: añadir una fuente nueva no afecta al resto. El fondo rosa (zona derecha) corresponde a los consumidores externos que interrogan el grafo a través de la capa MCP: el Bifrost corporativo (*gateway* MCP y LLM de la organización) y un cliente local con LM Studio [19] (servidor de inferencia LLM de código abierto). La tabla 3.3 detalla cada integración y lo que aporta.

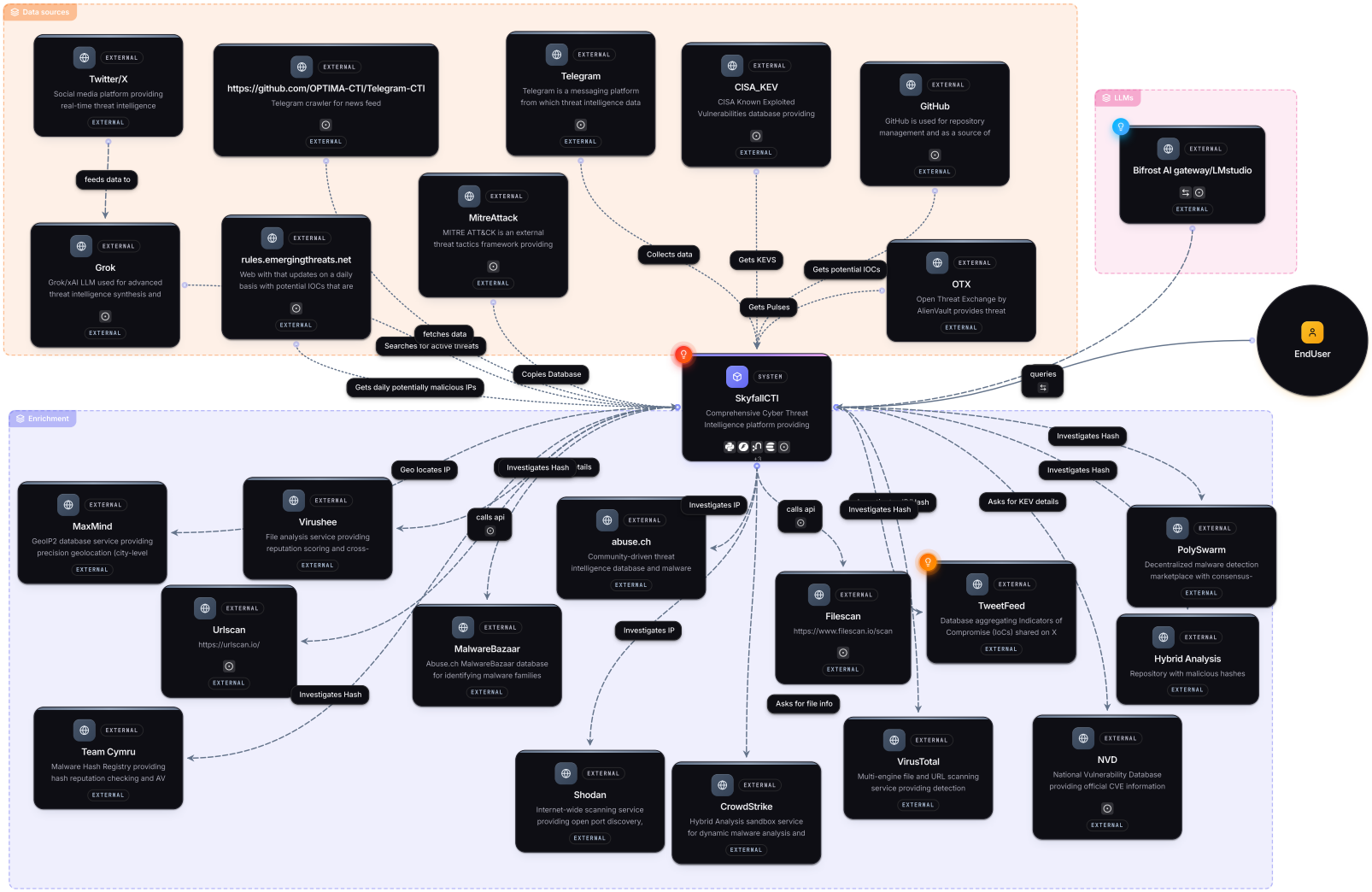


Figura 3.1: Diagrama de contexto (diagrama C4 nivel 1). Fondo naranja: fuentes de ingesta autónoma (MITRE ATT&CK, CISA KEV, OTX, fuentes OSINT, Grok). Fondo oscuro: fuentes de enriquecimiento bajo demanda a través de IntelOwl. Fondo rosa: consumidores externos de la capa MCP (Bifrost corporativo y LM Studio).

Tabla 3.3: Integraciones externas de Skyfall-CTI y su aportación.

Sistema externo	**Papel**	**Qué aporta**
MITRE ATT&CK [20]	Ingesta	Catálogo de técnicas, malware, herramientas y grupos adversarios.
CISA KEV [21]	Ingesta	Vulnerabilidades explotadas activamente, para priorizar las amenazas reales.
NVD [22]	Ingesta	Detalles y puntuación CVSS de cada CVE.
OTX (AlienVault)	Ingesta	Pulsos comunitarios con indicadores recientes agrupados por campaña.
GitHub	Ingesta	Repositorios con volcados de indicadores y hashes de malware.
Telegram	Ingesta	Canales de inteligencia con filtraciones e indicadores.
Emerging Threats	Ingesta	Listas diarias de direcciones IP potencialmente maliciosas.
Grok/xAI	Ingesta	Síntesis de amenazas activas mediante un modelo de lenguaje.
VirusTotal	Enriquecimiento	Reputación multimotor de IPs, dominios y hashes.
AbuseIPDB	Enriquecimiento	Reputación de direcciones IP a partir de reportes de abuso.
Shodan	Enriquecimiento	Puertos y servicios expuestos de una dirección IP.
MaxMind	Enriquecimiento	Geolocalización de direcciones IP.
MalwareBazaar	Enriquecimiento	Muestras y familia de malware asociadas a un hash.
Team Cymru	Enriquecimiento	Reputación de hashes mediante su *Malware Hash Registry*.
Hybrid Analysis	Enriquecimiento	Análisis dinámico de ficheros en entorno aislado.
PolySwarm	Enriquecimiento	Veredicto consensuado de múltiples microanalizadores.
TweetFeed	Enriquecimiento	Indicadores extraídos de la red social X.
urlscan.io	Enriquecimiento	Análisis de URLs y captura del sitio web.
filescan.io	Enriquecimiento	Análisis estático y dinámico de ficheros.
Virushee	Enriquecimiento	Reputación y verificación cruzada de hashes.
Bifrost corporativo	Consumo	*Gateway* MCP de la organización que consulta el grafo en lenguaje natural mediante LLMs.
LM Studio	Consumo	Cliente LLM local que accede al grafo a través de la capa MCP.

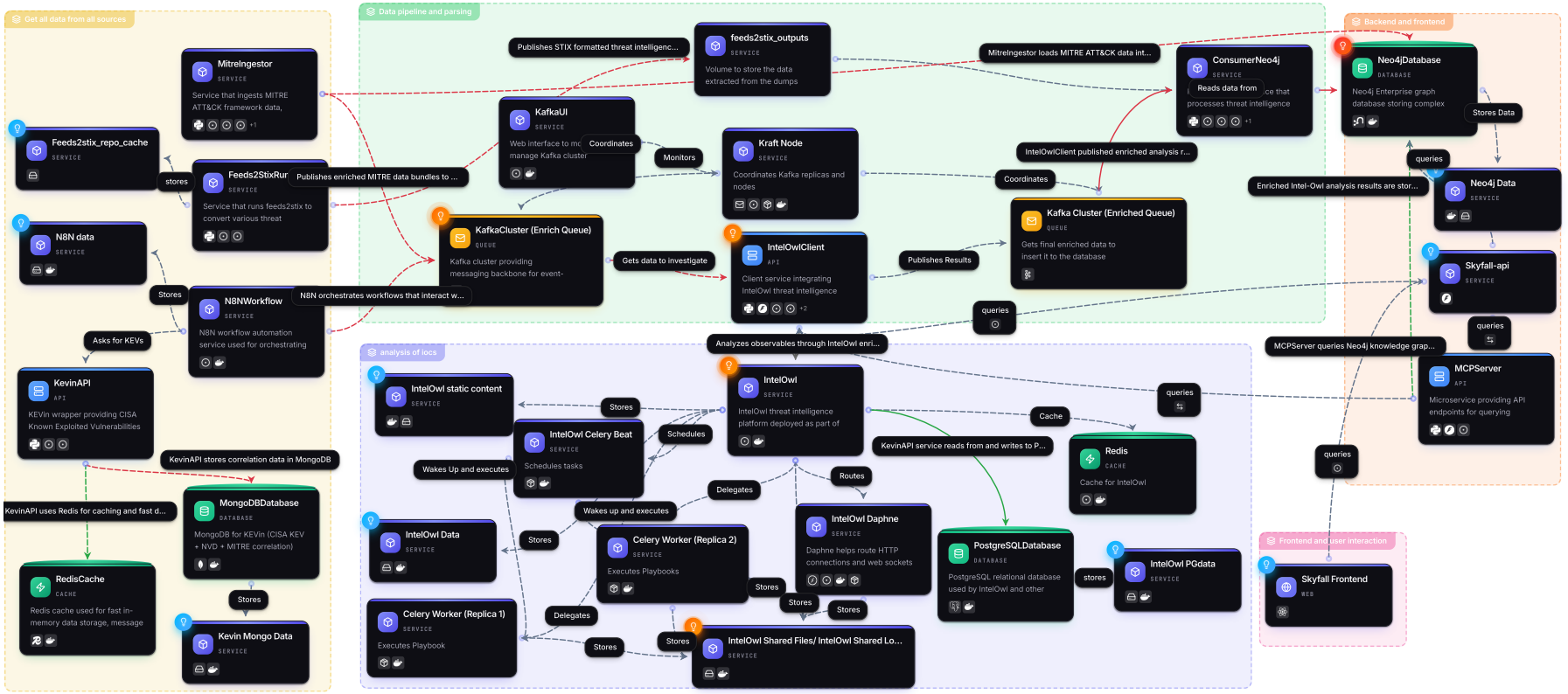


Figura 3.2: Vista de contenedores internos de Skyfall-CTI (diagrama C4 nivel 2). Los contenedores se agrupan en cinco bloques cromáticos correspondientes a las cinco capas lógicas descritas en el texto.

3.3.2 Vista de contenedores internos

La Figura 3.2 presenta la vista de contenedores de Skyfall-CTI, correspondiente al nivel 2 del modelo C4. Recoge los contenedores Docker que componen la plataforma, los volúmenes que persisten su estado y las relaciones de comunicación entre ellos. El diagrama agrupa los contenedores en cinco bloques cromáticos, cada uno con una responsabilidad bien delimitada. El flujo de datos recorre el diagrama de izquierda a derecha: las fuentes alimentan el bus de mensajería, el bus distribuye hacia el motor de análisis y la capa de persistencia, y la interfaz consulta el resultado ya correlacionado. A continuación se describe cada bloque.

El bloque de ingesta de fuentes (en amarillo) reúne los servicios que alimentan el grafo con inteligencia externa. El orquestador `n8n` [23] actúa como planificador: despierta y ejecuta los servicios de ingesta según la periodicidad configurada, y conserva sus flujos en el volumen *N8N data*. El servicio `mitre-ingestor` carga el corpus de MITRE ATT&CK (del inglés, *MITRE Adversarial Tactics, Techniques and Common Knowledge*) [20] y sondea los pulsos de OTX (del inglés, *Open Threat Exchange*). El servicio `feeds2stix-runner` convierte fuentes OSINT a STIX 2.1 y mantiene un clon local del repositorio en el volumen *feeds2stix_repo_cache*. Por su parte, `kevin-api` consolida el catálogo de la CISA (del inglés, *Cybersecurity and Infrastructure Security Agency*) con las vulnerabilidades explotadas conocidas (KEV, del inglés *Known Exploited Vulnerabilities*), las métricas CVSS (del inglés, *Common Vulnerability Scoring System*) de la NVD (del inglés, *National Vulnerability Database*) y las fichas de MITRE ATT&CK; se apoya en Redis como caché de respuestas y persiste las correlaciones resultantes en MongoDB (volumen *Kevin Mongo Data*).

