Universidad Internacional de La Rioja

Escuela Superior de Ingeniería y Tecnología

Máster Universitario en Ciberseguridad

Sistema On-Premise de detección de phishing en tiempo real utilizando LLMs

Trabajo fin de estudio presentado por:	Andrés Camilo Gutiérrez Escobar Mónica Patricia Moreno Moreno
Tipo de trabajo:	Desarrollo de software
Director/a:	María Teresa Pérez Morales
Fecha:	5 de febrero de 2025

Resumen

Este proyecto tiene como objetivo desarrollar un sistema on-premise de monitoreo y detección de phishing en tiempo real mediante el uso de Modelos de Lenguaje Grandes (LLMs) de código abierto, el propósito de este sistema es superar las limitaciones de las soluciones tradicionales basadas en listas negras y heurísticas estáticas, las cuales no logran adaptarse eficazmente a las nuevas variantes de phishing, y para su desarrollo, la metodología incluye la integración de técnicas avanzadas como Fine-Tuning, Prompt Engineering, y Retrieval-Augmented Generation (RAG) para ajustar los LLMs a las características específicas de los ataques de phishing, además este sistema permite analizar correos electrónicos en tiempo real sin exponer los datos a la nube, mejorando así la privacidad y seguridad corporativa, permitiendo de esta manera un aumento en la precisión y la capacidad del sistema para adaptarse a las amenazas emergentes, manteniendo el control total sobre la información crítica de la organización.

Palabras clave: Phishing, LLM, NLP, ciberseguridad y monitoreo

Abstract

This project aims to develop an on-premise real-time phishing monitoring and detection system through the use of open source Large Language Models (LLMs), the purpose of this system is to overcome the limitations of traditional solutions based on blacklists and static heuristics, which fail to effectively adapt to new phishing variants, and for its development, the methodology includes the integration of advanced techniques such as Fine-Tuning, Prompt Engineering, and Retrieval-Augmented Generation (RAG) to adjust the LLMs to the specific characteristics of phishing attacks, in addition this system allows to analyze emails in real time without exposing the data to the cloud, thus improving corporate privacy and security, allowing in this way an increase in the precision and capacity of the system to adapt to emerging threats, while maintaining total control over the critical information of the organization.

Keywords: Phishing, LLM, NLP, cybersecurity y monitoring.

Índice de contenidos

1. Introducción 1
1.1. Motivación 2
1.2. Planteamiento del problema 3
1.3. Estructura del trabajo 3
2. Estado del arte 6
2.1. Phishing 6
2.1.1. Evolución del phishing 10
2.1.2. Técnicas de phishing 11
2.1.3. Clasificación del phishing 12
2.1.4. Seguridad de Correo Electrónico 12
2.1.5. Herramientas de análisis 13
2.1.6. Impacto global del phishing 14
2.2. Enfoques tradicionales: Listas negras y heurísticas 15
2.2.1. Listas negras 15
2.2.2. Heurísticas basadas en reglas 16
2.3. Evolución hacia modelos de Machine Learning 17
2.3.1. Machine learning tradicional 17
2.3.2. Modelos de redes neuronales recurrentes (RNN) y LSTM 18
2.3.3. Redes neuronales convolucionales (CNN) 19
2.4. Inteligencia artificial y machine learning 20
2.4.1. Historia de la Inteligencia artificial 20
2.4.2. Historia del machine learning 20
2.4.3. Evolución y utilidades de la IA y el ML 21
2.4.4. Impacto en la Ciberseguridad 21

2.5. Proyectos UNIR relacionados con Phishing 23
2.6. LLMs y su impacto en la detección de Phishing 24
2.6.1. Modelos basados en procesamiento de lenguaje natural (NLP) 25
2.6.2. Modelos de lenguaje de gran tamaño (LLM) 26
2.6.3. Comparativas de los modelos LLMs 27
2.6.4. Aplicación de LLMs en la detección de Phishing 30
2.6.5. Soluciones comerciales basadas en LLMs 31
2.7. Detección de Phishing con aprendizaje profundo 32
2.7.1. URLNet 32
2.7.2. Fine-Tuning en modelos de lenguaje 32
2.7.3. Retrieval-Augmented Generation (RAG) 33
2.7.4. Modelos Multimodales 33
2.7.5. Eficiencia de métodos avanzados 34
2.8. Implementaciones On-Premise de LLMs 35
2.8.1. Ventajas de los modelos LLMs en ambientes On-Premise 36
2.8.2. Aplicaciones on-Premise basadas en LLMs 39
2.8.3. Nuevos modelos de LLMs 40
2.8.4. Comparativas de modelos LLMs para entornos On-Premise 41
2.8.5. Desafíos de los LLMs On-Premise 43
2.9. Conclusiones del estado del arte 46
3. Objetivos concretos y metodología de trabajo 48
3.1. Objetivo general 48
3.2. Objetivos específicos 48
3.3. Metodología del trabajo 49
4. Desarrollo específico de la contribución 51

4.1. Desarrollo de software 51
4.1.1. Criterios de diseño y descripción del producto 51
4.1.2. Identificación de requisitos 52
4.1.3. Descripción de la herramienta software desarrollada 56
4.1.4. Detalles del proceso de desarrollo 75
4.1.5. Evaluación 97
4.2. Análisis de resultados obtenidos 100
4.2.1. Objetivo 1: Análisis de tecnologías actuales de detección de phishing 101
4.2.2. Objetivo 2: Selección y ajuste del modelo LLM 101
4.2.3. Objetivo 3: Desarrollo del sistema de detección de phishing funcional 102
4.2.4. Objetivo 4: Evaluación del rendimiento del sistema 102
4.2.5. Objetivo 5: Configuración del sistema para respuestas automáticas 103
4.3. Contribución 104
4.3.1. Contribución tecnológica 104
4.3.2. Contribución a la ciberseguridad corporativa 104
4.3.3. Contribución académica y de investigación 105
4.3.4. Contribución a la Industria 105
4.3.5. Contribución al cumplimiento normativo y seguridad de los datos 105
4.3.6. Impacto social y concienciación 106
5. Conclusiones y trabajo futuro 107
5.1. Conclusiones 107
5.2. Trabajos futuros 110
Referencias bibliográficas 112
Anexo A. 116
Manual de usuario: Sistema On-premise de detección de phishing 116

Manual de desarrollo: Sistema On-premise de detección de phishing.....128

Índice de figuras

Figura 1 *Incremento Anual de Ataques de Phishing Observados (2020-2022)* 8

Figura 2 *Distribución de ataques de Phishing por sector industrial en 2023* 10

Figura 3 *Comparativa de modelos de lenguaje de gran tamaño (LLMs) por capacidades y parámetros (2024)* 29

Figura 4 *Comportamiento de los modelos en términos de velocidad de generación de tokens* 42

Figura 5 *Comportamiento de los modelos en tiempo hasta el primer token (TTFT)* 43

Figura 6 *Modelo Entidad-Relación (ER)* 64

Figura 7 *Interfaz Python* 70

Figura 8 *SodpIA - Plataforma de gestión, configuración y reporte* 72

Figura 9 *Entorno de Anaconda* 75

Figura 10 *Página principal para la instalación de Ollama* 76

Figura 11 *Instalación de los LLMs en Ollama* 76

Figura 12 *Servidor Windows Server 2022* 77

Figura 13 *Configuración de hMailServer* 77

Figura 14 *Conexión de Thunderbird con el servidor de correo hmailServer* 78

Figura 15 *Prueba del entorno local - Servidor de correo* 78

Figura 16 *Características de CPU y RAM* 79

Figura 17 *Características de la GPU* 80

Figura 18 *Código fuente de la aplicación – Parte 1* 83

Figura 19 *Código fuente de la aplicación – Parte 2* 83

Figura 20 *Código fuente de la aplicación – Parte 3* 84

Figura 21 *Código fuente de la aplicación – Parte 4* 84

Figura 22 *Código fuente de la aplicación – Parte 5* 85

Figura 23 *Reporte de estados de los correos* 86

Figura 24 *Reporte de los patrones de phishing* 86

Figura 25 *Reporte de acciones ejecutadas* 87

Figura 26 *Reporte generado en Excel* 87

Figura 27 *Proceso de análisis del correo* 89

Figura 28 *Ajuste de Código fuente para la lectura de email de la base de datos Corpus* 92

Índice de tablas

Tabla 1 *Organización del trabajo en grupo* XI

Tabla 2 *Comparativa de modelos LLM* 28

Tabla 3 *Eficiencia de métodos avanzados* 35

Tabla 4 *Actores y sus niveles de confianza* 57

Tabla 5 *Activos de información* 58

Tabla 6 *Matriz de control de acceso a los datos* 59

Tabla 7 *Puntos de entrada de la aplicación* 60

Tabla 8 *Puntos de salida de la aplicación* 61

Tabla 9 *Rendimiento por modelo LLM* 93

Tabla 10 *Tiempos de respuesta promedio de los modelos LLMs* 94

Tabla 11 Desempeño por tipo de clasificación 95

Organización del trabajo en grupo

En este apartado se detallarán las distintas partes en las que se ha dividido el trabajo entre los componentes del grupo, los objetivos perseguidos en cada una de ellas y los mecanismos de coordinación empleados.

Partes que aborda el TFE, distribución y estructura de la memoria

A continuación, se detallan los elementos/partes que, individualmente, cada estudiante debe abordar, lo que podrían suponer un TFE por sí mismas.

Tabla 1

Organización del trabajo en grupo

Organización del trabajo en grupo - Desarrollo de la memoria	
Apartado de la memoria	**Responsables**
Introducción	Mónica Patricia Moreno Moreno Andrés Camilo Gutiérrez Escobar
Estado del arte	Mónica Patricia Moreno Moreno Andrés Camilo Gutiérrez Escobar
Objetivos y metodología de trabajo	Mónica Patricia Moreno Moreno Andrés Camilo Gutiérrez Escobar
Desarrollo específico de la contribución: Diseño del modelo de datos y la programación de adquisición de datos del servidor de correo.	Mónica Patricia Moreno Moreno
Desarrollo específico de la contribución: Elección del LLM, desarrollo del ajuste del modelo, la conexión con el sistema y la generación de reportes.	Andrés Camilo Gutiérrez Escobar
Desarrollo específico de la contribución: parte 4	Mónica Patricia Moreno Moreno Andrés Camilo Gutiérrez Escobar
Conclusiones y trabajo futuro	Mónica Patricia Moreno Moreno Andrés Camilo Gutiérrez Escobar

Nota. Fuente elaboración propia.

Objetivo del TFE desde el punto de vista de la adquisición de conocimientos

Este trabajo fin de estudios (TFE) tiene como objetivo no solo la aplicación de conocimientos previos, sino también la adquisición de nuevos saberes y competencias multidisciplinarias, destacándose por la integración de varios segmentos diferenciados desde el punto de vista de la adquisición de conocimientos, este TFE involucra 2 grandes áreas que son:

- *IA y modelos de lenguaje grandes (LLMs):* Se profundiza sobre la arquitectura, funcionamiento, y ajuste de los LLMs para tareas específicas, para el caso del proyecto seria la detección de phishing, lo cual implica un conocimiento profundo de técnicas avanzadas de procesamiento de lenguaje natural (NLP), como fine-tuning, prompt engineering, y retrieval-augmented generation (RAG).
- *Ciberseguridad aplicada a la detección de phishing en entornos on-premise:* Se profundiza en los conceptos de ciberseguridad, específicamente en las nuevas metodologías basadas en machine learning, por tanto se analizan las vulnerabilidades y las tácticas de phishing más avanzadas, además de ello la implementación de soluciones tecnológicas que funcionen dentro de la infraestructura de la organización, lo que implica no solo competencias técnicas, sino también la capacidad de asegurar que el sistema cumpla con los requisitos de privacidad y seguridad corporativa.

Por lo cual, desde el punto de vista de la adquisición de conocimientos, este TFE, incorpora varios segmentos suficientemente diferenciados, se puede observar la multidisciplinariedad de este TFE radica en la combinación de áreas como la inteligencia artificial, ciberseguridad, análisis de datos y tecnologías de implementación on-premise, integrando estos campos, se adquiere una visión global y práctica del desarrollo de sistemas de seguridad avanzados, además de la capacidad de aplicar estos conocimientos en situaciones reales dentro de su entorno profesional.

Mecanismos de coordinación empleados

En este proyecto, se ha adoptado una serie de mecanismos de coordinación y herramientas que nos permiten trabajar de manera eficiente y garantizar la cohesión en el desarrollo del trabajo, estos mecanismos son los siguientes:

- Gestión del proyecto y distribución de tareas: Al inicio del proyecto, se realizó una planificación detallada con la distribución de tareas basada en las competencias y fortalezas de cada uno de los integrantes, de esta manera se asignan las responsabilidades para poder hacer seguimiento de los progresos.
- Reuniones periódicas: Para mantener una comunicación fluida y asegurar el desarrollo de las actividades planificadas y que los resultados estén alineados con los objetivos, se estableció un calendario de reuniones semanales, las cuales son fundamentales para revisar avances, discutir problemas y ajustar el cronograma.
- Comunicación instantánea: Para resolver cuestiones urgentes o realizar consultas rápidas, lo que permitirá mantener una constante comunicación en tiempo real y de esta forma poder responder rápidamente a los imprevistos facilitando la toma de decisiones en conjunto.
- Repositorio compartido: Usamos Microsoft OneDrive para almacenar y compartir los documentos y archivos del proyecto, esto con la finalidad de facilitar el trabajo colaborativo en tiempo real, ya que podemos editar y comentar en los archivos de manera simultánea.

1. Introducción

El phishing sigue siendo una de las amenazas más persistentes y devastadoras en el ámbito de la ciberseguridad empresarial, por ello, el phishing es uno de los desafíos más críticos para la ciberseguridad en el entorno corporativo. A pesar de los avances tecnológicos, las organizaciones enfrentan cada vez más ataques sofisticados que logran evadir los sistemas de detección tradicionales, ya que las soluciones convencionales dependen de enfoques basados en reglas estáticas o heurísticas, listas negras y, en algunos casos, machine learning, los cuales presentan una serie de limitaciones significativas, entre ellas, que no pueden comprender el contexto del correo ni adaptarse de manera ágil a las nuevas variantes de ataques, especialmente en casos donde los atacantes utilizan técnicas avanzadas de suplantación de identidad o lenguaje ambiguo, además, la mayoría de las soluciones modernas que utilizan sistemas de detección más complejos y estructurados han estado orientadas hacia implementaciones en la nube, debido a la gran cantidad de recursos computacionales necesarios para ejecutar estos algoritmos, y al hecho de que estos son algoritmos propietarios, que centralizan su consulta en la nube y que requieren acceso controlado mediante APIs.

Hasta hace poco, los LLMs que se podrían utilizar eran propietarios y requerían infraestructuras de gran escala, lo que dificultaba su adopción en implementaciones locales (on-premise), sin embargo, el panorama ha cambiado con la aparición de modelos más pequeños, eficientes, pero sobre todo open source, como Llama 3.2 de Meta, y Phi 4 de Microsoft, que ofrecen modelos que varían entre 8.000 y 14.000 millones de parámetros en sus versiones intermedias, características que los hacen adecuado para ser implementado en dispositivos con capacidad de cómputo limitadas, utilizando simplemente hardware más accesible como tarjetas gráficas (GPUs), pero permitiendo ejecutar razonamiento complejo y a la vez ofreciendo una gran oportunidad para mejorar la seguridad sin comprometer la privacidad de los datos.

El objetivo de este proyecto es desarrollar un sistema on-premise de monitoreo y detección de phishing en tiempo real mediante el uso de LLMs de código abierto, permitiendo a las organizaciones analizar y procesar correos electrónicos dentro de su propia infraestructura, sin exponer datos sensibles a la nube, lo que representa una mejora significativa en la seguridad.

Además, estos LLMs se pueden someter a un proceso de mejoras al integrar técnicas como Fine-Tuning, Prompt Engineering y Retrieval-Augmented Generation (RAG), que permitirán ajustar los modelos a los patrones de ataque más recientes y de esta manera mejorar la precisión de la detección.

El uso de LLMs para la detección de phishing está justificado en el hecho de que estos ofrece ventajas significativas frente a los enfoques tradicionales, ya que LLMs tienen la capacidad de comprender el contexto completo de los correos electrónicos, permitiendo identificar la intencionalidad del atacante, más allá de simples palabras clave, con esto no solo se mejora la precisión, sino que también permite que el sistema se adapte a nuevas amenazas sin depender de reglas estáticas o listas predefinidas, y a eso le agregamos que la implementación se hará de forma local, las empresas podrán garantizar que sus datos permanecerán seguros, al no requerir de la externalización de la información a plataformas en la nube.

1.1. Motivación

El problema por resolver son las limitaciones de las soluciones tradicionales en la detección de phishing frente a las nuevas variantes de ataques, ya que las herramientas basadas en listas negras o reglas estáticas no pueden adaptarse a las tácticas más avanzadas empleadas por los atacantes, especialmente cuando utilizan lenguaje ambiguo o técnicas de ingeniería social altamente personalizadas. Además, los modelos propietarios de LLMs, que han demostrado ser efectivos en la detección de amenazas mediante NLP, presentan limitaciones por su dependencia de la infraestructura en la nube, lo que expone los datos sensibles a riesgos de privacidad.

