Openlysis: Análisis de SMS, Correos, Archivos y URLs para Combatir el Phishing y Smishing en Android

Pedro Pablo Rodríguez Carranza

Proyecto de grado

Programa de Ingeniería de Sistemas

Universidad Nacional Abierta y a Distancia UNAD

Escuela de Ciencias Básicas, Tecnología e Ingeniería ECBTI

Openlysis: Análisis de SMS, Correos, Archivos y URLs para Combatir el Phishing y Smishing en Android

Pedro Pablo Rodríguez Carranza

Director

Ing. Iván Guillermo Duarte Pacheco

Programa de Ingeniería de Sistemas

Universidad Nacional Abierta y a Distancia UNAD

Escuela de Ciencias Básicas, Tecnología e Ingeniería ECBTI

Nota de aceptación:

_____

_____

_____

_____

_____

Firma del director del proyecto

Ing. Iván Guillermo Duarte Pacheco

_____

Firma del presidente del jurado

_____

Firma del jurado

_____

Firma del jurado

Bogotá D.C., 2025

Dedicatoria

A mi familia, por ser mi mayor fuente de inspiración y fortaleza. Por su amor incondicional, por creer en mí incluso en los momentos más difíciles, y por darme siempre su apoyo, paciencia y comprensión. Cada palabra de aliento y cada gesto de cariño me impulsaron a continuar y alcanzar esta meta. Este logro no solo me pertenece, sino que también es reflejo del esfuerzo, los valores y el ejemplo que me han transmitido.

Agradecimientos

Agradezco profundamente a mi familia por acompañarme durante todo este proceso académico y personal, por su apoyo constante y por motivarme a dar siempre lo mejor de mí.

Agradezco a mis docentes y tutores de la Universidad Nacional Abierta y a Distancia, quienes con su guía, conocimientos y compromiso hicieron posible la culminación de este proyecto.

Resumen

El phishing y el smishing representan amenazas críticas para la seguridad de los usuarios móviles, con ataques cada vez más sofisticados que causan pérdidas millonarias a nivel global. Aunque existen servicios de análisis avanzados como VirusTotal, su complejidad puede crear una brecha de accesibilidad y agilidad para los usuarios no técnicos. Este proyecto busca cerrar esta brecha mediante el desarrollo de Openlysis, un prototipo gratuito y de código abierto diseñado inicialmente para Android. Utilizando las metodologías CDIO y Scrum, el sistema integra múltiples servicios externos para analizar SMS, correos electrónicos, archivos y URLs, identificando contenido malicioso de forma simple e intuitiva. Como resultado, se espera obtener una arquitectura funcional compuesta por un sistema back-end resiliente, una aplicación móvil nativa para Android y los repositorios de código fuente correspondientes. El principal valor de Openlysis reside en democratizar el acceso a herramientas de ciberseguridad, proporcionando una capa de protección amigable y eficiente que no requiere conocimientos técnicos. De esta manera, el proyecto no solo aborda una necesidad de seguridad actual, sino que también contribuye a la creación de un entorno digital más seguro para todos, alineándose a los objetivos de desarrollo sostenible (ODS), aportando a la protección del bienestar digital (ODS 3: Salud y Bienestar), fomentando la innovación en ciberseguridad (ODS 9: Industria, Innovación e Infraestructura) y reforzando la seguridad digital para prevenir delitos cibernéticos (ODS 16: Paz, Justicia e Instituciones Sólidas).

Palabras Claves: Phishing**,** Smishing, VirusTotal, CDIO, SMS, API

Abstract

Phishing and smishing represent critical threats to mobile user security, with increasingly sophisticated attacks causing millions of dollars in losses worldwide. Although advanced analysis services such as VirusTotal exist, their complexity can create a gap in accessibility and agility for non-technical users. This project aims to bridge that gap through the development of Openlysis, a free and open-source prototype initially designed for Android. Using the CDIO and Scrum methodologies, the system integrates multiple external services to analyze SMS messages, emails, files, and URLs, identifying malicious content in a simple and intuitive way. As a result, the project delivers a functional architecture composed of a resilient back-end system, a native Android mobile application, and the corresponding source code repositories. The core value of Openlysis lies in democratizing access to cybersecurity tools, providing a user-friendly and efficient protection layer that requires no technical expertise. In doing so, the project not only addresses a current security need but also contributes to creating a safer digital environment for everyone. It aligns with the Sustainable Development Goals (SDGs) by promoting digital well-being (SDG 3: Good Health and Well-Being), fostering innovation in cybersecurity (SDG 9: Industry, Innovation, and Infrastructure), and strengthening digital security to prevent cybercrime (SDG 16: Peace, Justice, and Strong Institutions).

Keywords: Phishing, Smishing, VirusTotal, CDIO, SMS, API

Tabla de Contenido

Lista de Tablas

Lista de Figuras

Lista de Apéndices

Planteamiento del Problema

En el contexto actual, la ciberseguridad se ha convertido en un factor crítico en la cotidianidad de los usuarios, puesto que el uso constante de los dispositivos tecnológicos ha representado una oportunidad para que los delincuentes, encuentren formas más eficaces de lograr sus objetivos. En 2024, se registró un gran aumento del 71% en ciberataques que emplean credenciales robadas o comprometidas en comparación con el año anterior, y el 74% de todas las brechas de seguridad involucraron un elemento humano (ingeniería social) (Bonnie, 2025). Además, el 94% de las organizaciones reportaron haber sido víctimas de ataques de phishing, lo que subraya la gravedad de estas amenazas (Figueroa, 2024). En términos económicos, las pérdidas directas por ciberdelincuencia en el sector financiero global alcanzaron los 2.500 millones de dólares desde 2020, según el Fondo Monetario Internacional.

La interacción constante con dispositivos digitales expone a los usuarios a diversas amenazas que pueden comprometer su seguridad y bienestar. Los ciberdelincuentes aprovechan esta vulnerabilidad mediante técnicas diseñadas para persuadir a los usuarios y obtener información sensible, entre las cuales se destacan el phishing y el smishing.

El phishing consiste en el envío de correos electrónicos fraudulentos que imitan comunicaciones legítimas para inducir al usuario a acceder a enlaces que redirigen a sitios falsos, diseñados específicamente para solicitar información personal, credenciales bancarias u otros datos críticos. Un ejemplo común, es el de mensajes relacionados con compras en línea, en los que se incita al usuario a realizar un pago adicional para evitar la pérdida de un paquete, aprovechándose de situaciones reales para aumentar la credibilidad del fraude.

En otras ocasiones, alrededor del 10% de los casos, los correos contienen archivos adjuntos que pueden contener enlaces, ser parte de un proceso de ingeniería social más avanzado

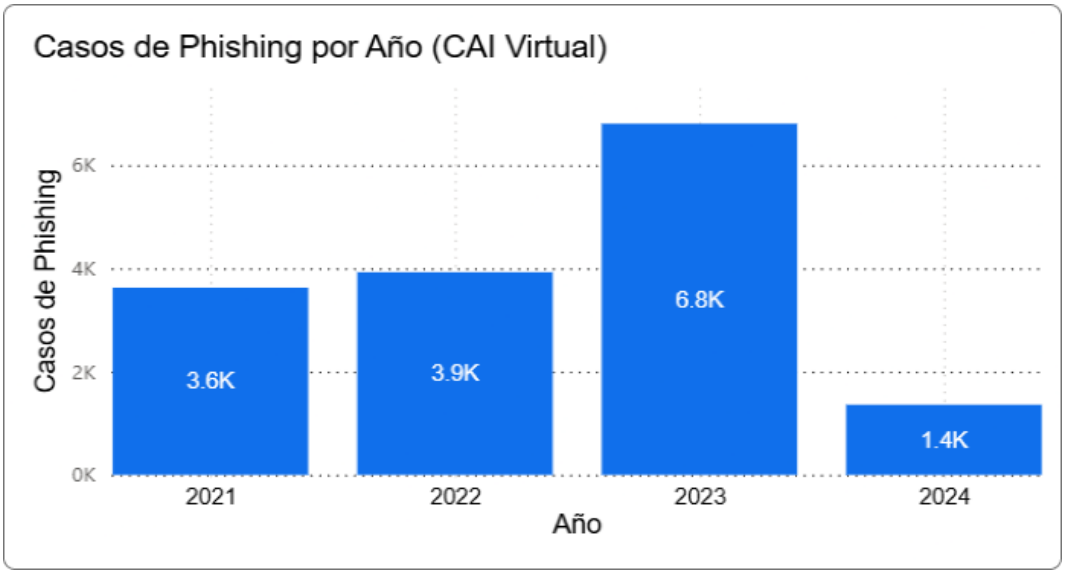
o directamente ejecutan software malicioso en el dispositivo del usuario, representando este último, un incidente más grave ya que el atacante podría obtener acceso a las cuentas del usuario, sin necesidad de robar sus credenciales. Este tipo de phishing con archivos adjuntos tiene un porcentaje mayor de éxito en comparación al uso de enlaces de manera directa (Baker & Cartier, 2025).

De forma similar, el smishing utiliza mensajes de texto SMS con el mismo objetivo: engañar al usuario para que acceda a enlaces maliciosos o comparta información confidencial, manipulando tanto los identificadores del remitente (ya sea números o direcciones de correo) como el contenido del mensaje.

A nivel nacional, como se muestra en la Figura 1, en los últimos 4 años se han reportado 15.714 denuncias, según los boletines del CAI Virtual de la Policía Nacional de Colombia, destacando el año 2023 en el cual se presentaron el 43,3% de los casos y de forma alarmante se presentó un incremento de casi el 90% entre el 2021 y el 2023. También es importante destacar que para el año 2024 las cifras se redujeron drásticamente un 80%, lo que podría reflejar la pasividad de los usuarios al denunciar esta problemática (Policía Nacional de Colombia, s.f.).

Figura 1.

Casos de phishing por año a nivel nacional

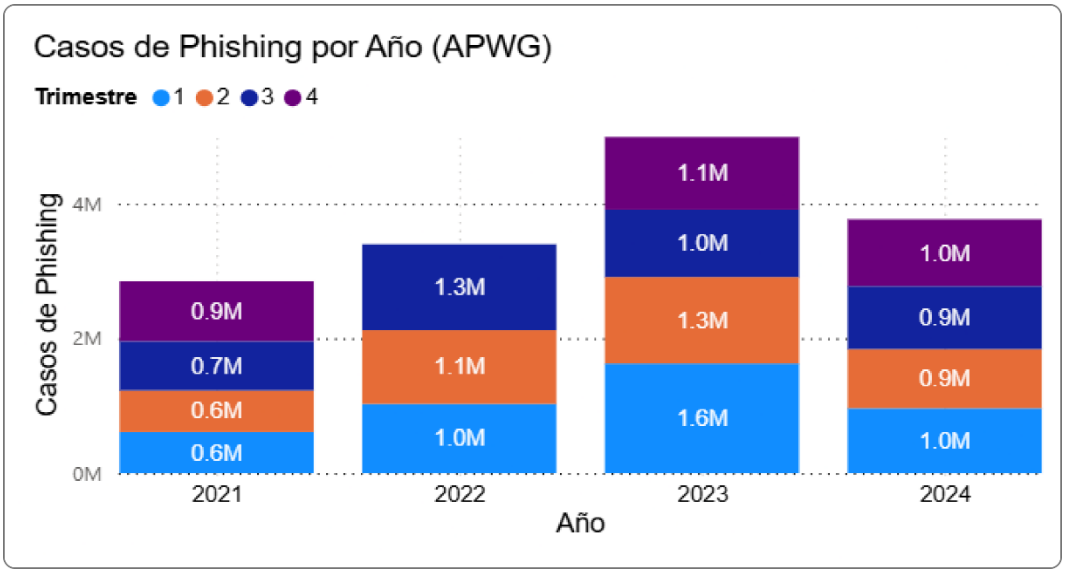


Fuente. Datos del CAI Virtual de la Policía Nacional de Colombia. Elaboración Propia.

A nivel global, como se muestra en la Figura 2, en base a los reportes de Anti-Phishing Working Group (APWG), se han presentado aproximadamente 16 millones de casos de Phishing en los últimos 4 años, que al igual que a nivel nacional, el año 2023 ha sido en el cual se ha registrado la mayor cantidad con un aproximado de 5 millones de casos. A nivel trimestral, destaca el primer trimestre del año 2023, que excepcionalmente registro un total de 1.6 millones casos. Por último, en concordancia con el panorama a nivel nacional, el 2024 también registro una disminución, pero que, en contraste, fue de tan solo un 24% (Anti-Phishing Working Group, s.f.).

Figura 2.

Casos de phishing por año a nivel internacional

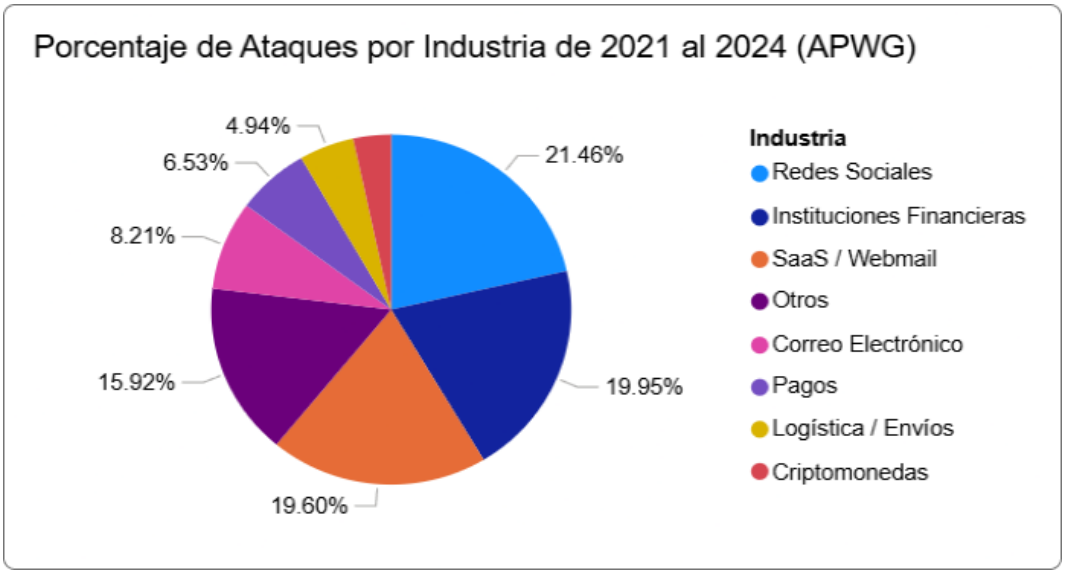


Fuente. Datos de los reportes trimestrales de APWG. Elaboración Propia.

Este volumen de ataques tiene una mayor incidencia en ciertas industrias. Como se observa en la Figura 3, las instituciones financieras han representado el 19,95% de los ataques de phishing en los últimos cuatro años. Junto con las redes sociales y los servicios de Software como Servicio (SaaS) y Webmail, concentran aproximadamente el 61% del total. Esto refleja las principales modalidades de operación de los ciberdelincuentes, quienes buscan suplantar la identidad de los usuarios a través de redes sociales, obtener acceso a cuentas de servicios en línea —incluyendo plataformas SaaS y correo electrónico— y, finalmente, sustraer credenciales bancarias con fines económicos.

Figura 3.

Porcentaje de ataques por industria desde 2021 al 2024



Fuente. Datos de los reportes trimestrales de APWG. Elaboración Propia.

Para comprender porque el elemento humano juega un papel fundamental para el éxito de estos ataques, la investigación de Jason E. Thomas (2018) analizó las estrategias para fortalecer las habilidades de los empleados frente a ataques de phishing. En dicha investigación se segmentaron a los empleados en función de características específicas, identificando que aquellos en altos cargos, los que gestionan gran cantidad de correos electrónicos o los que confían ciegamente en sus antivirus, presentan una mayor susceptibilidad a estos ataques. Dos características resaltan en estos segmentos: la falta de tiempo, derivada de sus responsabilidades o de la naturaleza de su cargo, y la excesiva confianza en mecanismos de protección que, al emplear motores propios, pueden tener falencias y no ser lo suficientemente avanzados para detectar malware o sitios fraudulentos.

Está problemática se hace más presente en los dispositivos móviles, el medio principal de interacción para muchos usuarios. Una investigación que incluyó a 29 usuarios de teléfonos

móviles expuso las diferencias entre el smishing y el phishing, destacando que en los dispositivos móviles resulta imposible hacer *hover* (puntero del mouse encima de algún elemento gráfico) sobre enlaces, lo que incrementa el nivel de ofuscación de las URLs. Asimismo, se determinó que los usuarios con menor entrenamiento o concientización sobre seguridad digital son mucho más susceptibles a ser engañados. La investigación subrayó la importancia de incorporar iconos, verificaciones y elementos visuales que proporcionen indicios de la veracidad de los mensajes y alerten a los usuarios (Tabassum, Faklaris, & Lipford, 2024).

Adicionalmente, en algunas aplicaciones, especialmente en servicios de mensajería (por ejemplo, Gmail), se implementan sistemas de escaneo para evaluar la fiabilidad de los archivos adjuntos; aun así, no es suficiente para detener las amenazas, puesto que estos sistemas siguen siendo una medida de protección básica. En este sentido, es fundamental la integración de servicios más sofisticados que ofrezcan análisis avanzados y con mayor transparencia respecto a los métodos detección de contenido malicioso. Actualmente, existen numerosos servicios en línea (como VirusTotal, Filescan o Hybrid Analysis), herramientas Sandbox (por ejemplo, Hybrid Analysis) y bases de datos de dominios, números telefónicos y correos reportados (como PhishTank, URLVoid o ScamSearch). La combinación de estas herramientas puede aumentar significativamente la probabilidad de detectar contenido malicioso. No obstante, dado que estas funcionalidades se ofrecen individual y principalmente a través de una API o de un sistema web, su uso resulta complejo para usuarios no técnicos o con limitaciones de tiempo, lo que representa una oportunidad para desarrollar una solución integral que centralice y simplifique el acceso a estas funciones.

Justificación

En la era digital actual, el uso diario y constante de los dispositivos tecnológicos ha expuesto a los usuarios a amenazas significativas que pueden comprometer su seguridad y bienestar, en especial, el phishing o el smishing, cuyos objetivos son estafar al usuario o infectar su dispositivo, por medio de mensajes de texto SMS o correos electrónicos, y posteriormente, obtener beneficios a costa de esto.

En los últimos años, con el crecimiento de la inteligencia artificial, estos se mensajes se han vuelto más eficaces, llegando a tal punto de ser personalizados, precisos y formales, eliminando los errores gramaticales que anteriormente los caracterizaban y permitiendo generar campañas de phishing y smishing en tan solo cinco minutos. Además, estos peligros han causado el robo de datos masivos de organizaciones, ya que tan solo un error humano, puede permitirles a los atacantes, evadir todo un esquema complejo de seguridad, lo que da lugar a grandes pérdidas económicas (Khalil, 2025).

La situación anterior, no solo expone los riesgos de las organizaciones, sino que también permite comprender el entorno al que se exponen los usuarios digitales, ya que, si la problemática es notable en los sectores organizados, entonces, las personas comunes pueden estar mucho más desprotegidas.

Afortunadamente, el auge de servicios de seguridad como VirusTotal, Filescan, Hybrid Analysis, entre otros, ha puesto a disposición herramientas muy poderosas para combatir y reducir estos riesgos, ofreciendo mecanismos de análisis eficaces para todo tipo de contenido digital. Sin embargo, estos servicios son utilizados principalmente en los sectores empresariales, o por usuarios con conocimientos avanzados en el entorno digital, lo que crea una brecha que impide a los usuarios no técnicos beneficiarse de ellos, principalmente por desconocimiento.

Además, aunque algunas de estas herramientas disponen de sitios web, su utilización consume tiempo, lo que se convierte en otra barrera para quienes disponen de recursos limitados para verificar la veracidad de mensajes de texto SMS, correos electrónicos e incluso archivos.

A partir de estas necesidades, surge la idea de desarrollar una solución gratuita y de código abierto, que facilite el uso de estos poderosos servicios. Este sistema permitiría a cualquier usuario del mundo digital, acceder fácilmente a la protección que ofrecen, sin tener que lidiar con la complejidad y el consumo de tiempo que actualmente requieren. Asimismo, el sistema se enfocaría principalmente en ofrecer análisis de mensajes de texto SMS, correos electrónicos y archivos adjuntos, que son los datos más relevantes particularmente en el phishing y el smishing. El desarrollo inicial tendrá un alcance para usuarios de dispositivos móviles de Android, dado que este dispositivo concentra el 61,92% del uso entre todas las categorías de dispositivos (Statscounter, s.f.), y el sistema operativo, el 73,9% dentro de los dispositivos móviles (Kumar, 2025). Es importante aclarar, que no se concibe como una solución definitiva a la problemática, sino que, en su lugar, como un mecanismo de protección amigable, intuitivo y eficiente.

Este proyecto representa la aplicación de los conceptos adquiridos durante el proceso de aprendizaje. El desarrollo implica la adquisición y evolución de habilidades técnicas en el diseño de arquitecturas de software distribuidas, la creación de APIs RESTful, la implementación de procesos robustos de comunicación entre servicios, desarrollo móvil nativo en Android y el manejo de patrones de diseño para crear código mantenible y escalable.

Este trabajo de grado también permite fortalecer y aplicar conceptos relacionados a la ingeniería y arquitectura de software, la seguridad informática y los sistemas distribuidos. De esta forma, el proyecto busca proponer una alternativa que tenga el potencial de beneficiar a los

usuarios del mundo digital y, asimismo, consolidar el perfil profesional del autor para materializar ideas en sistemas robustos y de alta calidad.

Objetivos

Objetivo General

Desarrollar una solución integral de análisis de seguridad para contenido digital (mensajes de texto SMS, correos electrónicos y archivos adjuntos), mediante el uso automatizado de servicios de escaneo especializados. Esta solución generará indicadores claros de confiabilidad y riesgos potenciales, permitiendo a los usuarios sin conocimientos técnicos tomar decisiones informadas de manera rápida y optimizar su protección contra amenazas cibernéticas en dispositivos móviles Android a partir de la versión 7.0.

Objetivos Específicos

Definir el soporte de la solución, detallando las funcionalidades fundamentales, la base de datos del sistema y los servicios externos de escaneo a implementar, estableciendo así las bases para el desarrollo de la solución.

Diseñar la estructura de la API utilizando una arquitectura limpia basada en capas con el framework .NET, asegurando alta eficiencia en la gestión de peticiones y respuestas, adecuado desacoplamiento de las aplicaciones front-end, cumplimiento con estándares definidos bajo la arquitectura REST, y una separación efectiva de la base datos y los servicios externos de escaneo.

Desarrollar la API de acuerdo con los lineamientos diseñados, logrando un sistema de alto rendimiento, totalmente asíncrono, robusto, resiliente y seguro, minimizando fallos temporales en el servicio, y de la misma manera, incorporando una base de datos desacoplada y escalable en conjunto con los servicios externos definidos.

Diseñar un prototipo navegable para una aplicación móvil de Android, presentando los resultados de los análisis mediante interfaces fundamentadas en principios de usabilidad, asegurando una buena experiencia de usuario para todas las personas.

Desarrollar una aplicación móvil para Android 7.0, alineándose a los diseños del prototipo, que ofrezca herramientas de análisis tanto manuales como automatizadas, y sus respectivos resultados, usando la API REST para garantizar la funcionalidad y eficacia en la protección del usuario.

Marco Referencial

En los ambientes digitales actuales, la seguridad de la información se ha convertido en un cuidado fundamental para los usuarios de dispositivos móviles. Dos de las amenazas más prevalentes son el phishing y el smishing, que usan técnicas engañosas para robar datos sensibles. El phishing, que se manifiesta a través de correos electrónicos fraudulentos, ha crecido con el aumento del uso de smartphones, mientras que el smishing, su contraparte en mensajes de texto ha demostrado ser igualmente efectivo, afectando a un porcentaje significativo de usuarios e incluso a organizaciones. Estas prácticas delictivas son particularmente peligrosas para las personas mayores, quienes usualmente carecen de conocimientos actualizados sobre seguridad digital.

A pesar de estas amenazas, el avance tecnológico ha generado el desarrollo de herramientas de análisis de contenido digital, como VirusTotal e Hybrid Analysis, que emplean múltiples motores y mecanismos de detección para identificar malware y fraudes. La integración de inteligencia artificial en estas herramientas ha mejorado la precisión en la detección de ataques, permitiendo una respuesta más efectiva ante estos riesgos. Sin embargo, a pesar de los avances, persiste la necesidad de una solución integral que combine funcionalidad y usabilidad, abordando las vulnerabilidades existentes en el entorno digital.

Marco Teórico

El proyecto se sustenta en dos problemáticas actuales del mundo digital que representan amenazas significativas para cualquier usuario, sin importar su comunidad. Por un lado, el phishing se ha popularizado en los últimos años debido al creciente uso de smartphones y dispositivos digitales en general. Esta técnica consiste en enviar correos electrónicos con información falsa para incitar a los usuarios a realizar acciones específicas, como acceder a sitios

web fraudulentos mediante URLs, y proporcionar datos sensibles que los ciberdelincuentes pueden utilizar para obtener beneficios económicos u otros fines. Aunque estas situaciones se presentan con mayor recurrencia en contextos organizacionales, como lo han demostrado investigaciones en ámbitos empresariales y académicos, un estudio publicado en 2024 coincide en la mayoría de sus hallazgos, pero en este caso, desde una perspectiva más personal centrada en el ámbito privado de los usuarios (Köhler, Pünter, & Meinel, 2024).

Por otro lado, surge el smishing, que sigue la misma modalidad del phishing, pero utilizando mensajes de texto SMS. Aunque la mayoría de los usuarios logran evitar estos fraudes, un estudio de 2022 reveló que, de un total de 256 usuarios, el 16,92 % respondió a un mensaje smishing y, en un segundo ataque, el 12,82 % volvió a caer (Rahman & Timko, 2022). Cabe destacar que los ciberdelincuentes se centran en usuarios de dispositivos móviles, aprovechando que este método permite enviar mensajes en masa, lo que aumenta tanto la rentabilidad como la probabilidad de éxito del fraude (Wahsheh & Zahrani, 2022).

Estas amenazas, sumadas a la falta de conocimientos actualizados sobre seguridad digital, representan un gran problema para la población de mayor edad, sin embargo, no deja de ser un riesgo latente para cualquier usuario.

En respuesta tanto a estos como a otros desafíos, en los últimos años se han desarrollado tecnologías que ofrecen servicios avanzados de análisis de contenido digital. Herramientas como VirusTotal, Filescan e Hybrid Analysis se han destacado por emplear múltiples motores de antivirus y mecanismos de detección basados en reglas YARA –patrones que describen características específicas de software malicioso--, y, además, ejecutan análisis en entornos aislados para evaluar el comportamiento de sitios web y archivos.

El auge de la inteligencia artificial también ha dado lugar a herramientas gratuitas y de código abierto que facilitan el análisis de contenido digital, mensajes de texto y correos electrónicos. Diversas investigaciones han utilizado modelos de lenguaje para evaluar la precisión en la detección y clasificación de ataques de phishing y smishing. Por ejemplo, un estudio publicado en 2021 presentó un sistema capaz de detectar mensajes smishing en dos fases: primero verificando la autenticidad de la URL y, a continuación, clasificando el mensaje mediante inteligencia artificial, alcanzando una precisión del 97,93% (Mishra & Soni, 2021).

Adicionalmente, la integración de estos servicios en un sistema robusto y complementario permite aprovechar las fortalezas de cada herramienta.

- VirusTotal ofrece análisis que combina diversos motores de antivirus, siendo ideal para detectar malware en archivos adjuntos y verificar URLs mediante blacklists –listas con enlaces previamente identificados como sospechosos o maliciosos--.
- PhishTank emplea una base de datos que permite registrar y almacenar URLs, que, a través de la participación de usuarios de la comunidad digital en general, se determina si un sitio web es phishing.

Asimismo, las herramientas sandbox, que ofrecen VirusTotal, Hybrid Analysis y Filescan permiten emular el acceso a sitios web para observar su comportamiento en un entorno controlado. Esta capacidad no solo complementa los mecanismos de escaneo basados en blacklists, sino que también posibilita que la inteligencia artificial analice de forma más interactiva los detalles del veredicto, determinando con mayor precisión si un sitio web es malicioso.

Por último, un artículo publicado en 2021, que combinó el uso de varias APIs demostró un alto nivel de éxito en la detección de phishing, especialmente utilizando VirusTotal y Safe-Browsing, destacando que, según las necesidades (precisión o tiempo de respuesta), algunos servicios pueden ser más adecuados. El estudio destaco a VirusTotal, que alcanzó una precisión del 99,27 %, aunque con un tiempo promedio de respuesta de 12 a 15 segundos, en contraste con Safe-Browsing, el cual logró un 87 % en apenas 0,15 ms (milisegundos) (Thorpe & Phadke, 2021).

Todo lo anterior da cuenta de que las amenazas existentes a pesar de que han tenido un sinfín de investigaciones con distintos modelos y sistemas, la mayoría se han limitado al apartado funcional, lo cual, ha dejado la problemática con soluciones potenciales, pero sin hechos que destaquen a una herramienta definitiva integral tanto desde la parte funcional como visual (interfaz de usuario).

Marco Conceptual

En esta sección se presentan los conceptos fundamentales vinculados al proyecto respaldados a partir de revisiones bibliográficas y fuentes reconocidas. Primero se abordan nociones generales respecto al entorno general, posteriormente, se exponen los términos directamente relacionados con la problemática, y, por último, se definen los conceptos asociados a la solución propuesta.

Entorno Digital

En este apartado se describen las nociones generales en el entorno digital.

Enlaces o URLs. Los enlaces o mayormente referidos en el mundo digital como "*Uniform Resource Locator*" por sus siglas URL, según MDN Web Docs (*Mozilla Developer Network Web Documentation*), son direcciones que apuntan a recursos únicos (existen excepciones) en internet y permiten obtener dichos recursos, ya sean documentos HTML (*Hyper Text Markup Language*), CSS (*Cascading Style Sheets*), imágenes u otro tipo de contenido digital. Son el mecanismo principal por el cual los navegadores acceden a sitios web (MDN, 2025).

Problemática

En este apartado se definen los términos relacionados con la problemática abordada en el proyecto: malware, phishing, smishing y SMS.

Malware. Software malicioso generalmente referido como "Malware", son todos aquellos programas cuyo objetivo es realizar acciones dañinas contra un sistema o un usuario buscando comprometer la seguridad de este y corromper las operaciones normales (Guven, 2024). Este tipo de software usualmente es instalado sin el consentimiento del usuario y generalmente ocurre cuando un usuario descarga y ejecuta programas de dudosa procedencia.

Este término abarca diversos tipos como Ransomware, Spyware, Troyanos, Rootkits, etc., sin embargo, todos buscan causar daños ya sea a usuarios u otros sistemas informáticos para obtener algún beneficio, convirtiéndolos en software altamente peligroso.

Phishing. Zienie, Masari, y Calzarossa (2023) afirman que el phishing es una técnica utilizada para robar la información de los usuarios, generalmente a través de sitios web., es considerada como una técnica de ingeniería social que incita a los usuarios a acceder a URLs que direccionan a sitios fraudulentos con una apariencia bien diseñada para aparentar legitimidad y oficialidad, con el fin de aumentar la probabilidad de que los ciberdelincuentes logren su objetivo. Estos sitios web contiene campos de entrada para que la víctima introduzca información sensible (Nombres, apellidos, contraseñas, correos electrónicos, etc.) y los datos se envíen al destino deseado por el ciberdelincuente. A raíz de todo esto, esta modalidad es bastante peligrosa y compromete la integridad, confidencialidad y disponibilidad de los datos de los usuarios.

Smishing. Es una variante del phishing distinguida por ser llevada a cabo a través de mensajes de texto SMS (*Short Message Service*), para manipular a los usuarios y lograr la misma finalidad –la obtención ilícita de información sensible-- (Kosinski, 2024).

SMS. Refiere a "*Short Message Service*" o servicio de mensajes cortos por su traducción, es una forma de intercambiar mensajes, con una longitud máxima de 160 caracteres, a través de dispositivos móviles. Usualmente se utiliza para la comunicación entre negocios por su forma conveniente de enviar mensajes concisos a individuos o grupos de usuarios (AWS).

Solución

En este apartado se exponen los conceptos asociados a la solución propuesta: API, RESTful API, HTTP, back end, front end, análisis de malware y entorno sandbox.

API. Las interfaces de programación de aplicaciones, usualmente referidas como API (*Application Programming Interface*), hacen parte de un tipo de aplicación que dan acceso a funcionalidades mediante puntos de acceso estandarizados y denominados "*Endpoints*", que como lo mencionan Lamothe, Guéhéneuc, y Shang (2021), permiten ahorrar tiempo, ya que desacoplan distintos sistemas (por ejemplo, el back end y el front end) promoviendo la reutilización.

RESTful API. Una "API REST" o "API RESTful" es una API que sigue un estilo de arquitectura de diseño llamada "REST" que establece un conjunto de criterios lo suficientemente restrictivas para estandarizar una API, pero suficientemente flexibles para ofrecer aplicaciones altamente escalables y livianas. Una de las características principales es que el sistema no debe depender ni conocer el estado del cliente y viceversa (Red Hat, 2020).

HTTP. Por sus siglas "*Hyper Text Transfer Protocol*", es un protocolo de transmisión de datos para obtener recursos de servicios web (por ejemplo, páginas web), por medio de peticiones estandarizadas e individuales facilitando la comunicación asíncrona entre un cliente y un servidor (MDN, 2025).

Back end. Refiere a la aplicación que realiza la mayor parte de la funcionalidad de un sistema, por ejemplo, acceder a la base de datos, a diversos servicios u otras APIs. También puede ser referida como "*server-side*" y se comunica a través del front end usualmente con peticiones y respuestas HTTP (AWS).

Respecto a la utilización gramatical, especialmente en inglés, la conjunción 'back end' se utiliza para representar un sustantivo, mientras que 'back-end', funciona como adjetivo.

Front end. Refiere a la aplicación o interfaz gráfica con la que los usuarios interactúan de forma directa para acceder y usar funcionalidades mediante botones, campos de entrada y otros elementos gráficos (AWS). Durante la interacción, el front end enviará peticiones, generalmente en formato HTTP, al back end y recibirá respuestas que serán reflejadas mediante otros elementos gráficos favoreciendo la comprensibilidad del usuario.

Al igual que el concepto back end, la conjunción 'front end' se utiliza para representar un sustantivo, mientras que 'front-end' es un adjetivo.

Análisis de Contenido Digital. Proceso de comprender la fiabilidad de un archivo, URL o software a través de análisis estáticos, en los cuales se verifican firmas, certificados, propiedades y código; análisis dinámicos que ejecutan y emulan la interacción en entornos sandbox o análisis híbridos que emplean ambas técnicas (Baker K. , 2023).

Entorno Sandbox. De acuerdo con Lenaerts-Bergmans (2023), el término sandbox refiere a entornos aislados y seguros, que ayudan a entender el comportamiento de una pieza de código o de un programa cuando está en ejecución, dando lugar, en este caso, a determinar la fiabilidad del software. Estos entornos pueden simular la conexión a internet y otros servicios esenciales para asegurar un estudio completo.

C#. Es un lenguaje de programación orientado a objetos creado por Microsoft, que pertenece a la familia de lenguajes C, tiene una sintaxis bastante similar a Java y soporta la ejecución en paralelo, la asincronía y la concurrencia, lo que permite desarrollar aplicaciones altamente eficientes y robustas (Wagner, 2025).

.NET. Es un marco de trabajo (*framework*)*,* comúnmente utilizado con C#, el cual permite desarrollar aplicaciones para diferentes plataformas, como lo puede ser un servidor o el dispositivo portátil de un usuario. Ofrece un conjunto de componentes que se encargan de compilar el código, ejecutar las aplicaciones y optimizar el trabajo a los desarrolladores con librerías, herramientas y otros marcos de trabajo basados en .NET, pero con objetivos más particulares, por ejemplo, ASP.NET se enfoca en la creación de APIs (Microsoft, 2024).

Kotlin. Es un lenguaje moderno y de código abierto que soporta la programación tanto orientada a objetos como funcional, fue en lanzado en 2016 y, por ende, incorpora funcionalidades únicas e interesantes de otros lenguajes, siendo así, que una de sus características es la interoperabilidad con Java.

Usualmente, este lenguaje es asociado con el desarrollo de aplicaciones móviles para Android, ya que *Google* en conjunto con Jetbrains fundaron Kotlin Foundation, que es un grupo encargado de administrar el lenguaje de programación y actualmente recomiendan emplear *Kotlin,* para la creación de aplicaciones de Android (Google, 2023).

VirusTotal. Servicio de análisis que incorpora alrededor de 70 motores de antivirus, ofrecidos a través de un sitio web y una API mediante la cual permite subir archivos o enlaces para detectar software malicioso (Cibersecurity & Infrastructure Security Agency, s.f.).

Antecedentes

A lo largo de los años se han realizado numerosas investigaciones que han estudiado diversos mecanismos de detección de phishing o smishing, algunos enfocados en relacionar patrones del lenguaje del mensaje o en las URLs, otros implementan marcos de trabajos basados en reglas, y los más recientes, utilizan modelos de inteligencia artificial o algoritmos de aprendizaje de automático, que han demostrado ser bastante precisos pero con diversas limitaciones frente a técnicas de phishing más elaboradas y sofisticadas.

Abuadbba et al. (2022) crearon un modelo denominado Anti-SubtlePhish, cuyo objetivo fue detectar sitios web de phishing emergentes, o comúnmente conocidos como *zero-day*, que explotan vulnerabilidades no conocidas y son difíciles de detectar en etapas tempranas. Para el desarrollo, compararon los datos de las URLs y de los sitios web renderizados con información de terceros, como Google Safe Browsing o WHOIS, utilizando el coeficiente de Pearson. Posteriormente, clasificaron los resultados mediante un modelo de regresión logística, entrenado con un conjunto de datos de 100 000 URLs (50 000 legítimas y 50 000 de phishing). La evaluación se llevó a cabo midiendo la precisión, sensibilidad, exactitud global y tasas de error en tres conjuntos de datos, además de ataques *zero-day* creados por los investigadores. Los resultados mostraron una precisión del 100 % frente a los casos *zero-day*, un desempeño general del 96 % y una tasa de error inferior al 4 %. Estos logros fueron mayormente superiores en comparación con VirusTotal y otros modelos de aprendizaje automático, que presentan debilidades frente a ataques *zero-day* debido a la falta de interacción dinámica con los sitios web, característica que puede ser suplida mediante herramientas *sandbox* (Abuadbba, y otros, 2022).

Aljofey et al. (2022) propusieron un sistema de detección de phishing basado en la extracción de características de URLs y del contenido HTML. La implementación consta de dos fases: primero, la detección de los datos necesarios en los enlaces y en los documentos HTML; y luego, la clasificación mediante el modelo de aprendizaje automático XGBoost para determinar si la página es phishing. Para ello, utilizaron dos conjuntos de datos, uno propio y otro de referencia, ambos con sitios categorizados como seguros o maliciosos. En el primer conjunto, lograron una precisión del 96.76% y una tasa de falsos positivos del 1.39%, mientras que en el de referencia, alcanzaron una precisión del 98.48% y una tasa de falsos positivos del 2.09%. Los resultados demuestran la alta eficacia del modelo, que extrae características tanto de las URLs como de los HTML, resaltando la importancia de estas técnicas que en cierto modo también son implementadas por servicios externos como urlquery o Hybrid Analysis, que se especializan en análisis dinámico de sitios web mediante herramientas sandbox (Aljofey, et al., 2022).

Choo et al. (2022) en su investigación, buscaron proponer un mecanismo altamente preciso para detectar y, especialmente, determinar los tipos de ataques asociados a enlaces maliciosos, empleando como primer factor, la herramienta VirusTotal (VT). Los autores destacaron el ruido que generan los diferentes escáneres individuales empleados por VT, que afectan la manera en que se decide si un enlace es realmente malicioso, y de igual manera, incide negativamente al categorizar el tipo de ataque (por ejemplo, malware o phishing). Para cumplir con el objetivo y abordar los aspectos mencionados, se recolectaron un aproximado de 1.5 billones de enlaces que fueron enviados a VT desde julio de 2019 a enero 2022, verificaron cada uno de estos enlaces para distinguirlos entre malware o phishing haciendo uso de técnicas de muestreo aleatorio y análisis estadístico, junto con modelos de aprendizaje automático que consideraron las características de los informes, como el grado de conflicto entre etiquetas, las

correlaciones y especializaciones de los escáneres y los patrones temporales de detección. Los resultados mostraron que los escáneres tienden a ofrecer categorizaciones conflictivas, especialmente en campañas de phishing, y que los métodos tradicionales basados en umbrales fijos y votaciones mayoritarias son limitados frente a la complejidad de los datos, en cambio, los modelos de aprendizaje integrados que aprovechan las relaciones y características específicas de los escáneres, ofrecen una mejor detección, alcanzando una precisión superior al 97% (Choo, Nabeel, De Silva, Yu, & Khalil, 2022).

Abdul Karim et al. (2023) en su investigación buscaron desarrollar un sistema de detección de phishing basado en enlaces, que fuese preciso y eficiente usando un modelo híbrido, que combinará técnicas de aprendizaje automático. Para su creación se midió el rendimiento de varios algoritmos de clasificación: Decision Tree (DT), Linear Regression (LR), Support Vector Classifier (SVC), Support Vector Machine, Gradient Boosting Machine, Random Forest, Naive Bayes, K Nearest Neighbors (KNN) Classifier y el modelo propuesto que es una combinación híbrida entre los algoritmos DT, LR y SVC; posteriormente fueron evaluados utilizando un conjunto de datos ofrecidos en Kaggle, que proporcionaban atributos esenciales para detectar phishing, como el uso HTTPS en el sitio web o si el enlace tiene una gran longitud. Finalmente, se establecieron métricas para medir la precisión y el rendimiento, encontrando que el modelo híbrido logró una precisión del 99.2%, superior a todos los algoritmos individuales, que alcanzaron máximos de alrededor del 97,8%. Estos resultados demostraron claramente que el modelo propuesto no solo cumplió con el objetivo del estudio, sino que también ofreció una mejora significativa en la detección de sitios de phishing, consolidando su eficacia en entornos reales (Brahim Belhaouari, Karim, & Shahroz, 2023).

Huamanchumo Trujillo et al. (2024) tuvieron como objetivo desarrollar un sistema de protección contra quishing, una variante del phishing que utiliza códigos QR, empleando aprendizaje automático y la API de VirusTotal. Para lograrlo, primeramente, utilizaron un script de Python con OpenCV para extraer el enlace del código QR que posteriormente era analizado mediante el envío de una petición GET en HTTP a la API de VirusTotal y el resultado se clasificaba en malicioso, seguro o desconocido para finalmente emitir una alerta preventiva. En segundo lugar, se evaluó la precisión utilizando un conjunto de datos con 50 códigos QR catalogados como maliciosos y otros 50 como seguros, a partir de esto, se establecieron métricas de medición de precisión, sensibilidad, tasa de error, tiempo de respuesta y tiempo de respuesta promedio. Los resultados mostraron que VirusTotal obtuvo una precisión de detección del 100%, sensibilidad del 95%, pero una tasa de error del 14%, puesto que algunos enlaces catalogados como seguros, fueron clasificados como desconocidos por la falta de información, sin embargo, esto quiere decir que el sistema es eficaz y puede ofrecer una capa adicional de seguridad para prevenir el acceso a enlaces maliciosos (Trujillo, Gamarra, Saldaña, & De Los Santos, 2024).

Zhang et al. (2025) en su estudio propusieron un mecanismo basado en URLs para detectar sitios web de phishing que emplean características dinámicas para evadir los métodos tradicionales desarrollados por otras investigaciones. La metodología integró dos componentes principales: primero, una propiedad basada en tokens embebidos para la extracción de características intrínsecas de las URLs; y segundo, un modelo de detección de múltiples canales que combina redes neuronales convolucionales para extraer características locales, las cuales se fusionan con otras características más globales, además de incorporar una función de detección de deriva conceptual (concept drift detection - CDD) que permite al modelo evolucionar y aprender de manera autónoma, sin depender necesariamente de conjuntos de datos previamente

categorizados como maliciosos o seguros. Los resultados demostraron que el modelo propuesto alcanzó una precisión del 91.22% en conjuntos de datos relacionados con la deriva de concepto, superando a otros modelos de investigaciones previas en un rango entre el 6.20% y el 42.93% (Zhang, Wu, Lu, Shi, & Liu, 2025).