El bloque de mensajería (en verde) constituye la columna vertebral asíncrona del sistema. Apache Kafka transporta los mensajes entre productores y consumidores. Se diferencian dos colas: la de entrada `enrichment.requests` (*Enrich Queue*) y la de salida `enrichment.results` (*Enriched Queue*). El servicio `intelowl-client` hace de puente entre Kafka e IntelOwl: consume las solicitudes de la cola de entrada y publica los resultados enriquecidos en la de salida. El servicio `consumer-neo4j` lee de todos los tópicos STIX (concretamente, los listados en la tabla 3.4) y del volumen *feeds2stix_outputs*, y persiste los paquetes en el grafo. La interfaz *KafkaUI* permite monitorizar el estado del clúster. Este desacoplamiento mediante colas permite reiniciar, escalar o reprocesar cada componente de forma independiente. La tabla 3.4 recoge el conjunto fijo de tópicos Kafka, cada uno asociado a una fuente o etapa del flujo.

Tabla 3.4: Tópicos Kafka de Skyfall-CTI

Tópico	**Flujo (productor → consumidor)**	**Contenido**
`enrichment.requests`	UI → intelowl-client	Solicitudes de análisis de IOCs.
`enrichment.results`	intelowl-client, OTX → consumer-neo4j	Resultados STIX del enriquecimiento.
`stix.cve`	kevin-api → consumer-neo4j	Vulnerabilidades CVE/KEV.
`stix.osint`	feeds2stix → consumer-neo4j	Fuentes OSINT normalizadas.
`stix.telegram`	telegram-crawler	IOCs de canales Telegram.
`stix.dumps`	dumps-crawler	IOCs de repositorios GitHub.
`stix.mitre`	mitre-ingestor	Objetos MITRE ATT&CK.

El bloque de análisis IntelOwl (en morado) materializa el enriquecimiento multi-fuente. La plataforma IntelOwl pone a disposición más de 300 analizadores en total; Skyfall-CTI configura y activa más de 20 de ellos, concretamente los más relevantes para el análisis de IOCs. Varias réplicas de *Celery Worker* ejecutan los perfiles de análisis (conjuntos de analizadores preconfigurados para un tipo de observable)

en paralelo, apoyadas por un planificador de tareas periódicas y los servicios de base de datos y caché propios de IntelOwl. Los volúmenes asociados conservan los artefactos y registros compartidos entre sus contenedores.

El bloque de backend y persistencia (en naranja) concentra el núcleo de consulta y el almacén de conocimiento. La base de datos Neo4j custodia el grafo principal como nodos y aristas tipados con etiquetas semánticas STIX, y persiste su estado en el volumen *Neo4j Data*. El servicio `skyfall-api` (FastAPI [24]) es la API central: ejecuta consultas Cypher de solo lectura sobre el grafo, delega los análisis bajo demanda y calcula la puntuación de riesgo y los informes. El contenedor *MCPServer* expone el grafo a los LLMs para la consulta en lenguaje natural; esta capa MCP se rediseñó posteriormente en tres bridges compatibles con clientes corporativos, como se detalla en la sección 5.6.

El bloque de interfaz (en rosa) es el punto de entrada del analista. *Skyfall Frontend*, servido por Nginx [25], constituye la aplicación web desde la que se lanzan análisis, se visualizan resultados y se explora el grafo. Todas sus peticiones se encaminan hacia `skyfall-api`, de modo que el cliente nunca accede directamente al almacén de datos.

Capítulo 4

Casos de uso

Este capítulo describe los cinco casos de uso que articulan el diseño de Skyfall-CTI e ilustra, mediante un diagrama de secuencia, el flujo interno del escenario de mayor complejidad técnica. En primer lugar se presenta el diagrama UML de casos de uso, donde se agrupan los cinco escenarios identificados, el analista como actor principal y los sistemas externos con los que cada uno se comunica. Después, se desarrolla el diagrama de secuencia de la consulta en lenguaje natural, el caso que involucra el mayor número de componentes y la interacción más rica entre ellos. Por último, se muestra el aspecto de la plataforma en dos escenarios representativos.

4.1 Casos de uso

La Figura 4.1 presenta el diagrama de casos de uso de Skyfall-CTI, donde el analista actúa como actor principal en los cinco escenarios identificados. Concretamente, se han definido los siguientes casos de uso:

1. **Analizar dirección IP:** este escenario cubre el escenario de evaluación de reputación de un observable de red. El analista introduce la dirección en la interfaz web; el sistema detecta automáticamente el tipo de observable y lanza en paralelo los analizadores configurados contra las fuentes OSINT externas: VirusTotal, AbuseIPDB, Shodan, MaxMind, CrowdSec y GreyNoise, entre otras. El resultado de cada fuente se normaliza en un paquete STIX 2.1 y se persiste en el grafo. La interfaz presenta la información en dos perspectivas: la externa, con las detecciones individuales de cada analizador y la puntuación de riesgo compuesta; y la local, que navega el grafo para exponer campañas, actores de amenaza, técnicas MITRE ATT&CK y vulnerabilidades previamente asociadas a esa dirección.
2. **Investigar dominio:** este escenario sigue la misma dinámica pero añade una capacidad de resolución automática. El analista introduce el dominio en la interfaz web; el sistema lo analiza contra las fuentes OSINT configuradas y normaliza los resultados en un paquete STIX 2.1 que se persiste en el grafo, ofreciendo igualmente la perspectiva externa de reputación y la perspectiva local de relaciones. Adicionalmente, el sistema resuelve de forma automática las IPs a las que apunta el dominio y lanza sobre cada una el análisis de dirección correspondiente. De este modo, un dominio recién observado puede vincularse con infraestructura de ataque ya documentada en el grafo sin intervención manual adicional.
3. **Analizar *hash* de malware:** este escenario centra el enriquecimiento en muestras de ficheros. El analista introduce un *hash* SHA-256 y el sistema lanza el perfil de análisis de reputación de *hashes*, que consulta VirusTotal, MalwareBazaar, Team Cymru, Hybrid Analysis, PolySwarm, TweetFeed y filescan.io, entre otros analizadores especializados. Los resultados se convierten a un paquete STIX con objetos `file`, `malware` y `sighting`. Desde la vista local del grafo, el analista puede navegar desde el *hash* hasta la familia de malware identificada, las técnicas MITRE ATT&CK que emplea y

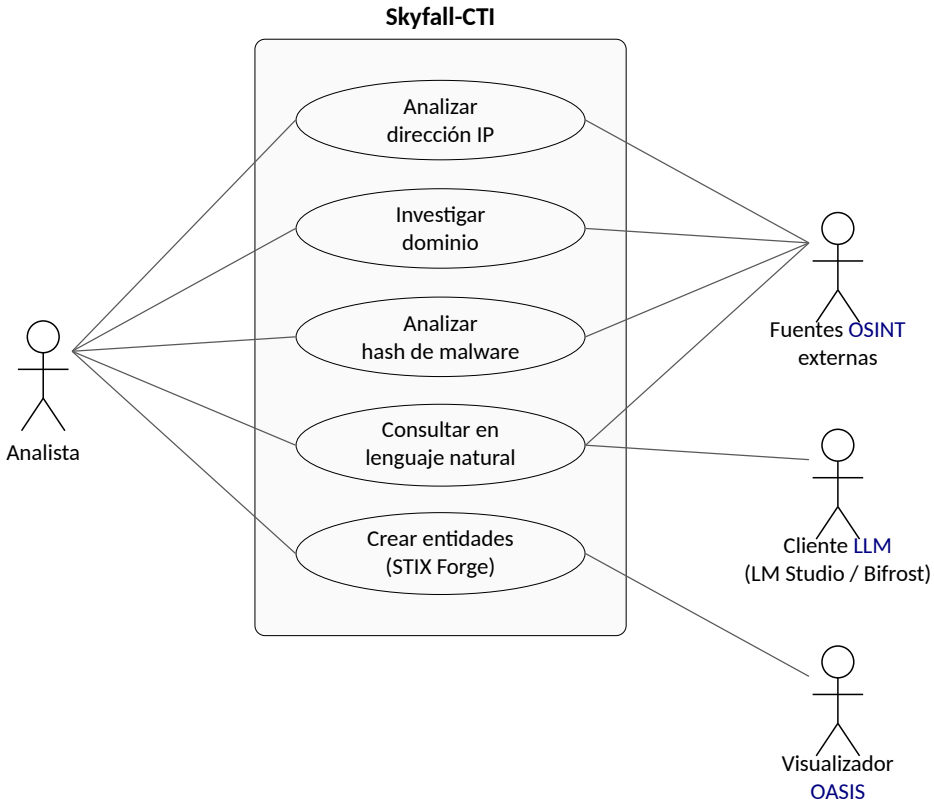


Figura 4.1: Diagrama de casos de uso de Skyfall-CTI en notación UML. El analista actúa como actor principal en los cinco escenarios. Las fuentes OSINT externas participan en los tres casos de enriquecimiento de observables. El cliente LLM es el actor externo de la consulta en lenguaje natural. El visualizador OASIS es un actor opcional en la creación manual de entidades con STIX Forge.

los grupos de amenaza que la han utilizado, obteniendo en un único análisis el contexto completo de la muestra.

4. **Consultar en lenguaje natural:** este escenario es el de mayor complejidad técnica; su flujo interno se detalla en la Sección 4.2. El analista formula una pregunta en texto libre; el cliente LLM (LM Studio o el Bifrost corporativo) la interpreta y, de forma autónoma, encadena las llamadas a herramienta necesarias a través de la capa MCP para construir la respuesta: puede consultar el grafo de conocimiento acumulado y, simultáneamente, lanzar el análisis de un observable concreto contra las fuentes OSINT externas configuradas en IntelOwl. En ningún momento el analista necesita conocer el lenguaje de consulta subyacente ni el esquema del grafo.

5. **Crear entidades con STIX Forge:** este escenario cubre la incorporación de inteligencia que no procede de ninguna fuente automatizada, como los hallazgos propios de una investigación o la documentación de un incidente reciente. El analista construye el grafo de inteligencia visualmente: selecciona los tipos de objetos STIX 2.1 que desea crear (indicadores, malware, vulnerabilidades, actores de amenaza y las relaciones entre ellos), los conecta en el lienzo de arrastrar y soltar, y el sistema los traduce a un paquete válido que ingiere directamente en Neo4j. A medida que se añaden entidades, la interfaz muestra en tiempo real las correlaciones detectadas con el conocimiento preexistente en el grafo. De forma opcional, el paquete puede exportarse al visualizador de la organización OASIS (del inglés, *Organization for the Advancement of Structured Information Standards*) externo para verificar su conformidad con el estándar antes de su publicación.

4.2 Diagrama de secuencia MCP

La Figura 4.2 muestra el diagrama de secuencia en notación UML del caso de uso de consulta en lenguaje natural. Como se ha descrito anteriormente, se detalla únicamente esta secuencia de mensajes al ser la de mayor complejidad. El analista formula una única pregunta; a partir de ahí, el cliente LLM actúa de forma autónoma encadenando las llamadas a herramienta necesarias para construir la respuesta sin nueva intervención del analista.

En la primera llamada a herramienta, el modelo solicita una consulta al grafo de conocimiento. La petición atraviesa la puerta de acceso MCP, que verifica la identidad del cliente y la reenvía al conector del grafo; este traduce la intención del modelo a la consulta interna correspondiente y la ejecuta sobre Neo4j en modo de solo lectura. Los resultados regresan al modelo sin que el analista intervenga ni deba conocer el lenguaje de consulta del grafo. Este mecanismo permite explorar relaciones complejas (como qué grupos APT explotan vulnerabilidades del catálogo CISA KEV) de forma completamente transparente.

Con la respuesta del grafo en su contexto, el modelo encadena una segunda llamada a herramienta: solicita el análisis del observable a IntelOwl a través del mismo canal de comunicación. El conector de IntelOwl lanza los analizadores configurados, espera a que todos completen su ejecución y devuelve al modelo un resumen con el veredicto global, las puntuaciones de cada fuente y las etiquetas de amenaza identificadas.

Con ambos resultados en su contexto de conversación, el cliente LLM construye la respuesta final y la presenta al analista.