El uso de LLMs de código abierto, como Llama 3.2 o Phi 4, marca una gran diferencia, ya que permite la ejecución de soluciones de detección de phishing en entornos on-premise, preservando la confidencialidad de los datos, estos modelos son capaces de comprender el contexto completo de un correo electrónico, identificando no solo las palabras clave, sino también la intencionalidad del atacante, además de ser relevante para la comunidad científica y educativa, ya que combina la tecnología más avanzada con un enfoque práctico y seguro, contribuyendo significativamente a la mejora de la ciberseguridad empresarial.

1.2. Planteamiento del problema

El problema central que se pretende resolver en este trabajo de fin de máster es la falta de soluciones locales (on-premise) que integren técnicas avanzadas de NLP para la realizar tareas avanzadas de lenguaje natural y ofrecer una detección más precisa, adaptativa y segura de phishing en tiempo real, sin comprometer la privacidad de los datos corporativos, todo esto entiendo que las soluciones tradicionales no solo carecen de la capacidad de comprender el contexto completo de un correo electrónico, sino que también dependen de enfoques como listas negras y reglas estáticas, que son ineficaces frente a ataques más complejos y personalizados.

Este proyecto propone una solución innovadora basada en LLMs de código abierto, que puede ser implementado localmente para asegurar que los datos críticos no se externalicen a plataformas en la nube, integrando técnicas avanzadas como Fine-Tuning y RAG, el sistema podrá ajustarse y adaptarse a las nuevas amenazas de phishing, proporcionando un análisis profundo del contenido y la intencionalidad de los correos electrónicos maliciosos, permitiendo a las empresas no solo mejorar su seguridad, sino también mantener el control total de su información crítica.

1.3. Estructura del trabajo

Este trabajo se organiza en varios capítulos, cada uno de los cuales aborda diferentes aspectos clave en el desarrollo y evaluación de un sistema on-premise para la detección de phishing mediante el uso de Modelos de Lenguaje Grandes (LLMs). A continuación, se describe brevemente el contenido de cada capítulo.

Capítulo 2: Estado del arte

Este capítulo proporciona una revisión exhaustiva de las soluciones existentes para la detección de phishing, se exploran los enfoques tradicionales como las listas negras y heurísticas basadas en reglas, así como los avances en machine learning y NLP, además, se analiza el papel emergente de los LLMs y su impacto en la ciberseguridad, con énfasis en los modelos de código abierto como Llama y Phi 4 que son relevantes para este proyecto.

Capítulo 3: Objetivos concretos y metodología de trabajo

En este capítulo se exponen los objetivos generales y específicos del proyecto, así como la metodología a seguir para alcanzar dichos objetivos, incluyendo los siguientes ítems:

- Se detalla el objetivo principal, que es desarrollar un sistema de detección de phishing en tiempo real que combine la eficiencia de los LLMs de código abierto con la privacidad de una implementación on-premise.
- Se definen los objetivos específicos, que incluyen la adaptación de los LLMs, la implementación de técnicas como Fine-Tuning y RAG, y la evaluación del sistema.
- Se describe el enfoque metodológico, que abarca desde la identificación de requisitos hasta el desarrollo y la evaluación del sistema, además de establecer los pasos para el ajuste del modelo, así como las herramientas y tecnologías utilizadas.

Capítulo 4: Desarrollo específico de la contribución

Este capítulo aborda el desarrollo detallado del sistema de detección de phishing, destacando la contribución original de este proyecto, incluyendo los siguientes ítems:

- Se describe el enfoque adoptado para el desarrollo del software, desde la planificación hasta el desarrollo.
- Se identifican los requisitos funcionales y no funcionales del sistema, incluyendo aspectos de seguridad, rendimiento y escalabilidad.
- Se presenta la arquitectura del sistema, las tecnologías empleadas y el flujo de trabajo del software, detallando las funcionalidades principales de la herramienta.
- Se analizan los resultados obtenidos del sistema, incluyendo pruebas de detección de phishing en tiempo real.

Capítulo 5: Conclusiones y trabajo futuro

En este capítulo se presentan las conclusiones generales del proyecto, evaluando el éxito de los objetivos alcanzados y las mejoras que ofrece el sistema frente a soluciones existentes. También se discuten las limitaciones del trabajo y las posibles líneas de investigación futura, como la expansión de la funcionalidad del sistema para detectar otros tipos de amenazas cibernéticas o la integración con otros sistemas de seguridad.

Referencias bibliográficas: En esta sección se incluyen todas las fuentes académicas y técnicas utilizadas a lo largo del trabajo.

Anexo A: Este anexo incluye información adicional y la documentación técnica del desarrollo y evaluación del sistema.

2. Estado del arte

El phishing sigue siendo una de las amenazas más persistentes y evolucionadas en el ámbito de la ciberseguridad, con los atacantes aprovechando la ingeniería social para engañar a los usuarios y robar información confidencial.

La aparición de nuevas tecnologías de inteligencia artificial también ha aumentado la complejidad creciente de estos ataques, y esto ha generado que el panorama de soluciones de detección de phishing también haya evolucionado significativamente, integrando tanto enfoques tradicionales de machine learning como métodos más avanzados basados en modelos de lenguaje grandes (LLMs) y aprendizaje automático.

Este estado del arte explora las soluciones actuales para la detección de phishing en tiempo real, abarcando tanto técnicas basadas en listas negras y heurísticas como soluciones más recientes que implementan LLMs de código abierto y tecnologías on-premise.

2.1. Phishing

El phishing es una técnica de ingeniería social que busca engañar a las personas para que revelen información confidencial, como credenciales de acceso, datos bancarios o información personal. Es una de las amenazas más comunes en el ámbito de la ciberseguridad, debido a su adaptabilidad y bajo costo de implementación. Según Hong (2012), “el phishing es exitoso porque explota vulnerabilidades humanas en lugar de fallas tecnológicas, utilizando métodos que imitan comunicaciones legítimas para manipular a los usuarios”[1].

Con el tiempo, las técnicas de phishing han evolucionado desde correos electrónicos masivos poco sofisticados hasta ataques dirigidos como el spear phishing y el whaling, que emplean estrategias más personalizadas para aumentar su eficacia. Jakobsson y Myers (2006)[2] destacan que, aunque las herramientas tecnológicas han mejorado la capacidad para detectar

[1] Hong, J. (2012). The state of phishing attacks. Communications of the ACM, 55(1), 74–81. https://doi.org/10.1145/2063176.2063197
[2] Jakobsson, M., & Myers, S. (Eds.). (2006). Phishing and countermeasures: Understanding the increasing problem of electronic identity theft. Wiley. https://doi.org/10.1002/0470086106

phishing, los ataques siguen siendo efectivos debido a la confianza que las víctimas depositan en los remitentes o plataformas falsas.

El phishing, como estrategia de ataque cibernético, no es un fenómeno nuevo, se cuentan con registro documentados de los primeros intentos de phishing a principios de los años 2000, cuando los atacantes enviaban correos electrónicos masivos simulando ser instituciones financieras y a medida que las defensas cibernéticas evolucionaron, también lo hicieron las tácticas de phishing, diversificándose en variantes más sofisticadas como spear phishing, whaling, smishing y vishing.

El phishing ha experimentado una evolución constante desde su aparición, pasando de ataques masivos y genéricos a tácticas más específicas y dirigidas. Según el Anti-Phishing Working Group (APWG)[3], los ataques de phishing alcanzaron un máximo histórico en 2023, con un promedio de 1.270.883 intentos de phishing reportados mensualmente, lo que representa un aumento significativo con respecto a años anteriores. En su informe del cuarto trimestre de 2023, el APWG destacó que los sectores más atacados fueron las plataformas de medios sociales (42,8%) y los servicios financieros (32,5%), reflejando la adaptabilidad de los atacantes para dirigirse a objetivos rentables y con alta probabilidad de éxito.

Estadísticas recientes muestran los factores determinantes como el aumento en el volumen de ataques, los impactos comerciales y económicos.

En cuanto al volumen de ataques, se pudo establecer según el FBI Internet Crime Complaint Center (IC3)[4], en 2023 se reportaron más de 298,000 incidentes de phishing en estados unidos, lo que representa un aumento del 15% respecto a 2022, además de ello es importante indicar que el phishing sigue siendo el vector de ataque más común, siendo el responsable del 36% de todas las violaciones de datos, según Verizon (2022)[5].

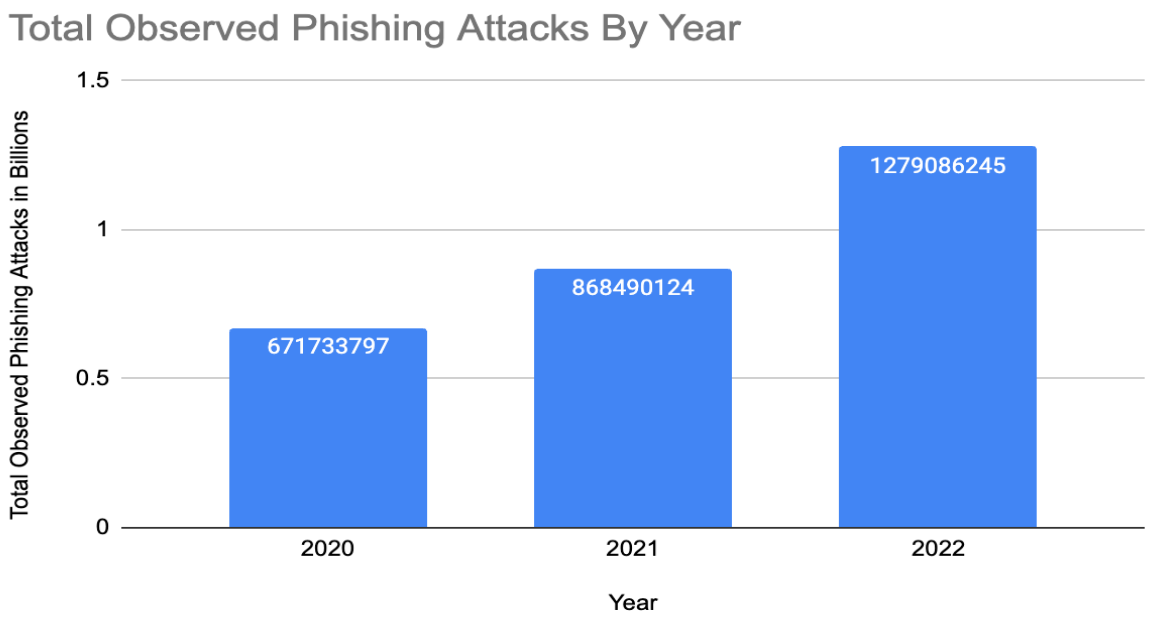
Los datos estadísticos muestran que el número total de ataques de phishing observados entre 2020 y 2022, han tenido un aumento constante y significativo en el volumen de ataques de phishing reportados globalmente, donde, en el 2020 se registraron aproximadamente 671.7

[3] Anti-Phishing Working Group. (2023). Phishing Activity Trends Report Q4 2023. Disponible en: https://apwg.org.
[4] FBI Internet Crime Complaint Center (IC3). (2023). Internet Crime Report 2023. Disponible en: https://www.ic3.gov.
[5] Verizon. (2022). Data Breach Investigations Report. Disponible en: https://www.verizon.com.

millones de ataques de phishing. Y en 2021 los ataques aumentaron a 868.4 millones, representando un crecimiento significativo del 29% respecto al año anterior, y luego 2022 el número de ataques ascendió a 1.28 mil millones, un incremento del 47% en comparación con 2021, como se indica en el siguiente grafico de generado por zscaler(2023)[6].

Figura 1

Incremento Anual de Ataques de Phishing Observados (2020-2022)



La gráfica refleja una tendencia creciente en los ataques de phishing, lo que subraya la evolución y expansión de esta amenaza a nivel global, el 2022 muestra el mayor crecimiento anual, lo que sugiere una diversificación y mayor eficacia de las tácticas utilizadas por los atacantes. Todo esto resalta la urgencia de implementar tecnologías avanzadas como inteligencia artificial y análisis de comportamiento para mitigar los riesgos asociados al phishing.

En 2023, los intentos de phishing globales aumentaron un 58.2% respecto al año anterior, según Zscaler[7], este crecimiento refleja la persistencia y creatividad de los ciberdelincuentes, quienes han adoptado tecnologías emergentes para amplificar sus ataques.

[6] Desai, D., Hegde, R., Laufer, E., & Wang, J. (2023, abril 18). El informe de phishing de 2023 revela un aumento del 47.2 % en los ataques de phishing el año pasado. *Zscaler.com*. https://www.zscaler.com/mx/blogs/security-research/2023-phishing-report-reveals-47-2-surge-phishing-attacks-last-year?utm_source=chatgpt.com

[7] Zscaler ThreatLabz(2024). Informe sobre phishing 2024. Recuperado de https://info.zscaler.com/resources-industry-reports-threatlabz-ai-security-2024-es

A todo lo anterior se debe adicionar el uso de la inteligencia artificial generativa ha permitido a los atacantes automatizar y personalizar campañas con una eficacia sin precedentes.

En cuanto al impacto financiero, en 2023, las pérdidas asociadas a ataques de compromiso de correo electrónico empresarial (BEC, por sus siglas en inglés) superaron los 2,9 mil millones de dólares, según el Informe sobre Delitos en Internet del Centro de Denuncias de Delitos en Internet (IC3) del FBI. Este dato refleja un incremento significativo respecto a los 2,4 mil millones registrados en 2021, subrayando la persistencia y la sofisticación creciente de este tipo de amenazas cibernéticas.

Aunque el phishing y el BEC se clasifican de manera independiente, muchas organizaciones tienden a agrupar los ataques BEC dentro de las estadísticas de phishing, lo que podría generar cierta confusión en la evaluación de la magnitud real de estos ataques. Es importante destacar que estos datos se basan únicamente en los incidentes y pérdidas reportadas, lo que plantea preguntas sobre el número de casos no denunciados que podrían inflar aún más las cifras reales de pérdidas económicas relacionadas con estas amenazas cibernéticas (FBI IC3, 2023)[8].

En cuanto a los sectores más afectados, el sector financiero en 2024, los ciberataques en el sector financiero aumentaron notablemente debido al uso de IA, con un incremento en troyanos bancarios y estafas relacionadas con criptomonedas y las PYMES son especialmente vulnerables, enfrentando una mayor proporción de correos electrónicos maliciosos en comparación con grandes corporaciones.

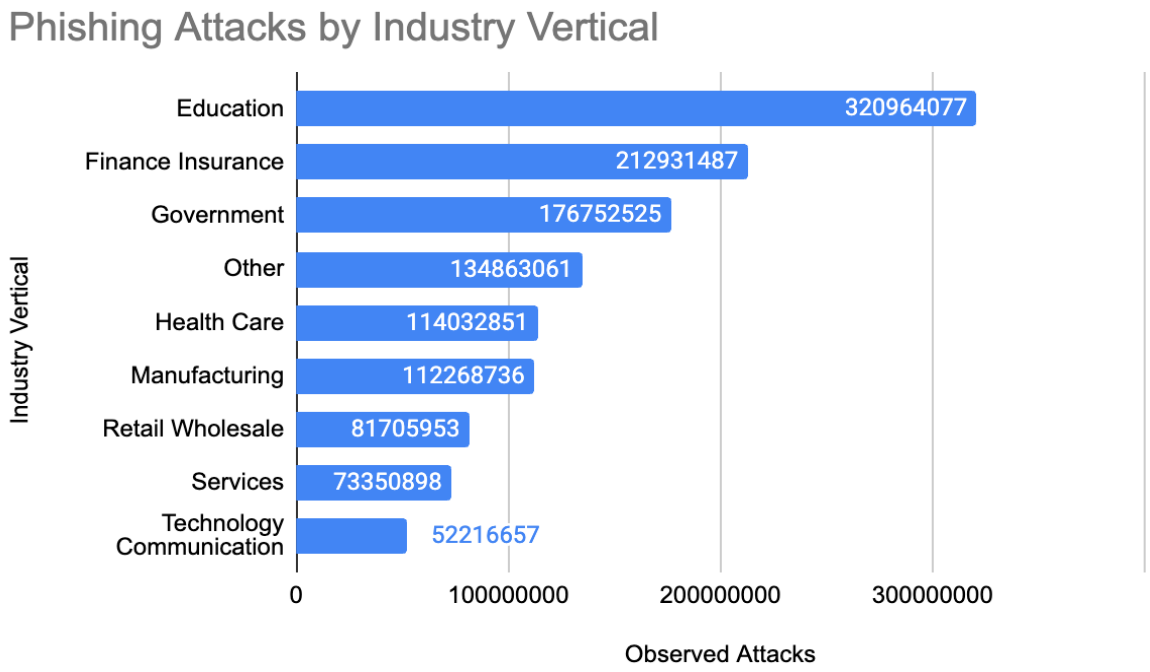
Por otro lado, el informe de Zscaler (2024)[9], revela que el phishing afecta de manera a los sectores industriales, destacando la educación como el más perjudicado con más de 320 millones de ataques observados, probablemente debido a su amplia base de usuarios y recursos compartidos, también el sector financiero y asegurador le sigue con aproximadamente 213 millones de ataques, siendo un objetivo prioritario por su acceso a datos financieros sensibles y la posibilidad de fraudes lucrativos, de igual forma el sector gubernamental, con 176 millones de incidentes, es un blanco crítico por la información

[8] Federal Bureau of Investigation (FBI). (2023). Internet Crime Complaint Center Report 2023. Recuperado de https://www.ic3.gov
[9] Zscaler ThreatLabz(2024). Informe sobre phishing 2024. Recuperado de https://info.zscaler.com/resources-industry-reports-threatlabz-ai-security-2024-es

confidencial que maneja y su rol en infraestructuras esenciales. Además, los sectores de salud y manufactura enfrentaron más de 110 millones de ataques cada uno, siendo la salud especialmente vulnerable por los datos sensibles de los pacientes y la manufactura por riesgos de espionaje industrial y sabotaje. Por otro lado, tecnología y comunicación registraron 52 millones de ataques, el número más bajo, pero siguen siendo objetivos importantes por su relevancia operativa en otras industrias. Este análisis subraya la necesidad de enfoques específicos de ciberseguridad adaptados a las vulnerabilidades de cada sector, continuación se muestra el gráfico de Zscaler(2024)[10].