Rao et al. (2025), desarrollaron una propuesta, llamada *Phish-Jam*, para detectar phishing utilizando directamente los enlaces y enfocándose en los entornos móviles, que cuentan con recursos limitados en comparación con servidores u otro tipo de dispositivos. El principal objetivo fue mitigar los problemas que presentan los mecanismos basados en listas negras (*blacklists*), análisis de código fuente y otros métodos que pueden ser tanto ineficaces contra ataques de phishing emergentes o que requieren un gran poder de computación. Para implementar la propuesta, se extrajeron características de los enlaces para obtener patrones semánticos y estructurales, que posteriormente fueron clasificados mediante diversos modelos de aprendizaje automático, tanto *Deep Learning* (DL) como *Machine Learning* (ML), implementados de forma híbrida en una aplicación móvil cuyas respuestas se agregaron para mejorar la precisión. En cuanto a los resultados, el sistema demostró un alto rendimiento, alcanzando una precisión del 98.93% y una velocidad de respuesta promedio de 480 milisegundos, lo que lo convierte en una propuesta escalable y eficiente, especialmente para entornos móviles en tiempo real (Rao, Kondaiah, Pais, & Lee, 2025).

Marco Jurídico

Es esencial entender que la propuesta comprende limitaciones y obligaciones fundamentales para garantizar la privacidad y protección de datos de los usuarios, considerando que se está implicando el escaneo de mensajes de texto SMS, correos electrónicos y archivos que pueden ser de carácter privado, por lo tanto, es crucial alertar a los usuarios de prevenir el uso del servicio con contenido digital de este tipo.

Respecto a la problemática, los ciber crímenes como el phishing y smishing son sancionados a través de leyes y artículos a nivel internacional y especialmente a nivel nacional, donde se establecen penas económicas y carcelarias.

Internacional

La importancia de la privacidad de datos a nivel mundial se regula a través del artículo 12 de la declaración universal de derechos humanos, en la cual, se establece el derecho a la intimidad que cada ser humano tiene tanto a nivel físico como digital (Derechos Humanos, 2018).

Por otro lado, el pacto internacional de derechos civiles y políticos en el artículo 17 establece lo siguiente “Nadie será objeto de injerencias arbitrarias o ilegales en su vida privada, su familia, su domicilio o su correspondencia, ni de ataques ilegales a su honra y reputación.”, fundamentando la protección de los usuarios ante las consecuencias del phishing y smishing. De la misma manera, se fortalece la premisa con un segundo punto que expresa: “Toda persona tiene derecho a la protección de la ley contra esas injerencias o esos ataques.” (Naciones Unidas).

Por último, la OCDE (Organización para la Cooperación y el Desarrollo Económico) proporciona ocho principios básicos que estandarizan el tratamiento de datos a nivel internacional: limitación de recogida, calidad de los datos, especificación de los fines, limitación

de uso, salvaguarda de la seguridad, transparencia, participación individual y, por último, responsabilidad (Organisation for Economic Co-operation and Development, 1980). Estos principios pretenden asegurar la privacidad de los usuarios de una forma honesta y clara, definiendo limites que den garantías a los usuarios de la protección y tratamiento de datos a nivel internacional, lo cual es relevante en este caso, para la propuesta tecnológica del proyecto.

Nacional

A nivel nacional, existen distintas leyes que regulan la privacidad y tratamiento de datos, e igualmente, algunas que establecen códigos penales estrechamente relacionados con el phishing, smishing y software malicioso en general:

La ley 1266 de 2008 (Colombia, 2008) y la ley estatutaria de 1581 de 2012 (Colombia, 2012) desarrolla el derecho constitucional destacando que las personas pueden conocer, actualizar y rectificar la información recogida sobre ellos en los bancos de datos. Si bien, la solución propuesta puede manejar datos personales a través de la creación de usuarios, se prevé únicamente la gestión de datos públicos o información no confidencial del usuario que pueda ser divulgada en la menor medida posible, a través del uso de los distintos servicios de análisis, pero siempre actuando bajo el consentimiento del usuario. Además, en conjunción con el decreto de 1337 de 2013, se establecen pautas para asegurar una correcta implementación de la reglamentación establecida en la ley de 1581 de 2012 (Colombia, 2013).

La ley 1273 de 2009, en el artículo 269E, establece una pena económica o carcelaria a aquellos que incurran en acciones relacionadas con software malicioso sin consentimiento informado. También, el artículo 269F y 269G introducen penas de forma directa que castigan el tratamiento indebido y robo de datos personales, a través de la suplantación de sitios web que

sean enviados, ya sea, con ventanas emergentes o enlaces, sin el consentimiento adecuado (Colombia, 2009).

Marco Tecnológico

El prototipo se fundamentará en un conjunto de tecnologías distribuidas en las siguientes áreas: hardware, back end, base de datos y front end. Esto asegura un alto nivel de funcionalidad, permitiendo que el sistema, especialmente el back end, pueda ser implementado y probado en cualquier momento y posibilite su uso con otros sistemas front-end.

Infraestructura

Aunque el proyecto no pretende culminar en una implementación completa del sistema en un servidor o servicio, si se desea alcanzar un desarrollo funcional que permita su puesta en marcha. En este sentido, se han definido tres tecnologías que serán la base del funcionamiento a nivel de hardware.

Back end

Considerando la necesidad de escalabilidad, las tendencias actuales y la preferencia por servicios en la nube, el sistema back-end estaría respaldado por el servicio Cloud Run de Google, el cual ofrece una capa gratuita de consumo para alojar distintos servicios internos del sistema back-end (Google, s.f.). Esta elección garantiza flexibilidad y capacidad de adaptación a futuras ampliaciones.

Base de Datos

Se opta por una base de datos relacional para almacenar registros estructurados que reflejen de manera estandarizada los resultados de los análisis provenientes de diversos servicios externos. Esto facilita la integración del back end con otros sistemas front-end, permitiendo extender la solución sin sobrecargar la lógica funcional del cliente.

En concreto, se utilizará PostgreSQL, ya que cumple con los estándares SQL de las bases de datos relacionales, y esencialmente, es de código abierto, lo que lo convierte en el sistema

óptimo para implementaciones de bajo costo o incluso para aplicaciones grandes que necesitan escalabilidad (IBM, 2021).

Dispositivo Android de Base

En el ámbito del front end, la aplicación se desarrollará y probará en dispositivos Android, garantizando compatibilidad a partir de la versión 7.0, lo que asegura una base sólida para la experiencia de usuario móvil y con una buena cobertura para todos los dispositivos en vigencia. No obstante, Google actualmente exige que las nuevas aplicaciones publicadas en Google Play estén dirigidas a versiones más recientes, a partir de Android 14.0, debido a que contienen mejoras acumuladas respecto a la seguridad, rendimiento y características en los dispositivos más actuales (Google, 2025). Por lo tanto, el desarrollo se alineará con este requerimiento.

Software

Back end

El desarrollo del sistema back-end se realizará utilizando ASP.NET Core en la versión .NET 8, tecnología que proporciona alto rendimiento y favorece la escalabilidad, el mantenimiento, la organización y la extensibilidad del sistema, permitiendo agregar nuevas funcionalidades con facilidad. Además, .NET Core es multiplataforma y puede compilarse a código nativo, lo que posibilita su funcionamiento en cualquier servidor, servicio o contenedor (Microsoft, 2024).

Arquitectura. Para garantizar la independencia de la aplicación back-end respecto a la infraestructura y a los servicios externos, se empleará una arquitectura limpia (*Clean Architecture*), la cual propone un sistema basado en capas: dominio, aplicación (casos de uso), presentación (API o interfaz gráfica del usuario) y la infraestructura.

La capa de dominio es la base fundamental, porque allí se definen los modelos que se gestionarán a través de toda la aplicación; la capa de aplicación, implementa las funcionalidades que satisfacen los requerimientos, mediante la utilización de servicios y gestión de los modelos del dominio; la capa de presentación se encarga de ofrecer acceso y representar los modelos de dominio adecuadamente, y finalmente, la capa de infraestructura, donde se incorpora el acceso a la base de datos, los servicios de análisis e implementaciones de abstracciones definidas en la de aplicación (Smith, 2023).

Esta arquitectura fomentará la separación de responsabilidades, ayudará a organizar el código, reducirá la duplicación de lógica y permitirá establecer límites y restricciones que asegurarán un alto nivel de calidad.

Front end

El front end incluye la etapa de diseño y desarrollo, las cuales implica el uso de distintas herramientas que serán esenciales para obtener resultados de alta calidad.

Diseño. La creación del prototipo navegable se llevará a cabo utilizando la herramienta Figma, que ofrece una gran experiencia para realizar diseños interactivos, y acoplar desde una etapa temprana los flujos de navegación. Su utilización no solo se limitará a esta fase, puesto que también, es una herramienta pensada para que los desarrolladores fácilmente puedan implementar los colores, la tipografía y demás estilos mientras se codifica haciendo uso de su denominado *Dev Mode* (Figma, 2019).

Desarrollo. La aplicación Android se creará en Kotlin, un lenguaje moderno y orientado al desarrollo móvil, que no solo mejora el rendimiento de la aplicación, sino que también asegura una buena experiencia de usuario, puesto que el código es compilado de forma nativa. Además, como Google lo describe, es un lenguaje expresivo y conciso que permite desarrollar eficientemente y el funcionamiento de la aplicación es menos propenso a tener errores (Google, 2024).

Metodología

Metodología de Investigación

Para el proyecto se empleó una investigación cuantitativa descriptiva que se abordó mediante la revisión de antecedentes, indagaciones, boletines y principalmente reportes tanto a nivel nacional como internacional, con el objetivo de describir la problemática, sus variantes, métodos de operación y los efectos que inciden directamente en los usuarios.

La recolección del material bibliográfico se realizó a través de sitios web oficiales de ciberseguridad y de denuncias digitales como Anti-Phishing Working Group (APWG) y el CAI Virtual de la Policía Nacional de Colombia, los cuales almacenan datos y generan reportes que exponen la cantidad de casos registrados de usuarios y organizaciones que han sido víctimas. La elegibilidad del material se determinó mediante una longevidad máxima de 5 años.

Para el análisis, se integraron los datos en un libro de Excel y posteriormente, utilizando Power BI, se diseñaron gráficos de columnas y gráficos circulares determinando la cantidad máxima casos de phishing y el porcentaje de estos.

Metodología de Desarrollo

Dadas las grandes ventajas que ofrecen las metodologías ágiles, se utilizó **SCRUM**, ya que es una metodología flexible que permite ser ajustada según las necesidades, pero ofreciendo lineamientos estandarizados, que soportan el desarrollo de productos eficaces. Es importante destacar que las limitaciones de ser un desarrollador individual dificultarían la aplicación de esta metodología en su totalidad, por lo cual, se extrajeron y emplearon dos de sus características principales: las iteraciones y el *Product Backlog*, aprovechando la organización y misma capacidad de adaptación que son esenciales durante el desarrollo.

Roles

En este caso, al ser un proyecto con un solo participante, los roles fueron cumplidos por el mismo desarrollador.

Product Backlog

El *Product Backlog* como lo define la metodología, es la fuente que indica lo que se necesita para construir y mejorar el producto a partir de una lista de elementos. Para este proyecto en concreto, se utilizaron tres tipos de elementos: *Epic*, tarea y *bug.*

Elemento de Tipo Epic. Este elemento representó aquellas funcionalidades principales del sistema que requerían de varias tareas para ser cumplido en su totalidad. Para la definición de cada elemento se emplearon los casos de uso y requerimientos.

Elemento de Tipo Tarea. Este se empleó para desglosar los elementos de tipo *Epic*, en unidades de trabajo más pequeñas que representasen objetivos más concretos, y, también, permitió reducir la complejidad de otras tareas.

Elemento de Tipo Bug. Este elemento se utilizó con el fin de capturar errores del sistema. Para ser registrados, el comportamiento debía ser considerado incorrecto de acuerdo con los casos de uso y los diagramas de secuencia.

Iteraciones

Cada iteración tuvo una duración de 2 semanas y un objetivo que se buscaba cumplir (*Product Goal*), que, en definitiva, era la implementación de un elemento *Epic* seleccionado del *Product Backlog*.

Sprint Planning. A partir del *Epic* establecido para la iteración, se seleccionaron las tareas relacionadas y estas fueron desglosadas en otras más pequeñas para facilitar el desarrollo y reflejar las acciones necesarias en el aporte al cumplimiento de la funcionalidad.

Daily Scrum. Si bien este evento es usualmente llevado a cabo por equipos para ajustar el plan de la iteración, a manera individual, durante el desarrollo continuo, se presentaron obstáculos que requirieron de cambios en las tareas anteriormente definidas.

Sprint Review. Este evento fue fundamental para fortalecer la solución y a pesar de que no se contó con usuarios directos, si se realizaron pruebas manuales para hallar errores e incluso encontrar nuevas áreas de mejora, que incrementaron la utilidad de las funcionalidades ofrecidas y permitieron pulir la experiencia de usuario, siempre teniendo en cuenta los objetivos del proyecto.

Sprint Retrospective. La retrospectiva, para este caso, no se utilizó.

Cronograma de Actividades

Tabla 1.

Cronograma de actividades del proyecto

Actividades	Mes 1	Mes 2	Mes 3	Mes 4	Mes 5	Mes 6
Sprint 1: Definición del producto mínimo viable, casos de uso y requerimientos.	X					
Sprint 2: Diseño del sistema back-end.	X					
Sprint 3: Diseño del esquema de la base de datos del sistema back-end.		X				
Sprint 4: Implementación de los endpoints de análisis de archivos en el sistema back-end.		X				
Sprint 5: Implementación de los endpoints de análisis de URLs en el sistema back-end.			X			
Sprint 6: Implementación de los endpoints de análisis de mensajes de texto SMS y correos electrónicos en el sistema back-end.			X			
Sprint 7: Implementación de los endpoints para registrar usuarios, iniciar sesión y refrescar tokens de autenticación en el sistema back-end.				X		
Sprint 8: Diseño de la interfaz gráfica del usuario del sistema front-end.				X		
Sprint 9: Desarrollo de las pantallas de las herramientas de análisis completamente funcionales en el sistema front-end.					X	

Actividades	Mes 1	Mes 2	Mes 3	Mes 4	Mes 5	Mes 6
Sprint 10: Desarrollo de las pantallas de los resultados de los análisis completamente funcionales en el sistema front-end.					X	
Sprint 11: Desarrollo de las pantallas de autenticación y configuración en el sistema front-end.						X
Sprint 12: Desarrollo de la funcionalidad para detectar, notificar y analizar mensajes de texto SMS entrantes en el sistema front-end.						X

Fuente. Autoría propia.

Resultados Esperados

Tabla 2.

Resultados y productos esperados del proyecto

Resultado/Producto Esperado	Indicador	Beneficiario
Un sistema de análisis de mensajes de texto SMS, correos electrónicos, archivos y URLs para los usuarios.	Un sistema compuesto por una aplicación back-end encargada de realizar los análisis de mensajes de texto SMS, correos electrónicos, archivos y URLs, y, una aplicación móvil de Android, que representa los resultados de análisis mediante una interfaz gráfica simple e intuitiva.	Usuarios técnicos y no técnicos de dispositivos móviles de Android.
Repositorio público con el código fuente del sistema de análisis.	Repositorios de GitHub con licencia MIT, que ofrecen acceso de lectura a cualquier persona, al código fuente de la aplicación back-end y front-end.	Desarrolladores e investigadores.

Fuente. Autoría propia.

Análisis

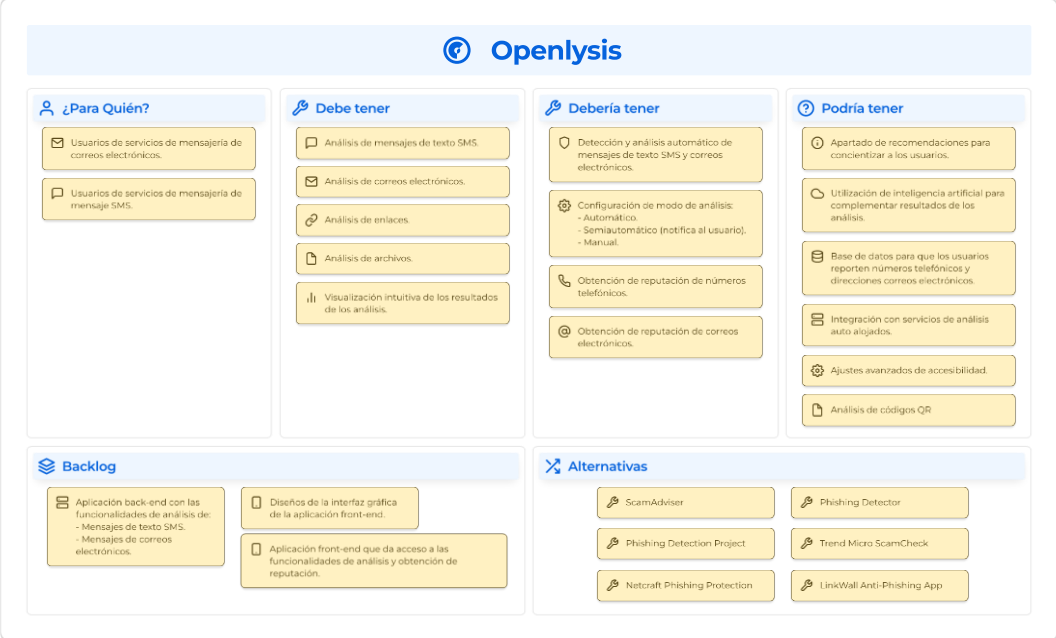
Definida la magnitud del phishing y el smishing, en esta sección de proceder a realizar un análisis del sistema Openlysis, en el que, a partir de los patrones de ataque, se diseña un producto mínimo viable y se identifican y documentan los casos de uso y requerimientos tanto funcionales como no funcionales, para guiar el diseño y desarrollo del sistema propuesto.

Canvas del Producto Mínimo Viable

El producto mínimo viable (MVP), es una herramienta crucial para establecer las bases mínimas que ayudarán a definir los casos de uso y requerimientos de la solución. Para su representación, en la Figura 4, se ha utilizado un canvas MVP enfocado al desarrollo de software.

Figura 4.

Producto mínimo viable de Openlysis



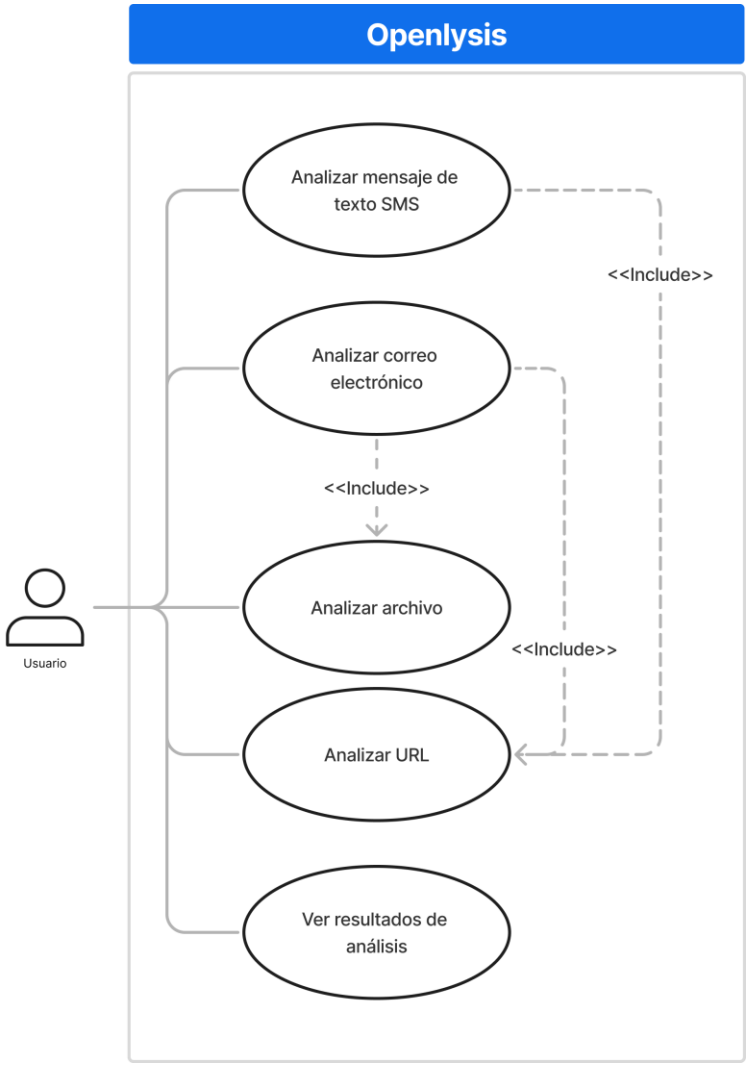
Fuente. Autoría propia.

Casos de Uso

El sistema propuesto se compone de cinco casos de uso principales enfocados al análisis de datos, cada uno conteniendo un flujo principal denominado flujo básico, y, en su posibilidad, flujos alternativos, definiendo diferentes maneras para alcanzar el objetivo de cada caso de uso.

Figura 5.

Casos de uso de Openlysis



Fuente. Autoría propia.

Tabla 3.

Caso de uso de análisis de mensajes de texto SMS

Elemento	Descripción
ID	CU-01
Nombre	Analizar mensaje de texto SMS
Actor Principal	Usuarios que desean analizar algún mensaje de texto SMS recibido.
Objetivo	Determinar si el mensaje de texto de SMS no es smishing o malicioso
Precondiciones	El usuario ha iniciado sesión en la aplicación front-end.
Eventos	• El usuario recibe algún mensaje de texto SMS en su dispositivo móvil. • La aplicación móvil detecta un mensaje de texto entrante.
Flujo Básico	1. Usuario accede a la pantalla de herramientas. 2. Usuario accede a la herramienta de análisis de mensajes de texto SMS. 3. Usuario ingresa remitente del mensaje. 4. Usuario ingresa contenido del mensaje. 5. Sistema valida los datos ingresados. 6. Sistema inicia el análisis. 7. Sistema muestra el estado actualizado del análisis. 8. Sistema notifica la finalización del análisis. 9. Sistema muestra los detalles del análisis y el veredicto del mensaje de texto SMS analizado.
Flujos Alternativos	**FA-01:** 1. Sistema detecta un mensaje de texto SMS entrante. 2. Sistema envía notificación requiriendo permisos para empezar el análisis. 3. Usuario concede permisos. 4. Sistema notifica el inicio del análisis. 5. Sistema notifica el estado del análisis. 6. Sistema notifica la finalización del análisis y el veredicto del mensaje de texto SMS.
Postcondiciones	Los resultados del análisis son almacenados en el dispositivo del usuario.

Fuente. Autoría propia.

Tabla 4.

Caso de uso de análisis de correos electrónicos

Elemento	Descripción
ID	CU-02
Nombre	Analizar correo electrónico.
Actor Principal	Usuarios que desean analizar algún correo electrónico recibido.
Objetivo	• Determinar si el correo electrónico es algún intento de phishing. • Determinar si el archivo adjuntado a un correo electrónico es malicioso.
Precondiciones	El usuario ha iniciado sesión en la aplicación front-end.
Eventos	El usuario recibe algún correo electrónico en alguna de sus cuentas.
Flujo Básico	1. Usuario accede a la pantalla de herramientas. 2. Usuario accede a la herramienta de análisis de correos electrónicos. 3. Usuario ingresa el remitente del mensaje. 4. Usuario ingresa el asunto del mensaje. 5. Usuario ingresa el contenido del mensaje. 6. Usuario selecciona los archivos adjuntos al correo electrónico. 7. Sistema valida los datos ingresados. 8. Sistema valida los archivos seleccionados. 9. Sistema inicia el análisis. 10. Sistema muestra el estado actualizado del análisis. 11. Sistema notifica la finalización del análisis. 12. Sistema muestra los detalles del análisis y el veredicto del correo electrónico analizado.
Postcondiciones	Los resultados del análisis son almacenados en el dispositivo del usuario.

Fuente. Autoría propia.

Tabla 5.

Caso de uso de análisis de archivos

Elemento	Descripción
ID	CU-03
Nombre	Analizar archivo
Actor Principal	Usuarios que desean analizar algún archivo descargado o recibido.
Objetivo	Determinar si el archivo es malicioso.
Precondiciones	El usuario ha iniciado sesión en la aplicación front-end.
Eventos	El usuario descarga algún archivo.
Flujo Básico	1. Usuario accede a la pantalla de herramientas. 2. Usuario accede a la herramienta de análisis de archivos. 3. Usuario selecciona el archivo para analizar. 4. Sistema valida el archivo. 5. Sistema inicia el análisis. 6. Sistema muestra el estado actualizado del análisis. 7. Sistema notifica la finalización del análisis. 8. Sistema muestra los detalles del análisis y el veredicto del archivo analizado.
Postcondiciones	Los resultados del análisis son almacenados en el dispositivo del usuario.

Fuente. Autoría propia.

Tabla 6.

Caso de uso de análisis de URLs

Elemento	Descripción
ID	CU-04
Nombre	Analizar URL.
Actor Principal	Usuarios que desean analizar alguna URL.
Objetivo	Determinar si la URL es maliciosa o algún intento de phishing.
Precondiciones	El usuario ha iniciado sesión en la aplicación front-end.
Eventos	El usuario encuentra o tiene algún enlace para analizar.
Flujo Básico	1. Usuario accede a la pantalla de herramientas. 2. Usuario accede a la herramienta de análisis de URLs. 3. Usuario ingresa la URL para analizar. 4. Sistema valida la URL. 5. Sistema inicia el análisis. 6. Sistema muestra el estado actualizado del análisis. 7. Sistema notifica la finalización del análisis. 8. Sistema muestra los detalles del análisis y el veredicto de la URL analizada.
Postcondiciones	Los resultados del análisis son almacenados en el dispositivo del usuario.

Fuente. Autoría propia.

Tabla 7.

Caso de uso de visualización de resultados de análisis

Elemento	Descripción
ID	CU-05
Nombre	Ver resultados de análisis.
Actor Principal	Usuario que desea ver el veredicto y los detalles de un análisis.
Objetivo	Revisar a detalle los resultados de un análisis.
Precondiciones	El usuario ha analizado un mensaje de texto SMS, un correo electrónico, un archivo o una URL.
Eventos	• El usuario recibe una notificación sobre un análisis de mensaje de texto SMS que está en progreso o ha finalizado. • El usuario inicia un análisis de un mensaje de texto SMS, un correo electrónico, un archivo o una URL. • El usuario accede a la pantalla de resultados.
Flujo Básico	1. Usuario accede a la pantalla de resultados. 2. Usuario accede a resultados de análisis de URLs, archivos, SMS o correos. 3. Sistema verifica la disponibilidad de resultados para mostrar. 4. Sistema muestra lista de resultados. 5. Usuario selecciona un resultado. 6. Sistema muestra detalles del resultado.
Flujos Alternativo	**FA-01:** 1. Sistema muestra notificación de análisis en progreso o finalizado. 2. Usuario abre la notificación. 3. Sistema navega a la pantalla de resultado de análisis. 4. Sistema muestra detalles del resultado.
Postcondiciones	Ninguna.

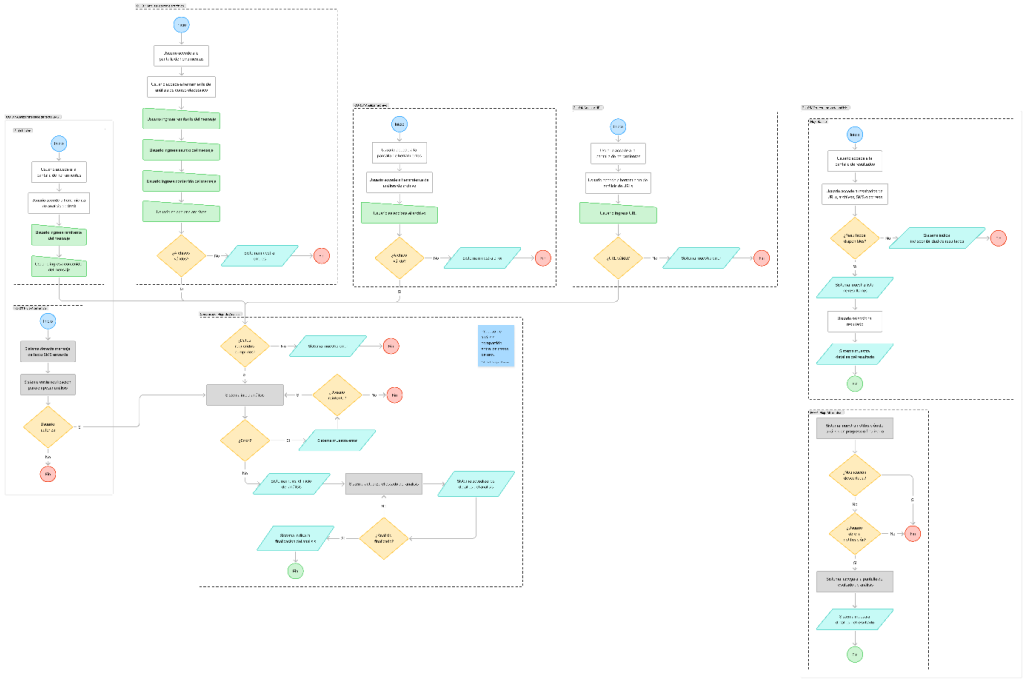
Fuente. Autoría propia.

Adicionalmente, en la Figura 6, se presenta un conjunto de diagramas de flujo que mejoran la interpretación de los casos de uso, definiendo de forma integrada el flujo principal,

los flujos alternativos y flujos excepcionales que pueden ocurrir durante la interacción y funcionamiento general del sistema con el usuario.

Figura 6.

Diagramas de flujo de los casos de uso



Fuente. Autoría propia.

Requerimientos

Definidos los casos de uso, se procede a identificar los requerimientos que serán esenciales principalmente para diseñar y desarrollar el sistema. Para ello, se realiza una división entre los requerimientos funcionales, que estarán enfocados a esas características del sistema para combatir la problemática, y, por otro lado, los requerimientos no funcionales, que serán fundamentales para garantizar una buena experiencia de usuario en el sistema front-end y un rendimiento optimo y robusto en el sistema back-end.

Requerimientos Funcionales

A partir de los casos de uso, se han identificado 25 requerimientos que abarcan las funcionalidades principales, que aseguran el cumplimiento del sistema con los objetivos definidos en el proyecto.

Tabla 8.

Requerimientos funcionales

ID	Nombre	Descripción	Fuente (Caso de Uso)	Priori dad
RF-01	Análisis de mensajes de texto SMS	El sistema debe procesar el remitente y contenido de un mensaje de texto SMS para determinar su nivel de riesgo potencial.	CU-01	Alta
RF-02	Análisis de correos electrónicos	El sistema debe procesar el remitente, asunto, contenido y archivos adjuntos de un correo electrónico para determinar su nivel de riesgo potencial.	CU-02	Alta
RF-03	Análisis de archivos	El sistema debe procesar un archivo para determinar su nivel de riesgo potencial.	CU-03	Alta
RF-04	Análisis de URLs	El sistema debe procesar una URL para determinar su nivel de riesgo potencial.	CU-04	Alta
RF-05	Agrupación de herramientas de análisis	El sistema debe permitir al usuario utilizar las herramientas de análisis a través de una interfaz gráfica unificada.	CU-01, CU-02, CU-03, CU-04	Alta
RF-06	Campos de entrada para análisis de mensaje de texto SMS	El sistema debe ofrecer campos entrada para introducir el remitente y contenido de los mensajes de texto SMS	CU-01	Alta
RF-07	Detección de mensajes de texto SMS	El sistema debe detectar mensajes de texto SMS entrantes en el dispositivo móvil del usuario.	CU-01	Baja

ID	Nombre	Descripción	Fuente (Caso de Uso)	Priori dad
RF-08	Notificaciones de análisis de mensajes de texto SMS	El sistema debe notificar al usuario de los análisis disponibles, iniciados, en progreso y finalizados de los mensajes de texto SMS.	CU-01	Alta
RF-09	Campos de entrada para análisis de correos electrónicos	El sistema debe ofrecer campos de entrada para introducir el remitente, asunto y contenido de los correos electrónicos.	CU-02	Alta
RF-10	Archivos adjuntos de correo electrónico	El sistema debe permitir adjuntar uno o varios archivos asociados a un correo electrónico.	CU-02	Alta
RF-11	Selección de archivos	El sistema debe permitir seleccionar archivos para analizarlos.	CU-03	Alta
RF-12	Validación de tamaño máximo de archivos seleccionados	El sistema debe validar que cada uno de los archivos adjuntos o seleccionados, cumplan con un tamaño máximo.	CU-02, CU-03	Alta
RF-13	Validación de cantidad máxima de archivos	El sistema debe verificar que la cantidad de archivos adjuntos, para analizar un correo electrónico, no superen el límite permitido.	CU-02	Alta
RF-14	Campos de entrada para análisis de URLs	El sistema debe ofrecer un campo de entrada para introducir la URL que debe ser analizada.	CU-04	Alta
RF-15	Validación de URLs	El sistema debe verificar que el texto introducido sea un formato de URL válido.	CU-04	Alta
RF-16	Validación de datos requeridos	El sistema debe validar y asegurar que los datos mínimos requeridos para analizar un mensaje de texto SMS, un correo electrónico, un archivo o una URL, hayan sido introducidos por el usuario en los campos de entrada.	CU-01, CU-02, CU-03, CU-04	Alta

ID	Nombre	Descripción	Fuente (Caso de Uso)	Prioridad
RF-17	Visualización de errores de validación	El sistema debe mostrar errores de validación ocurridos en los campos de entrada.	CU-01, CU-02, CU-03, CU-04	Media
RF-18	Visualización de progreso de análisis	El sistema debe mostrar el progreso del análisis categorizado como “En Cola”, “En Progreso”, “Completado” o “Fallido”.	CU-01, CU-02, CU-03, CU-04	Alta
RF-19	Visualización de veredicto de análisis	El sistema debe mostrar un veredicto, categorizado como “Limpio”, “Sospechoso” o “Malicioso”, como resultado principal de un análisis.	CU-01, CU-02, CU-03, CU-04	Alta
RF-20	Actualización de estado de análisis	El sistema debe permitir al usuario refrescar el estado de un análisis que esta “En Cola” o “En Progreso”.	CU-01, CU-02, CU-03, CU-04, CU-05	Alta
RF-21	Pantalla de resultados	El sistema debe mostrar una pantalla de resultados principal, para agrupar mediante una lista, los diferentes análisis por mensaje de texto SMS, correos electrónicos, archivos y URLs.	CU-05	Alta
RF-22	Navegación a resultados mediante notificación	El sistema debe navegar hacia los resultados de un análisis, cuando un usuario selecciona una de las notificaciones emitidas.	CU-05	Alta
RF-23	Visualización de resultados de análisis	El sistema muestra los resultados de un análisis agrupando el veredicto y el progreso en una sola pantalla.	CU-05	Alta
RF-24	Extracción de URLs	El sistema back-end debe extraer las URLs contenidas en los mensajes de texto SMS y correos electrónicos.	CU-01, CU-02	Alta
RF-25	Autenticación	El sistema debe ofrecer mecanismos de registros e inicios de sesión para utilizar las herramientas de análisis y visualizar los resultados.	Ninguno	Alta

Fuente. Autoría propia.

Requerimientos No Funcionales

Los requerimientos no funcionales incluyen aspectos que buscan complementar los requerimientos funcionales, y de igual manera, se enfocan en definir factores esenciales que permitirán a los usuarios aprovechar el sistema, por ejemplo, la compatibilidad con las versiones de Android es crucial ya que aún se usan dispositivos antiguos, y dejarlos de lado sería reducir el alcance de la propuesta.

Tabla 9.

Requerimientos no funcionales

ID	Nombre	Descripción	Categoría	Prioridad
RNF-01	Errores descriptivos	El sistema debe mostrar errores descriptivos y entendibles al validar los datos ingresados por el usuario o al presentar fallas ocurridas en el sistema.	Usabilidad	Alta
RNF-02	Alertas de privacidad	El sistema debe alertar a los usuarios de no analizar datos confidenciales o privados como lo podrían ser algunos archivos.	Usabilidad	Alta
RNF-03	Claridad del veredicto	El sistema debe presentar el veredicto utilizando un código de colores intuitivo: verde para “Limpio”, amarillo para “Sospechoso” y rojo para “Malicioso”.	Usabilidad	Alta
RNF-04	Idioma de la aplicación de Android	La aplicación de Android debe adaptarse al idioma del dispositivo del usuario, asegurando cobertura en inglés y español como base mínima.	Usabilidad	Alta
RNF-05	Compatibilidad de la aplicación de Android	La aplicación debe ser compatible con la versión 7.0 de Android (API Level 24) o versiones superiores.	Compatibilidad	Alta

ID	Nombre	Descripción	Categoría	Prioridad
RNF-06	Comunicación encriptada	El sistema back-end y front-end deben comunicarse a través de protocolos seguros como HTTPS.	Seguridad	Alta
RNF-07	Almacenamiento de datos de autenticación encriptados	Los datos empleados para autenticarse con el sistema back-end deben ser encriptados antes de ser almacenados por la aplicación de Android.	Seguridad	Alta
RNF-08	Almacenamiento y validación segura de contraseñas	El sistema back-end debe almacenar las contraseñas y validarlas utilizando Argon2id como algoritmo para obtener los valores *hash*.	Seguridad	Alta
RNF-09	Consumo de batería	La detección y análisis automático de mensajes de texto SMS en segundo plano, no debe aumentar el nivel de consumo de batería del dispositivo en más de un 5% en un ciclo normal.	Rendimiento	Media
RNF-10	Documentación del código	El código fuente tanto del sistema back-end como front-end debe incluir comentarios claros en funciones, clases y componentes con acceso público y en aquellos que sean complejos.	Mantenibilidad	Media
RNF-11	Modularidad del sistema back-end	El sistema back-end debe estar diseñado de forma modular, permitiendo que las modificaciones en un servicio de análisis no afecten al resto del sistema	Mantenibilidad	Alta

Fuente. Autoría propia.

Diseño

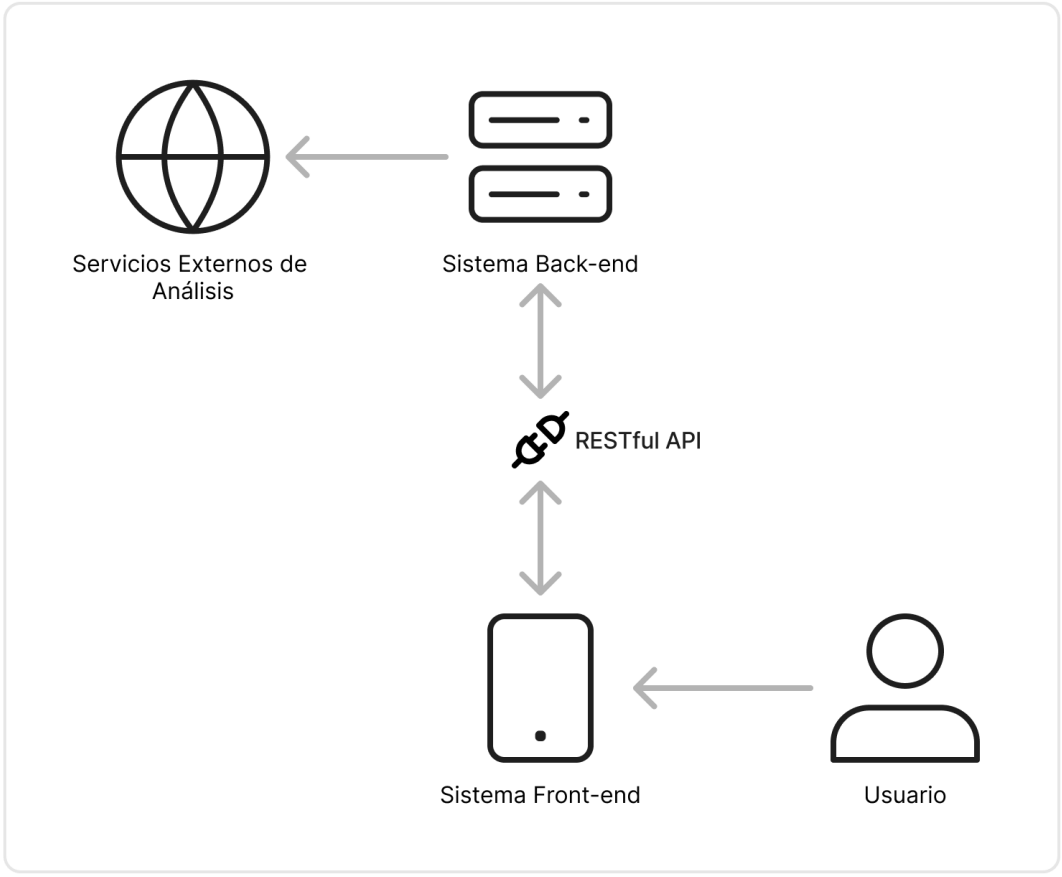
Expuesta la problemática, las necesidades para la solución propuesta y los casos de uso, se diseña un sistema para cumplir con los requerimientos de forma eficiente y robusta. Esto conlleva la preparación de un sistema back-end capaz de procesar peticiones de gran tamaño, la consideración del almacenamiento temporal de archivos y la creación de aplicación intuitiva con el usuario.

Estilo de Arquitectura Global

El sistema global está constituido de cuatro componentes clave: la base de datos para el almacenamiento persistente, el sistema back-end encargado de gestionar las peticiones de los usuarios, una zona de almacenamiento temporal de archivos que deben ser analizados y la aplicación de Android. Cada componente cumple una función esencial, que busca dar robustez, resiliencia y eficiencia al sistema en general.

Figura 7.

Arquitectura global de la solución



Fuente. Autoría propia.

Esta arquitectura propone la separación de roles permitiendo que el back end se enfoque en ofrecer las funcionalidades de análisis principales, mientras el front end se encarga de ofrecer una buena experiencia de usuario.

Beneficios del Sistema Back End

- **Maximización de recursos:** El sistema back-end puede optimizar y aprovechar al máximo el hardware del servidor, por ejemplo, iniciar análisis en paralelo utilizando los hilos disponibles en el procesador del servidor.

- **Escalabilidad:** El sistema back-end puede ser diseñado para funcionar de forma distribuida entre varios servicios en uno o más servidores, facilitando el uso de arquitecturas estandarizadas y enfocadas a sistemas back-end.
- **Mantenibilidad:** La actualización e implementación de características se favorece al tener un hardware dedicado y escalable, y logra que los usuarios se beneficien inmediatamente sin necesidad de lanzar actualizaciones.
- **Desacoplamiento de la presentación:** El sistema no se preocupa por presentar los resultados de los análisis y en su lugar ofrece las funcionalidades a través de una interfaz de programación, permitiendo en el futuro, implementar nuevos sistemas front-end para iOS, Windows, Linux o MacOS.

Beneficios del Sistema Front End

- **Optimización de recursos:** El sistema front-end se encarga de comunicarse con el back end y se concierne de utilizar la menor cantidad posible de recursos y batería en el dispositivo del usuario.
- **Mejor experiencia de usuario:** La tecnología de desarrollo utilizada puede ser elegida dependiendo del sistema operativo, en este caso Android, logrando gran rendimiento a través de aplicaciones nativas, livianas y seguras.
- **Enfocado al sistema operativo del usuario:** El sistema operativo puede tener características únicas que se pueden aprovechar con mayor facilidad a través de ciertos lenguajes de programación, por ejemplo, los denominados *Broadcast* en Android (eventos en el sistema), que pueden ser aprovechados para detectar mensajes de texto SMS entrantes.

Los servicios de análisis son esenciales para cumplir con el objetivo principal del proyecto y a su vez, son el motor principal del sistema back-end. Esto implica seleccionar varios servicios para asegurar diversidad en cuanto a las metodologías de análisis que ofrecen, para así obtener mayor precisión en la detección de phishing y smishing y prevenir ataques *zero-day*.