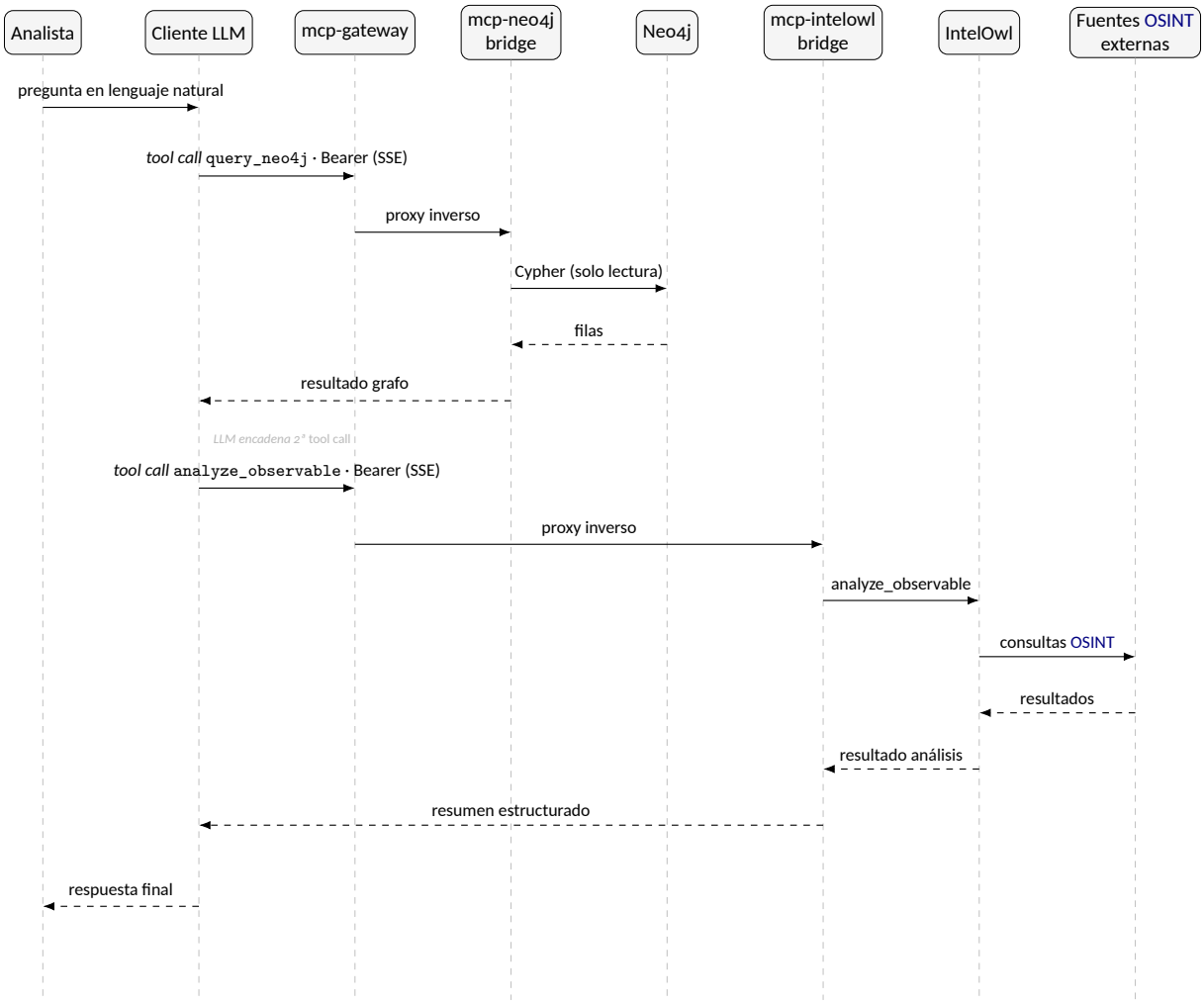
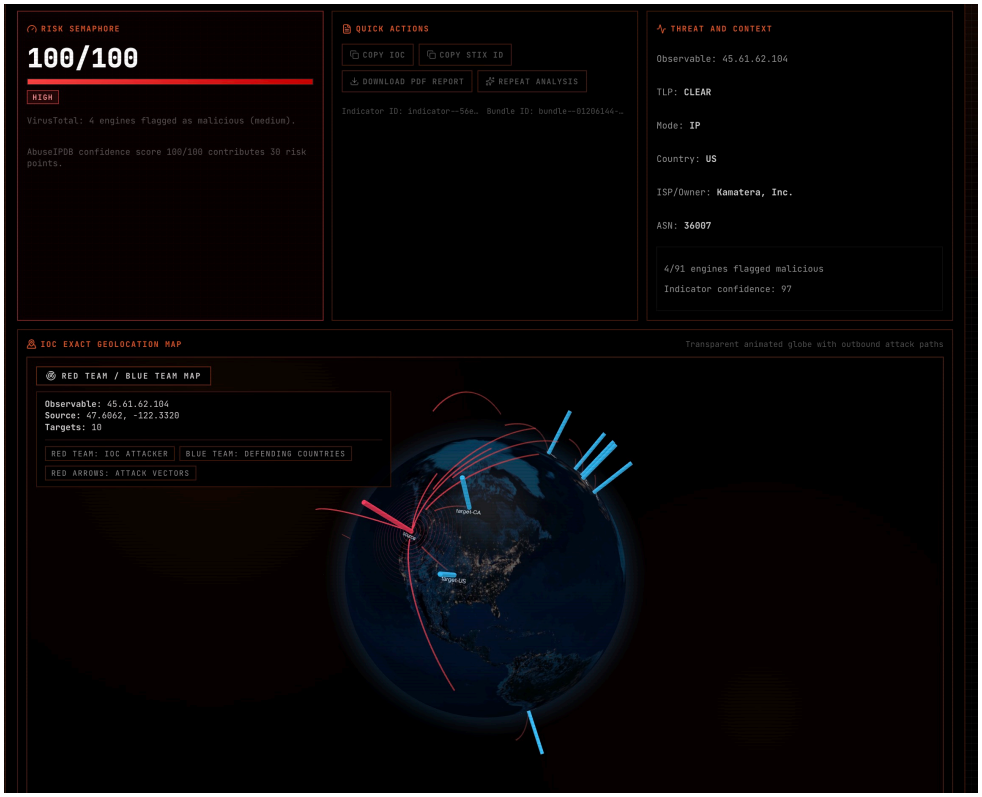


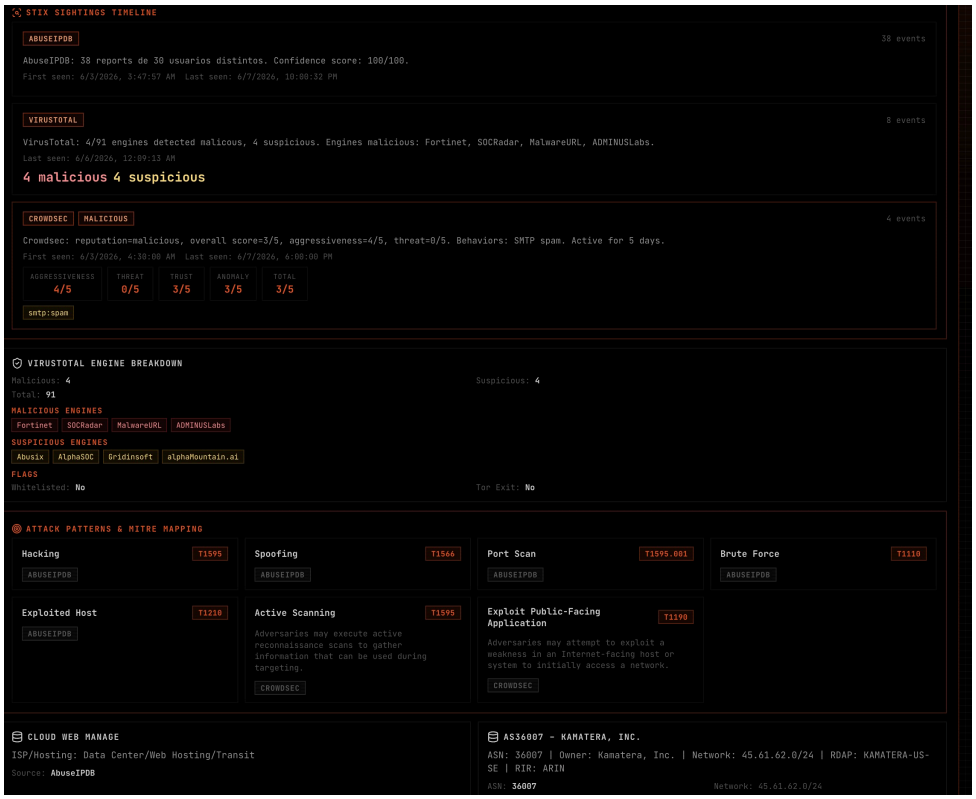
Figura 4.2: Diagrama de secuencia en notación UML de la consulta en lenguaje natural. El analista emite una única pregunta; el cliente LLM encadena de forma autónoma dos *tool calls*: primero `query_neo4j` para consultar el grafo de conocimiento vía `mcp-neo4j-bridge`, y después `analyze_observable` vía `mcp-intelowl-bridge`, que a su vez consulta las fuentes OSINT externas configuradas. La respuesta final combina el contexto del grafo con el resultado del análisis.

4.3 Muestra de la interfaz

La Figura 4.3 muestra dos capturas representativas del flujo de análisis automático de un observable: el informe de riesgo con veredicto operativo y contexto geográfico (arriba), y la línea temporal de detecciones con el desglose de veredictos por analizador (abajo).



(a) Skyfall Risk Score, veredicto operativo y contexto geográfico.



(b) Línea temporal de detecciones y desglose de veredictos por analizador.

Figura 4.3: Capturas representativas del análisis de una dirección IP.

Capítulo 5

Implementación

Este capítulo materializa sobre tecnologías concretas las decisiones del capítulo 3, describe la implementación, siguiendo el recorrido completo de una pieza de inteligencia a través de la plataforma. Tras justificar la elección de las tres tecnologías centrales y detallar el modelo de datos STIX 2.1, se sigue el ciclo de vida del dato. Primero se explica cómo se recogen las fuentes y se ingieren. Después, cómo se transportan de forma asíncrona a través de Apache Kafka hasta el motor de persistencia. A continuación, cómo se envían a enriquecer a IntelOwl y cómo se guardan en el grafo, incluyendo la fórmula de la puntuación de riesgo, y se publican en la instancia corporativa de OpenCTI. Seguidamente, cómo se consultan desde la interfaz web mediante la API central y mediante la capa MCP. Por último, cómo se habilita la búsqueda semántica sobre el grafo mediante generación aumentada por recuperación (RAG, del inglés *Retrieval-Augmented Generation*) [26] y cómo el analista puede crear entidades STIX de forma manual con STIX Forge.

5.1 Justificación tecnológica

La selección de las tres tecnologías centrales (base de datos, bus de mensajería y motor de enriquecimiento) se fundamentó en una evaluación comparativa. Para cada componente se examinaron las alternativas disponibles atendiendo a criterios de adecuación al dominio CTI, compatibilidad con STIX 2.1, peso operativo y disponibilidad en despliegue autoalojado.

La naturaleza relacional de la inteligencia de amenazas justifica el uso de una base de datos de grafos: un IOC se vincula a una IP, a una campaña, a técnicas y a actores, y estas conexiones son el objeto de consulta central. La tabla 5.1 compara las opciones evaluadas.

Tabla 5.1: Comparativa de soluciones de persistencia para el grafo CTI

Criterio	**Neo4j**	**PostgreSQL**	**Elasticsearch**
Consultas de relaciones	Nativo (Cypher)	JOINs recursivos	No nativo
Compatibilidad STIX 2.1	Alta	Parcial (JSON, del inglés *JavaScript Object Notation*)	Parcial (documentos)
Búsquedas de similitud	Integrado	pgvector (extensión)	Integrado
Peso operativo	Medio	Bajo	Alto
Visualización nativa	Neo4j Browser	—	Kibana

Neo4j [8] ofrece Cypher como lenguaje de consulta diseñado específicamente para navegar relaciones. La capacidad de expresar preguntas como «¿qué actores de amenaza utilizaron este IOC?» o «¿qué técnicas MITRE ATT&CK están asociadas a esta campaña?» mediante patrones de grafo resulta fundamentalmente más intuitiva y eficiente que cualquier alternativa relacional o documental [9]. Se adoptó como única capa de persistencia.

El requisito RNF2 de modularidad exige desacoplar los productores de inteligencia de los consumidores del grafo. La tabla 5.2 compara las alternativas de mensajería evaluadas.