Figura 2

Distribución de ataques de Phishing por sector industrial en 2023



2.1.1. Evolución del phishing

El phishing, una técnica de ingeniería social diseñada para obtener información confidencial, ha evolucionado drásticamente desde sus inicios en los años 90, desde este momento los primeros ataques se centraban en correos electrónicos genéricos y sitios web falsificados de bancos, y luego con el tiempo, los atacantes han adoptado tácticas más sofisticadas, como el

[10] Zscaler ThreatLabz(2024). Informe sobre phishing 2024. Recuperado de https://info.zscaler.com/resources-industry-reports-threatlabz-ai-security-2024-es

spear phishing y el compromiso de correos electrónicos empresariales (BEC), aprovechando eventos globales y nuevas tecnologías para maximizar su impacto.

En 2023, el Anti-Phishing Working Group (APWG) reportó más de 1,2 millones de ataques de phishing mensuales, marcando un incremento del 27% respecto a 2022 (APWG, 2023). La pandemia de COVID-19 también marcó un punto de inflexión, ya que los atacantes aprovecharon la incertidumbre global para enviar correos fraudulentos relacionados con salud y asistencia financiera (Symantec, 2023).

Los avances en inteligencia artificial han facilitado la creación de mensajes altamente personalizados, dificultando su detección. Por ejemplo, los modelos generativos como GPT-3 han sido utilizados para diseñar correos electrónicos que imitan perfectamente el estilo de comunicación de las organizaciones objetivo (Cofense, 2022)[11].

2.1.2. Técnicas de phishing

El phishing emplea una variedad de técnicas para atraer a las víctimas y evadir los sistemas de detección. Algunas de las más destacadas son:

- Phishing Tradicional: Correos electrónicos masivos con enlaces o archivos adjuntos maliciosos.
- Spear Phishing: Ataques dirigidos a individuos específicos utilizando información personalizada y según Symantec (2023)[12], representa el 65% de los ataques dirigidos.
- Smishing y Vishing: Uso de SMS y llamadas fraudulentas para obtener información personal, estos métodos han crecido un 27% en 2023, según el informe de Zscaler ThreatLabz (2024)[13].
- Zombie Phishing: Respuesta a correos legítimos con enlaces maliciosos, aprovechando la confianza previa entre remitente y destinatario (Cofense, 2022).
- Quishing: Uso de códigos QR maliciosos para redirigir a sitios fraudulentos, este método ha ganado popularidad con el auge de los pagos digitales y la digitalización.

[11] Cofense. (2022). Annual Phishing Report. Recuperado de https://www.cofense.com
[12] Symantec. (2023). Internet Security Threat Report. Recuperado de https://www.symantec.com
[13] Zscaler ThreatLabz. (2024). Phishing Report. Recuperado de https://www.zscaler.com

2.1.3. Clasificación del phishing

El phishing se puede clasificar según los canales utilizados y los objetivos del ataque:

Por canales de ataque

- Correo Electrónico: El método más común, con un 91% de los ataques iniciados a través de este medio (Verizon, 2023).
- Redes Sociales: Plataformas como Facebook y LinkedIn son utilizadas para recopilar información sobre las víctimas.
- Sitios Web Falsificados: Clonación de portales legítimos para obtener credenciales.

Por objetivos del ataque

- Robo de Credenciales: Busca obtener nombres de usuario y contraseñas para acceder a cuentas personales o corporativas.
- Extorsión: Incluye tácticas como la sextorsión, donde los atacantes amenazan con exponer información personal.
- Compromiso de Correo Empresarial (BEC): Ataques dirigidos a desviar fondos o manipular transacciones financieras (FBI IC3, 2023)[14].

2.1.4. Seguridad de Correo Electrónico

La seguridad del correo electrónico es esencial para proteger a las organizaciones contra el phishing, entre las estrategias más efectivas se encuentran:

Protocolos de autenticación

- SPF (Sender Policy Framework): Permite verificar si un correo proviene de un servidor autorizado.
- DKIM (DomainKeys Identified Mail): Agrega una firma digital al correo para garantizar su integridad.

[14] Federal Bureau of Investigation (FBI). (2023). Internet Crime Complaint Center Report 2023. Recuperado de https://www.ic3.gov

- DMARC (Domain-based Message Authentication, Reporting, and Conformance): Combina SPF y DKIM, protección avanzada contra la suplantación de identidad.

Pasarelas de Correo Seguro (SEG)

- Estas herramientas filtran correos maliciosos antes de que lleguen al usuario. Ejemplos incluyen Mimecast y Proofpoint.

Capacitación de Usuarios

- Los programas de simulación de phishing pueden reducir los incidentes hasta en un 70% (Proofpoint, 2021).

2.1.5. Herramientas de análisis

Las herramientas de análisis permiten identificar y mitigar ataques de phishing con mayor precisión:

Basadas en Machine Learning

Modelos como los propuestos por Nguyen et al. (2021)[15] utilizan redes neuronales convolucionales (CNN) para analizar patrones en URLs y detectar phishing en tiempo real.

Herramientas Comerciales

- Cofense Triage: Evalúa y clasifica correos sospechosos reportados por usuarios.
- Zscaler ThreatLabz: Monitorea amenazas globales en tiempo real.
- Barracuda Sentinel: Protege contra BEC y spear phishing.

Plataformas de Código Abierto

- PhishTank: Permite compartir y verificar reportes de phishing globalmente.
- Gophish: Utilizado para pruebas de simulación en empresas.

[15] Nguyen, T., Minh, T., & Huu, N. (2021). Real-time phishing detection using CNN. Journal of Cybersecurity Research.

2.1.6. Impacto global del phishing

El phishing se ha consolidado como una de las amenazas más persistentes y dañinas en el panorama de la ciberseguridad global, este tipo de ataque afecta tanto a usuarios individuales como a organizaciones, causando pérdidas económicas significativas y vulnerando la confianza en las plataformas digitales.

El impacto del phishing no se limita a las pérdidas económicas, este tipo de ataques también afecta la reputación y la continuidad operativa de las empresas, sectores como el financiero, la educación y la salud han sido los más afectados, enfrentando constantes intentos de robo de datos sensibles y compromisos de sus sistemas, según el informe de Zscaler ThreatLabz (2024), la educación lideró como el sector más atacado en 2023, con más de 320 millones de ataques, la dependencia de sistemas digitales y la falta de infraestructura de seguridad adecuada en muchas organizaciones hacen que estos sectores sean particularmente vulnerables, ampliando el alcance y las consecuencias de estos ciberataques.

Además del impacto directo en organizaciones, el phishing tiene implicaciones significativas en el ámbito personal, los usuarios individuales a menudo son víctimas de robo de identidad, extorsión y pérdida de datos confidenciales, lo que puede generar consecuencias psicológicas graves, como el estrés y la pérdida de confianza en las plataformas digitales, la sofisticación creciente de los ataques, que incluye técnicas como spear phishing y smishing, dificulta aún más la capacidad de los usuarios para identificar y evitar amenazas, el informe de Proofpoint (2021) destaca que la falta de conocimiento sobre las tácticas de phishing, especialmente en generaciones mayores, aumenta significativamente la probabilidad de ser víctimas.

A nivel global, el phishing también afecta la estabilidad de la infraestructura digital, con el crecimiento del comercio electrónico, las transacciones digitales y las plataformas en la nube, los ciberdelincuentes tienen un número cada vez mayor de objetivos, este panorama exige una respuesta coordinada entre gobiernos, empresas y usuarios para desarrollar estrategias efectivas de mitigación, las herramientas basadas en inteligencia artificial, los programas de formación y las soluciones de seguridad avanzadas, como las pasarelas de correo seguro (SEG) y los modelos de aprendizaje automático, se han identificado como pilares esenciales para combatir esta amenaza global, sin embargo, como concluyen múltiples informes, la lucha

contra el phishing debe ser un esfuerzo continuo y adaptativo, ya que los ciberdelincuentes seguirán evolucionando sus tácticas para explotar nuevas vulnerabilidades.

2.2. Enfoques tradicionales: Listas negras y heurísticas

Las soluciones tradicionales para la detección de phishing, como listas negras y heurísticas basadas en reglas, han sido pilares en la ciberseguridad durante décadas, estas primeras técnicas permitieron la detección y bloqueo de correos electrónicos maliciosos, URLs o dominios en función de reglas predefinidas o mediante el uso de bases de datos de fuentes maliciosas conocidas.

2.2.1. Listas negras

Los sistemas basados en listas negras almacenan una base de datos de URLs, dominios y direcciones de correo previamente identificadas como maliciosos, y cuando un correo electrónico o enlace coincide con una entrada en la lista negra, el sistema lo bloquea automáticamente el correo entrante. Si bien el enfoque es sencillo, este ha sido ampliamente adoptado por herramientas de seguridad, Soluciones comerciales como Spamhaus, PhishTank, OpenPhish, Microsoft Defender para Office 365, y Cisco Email Security, todo ellos utilizando listas negras actualizadas para prevenir que los usuarios accedan a sitios web maliciosos o abran correos electrónicos de phishing.

Sin embargo, uno de los problemas críticos de las listas negras es que son reactivas, lo que significa que solo protegen contra amenazas previamente identificadas, dejando a los usuarios vulnerables ante ataques de día cero o nuevas variantes de phishing que aún no han sido catalogadas. Esto ha sido documentado en estudios como los de Abu-Nimeh et al. (2007)[16] y Bergholz et al. (2009)[17], quienes demostraron que la efectividad de las listas negras disminuye significativamente con el aumento de los ataques de phishing personalizados o modificados.

[16] Abu-Nimeh, Saeed & Nappa, Dario & Wang, Xinlei & Nair, Suku. (2007). A comparison of machine learning techniques for phishing detection. ACM International Conference Proceeding Series.
[17] Bergholz, Andre & Beer, Jan & Glahn, Sebastian & Moens, Marie-Francine & Paass, Gerhard & Strobel, Siehyun. (2009). New Filtering Approaches for Phishing Email. Journal of Computer Security (JCS).

2.2.2. Heurísticas basadas en reglas

Los sistemas basados en heurísticas emplean un conjunto de reglas predefinidas para analizar correos electrónicos y detectar patrones de phishing comunes, estas reglas pueden incluir la detección de URLs acortadas, dominios sospechosos, encabezados alterados o un análisis superficial del contenido del correo, algunas plataformas y aplicaciones como Fortinet FortiMail, Sophos Email Security, y Proofpoint utilizan reglas heurísticas para filtrar correos electrónicos y proteger a los usuarios contra ataques de phishing.

las heurísticas basadas en reglas funcionan mediante la definición de una serie de patrones y características que son comunes en los correos electrónicos y sitios web maliciosos, estos patrones pueden incluir:

- Uso excesivo de URLs acortadas.
- Presencia de dominios no reconocidos o poco comunes.
- Uso de lenguaje urgente para generar acciones inmediatas (p. ej., "¡Su cuenta será cerrada hoy!" o "Acción urgente requerida").

Si bien las reglas heurísticas han sido útiles, también presentan desafíos similares a las listas negras, tienden a generar una alta tasa de falsos positivos y falsos negativos, lo que compromete su efectividad, además, los atacantes sofisticados pueden ajustar sus tácticas para evadir las reglas existentes, lo que hace que estos sistemas sean vulnerables a ataques avanzados de phishing.

Los estudios como el de Fette et al. (2007)[18] y Garera et al. (2007)[19] han mostrado cómo estas reglas heurísticas pueden ayudar a detectar phishing basado en comportamientos típicos, pero también mostraron que enfoques basados en heurísticas y listas negras tradicionales, tienen serias deficiencias cuando se trata de phishing dirigido o spear phishing, donde los atacantes emplean dominios legítimos o falsifican los encabezados de los correos electrónicos, además de que tienden a tener una mayor tasa de falsos positivos y menor capacidad para adaptarse a nuevos ataques.

[18] Fette, I., Sadeh, N.M., & Tomasic, A. (2007). Learning to detect phishing emails. The Web Conference.
[19] Garera, Sujata & Provos, Niels & Chew, Monica & Rubin, Aviel. (2007). A framework for detection and measurement of phishing attacks. WORM'07 - Proceedings of the 2007 ACM Workshop on Recurring Malcode.

Artículos más recientes como el de Patil y Dhage (2019)[20], que realizaron estudios sobre la metodología heurística aplicada a la detección de phishing, concluyeron que este enfoque tiene limitaciones críticas al enfrentarse a ataques complejos, ya que no puede detectar patrones sutiles ni identificar adecuadamente ataques en los que los atacantes modifican las tácticas tradicionales, por tanto, es claro que su efectividad disminuye considerablemente frente a ataques más sofisticados.

2.3. Evolución hacia modelos de Machine Learning

Con el aumento de los ataques de phishing dirigidos y las técnicas avanzadas de ingeniería social, los sistemas tradicionales demostraron ser insuficientes, ahora, para superar estas limitaciones, la comunidad de ciberseguridad comenzó a adoptar métodos más avanzados basados en machine learning (ML) y deep learning, los cuales aprenden patrones complejos a partir de grandes cantidades de datos, lo que permitió la mejora significativa de la precisión y la adaptabilidad de los sistemas anti-phishing.

2.3.1. Machine learning tradicional

Los primeros enfoques de ML, como Support Vector Machines (SVMs) y Naive Bayes, fueron aplicados en la detección de phishing para mejorar la precisión y reducir las tasas de falsos positivos en comparación con los sistemas tradicionales basados en listas negras y reglas heurísticas.

Fette et al. (2007)[21] desarrollaron PILFER, un sistema de detección basado en Naive Bayes, que lograba una precisión del 96% en la clasificación de correos electrónicos de phishing.

Abu-Nimeh et al. (2007)[22] compararon diferentes algoritmos de ML, incluidos SVMs, Naive Bayes y Redes Neuronales Artificiales (ANNs), demostrando que los enfoques de ML

[20] Patil, Srushti & Dhage, Sudhir. (2019). A Methodical Overview on Phishing Detection along with an Organized Way to Construct an Anti-Phishing Framework.
[21] Fette, I., Sadeh, N.M., & Tomasic, A. (2007). Learning to detect phishing emails. The Web Conference.
[22] Abu-Nimeh, Saeed & Nappa, Dario & Wang, Xinlei & Nair, Suku. (2007). A comparison of machine learning techniques for phishing detection. ACM International Conference Proceeding Series.

tradicionales superaban significativamente a las listas negras, pero aún presentaban dificultades para detectar ataques de phishing más sofisticados.

Sin embargo, estos modelos de ML tradicional enfrentaban desafíos importantes, especialmente en la comprensión del contexto de los correos electrónicos. Si bien podían detectar patrones conocidos, eran menos efectivos frente a ataques personalizados o variantes desconocidas de phishing, además, requerían grandes volúmenes de datos etiquetados para su entrenamiento y tenían dificultades para adaptarse a ataques zero-day, donde los métodos tradicionales de aprendizaje supervisado no eran suficientes.

2.3.2. Modelos de redes neuronales recurrentes (RNN) y LSTM

Las redes neuronales recurrentes (RNNs) y sus variantes, como las Long Short-Term Memory (LSTM), han revolucionado la detección de phishing al abordar la limitación clave que enfrentaban los modelos de ML tradicionales: la capacidad de procesar secuencias de texto y comprender la relación temporal entre diferentes elementos del mensaje.

En el estudio desarrollado por Sahingoz et al. (2019)[23] aplicaron RNNs para la detección de phishing en tiempo real, mostrando que los modelos eran capaces de analizar secuencias textuales de correos electrónicos y detectar patrones que los algoritmos de ML tradicionales no podían identificar.

Xiang et al. (2011)[24] implementaron un enfoque basado en LSTM que podía detectar phishing mediante la comprensión del flujo y la intención en correos electrónicos de phishing, logrando una precisión significativamente mayor que las listas negras y los enfoques heurísticos.

Basnet et al. (2008)[25] probaron modelos basados en RNNs para la detección de phishing, mostrando que estos enfoques podían identificar correos de phishing con mayor precisión en comparación con Naive Bayes y SVM, especialmente en ataques sofisticados.

---

[23] Sahingoz, Ozgur & Buber, Ebubekir & Demir, Onder & Diri, Banu. (2019). Machine learning based phishing detection from URLs. Expert Systems with Applications. 117. 345-357.
[24] Xiang, Guang & Hong, Jason & Rosé, Carolyn & Cranor, Lorrie. (2011). CANTINA+: A Feature-Rich Machine Learning Framework for Detecting Phishing Web Sites. ACM Trans.
[25] Basnet, Ram & Mukkamala, Srinivas & Sung, Andrew. (2008). Detection of Phishing Attacks: A Machine Learning Approach.