Algunos servicios se enfocan en analizar URLs y otros también incluyen análisis de archivos, por lo cual, se pueden reutilizar servicios para nutrir diferentes funcionalidades.

Sistema Back-end

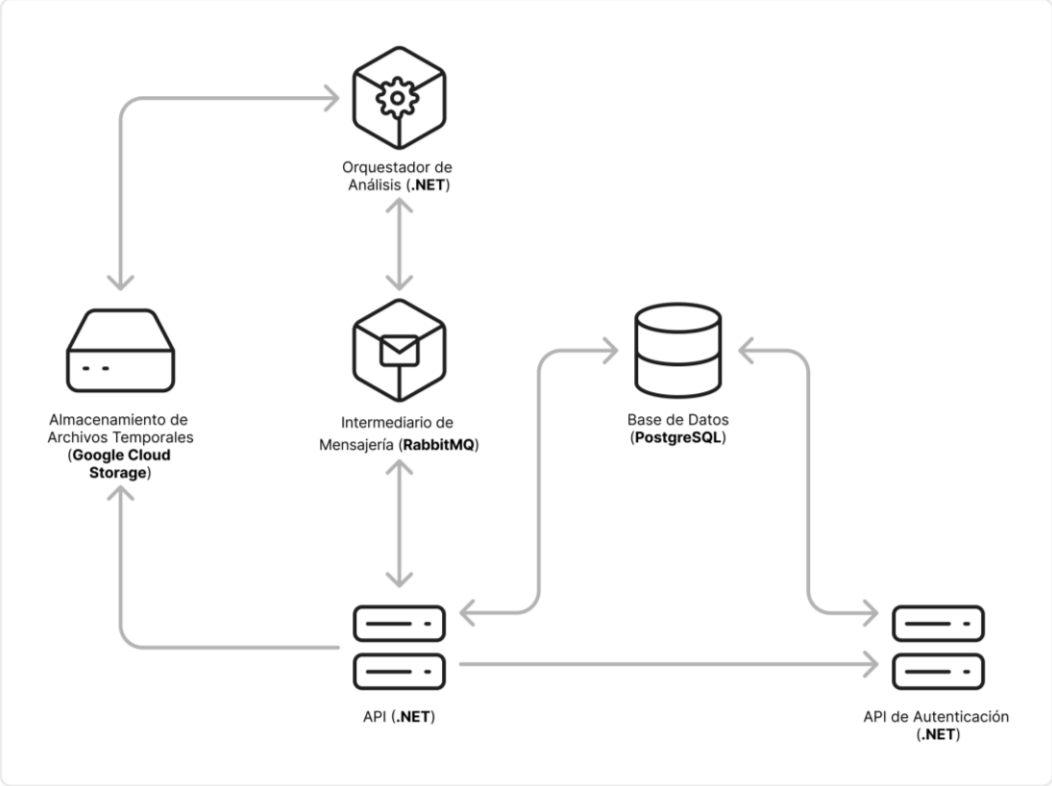
Este sistema se dedica a las funcionalidades principales, lo que implica analizar mensajes de texto SMS, correos electrónicos, URLs y archivos. Para lograr esto, se diseña un sistema interno que emplea los servicios de análisis externos para suplir las peticiones de los usuarios que lleguen desde el sistema front-end.

Estilo de Arquitectura

El sistema back-end está constituido de seis componentes, como se muestra en la Figura 8: La API principal, una API básica de autenticación, un servicio dedicado al inicio y actualización de análisis, un intermediario como mecanismo de comunicación, un almacenamiento para archivos temporales y una base de datos.

Figura 8.

Arquitectura del sistema back-end



Fuente. Autoría propia.

Orquestador de Análisis. Recibe mensajes para iniciar los análisis de archivos y enlaces empleando los servicios externos y de igual manera se encarga de obtener resultados actualizados de los análisis, que posteriormente, son notificados con otro mensaje a la API principal. Gestiona el ciclo de vida final de los archivos asegurándose que sean analizados y posteriormente eliminados. Su beneficio principal es permitir la escalabilidad y la replicación de instancias utilizando contenedores para aumentar la capacidad de análisis de forma concurrente.

API de Autenticación. Ofrece mecanismos básicos de registro e inicio de sesión para los usuarios empleando JSON *Web Tokens* que deben ser utilizados para enviar peticiones a la API principal. Para almacenar los usuarios y los *tokens* de refresco, emplea la base de datos compartida. Su beneficio principal es responsabilizarse por factores esenciales en cualquier sistema, para que en un entorno de producción pueda ser intercambiado o mejorado para ofrecer mecanismos avanzados de autenticación sin requerir de actualizar toda la infraestructura o la API principal.

API Principal. Gestiona las peticiones entrantes para comunicar los análisis que deben ser iniciados, extrae las URLs de los mensajes y correos electrónicos, sube los archivos temporales al almacenamiento, actualiza los análisis externos en la base de datos y verifica que cada petición sea válida y este autenticada. Su beneficio principal es su enfoque en tareas cortas o que no requieran de un uso constante de recursos para así soportar la mayor cantidad de peticiones de los usuarios.

Base de Datos. Medio principal de almacenamiento del sistema back-end, permite guardar los usuarios, los análisis y las relaciones respectivas.

Intermediario de Mensajería. Comunica el orquestador de análisis con la API principal gestionando los diferentes mensajes a través de colas y persistiéndolos en caso de fallas críticas en algún servicio del sistema back-end.

Almacenamiento de Archivos Temporales. Permite guardar los distintos archivos que deban ser analizados, funcionando como un repositorio centralizado para el orquestador de análisis y la API principal, a su vez, permite reducir el consumo de recursos de estos últimos para dedicarlos a la gestión de peticiones y análisis.

Estilo de Diseño

Así como es fundamental establecer una arquitectura para diseñar el sistema back-end, es crucial organizar el desarrollo interno para asegurar buena mantenibilidad y extensibilidad, para ello se utilizan los estilos de diseño, los cuales ayudan a estructurar el código dependiendo de las necesidades.

En este sentido, el sistema back-end, principalmente la API, se construye siguiendo el diseño denominado arquitectura limpia (*Clean Architecture*), que promueve la separación de responsabilidades a través de varias capas o niveles: capa de dominio, capa de aplicación, capa de infraestructura y capa de presentación. Esta estructura facilita la implementación de pruebas, y especialmente, permite que el sistema no dependa de la infraestructura, por ejemplo, no este atado a un sistema de base de datos en específico, sino que, por el contrario, todo se construya a partir del dominio, que se encarga de especificar como se representan los análisis externos y de qué manera se integran en un modelo con un veredicto unificado.

Capa de Dominio. Representa el dominio a través de objetos que cumplen reglas para asegurar consistencia en todo el sistema.

Capa de Aplicación. Contiene los servicios encargados de cumplir con los casos de uso definidos, esta capa depende únicamente de la capa de dominio para funcionar adecuadamente y expone sus necesidades mediante contratos.

Capa de Infraestructura. Define las tecnologías o servicios externos que se emplean para satisfacer las necesidades de la capa de aplicación. Este nivel depende del dominio y la aplicación.

Capa de Presentación. Encargado de representar el dominio a nivel externo y de emplear los servicios de la capa de aplicación para exteriorizar las funcionalidades del sistema back-end, esta capa en definitiva es la API *RESTful*. Este nivel depende de la capa de dominio, de aplicación y la capa de infraestructura.

Servicios de Análisis Externos

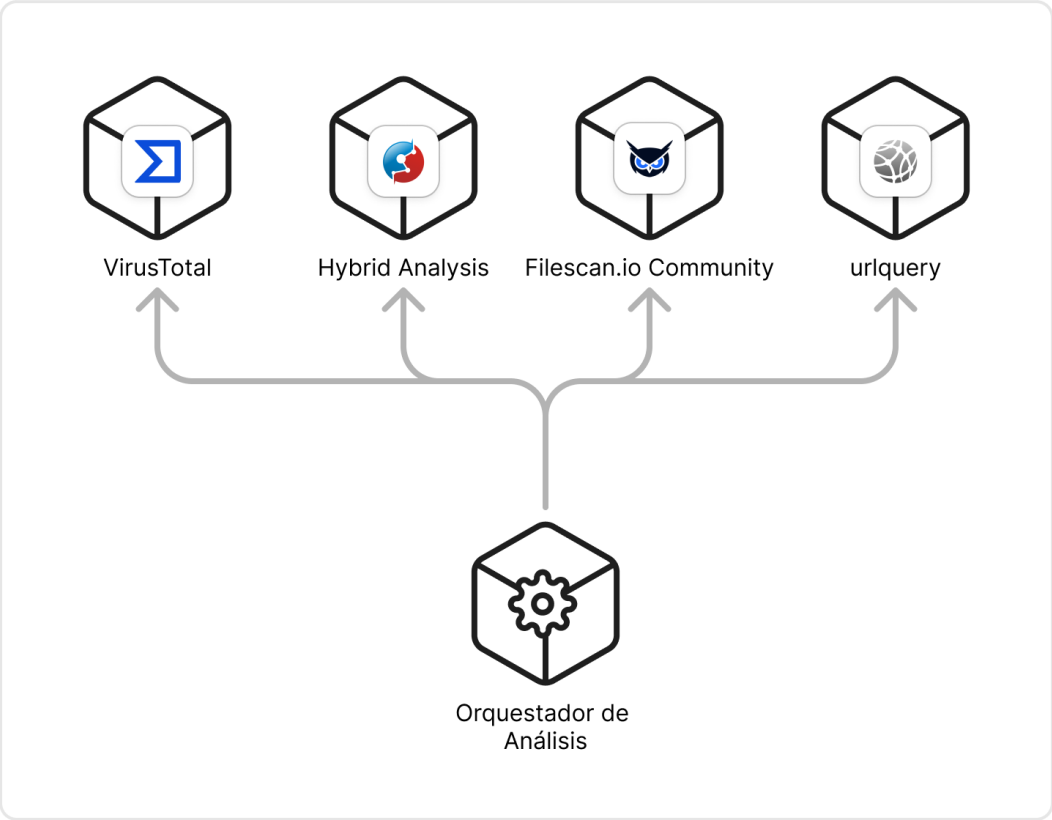
La idea de estos servicios es aprovechar su especialización en la detección de software malicioso y sus diferentes metodologías para alcanzar su objetivo, como la utilización de entornos *sandbox*, que ejecutan archivos o acceden a páginas web desde enlaces y realizan análisis dinámicos que aumentan la probabilidad de detección. La arquitectura de desarrollo que se emplea en el orquestador de análisis del sistema back-end da lugar a que el sistema sea flexible y no sea funcional únicamente con unos cuantos servicios de análisis, sino que, en su lugar, cada uno pueda ser acoplado o desacoplado sin afectar al orquestador.

Es clave destacar que algunos de los servicios mencionados a continuación, están sujetos a limitaciones de uso con el fin de acotar acciones malintencionadas por atacantes, como, por ejemplo, comprender los mecanismos de detección aplicados internamente por el servicio. Sin embargo, estos límites pueden ser removidos o ampliados realizando procesos más complejos en el cual se deben investigar los antecedentes del solicitante. Por lo tanto, en un entorno de producción, sería fundamental aplicar a estos procesos, o considerar la implementación de servicios basados en modelos de aprendizaje automático, que, de acuerdo con algunas investigaciones expuestas en los antecedentes, son altamente precisos y podrían ser una alternativa robusta y eficiente.

Destacado lo anterior, se enuncian los servicios de análisis que se implementan para el sistema back-end y que específicamente, son acoplados al orquestador de análisis, con el fin de ofrecer análisis estático, dinámico y enfocados al escaneo de sitios web contra phishing.

Figura 9.

Servicios de análisis externos acoplados en el back end



Fuente. Autoría propia.

VirusTotal. Servicio conocido por emplear más de 70 motores de antivirus para detectar malware en archivos, listas de bloqueo de URLs y dominios para encontrar enlaces maliciosos y herramientas para extraer datos y hallar patrones sospechosos. Empleado para analizar archivos y enlaces.

Hybrid Analysis. Como su nombre lo indica, es un servicio que utiliza una metodología híbrida en la que combina análisis manuales y automatizados en archivos y enlaces. Una de sus

principales ventajas es que ofrece los mencionados entornos *sandbox* proporcionando una capa de detección adicional. Empleado para analizar archivos y enlaces.

Filescan Community. Al igual que Hybrid Analysis, provee análisis dinámico, pero a través de unos motores emulados que ofrecen mayor velocidad en comparación con los métodos tradicionales como los entornos *sandbox*. Empleado para analizar archivos y enlaces.

Urlquery. En contraste con los servicios anteriores, este está enfocado únicamente al análisis de enlaces. Además, aunque sigue empleando mecanismos similares, este incluye el uso de NIDS (Sistema de detección de intrusiones basadas en la red) y analiza el contenido real del sitio web al que redirige el enlace.

Diagramas de Clase

Los diagramas de clase en este caso se diseñan para la capa de dominio de la API principal, cuyo objetivo es abstraer el código y visualizar los distintos objetos que se gestionan a través del sistema back-end, incluyendo en el orquestador de análisis. Esto estandariza la comunicación entre los distintos servicios y asegura consistencia tanto a nivel de funcionamiento como de almacenamiento, especialmente en el repositorio que se implementa en la capa de infraestructura.

Para visualizar adecuadamente la composición del dominio, se utilizan cinco diagramas: el primero representa los objetos compartidos, el segundo expone a aquellos relacionados con los análisis de URLs y archivos, el tercero con los análisis de mensajes de texto SMS y correos electrónicos, el cuarto muestra los modelos empleados para la autenticación y el último plasma la estructura jerárquica de herencia.

Respecto a las URLs y archivos, entendiendo que se incorporan varios servicios externos, cada uno retornando un objeto de forma independiente, se realiza una agrupación por archivo o

URL, los cuales se denominan FileMultiAnalysis y UrlMultiAnalysis, que, además, reflejan un estado final que tiene en cuenta a todos los análisis agrupados.

Cabe mencionar el enfoque *Domain Driven Design* (DDD) (Redis, s.f.), que se usa para el modelamiento, si bien no se aplica de forma estricta, si se utilizan principalmente los siguientes conceptos ya que ayudan de forma simple a mantener mayor consistencia.

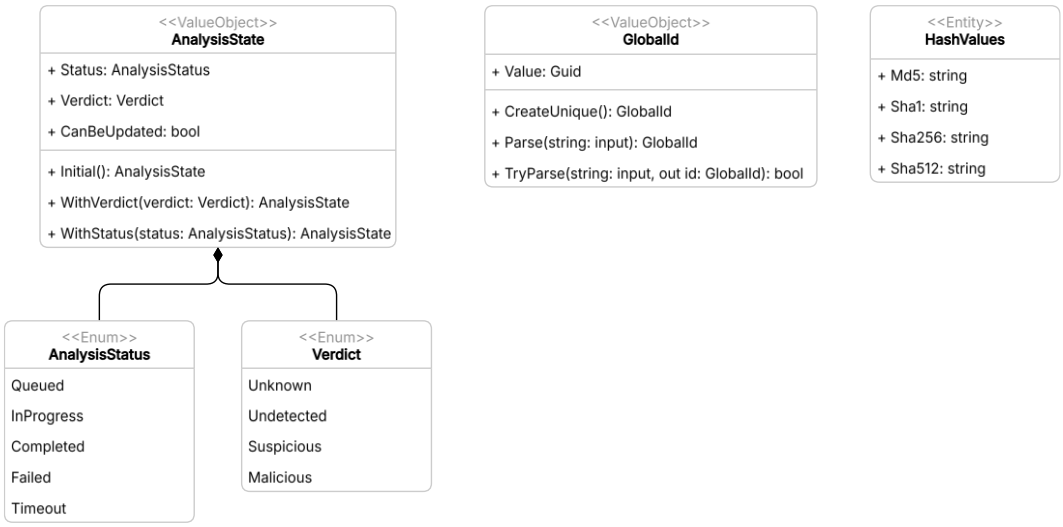
Entidades. Estos objetos, que en inglés son referidos como *entities*, poseen una identidad única. Generalmente se usa un identificador para saber si dos entidades son iguales.

Objetos de Valor. Estos objetos, que en inglés son referidos como *value objects*, son identificados a partir de sus atributos y no de un valor único, es decir, su igualdad únicamente se cumple si sus propiedades poseen los mismos valores. Algunos ejemplos pueden ser direcciones, URLs o cantidades monetarias.

Agregados. Estos objetos, que inglés son referidos como *aggregate root*, están encargados de agrupar y asegurar la integridad de entidades y objetos de valor relacionados entre sí. Al igual que las entidades, posee una identidad única, sin embargo, este provee métodos para modificar controladamente las entidades y objetos de valor agrupados.

Figura 10.

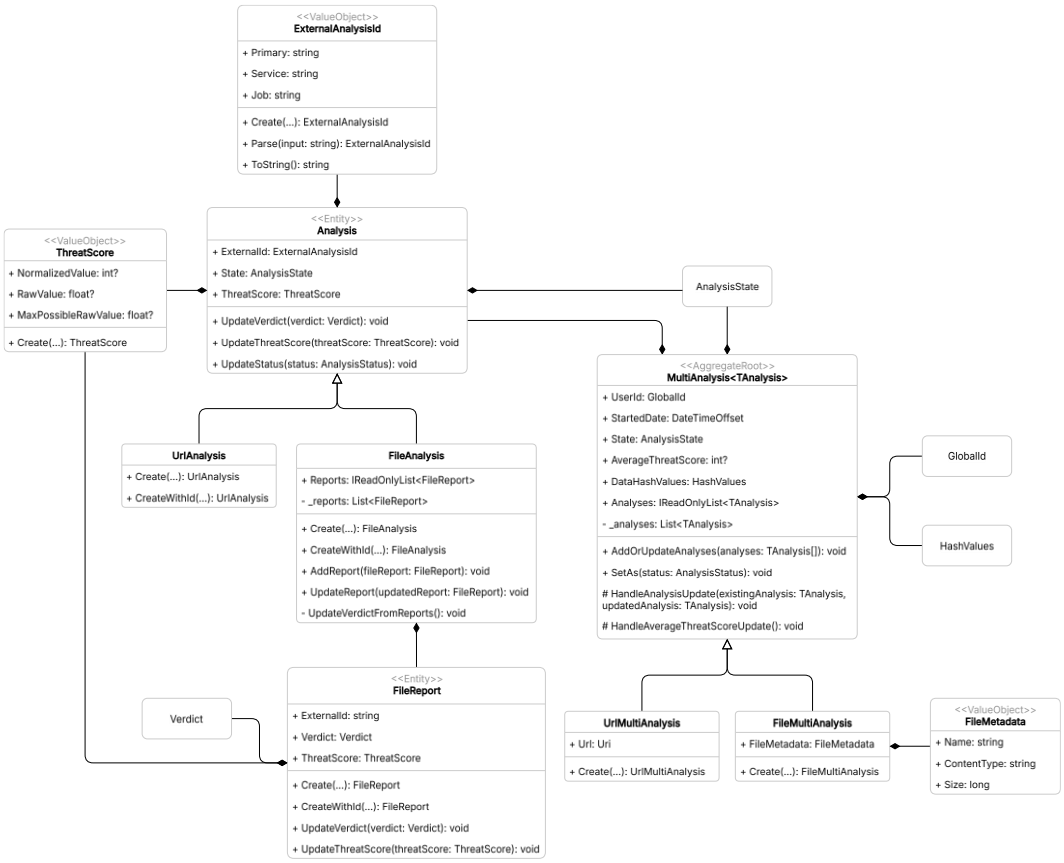
Diagrama de clases de los objetos compartidos en el back end



Fuente. Autoría propia.

Figura 11.

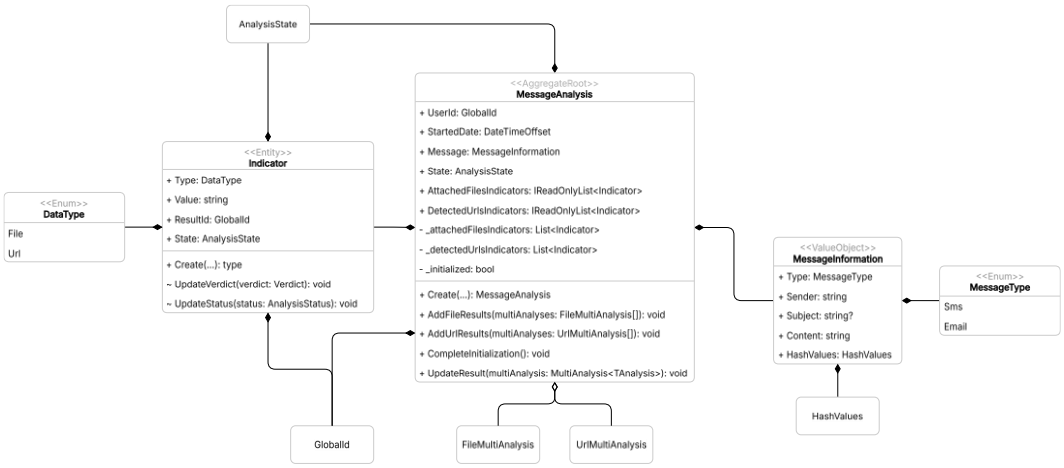
Diagrama de clases de los análisis de URLs y archivos del back end



Fuente. Autoría propia.

Figura 12.

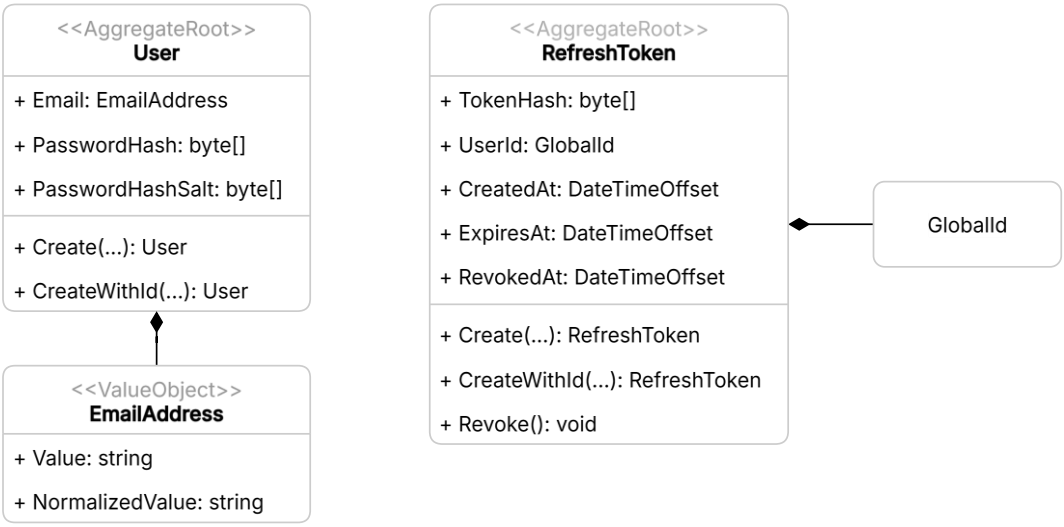
Diagrama de clases de los análisis de mensajes del back end



Fuente. Autoría propia.

Figura 13.

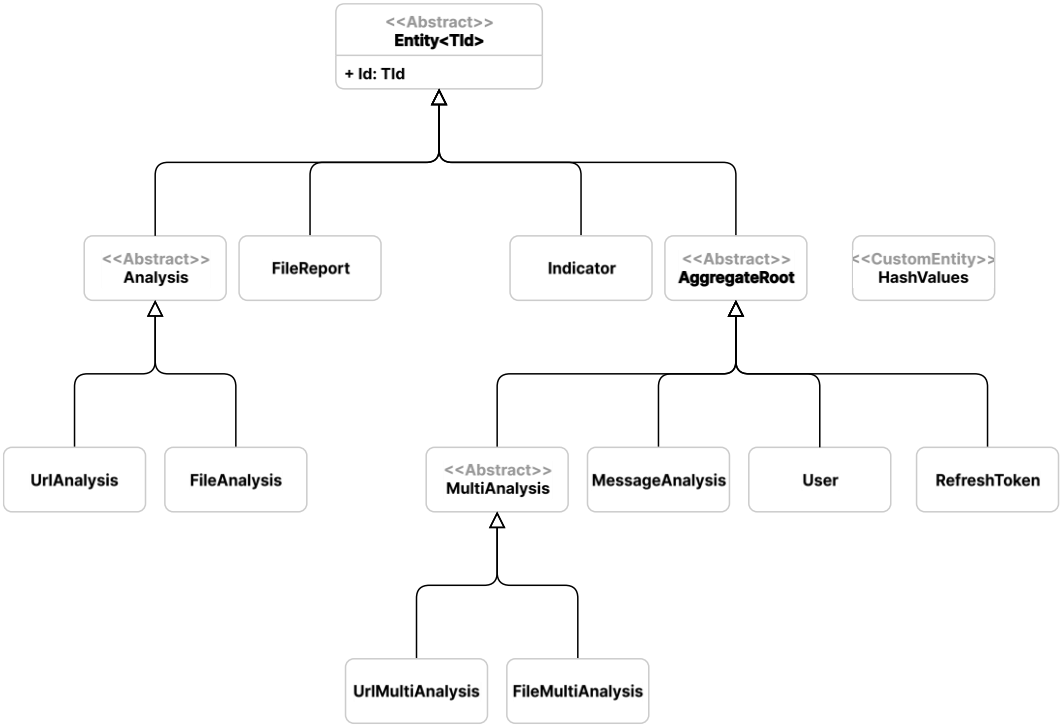
Diagrama de clases de los objetos de autenticación del back end



Fuente. Autoría propia.

Figura 14.

Diagrama de clases de la estructura jerárquica de herencia del back end



Fuente. Autoría propia.

Diagramas de Secuencia

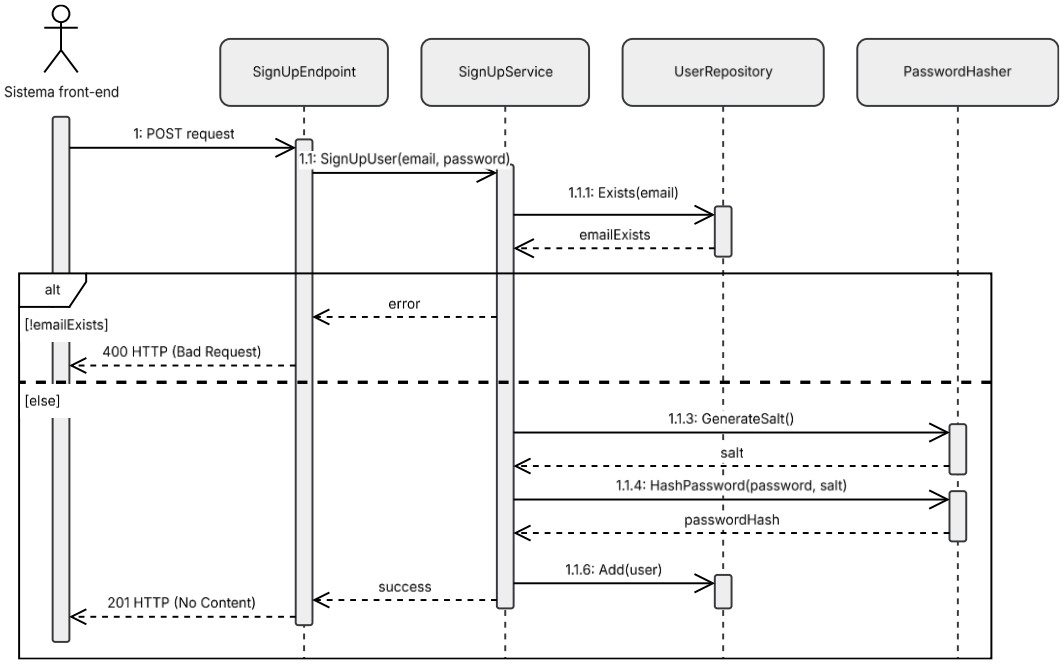
Para entender el funcionamiento del sistema, se emplean varios diagramas de secuencia que representan de forma global el conjunto de objetos que intervienen en la interacción de cada caso de uso. Adicionalmente, se incluyen diagramas que ilustran otras funcionalidades relevantes para la comprensión del sistema. Cabe recalcar que estos diagramas se enfocan en ofrecer una perspectiva técnica cercana al código.

En este sentido, se muestran ocho diagramas orientados a los siguientes procesos: el registro de usuarios, el inicio de sesión, el análisis de archivos, el análisis de URLs, el análisis de mensajes que incluye a los SMS y correos electrónicos, la coordinación de análisis con los

servicios externos, la actualización de los análisis coordinados en la base de datos y, finalmente, la obtención de los resultados de dichos análisis.

Figura 15.

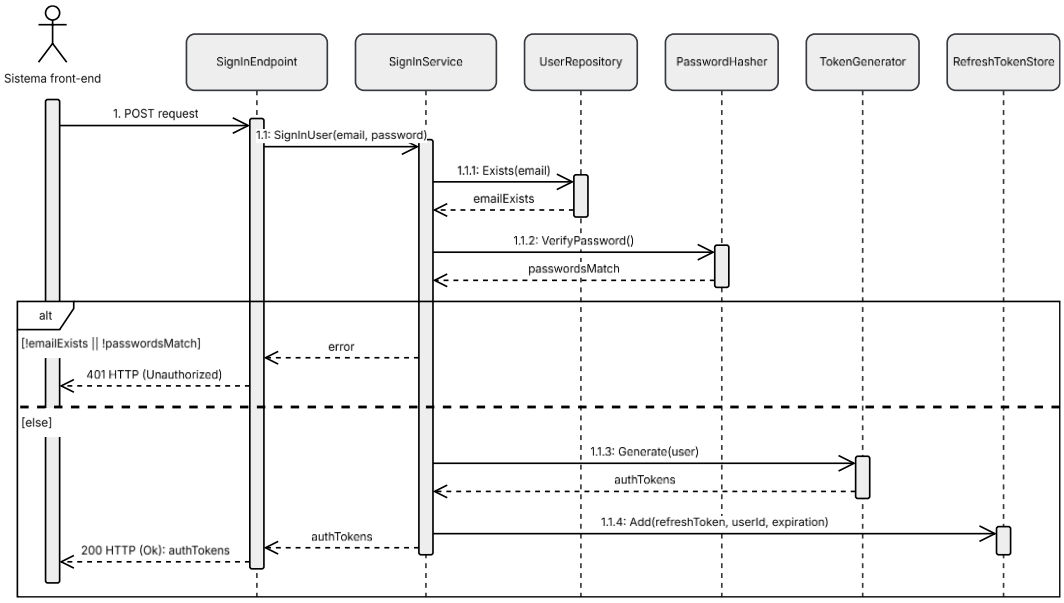
Diagrama de secuencia del registro de usuarios en el back end



Fuente. Autoría propia.

Figura 16.

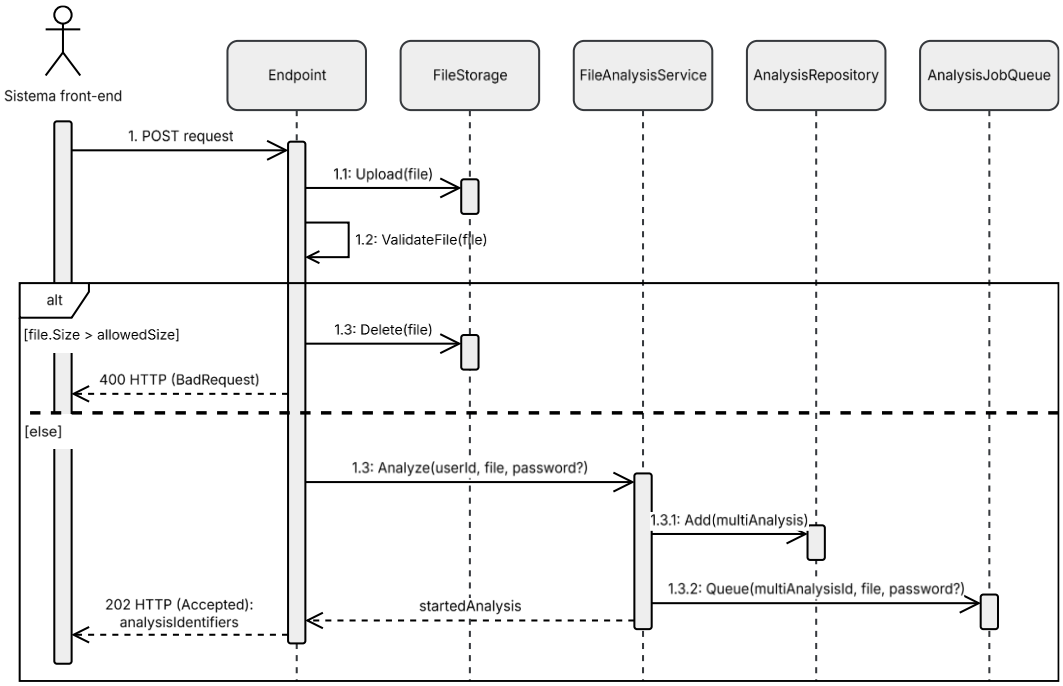
Diagrama de secuencia del inicio de sesión en el back end



Fuente. Autoría propia.

Figura 17.

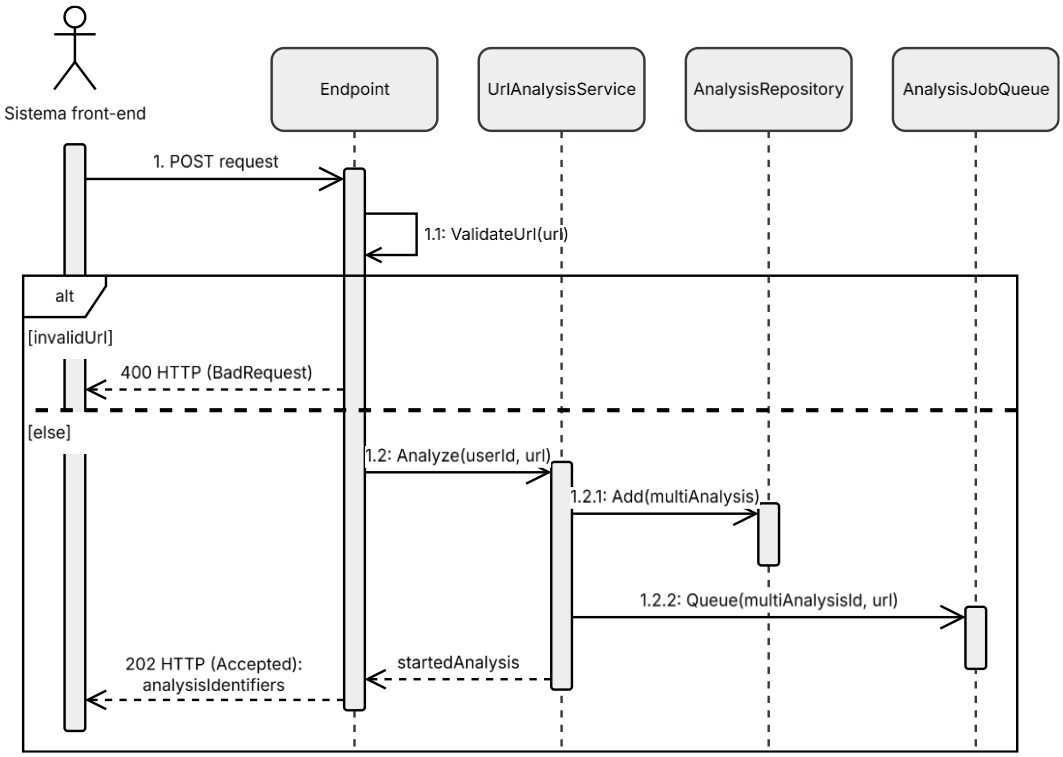
Diagrama de secuencia de los análisis de archivos en el back end



Fuente. Autoría propia.

Figura 18.

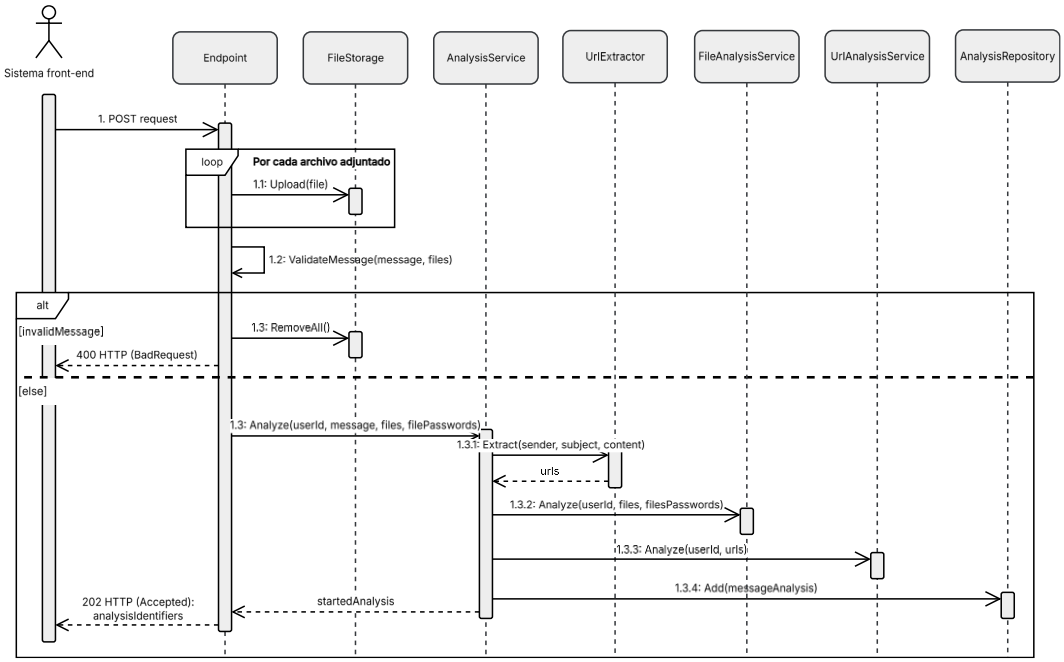
Diagrama de secuencia de los análisis de URLs en el back end



Fuente. Autoría propia.

Figura 19.

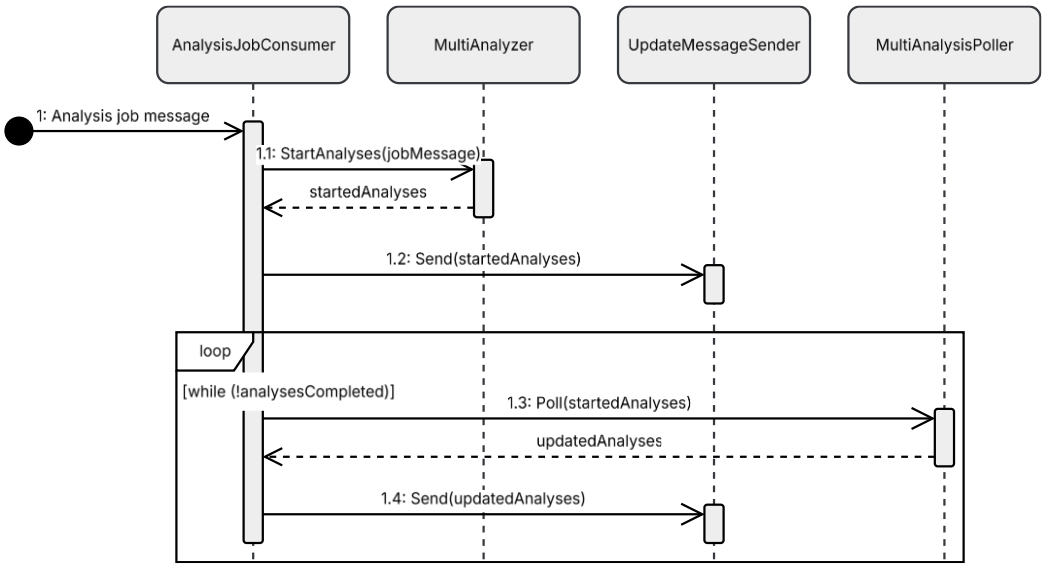
Diagrama de secuencia del análisis de mensajes en el back end



Fuente. Autoría propia.

Figura 20.

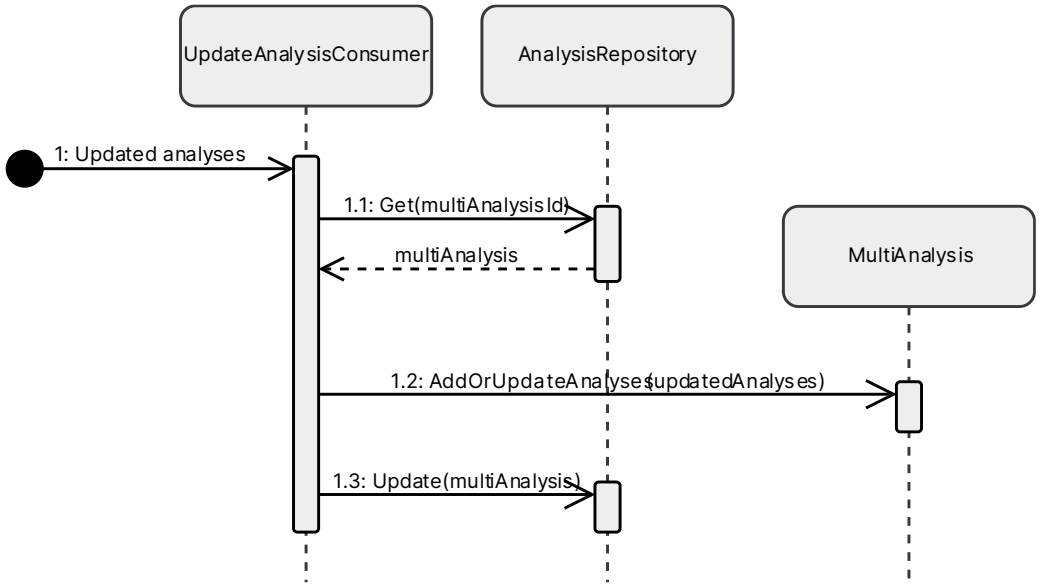
Diagrama de secuencia de la coordinación de análisis con servicios externos



Fuente. Autoría propia.

Figura 21.

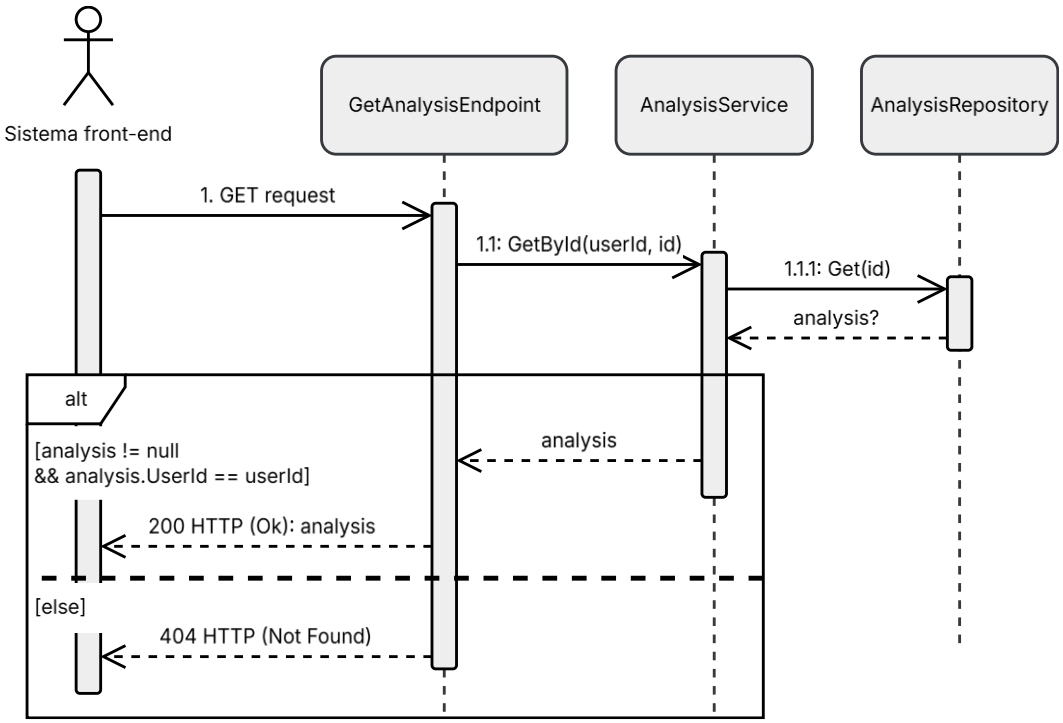
Diagrama de secuencia de la actualización de análisis en la base de datos



Fuente. Autoría propia.