Tabla 5.2: Comparativa de soluciones de mensajería para el desacoplamiento del flujo de mensajería

Criterio	**Apache Kafka**	**REST síncrono**	**RabbitMQ**
Desacoplamiento	Total	Ninguno	Alto
Durabilidad de mensajes	Persistente (disco)	Efímero	Configurable
Múltiples consumidores	Nativo (*groups*)	No	Parcial
Reprocesamiento	Sí (por posición de lectura)	No	No
Dependencias externas	Ninguna (KRaft)	Ninguna	Broker separado

Apache Kafka [27] permite que cada componente (crawlers, ingestores, IntelOwl, consumer-neo4j) evolucione, se reinicie o escale de forma independiente sin afectar a los demás. Los mensajes publicados en un tópico son durables y reprocesables en caso de error del consumidor, lo que añade resiliencia al sistema.

La alternativa a IntelOwl consistiría en implementar un cliente propio para cada fuente externa de enriquecimiento, lo que habría exigido mantener credenciales, clientes y lógica de reintento para decenas de APIs heterogéneas. Delegar esa responsabilidad en IntelOwl evita ese coste de mantenimiento: una única integración expone más de 300 analizadores ya soportados por la comunidad, devuelve sus resultados en un formato uniforme y centraliza la gestión de credenciales. Añadir una fuente nueva se reduce así a activar un analizador adicional, sin escribir ni mantener cliente alguno. La tabla 5.3 muestra las opciones barajadas.

Tabla 5.3: Comparativa entre IntelOwl y la integración directa con fuentes de enriquecimiento

Criterio	**IntelOwl**	**Integración directa**
Número de fuentes disponibles	+300 analizadores	1 cliente por fuente
Mantenimiento de integraciones	Externalizado	Interno
Formato de salida	JSON uniforme	Heterogéneo
Playbooks configurables	Sí	No aplicable
Gestión centralizada de credenciales	Sí	Distribuida

IntelOwl [3] centraliza la gestión de credenciales y la lógica de reintento de más de 300 analizadores y devuelve los resultados en un formato JSON uniforme. Skyfall-CTI activa más de 20 de ellos, concretamente los más relevantes para el análisis de IOCs.

5.2 Modelo de datos: STIX 2.1 en Neo4j

El modelo de datos de la plataforma adopta STIX 2.1 como lenguaje común y lo proyecta sobre el grafo de propiedades etiquetadas de Neo4j. STIX organiza la inteligencia en tres familias de objetos. Los objetos de dominio describen entidades analíticas, como indicadores, malware, vulnerabilidades, técnicas, campañas o actores. Los objetos observables representan artefactos técnicos concretos, como direcciones IP, dominios, URL o ficheros. Por último, los objetos de relación expresan los vínculos dirigidos entre los anteriores. Esta separación entre el dato técnico, su interpretación analítica y la relación que los conecta

encaja de forma natural con un grafo, en el que las dos primeras familias se materializan como nodos y la tercera como aristas dirigidas.

Cada tipo STIX 2.1 se mapea a una etiqueta semántica de Neo4j según la tabla 5.4. La proyección no es directa: cada tipo se traduce a una etiqueta válida y legible (e.g., `IP`, `Domain`, `Technique`), lo que permite formular consultas Cypher expresivas sobre nombres intuitivos en lugar de sobre los identificadores internos de los objetos STIX.

Tabla 5.4: Correspondencia entre tipos STIX 2.1 y etiquetas Neo4j en Skyfall-CTI.

Tipo STIX 2.1	**Etiqueta Neo4j**	**Fuentes principales**
`ipv4-addr` / `ipv6-addr`	`IP`	IntelOwl, OTX, feeds2stix
`domain-name`	`Domain`	IntelOwl, OTX, Threatview
`url`	`URL`	IntelOwl, telegram-crawler
`file`	`File`	IntelOwl (hashes MD5 (*Message-Digest Algorithm 5*) / SHA (*Secure Hash Algorithm*)-1 / SHA-256)
`email-addr`	`Email`	IntelOwl
`indicator`	`Indicator`	IntelOwl, OTX, telegram-crawler
`malware`	`Malware`	MITRE ATT&CK
`attack-pattern`	`Technique`	MITRE ATT&CK (T*xxxx*)
`tool`	`Tool`	MITRE ATT&CK
`threat-actor`	`ThreatActor`	MITRE ATT&CK, OTX
`intrusion-set`	`IntrusionSet`	MITRE ATT&CK
`campaign`	`Campaign`	MITRE ATT&CK
`vulnerability`	`Vulnerability`	CISA KEV, NVD
`course-of-action`	`Mitigation`	MITRE ATT&CK
`location`	`Location`	IntelOwl (geolocalización)
`identity`	`Identity`	Todas las fuentes
`report`	`Report`	OTX pulses, telegram-crawler
`sighting`	`Sighting`	IntelOwl
`note`	`Note`	IntelOwl

Las relaciones principales que conectan estos nodos son: `RESOLVES_TO` (dominio a IP), `RELATED_TO` (genérica entre observables), `USES` (ThreatActor a Malware o Technique), `INDICATES` (Indicator a Malware), `TARGETS` (Campaign a Location o Identity), `ATTRIBUTED_TO` (Campaign a ThreatActor), `REFERENCES` (Report a Indicator) y `CREATED_BY` (cualquier objeto a Identity). Cada arista conserva como propiedades el tipo de relación STIX original, un nivel de confianza y, cuando procede, un código de correlación que indica por qué dos entidades se vincularon, lo que permite auditar la procedencia de cada inferencia. La Figura 5.1 ilustra un fragmento representativo del modelo, en el que un observable recién analizado queda conectado con el conocimiento estructurado preexistente.

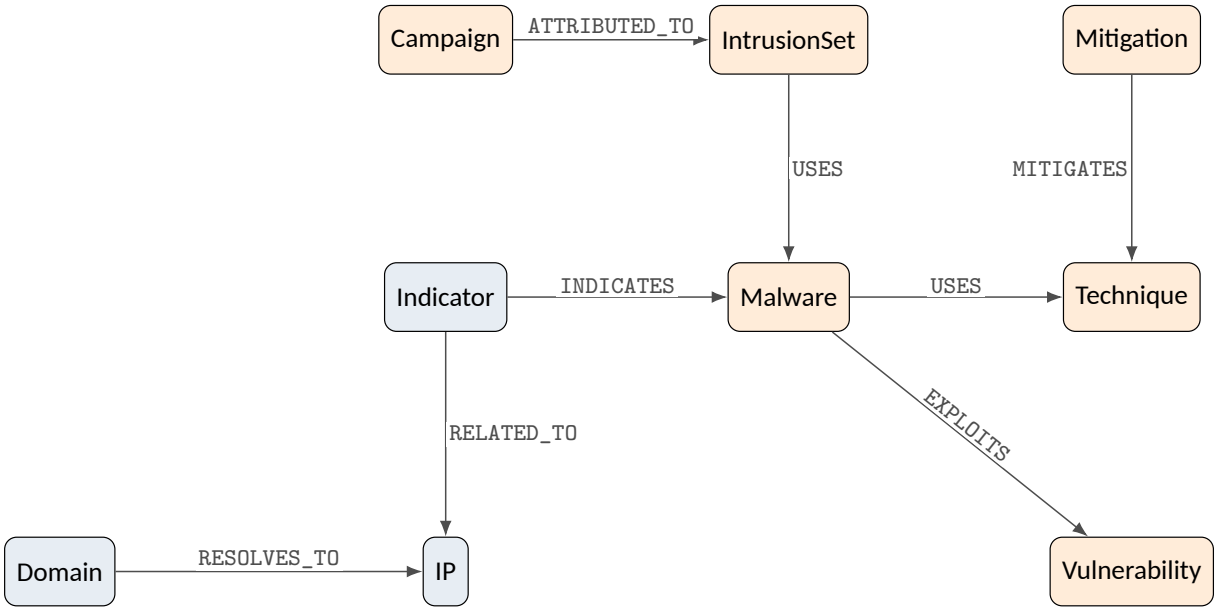


Figura 5.1: Fragmento del modelo de datos STIX 2.1 sobre Neo4j. Las aristas conservan el tipo de relación STIX original, su confianza y código de correlación.

Los resultados de los analizadores externos se persisten como propiedades extendidas con prefijo `x_`, como `x_vt_malicious`, `x_crowdsec_blocklists`, `x_mitre_id`, `x_otx_pulse` o `x_abuseipdb_abuse_confidence_score`. Este mecanismo de extensión permite enriquecer cada entidad con las métricas concretas de cada fuente sin alterar el esquema base ni perder la compatibilidad conceptual con la especificación STIX 2.1. Sobre estas propiedades se apoyan después tanto el cálculo de la puntuación de riesgo como las consultas de la interfaz. La plataforma dispone además de una visualización tridimensional interactiva del grafo de conocimiento accesible en línea [28].

5.3 Flujo de ingesta de inteligencia

La recopilación de inteligencia se estructura en torno a dos estrategias complementarias: la ingesta pasiva, que puebla el grafo de conocimiento de forma autónoma a partir de fuentes externas, y el enriquecimiento bajo demanda, que amplía el contexto de un observable concreto en el momento en que el analista lo solicita.

5.3.1 Fuentes periódicas orquestadas por n8n

La ingesta periódica de inteligencia se apoya en varios flujos de datos coordinados por n8n, el orquestador de flujos de trabajo incluido en la pila. n8n actúa como planificador externo: dispara las tareas de ingesta con la periodicidad configurada y puede encadenar pasos de transformación o enrutamiento antes de que los datos lleguen a Kafka.

OTX AlienVault. El servicio `mitre-ingestor` incluye, además de la carga masiva de MITRE ATT&CK (descrita en la sección 5.3.2), un bucle de sondeo periódico sobre la API de AlienVault OTX [29]. En cada ciclo obtiene la lista de pulsos suscritos modificados desde la última ejecución. Para cada pulso recupera hasta 100 indicadores y filtra únicamente los de tipo `IPv4`, `IPv6`, `domain` u `hostname`. Cada

indicador se convierte en un objeto `indicator` STIX 2.1 y el conjunto de indicadores de un mismo pulso se agrupa en un objeto `report` STIX. Finalmente, el paquete resultante se publica en el tópico Kafka `enrichment.results`.

***Fuentes* de IOC vía feeds2stix.** El servicio `feeds2stix` [30] mantiene un conjunto de procesadores de fuentes OSINT. Cada procesador descarga una fuente pública de amenazas, la normaliza y la convierte al formato STIX 2.1. Los paquetes resultantes se escriben en un directorio de salida compartido, monitorizado por el servicio `consumer-neo4j`, que los incorpora al grafo de manera autónoma en ciclos periódicos.

Se han integrado doce procesadores que cubren las principales familias de fuentes OSINT de reputación, *phishing* y distribución de malware, aunque la arquitectura del sistema permite que sea fácilmente extensible a más procesadores. La tabla 5.5 detalla cada uno con su fuente y el tipo de observable que produce.

Tabla 5.5: Procesadores de fuentes OSINT integradas en feeds2stix

Procesador	Fuente	Observable producido
`abuse_ch_malwarebazaar`	MalwareBazaar (abuse.ch)	Hashes de muestras de malware
`abuse_ch_threatfox`	ThreatFox (abuse.ch)	IOCs de C2 (IP, dominio, URL, hash)
`abuse_ch_urlhaus`	URLhaus (abuse.ch)	URL de distribución de malware
`abuse_ch_sslblacklist`	SSL Blacklist (abuse.ch)	Huellas de certificados TLS (del inglés, *Transport Layer Security*) maliciosos
`blocklist_de`	Blocklist.de	Direcciones IP atacantes
`certpl`	CERT Polska *warning list*	Dominios de *phishing*
`cinsscore`	CINS Army (cinsscore.com)	Direcciones IP maliciosas
`ipsum`	IPsum (lista agregada)	Direcciones IP maliciosas agregadas
`openphish`	OpenPhish	URL de *phishing*
`phishtank`	PhishTank	URL de *phishing* verificadas
`threatview`	ThreatView.io	IOCs (IP de C2, dominios, hashes)
`vxvault`	VX Vault	URL de distribución de malware

Flujos de orquestación. n8n programa cuatro flujos de ingesta periódica mediante disparadores temporales:

- Dominios maliciosos. Descarga y publica en Kafka la lista de Amitamkhar510 [31], un repositorio público de dominios maliciosos conocidos.
- IPs comprometidas. Obtiene y publica el fichero de Emerging Threats [32], una lista de direcciones IP comprometidas.
- Vulnerabilidades explotadas (KEV). El servicio `kevin-api` [33] consolida el catálogo CISA, la puntuación CVSS de la NVD y las fichas de técnicas MITRE ATT&CK asociadas. n8n interroga periódicamente este servicio, construye un objeto `vulnerability` STIX por cada entrada nueva o actualizada y lo publica en Kafka para su persistencia en Neo4j.
- Síntesis Grok/xAI. El flujo conecta n8n, a través del *gateway* Vercel AI [34], con el modelo Grok [35] para consultar amenazas activas en lenguaje natural y persiste el resultado como objetos `report` STIX 2.1 en el grafo.