Los modelos de LSTM son especialmente efectivos porque pueden retener información a lo largo de largas secuencias textuales, lo que permite identificar patrones persistentes que los métodos tradicionales no logran detectar, además, los modelos de LSTM pueden ser entrenados con datasets grandes y no estructurados, lo que aumenta su capacidad de adaptarse a ataques zero-day.

2.3.3. Redes neuronales convolucionales (CNN)

Además del análisis textual, las redes neuronales convolucionales (CNN) han sido aplicadas en la detección de phishing, particularmente en el análisis de URLs en correos electrónicos y sitios web sospechosos.

Le et al. (2018)[26] demostraron que las CNNs podían clasificar URLs con alta precisión al identificar patrones en las estructuras de los enlaces y su contenido, lo que mejoraba la capacidad de detectar URLs maliciosas utilizadas en ataques de phishing.

Otros estudios aplicaron CNNs en combinación con técnicas de NLP para analizar tanto el contenido del correo electrónico como las URLs incluidas, lo que incrementó la tasa de detección de ataques de phishing complejos.

Dam et al. (2024)[27] propone un enfoque basado en redes neuronales convolucionales (CNN) para la detección en tiempo real de sitios de phishing, el objetivo principal del estudio es prevenir ataques de phishing mediante una extensión de navegador que emplea modelos de deep learning para analizar y clasificar URLs maliciosas. Los autores desarrollaron un sistema que logra una precisión del 98.4% en la detección de URLs de phishing, utilizando un conjunto de datos con más de 650,000 muestras.

Las CNNs son capaces de extraer características de alto nivel a partir de los datos, lo que permite identificar patrones ocultos en las URLs que pueden pasar desapercibidos para los modelos tradicionales o las listas negras.

---

[26] Le, Hung & Pham, Quang & Sahoo, Doyen & Hoi, Steven. (2018). URLNet: Learning a URL Representation with Deep Learning for Malicious URL Detection.

[27] Dam, Minh-Linh & Hung, Ha & Chau, Han & Vu, Quang Sy & Tran, Thanh-Nam. (2024). Real-time phishing detection using deep learning methods by extensions. International Journal of Electrical and Computer Engineering (IJECE).

2.4. Inteligencia artificial y machine learning

La Inteligencia Artificial (IA) y el Machine Learning (ML) han revolucionado diversos sectores, transformando la forma en que interactuamos con la tecnología, procesamos datos y enfrentamos desafíos globales. Estas tecnologías, que comenzaron como conceptos teóricos, ahora son fundamentales en áreas como la ciberseguridad, donde juegan un papel crucial para detectar y mitigar amenazas como el phishing. Este documento ofrece una visión completa sobre la IA y el ML, abordando su historia, evolución, aplicaciones y su impacto específico en la ciberseguridad.

2.4.1. Historia de la Inteligencia artificial

La IA tiene sus raíces en la década de 1940, cuando los avances en matemáticas y computación comenzaron a sentar las bases teóricas. Alan Turing, con su famoso artículo de 1950 "Computing Machinery and Intelligence", planteó la posibilidad de que las máquinas pudieran pensar. Este concepto llevó al desarrollo de programas iniciales como el juego de damas de Arthur Samuel en los años 50.

En los años 60 y 70, los investigadores desarrollaron sistemas basados en reglas y lenguajes específicos como LISP, pero el progreso se vio limitado por la falta de potencia computacional. La llegada de los años 80 marcó un auge en la IA, gracias al desarrollo de sistemas expertos, mientras que los años 90 y 2000 vieron la transición hacia el aprendizaje automático y la explotación de grandes volúmenes de datos.

2.4.2. Historia del machine learning

El aprendizaje automático, una rama de la IA, comenzó a ganar relevancia en los años 50, cuando Arthur Samuel introdujo el término "Machine Learning" al desarrollar un programa capaz de aprender a jugar a las damas. Posteriormente, en los años 90, los avances en algoritmos y la explosión de datos dieron lugar a técnicas como las redes neuronales y los modelos de regresión.

El siglo XXI ha sido testigo de una revolución en el ML, con la llegada de técnicas avanzadas como el aprendizaje profundo (Deep Learning) y el procesamiento del lenguaje natural (NLP), impulsadas por la disponibilidad de potentes GPUs y grandes volúmenes de datos.

2.4.3. Evolución y utilidades de la IA y el ML

El avance de la IA y el ML ha sido impulsado por:

- Incremento en el poder de cómputo: La aparición de GPUs y TPUs ha permitido entrenar modelos más complejos en menos tiempo.
- Acceso a Big Data: Los datos masivos han permitido alimentar modelos con información más rica y variada.
- Desarrollo de arquitecturas avanzadas: Modelos como las redes neuronales convolucionales (CNNs) y los transformadores (como GPT) han mejorado la precisión en tareas de visión por computadora y lenguaje.

La IA y el ML tienen aplicaciones en múltiples áreas, entre ellas:

- Medicina: Diagnóstico de enfermedades mediante imágenes médicas y análisis de datos genómicos.
- Transporte: Sistemas autónomos como vehículos autoconducidos.
- Finanzas: Detección de fraudes en transacciones electrónicas.
- Entretenimiento: Recomendaciones personalizadas en plataformas como Netflix y Spotify.
- Ciberseguridad: Detección de amenazas como phishing, abordado en detalle en este documento.

2.4.4. Impacto en la Ciberseguridad

El phishing, una de las amenazas más prevalentes en la ciberseguridad, se ha beneficiado tanto de la evolución tecnológica como del uso de IA por parte de los atacantes, estas tecnologías también se han convertido en herramientas clave para mitigar estos riesgos, ya que las soluciones basadas en ML han demostrado ser más efectivas que los enfoques tradicionales, como las listas negras y los sistemas basados en reglas.

- Detección de patrones: Los modelos de ML pueden analizar patrones de comportamiento en grandes volúmenes de datos para identificar actividades sospechosas.
- Análisis en tiempo real: Los sistemas basados en IA pueden procesar grandes cantidades de información de manera inmediata, detectando amenazas antes de que se conviertan en ataques exitosos.
- Adaptación a nuevas amenazas: A diferencia de los sistemas estáticos, los modelos de ML pueden actualizarse constantemente para aprender de nuevas tácticas utilizadas por los atacantes.

Además, se han implementado nuevas técnicas avanzadas como:

- Redes Neuronales Convolucionales (CNNs): Usadas para analizar imágenes y detectar sitios web de phishing.
- Modelos de Procesamiento de Lenguaje Natural (NLP): Permiten comprender el contexto de correos electrónicos maliciosos y diferenciar entre mensajes legítimos y fraudulentos.
- Deep Learning: Utilizado para detectar patrones complejos en el tráfico de red y prevenir ataques avanzados.

2.4.3. Impacto de la IA en el Phishing

El phishing ha evolucionado desde simples correos genéricos hasta campañas personalizadas como el spear phishing. Según el FBI Internet Crime Complaint Center (IC3), los ataques de phishing representaron más del 40% de los delitos cibernéticos reportados en 2023. Los sistemas basados en ML han sido cruciales para contrarrestar esta tendencia.

Ejemplos de avances incluyen:

- Modelos como GPT-3: Pueden analizar el contenido semántico de correos electrónicos para identificar señales de phishing.
- Integración de IA en soluciones comerciales: Herramientas como Cofense Triage y Mimecast han adoptado ML para mejorar la detección de correos electrónicos maliciosos.

Los atacantes también han adoptado la IA para mejorar sus tácticas, utilizando herramientas generativas para crear mensajes más convincentes y diseñar sitios web falsificados que imitan perfectamente portales legítimos.

2.5. Proyectos UNIR relacionados con Phishing

La Universidad Internacional de La Rioja (UNIR) ha impulsado investigaciones centradas en la mitigación del phishing, con un enfoque en los riesgos asociados al correo electrónico en las PYMEs y la concienciación sobre delitos informáticos a nivel nacional.

Uno de los proyectos que ha impulsado corresponde a proyecto "Desarrollo de una metodología para el tratamiento de phishing en correos electrónicos en pymes" (Martín González, 2023)[28], este proyecto tiene como objetivo desarrollar un marco que permita la protección y gestión de correos electrónicos maliciosos en empresas pequeñas y medianas, donde los recursos y el conocimiento técnico suelen ser limitados, la investigación propone estrategias accesibles y efectivas, con un enfoque integral que incluye:

- Herramientas de análisis de cabeceras y URLs sospechosas: Se destacan procedimientos para identificar patrones de URLs maliciosas y remitentes no legítimos mediante algoritmos de filtrado y validación.
- Capacitación y simulaciones: El proyecto plantea la necesidad de incorporar entrenamientos prácticos para mejorar la identificación de amenazas por parte de los usuarios.

Se concluye que los entornos empresariales con menos presupuesto para infraestructura tecnológica requieren metodologías que combinen educación con herramientas de protección escalables y de fácil.

[28] **Martín González, G. A. (2023)**. *Desarrollo de una metodología para el tratamiento de phishing en correos electrónicos en pymes*. Trabajo fin de estudio presentado en la Universidad Internacional de La Rioja, Escuela Superior de Ingeniería y Tecnología, Máster Universitario en Ciberseguridad.

Otro estudio es promovido por la UNIR es "Análisis y Simulación de Phishing como Delito Informático en Ecuador" (Buitrago Forero et al. 2024)[29], en este estudio se enfoca en comprender la evolución del phishing en Ecuador, destacando su impacto económico y social mediante el uso de simulaciones avanzadas de ataques, entre los puntos clave destacan:

- Estadísticas de incidentes: El análisis muestra un aumento considerable de ataques dirigidos hacia usuarios de servicios financieros y gubernamentales.
- Evaluación de campañas de simulación: Se realizaron pruebas en diferentes sectores para evaluar las respuestas y detectar deficiencias en la identificación de correos maliciosos.
- Concienciación y políticas públicas: El proyecto resalta la necesidad de crear campañas de educación digital para fortalecer la confianza y la capacidad de respuesta de los ciudadanos ante ataques de phishing.

Se concluye que la educación y la simulación en un entorno controlado ayudan a minimizar los riesgos, especialmente en sectores críticos como el financiero, y asimismo, se evidencia que el uso de inteligencia artificial puede complementar los procesos de detección, mejorando la precisión y reduciendo falsos positivos.

2.6. LLMs y su impacto en la detección de Phishing

El desarrollo de los Modelos de Lenguaje Grandes (LLMs), como GPT-3, BERT, y modelos de código abierto como Llama y Phi, ha transformado la detección de phishing al permitir una comprensión más profunda del lenguaje y el contexto, estos modelos, entrenados en enormes cantidades de datos textuales, tienen la capacidad de detectar patrones sutiles y comprender la intención detrás de los correos electrónicos, permitido una mayor precisión en la detección de ataques de phishing sofisticados, en comparación con enfoques más tradicionales, como las listas negras y las heurísticas basadas en reglas.

[29] **Buitrago Forero, L. V., Ocampo Martínez, B. R., & Velásquez Durán, S. D. (2024)**. *Análisis y simulación de phishing como delito informático en Ecuador*. Universidad Internacional de La Rioja, Escuela Superior de Ingeniería y Tecnología.

Aunque los LLMs han mejorado significativamente la detección de phishing, existen desafíos importantes que deben tenerse en cuenta, como el alto consumo de recursos computacionales y la necesidad de entrenar los modelos con grandes volúmenes de datos etiquetados, además, de la existencia latente del riesgo de que los atacantes utilicen tácticas más avanzadas para eludir los modelos de IA, lo que hace necesario seguir investigando y mejorando los modelos de LLM.

2.6.1. Modelos basados en procesamiento de lenguaje natural (NLP)

El desarrollo de modelos basados en Procesamiento de Lenguaje Natural (NLP) ha permitido una mejora considerable en la detección de phishing, especialmente en la comprensión del contexto completo de los correos electrónicos. Los sistemas basados en NLP permiten que las máquinas entiendan no solo palabras clave, sino también la intención detrás de las frases, lo que es crucial para detectar ataques avanzados que emplean lenguaje ambiguo o ingeniería social.

Miyamoto et al. (2009)[30] desarrollaron un sistema que aplicaba técnicas de NLP para analizar el contenido semántico de los correos electrónicos, logrando una mayor precisión en la identificación de phishing dirigido.

Chandrasekaran et al. (2006)[31] investigaron el uso de NLP en conjunto con modelos de aprendizaje automático para analizar los patrones textuales en correos electrónicos de phishing, demostrando una tasa de detección mucho más alta en comparación con los sistemas tradicionales.

Los modelos de NLP permiten detectar ataques más sofisticados, donde los atacantes imitan con precisión el tono y estilo de los correos electrónicos legítimos. Además, estos modelos pueden aprender continuamente a partir de nuevos datos, lo que les permite adaptarse a nuevas variantes de phishing y ataques altamente personalizados.

[30] Miyamoto, Daisuke & Hazeyama, Hiroaki & Kadobayashi, Youki. (2008). An Evaluation of Machine Learning-Based Methods for Detection of Phishing Sites. Australian Journal of Intelligent Information Processing Systems.
[31] Chandrasekaran, M., Narayanan, K., & Upadhyaya, S. (2006). Phishing email detection based on structural properties. In NYS Cyber Security Conference, 3, 2-8.

2.6.2. Modelos de lenguaje de gran tamaño (LLM)

Los LLM son modelos de aprendizaje profundo con miles de millones de parámetros, estos modelos han demostrado capacidades impresionantes para resolver tareas de NLP en múltiples idiomas y dominios, utilizando preentrenamiento en grandes conjuntos de datos y ajuste fino (fine-tuning) para tareas específicas.

Características principales:

- Tamaño: Pueden superar los 175 mil millones de parámetros, como es el caso de GPT-3 de OpenAI (Brown et al., 2020)[32].
- Escalabilidad: Mayor tamaño generalmente correlaciona con mejor rendimiento en tareas de NLP (Kaplan et al., 2020)[33].
- Aprendizaje en contexto: Modelos como GPT-3 y GPT-4 pueden realizar tareas sin necesidad de entrenamiento adicional, utilizando demostraciones en contexto (few-shot learning).

Los modelos de lenguaje de gran tamaño (LLM) se dividen en varias categorías según su arquitectura y aplicaciones. Los modelos autoregresivos, como GPT-3 y GPT-4 de OpenAI (Brown et al., 2020)[34], generan texto palabra por palabra, prediciendo cada término basado en el contexto anterior. Por otro lado, los modelos encoders-decoders, como T5 de Google (Raffel et al., 2020)[35], están diseñados específicamente para tareas de entrada-salida, como la traducción automática y el resumen de textos. En un enfoque más avanzado, los modelos multimodales integran datos textuales y visuales, lo que permite aplicaciones como la generación de imágenes a partir de texto; un ejemplo destacado es Flamingo de DeepMind

[32] Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., & Amodei, D. (2020). Language models are few-shot learners. Advances in Neural Information Processing Systems, 33, 1877–1901. https://doi.org/10.48550/arXiv.2005.14165
[33] Kaplan, J., McCandlish, S., Henighan, T., Brown, T. B., Chess, B., Child, R., Gray, S., Radford, A., Wu, J., & Amodei, D. (2020). Scaling laws for neural language models. https://doi.org/10.48550/arXiv.2001.08361
[34] Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., ... & Amodei, D. (2020). Language models are few-shot learners. Advances in Neural Information Processing Systems, 33, 1877–1901. https://doi.org/10.48550/arXiv.2005.14165
[35] Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., & Liu, P. J. (2020). Exploring the limits of transfer learning with a unified text-to-text transformer. Journal of Machine Learning Research, 21(140), 1–67. https://jmlr.org/papers/v21/20-074.html

(Alayrac et al., 2022)[36]. Finalmente, los modelos híbridos, como Gemini de Google DeepMind, combinan capacidades autoregresivas y multimodales para adaptarse a una amplia variedad de tareas, maximizando su flexibilidad y alcance en aplicaciones complejas.

Las principales empresas líderes en el desarrollo de Modelos de Lenguaje de Gran Tamaño (LLM) han realizado contribuciones significativas al campo. OpenAI, con modelos como GPT-3 y GPT-4, ha revolucionado la generación de texto al introducir conceptos como el aprendizaje en contexto (few-shot y zero-shot learning), haciendo estos modelos ampliamente accesibles a través de API. Google, por su parte, es pionera en el uso de transformers y modelos encoder, como BERT, T5 y PaLM, destacándose en tareas de clasificación y traducción. Meta (Facebook) ha desarrollado modelos como LLaMA y OPT, priorizando el acceso académico y fomentando el uso abierto para la investigación. Microsoft ha contribuido con Turing-NLG, integrando modelos de lenguaje en aplicaciones prácticas dentro de su ecosistema Azure.

2.6.3. Comparativas de los modelos LLMs

De acuerdo con el articulo titulado "Los 5 mejores modelos de lenguajes grandes (LLM) en noviembre de 2024"[37] ofrece una visión detallada de los modelos de lenguaje más avanzados hasta esa fecha, destacando sus características clave, rendimientos en benchmarks y posibles aplicaciones, entre los modelos analizados se encuentran Claude 3 de Anthropic, lanzado en marzo de 2024, que representa un avance significativo en las capacidades de inteligencia artificial.