Figura 22.

Diagrama de secuencia de la obtención de los resultados de análisis



Fuente. Autoría propia.

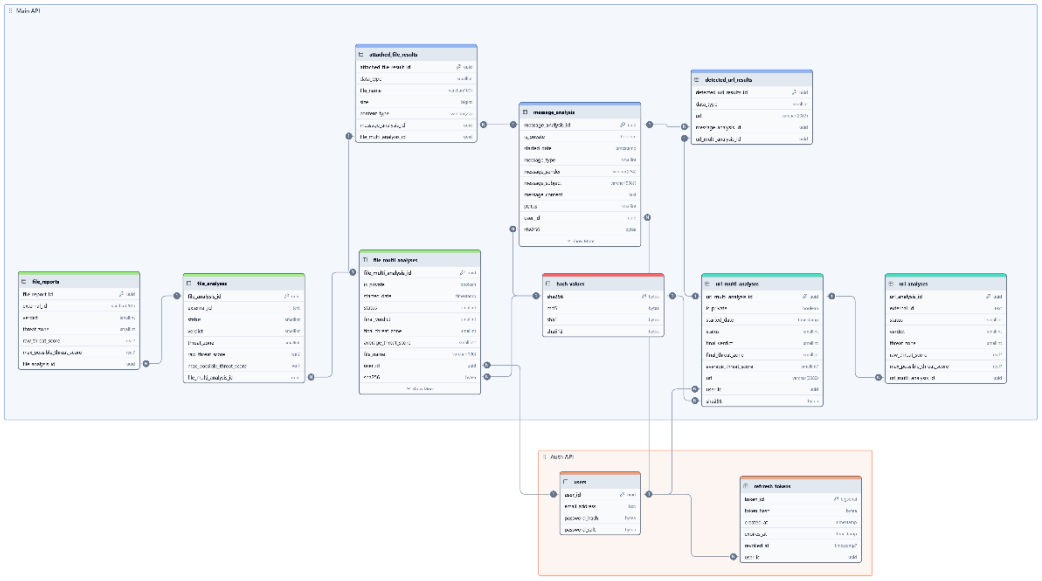
Base de Datos

En base al funcionamiento y principalmente a los modelos de dominio definidos anteriormente, se diseña el esquema de la base de datos, que es esencial para persistir los análisis, actualizarlos según su estado y recuperarlos cuando el sistema front-end lo requiera.

Este esquema incluye tablas enfocadas a los análisis de archivos, URLs y mensajes, y también para la gestión de usuarios y el control de autenticación. A su vez, se emplean tipos de datos optimizados como los arreglos de bytes (bytea) para almacenar los valores hash, y los identificadores universalmente únicos versión 7 (UUID v7) para asegurar la unicidad de las entidades y asegurar un rendimiento eficiente en las inserciones y consultas. El diagrama puede ser observado con mayor detalle usando el enlace referenciado en el Apéndice B.

Figura 23.

Esquema de la base de datos del back end



Fuente. Autoría propia.

Sistema Front-end

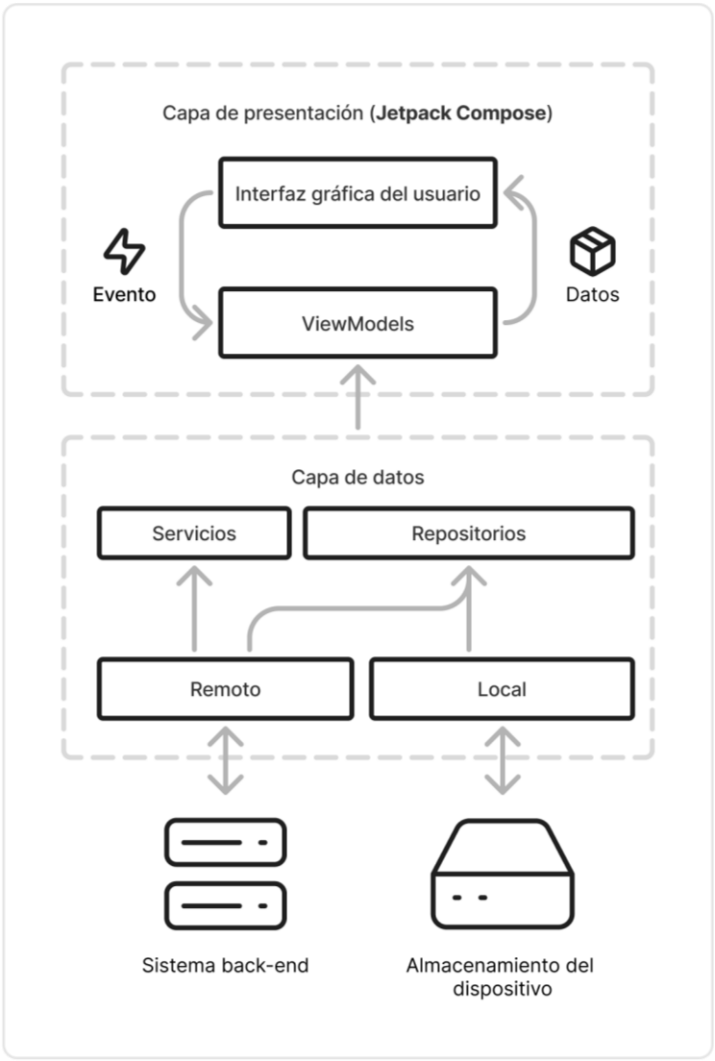
Este sistema en comparación con el back end es mucho más simple a nivel estructural, ya que solo es una aplicación de Android, esto ayuda en gran medida a que el rendimiento sea optimo en el dispositivo y que su principal tarea sea ofrecer una buena experiencia para el usuario. Sin embargo, esto involucra la creación de diseños para soportar el desarrollo de la interfaz y la utilización de características propias del sistema operativo.

Estilo de Arquitectura

La arquitectura de este sistema en vez de estar compuesta entre varios servicios se cimenta a través de capas focalizadas a cumplir responsabilidades separadas, dentro de la misma aplicación de Android. En la Figura 24, se ilustra la lógica general de estas capas.

Figura 24.

Arquitectura del sistema front-end



Fuente. Autoría propia.

Capa de Datos. Provee acceso al sistema back-end para utilizar las funcionalidades principales de forma remota y, al almacenamiento del dispositivo, este último, está orientado a cumplir con la filosofía offline-first que propone Android, la cual indica que una aplicación debe ser capaz de cumplir todas, o un subconjunto de las funcionalidades críticas de forma local, es decir, sin estar conectado a una red con acceso a internet (Google, 2025). Para este caso, el almacenamiento local, se usa principalmente para almacenar los diez análisis más recientes del

usuario. Adicionalmente, aquí se exponen los repositorios que centralizan el acceso a los datos tanto a nivel local como remoto, asegurando consistencia.

Capa de Presentación. Encargada de representar los elementos gráficos que permiten invocar eventos a través de la interacción con el usuario, y a su vez, muestra la información proveniente de la capa de datos. Este nivel incluye los llamados *ViewModels* o *state holders*, cuyos objetivos son encapsular la lógica que se debe aplicar cuando se genera un evento y, obtener la información que sea requerida por medio de los repositorios o servicios de la capa de datos (Google, 2025).

El uso de esta arquitectura facilita la mantenibilidad ya que permite modularizar la aplicación entre distintos componentes con responsabilidades claramente delimitadas, lo que ayuda a comprender y trabajar con mayor facilidad en el sistema. También ayuda a mejorar la gestión de recursos del dispositivo, en lugar de cargar datos en la memoria directamente cuando se inicia la aplicación, se emplean los servicios de la capa de datos únicamente cuando sea necesario y a través de métodos de almacenamiento adecuados y optimizados para sus propósitos (Google, 2025).

Estilo de Diseño

La aplicación de Android sigue un estilo orientado a la capa de presentación denominado *Model-View-ViewModel* (MVVM) que ayuda a mantener el código organizado, especialmente en la interfaz del usuario, su filosofía dispone tres componentes: las vistas (*View*), los modelos (*Model*) y el modelo de vista (*ViewModel*).

Vista. Representa la interfaz gráfica del usuario, la cual incluye la estructura, esquema de colores y apariencia en general del sistema. Este componente se encarga de dibujar los elementos gráficos, representar los datos que se encuentran en el modelo y notificar los eventos, invocados a través de la interacción con el usuario, al modelo de vista.

Modelo. Encapsula los distintos datos que se representan en la vista y usualmente son objetos inmutables, es decir, que no puede ser modificados una vez son creados, lo que asegura consistencia en la interfaz del usuario.

Modelo de Vista. Responsable de gestionar los eventos notificados desde la vista aplicando la lógica correspondiente, la cual puede incluir la recreación del modelo y el uso de servicios o repositorios definidos en la capa de datos, para obtener o actualizar información. Finalmente, expone el modelo actualizado, listo para que la vista nuevamente lo represente y refleje los cambios en la interfaz de usuario.

Adicionalmente a lo anterior, se deben agregar otros dos conceptos fundamentales que complementan la comprensión de este estilo de diseño, el estado de la interfaz de usuario y el patrón de flujo unidireccional de datos (UDF).

Estado de la Interfaz de Usuario (UI State). Este concepto, en inglés referido como *UI* state, está muy relacionado con el componente modelo de MVVM, sin embargo, además de los datos, también encapsula otra información esencial como errores de validación o condiciones, que están primordialmente relacionados con la interfaz de usuario, pero que se producen a partir de la lógica de los *ViewModels*. Básicamente, el *UI State*, indica lo que la vista debe representar y lo que el usuario debería ver (Google, 2025).

UDF. La comunicación mencionada anteriormente entre los componentes del MVVM se basa en el patrón UDF, que, de forma simple, propone la utilización de una comunicación unidireccional para notificar los eventos de la vista al *ViewModel*, y así mismo, el *ViewModel* expone el *UI State,* que es transmitido hacia la vista. Este flujo propone la separación de responsabilidades asegurándose que la vista únicamente se enfoque en representar el modelo y comunicar los eventos, y el *ViewModel* sea la única fuente de la verdad, cuyo compromiso es producir dicho modelo a partir de esos eventos, como se muestra en la (Google, 2025).

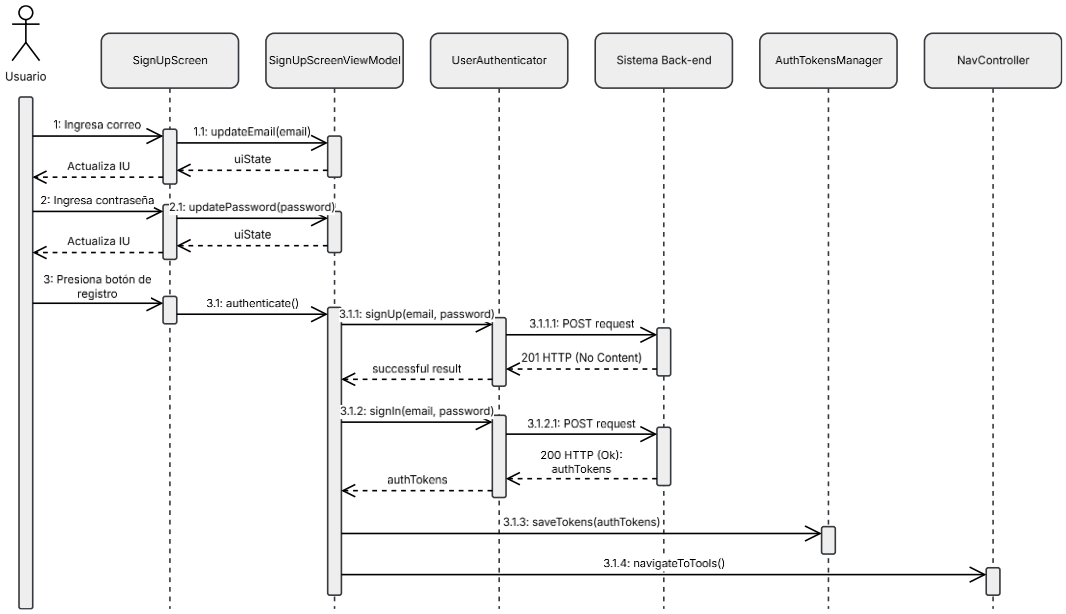
Diagramas de Secuencia

Para representar el comportamiento del sistema front-end a nivel general, se definen los diagramas de secuencia, que a diferencia de los diseñados para el sistema back-end incluyen una interacción directa con el usuario final, quien es el actor principal y origina los eventos que consecuentemente ejecutan las secuencias.

Los diagramas están dirigidos principalmente a los casos de uso, como el análisis de mensajes, URLs y archivos, sin embargo, al igual que en el sistema back-end, se incluye el registro e inicio de sesión de usuarios. Además, es importante resaltar que la lógica de inicio de los análisis es idéntica en los diferentes casos, por ello, se unifica mediante un diagrama compartido como se muestra en las Figura 27, Figura 28, Figura 29 y Figura 30, que hacen referencia a la Figura 31.

Figura 25.

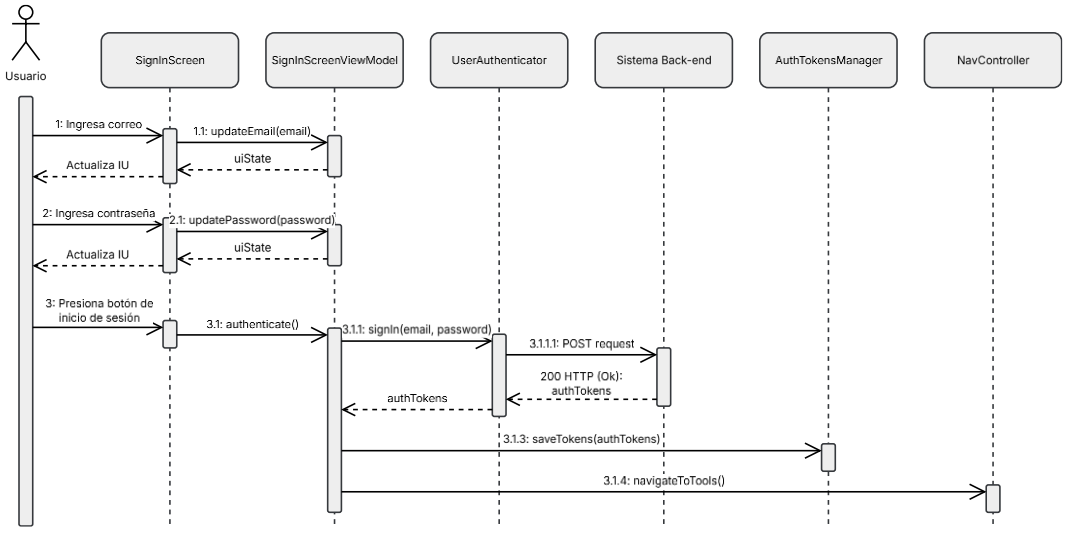
Diagrama de secuencia del registro de usuarios en el front end



Fuente. Autoría propia.

Figura 26.

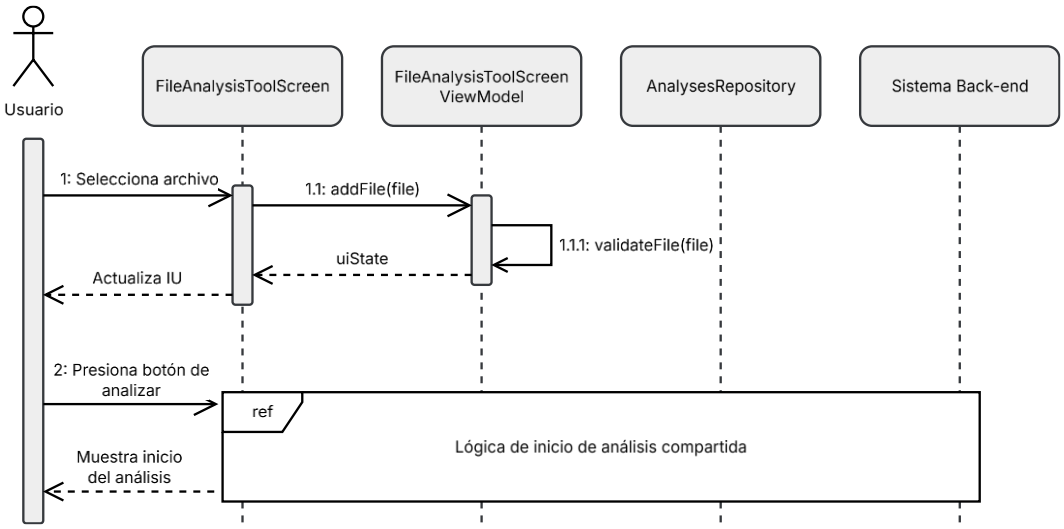
Diagrama de secuencia del inicio de sesión en el front end



Fuente. Autoría propia.

Figura 27.

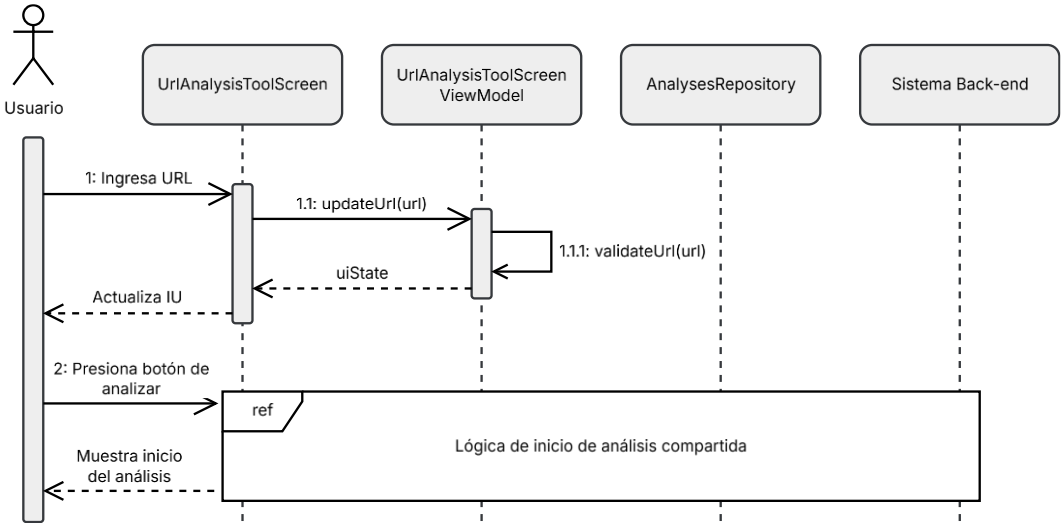
Diagrama de secuencia del análisis de archivos en el front end



Fuente. Autoría propia.

Figura 28.

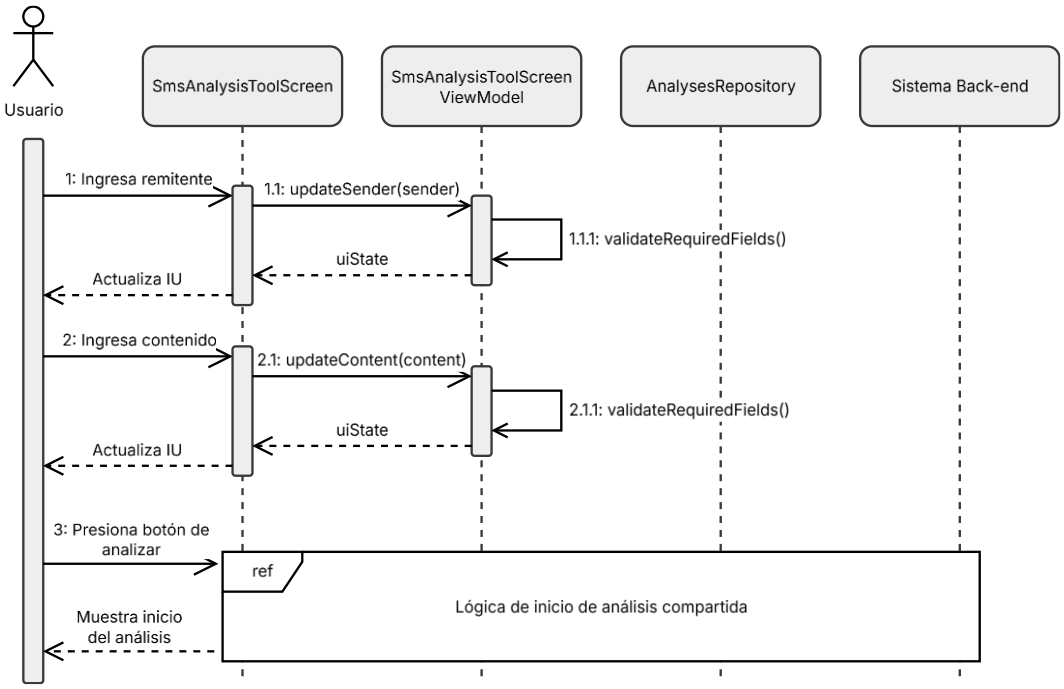
Diagrama de secuencia del análisis de URLs en el front end



Fuente. Autoría propia.

Figura 29.

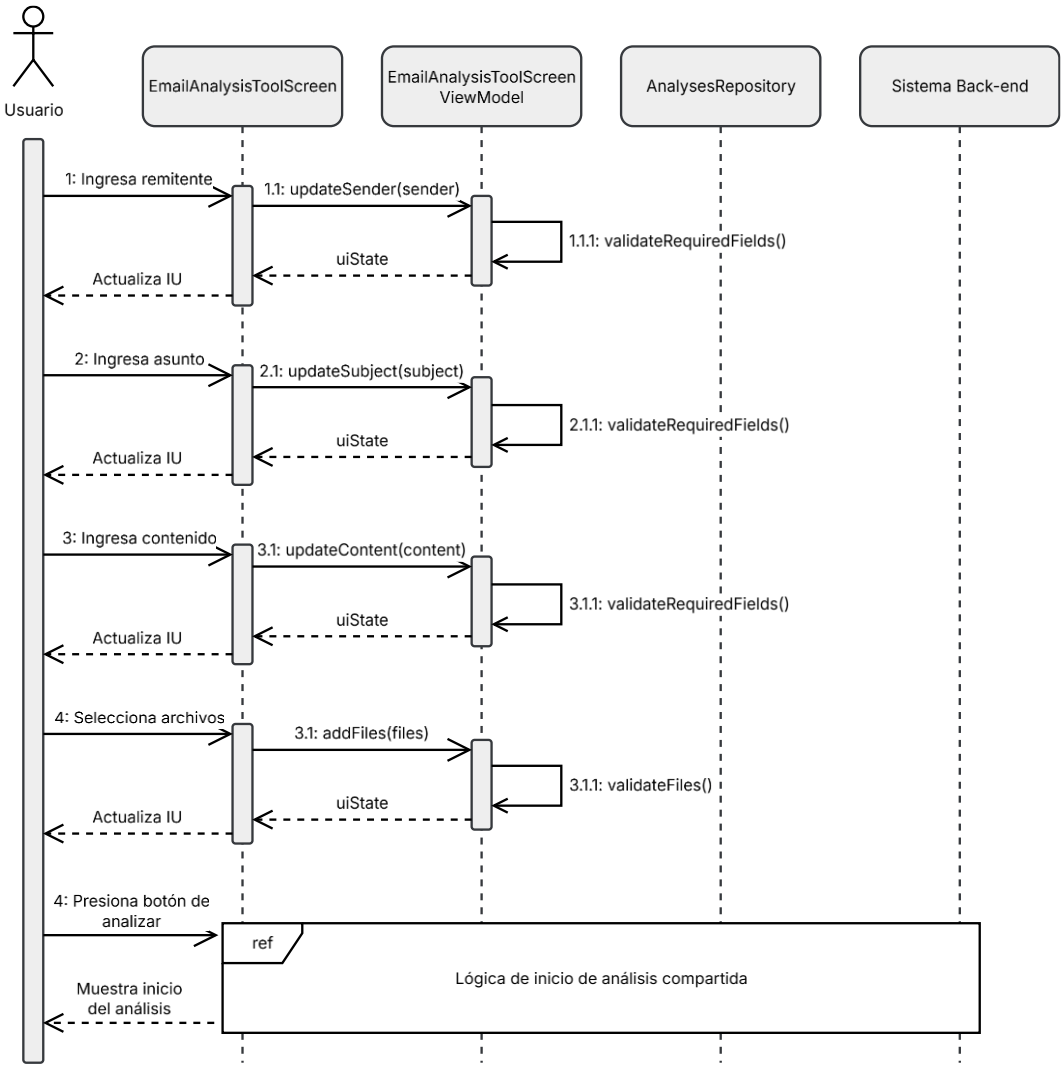
Diagrama de secuencia del análisis de mensajes de texto SMS en el front end



Fuente. Autoría propia.

Figura 30.

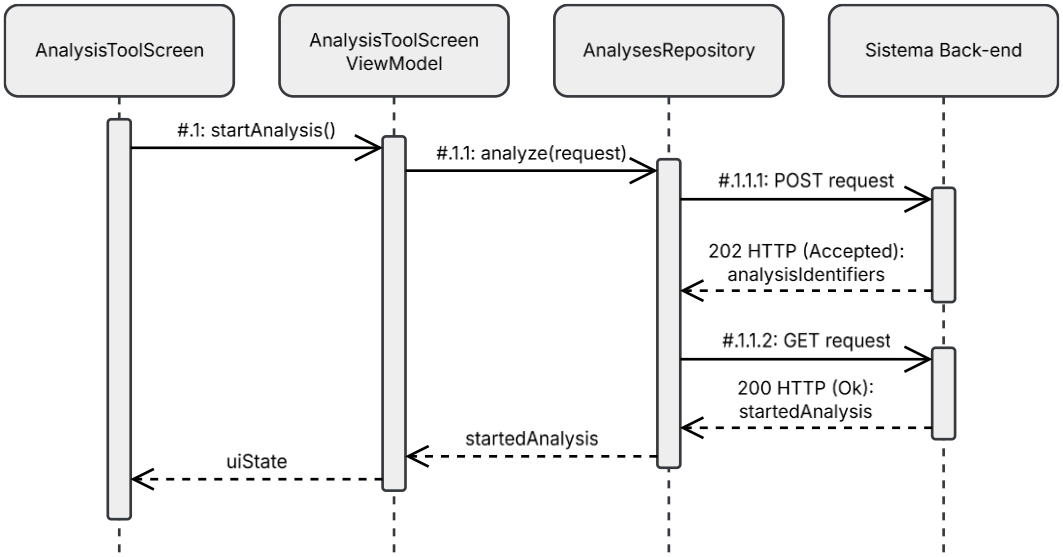
Diagrama de secuencia del análisis de correos electrónicos en el front end



Fuente. Autoría propia.

Figura 31.

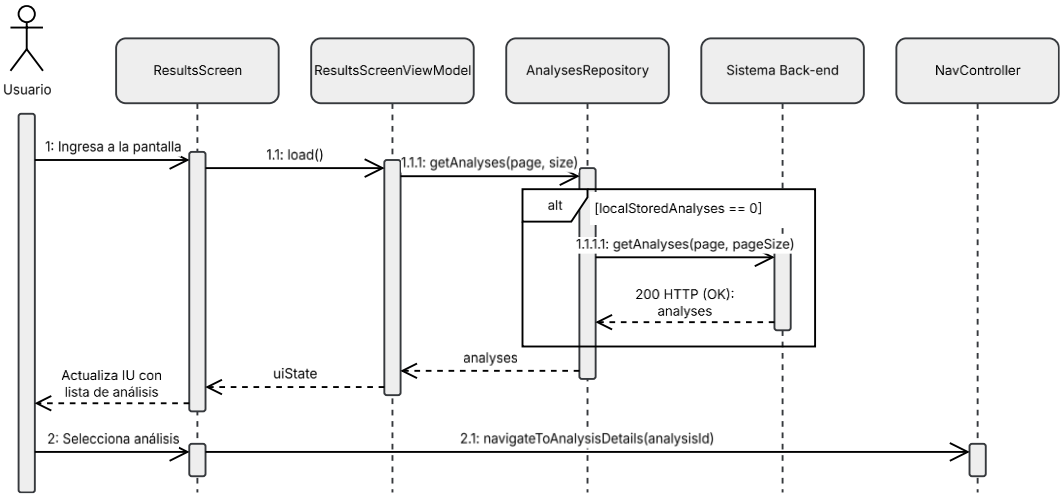
Diagrama de secuencia compartido del inicio de análisis en el front end



Fuente. Autoría propia.

Figura 32.

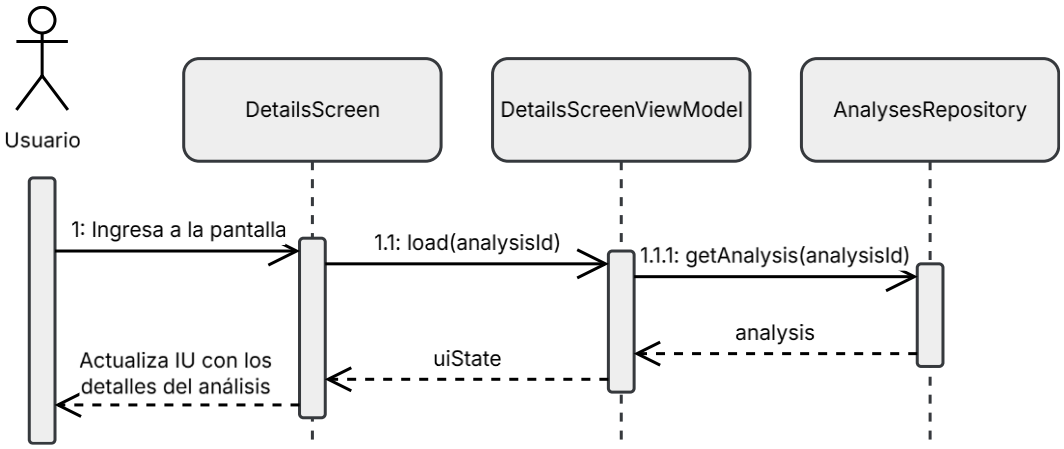
Diagrama de secuencia de la obtención de resultados de análisis en el front end



Fuente. Autoría propia.

Figura 33.

Diagrama de secuencia de la visualización de los detalles de análisis



Fuente. Autoría propia.

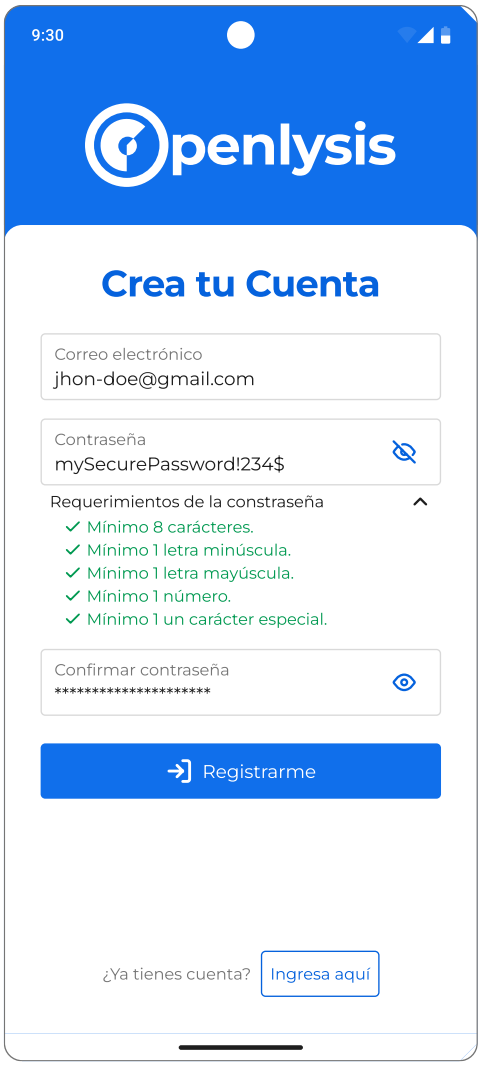
Interfaz Gráfica del Usuario

La característica fundamental del sistema front-end es la interfaz de usuario, la cual no solo incorpora los elementos gráficos, sino que también va más allá e incluye temas de accesibilidad, navegación y otros factores que influyen directamente a la experiencia de usuario. Asimismo, esta una parte esencial para cumplir con los objetivos del proyecto, ya que, a su vez, este será el método principal por el cual los usuarios podrán comprender los niveles de riesgo asociados a un mensaje, archivo o URL.

En concordancia a lo anterior, en esta sección se muestran los principales diseños que son base fundamental para el posterior desarrollo del sistema front-end, los cuales abarcan el registro e inicio de sesión de los usuarios, las herramientas de análisis, la lista de resultados, detalles de los análisis y una pantalla básica de configuración. En el Apéndice C se adjunta el enlace que permite acceder a los diseños y visualizarlos de una forma más interactiva.

Figura 34.

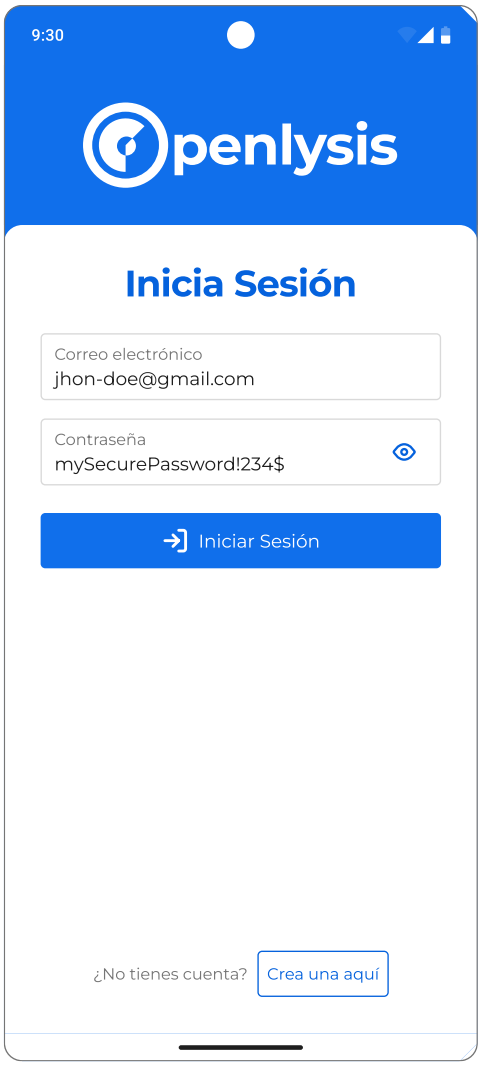
Diseño de la interfaz de registro



Fuente. Autoría propia.

Figura 35.

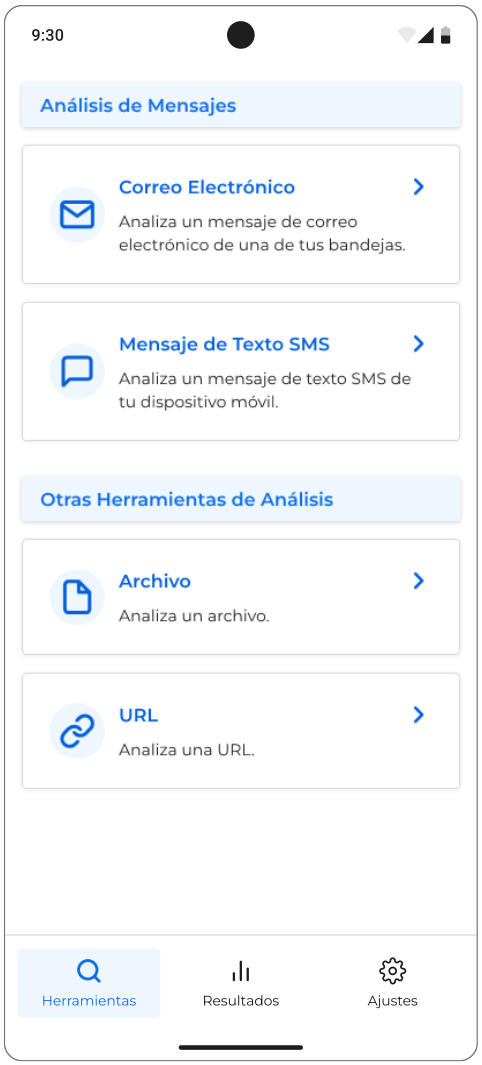
Diseño de la interfaz de inicio de sesión



Fuente. Autoría propia.

Figura 36.

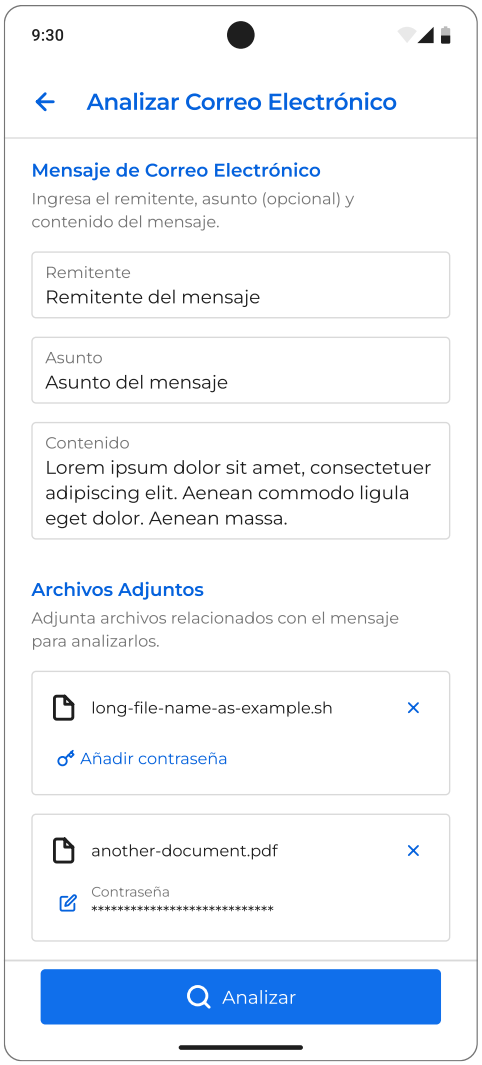
Diseño de la interfaz de las herramientas de análisis



Fuente. Autoría propia.

Figura 37.

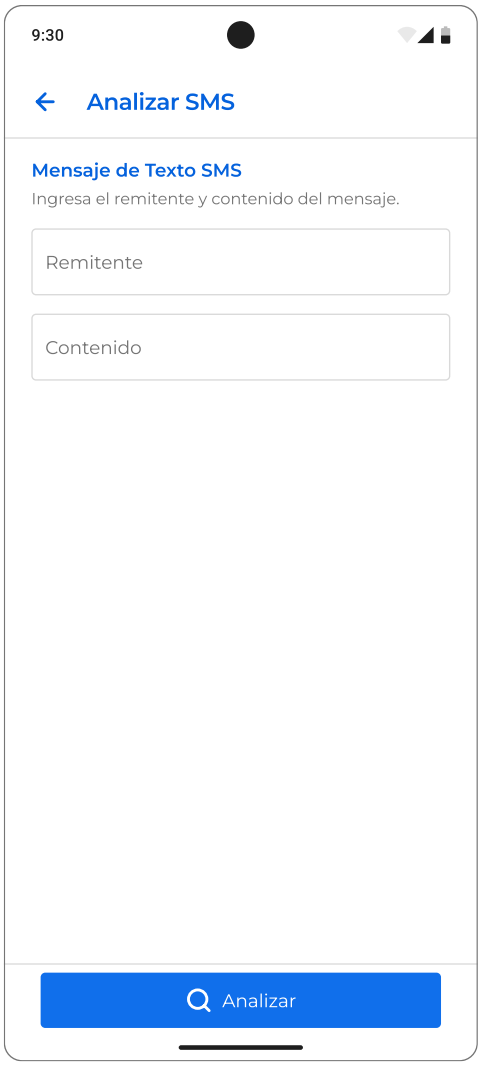
Diseño de la interfaz para analizar correos electrónicos



Fuente. Autoría propia.

Figura 38.

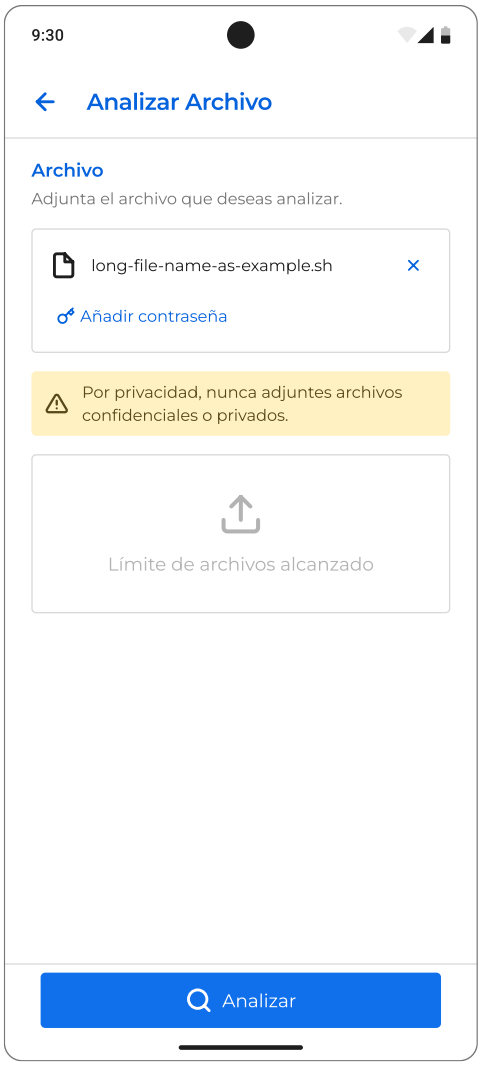
Diseño de la interfaz para analizar mensajes de texto SMS



Fuente. Autoría propia.

Figura 39.

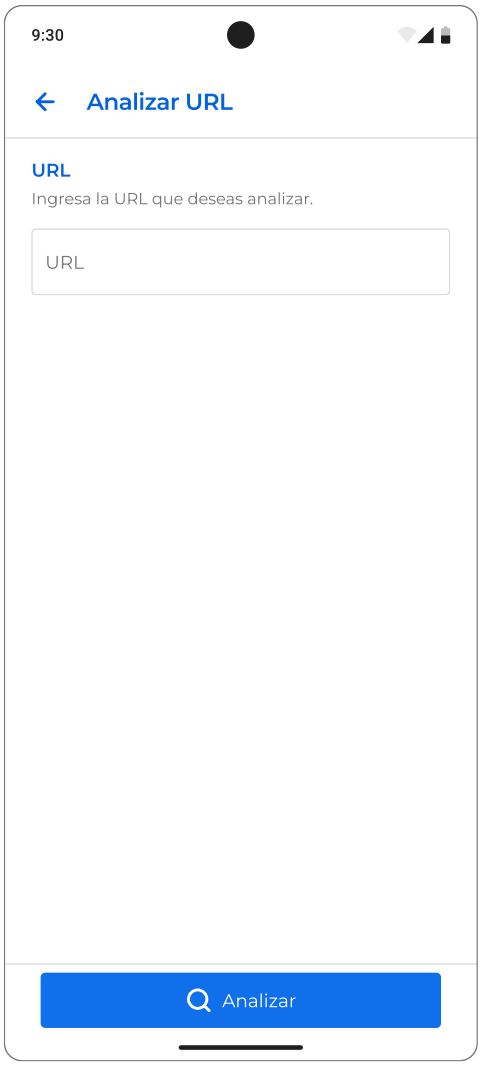
Diseño de la interfaz para analizar archivos



Fuente. Autoría propia.

Figura 40.