5.3.2 Fuentes de ingesta masiva

Ciertas fuentes presentan un carácter estático o requieren una descarga masiva inicial. Para estas fuentes se crearon contenedores Python autónomos que ejecutan su lógica una sola vez al iniciarse.

MITRE ATT&CK. El servicio `mitre-ingestor` descarga los tres dominios de la base de conocimiento MITRE ATT&CK [36] en formato STIX 2.1. El ingestor espera cierto tiempo a que Neo4j esté disponible y a continuación carga todos los objetos mediante sentencias `MERGE` agrupadas por lotes de 1.000 nodos, asignando a cada `attack-pattern` la etiqueta `Technique` y almacenando el identificador ATT&CK en la propiedad `x_mitre_id`.

Dataset CTI de Telegram vía GitHub. El servicio `telegram-crawler` actúa como un *fetcher* del repositorio público `OPTIMA-CTI/Telegram-CTI` [37]. El servicio enumera los ficheros del directorio de IOCs mediante la API de GitHub y descarga únicamente los ficheros nuevos o modificados. Para cada fila de la hoja de cálculo, infiere el tipo de observable y construye un paquete STIX compuesto por un `identity` estable, un `indicator` con el patrón STIX correspondiente y un `report` que registra el canal de procedencia.

5.3.3 El consumidor universal: consumer-neo4j

Todos los paquetes STIX producidos por las rutas periódicas convergen en `consumer-neo4j`, un servicio Python que lee los mensajes de Apache Kafka y vigila un directorio de salida compartido. Al iniciar, el consumidor crea en Neo4j un conjunto de restricciones de unicidad por tipo semántico e índices auxiliares sobre propiedades frecuentemente consultadas para garantizar la deduplicación y la velocidad de las consultas Cypher.

Cada mensaje Kafka contiene un paquete STIX serializado como JSON. El consumidor deserializa el paquete, separa los objetos en nodos y aristas, y los carga garantizando idempotencia: si el nodo ya existe simplemente actualiza sus propiedades sin crear duplicados. El tipo STIX se mapea a una etiqueta Neo4j semántica según la tabla 5.4.

Motor de correlación automática. La aportación diferencial de `consumer-neo4j` no es la mera persistencia, sino el motor de correlación que ejecuta tras cargar cada paquete: infiere relaciones nuevas entre los objetos recién llegados y el conocimiento preexistente, convirtiendo observables aislados en una red de evidencias conectadas. Cada arista inferida se etiqueta con un código de correlación `C-*`, un nivel de confianza y su fuente, lo que permite auditar después por qué se creó cada vínculo.

El proceso parte de una fase de normalización (consolidación de IOCs duplicados, reconstrucción de los nombres de CVEs y enumeraciones de debilidades (CWE, del inglés *Common Weakness Enumeration*), canonicalización de técnicas por su identificador ATT&CK y materialización de nodos de país, campaña e infraestructura) y, sobre ella, aplica varias familias de reglas: deduplicación de entidades equivalentes (`SAME_AS`), anclaje de IOCs y malware a técnicas MITRE ATT&CK (`EXHIBITS`), correlación de vulnerabilidades con software, actores y campañas a través de su CWE, reconstrucción de cadenas de infección (indicador $\rightarrow$ malware $\rightarrow$ vulnerabilidad $\rightarrow$ software) y agrupación geográfica por país de origen. La Figura 5.2 ilustra, para el caso concreto de una dirección IP recién ingerida, las nuevas relaciones que el motor de correlación es capaz de crear.

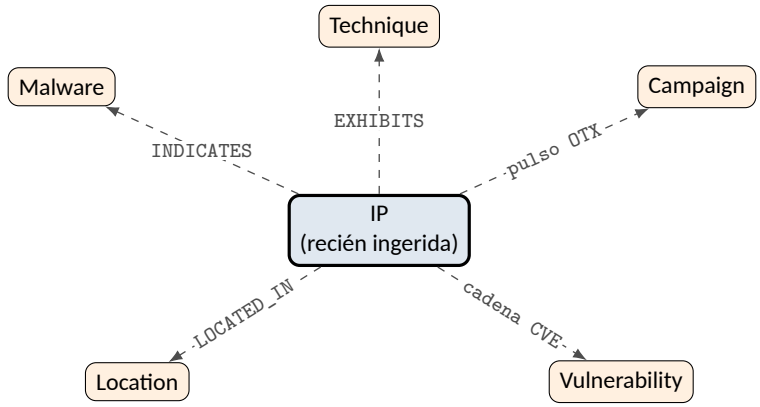


Figura 5.2: Ejemplo de las relaciones (en línea discontinua) que el motor de correlación infiere de forma automática a partir de una dirección IP recién ingerida, conectándola con el conocimiento estructurado preexistente.

5.4 Enriquecimiento de observables bajo demanda

Una vez poblado el grafo de forma autónoma con la ingesta pasiva, los observables que el analista investiga se enriquecen bajo demanda siguiendo el flujo que se describe a continuación. Cada IOC consultado se analiza en el momento contra múltiples fuentes externas, se persiste en el grafo y se correlaciona con el conocimiento ya almacenado, de modo que cada investigación amplía la memoria de inteligencia común.

5.4.1 Flujo de análisis con IntelOwl

Cuando el analista introduce un observable en la interfaz web, `skyfall-api` delega el análisis en IntelOwl mediante su API REST. IntelOwl ejecuta en paralelo los analizadores configurados (VirusTotal [38], AbuseIPDB [39], CrowdSec [40], Shodan, URLScan, GreyNoise [41], entre otros) y devuelve los resultados brutos de cada fuente. `intelowl-client`, el servicio intermediario de la pila, convierte esos resultados a un paquete STIX estructurado y lo publica en el tópico `enrichment.results` de Kafka, de modo que también queda persistido en Neo4j para consultas futuras. Paralelamente, `skyfall-api` calcula una puntuación de riesgo propia ponderando las señales de reputación.

Selección de perfil de análisis por tipo de observable. El cliente `intelowl-client` no ejecuta un conjunto fijo de analizadores, sino que identifica primero el tipo del observable (IP, dominio, URL o hash) y selecciona automáticamente el perfil de análisis de IntelOwl adecuado para ese tipo concreto de observable. La tabla 5.6 recoge la correspondencia entre tipos y perfiles. La concurrencia de análisis simultáneos se limita internamente para evitar saturar el servicio de enriquecimiento.

Tabla 5.6: Playbooks IntelOwl y analizadores por tipo de observable

Observable	Playbook	Analizadores principales
IP	`SkyfallCTIipReputation`	VirusTotal, AbuseIPDB, CrowdSec, Shodan, GreyNoise, FireHol, ThreatFox, URLhaus, UrlScan
Dominio / URL	`SkyfallCTIurlreputation`	UrlScan, CloudFlare Malicious Detector, VirusTotal, resolución DNS (del inglés, *Domain Name System*) de IP asociadas
Hash	`SkyfallCTIHashReputation`	VirusTotal, MalwareBazaar, Triage

Conversión a STIX y encadenamiento. Una vez finalizada la tarea, `intelowl-client` delega la traducción de los resultados brutos a STIX 2.1 en un conversor especializado por tipo. Cada conversor recorre los resultados de la tarea, extrae las señales relevantes de cada analizador y genera objetos `indicator`, `sighting` y `note`, además de la representación STIX del propio observable. Para dominios y URL, el conversor resuelve las direcciones IP asociadas y crea relaciones `resolves-to`; cada IP resuelta se reanaliza encadenando una nueva tarea con el *playbook* de IP, de modo que un único dominio puede arrastrar al grafo toda su infraestructura. Sobre estos datos se construye la puntuación de riesgo y, si está habilitada, la publicación en OpenCTI (sección 5.4.3).

5.4.2 Sistema de puntuación de riesgo Skyfall

El sistema de puntuación de riesgo es una métrica compuesta calculada íntegramente en el servidor que pondera señales de reputación heterogéneas en una escala [0, 100]: las detecciones de VirusTotal, la confianza y los informes de abuso de AbuseIPDB, los patrones de CrowdSec, la confianza del propio indicador y los veredictos de otros motores. Cada señal contribuye con un máximo de puntos acotado, la suma se limita al rango [0, 100] y, sobre el resultado, se asigna un nivel categórico (*low*, *medium* o *high*). La formulación matemática completa, con la función de escalado de VirusTotal, el peso de cada señal y los umbrales de nivel, se recoge en la sección B.3.

La puntuación no sustituye al criterio del analista, pero permite ordenar automáticamente las investigaciones y destacar los observables de mayor prioridad en el cuadro de mando.

5.4.3 Publicación en OpenCTI corporativo

La integración con la instancia OpenCTI corporativa (RF10) se resolvió tras valorar dos vías y descartar el conector nativo IntelOwl → OpenCTI en favor de exportar el paquete STIX 2.1 completo que el cliente ya genera.

La vía adoptada reutiliza el paquete existente y lo empuja a OpenCTI mediante la biblioteca oficial `pycti` [42]. La lógica se reparte en dos módulos. El módulo `opencti_client.py` actúa como capa de acceso a `pycti.OpenCTIApiClient`: gestiona la conexión con OpenCTI, cachea las etiquetas y clasificaciones TLP para evitar consultas redundantes, y establece la identidad autora de cada importación. El módulo `opencti_integration.py` contiene la lógica específica de Skyfall: antes de publicar, añade a cada objeto del paquete la clasificación TLP y la autoría de la identidad «Skyfall IntelOwl»; tras la importación, etiqueta cada indicador con el nombre del analizador que lo confirmó y el veredicto final.

La publicación está controlada por la variable de entorno `OPENCTI_PUSH_ENABLED`; si está desactivada, no se ejecuta. Para no bloquear el bucle de eventos asíncrono, el envío se ejecuta en segundo plano

y cualquier excepción se captura y registra sin abortar el análisis ni la persistencia local, conforme a la condición de desacoplamiento fijada en el diseño.

5.5 Interfaz web del analista

La interfaz de Skyfall-CTI es una aplicación web construida con Next.js [43] que concentra todas las capacidades de la plataforma en un único punto de acceso. La tabla 5.7 resume las rutas disponibles y la función de cada una.

Tabla 5.7: Rutas principales de la plataforma web Skyfall-CTI

Ruta	Función principal
`/`	Pantalla de inicio con análisis rápido de IOCs, navegación y noticias.
`/intelowl/analyze-ip`	Análisis de direcciones IP con reputación, geolocalización y relaciones en el grafo.
`/intelowl/analyze-hash`	Análisis de hashes con identificación de familia de malware y correlaciones.
`/intelowl/analyze-domain`	Análisis de dominios y URLs con resolución de infraestructura asociada.
`/explore`	Explorador en vivo de indicadores almacenados en el grafo y sus relaciones.
`/dashboards`	Cuadros de mando de IOCs, CVEs, malware, técnicas y estado del grafo.
`/chat`	Asistente CTI en lenguaje natural con modos de consulta general, observable y grupo APT.
`/explorer`	Explorador del grafo embebido con acceso de solo lectura.
`/news`	Noticias de ciberseguridad con filtros y actualización manual.
`/build`	STIX Forge: construcción visual y persistencia en Neo4j de paquetes STIX 2.1 con inferencia en tiempo real.
`/graph`	Visualización tridimensional e interactiva del grafo de conocimiento STIX 2.1.

5.5.1 STIX Forge: creación manual de entidades STIX en Neo4j

El módulo *STIX Forge* permite al analista construir y persistir objetos STIX 2.1 en Neo4j de forma manual, sin necesidad de conocer la sintaxis JSON del estándar. La interfaz, cuyo lienzo visual se inspiró en herramientas de diagramación como draw.io [44], se organiza en tres paneles: una paleta con 18 tipos de entidades STIX, un lienzo central donde el analista sitúa entidades y traza relaciones entre ellas, y un panel lateral de inferencia en tiempo real. Cada entidad se configura mediante un formulario guiado que cubre los campos del tipo seleccionado; las relaciones entre nodos se establecen directamente sobre el lienzo eligiendo el tipo de vínculo. Cuando el grafo de inteligencia supera la validación del estándar, se envía como paquete a Neo4j y se abre el teatro de inferencia, que muestra en directo el avance de las fases de autocorrelación hasta completar la ingestión.