El cuadro comparativo siguiente se presenta un análisis detallado del rendimiento de diversos modelos de lenguaje grande (LLMs), incluyendo Claude 3 (en sus variantes Opus, Sonnet y Haiku), GPT-4, GPT-3.5 y las versiones de Gemini 1.0 (Ultra y Pro).

[36] Alayrac, J.-B., Donahue, J., Luc, P., Miech, A., Barr, I., Hasson, Y., Zweifel, L., Lapedriza, A., Neverova, N., Raposo, D., Li, Y., Chuang, G., Carreira, J., & Zisserman, A. (2022). Flamingo: A visual language model for few-shot learning. arXiv preprint arXiv:2204.14198. https://doi.org/10.48550/arXiv.2204.14198
[37] McFarland, A. (2024, septiembre 19). *Los 5 mejores modelos de lenguajes extensos (LLM) en diciembre de 2024*. Unite.AI. https://www.unite.ai/es/mejores-modelos-de-lenguajes-grandes-llms/

Tabla 2

Comparativa de modelos LLM

	Claude 3 Opus	Claude 3 Sonnet	Claude 3 Haiku	GPT-4	GPT-3.5	Gemini 1.0 Ultra	Gemini 1.0 Pro
Undergraduate level knowledge *MMLU*	86.8% 5 shot	79.0% 5-shot	75.2% 5-shot	86.4% 5-shot	70.0% 5-shot	83.7% 5-shot	71.8% 5-shot
Graduate level reasoning *GPQA, Diamond*	50.4% 0-shot CoT	40.4% 0-shot CoT	33.3% 0-shot CoT	35.7% 0-shot CoT	28.1% 0-shot CoT	—	—
Grade school math *GSM8K*	95.0% 0-shot CoT	92.3% 0-shot CoT	88.9% 0-shot CoT	92.0% 5-shot CoT	57.1% 5-shot	94.4% Maj1@32	86.5% Maj1@32
Math problem-solving *MATH*	60.1% 0-shot CoT	43.1% 0-shot CoT	38.9% 0-shot CoT	52.9% 4-shot	34.1% 4-shot	53.2% 4-shot	32.6% 4-shot
Multilingual math *MGSM*	90.7% 0-shot	83.5% 0-shot	75.1% 0-shot	74.5% 8-shot	—	79.0% 8-shot	63.5% 8-shot
Code *HumanEval*	84.9% 0-shot	73.0% 0-shot	75.9% 0-shot	67.0% 0-shot	48.1% 0-shot	74.4% 0-shot	67.7% 0-shot
Reasoning over text *DROP, F1 score*	83.1 3-shot	78.9 3-shot	78.4 3-shot	80.9 3-shot	64.1 3-shot	82.4 Variable shots	74.1 Variable shots
Mixed evaluations *BIG-Bench-Hard*	86.8% 3-shot CoT	82.9% 3-shot CoT	73.7% 3-shot CoT	83.1% 3-shot CoT	66.6% 3-shot CoT	83.6% 3-shot CoT	75.0% 3-shot CoT
Knowledge Q&A *ARC-Challenge*	96.4% 25-shot	93.2% 25-shot	89.2% 25-shot	96.3% 25-shot	85.2% 25-shot	—	—
Common Knowledge *HellaSwag*	95.4% 10-shot	89.0% 10-shot	85.9% 10-shot	95.3% 10-shot	85.5% 10-shot	87.8% 10-shot	84.7% 10-shot

Nota. La tabla tiene como fuente Empresa Anthropic de Claude 3

En términos de conocimiento a nivel de pregrado (MMLU), Claude 3 Opus lidera con un 86.8% en la modalidad de 5-shot, seguido de cerca por GPT-4 con un rendimiento similar. Sin embargo, en tareas más avanzadas, como razonamiento a nivel de posgrado (GQPA, Diamond), los resultados descienden significativamente en todos los modelos, siendo Claude 3 Opus el mejor con un 50.4% en modalidad CoT (Chain of Thought), superando a GPT-4 y GPT-3.5.

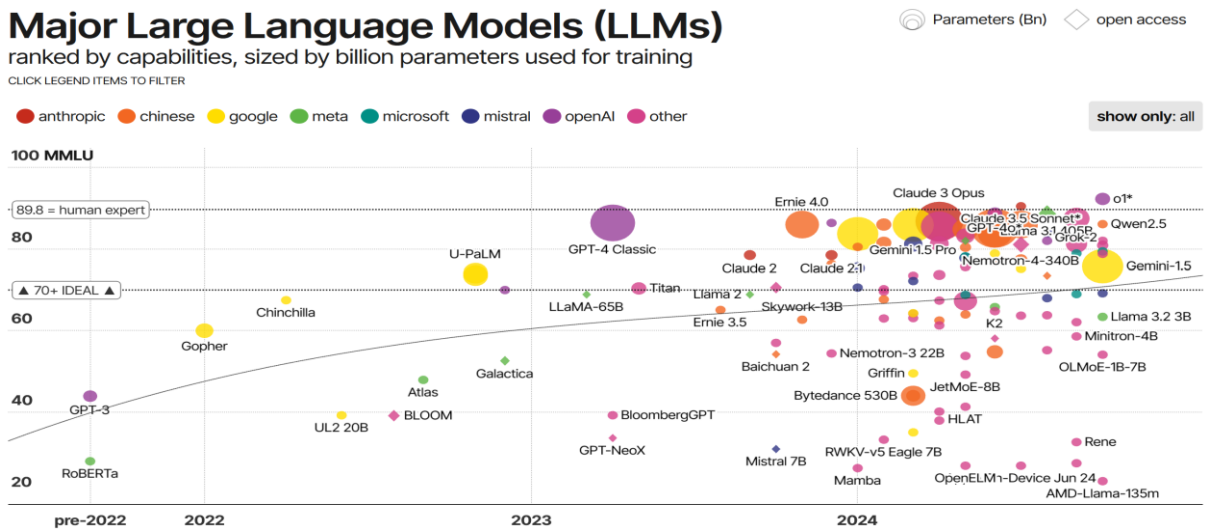
En habilidades específicas, como matemática escolar (GSM8K) y evaluación de código (HumanEval), Claude 3 Opus sobresale con un 95% y 84.9% respectivamente, demostrando su capacidad en tareas estructuradas. Por su parte, GPT-4 mantiene un rendimiento competitivo

en varias categorías, como conocimiento general (ARC-Challenge), con un 95.3%. Gemini 1.0 Ultra y Pro ofrecen resultados sólidos en evaluaciones generales, aunque quedan rezagados frente a Claude y GPT en las categorías más exigentes.

También podemos evaluar los modelos y su evolución en el tiempo, a continuación, podemos ver una línea temporal de (Information is Beautiful,. 2024)[38].

Figura 3

Comparativa de modelos de lenguaje de gran tamaño (LLMs) por capacidades y parámetros (2024)



En la gráfica anterior se presenta una evolución y comparación de los principales Modelos de Lenguaje de Gran Tamaño (LLMs) según sus capacidades (medidas en MMLU) y el número de parámetros utilizados para su entrenamiento (en miles de millones). Los modelos se agrupan por empresas como Anthropic, OpenAI, Google, Meta, Microsoft y otros desarrolladores. En esta representación, GPT-4 Classic de OpenAI destaca como uno de los modelos con mayor rendimiento, superando consistentemente los 80 puntos MMLU, lo que lo posiciona cerca del desempeño humano en tareas complejas. Claude 3 Opus, de Anthropic, también sobresale al superar el umbral de 70 puntos, considerado ideal para modelos avanzados, consolidándose como una de las opciones más robustas en 2024. (Information is Beautiful,. 2024)[39].

[38] The rise of generative AI: Large language models (LLMs) like ChatGPT. Recuperado el 4 de diciembre de 2024, de https://informationisbeautiful.net/visualizations/the-rise-of-generative-ai-large-language-models-llms-like-chatgpt/

[39] The rise of generative AI: Large language models (LLMs) like ChatGPT. Recuperado el 4 de diciembre de 2024, de https://informationisbeautiful.net/visualizations/the-rise-of-generative-ai-large-language-models-llms-like-chatgpt/

Además, se observa un notable incremento en la complejidad y capacidades de los modelos desarrollados más recientemente. Por ejemplo, Gemini 1.5 Pro y Claude 3.5 muestran un balance entre capacidad y accesibilidad, mientras que modelos como LLaMA 65B y Ernie 4 se posicionan como alternativas competitivas para tareas específicas. La tendencia refleja no solo el aumento en la cantidad de parámetros, sino también la diversificación de los objetivos y arquitecturas, como los modelos multimodales que integran datos visuales y textuales. Esta evolución subraya el impacto de la innovación en los LLMs, ampliando su aplicabilidad en diversas industrias y dominios. (Information is Beautiful,. 2024)[40].

2.6.4. Aplicación de LLMs en la detección de Phishing

Los modelos de lenguaje grandes (LLMs) han transformado la detección de phishing al permitir que los sistemas identifiquen patrones complejos y comprendan el significado contextual de los correos electrónicos más allá de simples palabras clave. Los LLMs, como GPT-3 y los modelos de código abierto Llama 3.2 y Phi 4, han demostrado ser significativamente más precisos en la detección de correos electrónicos de phishing en comparación con los enfoques tradicionales basados en listas negras o heurísticas.

Brown et al. (2020)[41], en su trabajo sobre GPT-3, demostraron que los LLMs podían detectar phishing avanzado con una precisión superior a la de los modelos tradicionales. El análisis contextual profundo permitió a GPT-3 detectar incluso correos electrónicos de spear phishing donde los atacantes intentan imitar comunicaciones genuinas.

El artículo Lee et al. (2024)[42] concluye que los Modelos de Lenguaje Grandes Multimodales (LLMs) mejoran significativamente la detección de páginas de phishing al combinar múltiples fuentes de datos, además de que estos modelos ofrecen una mayor precisión y adaptabilidad

[40] The rise of generative AI: Large language models (LLMs) like ChatGPT. Recuperado el 4 de diciembre de 2024, de https://informationisbeautiful.net/visualizations/the-rise-of-generative-ai-large-language-models-llms-like-chatgpt/

[41] Brown, Tom & Mann, Benjamin & Ryder, Nick & Subbiah, Melanie & Kaplan, Jared & Dhariwal, Prafulla & Neelakantan, Arvind & Shyam, Pranav & Sastry, Girish & Askell, Amanda & Agarwal, Sandhini & Herbert-Voss, Ariel & Krueger, Gretchen & Henighan, Tom & Child, Rewon & Ramesh, Aditya & Ziegler, Daniel & Wu, Jeffrey & Winter, Clemens & Amodei, Dario. (2020). Language Models are Few-Shot Learners. 10.48550/arXiv.2005.14165.

[42] Lee, J., Lim, P., Hooi, B., & Divakaran, D. (2024). Multimodal large language models for phishing webpage detection and identification. *ArXiv*, *abs/2408.05941*. https://doi.org/10.48550/arXiv.2408.05941

frente a los enfoques tradicionales basados únicamente en texto, ya que pueden captar información visual y contextual de los sitios web, lo que es esencial para detectar amenazas avanzadas. Los LLMs multimodales son particularmente efectivos contra tácticas de phishing cada vez más sofisticadas.

2.6.5. Soluciones comerciales basadas en LLMs

Varios proveedores de ciberseguridad han integrado LLMs en sus soluciones de detección de phishing para mejorar la capacidad de analizar correos electrónicos en tiempo real y comprender la intención del remitente. Estos modelos han permitido que las soluciones comerciales no solo detecten amenazas con mayor precisión, sino que también proporcionen una mejor explicación de los resultados o decisiones planteadas por los modelos.

Cofense Phishing Protection emplea modelos de LLM para identificar correos electrónicos de phishing mediante el análisis del comportamiento y la intención del remitente, pudiendo identificar intentos sofisticados de suplantación de identidad y ataques BEC (Business Email Compromise).

Ironscales utiliza LLMs para realizar una detección automática de phishing en tiempo real, el sistema analiza los patrones de comportamiento y lenguaje en los correos electrónicos y utiliza el aprendizaje continuo para mejorar su precisión con el tiempo.

Mimecast ha adoptado modelos de LLM para su sistema de seguridad de correo electrónico, lo que ha permitido una mayor protección contra ataques avanzados de phishing y ransomware, utilizando modelos para analizar patrones textuales en los correos y mejorar la precisión en la detección.

Barracuda ha integrado LLMs en sus soluciones de protección de correo electrónico, detectando amenazas basadas en la intención del mensaje y bloqueando correos electrónicos maliciosos antes de que lleguen a los usuarios finales. Su sistema emplea LLMs para identificar amenazas de phishing dirigidas.

2.7. Detección de Phishing con aprendizaje profundo

los avances en inteligencia artificial, particularmente el desarrollo de Modelos de Lenguaje Grandes (LLMs) como GPT-3, abren nuevas posibilidades, estas tecnologías permiten analizar patrones complejos y adaptarse rápidamente a nuevas variantes de phishing mediante técnicas como Fine-Tuning y RAG (Brown et al., 2020)[43].

2.7.1. URLNet

Representación Semántica de URLs URLNet emplea redes neuronales convolucionales (CNNs) para capturar características léxicas y semánticas de URLs, este enfoque mejora significativamente la capacidad para detectar URLs maliciosas mediante una representación no lineal optimizada (Le et al., 2018)[44].

URLNet representa un avance significativo al capturar patrones secuenciales y semánticos en URLs. Este enfoque logró una precisión del 98% en pruebas controladas, demostrando su eficacia frente a métodos tradicionales (Le et al., 2018)[45].

2.7.2. Fine-Tuning en modelos de lenguaje

Fine-Tuning permite personalizar LLMs preentrenados, como GPT-3, para tareas específicas de detección de phishing. Esto mejora significativamente la capacidad del modelo para adaptarse a datos específicos y reconocer patrones complejos en tiempo real (Fette et al., 2007; Le et al., 2018)[46].

Dentro de las principales ventajas están:

---

[43] Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., ... & Amodei, D. (2020). Language models are few-shot learners. Advances in Neural Information Processing Systems (NeurIPS), 33, 1877–1901.

[44] Le, H., Pham, Q., Sahoo, D., & Hoi, S. C. H. (2018). URLNet: Learning a URL representation with deep learning for malicious URL detection. Proceedings of the ACM Conference. https://doi.org/10.475/123_4

[45] Le, H., Pham, Q., Sahoo, D., & Hoi, S. C. H. (2018). URLNet: Learning a URL representation with deep learning for malicious URL detection. Proceedings of the ACM Conference. https://doi.org/10.475/123_4

[46] Le, H., Pham, Q., Sahoo, D., & Hoi, S. C. H. (2018). URLNet: Learning a URL representation with deep learning for malicious URL detection. Proceedings of the ACM Conference. https://doi.org/10.475/123_4

- Alta precisión en entornos dinámicos.
- Adaptabilidad a nuevos patrones de ataque.
- Reducción de falsos positivos.

2.7.3. Retrieval-Augmented Generation (RAG)

RAG combina recuperación de información y generación de texto, permitiendo al modelo consultar bases de datos externas durante la inferencia. Esto lo hace altamente efectivo para identificar URLs desconocidas y enriquecer el análisis contextual (Linh et al., 2024)[47].

Aplicaciones en la detección de phishing:

- Comparación de URLs con bases de datos externas.
- Análisis semántico y sintáctico de correos electrónicos.
- Generación de alertas explicativas basadas en patrones históricos.

2.7.4. Modelos Multimodales

Los modelos multimodales integran datos provenientes de múltiples fuentes, como texto y elementos visuales, para ofrecer un análisis más completo y preciso, en el contexto de la detección de phishing, esta capacidad es particularmente relevante, ya que muchas páginas de phishing combinan textos manipulados con elementos visuales engañosos para confundir a los usuarios y hacer que confíen en sitios web falsos. Estos modelos permiten procesar y analizar simultáneamente ambas dimensiones, capturando patrones que no serían detectables utilizando únicamente datos textuales (Linh et al., 2024)[48].

Además, estos modelos aprovechan el aprendizaje profundo para identificar correlaciones entre los textos y los elementos visuales de una página web, por ejemplo, pueden detectar inconsistencias entre el logotipo de una empresa legítima y los mensajes de texto asociados,

[47] Linh, D. M., Hung, H. D., Chau, H. M., Vu, Q. S., & Tran, T.-N. (2024). Real-time phishing detection using deep learning methods by extensions. International Journal of Electrical and Computer Engineering (IJECE), 14(3), 3021–3035. https://doi.org/10.11591/ijece.v14i3.pp3021-3035

[48] Linh, D. M., Hung, H. D., Chau, H. M., Vu, Q. S., & Tran, T.-N. (2024). Real-time phishing detection using deep learning methods by extensions. International Journal of Electrical and Computer Engineering (IJECE), 14(3), 3021–3035. https://doi.org/10.11591/ijece.v14i3.pp3021-3035

como gramática incorrecta o redacción poco profesional, esto es fundamental en escenarios donde los atacantes intentan replicar visualmente un sitio confiable, pero fallan al mantener la coherencia en otros aspectos. Los modelos multimodales destacan por su capacidad de análisis contextual y su habilidad para identificar discrepancias de forma automatizada (Linh et al., 2024).

Finalmente, la aplicación de estos modelos no solo mejora la precisión en la detección de phishing, sino que también reduce los falsos positivos al proporcionar un enfoque integral que evalúa múltiples variables de un sitio sospechoso. En entornos empresariales, esta tecnología puede ser utilizada para analizar grandes volúmenes de tráfico web, identificando amenazas en tiempo real y minimizando riesgos para los usuarios. Por lo tanto, los modelos de lenguaje multimodal representan una herramienta clave en la lucha contra amenazas avanzadas, integrando técnicas innovadoras para abordar los desafíos modernos de la ciberseguridad (Linh et al., 2024).