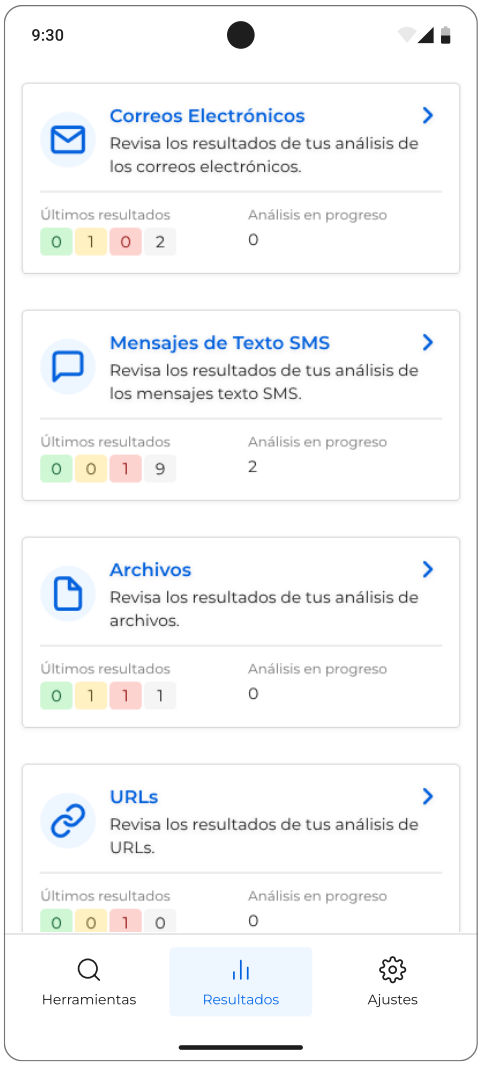
Diseño de la interfaz para analizar URLs



Fuente. Autoría propia.

Figura 41.

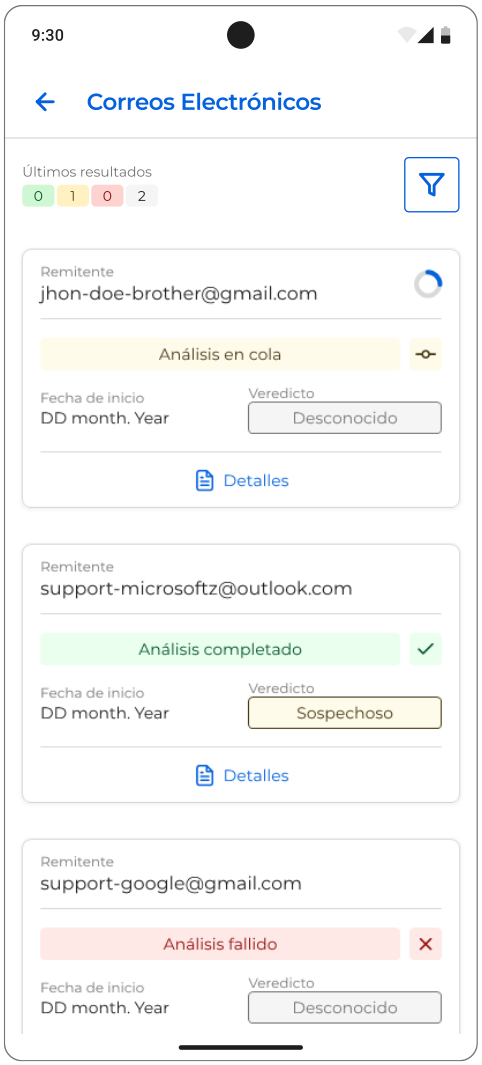
Diseño de la interfaz de la pantalla inicial de resultados de análisis



Fuente. Autoría propia.

Figura 42.

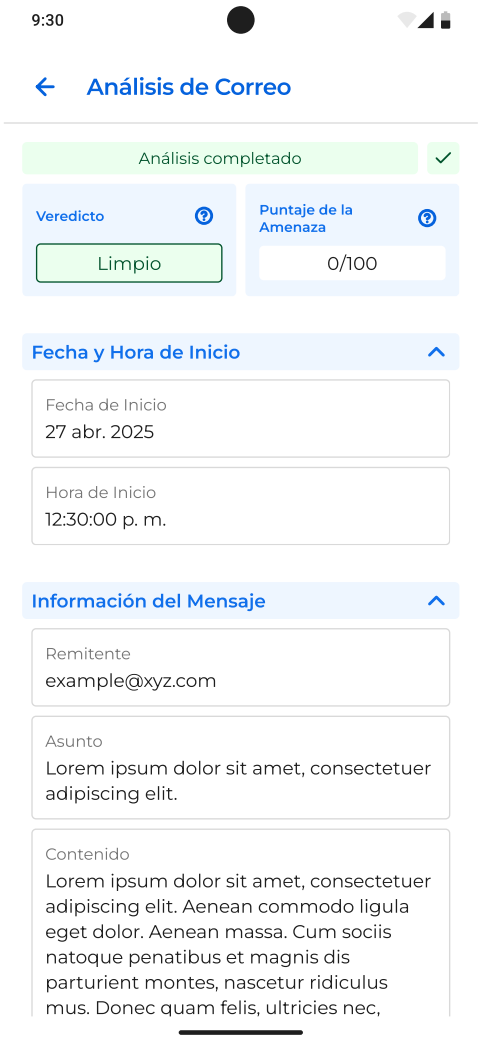
Diseño de la interfaz de la lista de resultados de análisis



Fuente. Autoría propia.

Figura 43.

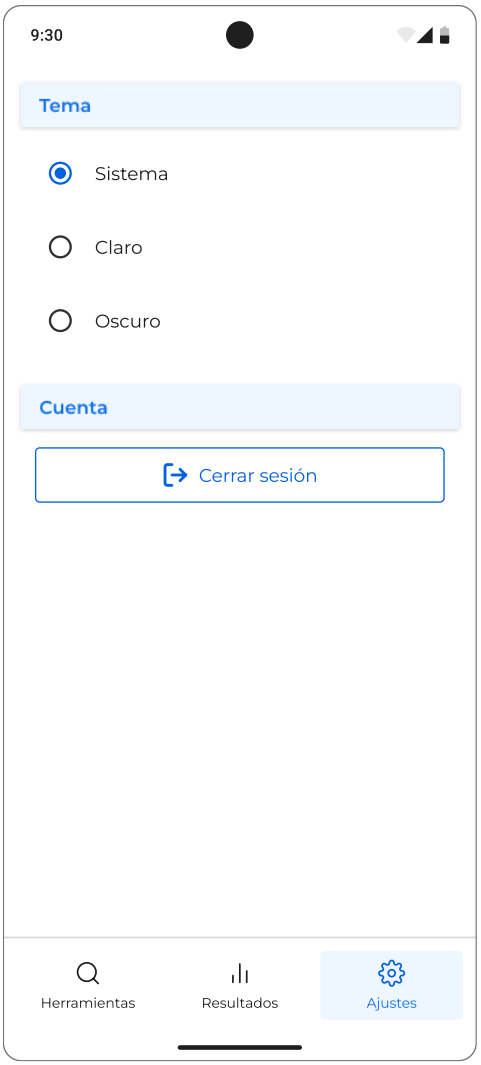
Diseño de la interfaz de los detalles de un resultado de análisis



Fuente. Autoría propia.

Figura 44.

Diseño de la interfaz de la pantalla de configuración



Fuente. Autoría propia.

Decisiones de Diseño

Tecnologías Back End

.NET. Para el desarrollo del servicio de análisis, la API principal y la API de autenticación se emplea el *framework* .NET a través del lenguaje de programación C#. Su uso se debe principalmente a los conocimientos previos del autor del proyecto, aun así, se debe destacar alguna de sus ventajas sobre otros *frameworks* como NodeJS, el cual se caracteriza por ser mono hilo por defecto (Sufiyan, 2025), que, en comparación, .NET es capaz de aprovechar toda la potencia del procesador. En cuanto a Python, con Django, Flask o FastAPI, son buenas alternativas, sin embargo, al ser un lenguaje de programación interpretado, su rendimiento es inferior a las APIs implementadas con .NET (Moore, 2024). De igual manera, existe Spring Boot, que es una opción muy robusta y bastante similar, pero el ecosistema principalmente favorece a desarrolladores familiarizados con Java.

PostgreSQL. Para la base de datos se emplea un modelo relacional, ya que permite estandarizar los modelos empleados a través del sistema y asegurar la consistencia que se requiere para proveer los resultados de los servicios a través de la API, y, representarlos eficazmente en la interfaz del usuario del sistema front-end. Por otro lado, se utiliza PostgreSQL ya que ofrece características únicas para extender sus funcionalidades, soporta los UUID de forma nativa y es de código abierto (PostgreSQL, s.f.), lo que permite que sea más barato adquirir instancias o servicios para desplegar la base de datos.

RabbitMQ. Este intermediario de mensajería (*message broker*) es una opción tradicional y ofrece un funcionamiento altamente robusto, el cual persiste los mensajes, y ayuda a que las peticiones de los usuarios tengan altas probabilidades de completarse, incluso ante fallas críticas de los servicios intercomunicados, esto se complementa con la librería *MassTransit* en .NET, que facilita la integración de intermediarios de mensajería y ofrece una capa adicional de resiliencia (Patterson, s.f.). Entre otras alternativas esta NATS, un sistema ligero y con mayor rendimiento construido con Go, no obstante, carece de una librería incorporada con *MassTransit*, lo que aumentaría el tiempo de desarrollo.

Tecnologías Front End

Jetpack Compose. Este *framework* se usa para desarrollar el sistema front-end con Kotlin, ya que es la opción más recomendada por Android para crear aplicaciones modernas, el tiempo de desarrollo es menor en comparación al uso de plantillas en XML y uso de vistas, a su vez, utiliza una API declarativa que facilita la legibilidad del código (Google, 2024). También se encuentran otras alternativas como React Native, que, a pesar de ofrecer un gran rendimiento, no es completamente nativa y su utilización se beneficia principalmente para desarrollar aplicaciones multiplataforma.

Alojamiento

Aunque el alcance de este proyecto culmina en el desarrollo de un prototipo de alta fidelidad y funcionalidad, en esta sección se describen los servicios pensados para desplegar el sistema Openlysis a un entorno de producción.

Supabase. Este servicio permite el alojamiento de bases de datos de PostgreSQL ofreciendo varias capas de precio e incluso un nivel gratuito que beneficia la etapa de desarrollo. Adicionalmente, ofrece otros servicios que pueden fortalecer la integración del sistema, en especial los mecanismos de autenticación.

Google Cloud Storage. El almacenamiento de archivos temporales esta soportado por este servicio de Google, que ofrece precios muy generosos e incluso una capa gratuita que es fundamental en la etapa de desarrollo.

Google Cloud Run. Este servicio se emplea para alojar la API, la API de autenticación y el servicio interno de para delegar los análisis, su uso se debe únicamente para integrarse con mayor facilidad al almacenamiento de archivos temporales de *Google Cloud Storage*.

CloudAMQP. Para alojar la instancia de RabbitMQ requerida para comunicar el sistema back-end, se utiliza CloudAMQP, que al igual que otros servicios, ofrece varios planes de precio que facilitan la escalabilidad.

Google Play Store. Aunque no es un servicio de alojamiento como tal, es fundamental para que los usuarios puedan descargar la aplicación de Android de forma segura y de igual manera, se puedan lanzar actualizaciones regularmente.

Seguridad y Privacidad

Teniendo en cuenta la importancia que tiene la privacidad de los usuarios, destacada en el marco jurídico del documento, a continuación, se exponen los aspectos esenciales que permiten al sistema en general, adherirse a las leyes y normativas. Como complemento, en el Apéndice D se adjunta un borrador de políticas de privacidad que podría ser utilizado en un escenario de producción.

Gestión de los Mensajes SMS y Correos Electrónicos

Emplear servicios de análisis externos implica ser conscientes de sus términos y condiciones de uso, que pueden ser bastante permisivos, por lo tanto, para asegurar que los datos de los usuarios, especialmente cuando se analizan mensajes, sean divulgados en su menor posibilidad, el sistema se diseña para no emplear toda la información, sino que solamente extraer las URLs y archivos adjuntos, minimizando la exposición de datos privados. Cabe recalcar que los datos de los mensajes aun así son almacenados en la base de datos, pero su propósito está estrictamente limitado a proveer al usuario un historial de sus propios análisis.

Consentimiento Informado y Control del Usuario

Para dar mayor libertad al usuario al analizar información, no se diseñan mecanismos de análisis completamente automatizados, lo que permite al usuario decidir qué y cuando analizar, dando un rol de mediador a la aplicación de Android.

Alertas Proactivas al Usuario

Los diseños de la interfaz incluyen advertencias para incitar a los usuarios de no incluir información confidencial o privada, en especial, cuando se analizan archivos, que pueden contener información altamente relevante.

Seguridad en Tránsito (HTTPS)

Para una implementación en producción, el tráfico de datos se debe realizar mediante una comunicación encriptada entre el sistema front-end y back-end, utilizando el protocolo HTTPS, que protege la información de posibles ataques o robos de datos.

Desarrollo

Definido el diseño del sistema en general, se procede al desarrollo del back end y front end, que incluye la utilización de herramientas de control de versiones, la creación inicial de un *product backlog* que evoluciona durante cada *sprint*, los hitos más destacables durante la codificación y los resultados principales.

El desarrollo se divide en dos proyectos, uno enfocado al sistema back-end y el otro al sistema front-end, con el fin de no mezclar código y mejorar la legibilidad, mantenibilidad y comprensión global durante toda la etapa. El código fuente de los proyectos se puede encontrar en los Apéndice F y

Apéndice G respectivamente.

Herramientas de Desarrollo

El desarrollo de sistemas funcionales no es una tarea que se base únicamente en el código implementado, sino que también, abarca otros factores fundamentales, que, para este caso, se destaca el uso de herramientas para controlar las versiones, y replicar el sistema en diversos entornos, que puede ser útil para otros desarrolladores, considerando que este proyecto está relacionado con el código abierto.

Git

Es un sistema que rastrea los cambios realizados a los distintos archivos de un proyecto a través de diferentes versiones, funcionando de forma local como una pequeña base de datos que difícilmente admite el borrado de información, lo cual ayuda a obtener y visualizar un historial claro y confiable de los cambios que se han realizado (Chacon & Straub, 2014). Esta herramienta es fundamental porque ayuda a implementar funcionalidades de análisis de forma incremental, experimentar sin afectar al sistema, resolver errores de manera aislada y tener un repositorio que detalle el paso a paso transparentemente.

GitHub

Entendiendo Git como una herramienta principalmente de uso a nivel local, GitHub en contraste, es un servicio en línea que permite alojar los repositorios de Git gratuitamente, además, ofrece funciones adicionales como la creación de pipelines de integración y despliegue continuo (GitHub, s.f.). Este servicio asegura que el código de los proyectos pueda ser visualizado y clonado desde cualquier dispositivo por cualquier persona.

Docker

Docker es un sistema que permite ejecutar aplicaciones en entornos aislados que son similares a máquinas virtuales, pero, más livianas y eficientes, llamados contenedores, cuya característica principal, es que integran todo lo necesario para que una aplicación funcione correctamente, sin depender de la maquina física (Docker, s.f.).

Esta herramienta se usa principalmente para el sistema back-end, ya que ayuda a integrar la base de datos, el intermediario de mensajería, el orquestador de análisis, la API de autenticación y la API principal, en un sistema organizado que se inicia de forma conjunta, sin necesidad de que el desarrollador deba instalar un editor de código, el servidor de la base de datos y otros instrumentos esenciales para poner en marcha al sistema back-end. Para lograr esto, se emplean dos funcionalidades clave que ofrece Docker:

Dockerfiles. La creación de los contenedores se lleva a cabo a través de las denominadas imágenes, que son inmutables y definen las dependencias que necesita una determinada aplicación para funcionar, además del código fuente. Estas imágenes se construyen utilizando *Dockerfiles*, que son archivos de texto que especifican instrucciones para construir el código fuente, descargar e instalar los paquetes de dependencias y definir el punto de entrada del futuro contenedor (Docker, s.f.).

Docker Compose. Es una herramienta que facilita la creación de múltiples contenedores utilizando un solo archivo en formato YAML, en el que se definen los servicios que estarán en funcionamiento al mismo tiempo, así mismo, se puede establecer dependencias entre estos servicios, por ejemplo, la API de análisis depende de la base de datos, por lo cual es fundamental que la API se inicie justo después (Docker, s.f.).

Sistema Back-end

El desarrollo de este sistema comprende varios elementos esenciales que se materializan en la API principal, que es el punto de entrada de las peticiones de análisis de los usuarios, aunque es difícil demostrar su desarrollo mediante gráficos, al final de esta sección, se muestra la documentación visual que expone el resultado obtenido.

Sprints

El flujo de los *sprints* sigue un orden lógico, priorizando primero la implementación de funcionalidades pequeñas. Posteriormente, se abordan las más complejas, que pueden aprovechar servicios ya desarrollados. Por ejemplo, el análisis de correos electrónicos se implementa después de haber completado el análisis de URLs, funcionalidad construida en los primeros *sprints*.

Además de estos *sprints* principales, también están otros que, en lugar de enfocarse en características específicas, están orientados a la resolución de errores, incorporación de mejoras, u otras situaciones que lo ameriten, sin embargo, debido a su imprevisión, no se destacan a continuación.

Tabla 10.

Sprints principales durante el desarrollo del sistema back-end

Sprint	Objetivo
No 1	Implementación del análisis de archivos.
No 2	Implementación del análisis de URLs.
No 3	Implementación del análisis de correos electrónicos y mensajes de texto SMS.
No 4	Implementación de API básica de autenticación.
No 5	Mejora de la recepción, carga y descarga de archivos utilizando Google Cloud Storage.

Fuente. Autoría propia.

Estructura de Módulos y Dependencias

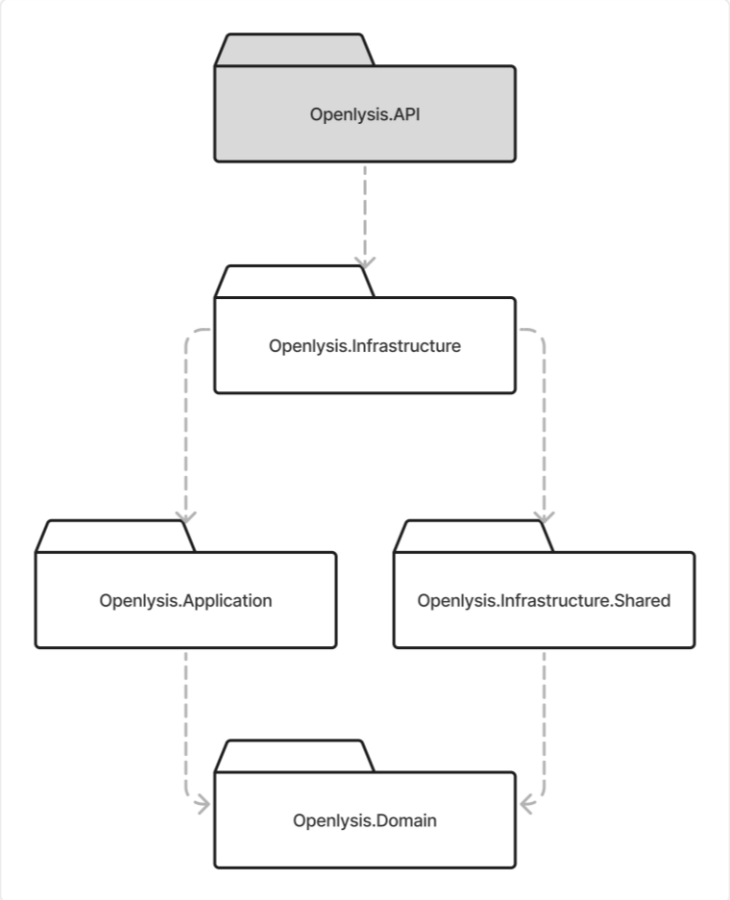
El código del sistema back-end contiene 13 módulos focalizados a la API principal, la API de autenticación y el orquestador de análisis. Para su representación, se diseñan tres diagramas en los cuales se muestran las dependencias entre estos módulos.

La Figura 45 muestra los módulos que soportan el funcionamiento de la API principal, los cuales reflejan la aplicación de la arquitectura limpia (*Clean Architecture*), que asegura bajo acoplamiento y alta cohesión, no solo en la API, sino que a nivel general en el sistema back-end. Esta figura se divide en un módulo que encapsula los modelos de dominio (Openlysis.Domain),

otro, que, adherido a los casos de uso, ofrece las funcionalidades de análisis (Openlysis.Application), la capa que expone dichas funciones a través de *endpoints* (Openlysis.API), y finalmente, el módulo responsabilizado de implementar los servicios relacionados a la infraestructura, por ejemplo, el receptor y emisor que usa el intermediario de mensajería (Openlysis.Infrastructure). Adicionalmente, existe un módulo que incluye utilidades compartidas en toda la infraestructura (Openlysis.Infrasturcture.Shared).

Figura 45.

Módulos y dependencias de la API principal del sistema back-end

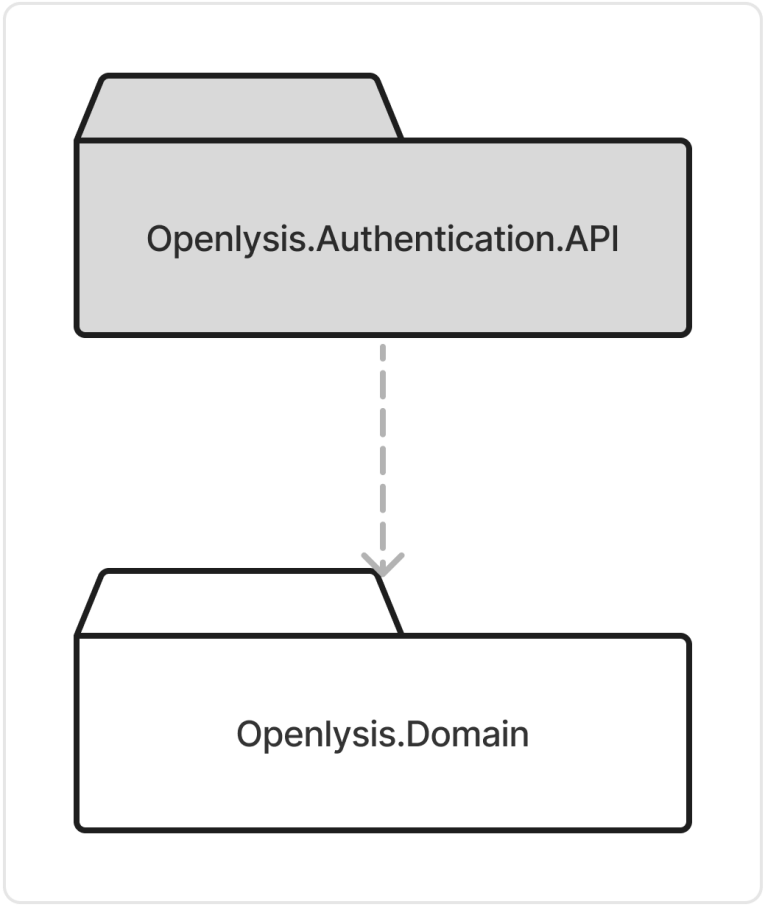


Fuente. Autoría propia.

La Figura 46, que representa a la API de autenticación, es mucho más simple y en esta se destaca la reutilización del módulo que contiene los modelos de dominio.

Figura 46.

Módulos y dependencias de la API de autenticación del sistema back-end

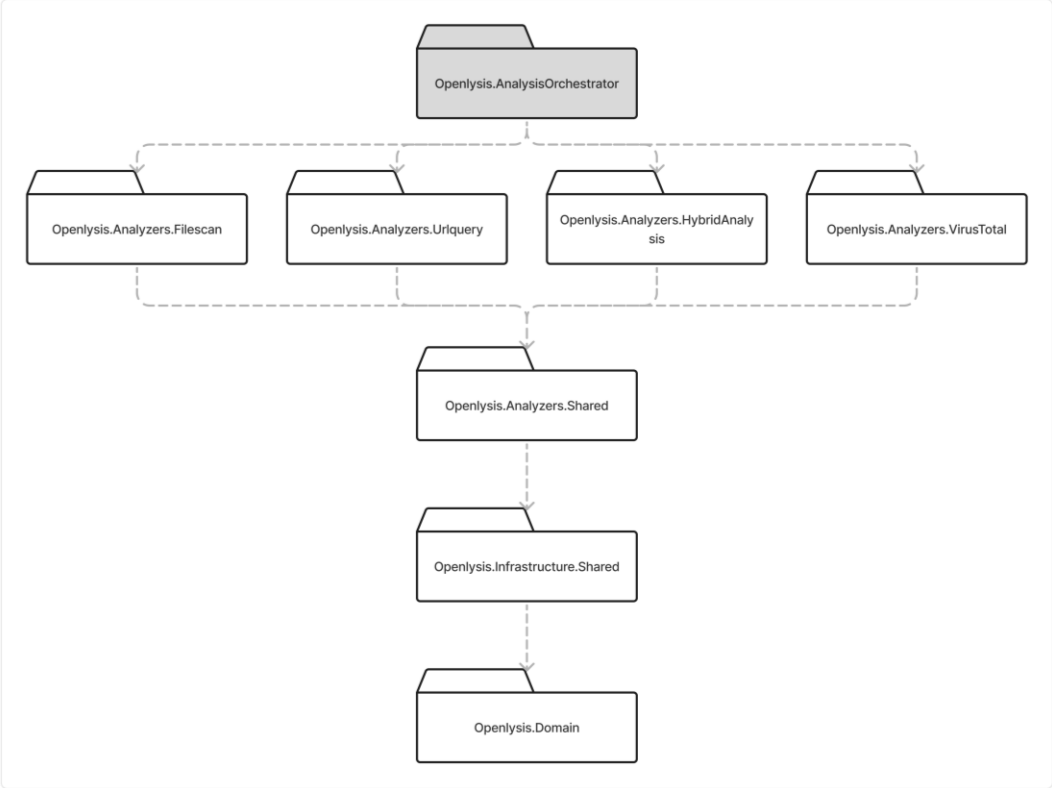


Fuente. Autoría propia.

Por último, la Figura 47, ilustra el diseño modular del orquestador de análisis. Este incorpora varias librerías, cuyas responsabilidades, son acoplar un determinado servicio de análisis, sin afectar al resto del sistema y cumpliendo con las reglas definidas en un módulo compartido y enfocado a los analizadores (Openlysis.Analyzers.Shared). Al igual que en la API principal, el orquestador depende de la librería compartida en la infraestructura (Openlysis.Infrastucture.Shared) y del módulo de los modelos de dominio (Openlysis.Domain).

Figura 47.

Módulos y dependencias del orquestador de análisis del sistema back-end



Fuente. Autoría propia.

Hitos de Implementación

Los hitos de implementación representan aquellos procesos que tuvieron desafíos durante el desarrollo. Aunque el sistema back-end en sí, no fue fácil de codificar, se pueden destacar dos resultados que fueron cruciales para obtener un buen rendimiento y asegurar la fiabilidad del sistema.

En primer lugar, se debe resaltar la gestión de los archivos entrantes, ya sea que estuviesen asociados a la petición para analizar un correo electrónico. La manipulación adecuada de estos datos es esencial, especialmente cuando tienen un tamaño considerable, ya que, si se emplea la memoria del servidor para almacenarlos, incluso temporalmente, sería acaparar

recursos que el mismo sistema back-end usa para procesar las peticiones, y en el peor de los casos, podría ocasionar fallas graves por la baja o nula disponibilidad de memoria. Esta preocupación dio lugar a la implementación de un mecanismo para retransmitir, en pequeños lotes de bytes, el contenido de los archivos entrantes, directamente a un espacio dedicado, que fuese capaz de almacenar grandes volúmenes de datos. La implementación en concreto se adapta de acuerdo con el contexto, en entornos de desarrollo, se maneja el almacenamiento local, y en producción, se usaría el servicio Google Cloud Storage, como se ilustra en la sección de diseño.

En segundo lugar, está el inicio y la actualización de los análisis de manera robusta, fundamentalmente en el orquestador de análisis. Si bien, la lógica es simple tanto para utilizar los múltiples servicios externos, como para obtener resultados actualizados, también es crucial asegurar la recuperación ante errores, garantizar la finalización y reflejar datos en los análisis de manera consistente, independientemente de los resultados finales. Para solventar esto, a través de la librería MassTransit, se desarrollaron procesos capaces de persistir y guardar el progreso durante toda la ejecución, de tal manera, que, si ocurre una falla, el proceso se reinicia y no pierde el progreso, evitando iniciar nuevamente análisis redundantes, y en su lugar, seguir con el proceso de actualización hasta la finalización, y, en caso de alcanzar un límite determinado de reinicios, el análisis se da por fallido, previniendo así estados inconsistentes. Adicionalmente, se implementan tiempos de espera que una vez caducados, dan por finalizados o caducados los análisis.

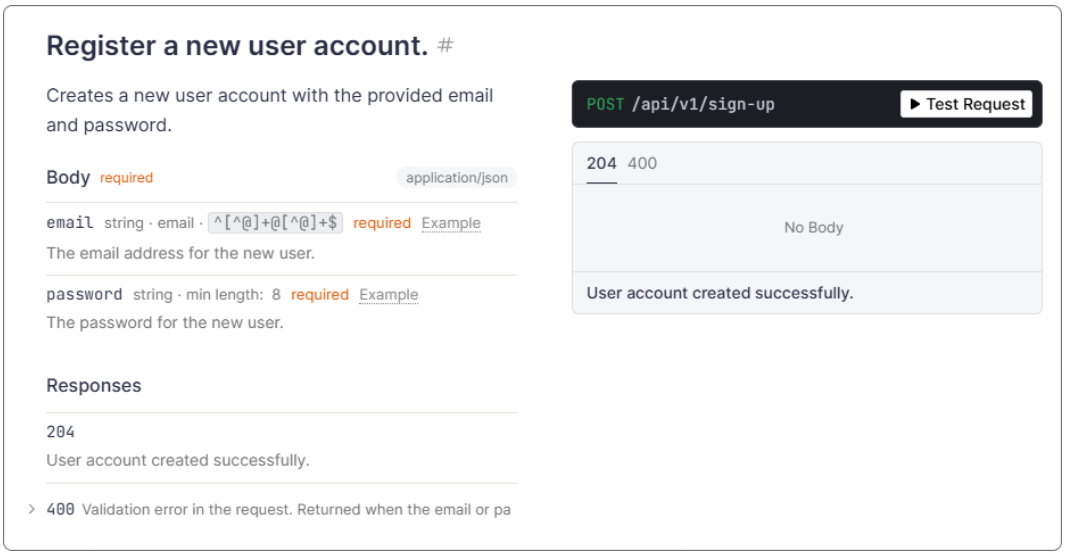
Interfaz Gráfica de la Documentación de las APIs

Las APIs del sistema back-end, tanto la principal como la de autenticación, son un buen punto de referencia para demostrar los resultados obtenidos durante el desarrollo, a pesar de no

ser el punto directo de interacción con el usuario, estas APIs ofrece documentación a partir de una interfaz gráfica que da cuenta de los *endpoints* implementados.

Figura 48.

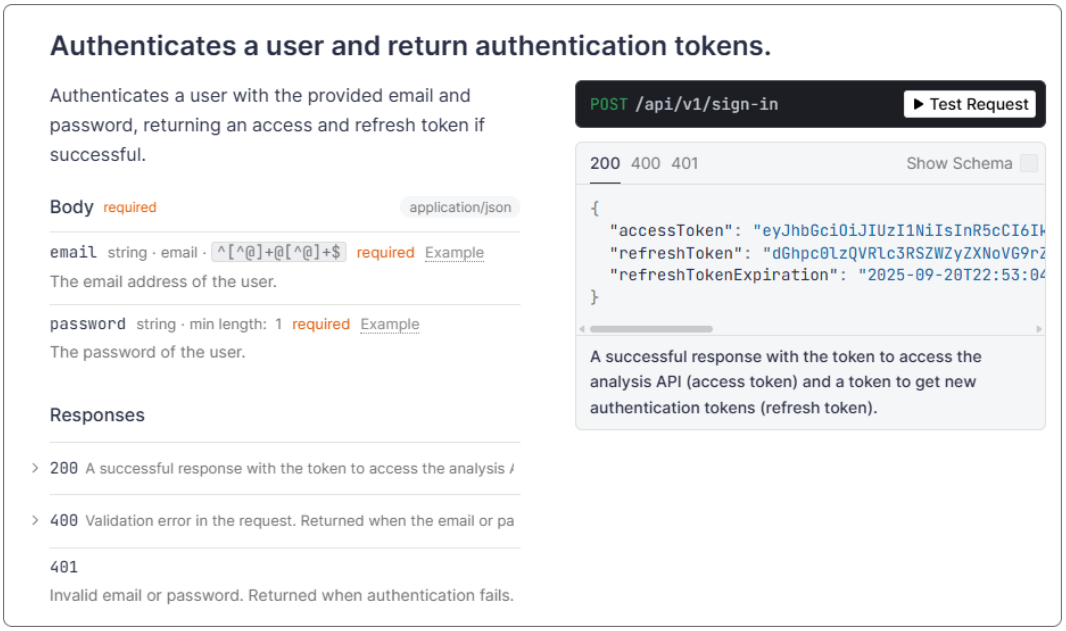
Interfaz gráfica del endpoint para registrar usuarios



Fuente. Autoría propia.

Figura 49.

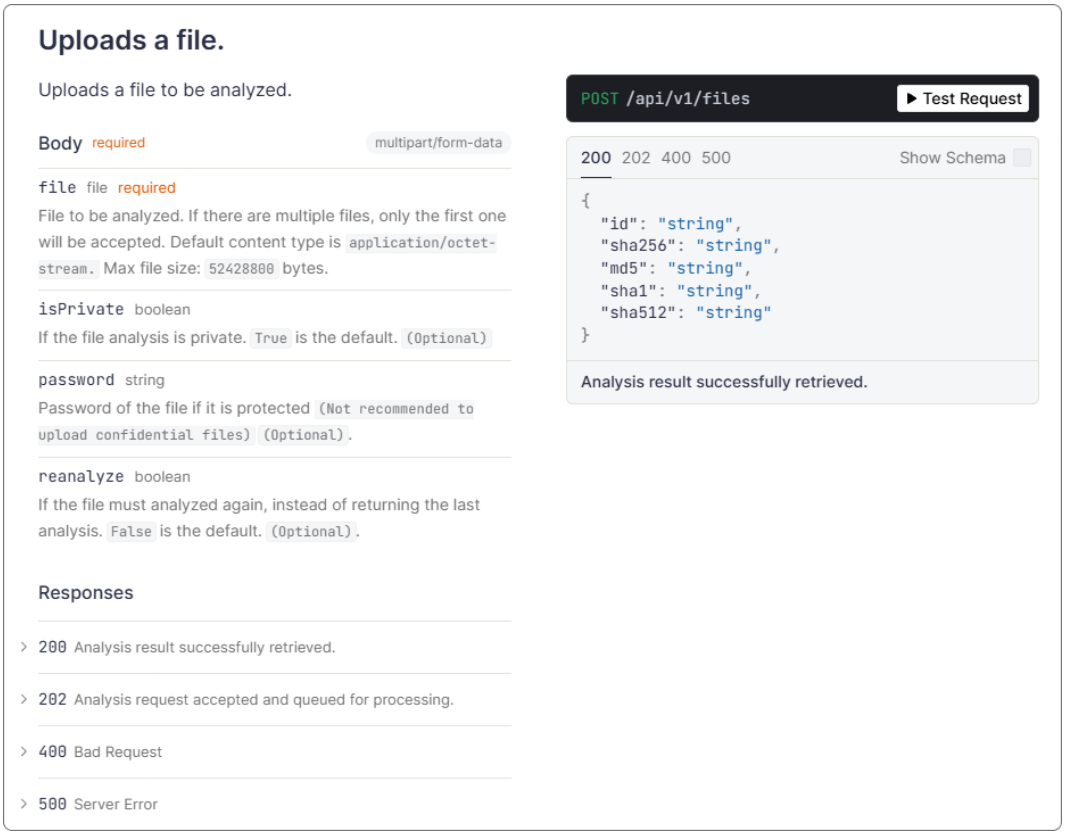
Interfaz gráfica del endpoint para iniciar sesión



Fuente. Autoría propia.

Figura 50.

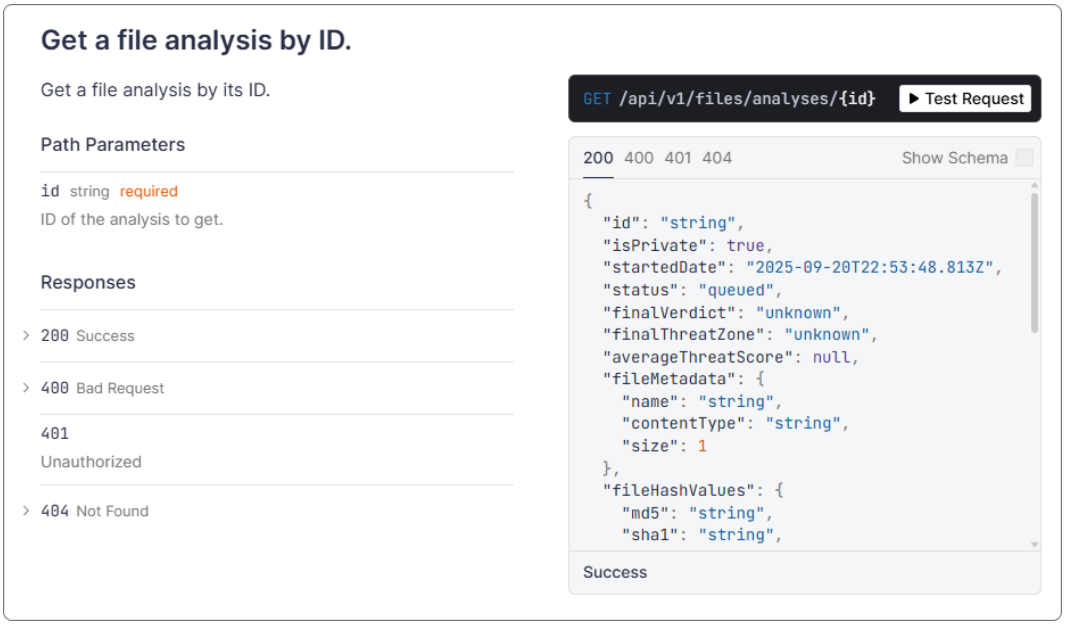
Interfaz gráfica del endpoint para analizar archivos



Fuente. Autoría propia.

Figura 51.

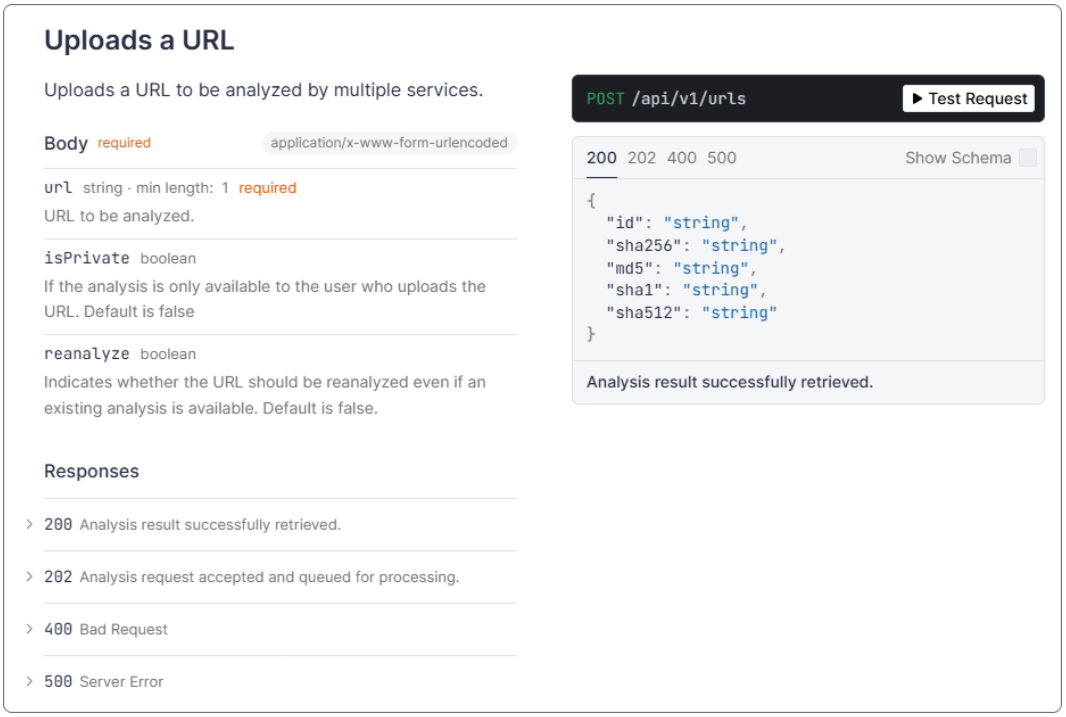
Interfaz gráfica del endpoint para obtener análisis de archivos



Fuente. Autoría propia.

Figura 52.

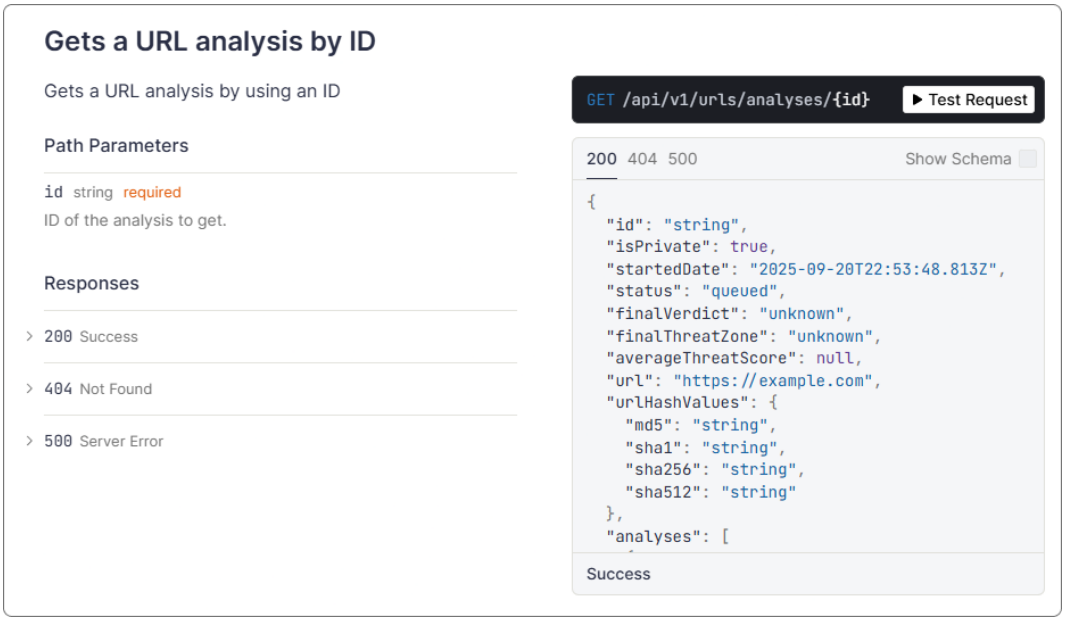
Interfaz gráfica del endpoint para analizar URLs



Fuente. Autoría propia.

Figura 53.