Aunque *STIX Forge* es una vía de creación manual, los objetos que produce no quedan aislados: tras su ingestión en el grafo, el motor de correlación los vincula con el conocimiento preexistente del mismo modo que los procedentes del enriquecimiento automático. Así, una técnica MITRE ATT&CK añadida manualmente puede quedar conectada con los grupos APT que ya la emplean, un *malware* creado a mano se relaciona con las campañas y vulnerabilidades conocidas, y un indicador registrado por el analista se cruza con los resultados de IntelOwl si ese observable fue analizado con anterioridad. Una demostración interactiva del módulo puede visitarse en línea [45].

5.6 Uso de LLMs

La plataforma integra modelos de lenguaje en dos capacidades complementarias: la búsqueda semántica sobre el grafo y la consulta interactiva en lenguaje natural. Ambas pueden operar de forma local o conectadas a la infraestructura corporativa.

Vectorización y búsqueda semántica. Para posibilitar las búsquedas semánticas, los nodos más relevantes del grafo (técnicas, malware, mitigaciones, conjuntos de intrusión, vulnerabilidades, herramientas y campañas) se vectorizan: se genera una representación numérica del nombre y la descripción de cada entidad y se almacena junto al nodo en siete índices vectoriales. Cuando el analista formula una consulta en lenguaje natural, la plataforma la convierte al mismo espacio vectorial y devuelve las entidades del grafo semánticamente más próximas.

El modelo de *embeddings* se selecciona según el entorno. En entorno local se emplea Qwen3-Embedding-4B [46] servido desde LM Studio [19], sin necesidad de conexión externa. En entorno corporativo, los vectores se generan con Qwen3-Embedding-8B [47], de mayor capacidad, a través del *gateway* Bifrost [48], con la ventaja de ofrecer mayor calidad de recuperación semántica.

Integración con modelos de lenguaje. El analista interactúa con Skyfall-CTI en lenguaje natural a través de la ruta `/chat`, empleando cualquier LLM compatible con el MCP. La plataforma soporta dos vías de conexión. La primera emplea LM Studio [19] con el modelo Qwen3-4B ejecutado en local, lo que permite operar sin conexión externa y sin enviar datos fuera de la máquina del analista. La segunda conecta con el Bifrost corporativo [48], que sirve el modelo Qwen3-27B [49] mediante vLLM, para quienes requieran mayor capacidad de razonamiento.

Arquitectura MCP. La capa MCP permite a los modelos de lenguaje interactuar con la plataforma a través de herramientas estructuradas. Se compone de tres elementos: dos puentes especializados y una puerta de acceso con autenticación.

El puente `mcp-neo4j-bridge` pone a disposición del modelo la capacidad de consultar el grafo de conocimiento en lenguaje natural, traduciéndolo internamente a consultas Cypher. El puente `mcp-intelowl-bridge` permite al modelo enviar un observable para su análisis contra las fuentes externas configuradas y recibir un resumen estructurado con el veredicto y los hallazgos de cada analizador.

Ambos puentes son internos a la plataforma y solo accesibles a través de la puerta de acceso `mcp-gateway`, que verifica la identidad del cliente antes de redirigir la petición. Este diseño permite que tanto LM Studio como el Bifrost corporativo se conecten a Skyfall-CTI como si fuera un servidor MCP remoto estándar. La Figura 5.3 resume esta arquitectura.

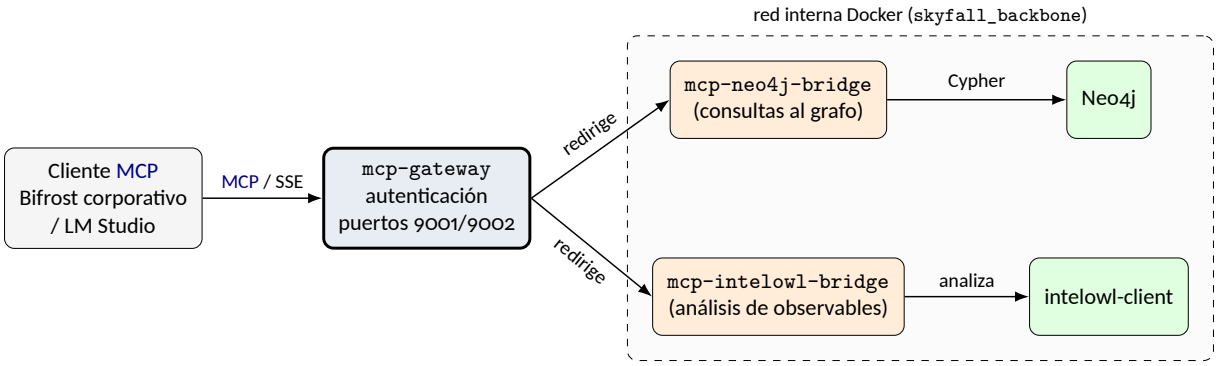


Figura 5.3: Arquitectura de la capa MCP. La puerta de acceso es el único componente visible desde el exterior; autentica cada petición y la redirige al puente correspondiente, que opera dentro de la red interna de Docker.

5.7 Validación e integración continua

Skyfall-CTI cuenta con un flujo de integración continua en GitHub Actions que se activa automáticamente en cada cambio sobre las ramas principales y en cada solicitud de integración, organizado en tres etapas encadenadas:

1. Validación de la configuración. Comprueba la sintaxis del fichero de composición de contenedores sin arrancar ningún servicio; si falla, las siguientes etapas se cancelan.
2. Construcción de imágenes. Construye en paralelo las imágenes de todos los servicios personalizados, detectando errores antes de levantar el *stack* completo.
3. Integración. Arranca el *stack* completo y ejecuta la batería de pruebas de integración; en caso de fallo, vuelca los registros de cada servicio para facilitar el diagnóstico.

Los tres trabajos se ejecutan sobre ejecutores alojados por GitHub con Docker Engine preinstalado. Al finalizar, contenedores y volúmenes se eliminan para garantizar ejecuciones reproducibles e independientes.

La batería cubre los servicios críticos del *stack* mediante más de 36 pruebas ejecutadas contra el entorno Docker Compose real: la conectividad y las consultas de Neo4j y Kafka, los componentes internos de IntelOwl (base de datos, caché, servidor de aplicación y procesos de cola), la conversión de resultados a STIX 2.1 y la detección de veredictos maliciosos en `intelowl-client`, la disponibilidad de los *workers* Python auxiliares, la accesibilidad de la interfaz web y el aislamiento de la red interna entre servicios.

Capítulo 6

Conclusiones y trabajo futuro

En este Trabajo de Fin de Grado se ha construido un repositorio de inteligencia de amenazas autoalojado que cubre el ciclo completo de la ciberinteligencia (ingesta, enriquecimiento, normalización, persistencia, correlación, consulta y visualización), cumpliendo con todos los objetivos planteados. La combinación de Apache Kafka, Neo4j, STIX 2.1, IntelOwl y Docker Compose se ha mostrado como una base sólida y modular, y la correlación en grafo ha demostrado convertir indicadores aislados en conocimiento persistente, reutilizable y vinculado a vulnerabilidades, actores, técnicas y evidencias previas. Las aportaciones propias más relevantes son la arquitectura distribuida de extremo a extremo, el cliente asíncrono de IntelOwl, el consumidor universal de paquetes STIX hacia Neo4j, el cálculo del sistema de puntuación de riesgo, la publicación de resultados en OpenCTI y la consulta agéntica del grafo mediante MCP. Las principales limitaciones se derivan de la dependencia de las cuotas y la disponibilidad de las fuentes externas, del arranque en frío de IntelOwl y de la necesidad de calibrar la puntuación de riesgo con casos reales antes de su uso en producción.

Como trabajo futuro se plantea ampliar la evaluación experimental con conjuntos de datos etiquetados y métricas reproducibles, reforzar el flujo de ingesta con validación automática de fuentes, detección de duplicados semánticos, expiración de indicadores obsoletos y soporte TAXII (del inglés, *Trusted Automated eXchange of Indicator Information*), endurecer la seguridad y la observabilidad de la plataforma mediante autenticación centralizada, roles, auditoría de consultas y paneles de salud, y profundizar en la integración con LLMs locales a través del MCP, priorizando la trazabilidad de cada respuesta hasta los nodos, las relaciones y las fuentes concretas del grafo.

Declaración de uso responsable de inteligencia artificial generativa

Durante la elaboración de este Trabajo de Fin de Grado se han utilizado herramientas de inteligencia artificial generativa (concretamente modelos de lenguaje de gran tamaño) como apoyo en tareas de redacción, revisión ortográfica y gramatical, búsqueda de referencias y depuración de código.

El autor declara que todo el contenido técnico, el diseño de la arquitectura, el análisis de resultados y las conclusiones son de su autoría, y que el uso de dichas herramientas se ha limitado a un papel auxiliar que en ningún momento ha sustituido el criterio, el razonamiento ni la toma de decisiones propios.

El autor asume plena responsabilidad sobre la veracidad, la corrección y la integridad de todo el contenido presentado en este documento.

Referencias

[1] T. Riebe, D. Wermke y C. Wressnegger, "Towards Comprehensive Cyber Threat Intelligence: Taxonomies, Automation and Graph-Based Analysis", *Applied Sciences*, vol. 13, n.º 14, pág. 8150, 2023. DOI: 10.3390/app13148150

[2] OASIS Open. "STIX™ Version 2.1 — OASIS Standard", OASIS Open, visitado 10 de abr. de 2026. dirección: https://docs.oasis-open.org/cti/stix/v2.1/stix-v2.1.html

[3] IntelOwl Project. "IntelOwl: Analyze Files, Domains, IPs in Multiple Ways from a Single API at Scale", IntelOwl Project, visitado 15 de abr. de 2026. dirección: https://intelowlproject.github.io/

[4] Filigran. "OpenCTI: Open Cyber Threat Intelligence Platform", Filigran, visitado 5 de abr. de 2026. dirección: https://filigran.io/solutions/open-cti/

[5] Anthropic. "Model Context Protocol: Introduction", Anthropic, visitado 20 de abr. de 2026. dirección: https://modelcontextprotocol.io/introduction

[6] Palo Alto Networks. "Cortex XSIAM: Extended Security Intelligence and Automation Management", Palo Alto Networks, visitado 18 de mayo de 2026. dirección: https://www.paloaltonetworks.com/cortex/cortex-xsiam

[7] E. Sharma, A. Fehnker y N. Matteucci, "IntelOwl: Orchestrate your Threat Intelligence Operations at Scale", *arXiv preprint*, 2021. arXiv: 2110.05520. dirección: https://arxiv.org/abs/2110.05520

[8] Neo4j, Inc. "Neo4j Graph Database Platform", Neo4j, Inc., visitado 1 de mayo de 2026. dirección: https://neo4j.com/

[9] Neo4j, Inc. "Graph Databases for Cybersecurity: Threat Detection and Intelligence", Neo4j, Inc., visitado 10 de mar. de 2026. dirección: https://neo4j.com/use-cases/fraud-detection/

[10] A. Patel. "StixToNeoDB: Importing STIX2 Objects into Neo4j", GitHub, visitado 25 de mar. de 2026. dirección: https://github.com/signalscorps/stix2-neo4j

[11] Idaho National Laboratory. "STIG: Structured Threat Intelligence Graph — Documentation", Idaho National Laboratory, visitado 25 de mar. de 2026. dirección: https://stig.inl.gov/docs/

[12] Idaho National Laboratory. "STIG: Structured Threat Intelligence Graph", Idaho National Laboratory, visitado 25 de mar. de 2026. dirección: https://stig.inl.gov/

[13] Cosive. "MISP vs. OpenCTI: Updated 2025 Guide", Cosive, visitado 5 de abr. de 2026. dirección: https://www.cosive.com/misp-vs-opencti

[14] D. Vasquez. "Integrating OpenCTI and MISP: A Practical Guide", Medium, visitado 1 de abr. de 2026. dirección: https://medium.com/@danielvasquez/integrating-opencti-and-misp

[15] MISP Project. "MISP — Open Source Threat Intelligence Platform and Open Standards for Threat Information Sharing", MISP Project, visitado 8 de abr. de 2026. dirección: https://www.misp-project.org/

[16] RootSwarm. "OpenCTI vs MISP: Choosing the Right Threat Intelligence Platform", RootSwarm, visitado 8 de abr. de 2026. dirección: https://rootswarm.com/opencti-vs-misp

[17] MISP Project. “MISP-STIX — Conversion Library between MISP and STIX Formats”, GitHub / MISP Project, visitado 10 de abr. de 2026. dirección: https://github.com/MISP/misp-stix