2.7.5. Eficiencia de métodos avanzados

La detección de phishing ha evolucionado significativamente gracias a la adopción de métodos avanzados que superan las limitaciones de enfoques tradicionales como las listas negras y las heurísticas estáticas, las listas negras, aunque rápidas y fáciles de implementar, presentan una precisión baja y carecen de adaptabilidad frente a nuevas amenazas. Al depender de bases de datos previamente conocidas, no logran identificar URLs generadas dinámicamente o tácticas novedosas empleadas por los atacantes. Las heurísticas estáticas, por su parte, ofrecen un nivel de precisión moderado al analizar características predefinidas, como la longitud de URLs o el uso de palabras clave sospechosas, pero su capacidad para adaptarse a patrones emergentes sigue siendo limitada. Ambos métodos presentan la ventaja de operar en tiempo real, pero su eficacia disminuye frente a amenazas complejas y sofisticadas.

Los enfoques basados en Fine-Tuning de Modelos de Lenguaje Grandes (LLMs) han transformado el panorama de la detección de phishing al ofrecer una alta precisión y adaptabilidad. Estos modelos, ajustados con datos específicos de phishing, son capaces de identificar patrones complejos tanto en texto como en estructuras de URLs, aunque el tiempo de respuesta en tiempo real puede ser ligeramente inferior al de los métodos tradicionales

debido a los recursos computacionales que requieren, los beneficios en términos de reducción de falsos positivos y capacidad para abordar amenazas emergentes compensan esta desventaja. Fine-Tuning permite personalizar los LLMs para ajustarse a necesidades específicas, mejorando su eficacia en entornos dinámicos y complejos.

Retrieval-Augmented Generation (RAG) y los modelos multimodales representan la cúspide de la eficiencia en la detección de phishing. RAG, con su capacidad para integrar bases de datos externas en tiempo real, y los modelos multimodales, que combinan análisis textual y visual, ofrecen precisión y adaptabilidad excepcionales, ambos enfoques destacan no solo por su capacidad para identificar amenazas conocidas y desconocidas, sino también por operar con alta eficacia en entornos de tiempo real. En particular, los modelos multimodales, con su habilidad para correlacionar texto y elementos visuales, abordan los desafíos más sofisticados, como el análisis de páginas de phishing visualmente convincentes. Estos métodos avanzados han establecido un nuevo estándar en la ciberseguridad, demostrando ser herramientas indispensables en la lucha contra el phishing.

La información resumida se puede observar en la siguiente tabla:

Tabla 3

Eficiencia de métodos avanzados

Método	Precisión	Adaptabilidad	Tiempo Real
Listas negras	Baja	Muy limitada	Alta
Heurísticas estáticas	Media	Limitada	Alta
Fine-Tuning (LLMs)	Alta	Alta	Media
RAG	Muy Alta	Muy Alta	Alta
Modelos multimodales	Alta	Alta	Alta

Nota. Elaboración propia.

2.8. Implementaciones On-Premise de LLMs

Uno de los principales desafíos en la adopción de Modelos de Lenguaje Grandes (LLMs) ha sido la alta demanda de recursos computacionales requerida para entrenar y ejecutar estos modelos, lo que inicialmente limitó su implementación a plataformas en la nube.

Los modelos como GPT-3 y BERT, al ser altamente complejos, requerían grandes cantidades de GPU y CPU, además de la infraestructura de almacenamiento necesaria para manejar los enormes conjuntos de datos, estas necesidades computacionales implicaban que las empresas y organizaciones que quisieran aprovechar los LLMs debían externalizar sus datos a la nube, lo que generaba preocupaciones importantes sobre la privacidad y seguridad de los datos.

Sin embargo, la reciente aparición de modelos de código abierto más ligeros como Llama 3.2, Phi 4, Vicuna, y Mixtral ha permitido que los LLMs puedan ser ejecutados en infraestructuras locales (on-premise), manteniendo los datos sensibles dentro de las instalaciones corporativas y eliminando la necesidad de enviar información a servidores externos, estas versiones más optimizadas permiten ejecutar modelos avanzados de NLP en hardware más accesible, como GPUs comerciales, lo que amplía las posibilidades de implementación en entornos corporativos que requieren altos niveles de privacidad.

Los LLMs on-premise presentan numerosas ventajas para las organizaciones, como la protección de la privacidad de los datos, el control total sobre la infraestructura, y la capacidad de personalizar los modelos según las necesidades específicas de la organización, sin embargo, también existen desafíos asociados con estas implementaciones, como la necesidad de hardware especializado (como GPUs) y la gestión de la infraestructura de soporte para el despliegue de modelos LLM de gran escala.

2.8.1. Ventajas de los modelos LLMs en ambientes On-Premise

Los Modelos de Lenguaje de Gran Escala (LLMs, por sus siglas en inglés) han revolucionado la manera en que concebimos la inteligencia artificial, por su capacidad para generar texto de manera similar al humano y comprender tareas relacionadas con el lenguaje natural, se han convertido en herramientas indispensables para organizaciones de diversos sectores, sin embargo, en un entorno donde las soluciones basadas en la nube predominan, las implementaciones on-premise (en local) han emergido como una alternativa sólida para aquellas empresas que priorizan la seguridad, el control y la personalización.

A continuación, se detallan las principales ventajas de los LLMs implementados en entornos on-premise.

2.8.1.1. Seguridad y protección de datos

Implementar LLMs en servidores locales garantiza que los datos sensibles permanezcan dentro de la infraestructura de la organización, esta característica es crucial para sectores como la banca, las instituciones sanitarias o las agencias gubernamentales, donde la protección de datos personales y confidenciales es prioritaria, y mantener los datos "en casa", reduce significativamente los riesgos de fugas de información y violaciones de seguridad.

- Cumplimiento normativo: Muchas regulaciones, como el Reglamento General de Protección de Datos (GDPR) y leyes locales de protección de datos, exigen que cierta información no sea procesada ni almacenada fuera de los servidores locales. Los LLMs on-premise facilitan el cumplimiento de estos estándares.
- Entornos aislados ("air-gapped"): En casos de extrema sensibilidad, los LLMs pueden ejecutarse en redes completamente desconectadas de Internet, proporcionando un nivel de aislamiento que elimina casi por completo la exposición a amenazas externas.

2.8.1.2. Independencia de servicios externos

Con una implementación on-premise, la organización tiene un control total sobre la operación del modelo, sin depender de terceros proveedores de servicios en la nube, lo que elimina la necesidad de enviar datos a servidores externos, eliminando potenciales vulnerabilidades y garantizando que la empresa pueda operar de manera autónoma.

- Control total: La empresa decide cuándo y cómo actualizar el sistema, lo que evita interrupciones imprevistas.
- Reducción de riesgos de dependencia: La falta de dependencia de terceros minimiza problemas relacionados con interrupciones de servicio en la nube, cambios en los términos de servicio o aumentos de precios imprevistos.

2.8.1.3. Personalización y control avanzado

Cada empresa tiene sus propias necesidades y particularidades operativas, y los LLMs en local permiten personalizar los modelos de lenguaje mediante técnicas como fine-tuning (ajuste fino) y prompt engineering, adaptándolos a tareas específicas del negocio.

- Ajuste personalizado: Se pueden ajustar los modelos con información y datos internos de la empresa para mejorar su rendimiento en tareas especializadas.
- Integración fluida: Al estar bajo el control total de la organización, los LLMs pueden integrarse de manera más eficiente con bases de datos internas, sistemas de gestión de contenido y herramientas de análisis propias.

2.8.1.4. Respuesta en tiempo real

Los LLMs ejecutados localmente suelen tener tiempos de respuesta más rápidos al no depender de la comunicación con servidores remotos, capacidad que es ideal para aplicaciones que requieren interacción en tiempo real, como los asistentes virtuales, los sistemas de soporte técnico automatizados o los traductores instantáneos.

- Procesamiento rápido: La baja latencia de los sistemas locales permite que las respuestas sean prácticamente instantáneas.
- Reducción del tráfico de red: Al no enviar y recibir datos a través de Internet, se evita la sobrecarga de la red y se mejora el desempeño general del sistema.

2.8.1.5. Fácil integración con sistemas existentes

Los LLMs on-premise pueden conectarse de manera directa a las bases de datos y herramientas ya utilizadas por la organización, permitiendo un flujo de información más seguro y eficiente, además de evitar problemas de compatibilidad o la necesidad de migrar grandes cantidades de datos a la nube.

- Flujo de trabajo unificado: Los datos críticos se mantienen en el entorno interno de la organización sin necesidad de transferencias externas.
- Compatibilidad total: Los modelos pueden ajustarse para operar con los sistemas existentes de la empresa sin cambios drásticos en la infraestructura.

2.8.1.6. Control de costes a largo plazo

Aunque la implementación inicial de LLMs on-premise puede ser costosa debido a la compra de hardware, licencias y configuraciones, a largo plazo puede ser una inversión más rentable al eliminar pagos recurrentes a servicios de terceros.

- Costes fijos: Una vez realizada la inversión inicial, los gastos se limitan al mantenimiento y actualizaciones internas, sin cuotas mensuales ni cargos adicionales por uso.
- Escalabilidad planificada: La empresa puede planificar el crecimiento de su infraestructura de manera controlada y acorde a sus propias necesidades.

2.8.1.7. Mejora de la privacidad y la soberanía de los datos

En entornos donde la privacidad es crucial, mantener los datos dentro de la red interna asegura un nivel de confidencialidad difícil de alcanzar con soluciones basadas en la nube. Además, la organización mantiene la soberanía sobre sus datos, sin riesgo de que estos sean explotados o accedidos por terceros.

- Privacidad garantizada: Los datos nunca abandonan los servidores de la organización, lo que brinda tranquilidad ante posibles accesos indebidos.
- Confianza de los clientes: Esta implementación puede fortalecer la confianza de los usuarios al demostrar un compromiso con la protección de su información.

2.8.2. Aplicaciones on-Premise basadas en LLMs

Los modelos de código abierto como Vicuna, Mixtral, Falcon LLM, y WizardLM han sido desarrollados específicamente para su uso en entornos on-premise, ofreciendo soluciones avanzadas para la detección de phishing en tiempo real sin necesidad de depender de plataformas en la nube, estas implementaciones permiten a las empresas y organizaciones controlar completamente el procesamiento de sus datos, lo que es crucial en sectores que manejan información sensible, como la banca, la salud, o las telecomunicaciones.

Vicuna ha sido diseñado para proporcionar capacidades avanzadas de procesamiento de lenguaje con recursos computacionales limitados, lo que lo hace ideal para entornos

corporativos que requieren seguridad y eficiencia sin necesidad de recurrir a soluciones en la nube.

Mixtral es otro ejemplo de un LLM optimizado para ser ejecutado de forma local, que ofrece alta precisión en la detección de phishing a la vez que minimiza el consumo de recursos, permitiendo su integración en redes corporativas con infraestructura interna.

Falcon LLM ha sido implementado en diversas industrias para el análisis en tiempo real de correos electrónicos y datos estructurados, ofreciendo soporte on-premise para tareas de detección de amenazas.

WizardLM ha demostrado ser eficaz para la detección de phishing y análisis de contenido en infraestructuras locales, con un diseño eficiente que optimiza los tiempos de respuesta.

Estas soluciones permiten a las organizaciones beneficiarse de las capacidades avanzadas de procesamiento de lenguaje natural y detección de amenazas, sin sacrificar la privacidad o seguridad de los datos al no depender de terceros.

2.8.3. Nuevos modelos de LLMs

En los últimos años, se ha producido un aumento en el desarrollo de nuevos modelos LLM que están específicamente diseñados para implementaciones on-premise, estos modelos ofrecen una combinación de eficiencia, seguridad, y capacidad de personalización, lo que les permite ser implementados en una amplia gama de entornos empresariales sin las limitaciones impuestas por las soluciones basadas en la nube.

Llama 3.2 y Llama 3.2 de Meta son dos de los modelos más prometedores, que permiten su ejecución en infraestructuras locales utilizando recursos computacionales más accesibles y están diseñados para ser más ligeros que los modelos de generación anterior, pero sin perder la capacidad de comprensión avanzada de lenguaje y procesamiento de grandes volúmenes de datos textuales.

Phi 4 de Microsoft ha sido optimizado para entornos corporativos y puede ejecutarse en servidores locales con configuraciones más modestas y está diseñado para adaptarse a tareas de detección de phishing y análisis de correos electrónicos en tiempo real, manteniendo un alto nivel de seguridad sin necesidad de externalizar los datos.

Mistral es otro modelo que ha sido optimizado para el análisis de datos en infraestructuras locales. Su arquitectura flexible permite su despliegue en entornos corporativos de alta demanda, proporcionando resultados rápidos sin comprometer la privacidad.

Además de estos modelos, otros desarrollos recientes incluyen modelos LLM con capacidades de aprendizaje federado, donde los datos no salen de la infraestructura local, pero el modelo se puede mejorar con contribuciones de múltiples fuentes sin comprometer la privacidad de los datos.

2.8.4. Comparativas de modelos LLMs para entornos On-Premise

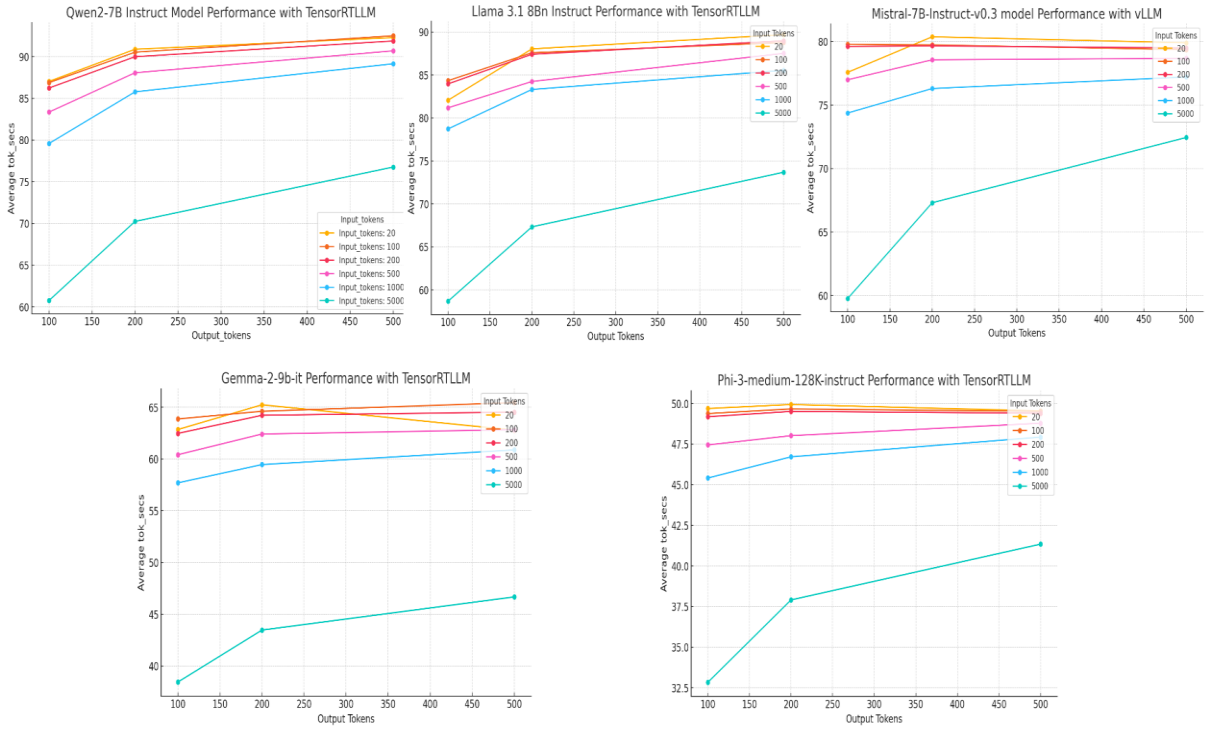
Para hablar de comparativas se deben definir los principales modelos hasta el momento, destacando sobre todo por grandes empresas del sector tecnológico que respalda con infraestructura, muchas data obtenida o generada, y sobre todo grandes equipos de desarrolladores, las diferentes implementaciones y mejoras en sus grandes modelos de generación de lenguajes natural, por eso hoy en día resalta los siguientes modelos:

- Qwen2-7B-Instruct, es un potente modelo de lenguaje de código abierto de Alibaba Cloud, desarrollado para sobresalir en la comprensión y la generación de texto similar a la humana a través de sus parámetros de 7.000 millones.
- Gemma-2-9B-it, lanzada por Google, ofrece una eficiencia y un rendimiento impresionantes por su tamaño, atrayendo a investigadores y desarrolladores con su accesibilidad e integración de código abierto con populares plataformas de IA.
- Llama-3.1-8B-Instruct, desarrollado por Meta, cuenta con capacidades multilingües mejoradas y una enorme ventana de contexto de 128k, haciéndolo cada vez más popular para diversas tareas de idiomas.
- Mistral-7B-Instruct-v0.3, creado por Mistral AI, cuenta con un vocabulario extendido y mejorado, y cuenta con un fuerte desempeño en varios puntos de referencia.
- Phi-3-medium-128k-instruct, introducido por Microsoft, sobresale en tareas de razonamiento y ofrece un rendimiento competitivo contra modelos mucho más grandes, atrayendo la atención por su eficiencia y largo manejo del contexto.