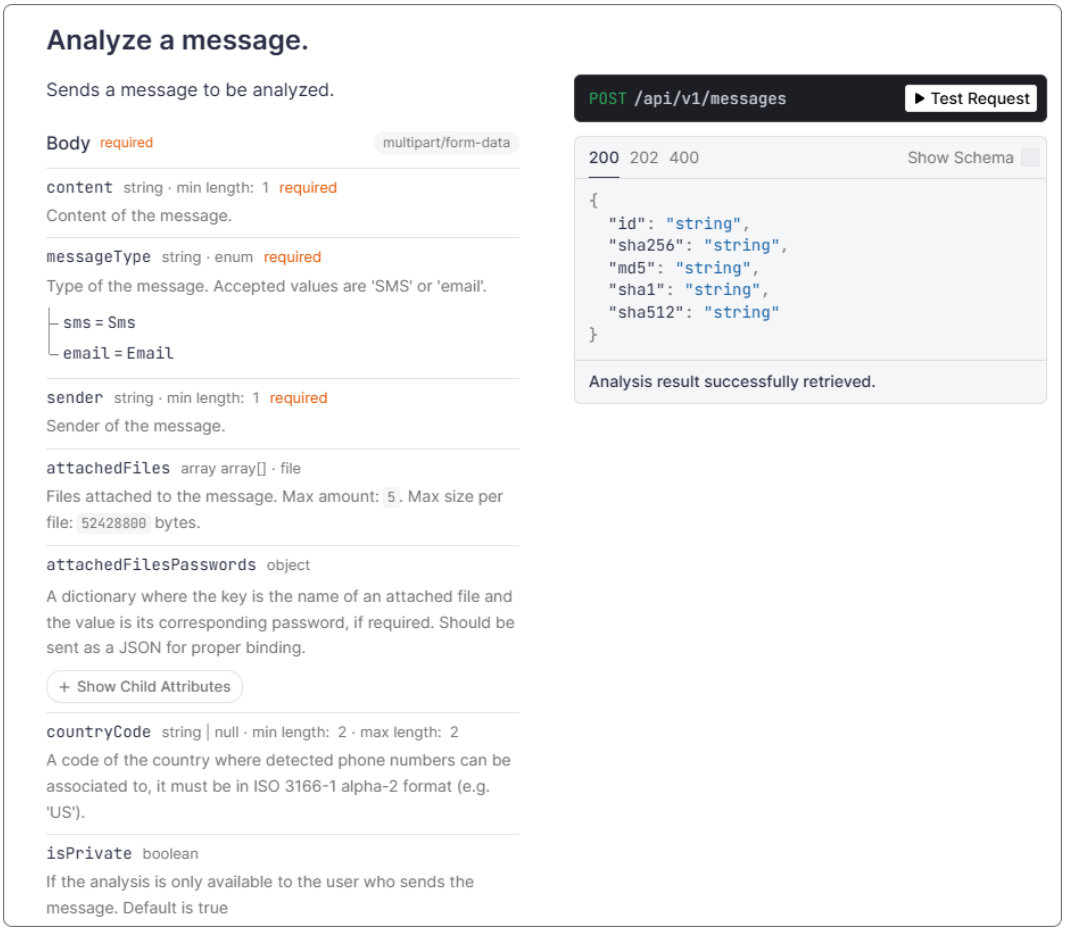
Interfaz gráfica del endpoint para obtener análisis de URLs



Fuente. Autoría propia.

Figura 54.

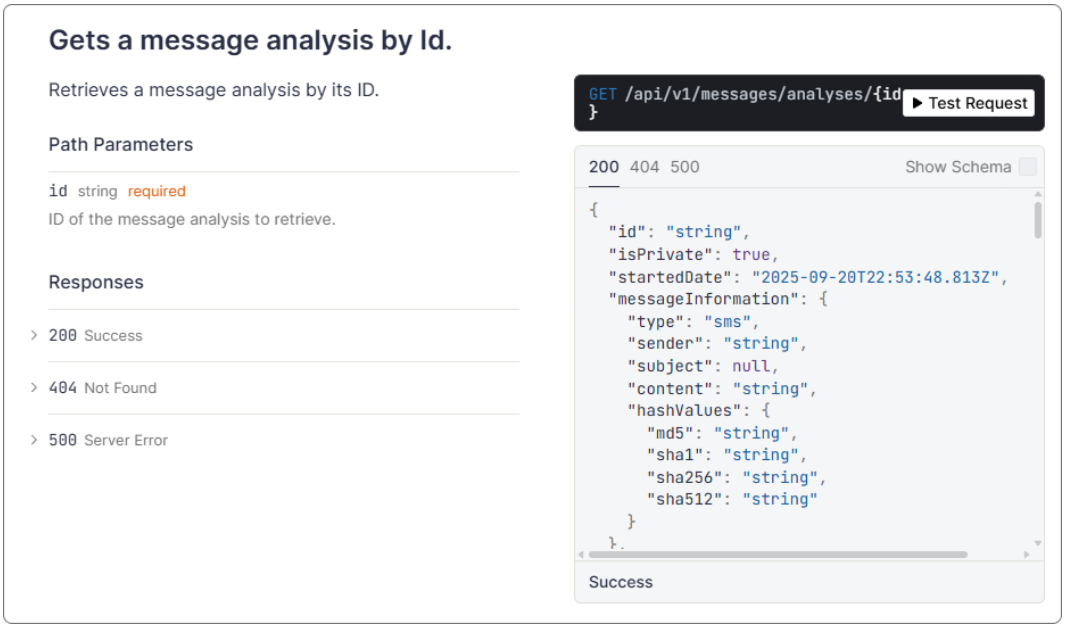
Interfaz gráfica del endpoint para analizar mensajes



Fuente. Autoría propia.

Figura 55.

Interfaz gráfica del endpoint para obtener análisis de mensajes



Fuente. Autoría propia.

Sistema Front-end

Sprints

Respecto a los *sprints* principales del sistema front-end, que se orientan al desarrollo de la interfaz gráfica del usuario, se destacan seis, que además de implementar los apartados visuales, también incluyen la lógica necesaria para ofrecer las funcionalidades finales al usuario.

Tabla 11.

Sprints principales durante el desarrollo del sistema front-end

Sprint	Objetivo
No 1	Implementar modulo del sistema de diseño de la aplicación.
No 2	Implementar pantallas de herramientas de análisis.
No 3	Implementar pantallas para mostrar los resultados de los análisis.
No 4	Implementar pantallas de registro e inicio de sesión de los usuarios.
No 5	Implementar análisis automatizado de mensajes de texto SMS.
No 6	Implementar pantallas para pedir permisos al usuario para detectar mensajes de texto SMS y enviar notificaciones.

Fuente. Autoría propia.

Estructura de Módulos y Dependencias

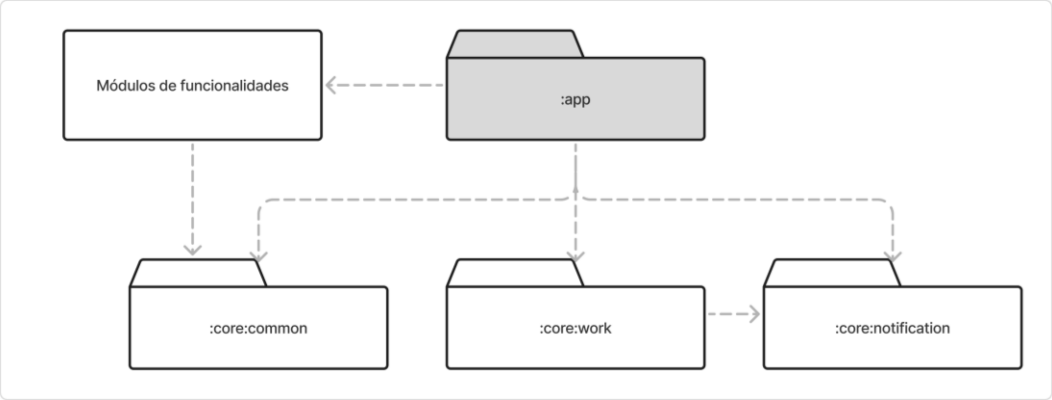
La aplicación de Android está dividida en 18 módulos que se basan en la arquitectura expuesta en la sección de diseño. Para su demostración se emplean dos diagramas, el primero ilustrado a nivel general, y el segundo, en más detalle, pero enfocado a las pantallas de la interfaz gráfica del usuario.

La Figura 56 representa cuatro módulos: el punto de entrada de la aplicación de Android (app), una librería con utilidades comunes reusadas en varios módulos del sistema entero (core:common), el módulo que proporciona acceso las notificaciones (core:notification) y el último, que provee servicios para ejecutar trabajos en segundo plano y analizar los mensajes de texto SMS (core:work).

El módulo *app*, también se encarga de referenciar a todos los demás, ejerciendo como un ensamblador, asegurándose de que todas las implementaciones para cada funcionalidad se incluyan en la construcción final de la aplicación. Sin embargo, estas relaciones no se ilustran de manera completa, debido a que reduciría la legibilidad del diagrama.

Figura 56.

Módulos y dependencias a nivel general del sistema front-end



Fuente. Autoría propia.

La Figura 57 expone a detalle los módulos que esencialmente soportan al sistema front-end, en esta se visualiza como se implementa la arquitectura que propone la capa de presentación y la capa de datos. Primeramente, en la capa de presentación, está el módulo denominado sistema de diseño (core:designsystem), que integra los colores, fuentes y bloques que permiten construir la interfaz gráfica del usuario de forma estandarizada e impregnando una identidad. También, están los módulos para cada característica, que tienen un prefijo llamado *feature* y agrupan las pantallas de acuerdo con su categoría. Posteriormente, están los módulos, que naturalmente, hacen parte de la capa de datos, su prefijo es *data*, componen al núcleo de la aplicación y son utilizados en la capa de presentación. Para visualizar a mayor detalle el diagrama, se puede utilizar el enlace que se encuentra en el Apéndice E.

Figura 57.

Módulos y dependencias de las funcionalidades del sistema front-end

Fuente. Autoría propia.

Hitos de Implementación

Los diseños de la interfaz gráfica del usuario fueron una base fundamental, porque facilitaron el desarrollo de las pantallas de la capa de presentación, aun así, la lógica de las funcionalidades se convirtieron un reto, especialmente, en la implementación de análisis automatizado de mensajes de texto SMS y la actualización tiempo real de los análisis.

El análisis automatizado no pertenece explícitamente a los casos de uso, pero si, es una mejora considerable a la experiencia de usuario. Este desafío abarca la detección de los mensajes SMS entrantes en el dispositivo, la generación de alertas para que el usuario decida si desea analizarlos y la ejecución de procesos en segundo plano que muestran el progreso del análisis hasta su finalización, siempre respetando la privacidad y decisiones del usuario. Para implementar esta funcionalidad, se utilizaron *broadcasts*, que permiten escuchar eventos específicos y activar la aplicación. Posteriormente, se añadieron tareas en segundo plano, para iniciar y obtener actualizaciones del estado de los análisis, a través de los llamados *workers* que

son una característica nativa de Android. En cuanto a las alertas de la detección y el progreso, se emplearon notificaciones del sistema, a través de un servicio abstraído. Finalmente, se desarrollaron unas pantallas que solicitan de manera opcional los permisos necesarios para detectar los mensajes SMS y, en los dispositivos más modernos, el acceso para mostrar notificaciones. Con ello, se logró proporcionar esta funcionalidad, que, a pesar de ser adicional, aporta un valor sustancial para reducir el trabajo manual del usuario.

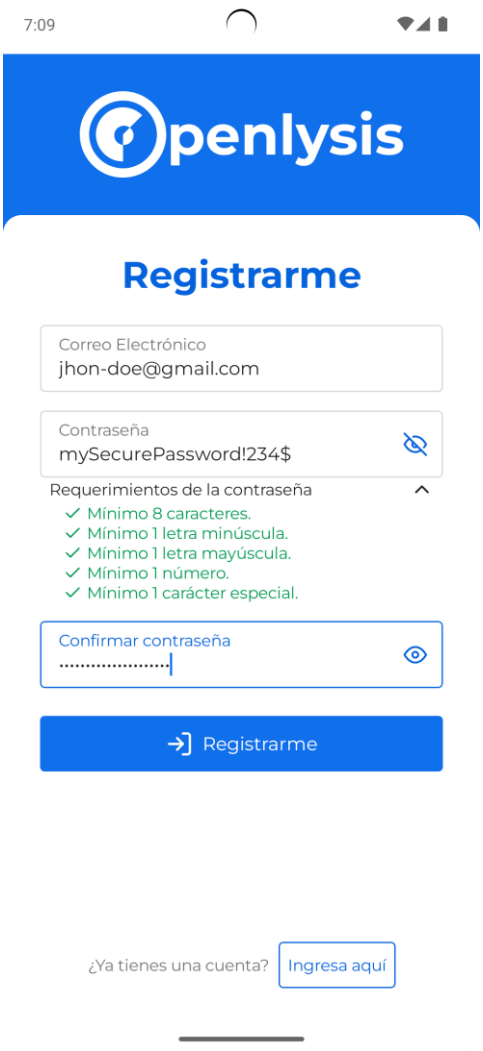
Las actualizaciones de los análisis, es otro factor crucial porque afecta directamente al rendimiento de la aplicación, además de estar caracterizada por ser en tiempo real, lo que supone un valor agregado a la experiencia del usuario y, otras funcionalidades recaen sobre esta, como la descrita anteriormente. En un principio, se implementó un proceso que en inglés se define como *polling*, el cual consiste en consultar constantemente un servicio para detectar nueva información, en este caso, enviar peticiones al sistema back-end para conocer el estado de los análisis. Aunque cumplía con su función, no era una comunicación en tiempo real y además era ineficiente. En consecuencia, se cambió la implementación y se utilizaron los WebSockets, que funcionan como una comunicación bidireccional entre un servidor y múltiples clientes. Esta nueva funcionalidad, incluía cambios en el back end, pero la mayor carga radicaba en el sistema front-end, ya que se debían implementar procesos nuevos de serialización que fueron simplificados en mayor parte, gracias al uso de SignalR, una librería que abstrae el uso de los WebSockets tanto en el sistema back-end como front-end. Este nuevo cambio, represento la reducción de consumo de recursos, añadió actualizaciones en tiempo real y, de manera indirecta, redujo la necesidad de que el usuario interactúe para refrescar los análisis, traduciéndose finalmente en una mejora a la experiencia de usuario.

Interfaz Gráfica del Usuario

Para demostrar los resultados, se utilizan las siguientes capturas, que exponen como los diseños se materializan después del desarrollo e incluso llegan a lucir casi idénticos. Para la demostración, se emplea un dispositivo Pixel 7 con Android 13 en el nivel de API 33.

Figura 58.

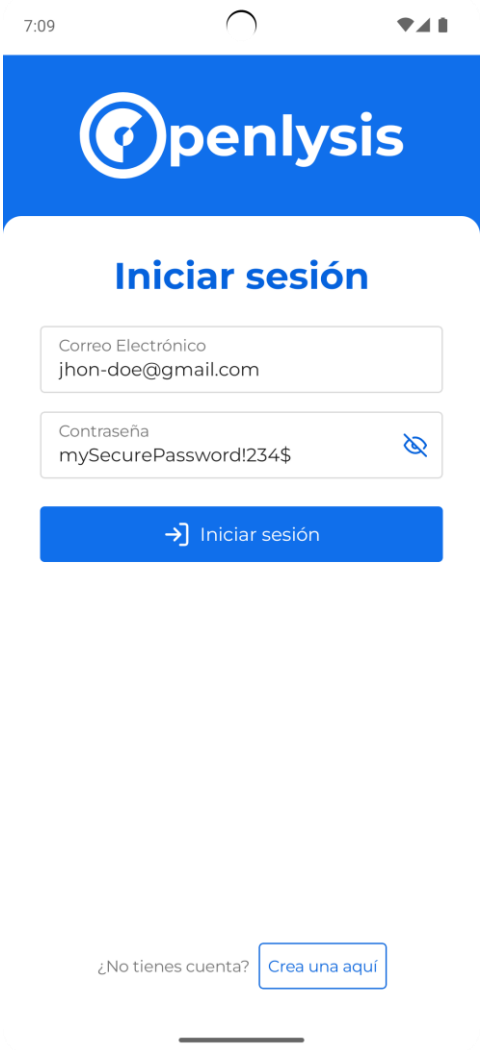
Pantalla desarrollada para el registro de usuarios



Fuente. Autoría propia.

Figura 59.

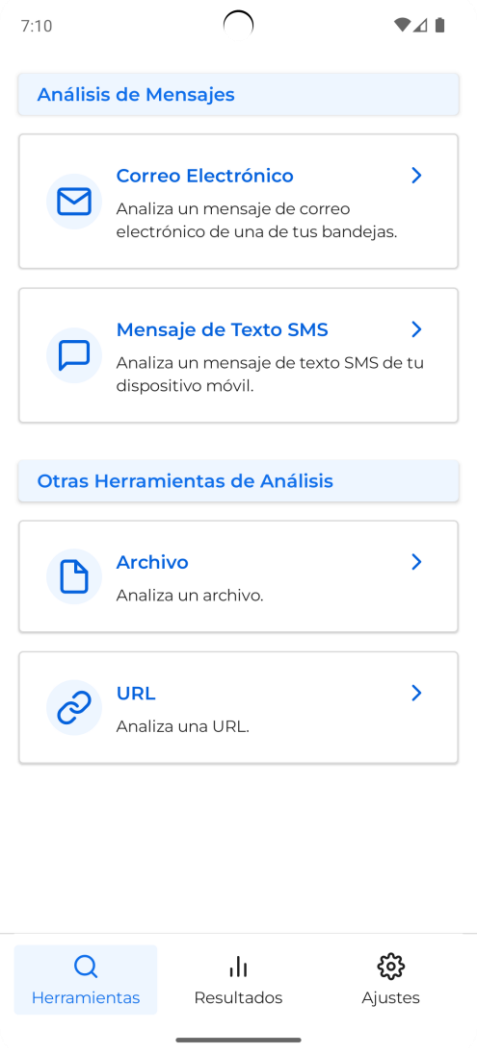
Pantalla desarrollada para el inicio de sesión



Fuente. Autoría propia.

Figura 60.

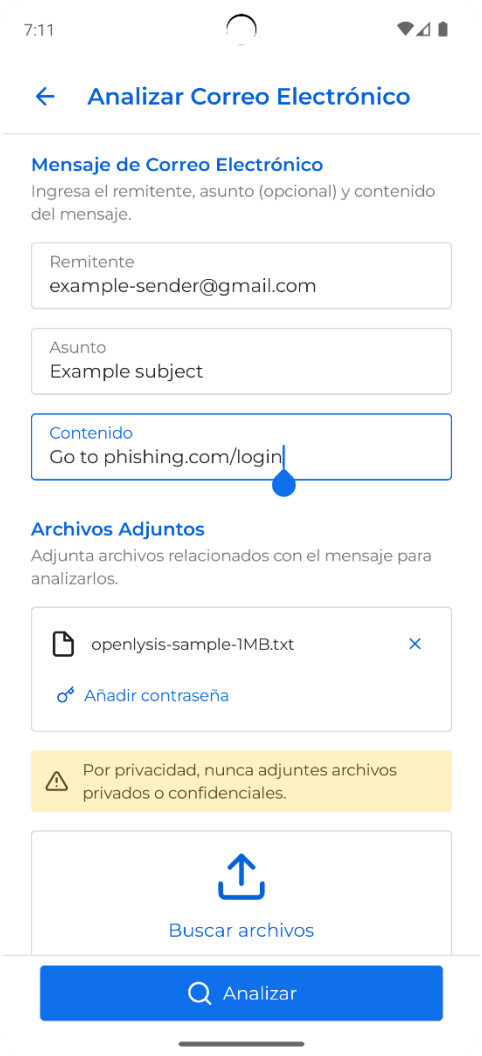
Pantalla desarrollada para las herramientas de análisis



Fuente. Autoría propia.

Figura 61.

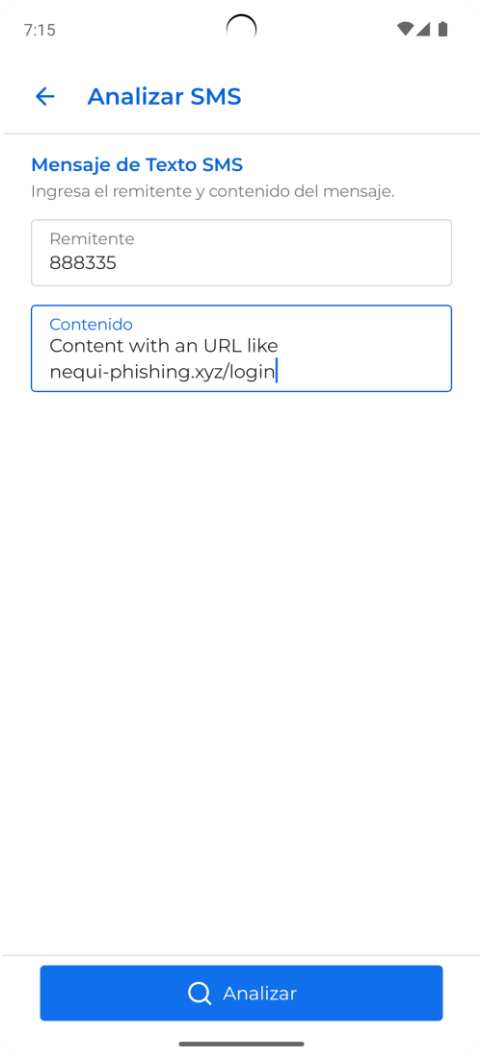
Pantalla desarrollada para analizar correos electrónicos



Fuente. Autoría propia.

Figura 62.

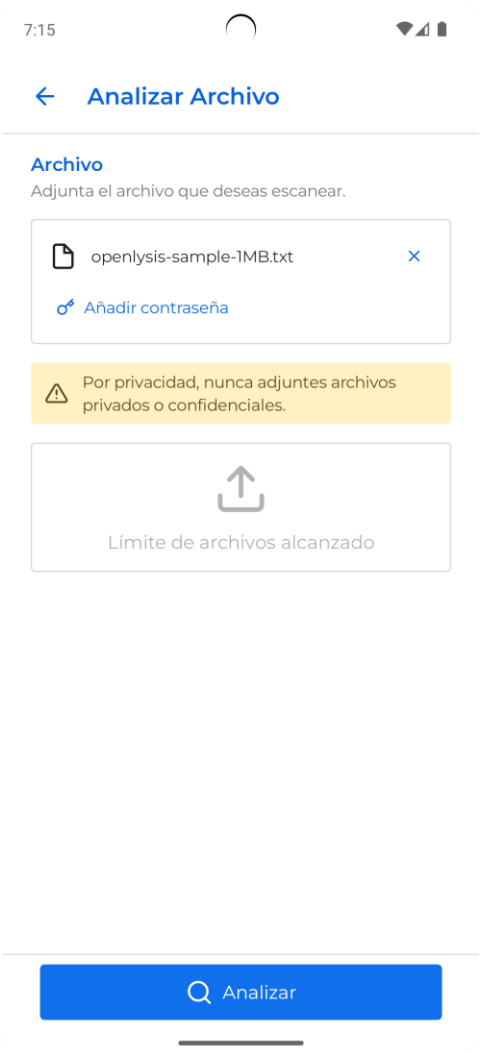
Pantalla desarrollada para analizar mensajes de texto SMS



Fuente. Autoría propia.

Figura 63.

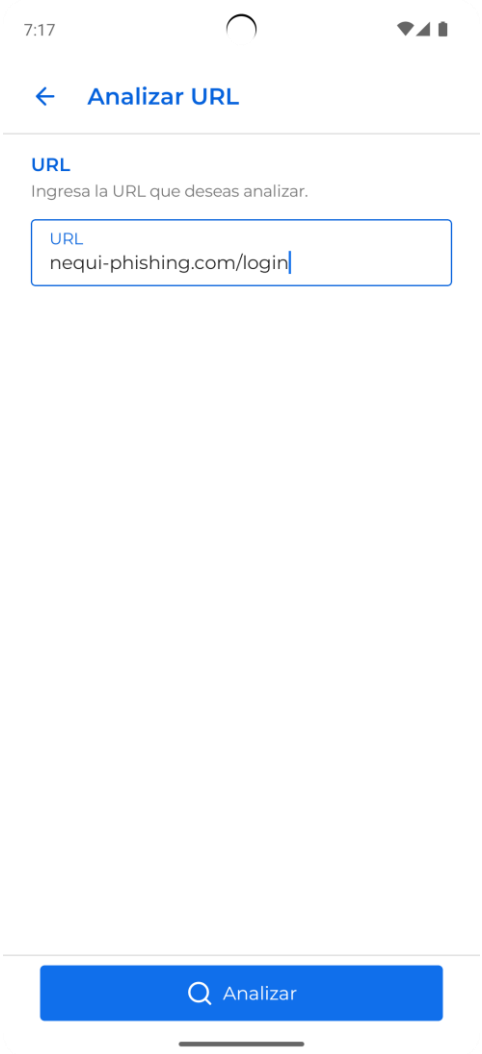
Pantalla desarrollada para analizar archivos



Fuente. Autoría propia.

Figura 64.

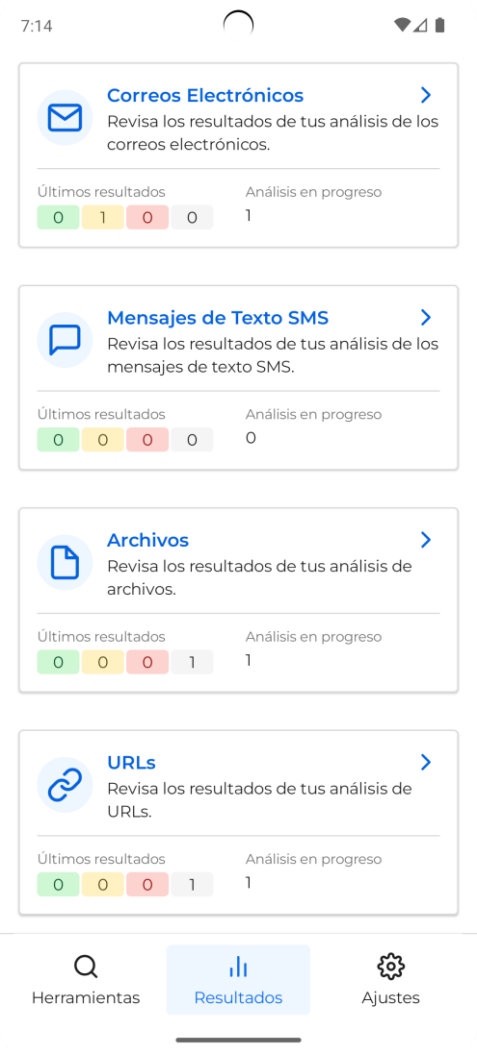
Pantalla desarrollada para analizar URLs



Fuente. Autoría propia.

Figura 65.

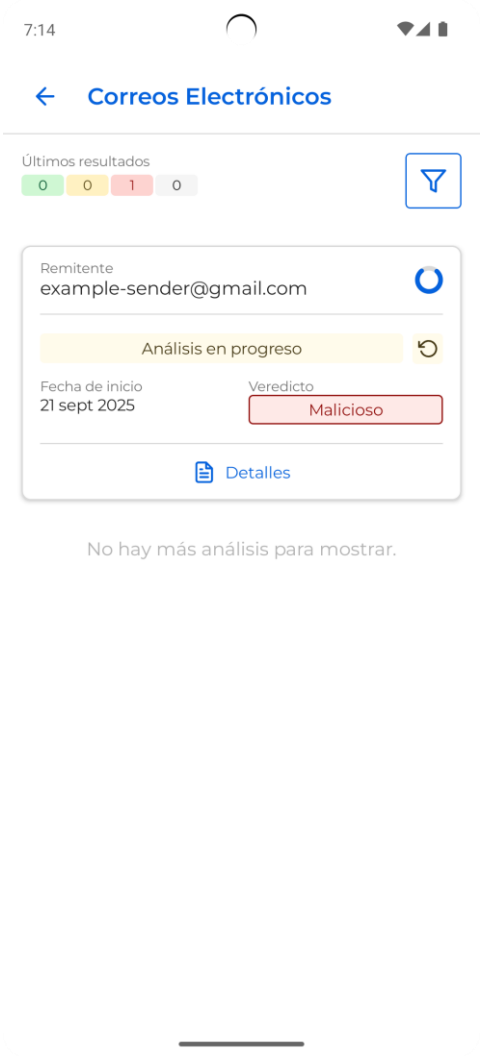
Pantalla desarrollada para la interfaz inicial de los resultados de análisis



Fuente. Autoría propia.

Figura 66.

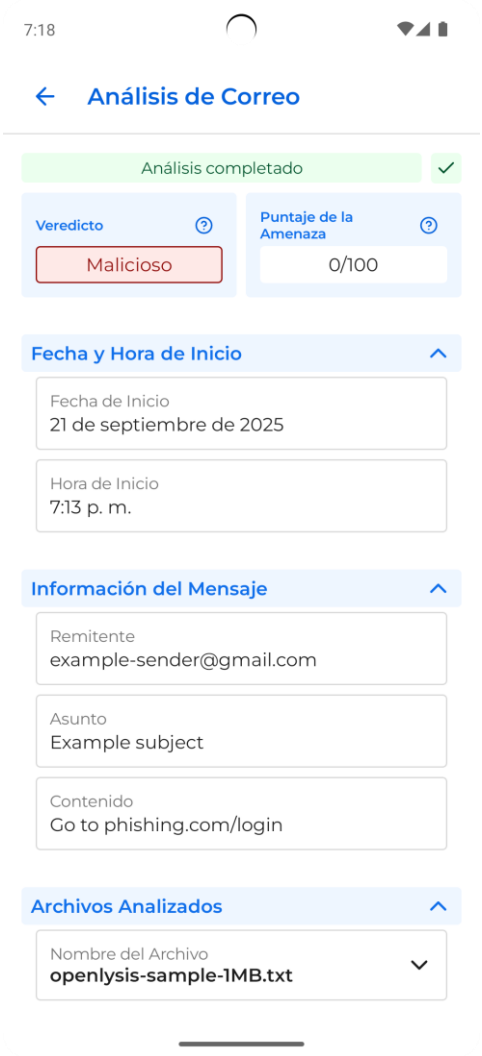
Pantalla desarrollada para la lista de resultados de análisis



Fuente. Autoría propia.

Figura 67.

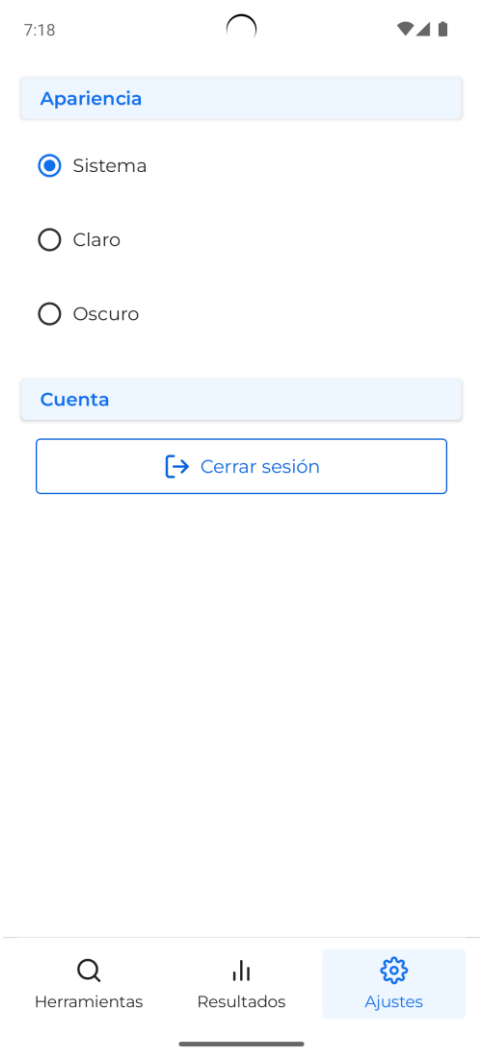
Pantalla desarrollada para los detalles de un resultado de análisis



Fuente. Autoría propia.

Figura 68.

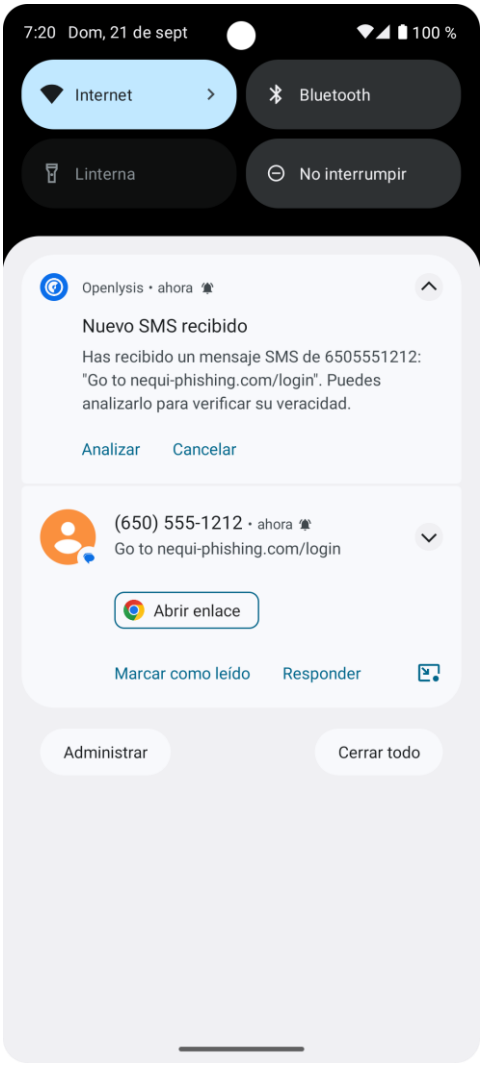
Pantalla desarrollada para la configuración



Fuente. Autoría propia.

Figura 69.

Detección automática de mensajes de texto SMS analizables



Fuente. Autoría propia.

Evaluación del Sistema

El desarrollo de los sistemas y su correcto funcionamiento son la muestra principal del cumplimiento de los objetivos establecidos para el proyecto, sin embargo, también es importante determinar la eficiencia y eficacia del sistema en general, ya que permiten destacar las fortalezas y debilidades, en especial en la detección de phishing y smishing, que pueden manifestar las oportunidades de mejora para futuros proyectos. Por lo tanto, a continuación, se destacan las metodologías de evaluación y los resultados obtenidos.

Sistema Back-end

El sistema back-end, siendo el actor principal para analizar datos, será evaluado desde dos aspectos esenciales: la capacidad de detección y el rendimiento general.

Metodología de Pruebas

Procedimiento de la Evaluación de Detección. Para comprender la capacidad de detectar datos maliciosos, el sistema es evaluado en base al análisis de archivos y URLs, mayormente a este último ya que es el elemento principal tanto del phishing como del smishing. Por ello, las funcionalidades de análisis de mensajes de texto SMS y correos electrónicos no son consideradas, dado que sus resultados serían redundantes e incrementarían la complejidad de la evaluación. Esto se debe a que el proceso de análisis de un mensaje consiste, en esencia, en extraer las URLs de su contenido para analizarlas con los archivos adjuntos. En consecuencia, al reutilizar las mismas URLs y archivos de muestra, los resultados obtenidos no aportarían información adicional ni variaciones.

Por otro lado, los resultados de la evaluación serán registrados empleando la matriz de confusión, que propone un sistema basado en cuatro variables: los verdaderos positivos, los verdaderos negativos, los falsos positivos y los falsos negativos (Murel & Kavlakoglu, 2024).

Habitualmente, esta técnica se aplica en modelos de aprendizaje automático, pero para esta situación, es una buena herramienta para evaluar al sistema.

Verdaderos Positivos (VP). Indica los datos catalogados como maliciosos, cuyo veredicto dado por el sistema fue coincidente.

Verdaderos Negativos (VN). Indica los datos catalogados como legítimos, cuyo veredicto dado por el sistema fue coincidente.

Falsos Positivos (FP). Indica los datos catalogados como legítimos, pero el veredicto del sistema fue malicioso, es decir, un resultado incorrecto.

Falsos Negativos (FN). Indica los datos catalogados como maliciosos, pero el veredicto del sistema fue legítimo, es decir, un resultado incorrecto.

Es relevante comprender que el sistema back-end agrupa los análisis de los servicios externos y determina uno de los siguientes veredictos a rasgos generales: desconocido, limpio, sospechoso y malicioso. Esto señala la necesidad de mapear los veredictos en valores compatibles con las cuatro métricas mencionadas anteriormente y para ello se expone la siguiente tabla.

Tabla 12.

Mapeo de veredictos del sistema a las métricas de la matriz de confusión

Veredicto del Sistema	Clasificación Binaria	Justificación
Malicioso	Positivo (Malicioso)	Detección de una amenaza confirmada.
Sospechoso	Positivo (Malicioso)	Detección de indicadores de riesgo, lo que se considera como una alerta válida, principalmente por precaución.
Limpio	Negativo (Legítimo)	No se encontró ninguna evidencia para catalogarlo como amenaza.
Desconocido	Negativo (Legítimo)	No se pudo determinar un resultado, lo que no genera una alerta de amenaza.

Fuente. Autoría propia.

Finalmente, en base a esas cuatro métricas, se pueden generar distintos indicadores clave que sean aptos para ser analizados y detallen la eficacia del sistema, para este caso se definen los siguientes.

Exactitud. Indica el porcentaje total de aciertos: $\frac{VP+VN}{Total}$.

Precisión. Indica la fiabilidad del sistema al detectar datos como maliciosos: $\frac{VP}{VP+FP}$.

Sensibilidad. Indica la capacidad para detectar datos que realmente son maliciosos: $\frac{VP}{VP+FN}$.

Procedimiento de la Evaluación del Rendimiento. La medición del rendimiento se lleva a cabo aplicando tres tipos de pruebas de carga: las pruebas de humo, encargadas de verificar la funcionalidad del sistema con una carga mínima en un lapso corto; las pruebas de carga promedio, que evalúan la capacidad del sistema bajo un ambiente similar al de producción y las pruebas de estrés, que ayudan a determinar el comportamiento del sistema bajo cargas altamente superiores a las esperadas (Grafana, s.f.).

Para estas pruebas, se simulan interacciones mediante usuarios virtuales (VUs), los cuales envían peticiones de forma iterativa y constante durante un tiempo específico, lo que permite evaluar el sistema bajo ambientes configurables y similares a situaciones reales.

En este caso en concreto, las pruebas se aplican al análisis de archivos, URLs y mensajes, teniendo una duración de un minuto en las pruebas de humo, diez minutos en las de carga y un total de 27 minutos para las pruebas de estrés.

A continuación, se explican las reglas y otros factores tenidos en cuenta para la creación y ejecución de las pruebas.

- La ejecución de los análisis se simula con una duración de entre 3 y 6 minutos y un veredicto aleatorio. Esto se logra a través de un pequeño modulo dedicado, que permite controlar la duración y los resultados de los análisis. Aun así, se debe considerar que este módulo consume recursos ya que se acopla directamente al orquestador de análisis, pero su impacto es mínimo.
- Cada prueba se aplica a un sistema back-end completamente restablecido, para evitar acumular efectos con otras pruebas ya realizadas.

- El sistema back-end no utiliza el servicio de Google Cloud Storage para evitar costos económicos en caso de que el sistema sea capaz de procesar un significativo volumen de peticiones al analizar archivos o mensajes.
- El tamaño de los archivos empleados en las pruebas es de 80 kibibytes (KiB), para prevenir un uso excesivo del almacenamiento de la maquina física. Además, son generados con bytes aleatorios y posteriormente convertidos a una cadena texto en base64.
- Las URLs utilizadas se generan de forma aleatoria, pero garantizando que el formato sea válido.
- Los mensajes empleados para las pruebas contienen una URL en el cuerpo y un archivo adjunto.
- Las mediciones de rendimiento se registran de forma automática mientras las pruebas de carga están en ejecución.

Hardware. Los recursos que soportan al sistema back-end durante las pruebas de rendimiento son indicadores muy útiles para planear despliegues a entornos de producción y proyectar costos, independientemente de cuales sean los proveedores utilizados. Por lo tanto, el orquestador de análisis y la API principal, dispondrán cada uno de un núcleo de un procesador Ryzen 5600G y 2 GB de RAM. La API de autenticación no es considerada en las evaluaciones puesto que es un servicio básico y no enteramente asociado al análisis de datos, por lo que sus resultados no tendrían mayor importancia. Lo mismo ocurre con la base de datos y el intermediario de mensajería, al ser servicios no desarrollados en este proyecto y contar con proveedores que facilitan su escalabilidad, su medición es prescindible. Aun así, para mantener un entorno controlado y evitar el consumo de recursos de forma excesiva, estos últimos tres servicios estarán limitados a la mitad de un solo núcleo y 1 GB de RAM cada uno.

Herramientas Utilizadas. El sistema back-end se despliega en contenedores de Docker, lo que permite ejecutar a los servicios internos en un entorno aislado y controlar los recursos disponibles para cada uno. Las pruebas de rendimiento y carga se realizan con K6, una herramienta de código abierto que ayuda a comprobar la resiliencia de aplicaciones, simular peticiones y recolectar métricas de consumo de latencia y concurrencia (Grafana, 2025).

Conjuntos de Datos de Pruebas de Detección

Para evaluar la eficacia respecto a la detección de phishing, smishing y malware en general, se usa un conjunto de datos de 100 muestras, que, si bien puede ser pequeño, aporta un gran valor porque ayuda a detectar aquellas falencias que en trabajos posteriores se puedan mejorar y, permite adherirse a las limitaciones de uso de los servicios externos. El conjunto de datos se constituye por 50 muestras legitimas y otras 50 maliciosas, las cuales serán obtenidas mediante conjuntos y bases de datos existentes.

Archivos. Los archivos legítimos son extraídos de un conjunto de datos ofrecido por la universidad de New Brunswick de Canadá y los maliciosos se generan utilizando un programa escrito en Python, los enlaces se pueden encontrar en el Apéndice H e Apéndice I respectivamente. Los archivos están en el formato de documento portátil (PDF), ya que este es el más propenso a utilizarse en correos electrónicos y para un usuario común, sería mucho más confiable abrir este tipo de archivos, que otros en diferentes formatos, puesto que a simple vista se verían extraños.

URLs. Para obtener las URLs se utiliza un conjunto de datos de Kaggle, que se encuentra en el Apéndice J, del cual se extraen únicamente las URLs legítimas a pesar de ofrecer muestras maliciosas, ya que, para estas últimas, el tiempo de vida suele ser corto y pueden contener datos antiguos, lo que puede afectar el proceso de detección y no brindar resultados realistas. En su lugar, se usa PhishTank, una base de datos que se especializa en almacenar URLs asociadas a phishing, que están verificadas y, además, indica si los sitios web están en vigencia, lo cual es un valor agregado para las pruebas. Es importante aclarar que estas muestras no incluyen *zero-day* URLs, un concepto que abarca phishing emergente, debido a que requieren elaborar sitios web de prueba, lo que aumentaría exponencialmente la complejidad de la evaluación.

Resultados de Pruebas de Detección

Ejecutadas las pruebas y calculado los resultados de detección a través de la matriz de confusión, se obtiene una exactitud del 85%, una precisión del 77% y una sensibilidad del 100% para el análisis de URLs. Por su lado, el análisis de archivos tiene mejor puntuación, con una exactitud del 98%, una precisión del 96% y una sensibilidad del 100%. Para la observación de los resultados a detalle se puede utilizar el enlace del Apéndice K.

Tabla 13.

Resultados de la evaluación de detección en el sistema back-end

Variable	Resultados de Análisis de Archivos	Resultados de Análisis de URLs
Verdaderos Positivos	50	50
Verdaderos Negativos	48	35
Falsos Positivos	2	15
Falsos Negativos	0	0
Exactitud	0.98 (98%)	0.85 (85%)
Precisión	0.96 (96%)	0.77 (77%)
Sensibilidad	1 (100%)	1 (100%)

Fuente. Autoría propia.

Los resultados de la evaluación demuestran que el sistema es capaz de detectar eficazmente URLs maliciosas sin dejar pasar una sola amenaza, como lo indica la sensibilidad, sin embargo, el sistema aún tiene oportunidades de mejoras, en especial, en la fiabilidad, a pesar de que el porcentaje de precisión es alto, no es lo suficiente como para que los usuarios puedan estar totalmente confiados en los veredictos del sistema, lo que propone cambios en la interpretación de los análisis de los servicios de externos, por ejemplo, acoplar un modelo de aprendizaje automático que sea efectivo en la combinación de los análisis para determinar un veredicto final.

Por su parte, el análisis de archivos es más eficaz, particularmente en la precisión, demostrando que los veredictos sobre los ficheros PDF son confiables, lo que manifiesta que los análisis se benefician de este formato al contener datos principalmente estáticos, facilitándole la detección de indicadores de compromiso en el archivo, como, por ejemplo, la ejecución de código de JavaScript. Aun así, se debe considerar el alcance de esta evaluación, por ello sería interesante incluir otros formatos de archivos que no necesariamente se relacionen al phishing, pero que si sean un complemento a la seguridad de los usuarios.