[18] S. Brown, *The C4 Model for Visualising Software Architecture*. Leanpub, 2019. visitado 11 de jun. de 2026. dirección: https://leanpub.com/visualising-software-architecture

[19] LM Studio. “LM Studio: Discover, Download and Run Local LLMs”, LM Studio, visitado 20 de mayo de 2026. dirección: https://lmstudio.ai/

[20] MITRE Corporation. “MITRE ATT&CK® : Adversarial Tactics, Techniques and Common Knowledge”, MITRE Corporation, visitado 10 de abr. de 2026. dirección: https://attack.mitre.org/

[21] Cybersecurity and Infrastructure Security Agency. “Known Exploited Vulnerabilities Catalog”, CISA, visitado 20 de mar. de 2026. dirección: https://www.cisa.gov/known-exploited-vulnerabilities-catalog

[22] National Institute of Standards and Technology. “NVD Vulnerability API v2.0 — Developer Documentation”, NIST, visitado 10 de abr. de 2026. dirección: https://nvd.nist.gov/developers/vulnerabilities

[23] n8n GmbH. “n8n Docs: Workflow Automation and Integrations”, n8n GmbH, visitado 18 de mayo de 2026. dirección: https://docs.n8n.io/

[24] S. Ramírez. “FastAPI: A Modern, Fast (High-Performance) Web Framework for Building APIs with Python”, FastAPI, visitado 1 de mayo de 2026. dirección: https://fastapi.tiangolo.com/

[25] NGINX, Inc. “NGINX: High Performance Load Balancer, Web Server and Reverse Proxy”, NGINX, visitado 1 de mayo de 2026. dirección: https://nginx.org/en/

[26] P. Lewis et al., “Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks”, vol. 33, págs. 9459-9474, 2020. dirección: https://arxiv.org/abs/2005.11401

[27] Apache Software Foundation. “Apache Kafka: A Distributed Streaming Platform”, Apache Software Foundation, visitado 1 de mayo de 2026. dirección: https://kafka.apache.org/

[28] C. Solana Melero. “Skyfall-CTI — Visualización 3D interactiva del grafo de conocimiento”, Universidad de Zaragoza, EINA, visitado 12 de jun. de 2026. dirección: https://skyfall-landing.carlossolanamelero.workers.dev/graph

[29] AT&T Cybersecurity. “AlienVault Open Threat Exchange (OTX): Free and Open Threat Intelligence Community”, AT&T Cybersecurity, visitado 15 de abr. de 2026. dirección: https://otx.alienvault.com/

[30] C. Solana Melero. “feeds2stix: Modular Feed-to-STIX Conversion Framework”, Skyfall-CTI Project, visitado 18 de mayo de 2026. dirección: https://github.com/Csolanascript/feeds2stix

[31] Amitamkhar510. “Malicious-Domain-Threat-List: Curated List of Known Malicious Domains”, GitHub, visitado 11 de jun. de 2026. dirección: https://github.com/amitamkhar510/Malicious-Domain-Threat-List

[32] Proofpoint / Emerging Threats. “compromised-ips.txt: List of Compromised IP Addresses”, Emerging Threats, visitado 11 de jun. de 2026. dirección: https://rules.emergingthreats.net/blockrules/compromised-ips.txt

[33] S. Finner. “KEVin: A Flask API for the CISA Known Exploited Vulnerabilities Catalog”, synfinner, visitado 18 de mayo de 2026. dirección: https://github.com/synfinner/KEVin

[34] Vercel. “Vercel AI SDK: Build AI-Powered Applications”, Vercel, visitado 11 de jun. de 2026. dirección: https://sdk.vercel.ai/

[35] xAI. “Grok: Generative AI by xAI”, xAI, visitado 11 de jun. de 2026. dirección: https://x.ai/grok

[36] MITRE Corporation. “ATT&CK Workbench TAXII Server — STIX 2.1 Collections”, MITRE Corporation, visitado 12 de abr. de 2026. dirección: https://github.com/mitre-attack/attack-workbench-taxii-server

[37] OPTIMA-CTI. “Telegram-CTI: IoC Dataset Extracted from Cybersecurity Telegram Channels”, OPTIMA-CTI, visitado 18 de mayo de 2026. dirección: https://github.com/OPTIMA-CTI/Telegram-CTI

[38] Google LLC. “VirusTotal: Analyze Suspicious Files, Domains, IPs and URLs”, Google / VirusTotal, visitado 15 de abr. de 2026. dirección: https://www.virustotal.com/

[39] AbuseIPDB. “AbuseIPDB: IP Address Abuse Reports”, AbuseIPDB, visitado 15 de abr. de 2026. dirección: https://www.abuseipdb.com/

[40] CrowdSec. “CrowdSec: Open Source & Participative IPS and Threat Intelligence”, CrowdSec, visitado 15 de abr. de 2026. dirección: https://www.crowdsec.net/

[41] GreyNoise Intelligence. “GreyNoise: Internet Background Noise Analysis”, GreyNoise Intelligence, visitado 18 de abr. de 2026. dirección: https://www.greynoise.io/

[42] Filigran. “pycti: Python client for the OpenCTI platform”, Filigran, visitado 3 de jun. de 2026. dirección: https://github.com/OpenCTI-Platform/client-python

[43] Vercel. “Next.js: The React Framework for the Web”, Vercel, visitado 1 de mayo de 2026. dirección: https://nextjs.org/

[44] JGraph Ltd. “draw.io (diagrams.net) — Online Diagramming Tool”, JGraph Ltd., visitado 8 de jun. de 2026. dirección: https://www.drawio.com/

[45] C. Solana Melero. “Skyfall-CTI — STIX Forge: demostración interactiva del editor visual STIX 2.1”, Universidad de Zaragoza, EINA, visitado 12 de jun. de 2026. dirección: https://skyfall-landing.carlossolanamelero.workers.dev/forge

[46] Qwen Team, Alibaba Cloud. “Qwen3-Embedding-4B: A Unified Text Embedding Model”, Alibaba Cloud, visitado 20 de mayo de 2026. dirección: https://huggingface.co/Qwen/Qwen3-Embedding-4B

[47] Qwen Team, Alibaba Cloud. “Qwen3-Embedding-8B: A Unified Text Embedding Model”, Alibaba Cloud, visitado 20 de mayo de 2026. dirección: https://huggingface.co/Qwen/Qwen3-Embedding-8B

[48] Anthropic. “Model Context Protocol — GitHub Repository”, GitHub / Anthropic, visitado 20 de abr. de 2026. dirección: https://github.com/modelcontextprotocol/

[49] Qwen Team, Alibaba Cloud. “Qwen3-27B: Large Language Model”, Alibaba Cloud, visitado 11 de jun. de 2026. dirección: https://huggingface.co/Qwen/Qwen3-27B

Lista de acrónimos

API *Application Programming Interface*. 6, 8, 15, 21, 24, 26, 27

APT *Advanced Persistent Threat*. 2, 18, 29

CISA *Cybersecurity and Infrastructure Security Agency*. 25

CTI *Cyber Threat Intelligence*. 1–4, 21, 29

CVE *Common Vulnerabilities and Exposures*. 2, 12, 14, 26, 29

CVSS *Common Vulnerability Scoring System*. 25

CWE *Common Weakness Enumeration*. 26

IOC *Indicator of Compromise*. 1, 2, 4, 6, 8, 9, 14, 21, 22, 25–27, 29

JSON *JavaScript Object Notation*. 22, 26, 29

KEV *Known Exploited Vulnerabilities*. 14, 18, 25

LLM *Large Language Models*. 3, 10, 12, 15, 17–19, 30, 32

MCP *Model Context Protocol*. 1–3, 8, 10–12, 15, 17, 18, 21, 30–32, III

MISP *Malware Information Sharing Platform*. 5, 6, 43

MITRE ATT&CK *MITRE Adversarial Tactics, Techniques and Common Knowledge*. 14, 16, 21, 25, 29

NVD *National Vulnerability Database*. 25

OASIS *Organization for the Advancement of Structured Information Standards*. 17

OSINT *Open Source Intelligence*. 2, 3, 9–11, 14, 16, 17, 19, 25, 38, VI

OTX *Open Threat Exchange*. 14, 23, 24

PDF *Portable Document Format*. 8

STIX *Structured Threat Information Expression*. 2–5, 8, 9, 14–17, 21–29, 31, 32, 43, 46

TFG Trabajo de Fin de Grado. 2

TLP *Traffic Light Protocol*. 28

URL *Uniform Resource Locator*. 29

XSIAM *Extended Security Intelligence and Automation Management*. 2

Glosario de términos

Término	Definición
Amenaza persistente avanzada (APT)	Actor de amenaza con recursos, motivación y capacidad para desarrollar campañas de ataque prolongadas y dirigidas contra objetivos específicos. Generalmente patrocinado por estados-nación u organizaciones con financiación significativa.
Apache Kafka	Plataforma de mensajería distribuida que actúa como bus asíncrono entre productores y consumidores. En Skyfall-CTI se despliega en modo KRaft y constituye la columna vertebral de comunicación entre microservicios.
API (Application Programming Interface)	Interfaz que expone un conjunto de operaciones para que un programa interactúe con un servicio. La `skyfall-api` es la API REST central de la plataforma.
ASGI (Asynchronous Server Gateway Interface)	Especificación de interfaz entre servidores web y aplicaciones Python asíncronas. La emplea Daphne para servir la interfaz de IntelOwl sobre HTTP y WebSocket.
Celery	Cola de tareas distribuida para Python. IntelOwl la utiliza para ejecutar sus analizadores de forma concurrente mediante *workers*.
Ciberinteligencia (CTI)	Conocimiento basado en evidencias sobre amenazas o actores adversarios, obtenido mediante análisis de datos técnicos y contextuales, que permite tomar decisiones de seguridad informadas.
Contenedor	Unidad de software empaquetada con todas sus dependencias que se ejecuta de forma aislada del sistema anfitrión. Se gestiona mediante Docker y se orquesta con Docker Compose.
CORS (Cross-Origin Resource Sharing)	Mecanismo HTTP que autoriza a una página web a solicitar recursos servidos desde un origen distinto al suyo.
Rastreador	Servicio que recorre automáticamente una fuente externa (canales de Telegram, repositorios de GitHub o fuentes OSINT) para extraer indicadores de compromiso.
Cypher	Lenguaje declarativo de consulta de grafos propio de Neo4j. Permite expresar patrones de nodos y relaciones de forma intuitiva.
DNS (Domain Name System)	Sistema jerárquico que traduce nombres de dominio en direcciones IP. Su resolución permite vincular un dominio con la infraestructura que lo aloja.
Docker Compose	Herramienta que define y orquesta un conjunto de contenedores y volúmenes mediante un único fichero YAML declarativo. Automatiza el despliegue completo de Skyfall-CTI.
Embedding	Representación vectorial densa de un texto en un espacio de alta dimensión, generada por un modelo de lenguaje. Dos textos semánticamente próximos producen vectores con alta similitud coseno.

Término	Definición
EPSS (Exploit Prediction Scoring System)	Métrica que estima la probabilidad de que una vulnerabilidad concreta sea explotada en un plazo determinado.
Feed de amenazas	Canal estructurado de datos que publica indicadores de compromiso de forma periódica. Ejemplos: listas de IPs maliciosas de Emerging Threats o *pulses* de OTX AlienVault.
Grafo de conocimiento	Base de datos que representa entidades como nodos y sus relaciones como aristas dirigidas con propiedades. Permite expresar y consultar conexiones complejas de forma nativa mediante Cypher.
Hash criptográfico	Huella digital de longitud fija calculada a partir del contenido de un fichero (MD5, SHA-1 o SHA-256). Identifica de forma unívoca una muestra de malware con independencia de su nombre.
HNSW (Hierarchical Navigable Small World)	Algoritmo de búsqueda aproximada de vecinos más próximos organizado en capas jerárquicas de grafos de proximidad. Subyace a los índices vectoriales de Neo4j.
HTTP / HTTPS	Protocolo de transferencia de hipertexto sobre el que se comunican los servicios web, y su variante cifrada mediante TLS.
Indicador de compromiso (IOC)	Artefacto observable (dirección IP, nombre de dominio, URL, hash criptográfico o correo electrónico) que evidencia una posible intrusión o actividad maliciosa en un sistema o red.
Ingesta	Proceso de incorporación de datos de inteligencia al grafo de conocimiento, normalizados en formato STIX 2.1 mediante sentencias `MERGE` idempotentes en Neo4j.
IntelOwl	Plataforma de código abierto que agrega más de 300 analizadores de *threat intelligence* bajo una API común. Es el motor de enriquecimiento de observables de Skyfall-CTI.
IP (Internet Protocol)	Dirección numérica que identifica de forma única un equipo dentro de una red. Constituye uno de los observables fundamentales de la ciberinteligencia.
JSON (JavaScript Object Notation)	Formato ligero de intercambio de datos en texto plano. Es la serialización empleada por los paquetes STIX y por las APIs REST de la plataforma.
KRaft	Modo de consenso interno de Apache Kafka que gestiona los metadatos del clúster sin depender de ZooKeeper.
Microservicio	Componente independiente y desplegable por separado que cumple una única responsabilidad y se comunica con el resto a través de la red.
MISP (Malware Information Sharing Platform)	Plataforma abierta orientada al intercambio colaborativo de indicadores de compromiso entre organizaciones.
Modelo de lenguaje (LLM)	Red neuronal entrenada sobre grandes volúmenes de texto para comprender y generar lenguaje natural. En Skyfall-CTI consulta el grafo a través de MCP.
Model Context Protocol (MCP)	Protocolo abierto que conecta modelos de lenguaje con herramientas y fuentes de datos externas mediante un conjunto estandarizado de operaciones.
n8n	Motor de orquestación de flujos de trabajo que planifica y encadena las tareas de ingesta periódica de la plataforma.