A continuación, se presentará el análisis por medio de imágenes comparativas que ilustran los resultados del rendimiento de diferentes modelos LLMs, destacando su comportamiento en términos de velocidad de generación de tokens y tiempo hasta el primer token (TTFT).

Figura 4

Comportamiento de los modelos en términos de velocidad de generación de tokens[49]



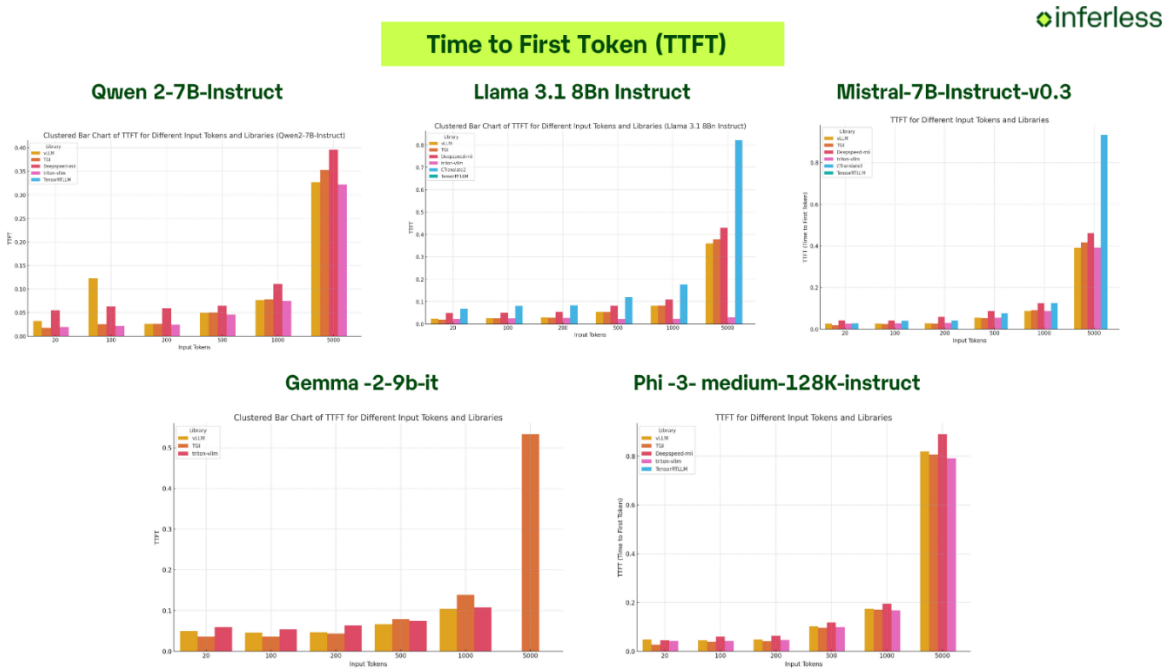
Se puede determinar que, en terminos de velocidad y rendimiento, los modelos de Mistral y Qwen2 destacan en escenarios de alta velocidad, siendo ideales para procesamiento intensivo de datos en tiempo real. Llama-3.1 y Phi-3 son consistentes en tareas multilingües y contextos largos. Pero el que destanca en recomendaciones de Tokens, es el modelo Llama-3, ya que este maneja contextos largos de manera estable, ideal para generación de texto compleja, mientras que Phi-3 se recomienda para tareas de razonamiento estructurado.

Tambien es importante recalcar que en interpretación y adaptabilidad, el modelo Qwen2 se recomienda para aplicaciones de IA con necesidades rápidas de respuesta inicial, mientras que Mistral lidera en escalabilidad para grandes cargas.

[49] Inferless. (s.f.). *Exploring LLMs Speed Benchmarks: Independent Analysis - Part 3*. Recuperado de [https://www.inferless.com/learn/exploring-llms-speed-benchmarks-independent-analysis---part-3

Figura 5

Comportamiento de los modelos en tiempo hasta el primer token (TTFT)[50]



Los modelos como Llama-3.1 mantienen un TTFT bajo y estable, lo que los hace ideales para aplicaciones en tiempo real que requieren procesar contextos largos, también tenemos que resaltar que, Qwen2 se destaca en tareas con múltiples tokens por segundo, ofreciendo respuestas iniciales rápidas, estas características posicionan ambos modelos como opciones eficientes en escenarios de alto rendimiento.

Por otro lado, Mistral equilibra la latencia y la eficiencia, mostrando mejoras significativas en infraestructuras escalables, sin embargo, los modelos menos optimizados presentan limitaciones al enfrentar inferencias prolongadas, evidenciando un aumento notable en los tiempos de respuesta bajo grandes cargas de datos.

2.8.5. Desafíos de los LLMs On-Premise

Aunque los modelos de lenguaje de gran escala (LLMs) implementados en entornos on-premise ofrecen numerosas ventajas, también presentan desafíos significativos desde una

[50] Inferless. (s.f.). *Exploring LLMs Speed Benchmarks: Independent Analysis - Part 3*. Recuperado de [https://www.inferless.com/learn/exploring-llms-speed-benchmarks-independent-analysis---part-3

perspectiva técnica, operativa y de investigación, como se ha documentado en múltiples artículos científicos y trabajos académicos. A continuación, se describen los principales desafíos respaldados por investigaciones relevantes:

2.8.5.1. Costos iniciales elevados de implementación

La puesta en marcha de un sistema on-premise requiere una inversión considerable en infraestructura de hardware, almacenamiento de datos y recursos de cómputo de alto rendimiento. Según estudios como los de Le et al. (2018) sobre la detección de URLs maliciosas mediante redes neuronales profundas, los sistemas de aprendizaje profundo como los LLMs requieren GPUs o TPUs de última generación y grandes volúmenes de memoria para procesar modelos con millones de parámetros. Estos costos iniciales pueden ser prohibitivos para empresas pequeñas y medianas, limitando su adopción, un ejemplo de ellos es "URLNet: Learning a URL Representation with Deep Learning" el cual menciona cómo los costos de infraestructura son críticos para lograr un rendimiento eficiente en tareas de procesamiento de lenguaje.

2.8.5.2. Mantenimiento y actualizaciones continuas

Los LLMs on-premise necesitan mantenimiento constante, actualizaciones de software y parches de seguridad para prevenir vulnerabilidades, además, la calibración y mejora de los modelos mediante fine-tuning requieren profesionales altamente capacitados, lo que implica un gasto adicional en recursos humanos especializados, también es importante resaltar que investigaciones de Linh et al. (2024) destacan la complejidad de mantener modelos que dependen de grandes volúmenes de datos actualizados para prevenir ataques de phishing en tiempo real.

2.8.5.3. Escalabilidad y capacidad de adaptación

Escalar un sistema on-premise implica inversiones adicionales en hardware y reconfiguración de sistemas, lo que puede ocasionar tiempos de inactividad y mayores costos, a diferencia de los servicios en la nube, donde los recursos pueden ampliarse automáticamente bajo

demanda, en un entorno on-premise el crecimiento requiere un análisis exhaustivo de los recursos disponibles y de las capacidades futuras, y la investigación de Brown et al. (2020) sobre modelos masivos como GPT-3 resalta que las implementaciones locales de grandes modelos enfrentan dificultades para escalarse sin afectar el rendimiento.

2.8.5.4. Limitaciones de acceso remoto seguro

Aunque los sistemas on-premise ofrecen un entorno seguro, el acceso remoto para empleados o sistemas distribuidos requiere el uso de canales seguros como redes privadas virtuales (VPN) o túneles cifrados, esto puede generar retrasos en la comunicación y ralentizar las aplicaciones que necesitan respuestas en tiempo real.

2.8.5.5. Complejidad en el procesamiento de grandes volúmenes de datos

El entrenamiento y ajuste de los LLMs requiere el manejo eficiente de grandes volúmenes de datos para garantizar la precisión de los resultados. Según el trabajo de María C. Maestre (2021) sobre la detección de URLs maliciosas, los modelos que manejan datos sensibles y voluminosos pueden experimentar cuellos de botella si no se optimizan adecuadamente los flujos de procesamiento, además el trabajo de NeurIPS (2020) sobre GPT-3 destacó cómo el uso intensivo de grandes lotes de datos puede causar problemas de memoria y tiempo de procesamiento.

2.8.5.6. Reto en la protección contra ataques internos

Aunque los LLMs on-premise están protegidos frente a accesos externos, las organizaciones deben implementar medidas robustas contra ataques internos o errores humanos, y los estudios sobre ingeniería social y phishing muestran cómo el acceso indebido a través de credenciales comprometidas puede derivar en incidentes de seguridad.

Los desafíos de los LLMs on-premise requieren un enfoque integral que combine una inversión inicial adecuada con estrategias de gestión de recursos, mantenimiento y medidas de seguridad sólidas. La evidencia científica sugiere que, aunque las implementaciones on-premise pueden ser complejas y costosas, su capacidad para ofrecer un entorno seguro,

privado y personalizado los convierte en una opción viable para organizaciones que priorizan la protección de datos y el control total de sus sistemas de inteligencia artificial.

2.9. Conclusiones del estado del arte

A lo largo de este estado del arte, se ha observado cómo las soluciones tradicionales de detección de phishing, basadas en listas negras y heurísticas, siguen siendo ampliamente utilizadas, pero presentan limitaciones frente a los ataques avanzados, la integración de machine learning y, más recientemente, los Modelos de Lenguaje Grandes (LLMs) han mejorado significativamente la precisión y adaptabilidad de los sistemas de detección.

El desarrollo de modelos LLMs de código abierto, capaces de ejecutarse en infraestructuras locales, ha permitido que las empresas adopten soluciones más seguras y eficientes, garantizando la privacidad de los datos, a medida que los ataques de phishing continúan evolucionando, la combinación de técnicas avanzadas de NLP, aprendizaje automático y modelos on-premise será clave para mantenerse a la vanguardia en la detección de amenazas cibernéticas en tiempo real.

Este análisis muestra que la implementación de LLMs on-premise ha permitido superar las limitaciones de los modelos en la nube en términos de privacidad y control de datos, además los modelos de código abierto más ligeros han permitido que las empresas accedan a tecnologías avanzadas de procesamiento de lenguaje sin comprometer la seguridad.

Además, los modelos de lenguaje de gran tamaño (LLMs), como GPT-3, Llama 3.2 y Phi 4, destacan por su capacidad de interpretar el lenguaje y comprender el contexto, mejorando significativamente la detección de phishing sofisticado, sin embargo, se enfrenta al desafío del alto consumo de recursos computacionales y la necesidad de grandes volúmenes de datos para su entrenamiento, y también se resaltó la eficacia de los modelos multimodales, que integran análisis textual y visual, y el uso de implementaciones on-premise para garantizar la privacidad de los datos.

Finalmente, el estado del arte subraya la importancia de un enfoque holístico para enfrentar el phishing, combinando tecnologías avanzadas de inteligencia artificial con estrategias de formación y concienciación para los usuarios, la sofisticación de los ataques exige que las

organizaciones adopten medidas proactivas y adaptativas, utilizando modelos de lenguaje avanzados que no solo detecten amenazas conocidas, sino que también puedan anticiparse a nuevas tácticas emergentes, por tanto, solo mediante la integración de IA, aprendizaje profundo y seguridad basada en comportamiento se podrá abordar eficazmente esta creciente amenaza global.

3. Objetivos concretos y metodología de trabajo

3.1. Objetivo general

Desarrollar un sistema de monitoreo y detección de phishing en tiempo real, utilizando modelos de lenguaje grandes (LLMs) de código abierto, que permitan comprender el contexto completo de los correos electrónicos, y mejorando la eficacia de la detención por medio de técnicas avanzadas de ajuste de modelos, y que además funcione dentro de la infraestructura local de la empresa, evitando transferir información corporativa a plataformas externas.

3.2. Objetivos específicos

Para alcanzar el objetivo general, se plantean los siguientes objetivos específicos:

- Realizar un análisis exhaustivo de las tecnologías actuales de detección de phishing, identificando y documentando en un plazo máximo de un mes al menos 3 soluciones actuales de detección de phishing priorizando las locales y basadas en LLMs con la elaboración de un informe detallado.
- Seleccionar y ajustar uno de los modelos LLM de código abierto, eligiendo el más adecuado según métricas de rendimiento, hardware necesario, adaptabilidad y uso on-premise, y realizar ajustes mediante prompt engineering, finalizando este proceso en un plazo máximo de 2 meses.
- Desarrollar un sistema de detección de phishing funcional, implementando un sistema que reciba y procese correos electrónicos en tiempo real, implementando proceso que incluyen desde la recepción en el servidor hasta el análisis de contenido por el LLM, logrando en un plazo no mayor a 2 meses, una tasa de procesamiento mínima del 90% de los correos recibidos.
- Evaluar el rendimiento del sistema mediante simulaciones controladas, logrando que el sistema procese al menos 100 correos simulados, alcanzando una precisión de al menos 80% en la detección de phishing, completando esta etapa en 2 meses tras el desarrollo de sistema de detección.

- Configurar el sistema para la generación de respuestas automáticas, implementando módulos que realicen acciones automáticas (alertas o bloqueos) en función del análisis, asegurando un tiempo de respuesta menor a 5 segundos por correo, finalizando esta fase en máximo 1 mes después de la evaluación.

3.3. Metodología del trabajo

La metodología para llevar a cabo este proyecto se desarrollará en varias fases, asegurando un enfoque sistemático para alcanzar los objetivos planteados:

- Revisión de la Literatura: Se realizará un análisis exhaustivo del estado del arte, evaluando las soluciones existentes para la detección de phishing, tanto en entornos en la nube como on-premise, así como los modelos LLMs de código abierto más utilizados, como Llama y Phi.
- Identificación de Requisitos: En esta fase se identificarán los requisitos del sistema, considerando tanto los requisitos funcionales y no funcionales.
- Selección y Ajuste de Modelos LLM: Se seleccionará el modelo de código abierto más adecuado (por ejemplo, Llama 3.2 o Phi 4), y se procederá a su ajuste mediante técnicas avanzadas. El Fine-Tuning permitirá adaptar el modelo a las características específicas del phishing en un entorno empresarial, mientras que prompt engineering mejorarán la precisión de las respuestas y la capacidad de recuperación de información relevante para cada caso.
- Desarrollo del Sistema: Durante esta fase se desarrollará el software para la detección de phishing, integrando los modelos LLMs en un sistema que pueda monitorear y analizar correos electrónicos en tiempo real, luego se implementarán módulos para procesar los correos electrónicos, analizar el contenido mediante el modelo ajustado, y generar alertas o bloquear mensajes sospechosos automáticamente.
- Pruebas y Evaluación: Se llevarán a cabo pruebas del sistema en un entorno simulado, se medirán las métricas de rendimiento, como la precisión y la tasa de falsos positivos, y se compararán con las soluciones tradicionales existentes, para evaluar la efectividad del sistema propuesto.

- Documentación y Resultados: Finalmente, se documentará el desarrollo completo del sistema, los resultados obtenidos durante la fase de evaluación y las recomendaciones para la implementación en entornos productivos, con ello se analizará el impacto del sistema en la mejora de la seguridad corporativa y se propondrán futuras líneas de investigación para optimizar el uso de LLMs en la detección de otras ciberamenazas.

4. Desarrollo específico de la contribución

4.1. Desarrollo de software

El desarrollo del sistema On-Premise de detección de phishing en tiempo real se llevó a cabo siguiendo un enfoque estructurado que priorizó la privacidad de los datos, la precisión en la detección de amenazas y la adaptabilidad del sistema a diferentes entornos corporativos.

Este proyecto se diseñó para abordar las limitaciones de las soluciones tradicionales basadas en listas negras y heurísticas, incorporando modelos de lenguaje grande (LLMs) de código abierto ajustados a través de técnicas avanzadas como Fine-Tuning, Prompt Engineering o Retrieval-Augmented Generation (RAG).

4.1.1. Criterios de diseño y descripción del producto

Los criterios de diseño estuvieron orientados a cumplir con los requisitos específicos del entorno empresarial y las necesidades particulares de la ciberseguridad actual. Estos criterios incluyeron:

- Privacidad de los datos: Garantizar que toda la información procesada por el sistema permaneciera dentro de la infraestructura local de la empresa, eliminando la necesidad de transferir datos a la nube.
- Escalabilidad y adaptabilidad: Diseñar un sistema que pudiera ajustarse a diferentes volúmenes de correos electrónicos y patrones de amenazas, asegurando su funcionalidad tanto en pequeñas como en grandes organizaciones.
- Eficiencia computacional: Optar por modelos LLMs más ligeros, como Llama 3.2 y Phi 4, que permiten un equilibrio entre rendimiento y requisitos de hardware accesibles, como GPUs estándar.
- Interoperabilidad: Asegurar que el sistema pudiera integrarse con plataformas de correo corporativas existentes, utilizando estándares de comunicación como IMAP y SMTP para la lectura y análisis de correos electrónicos.
- Capacidades avanzadas de detección: Incorporar técnicas de procesamiento de lenguaje natural (NLP) para comprender el contexto de los correos electrónicos y diferenciar entre mensajes legítimos y maliciosos.