Resultados de Pruebas de Rendimiento

Ejecutadas las pruebas de carga y medido el rendimiento y consumo de recursos en el sistema back-end, se registran los datos más relevantes en un conjunto de tablas que muestran los resultados para los análisis de archivos, URLs y mensajes. Además, en el Apéndice L se adjunta el enlace para observar en detalle los registros de cada una de las pruebas.

Primeramente, la Tabla 14 muestra, que, en una carga de estrés de análisis de archivos, el sistema es capaz de gestionar un total de 39.806 peticiones en 27 minutos, presentando fallas en el 0.80% del total, es decir, 320 peticiones. Sin embargo, bajo una carga esperada, el sistema no presenta fallas.

Tabla 14.

Resultados de pruebas de carga de análisis de archivos en el back end

Tipo de Prueba	VUs	Peticiones Totales	Peticiones por Segundo	Peticiones Fallidas	Latencia p (95) (Milisegundos)
Prueba de Humo	10	4771	72,21	0	192,02
Prueba de Carga	30	23041	37,34	0	1400
Prueba de Estrés	100	39806	23,76	320	8070

Fuente. Autoría propia.

En cuanto a los análisis de URLs, el sistema es considerablemente más eficiente, como se muestra en la Tabla 15. En la prueba de estrés, se gestionaron 100.721 peticiones, lo que representa un aumento del ~253% en comparación con las pruebas de análisis de archivos y, un ~246% en las pruebas de cargas esperadas. La diferencia de los resultados principalmente se debe al costo adicional de gestionar los archivos, ya que incrementan significativamente el tamaño de la petición, haciendo que la API principal tarde más en leerla y, por ende, no se reciban las mismas peticiones por segundo.

Tabla 15.

Resultados de pruebas de carga de análisis de URLs en el back end

Tipo de Prueba	VUs	Peticiones Totales	Peticiones por Segundo	Peticiones Fallidas	Latencia p (95) (Milisegundos)
Prueba de Humo	10	9188	132,12	0	93,11
Prueba de Carga	30	56710	91,66	0	498,73
Prueba de Estrés	100	100721	60,12	503	2990

Fuente. Autoría propia.

Por otro lado, están los análisis de mensajes, que incorporan una URL y un archivo, lo que trae consigo las limitaciones reflejadas en los resultados anteriores, además del sobrecoste de extraer las URLs del contenido del mensaje. La Tabla 16 muestra que esta es la funcionalidad menos eficiente puesto que gestiona 24.122 peticiones en 27 minutos, pero lo más notable son las 1.551 peticiones fallidas que representan el 6,42% de la totalidad, lo que, en resultados anteriores, apenas se acercaba al 1% como máximo. Adicionalmente, en la prueba de carga esperada, se procesaron 14.727 peticiones, 36% menos que en los análisis de archivos. Esto manifiesta que la API principal podría delegar el trabajo de extracción y manipulación de mensajes a otro servicio interno, para únicamente enfocarse en recibir las peticiones provenientes.

Tabla 16.

Resultados de pruebas de carga de análisis de mensajes en el back end

Tipo de Prueba	VUs	Peticiones Totales	Peticiones por Segundo	Peticiones Fallidas	Latencia p (95) (Milisegundos)
Prueba de Humo	10	2244	33,03	0	285,25
Prueba de Carga	30	14727	23,76	0	1690
Prueba de Estrés	100	24122	14,4	1551	14490

Fuente. Autoría propia.

Respecto al consumo de recursos, la Tabla 17 muestra un uso del procesador decreciente basado en la carga, lo cual concuerda con los resultados de latencia registrados anteriormente, particularmente en la API principal, ya que, entre mayor latencia de respuesta, menor es la cantidad de peticiones de los usuarios virtuales, lo que al final se traduce en un consumo menor del procesador. Esto se debe a los tiempos de espera durante la lectura de archivos, debido a que no existe una carga sobre el procesador, al ser un factor netamente asociado al disco de almacenamiento y a la red. Por lo tanto, esta funcionalidad puede representar un cuello de botella en un escenario de producción, siendo mayormente causado por la velocidad de red para subir archivos a la zona de almacenamiento, en especial aquellos con un gran tamaño, lo que igualmente se puede reflejar en el orquestador de análisis durante las descargas.

Por otro lado, el consumo de memoria es un aspecto interesante que destaca la eficiencia de los servicios al emplearla, utilizando como máximo el ~13,8% de la disponible, en el caso de la API principal.

Tabla 17.

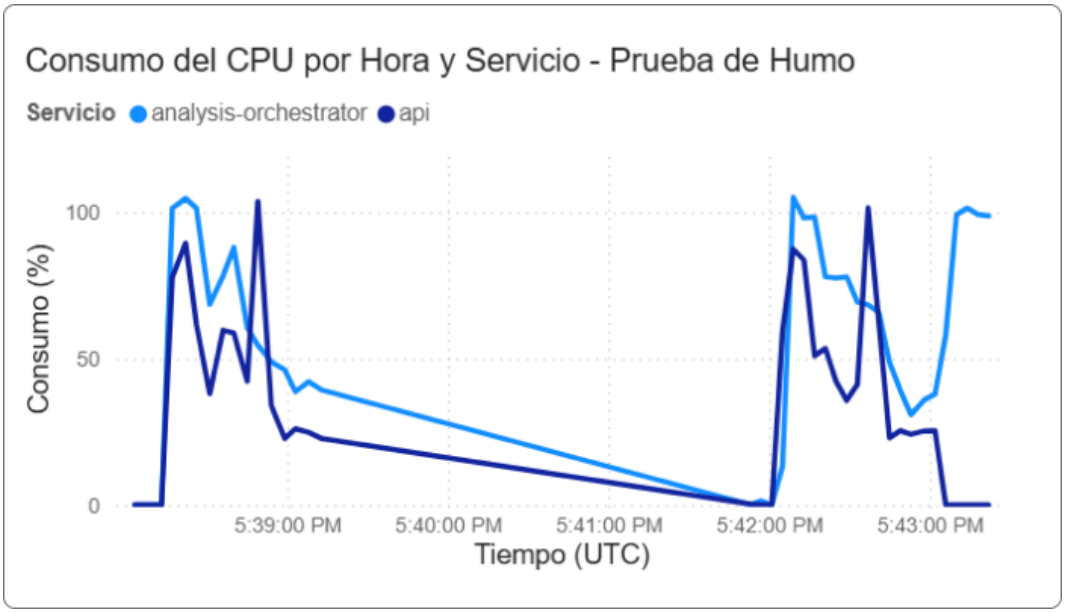
Consumo de recursos en las pruebas de análisis de archivos en el back end

Tipo de Prueba	Servicio	Consumo Promedio de CPU (%)	Consumo Mínimo de Memoria (MiB)	Consumo Máximo de Memoria (MiB)
Prueba de Humo	API principal	36,04	121,3	165
	Orquestador	58,38	110,1	161,9
Prueba de Carga	API principal	11.97	121.80	224.20
	Orquestador	26,68	117,20	184,2
Prueba de Estrés	API principal	9,12	121,6	275
	Orquestador	12,92	111,2	179,9

Fuente. Autoría propia.

Figura 70.

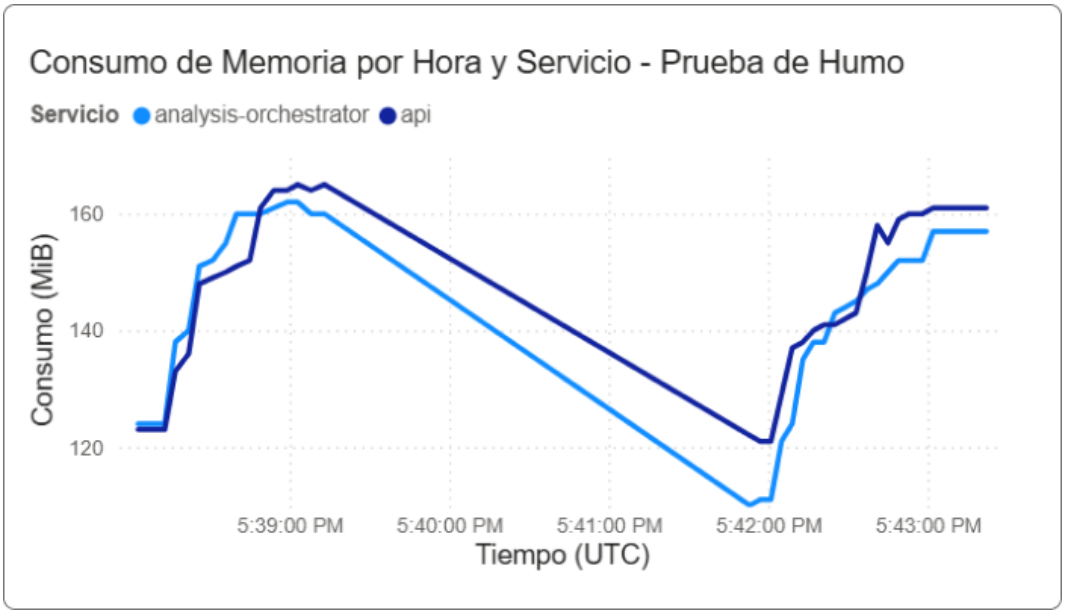
Consumo del CPU en la prueba de humo de análisis de archivos



Fuente. Autoría propia.

Figura 71.

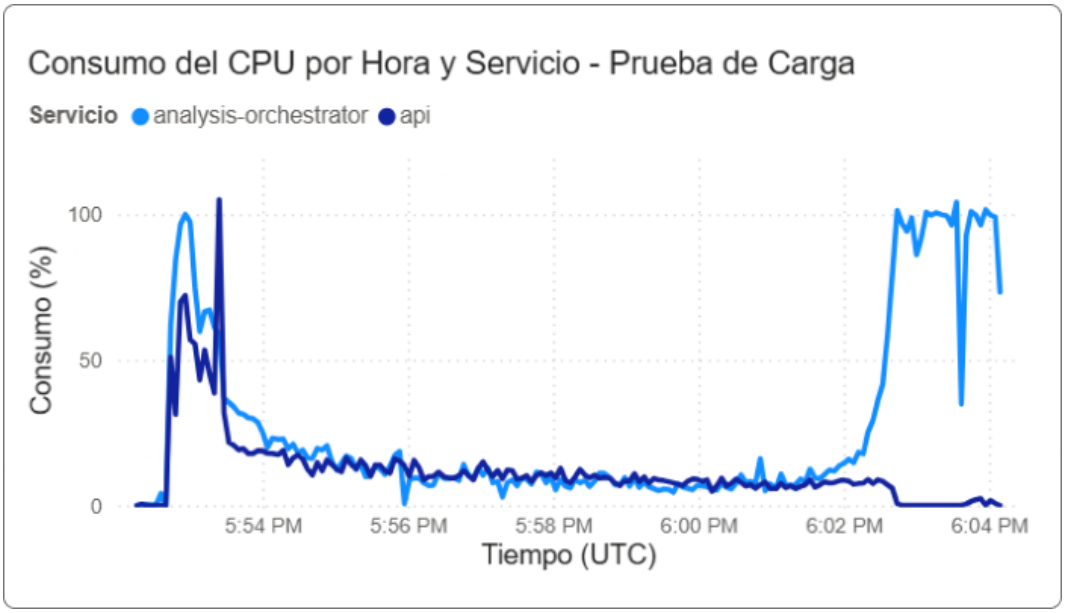
Consumo de memoria en la prueba de humo de análisis de archivos



Fuente. Autoría propia.

Figura 72.

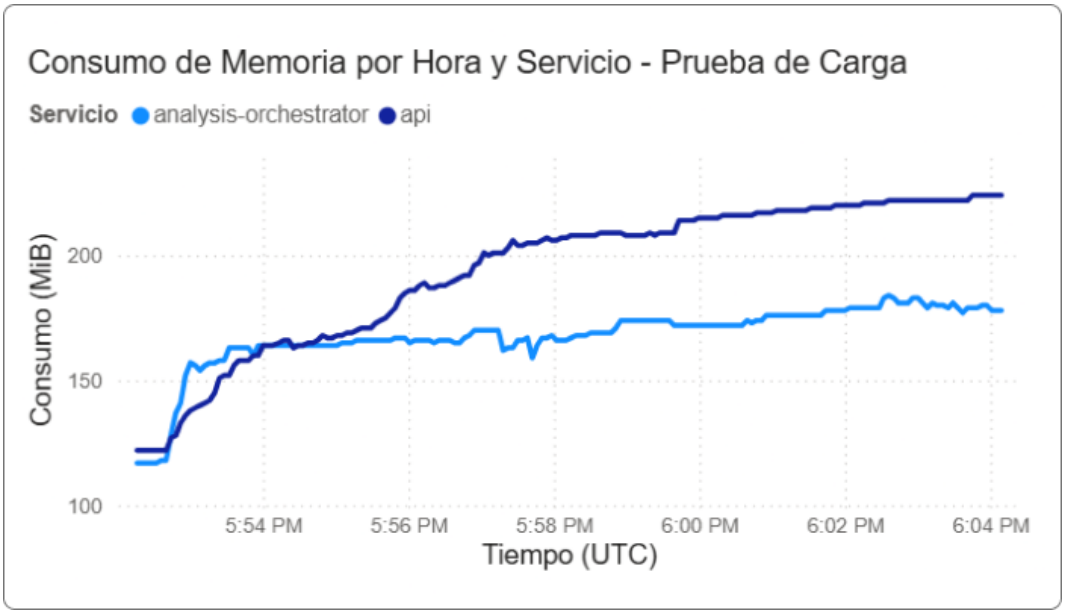
Consumo de CPU en la prueba de carga de análisis de archivos



Fuente. Autoría propia.

Figura 73.

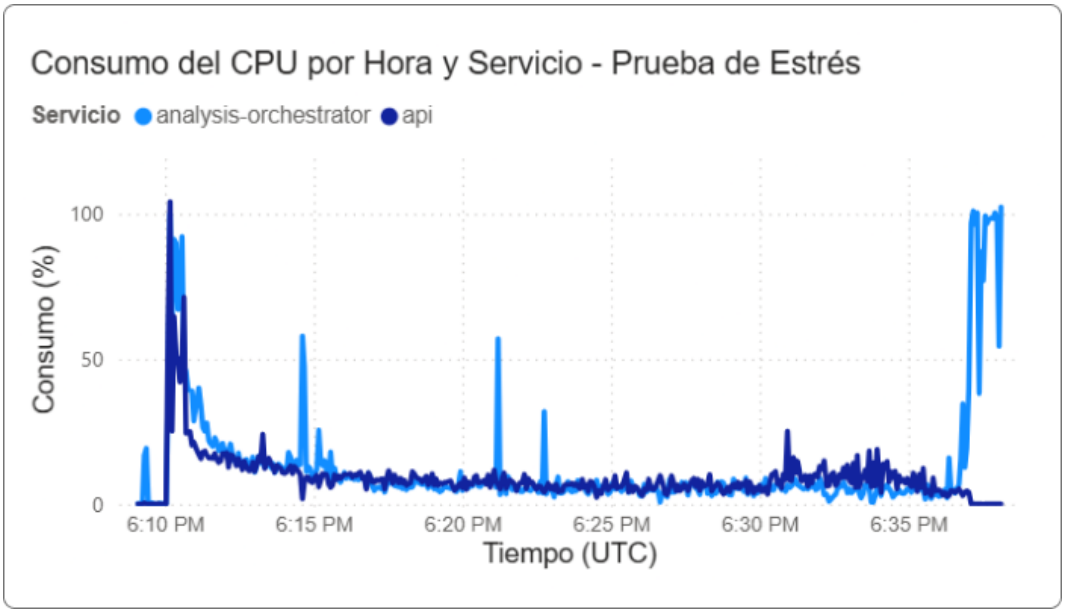
Consumo de memoria en la prueba de carga de análisis de archivos



Fuente. Autoría propia.

Figura 74.

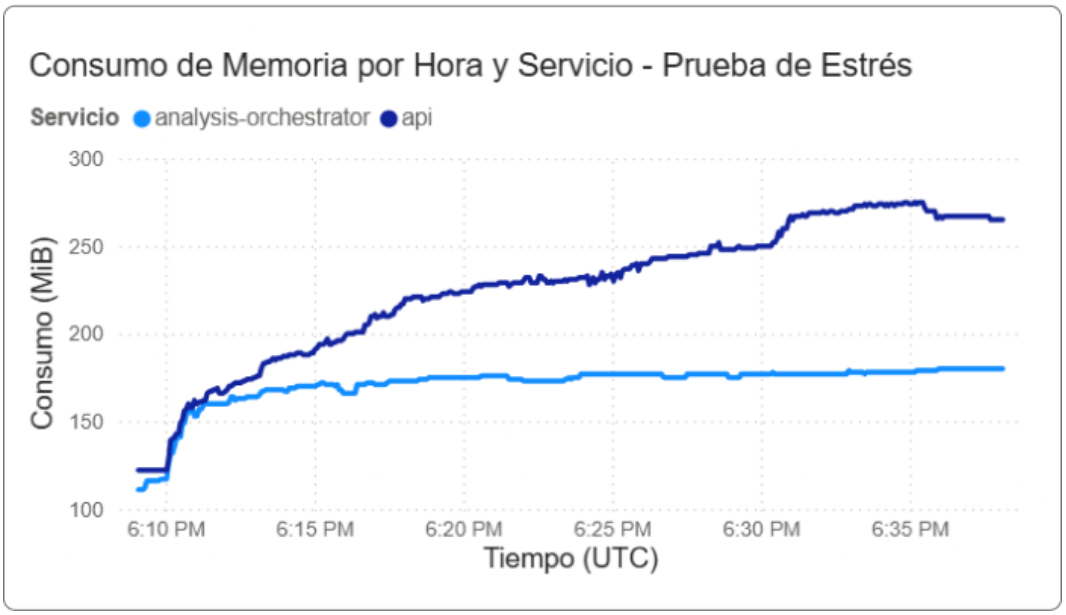
Consumo de CPU en la prueba de estrés de análisis de archivos



Fuente. Autoría propia.

Figura 75.

Consumo de memoria en la prueba de estrés de análisis de archivos



Fuente. Autoría propia.

La Tabla 18 demuestra que el análisis de URLs es una funcionalidad ligera, también coincidente con la baja latencia obtenida durante las pruebas. La utilización del procesador en promedio tiene valores más altos en contraste con los análisis de archivos, reforzando las conclusiones realizadas anteriormente. Aun así, esto recalca que ambos servicios internos están aprovechando eficientemente el procesador, al gestionar tantas peticiones con tan solo un núcleo y sin siquiera llegar a utilizarlo en su mitad de capacidad.

En cuanto al consumo de memoria, nuevamente se mantiene en niveles bajos, llegando a aumentar aproximadamente 100 mebibytes (MiB) en el caso de carga más extremo, considerando que se procesaron 100.721 peticiones.

Tabla 18.

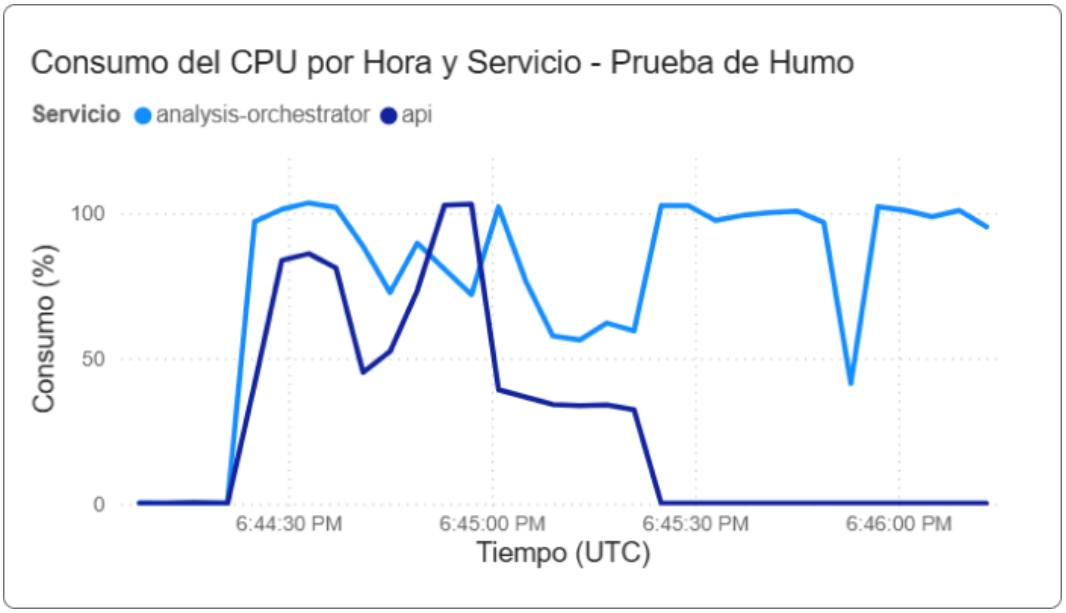
Consumo de recursos en las pruebas de análisis de URLs en el back end

Tipo de Prueba	Servicio	Consumo Promedio de CPU (%)	Consumo Mínimo de Memoria (MiB)	Consumo Máximo de Memoria (MiB)
Prueba de Humo	API principal	27,51	117,6	149,1
	Orquestador	76,95	109,7	160,9
Prueba de Carga	API principal	22,72	117,3	163,4
	Orquestador	30,14	112,8	173,2
Prueba de Estrés	API principal	15,44	114	200,5
	Orquestador	18,83	112,3	179,4

Fuente. Autoría propia.

Figura 76.

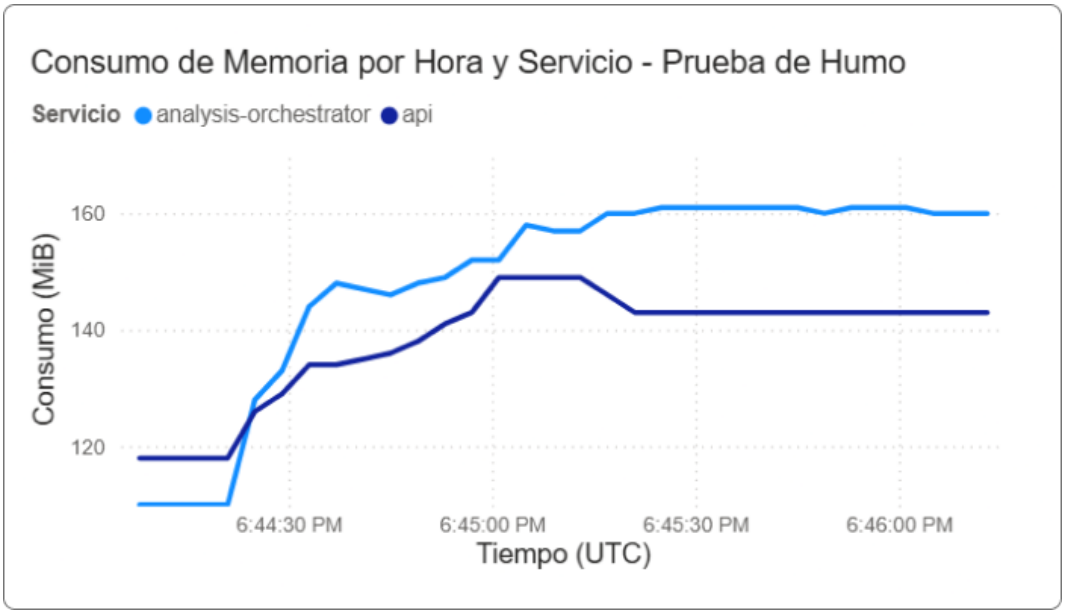
Consumo del CPU en la prueba de humo de análisis de URLs



Fuente. Autoría propia.

Figura 77.

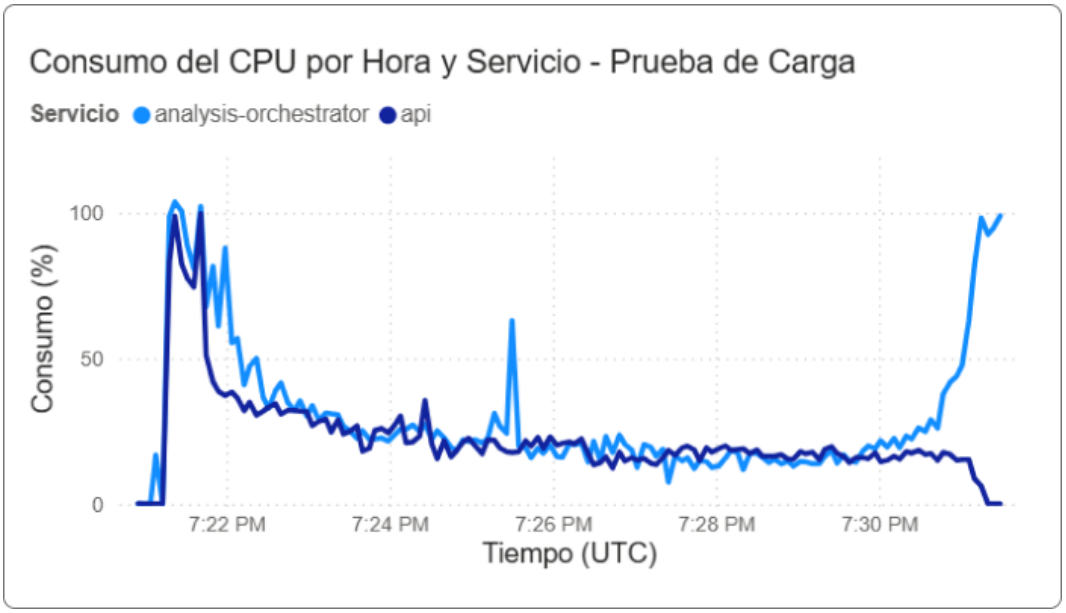
Consumo de memoria en la prueba de humo de análisis de URLs



Fuente. Autoría propia.

Figura 78.

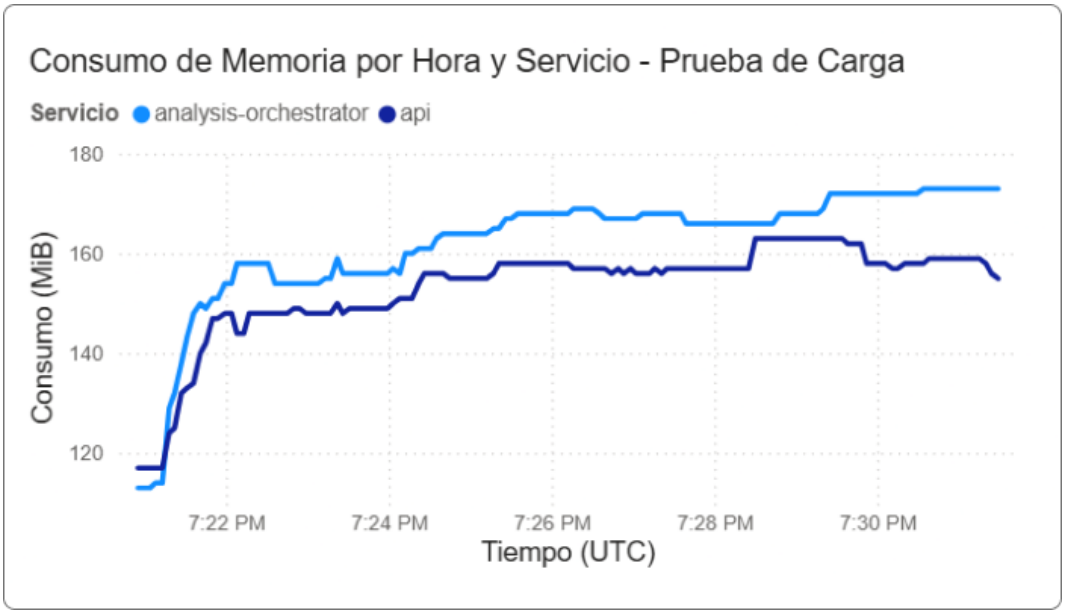
Consumo de CPU en la prueba de carga de análisis de URLs



Fuente. Autoría propia.

Figura 79.

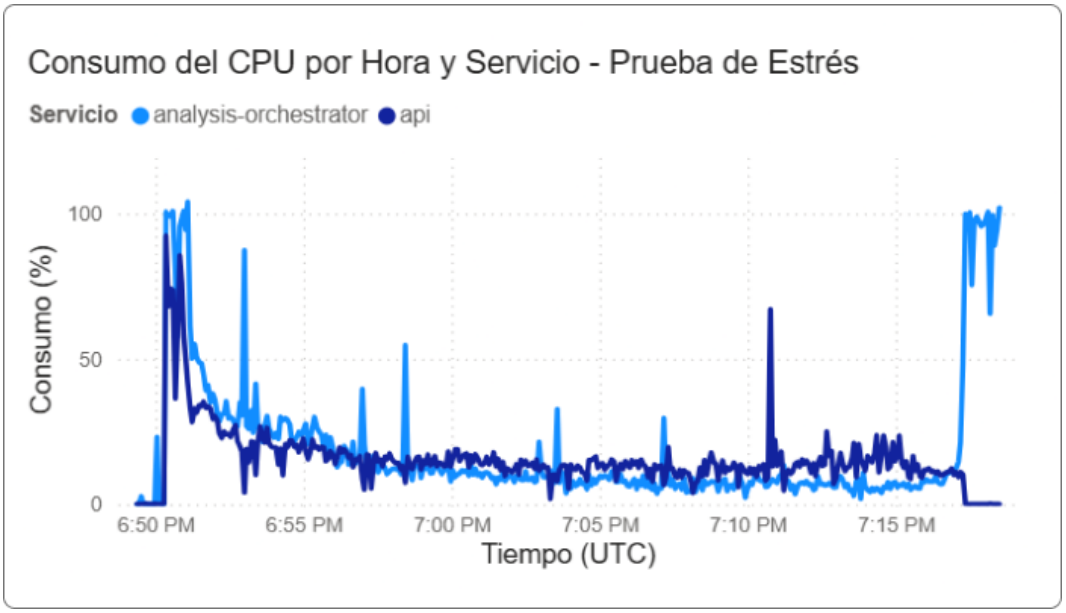
Consumo de memoria en la prueba de carga de análisis de URLs



Fuente. Autoría propia.

Figura 80.

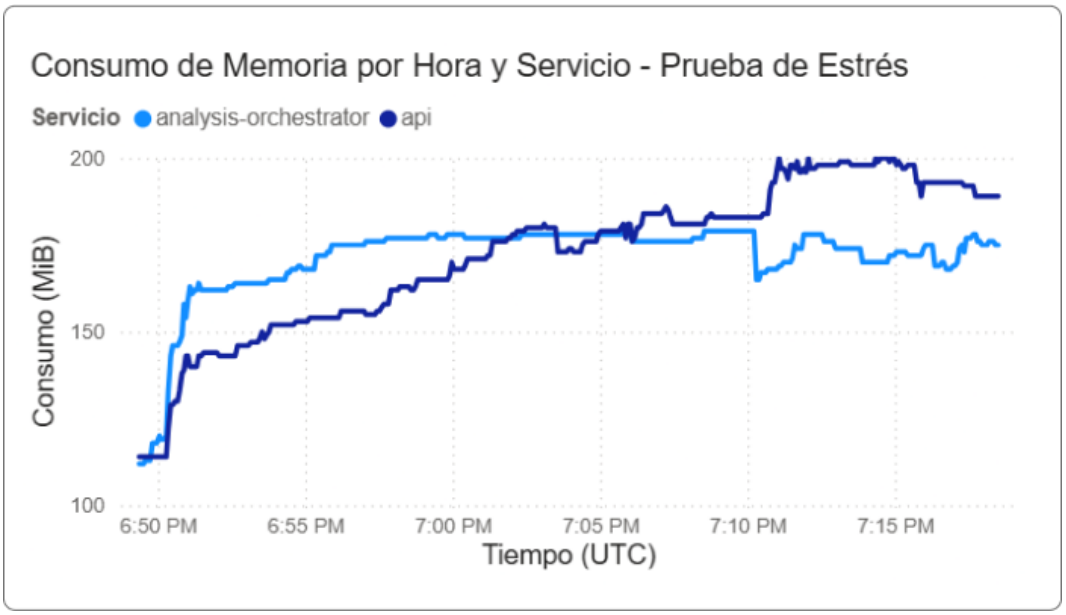
Consumo de CPU en la prueba de estrés de análisis de URLs



Fuente. Autoría propia.

Figura 81.

Consumo de memoria en la prueba de estrés de análisis de URLs



Fuente. Autoría propia.

El análisis de mensajes es la funcionalidad que mayor consumo promedio del procesador registro durante las pruebas, en especial la API principal, como se muestra en la Tabla 19. Estos resultados destacan el impacto que tiene la actualización de los análisis de mensajes, según los resultados de los archivos y URLs analizadas, puesto que, al ser otra tarea llevada a cabo por la API, también emplea recursos. Además, esto explica porque gestiono menor cantidad de peticiones, debido a que el procesador y la velocidad de respuesta se ven limitados al cumplir con dos tareas de carga considerable, mayormente atadas al tiempo de espera de entrada y salida (I/O), como la lectura de archivos y las actualizaciones en la base de datos. Esta situación representa una oportunidad de mejora, porque la API principal podría delegar las tareas relacionadas al procesamiento de mensajes a otro servicio interno, lo cual reduciría la sobrecarga significativamente.

Tabla 19.

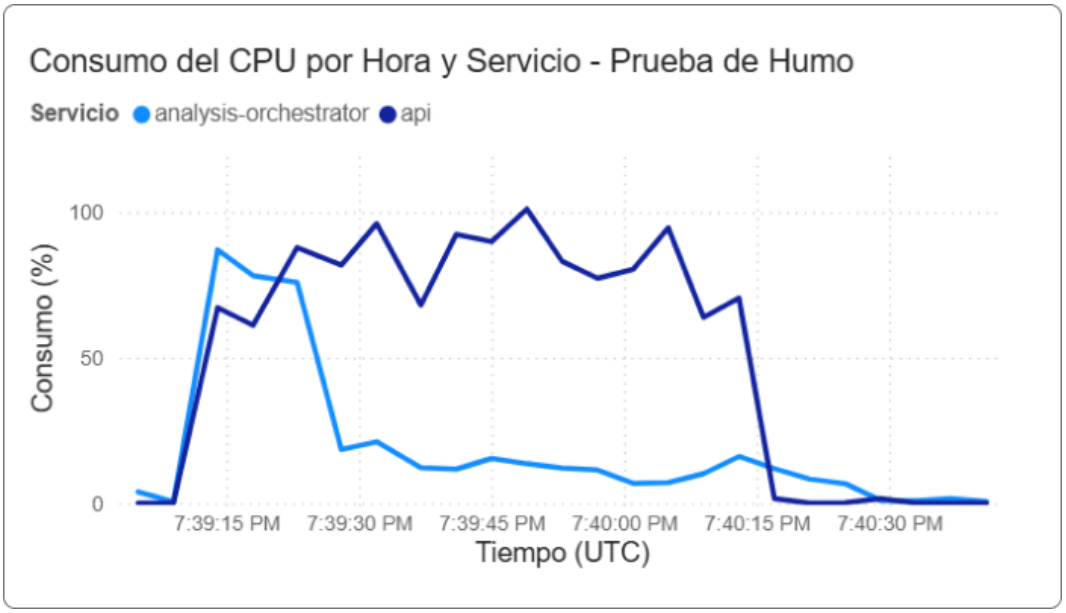
Consumo de recursos en las pruebas de análisis de mensajes en el back end

Tipo de Prueba	Servicio	Consumo Promedio de CPU (%)	Consumo Mínimo de Memoria (MiB)	Consumo Máximo de Memoria (MiB)
Prueba de Humo	API principal	50,82	142	199,2
	Orquestador	18,05	119,5	158,2
Prueba de Carga	API principal	34,77	127,1	229,2
	Orquestador	32,90	111,6	178,1
Prueba de Estrés	API principal	26,66	129,7	275,5
	Orquestador	17,47	110,3	188,3

Fuente. Autoría propia.

Figura 82.

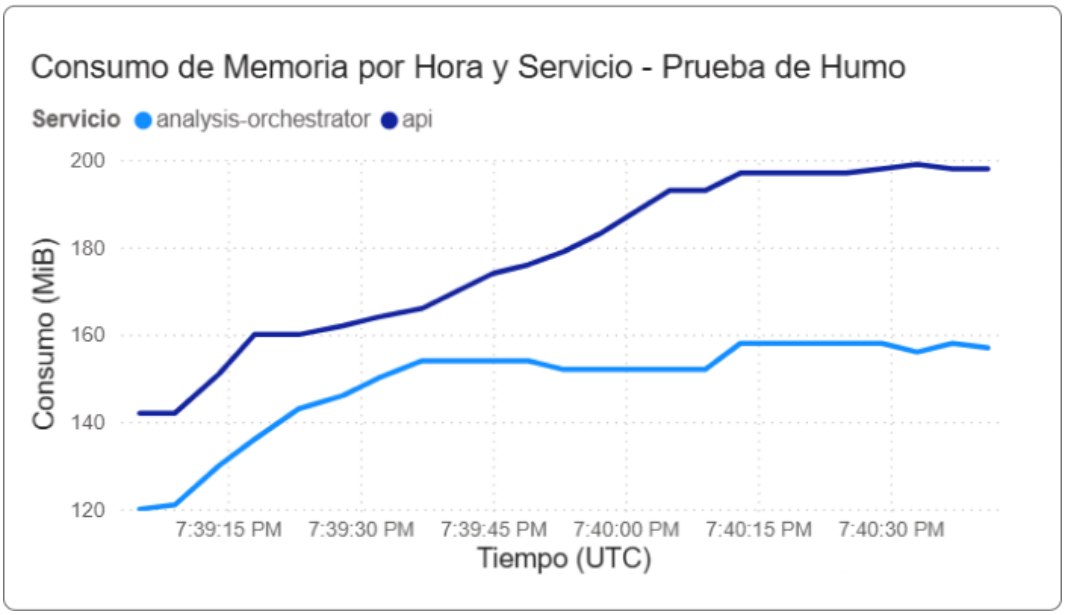
Consumo del CPU en la prueba de humo de análisis de mensajes



Fuente. Autoría propia.

Figura 83.

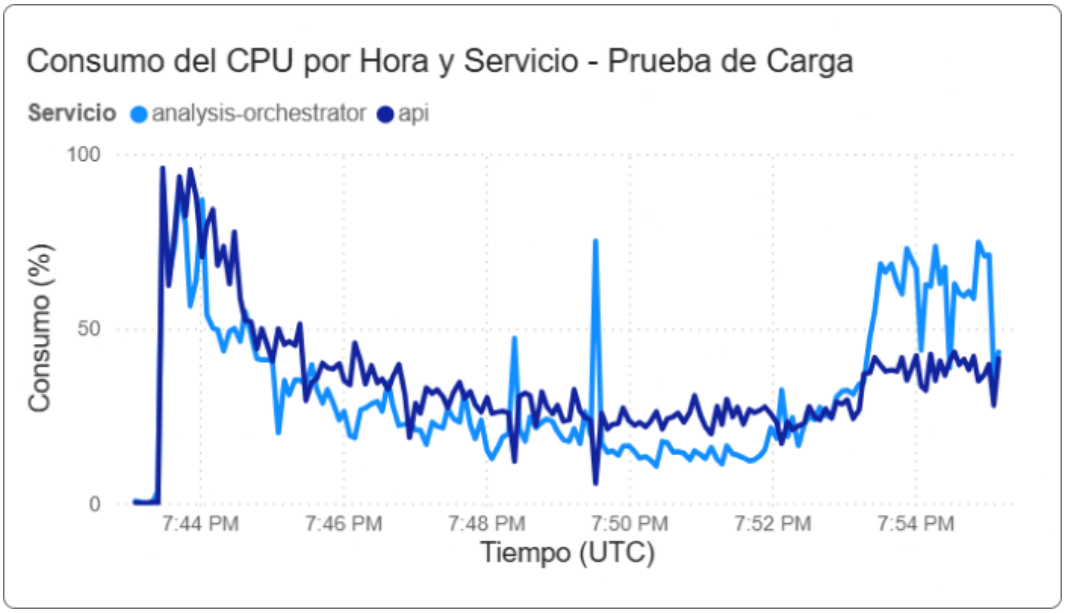
Consumo de memoria en la prueba de humo de análisis de mensajes



Fuente. Autoría propia.

Figura 84.

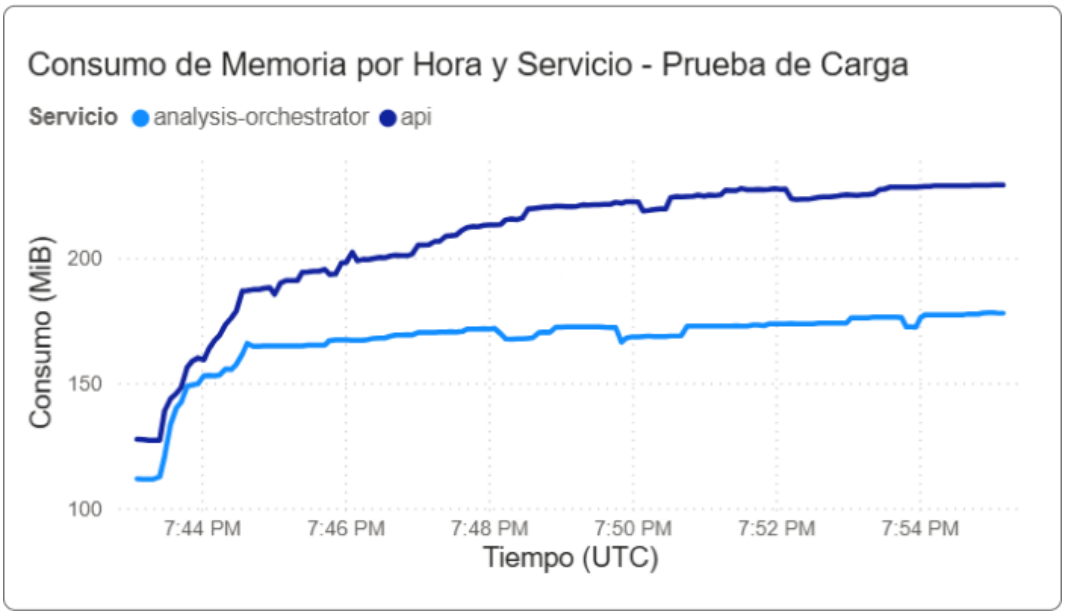
Consumo de CPU en la prueba de carga de análisis de mensajes



Fuente. Autoría propia.

Figura 85.

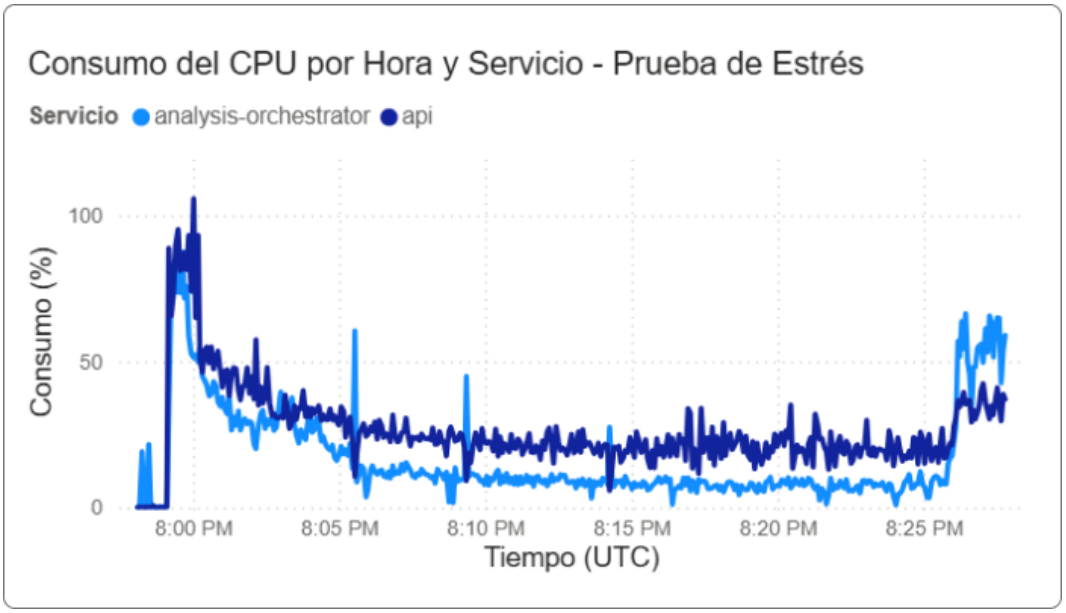
Consumo de memoria en la prueba de carga de análisis de mensajes



Fuente. Autoría propia.