Término	Definición
Neo4j	Sistema de gestión de bases de datos orientado a grafos. Almacena el grafo de conocimiento central de Skyfall-CTI modelado según STIX 2.1.
Nginx	Servidor web de alto rendimiento que actúa como proxy inverso, sirviendo la interfaz y enrutando las peticiones hacia los servicios internos.
Observable	Elemento técnico observable en un sistema o red que puede ser analizado para determinar su naturaleza maliciosa o benigna: dirección IP, dominio, hash, URL o correo electrónico.
OpenCTI	Plataforma de código abierto para la gestión de conocimiento de ciberinteligencia basada en STIX 2.1, desarrollada por Filigran.
OSINT (Open Source Intelligence)	Inteligencia obtenida a partir de fuentes de información de acceso público, sin recurrir a medios encubiertos.
Paquete STIX	Contenedor del estándar STIX 2.1 que agrupa objetos de dominio y de relación en un único mensaje JSON. Se emplea como unidad atómica de intercambio entre los servicios de Skyfall-CTI.
Playbook	En IntelOwl, conjunto predefinido de analizadores ejecutados en paralelo sobre un observable concreto. Skyfall-CTI define tres playbooks propios: `SkyfallCTIipReputation`, `SkyfallCTIurlreputation` y `SkyfallCTIHashReputation`.
Proxy inverso	Servidor intermediario que recibe las peticiones externas y las reenvía al servicio interno adecuado, ocultando la topología de la red. Lo desempeña Nginx.
Puntuación de riesgo	Métrica compuesta en el rango [0–100] calculada por Skyfall-CTI que pondera señales de reputación de múltiples fuentes para caracterizar la peligrosidad de un observable.
RAG (Retrieval-Augmented Generation)	Técnica que enriquece las respuestas de un modelo de lenguaje recuperando previamente fragmentos relevantes de una base de conocimiento.
Redis	Almacén de datos en memoria empleado como caché de respuestas y como *broker* de mensajería interna de IntelOwl y KevinAPI.
REST (Representational State Transfer)	Estilo de arquitectura para el diseño de servicios web sobre HTTP basado en recursos y operaciones sin estado.
SIEM (Security Information and Event Management)	Sistema que centraliza, normaliza y correlaciona los eventos de seguridad de una organización.
SOAR (Security Orchestration, Automation and Response)	Conjunto de tecnologías que automatizan y orquestan la respuesta ante incidentes de seguridad.
SSE (Server-Sent Events)	Mecanismo HTTP que permite a un servidor enviar un flujo continuo de eventos al cliente sobre una única conexión unidireccional.
SSH (Secure Shell)	Protocolo de acceso remoto cifrado a sistemas. Se emplea en la administración y el despliegue de la infraestructura.
STIX (Structured Threat Information Expression)	Estándar OASIS para la representación estructurada de inteligencia de amenazas. Constituye el modelo de datos común de Skyfall-CTI.

Término	Definición
Tácticas, Técnicas y Procedimientos (TTP)	Descripción estructurada del comportamiento de un actor adversario: qué objetivos persigue (táctica), cómo los consigue (técnica) y con qué herramientas o métodos concretos (procedimiento). Formalizados en MITRE ATT&CK.
TAXII (Trusted Automated eXchange of Indicator Information)	Protocolo OASIS de transporte para el intercambio automatizado de colecciones STIX entre plataformas CTI.
TLP (Traffic Light Protocol)	Esquema de etiquetado (CLEAR, GREEN, AMBER, RED) que regula el grado de diseminación permitido de una pieza de información.
TLS (Transport Layer Security)	Protocolo criptográfico que cifra y autentica las comunicaciones de red. Es la base de HTTPS.
URL (Uniform Resource Locator)	Cadena que identifica y localiza un recurso en la web. Constituye un observable analizable de la plataforma.
Vectorización	Proceso de generación de representaciones vectoriales para los nodos del grafo de conocimiento, que permite realizar búsquedas semánticas mediante los índices vectoriales HNSW de Neo4j.
Webhook	Llamada HTTP que un servicio emite automáticamente hacia otro cuando se produce un evento, habilitando integraciones reactivas.
XSIAM (Extended Security Intelligence and Automation Management)	Plataforma comercial de detección y respuesta extendida de Palo Alto Networks empleada por la organización de acogida.

Anexos

Apéndice A

Planificación y dedicación

Este anexo recoge la planificación temporal del proyecto y la dedicación estimada por fase. Se incluye un resumen de horas desglosado por actividad y el diagrama de Gantt que refleja la distribución y paralelismo del desarrollo a lo largo de los meses de trabajo.

A.1 Resumen de horas

La tabla A.1 recoge la dedicación estimada por fase del proyecto. El trabajo se desarrolló entre febrero y junio de 2026 a un ritmo aproximado de 20 horas semanales. Las fases de desarrollo transcurrieron en paralelo, tal y como refleja el diagrama de Gantt de la Figura A.1, por lo que la suma de horas no equivale a una secuencia lineal sino a una distribución de esfuerzo concurrente entre actividades.

Tabla A.1: Dedicación estimada por fase del proyecto Skyfall-CTI

Fase	Período	Horas
Investigación y estado del arte (MISP, OpenCTI, STIX 2.1)	Feb. 2026	30
Diseño de arquitectura e infraestructura Docker	Feb. – Mar. 2026	25
Desarrollo: ingesta y normalización STIX	Feb. – Mar. 2026	45
Desarrollo: enriquecimiento OSINT con IntelOwl	Feb. – Abr. 2026	45
Desarrollo: grafo de conocimiento Neo4j	Mar. – Abr. 2026	40
Desarrollo: API REST, servidor MCP y lógica de acceso	Abr. – May. 2026	40
Desarrollo: soporte RAG e integración LLM local	May. 2026	20
Desarrollo: interfaz web (Next.js, globo 3D, grafo, cuadros de mando)	Abr. – Jun. 2026	55
Integración y pruebas funcionales end-to-end	Jun. 2026	25
Redacción de memoria y documentación técnica	May. – Jun. 2026	40
Total	**Feb. – Jun. 2026**	**365**

A.2 Diagrama de planificación

La Figura A.1 muestra el diagrama de Gantt con la distribución temporal de las fases a lo largo del proyecto. Las barras de colores reflejan la naturaleza paralela del desarrollo: mientras se implementaba la capa de ingesta y enriquecimiento (febrero–abril) se avanzó en paralelo en el diseño del modelo de datos y la arquitectura de despliegue, y a partir de abril comenzó la fase de interfaz web de forma concurrente con la finalización de esa capa.

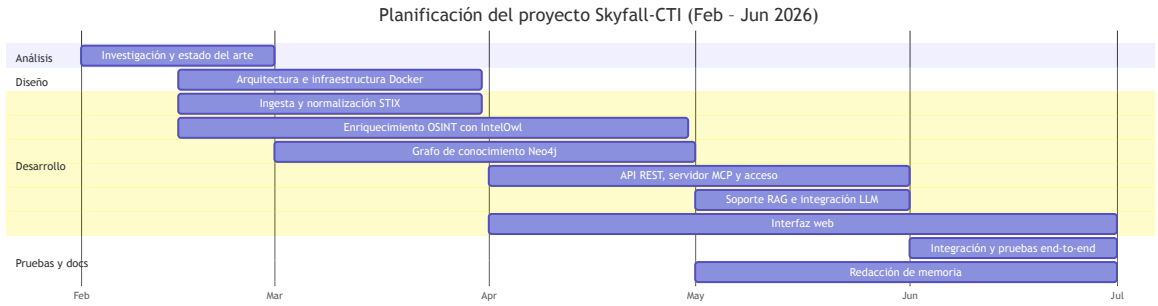


Figura A.1: Diagrama de Gantt del proyecto Skyfall-CTI (febrero – junio de 2026).

Apéndice B

Formulación matemática del sistema de puntuación de riesgo

Este anexo detalla la formulación matemática completa del sistema de puntuación de riesgo implementado en Skyfall-CTI. Se describen las señales de reputación utilizadas y su peso máximo, los niveles categóricos asignados sobre la puntuación final, y la función de cálculo completa que combina todas las señales en una métrica compuesta en el rango $[0, 100]$.

B.1 Señales de reputación consideradas

La tabla B.1 resume las señales de reputación que contribuyen a la puntuación, su fuente y la aportación máxima de cada una al resultado final.

Tabla B.1: Señales de reputación del sistema de puntuación de riesgo y su peso máximo

Señal	Fuente	Puntos máx.
Detecciones positivas	VirusTotal	35
Confianza de abuso	AbuseIPDB	30
Veredictos adicionales «malicioso»	Otros motores	25
Patrones de comportamiento	CrowdSec	15
Veredictos adicionales «sospechoso»	Otros motores	15
Confianza del indicador STIX	Objeto `indicator`	10
Número de informes de abuso	AbuseIPDB	10

B.2 Clasificación categórica del resultado

Sobre la puntuación final $S_{\text{final}} \in [0, 100]$, se asigna uno de tres niveles de riesgo:

Nivel	Rango
high	$S_{\text{final}} \geq 70$
medium	$35 \leq S_{\text{final}} < 70$
low	$S_{\text{final}} < 35$

B.3 Función de cálculo completa

La función `_skyfall_risk_score` de `skyfall-api` produce una puntuación bruta S_{raw}, calculada como:

$$S_{\text{raw}} = f_{\text{VT}}(v) \;+\; \left\lfloor \frac{30a}{100} + 0{,}5 \right\rfloor \;+\; \text{mín}(15,\, 5b) \;+\; \left\lfloor \frac{10c}{100} + 0{,}5 \right\rfloor \\ +\; \text{mín}\big(10,\, \lfloor 2\sqrt{r} + 0{,}5 \rfloor\big) \;+\; \text{mín}(25,\, 10m) \;+\; \text{mín}(15,\, 5s) \tag{B.1}$$

donde $\lfloor x + 0{,}5 \rfloor$ denota el redondeo al entero más próximo, y:

- v: número de motores VirusTotal que marcan el observable como malicioso. La función f_{VT} aplica una escala de cuatro escalones (máximo 35 puntos):

$$f_{\text{VT}}(v) = \begin{cases} 35 & v \geq 10 \\ 26 & 5 \leq v < 10 \\ 18 & 2 \leq v < 5 \\ 10 & v = 1 \\ 0 & v = 0 \end{cases}$$

- $a \in [0, 100]$: puntuación de confianza de abuso de AbuseIPDB (máximo 30 puntos, proporcional).
- b: número de patrones de comportamiento detectados por CrowdSec (máximo 15 puntos, amortiguado).
- $c \in [0, 100]$: nivel de confianza del objeto `indicator` STIX (máximo 10 puntos, proporcional).
- r: número total de informes de abuso independientes en AbuseIPDB (máximo 10 puntos, amortiguado con raíz cuadrada).
- m: número de motores de inteligencia adicionales que devuelven una conclusión “malicioso” (máximo 25 puntos).
- s: número de motores adicionales que devuelven una conclusión “sospechoso” (máximo 15 puntos).

La puntuación final se obtiene limitando S_{raw} al rango $[0, 100]$:

$$S_{\text{final}} = \text{mín}(100,\, S_{\text{raw}}) \tag{B.2}$$