Figura 86.

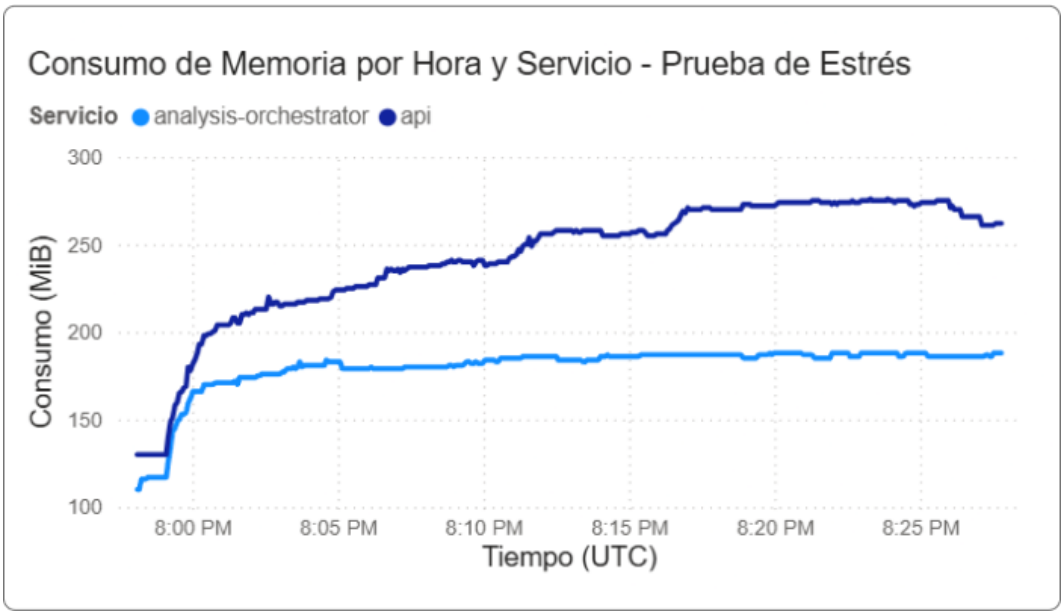
Consumo de CPU en la prueba de estrés de análisis de mensajes



Fuente. Autoría propia.

Figura 87.

Consumo de memoria en la prueba de estrés de análisis de mensajes



Fuente. Autoría propia.

Sistema Front-end

La aplicación de Android, al ser instalada y utilizada directamente en el dispositivo del usuario final, constituye un componente esencial que debe garantizar un desempeño eficiente en todos sus aspectos. Por ello, a continuación, se presenta su evaluación desde el apartado de rendimiento.

Metodología de Pruebas

Procedimiento de la Evaluación del Rendimiento. El sistema se evalúa en base a tres funcionalidades: análisis de URLs, análisis de archivos y análisis de mensajes. Para la ejecución de las pruebas se consideran los siguientes aspectos:

- En cada funcionalidad se realizan tres repeticiones, en las que se mide el consumo del procesador, memoria, batería y red. Los resultados se utilizan para calcular un promedio que permita descartar anomalías de repeticiones iniciales.
- Cada repetición lleva a cabo una tarea típica que realizaría el usuario, lo que significa, que la medición se realiza desde que el usuario inicia el análisis hasta que este finaliza.
- Los análisis en el sistema back-end son simulados, pero en este caso con una duración fija de 2 minutos, con el fin de acelerar las pruebas y tener un entorno controlado que permita comparar los resultados con total fiabilidad.

Hardware. El sistema es ejecutado en la versión 11 de Android, específicamente en el nivel 30 de la API en un dispositivo Redmi Note 8, que cuenta con 4 GB de memoria, un procesador Snapdragon 665 con 8 núcleos a una capacidad máxima de 2.01 GHz y una batería de 4000 mAh.

Herramientas Utilizadas. Para medir el consumo del procesador y la memoria se emplea Android Studio Profiler, que es una herramienta integrada al editor de código Android Studio, que ayuda a rastrear el uso de recursos de las aplicaciones. En cuanto al consumo de batería y de red, se utiliza Android Debug Bridge (ADB), una herramienta de línea de comandos perteneciente al mismo ecosistema, que es capaz de comunicarse con el dispositivo de Android y ofrecer el historial de estos recursos (Google, 2025).

Resultados de Pruebas de Rendimiento

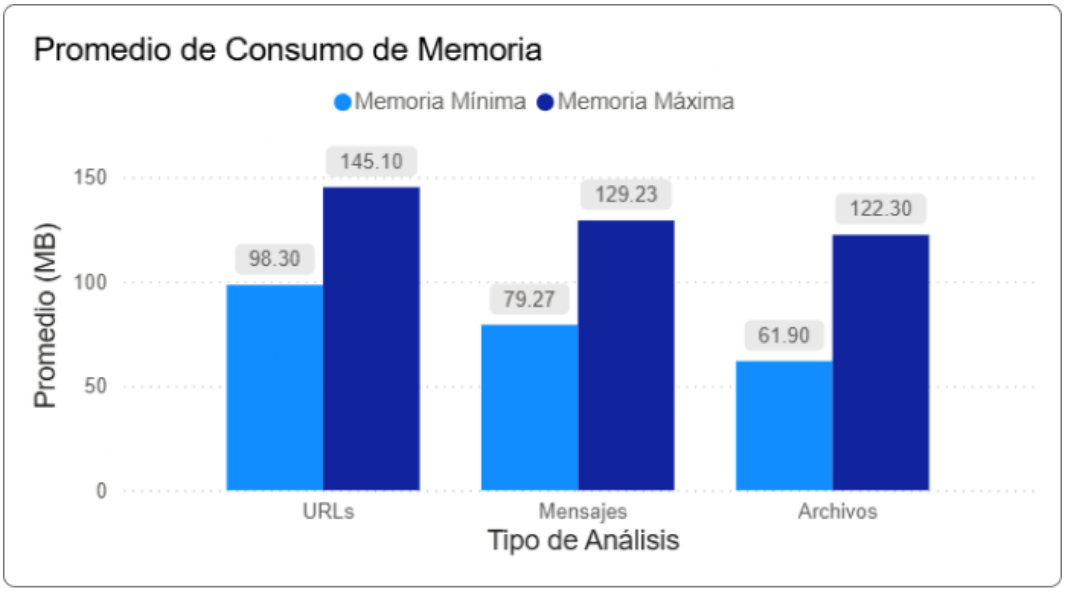
Ejecutadas las pruebas y medido el rendimiento, se registran los resultados en tablas, se obtienen los promedios y se diseñan cuatro gráficas que ponen en comparativa el consumo de recursos para las tres funcionalidades de análisis. Los resultados se pueden observar a mayor detalle en el enlace que se encuentra en el Apéndice M.

La Figura 88 muestra que el análisis de URLs es la funcionalidad que más memoria utiliza, consumiendo ~98 megabytes en su pico más bajo y ~145 en el más alto. Por su parte, el análisis de mensajes se posiciona en segundo lugar y en el último, el análisis de archivos.

A priori se podría intuir que el análisis de mensajes requiere de mayor memoria ya que su interfaz gráfica tiene mayores componentes que deben ser almacenados y renderizados, tanto en la pantalla de herramientas como en la de resultados. Sin embargo, estas pantallas son principalmente estáticas, lo que difiere en el caso del análisis de URLs, ya que la pantalla de herramientas, está constituida por un campo de entrada al que se aplica validación en tiempo real cada vez que el usuario escribe un carácter, lo que ocasiona el cambio de estado de la interfaz, especialmente cuando se detecta un error en el formato de la URL, por lo tanto, se requiere de más memoria para almacenar las nuevas visuales que también pueden incluir animaciones.

Figura 88.

Promedio del consumo de memoria del sistema front-end

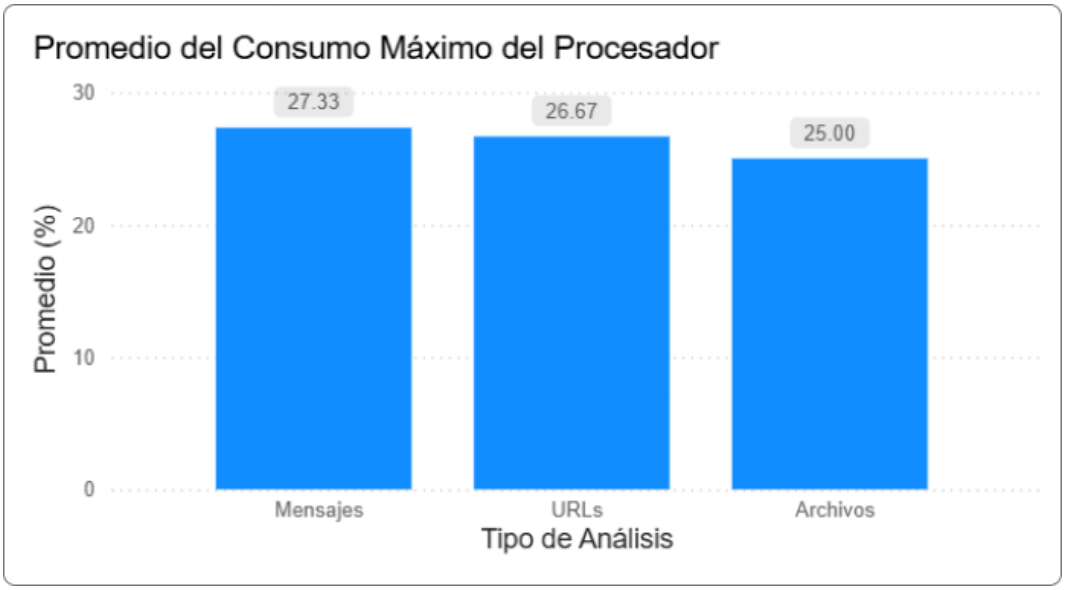


Fuente. Autoría propia.

Respecto al procesador, el análisis de mensajes es el que más lo utiliza, como se observa en la Figura 89. Sin embargo, este consumo no es significativo, teniendo apenas una diferencia del 2% en relación con el último, que es el análisis de archivos. Aun así, esto es comprensible ya que, al mostrar los resultados de un análisis de un mensaje, se deben dibujar más componentes en la interfaz gráfica del usuario, lo que incrementa el uso de la gráfica que está integrada en el procesador.

Figura 89.

Promedio del consumo máximo del procesador del sistema front-end



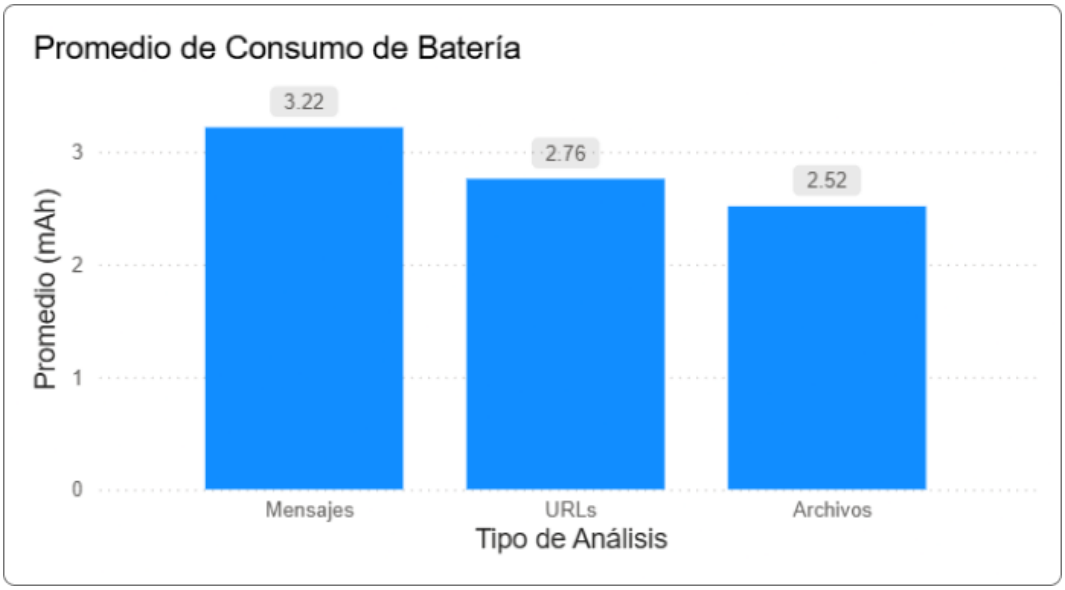
Fuente. Autoría propia.

Por otro lado, la Figura 90, expone una relación directa con el uso del procesador, ya que el análisis de mensajes es la funcionalidad que más batería consume, con un total de 3,22 miliamperios por hora, dejando en claro que, el procesador entre mayor uso, mayor es la energía que necesita.

Esta gráfica demuestra un uso eficiente de la batería en la funcionalidad que mayor procesamiento requiere, puesto que, en dos minutos de análisis, apenas se alcanza un gasto por hora que representa el ~0,08% de la capacidad de la batería, esto bajo un ambiente que requiere renderizar la interfaz. Lo que a su vez indica que la detección y análisis automatizado de mensajes de texto SMS, probablemente consuma un porcentaje mucho menor, al ser una tarea en segundo plano, cumpliendo así con el requerimiento no funcional número nueve.

Figura 90.

Promedio del consumo de batería del sistema front-end



Fuente. Autoría propia.

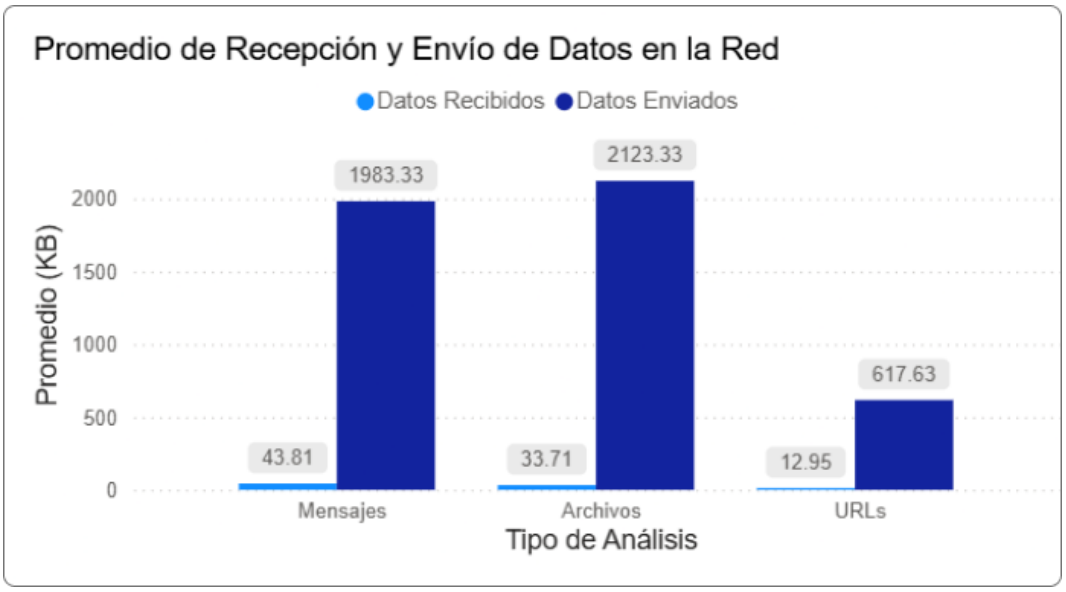
Finalmente, la Figura 91 expone el consumo de datos de red, un aspecto que se vuelve esencial cuando se habla del uso de datos móviles limitados. En primer lugar, está el análisis de archivos, el cual envío ~2123 kilobytes (KB) de datos y, en segundo lugar, el análisis de mensajes con ~1983 KB. Estos resultados están mayormente atados al tamaño de los archivos que se adjuntaron para ser analizados, por ello se observa una diferencia notable con los datos enviados en el análisis de URLs, en el que fue de apenas ~617 KB, sin contar que, en la primera repetición, el envío fue de ~1840 KB, lo que se consideraría como una anomalía ya que en las otras dos restantes alcanzo como máximo ~6,48 KB.

En cuanto a la recepción de datos, el tamaño se reduce de forma considerable, alcanzando difícilmente ~44 KB, lo que se debe al uso del formato JSON, usado por el sistema back-end para retornar la respuesta una vez que se inicia el análisis y, MessagePack, otro formato, pero

más eficiente que el anterior y que se utiliza para enviar actualizaciones de los análisis en tiempo real a la aplicación de Android.

Figura 91.

Promedio del consumo de red del sistema front-end



Fuente. Autoría propia.

Recursos Necesarios

En esta sección se exponen los posibles costos estimados de llevar el sistema a un entorno de producción y los gastos durante la ejecución de este proyecto.

Los costos de alojamiento del sistema son calculados en base a un escenario con una actividad de mil usuarios, cada uno realizando cinco análisis de correos electrónicos diarios conteniendo un archivo de 5 MiB. Además, considerando los resultados obtenidos en las pruebas de rendimiento, la API principal, el orquestador de análisis y la API de autenticación contarán con un núcleo virtual del procesador (vCPU) y un gibibyte (GiB) de memoria, que serían suficientes para abarcar las 5000 peticiones diarias. Los costos de los servicios con el proveedor de Google pueden ser apreciados a mayor detalle en el enlace que se encuentra en el Apéndice N.

Tabla 20.

Costos estimados para el sistema

Concepto	Servicio	Especificaciones	Costo (dólares estadounidenses)	Frecuencia
Alojamiento de la API principal	Google Cloud Run	1 vCPU. 1 GiB RAM.	1,61	Mensual
Alojamiento del orquestador de análisis	Google Cloud Run	1 vCPU. 1 GiB RAM.	1,61	Mensual
Alojamiento de la API de autenticación	Google Cloud Run	1 vCPU. 1 GiB RAM.	1,61	Mensual
Intermediario de mensajería	CloudAMQP	Sassy Squirrel - 1 node	50	Mensual
Base de datos	Supabase	Pro Tier	25	Mensual
Almacenamiento de archivos temporales	Google Cloud Storage	-	1,41	Mensual
Dominio de la API de análisis	Don Dominios	-	18,75	Anual
Cuota de registro de cuenta de Google Play Console	Play Store	-	25	Pagó de por vida
Total			124,99 dólares el primer mes 99,99 dólares al año 81,24 dólares al mes	

Fuente. Autoría propia.

Tabla 21.

Recursos necesitados para el proyecto

Recurso	Descripción	Presupuesto (COP)
Equipo Humano	NA	0
Equipos de Software	NA	0
Viajes y Salidas de Campo	NA	0
Materiales y Suministros	NA	0
Bibliografía	NA	0
Total		0

Fuente. Autoría propia.

Conclusiones

Se diseño y desarrollo con éxito un prototipo funcional capaz analizar contenido digital y ofrecer indicadores clave a los usuarios de dispositivos móviles de Android, usando servicios de análisis como VirusTotal, Filescan, Hybrid Analysis y Urlquery. Este prototipo demuestra la viabilidad de democratizar el acceso a herramientas de ciberseguridad avanzadas a usuarios no técnicos, proporcionando una capa de protección adicional contra amenazas como el phishing y el smishing.

La arquitectura implementada posibilito el desarrollo de dos sistemas orientados a cumplir con objetivos distintos, que, en conjunto, conformaron un prototipo altamente escalable, modular, eficiente y fácil de usar, lo que, en entornos de producción, sería crucial para optimizar los costos y asegurar la satisfacción del usuario.

El sistema back-end encargado de ofrecer las funcionalidades núcleo de análisis, demostró tener un rendimiento eficaz, eficiente y resiliente, capaz de gestionar un gran número de peticiones, gracias al intermediario de mensajería y, ser escalado sin que los recursos sean la preocupación principal. Asimismo, su desacoplamiento interno, especialmente en el orquestador de análisis, da lugar a que futuras mejoras favorezcan la detección de contenido malicioso y aborden debilidades desde diferentes áreas.

El sistema front-end se construyó a partir de una interfaz diseñada para ser intuitiva y simple, destacando la importancia de no solo analizar y mostrar resultados, sino que también de representarlos adecuadamente, incluyendo colores e iconos que apoyen la interpretación del usuario. Además, el desarrollo de características adicionales como la detección y análisis de mensajes de texto SMS, represento una oportunidad para pulir la experiencia de usuario, reduciendo las interacciones manuales y maximizando el tiempo, a la vez que se protege su

privacidad y decisiones. Respecto a las pruebas de rendimiento, se reflejó su bajo impacto en los recursos del sistema Android durante su ejecución, que incluso en las tareas y momentos más demandantes, utiliza una ínfima parte de la batería, consume pequeñas cantidades de red y su uso de memoria y CPU son mínimos.

La evaluación del sistema evidencio la eficacia para detectar archivos maliciosos, principalmente en formato PDF, que usualmente son adjuntados en correos electrónicos. También se destacó su precisión para determinar aquellos que son legítimos, lo que es fundamental para dar confiabilidad al usuario. Por su lado, la detección de URLs, aunque fue infalible identificando los enlaces maliciosos, si demostró falencias en la precisión al detectar un gran porcentaje de las muestras legitimas como maliciosas, exhibiendo la importancia de incluir procesos más complejos para determinar los veredictos en este caso en concreto.

Recomendaciones

El sistema back-end esta desarrollado con una arquitectura fundamentalmente modular, pero, en las pruebas de rendimiento, en especial al gestionar peticiones de análisis de mensajes de texto SMS y correos electrónicos, se encontró que la API principal puede representar un cuello de botella para el orquestador de análisis, por lo cual, se recomienda implementar un servicio interno encargado de recibir y manipular estas peticiones, esencialmente para extraer las URLs y actualizar los análisis en la base de datos. Esta nueva integración ayudaría a reducir la sobrecarga e incrementar la escalabilidad, posibilitando el levantamiento de nuevas instancias según la funcionalidad que lo requiera.

De igual manera, el orquestador de análisis en el sistema back-end, integra varios servicios externos que tienen un uso limitado como VirusTotal e Hybrid Analysis, que para el entorno de desarrollo y el prototipo actual son suficientes, pero, en un escenario de real, estos límites fácilmente podrían alcanzarse. Por ello, lo más adecuado sería solicitar la eliminación o ampliación de los límites, aplicando a un proceso de registro en el cual se realiza una investigación de antecedentes al solicitante. De manera alternativa, también se podrían integrar otros servicios, como, por ejemplo, ClamAV y CAPESandbox.

El sistema front-end cuenta con un desarrollo muy sólido que aún contiene potenciales mejoras principalmente en la experiencia de usuario, entre las que se encuentra la automatización de análisis de correos electrónicos, la cual sería una característica que se podría implementar para detectar mensajes entrantes y notificar al usuario de un análisis disponible, cumpliendo así con un beneficio que no sería intrusivo. Por otra parte, la pantalla de configuración tiene un diseño básico y ofrece opciones limitadas, por lo que se recomienda incluir nuevos parámetros

para satisfacer las diversas necesidades de los usuarios, como el cambio de contraseña, eliminación de la cuenta, ajustes de privacidad, entre otros.

Otra área fundamental es la autenticación. Actualmente el prototipo cuenta con una API que ofrece el mecanismo más básico y tradicional, a través de usuario y contraseña, sin embargo, hoy en día existen otros métodos que son mucho más seguros y rápidos, como los múltiples factores de autenticación (MFA) y OAuth 2.0. Por lo tanto, integrar un nuevo servicio de autenticación más avanzado sería crucial para un entorno de producción, que no solo ofrezca funcionalidades de seguridad digital a los usuarios, sino que también, proteja sus credenciales.

Por otro lado, las pruebas utilizadas en la evaluación de detección son visiblemente limitadas al contar con pocas muestras de archivos y URLs. La aplicación de un sistema de evaluación a mayor escala y más elaborado, permitiría encontrar nuevas áreas de mejora en el sistema back-end. Para ello, lo ideal sería incluir URLs de sitios web maliciosos conocidos como ataques *zero-day*, que pueden ser creados con kits de phishing y utilizar procedimientos enfocados a eludir la detección, lo que daría mayor valor a los resultados. De la misma manera con los archivos, en este caso, el factor principal seria la diversidad, incluyendo formatos tanto conocidos como exóticos.

El trabajo a futuro también podría orientarse a implementar procesos de análisis utilizando modelos de aprendizaje automático (ML), que emplean ingeniosamente el texto de los mensajes, contenido propio de los sitios web o incluso de las URLs, para extraer características y detectar patrones sospechosos asociados al phishing. El orquestador de análisis, gracias a su ya mencionado desacoplamiento, facilitaría la inclusión de este nuevo servicio.

Igualmente, los modelos de ML al ser tan versátiles con las entradas de datos posibilitan su integración en una cadena de procesamiento que reciba los resultados de los análisis de los

servicios externos y, los combine sabiamente para proveer un veredicto que sea mucho más realista y acertado, lo que, en definitiva, beneficiaría la precisión del sistema back-end.

En el área del sistema front-end, actualmente se ofrece una aplicación móvil para Android, pero, la arquitectura del prototipó, favorece y potencia la inclusión de otras aplicaciones para sistemas operativos como iOS, e incluso, para diferentes plataformas de escritorio como Windows. Además, la naturaleza del proyecto no se enfoca en distinguir usuarios, sino que abordar en su mayor posibilidad, las preocupaciones de seguridad, por ello, la extensión a nuevos dispositivos sería uno de los objetivos principales que se podrían considerar a futuro.

Por último, hay que entender la inmensidad del mundo digital, que inherentemente trae consigo otras amenazas que no se han tenido en cuenta, como el quishing, que pertenece a la familia del phishing, pero en este caso utiliza códigos QR. Esto propone la expansión del sistema para aprovechar lo ya construido y emplearlo en nuevas metodologías de ataque.

Bibliografía

Abuadbba, A., Wang, S., Almashor, M., Ahmed, M. E., Gaire, R., Camtepe, S., & Nepal, S. (Abril de 2022). Towards Web Phishing Detection Limitations and Mitigation. doi:https://doi.org/10.48550/arXiv.2204.00985

Aljofey, A., Jiang, Q., Rasool, A., Chen, H., Liu, W., Qu, Q., & Wang, Y. (2022, Mayo 25). An effective detection approach. *Scientific Reports*. doi:https://doi-org.bibliotecavirtual.unad.edu.co/10.1038/s41598-022-10841-5

Anti-Phishing Working Group. (s.f.). *APWG*. Obtenido de Phishing Activity Trends Reports: https://apwg.org/trendsreports/

AWS. (s.f.). *What is SMS?* Obtenido de AWS: https://aws.amazon.com/what-is/sms/

AWS. (s.f.). *What's the Difference Between Frontend and Backend in Application Development?* Obtenido de AWS: https://aws.amazon.com/compare/the-difference-between-frontend-and-backend/

Baker, E., & Cartier, M. (2025). *HoxHunt*. Obtenido de Phishing Trends Report: https://hoxhunt.com/guide/phishing-trends-report#report-methodology-ampnbspkey-terms

Baker, K. (17 de Abril de 2023). *Malware Analysis.* Obtenido de Crowd Strike: https://www.crowdstrike.com/en-us/cybersecurity-101/malware/malware-analysis/

Bonnie, E. (Enero de 3 de 2025). *110+ of the Latest Data Breach Statistics [Updated 2025]*. Obtenido de Secureframe: https://secureframe.com/blog/data-breach-statistics

Brahim Belhaouari, S., Karim, A., & Shahroz, M. (2023). Phishing Detection System through Hybrid Machine Learning Based on URL. *IEEE Access*. doi:http://dx.doi.org/10.1109/ACCESS.2023.3252366

Chacon, S., & Straub, B. (2014). What is Git. En S. Chacon, & B. Straub, *Pro Git* (págs. 14-18). Apress. Obtenido de https://git-scm.com/book/en/v2/Getting-Started-What-is-Git%3F

Choo, E., Nabeel, M., De Silva, R., Yu, T., & Khalil, I. (26 de Mayo de 2022). A Large Scale Study and Classification of VirusTotal Reports on Phishing and Malware URLs.

Cibersecurity & Infrastructure Security Agency. (s.f.). *VirusTotal*. Obtenido de Cibersecurity & Infrastructure Security Agency: https://www.cisa.gov/resources-tools/services/virustotal

Colombia. (2008). *Ley 1266 de 2008.* Obtenido de Función Pública: https://www.funcionpublica.gov.co/eva/gestornormativo/norma.php?i=34488#3.k

Colombia. (2009). *Ley 1273 de 2009.* Obtenido de Funci{on P{ublica: https://www.funcionpublica.gov.co/eva/gestornormativo/norma.php?i=34492

Colombia. (2012). *Ley 1581 de 2012.* Obtenido de Función Pública: https://www.funcionpublica.gov.co/eva/gestornormativo/norma.php?i=49981

Colombia. (2013). *Decreto 1377 de 2013.* Obtenido de Funcin P{ublica: https://www.funcionpublica.gov.co/eva/gestornormativo/norma.php?i=53646

Derechos Humanos. (21 de Noviembre de 2018). *Artículo 12: derecho a la intimidad.* Obtenido de Noticias ONU: https://news.un.org/es/story/2018/11/1446671

Docker. (s.f.). *Dockerfile overview*. Obtenido de Docker Docs: https://docs.docker.com/build/concepts/dockerfile/

Docker. (s.f.). *How compose works*. Obtenido de Docker Docs: https://docs.docker.com/compose/intro/compose-application-model/

Docker. (n.d.). *What is Docker?* Retrieved from Docker Docs: https://docs.docker.com/get-started/docker-overview/

Figma. (2019). *What is Figma?* Obtenido de Figma Learn: https://help.figma.com/hc/en-us/articles/14563969806359-What-is-Figma

Figueroa, N. (14 de Junio de 2024). *Estadísticas sobre phishing: las últimas que debe conocer en 2024*. Obtenido de Mobile Data Solutions: https://www.mds.pe/2024/06/14/estadisticas-sobre-phishing-las-ultimas-que-debe-conocer-en-2024/

GitHub. (s.f.). *About GitHub and Git*. Obtenido de GitHub Docs: https://docs.github.com/en/get-started/start-your-journey/about-github-and-git#about-github

Google. (1 de Marzo de 2023). *Kotlin overview*. Obtenido de Android Developers: https://developer.android.com/kotlin/overview

Google. (27 de Junio de 2024). *Android's Kotlin-first approach*. Obtenido de Android Developers: https://developer.android.com/kotlin/first

Google. (19 de Septiembre de 2024). *Why adopt Compose*. Obtenido de Android Developers: https://developer.android.com/develop/ui/compose/why-adopt#intuitive

Google. (29 de Septiembre de 2025). *Android Debug Bridge (adb)*. Obtenido de Android Developers: https://developer.android.com/tools/adb

Google. (10 de Febrero de 2025). *Build an offline-first app*. Obtenido de Android Developers: https://developer.android.com/topic/architecture/data-layer/offline-first

Google. (10 de Febrero de 2025). *Guide to app architecture*. Obtenido de Android Developers: https://developer.android.com/topic/architecture#architecture-benefits

Google. (20 de Mayo de 2025). *Meet Google Play's target API level requirement*. Obtenido de Android Developers: https://developer.android.com/google/play/requirements/target-sdk

Google. (10 de Febrero de 2025). *UI layer*. Obtenido de Android Developers: https://developer.android.com/topic/architecture/ui-layer

Google. (s.f.). *Build apps or websites quickly on a fully managed platform*. Obtenido de Google Cloud: https://cloud.google.com/run

Grafana. (22 de Septiembre de 2025). *Grafana k6*. Obtenido de Grafana: https://grafana.com/docs/k6/latest/

Grafana. (s.f.). *Types of load testing*. Obtenido de Grafana: https://grafana.com/load-testing/types-of-load-testing/

Guven, M. (Septiembre de 2024). Dynamic Malware Analysis Using a Sandbox Environment, Network Traffic Logs, and Artificial Intelligence. *International Journal of Computational and Experimental Science and Engineering*, 1. doi:10.22399/ijcesen.460

IBM. (15 de Octubre de 2021). *What is PostgreSQL?* Obtenido de IBM: https://www.ibm.com/think/topics/postgresql

Khalil, M. (29 de Abril de 2025). *Phishing Statistics 2025: AI-Driven Attacks, Costs & Trends*. Obtenido de DeepStrike: https://deepstrike.io/blog/Phishing-Statistics-2025

Köhler, D., Pünter, W., & Meinel, C. (2024). We have Phishing at Home: Quantitative Study on Email Phishing Susceptibility in Private Contexts. En N. Mouha, & N. Nikiforakis (Ed.), *Information Security: ISC 2024* (págs. 246-265). Springer Cham. doi:https://doi.org/10.1007/978-3-031-75764-8_13

Kosinski, M. (10 de Junio de 2024). *What is smishing (SMS phishing)?* Obtenido de IBM: https://www.ibm.com/think/topics/smishing

Kumar, N. (10 de Septiembre de 2025). *Android Usage Statistics (2025) – Global Market Share*. Obtenido de Demandsage: https://www.demandsage.com/android-statistics/

Lamothe, M., Guéhéneuc, Y.-G., & Shang, W. (Octubre de 2021). A Systematic Review of API Evolution Literature. *ACM Computing Surveys*, 2. doi:10.1145/3470133

Lenaerts-Bergmans, B. (11 de Septiembre de 2023). *What is Cybersecurity Sandboxing?* Obtenido de Crowd Strike: https://www.crowdstrike.com/en-us/cybersecurity-101/threat-intelligence/cybersecurity-sandboxing/

MDN. (14 de Marzo de 2025). *An overview of HTTP*. Obtenido de MDN: https://developer.mozilla.org/en-US/docs/Web/HTTP/Guides/Overview

MDN. (18 de Febrero de 2025). *What is a URL?* Obtenido de MDN Web Docs: https://developer.mozilla.org/en-US/docs/Learn_web_development/Howto/Web_mechanics/What_is_a_URL

Microsoft. (2024, Enero 10). *Introduction to .NET*. Retrieved from Microsoft Learn: https://learn.microsoft.com/en-us/dotnet/core/introduction?WT.mc_id=dotnet-35129-website

Mishra, S., & Soni, D. (2021). DSmishSMS-A System to Detect Smishing SMS. Obtenido de https://link.springer.com/article/10.1007/s00521-021-06305-y

Moore, M. (16 de Septiembre de 2024). *Choosing Your Champion: Python vs C#*. Obtenido de Shakuro: https://shakuro.com/blog/python-vs-c-sharp

Murel, J., & Kavlakoglu, E. (19 de Enero de 2024). *¿Qué es una matriz de confusión?* Obtenido de IBM: https://www.ibm.com/es-es/think/topics/confusion-matrix

Naciones Unidas. (s.f.). *Pacto Internacional de Derechos Civiles y Políticos.* Obtenido de Pacto Internacional de Derechos Civiles y Políticos: https://www.ohchr.org/es/instruments-mechanisms/instruments/international-covenant-civil-and-political-rights

Organisation for Economic Co-operation and Development. (23 de Septiembre de 1980). DIRECTRICES DE LA OCDE QUE REGULAN LA PROTECCIÓN DE LA.

Patterson, C. (s.f.). *RabbitMQ Transport*. Obtenido de MassTransit: https://masstransit.io/documentation/transports/rabbitmq

Policía Nacional de Colombia. (s.f.). *CAI Virtual*. Obtenido de Observatorio del Cibercrimen: https://caivirtual.policia.gov.co/observatorio/analisis-cibercrimen

PostgreSQL. (s.f.). *About*. Obtenido de PostgreSQL: https://www.postgresql.org/about/

Rahman, M. L., & Timko, D. (2022). Users Really Do Respond To Smishing. Obtenido de http://dx.doi.org/10.48550/arXiv.2212.13312

Rao, R. S., Kondaiah, C., Pais, A. R., & Lee, B. (15 de Mayo de 2025). A hybrid super learner ensemble for phishing detection on mobile devices. *Scientific Reports*. Obtenido de https://doi-org.bibliotecavirtual.unad.edu.co/10.1038/s41598-025-02009-8

Red Hat. (8 de Mayo de 2020). *What is a REST API?* Obtenido de Red Hat IT Topics: https://www.redhat.com/en/topics/api/what-is-a-rest-api

Redis. (s.f.). *Domain-Driven Design (DDD)*. Obtenido de Redis: https://redis.io/glossary/domain-driven-design-ddd/

Smith, S. (2023). Clean architecture. En S. Smith, *Architecting Modern Web Applications with ASP.NET Core and Azure* (págs. 23-30). Microsoft. Obtenido de https://dotnet.microsoft.com/en-us/download/e-book/aspnet/pdf

Statscounter. (s.f.). *Desktop vs Mobile vs Tablet Market Share Worldwide*. Obtenido de Statscounter: https://gs.statcounter.com/platform-market-share/desktop-mobile-tablet/worldwide/#monthly-202408-202508-bar

Sufiyan, T. (19 de Junio de 2025). *Node.js Overview: What is Node.js and Why It Matters*. Obtenido de Simplilearn: https://www.simplilearn.com/tutorials/nodejs-tutorial/what-is-nodejs

Tabassum, S., Faklaris, C., & Lipford, H. R. (2024). What Drives SMiShing Susceptibility? A U.S. Interview Study of How and Why Mobile Phone Users Judge Text Messages to be Real or Fake. 12-13.

Thomas, J. (2018). Individual Cyber Security: Empowering Employees to Resist Spear Phishing to Prevent Identity Theft and Ransomware Attacks. *International Journal of Business and Management*, 15-16. doi:10.5539/ijbm.v13n6p1

Thorpe, C., & Phadke, P. (2021). Analysis of API Driven Application to Detect Smishing Attacks. Obtenido de https://www.researchgate.net/publication/350695594_Analysis_of_API_Driven_Application_to_Detect_Smishing_Attacks

Trujillo, F. H., Gamarra, A. C., Saldaña, R. G., & De Los Santos, A. M. (17 de Diciembre de 2024). Implementación de un sistema antimalware inteligente para detección de enlaces maliciosos en códigos QR. *Ingeniería Investiga, 6*, 7-12. doi:http://dx.doi.org/10.47796/ing.v6i00.1078

Wagner, B. (21 de Marzo de 2025). *A tour of the C# language*. Obtenido de Microsoft Learn: https://learn.microsoft.com/en-us/dotnet/csharp/tour-of-csharp/overview

Wahsheh, H. A., & Zahrani, M. S. (2022). Lightweight Cryptographic and Artificial Intelligence Models for Anti-smishing. Obtenido de http://dx.doi.org/10.1007/978-3-030-85990-9_39

Zhang, Z., Wu, J., Lu, N., Shi, W., & Liu, Z. (29 de Abril de 2025). AdaptPUD: An accurate URL-based detection approach against tailored deceptive phishing websites. *Computer Networks*. doi:https://doi.org/10.1016/j.comnet.2025.111303

Zieni, R., Massari, L., & Calzarossa, M. C. (Enero de 2023). Phishing or Not Phishing? A Survey on the Detection of Phishing Websites. *IEEE Access*. doi:10.1109/ACCESS.2023.3247135

Apéndices

Apéndice A

Enlace de los diagramas empleados en la sección de análisis y diseño

Diagramas empleados en la sección de análisis y diseño.

Apéndice B

Enlace del diagrama del esquema de la base de datos

Diagrama del esquema de la base de datos.

Apéndice C

Enlace de los diseños interactivos de la interfaz gráfica del usuario

Prototipo navegable del sistema front-end.

Apéndice D

Borrador de políticas de privacidad

El siguiente texto es un borrador generado con inteligencia artificial con fines académicos. Para una aplicación en producción, este documento debe ser extendido, revisado y aprobado por un abogado profesional, y posteriormente, ser incluido en el sistema para requerir consentimiento explícito del usuario.

Política de Privacidad de Openlysis

Fecha de última actualización: 5 de octubre de 2025

1. Introducción

La presente Política de Privacidad describe cómo Openlysis (en adelante, "el Servicio"), un prototipo desarrollado en el marco de un proyecto académico recopila, utiliza y protege su información. Al utilizar nuestro Servicio, usted acepta las prácticas descritas en este documento.

2. Responsable del Tratamiento de Datos

El responsable del tratamiento de sus datos personales es **[Nombre de la Empresa/Persona Responsable]**, en cumplimiento de la Ley 1581 de 2012.

3. Información que Recopilamos

A. **Datos de Cuenta:** Para crear una cuenta, recopilamos su dirección de correo electrónico. Su contraseña nunca se almacena en texto plano; en su lugar, se

protege mediante el algoritmo de hash Argon2id antes de ser almacenada en nuestra base de datos.

B. **Contenido de Análisis:** Cuando usted solicita un análisis de un SMS, correo electrónico o archivo, el Servicio procesa y almacena el contenido completo del mensaje (remitente, asunto y cuerpo), así como los indicadores extraídos (URLs y archivos). Este contenido se almacena con el único propósito de proveerle a usted un historial contextualizado de sus análisis dentro de la aplicación.

4. Uso de la Información

Utilizamos su información exclusivamente para:

- Proveer, mantener y mejorar el servicio.
- Permitirle acceder a su cuenta y a su historial de análisis contextualizado.
- Realizar los análisis de seguridad que usted solicite.

5. Cómo Compartimos tu Información

Reafirmamos que su privacidad es fundamental. El contenido personal y contextual de sus mensajes (remitente, asunto, cuerpo) nunca abandona nuestros servidores. Para realizar los análisis, compartimos única y exclusivamente los siguientes datos extraídos con servicios de análisis externos de confianza (VirusTotal, Filescan, etc.):

- URLs (enlaces web) encontradas en el contenido.
- Archivos que usted elija analizar.

6. Seguridad de los Datos (Ley 1273 de 2009)

Tomamos medidas de seguridad para proteger su información. Toda la comunicación entre la aplicación y nuestros servidores se realiza a través del protocolo HTTPS. Las contraseñas se protegen con algoritmos de hash modernos y el acceso a la base de datos está restringido para prevenir el acceso no autorizado.

7. Sus Derechos (Habeas Data - Ley 1581 de 2012)

De acuerdo con la ley colombiana, usted tiene derecho a conocer, actualizar, rectificar y solicitar la supresión de sus datos personales. Para ejercer estos derechos, puede contactarnos a través de **[Correo Electrónico de Contacto de Openlysis]**.

8. Retención de Datos

Sus datos personales y de análisis se conservarán de forma indefinida mientras usted mantenga una cuenta activa en el Servicio, para garantizar el acceso ininterrumpido a su historial. Si elimina su cuenta, se procederá a la supresión de todos sus datos asociados.

Apéndice E

Enlace de los diagramas empleados en la sección de desarrollo

Diagramas empleados en la sección de desarrollo.

Apéndice F

Enlace del repositorio de GitHub con el código fuente del sistema back-end

Repositorio de GitHub con el código del sistema back end.

Apéndice G

Enlace del repositorio de GitHub con el código fuente del sistema front-end

Repositorio de GitHub con el código de la aplicación de Android.

Apéndice H

Enlace del conjunto de datos con archivos legítimos

Conjunto de datos con archivos legítimos y maliciosos.

Apéndice I

Enlace del programa de Python utilizado para generar archivos maliciosos

Programa de Python que genera archivos PDF maliciosos.

Apéndice J

Enlace del conjunto de datos con muestras de URLs legítimas y maliciosas

Conjunto de datos con URLs legítimas y maliciosas.

Apéndice K

Enlace de los resultados de la evaluación de detección de URLs

Muestras y resultados de evaluación de análisis de URLs legítimas y maliciosas.

Apéndice L

Enlace de los resultados de rendimiento del sistema back-end

Resultados de las pruebas de carga y rendimiento en el sistema back-end.

Apéndice M

Enlace de los resultados de rendimiento del sistema front-end

Resultados de las pruebas de rendimiento en el sistema front-end.

Apéndice N

Enlace de los costos estimados en un entorno de producción

Costos estimados de la API principal, el orquestador de análisis, la API de autenticación y el almacenamiento de archivos temporales.

Apéndice O

Vídeo de guía del uso básico de la aplicación de Android

Guía básica del usuario – Video.