El sistema desarrollado consta de los siguientes módulos principales:

- Conexión segura al servidor de correo: Un módulo que permite la integración con los servidores de correo electrónico corporativos, configurado para monitorear y recibir correos entrantes en tiempo real.
- Procesamiento y análisis con LLMs: El núcleo del sistema utiliza un modelo LLM ajustado, capaz de analizar el contenido textual de los correos electrónicos, identificando patrones y señales de phishing mediante técnicas avanzadas de NLP.
- Generación de respuestas automatizadas: En función del análisis realizado, el sistema genera respuestas o acciones automatizadas, como alertar al usuario, marcar correos como sospechosos o bloquear mensajes potencialmente maliciosos.
- Interfaz de gestión y reportes: Un panel de control que permite a los administradores monitorear el rendimiento del sistema, visualizar estadísticas de detección y configurar reglas personalizadas.

4.1.2. Identificación de requisitos

El desarrollo del sistema On-Premise de detección de phishing en tiempo real requirió una fase inicial de trabajo previo para establecer los fundamentos que guiarían su diseño e implementación. Este trabajo incluyó la identificación del problema central, el análisis del contexto habitual de uso y la recopilación de requisitos funcionales, no funcionales y de implementación. Esta etapa fue crucial para garantizar que el sistema respondiera de manera eficaz a las necesidades específicas de las organizaciones objetivo y al desafío del phishing en un entorno de ciberseguridad en constante evolución.

4.1.2.1. Trabajo previo y análisis del problema

El problema identificado fue la incapacidad de las soluciones tradicionales basadas en listas negras y heurísticas estáticas para detectar ataques de phishing cada vez más sofisticados. Estas soluciones presentan una baja capacidad de adaptación a nuevas variantes de phishing, como el spear phishing, y no comprenden el contexto del contenido analizado, lo que limita su eficacia frente a amenazas emergentes. Además, muchas herramientas modernas

dependen de servicios en la nube, lo que genera riesgos de privacidad al procesar datos corporativos sensibles fuera de las instalaciones de las empresas.

El contexto habitual de uso para el sistema desarrollado son organizaciones empresariales, instituciones educativas y gubernamentales que manejan grandes volúmenes de información sensible a través de correos electrónicos, estas entidades necesitan una solución que combine alta precisión en la detección de amenazas con la capacidad de mantener los datos dentro de su infraestructura, cumpliendo con normativas de privacidad y seguridad.

La necesidad de un sistema adaptable y robusto se hizo evidente tras el creciente auge de la inteligencia artificial para producir métodos de ataques, además de realizar un análisis exhaustivo de estudios recientes en ciberseguridad, que destacaron la creciente sofisticación de los ataques de phishing y la importancia de mantener la integridad de los datos corporativos.

4.1.2.2. Requisitos funcionales

- Monitoreo en tiempo real: El sistema debe monitorear continuamente los correos electrónicos entrantes y analizar su contenido en tiempo real para identificar patrones maliciosos.
- Capacidad de detección basada en LLMs: Utilizar modelos de lenguaje grande (LLMs) ajustados, como Llama 3.2 o Phi 4, para comprender el contexto y detectar señales de phishing más allá de palabras clave simples.
- Clasificación de amenazas: Proporcionar una clasificación clara de los correos electrónicos como seguros (Corporativos), sospechosos (Personales) o maliciosos (Phishing).
- Generación de acciones automáticas: Implementar respuestas automáticas como alertas, movimientos de correos o notificaciones a los administradores en función del análisis realizado.
- Panel de administración: Ofrecer una interfaz gráfica que permita gestionar configuraciones, visualizar estadísticas y generar reportes detallados.

4.1.2.3. Requisitos no funcionales

- Privacidad de los datos: Procesar todos los correos electrónicos localmente para evitar la transferencia de información sensible a servidores externos.
- Escalabilidad y rendimiento: Garantizar que el sistema pueda manejar un creciente volumen de correos electrónicos sin degradar su rendimiento.
- Compatibilidad: Integrarse con plataformas de correo existentes mediante protocolos estándar como IMAP y SMTP.
- Seguridad del sistema: Proteger el sistema contra accesos no autorizados y garantizar la integridad de los datos procesados.

4.1.2.4. Requisitos de implementación

- Modelos de código Abierto: Seleccionar modelos LLMs eficientes, como Llama 3.2 o Phi 4, que sean compatibles con infraestructuras locales.
- Técnicas de ajuste: Aplicar Fine-Tuning y Prompt Engineering para personalizar los modelos a las características específicas del phishing en el entorno empresarial.
- Infraestructura tecnológica: Utilizar hardware accesible, como GPUs estándar, y herramientas de desarrollo como Python y frameworks como TensorFlow o PyTorch.
- Protocolos de comunicación Segura: Implementar SSL/TLS para proteger las conexiones entre el sistema y los servidores de correo.
- Documentación técnica: Desarrollar una guía completa que detalle la instalación, configuración y uso del sistema.

4.1.2.5. Método para la identificación de requisitos

El proceso de identificación de requisitos se realizó mediante la revisión de material informativos de expertos en ciberseguridad, análisis de las necesidades empresariales y revisión de literatura académica y técnica, además, se consideraron los datos de estudios previos sobre las limitaciones de los sistemas de detección tradicionales y las capacidades avanzadas de los modelos de lenguaje grande.

Esta identificación de requisitos asegura que el sistema desarrollado sea práctico, eficaz y capaz de abordar los desafíos específicos del phishing en entornos corporativos, estableciendo una base sólida para su diseño e implementación.

4.1.2.6. Contexto de uso

El sistema está diseñado principalmente para entornos corporativos que manejan información sensible a través de correos electrónicos, sectores como el financiero, la salud y la educación presentan un alto riesgo de sufrir ataques de phishing, y este sistema está orientado a proteger estas industrias mediante una solución eficiente, privada y escalable.

Adicionalmente, el sistema es aplicable en pequeñas y medianas empresas que necesitan herramientas robustas pero accesibles para fortalecer su postura de ciberseguridad.

Este apartando sea establecen las bases del desarrollo, asegurando que las decisiones de diseño e implementación estén alineadas con los objetivos y las necesidades del usuario final.

4.1.2.7. Alcances y limitaciones

El software desarrollado tiene como objetivo principal la detección y clasificación de correos electrónicos potencialmente maliciosos mediante el uso de modelos de lenguaje de gran escala (LLMs) de código abierto, el sistema es capaz de analizar correos en tiempo real, asignando un nivel de riesgo basado en la interpretación contextual proporcionada por los modelos implementados, además, automatiza acciones defensivas como el bloqueo, eliminación o cuarentena de correos sospechosos (inicialmente para la prueba de concepto se estableció que el sistema moviera los correos a carpetas), garantizando una respuesta ágil ante amenazas de phishing.

Su principal fortaleza es que la solución opera completamente dentro de la infraestructura local de la empresa, asegurando que no se transfiera información sensible a plataformas externas y manteniendo así la confidencialidad de los datos.

Entre las principales limitaciones del software se encuentra su dependencia de la infraestructura local, lo que puede afectar el rendimiento si los recursos disponibles no son

suficientes para procesar grandes volúmenes de correos o ejecutar modelos LLM de alta complejidad.

Es importante resaltar que la precisión en la detección de amenazas está sujeta a las capacidades y la calidad de los modelos de lenguaje utilizados; es decir, el software facilita la integración y el análisis, pero la exactitud de los resultados depende directamente del modelo seleccionado, pero sin ninguna duda la principal limitación es la necesidad de ajustes continuos en los modelos y parámetros para adaptarse a nuevas tácticas de phishing, lo que requiere intervención técnica especializada.

4.1.3. Descripción de la herramienta software desarrollada

La herramienta software desarrollada es un sistema on-premise diseñado para la detección de phishing en tiempo real, priorizando la privacidad de los datos y la eficacia en la detección de amenazas, integrando modelos de lenguaje grande (LLMs) de código abierto ajustados a las necesidades particulares y al contexto mediante técnicas avanzadas como Fine-Tuning y Prompt Engineering, lo que permite una solución adaptable, escalable y segura frente a los crecientes desafíos del phishing.

El proceso de desarrollo siguió un enfoque estructurado en fases, cada una con hitos específicos que aseguraron la calidad y funcionalidad del producto final.

4.1.3.1. Actores y sus niveles de confianza

En el sistema de monitoreo y detección de phishing, cada actor en el sistema tiene un rol específico, con permisos y accesos diferenciados, lo que asegura un manejo adecuado de la información sensible, además, la asignación de identificadores únicos a cada actor facilita la trazabilidad, la gestión de accesos y la aplicación de políticas de seguridad adecuadas, y esto es especialmente importante dado que el sistema opera en la infraestructura local de la empresa, donde la protección de los datos corporativos es prioritaria.

A continuación, se listan los actores involucrados en el sistema, junto con su descripción y nivel de confianza.

Tabla 4

Actores y sus niveles de confianza

Id	Nombre	Descripción	Nivel de Confianza
1	Usuario Anónimo	Persona que intenta interactuar con el sistema sin autenticarse, no tienen ningún permiso para acceder a funciones.	Bajo
2	Empleado Autenticado	Usuario corporativo que accede al sistema tras autenticarse, puede revisar la información de sus correos etiquetados.	Medio
3	Administrador del Sistema	Responsable de la configuración, monitoreo y mantenimiento del sistema, tiene funciones de gestión general y modelos LLM.	Alto
4	Administrador de Base de Datos	Encargado de la gestión y mantenimiento de la base de datos, asegurando integridad y disponibilidad de datos.	Alto
5	Proceso de Análisis Automático	Proceso interno que analiza correos en tiempo real utilizando modelos LLM y clasifica posibles amenazas.	Medio
6	Proceso de Acciones Automatizadas	Ejecuta respuestas automáticas (bloqueo, eliminación o cuarentena) en base a la clasificación de los correos.	Medio
7	Proveedor de Autenticación Local	Servicio interno encargado de la autenticación de usuarios mediante protocolos seguros dentro de la empresa.	Alto

4.1.3.2. Activos de información

El software desarrollado para la detección de phishing en correos electrónicos utiliza una variedad de activos clave que garantizan su correcto funcionamiento y la seguridad de la información procesada, entre los activos primarios se encuentran el servicio de análisis de correos, que automatiza la recepción y evaluación de correos electrónicos mediante modelos de lenguaje LLM, y los datos de correos electrónicos, que comprenden información como el remitente, destinatario, asunto y contenido del mensaje, además, los resultados de análisis y los datos de patrones de phishing permiten identificar y clasificar amenazas potenciales, proporcionando información valiosa para la toma de decisiones y la implementación de medidas de seguridad.

En cuanto a los activos secundarios, destacan la base de datos relacional, que almacena de forma segura toda la información del sistema, y la infraestructura de hardware, compuesta por servidores con capacidades de procesamiento avanzado que permiten ejecutar modelos de IA de manera eficiente y finalmente, el panel de control y reportes ofrece una interfaz visual para que los administradores puedan gestionar el sistema, revisar resultados y generar informes.

A continuación, se listan los activos involucrados en el sistema, junto con su descripción y los actores relacionados con el activo.

Tabla 5

Activos de información

Tipo	Id	Nombre	Descripción	Actores
Activos primarios de información y servicios	1	Servicio de análisis de correos	Servicio que permite la recepción y análisis de correos electrónicos utilizando modelos LLM.	(1), (2), (3), (5)
	2	Datos de correos electrónicos	Información contenida en los correos electrónicos, incluyendo remitente, destinatario, asunto y contenido.	(2), (3), (4), (5)
	3	Resultados de análisis	Clasificación y resultados generados por los modelos LLM sobre la naturaleza de los correos (Phishing, etc.).	(2), (4), (5)
	4	Datos de patrones de phishing	Información sobre patrones detectados en correos que indican intentos de phishing.	(2), (4), (5)
Activos secundarios de sistema y aplicación	5	Base de datos relacional (MySQL)	Almacena correos, resultados de análisis, patrones de phishing y acciones tomadas por el sistema.	(5), (4)
	6	Infraestructura de hardware	Servidores y equipos con capacidades de procesamiento de GPU necesarios para la operación del sistema.	(5), (4)
	7	Panel de control y reportes	Interfaz que permite a los administradores visualizar resultados de análisis y configurar el sistema.	(4), (5)

4.1.3.3. Matriz de control de acceso a los datos

La matriz de control de acceso a los datos es esencial para garantizar la seguridad y la correcta manipulación de la información en el software de detección de phishing, en esta se define los permisos y privilegios (Crear [C], Leer [R], Actualizar [U], Eliminar [D]) que los actores tienen sobre diferentes tipos de datos, asegurando que solo los usuarios autorizados puedan acceder o modificar la información sensible.

A continuación de muestra la matriz de control de acceso a los datos, relacionado los datos (activos) con los permisos que pueden tener los usuarios.

Tabla 6

Matriz de control de acceso a los datos

Datos	Administrador	Usuario final	Proceso de análisis	API de integración
Datos de correos electrónicos	C, R, U, D	R	C, R, U, D	R
Resultados de análisis	C, R, U, D	R	C, R, U, D	R
Datos de patrones de phishing	C, R, U, D	-	C, R, U, D	R
Datos de usuarios	C, R, U, D	R, U	-	-
Datos de autenticación	C, R, U, D	C, R, U	-	-
Configuración del sistema	C, R, U, D	-	-	-
Registros y monitoreo de eventos	C, R, U, D	-	R	-

4.1.3.4. Puntos de entrada de la aplicación

En este software de detección de correos phishing mediante modelos LLM, estos puntos incluyen interfaces web para el inicio de sesión, carga y análisis de correos, así como APIs que permiten la integración con servicios externos o la automatización de procesos, estos puntos

de entrada son críticos desde la perspectiva de seguridad, ya que son susceptibles a intentos de acceso no autorizado, inyecciones de datos maliciosos o explotación de vulnerabilidades.

Tabla 7

Puntos de entrada de la aplicación

ID	Nombre	Descripción	Actores
1	Puerto HTTPS	El sistema es accesible exclusivamente a través de HTTPS para garantizar comunicaciones cifradas.	(1), (2), (3), (4)
2	Interfaz gráfica WEB	Ventana principal que permite la interacción del usuario con las funcionalidades del software.	(2), (4)
3	API REST de recepción de correos	Permite la integración con servicios externos para la recepción automática de correos.	(3), (4)
4	API REST de resultados de análisis	Permite la consulta de resultados de análisis de phishing por aplicaciones externas.	(4)
5	Formulario de autenticación	Página para el inicio de sesión de usuarios mediante credenciales seguras.	(2), (5)
6	Configuración del sistema	Sección exclusiva para administradores donde se gestionan usuarios y configuraciones.	(1)

4.1.3.5. Puntos de salida de la aplicación

En el contexto de este software de detección de correos phishing mediante modelos LLM, estos puntos de salida de la aplicación incluyen la visualización de resultados de análisis de correos, reportes de riesgos detectados, notificaciones automáticas sobre correos sospechosos, y la exportación de datos para auditorías o análisis adicionales.

De deben establecer estos puntos, ya que son críticos en términos de seguridad, y podrían convertirse en fuentes de filtración de información confidencial si no se gestionan adecuadamente.

Tabla 8

Puntos de salida de la aplicación

ID	Nombre	Descripción	Actores
1	Resultados de Análisis	Muestra los resultados del análisis de correos para identificar amenazas de phishing.	(2), (4)
2	Exportación de Reportes	Permite a los administradores y usuarios autorizados exportar reportes detallados.	(1), (2)
3	Notificaciones por Correo Electrónico	Envío de notificaciones automáticas a los usuarios sobre la detección de amenazas.	(2), (4)
4	Panel de Monitoreo de Eventos	Visualización en tiempo real de la actividad del sistema y de los análisis realizados.	(1)
5	API de Consulta de Resultados	Permite que aplicaciones externas consulten los resultados de análisis.	(4)
6	Informes de Auditoría	Generación de informes para la revisión y auditoría del sistema.	(1), (5)
7	Log de Eventos del Sistema	Registro detallado de las actividades y eventos del sistema.	(1), (5)

4.1.3.6. Diseño de la arquitectura del sistema

El sistema propuesto está diseñado para operar como una solución on-premise que detecta phishing en tiempo real mediante el uso de modelos de lenguaje grande (LLMs) de código abierto.

La arquitectura está compuesta por tres componentes principales que trabajan de manera integrada.

Base de datos relacional (MySQL):

- Almacena correos electrónicos, URLs analizadas, patrones de phishing, resultados de análisis y las acciones tomadas por el sistema.
- Garantiza una gestión eficiente y segura de los datos mediante relaciones bien definidas en el modelo entidad-relación.

- La base de datos relacional MySQL almacena y organiza la información de correos electrónicos, resultados de análisis, patrones de phishing y acciones automatizadas.
- Gestiona las relaciones entre tablas como usuarios, modelos LLM, clasificaciones y acciones, facilitando consultas complejas y reportes.
- Asegura la integridad de los datos mediante claves primarias, foráneas y restricciones, garantizando que la información sea coherente y precisa.

Interfaz de proceso automático Python (Cliente de automatización):

- El programa se conecta e interactúa con modelos de lenguaje LLM mediante la biblioteca ollama para analizar y clasificar correos electrónicos en categorías como phishing, corporativo o personal.
- Integra la conexión con servidores de correo usando IMAP a través de imaplib, permitiendo la recuperación, análisis y manipulación de correos electrónicos en tiempo real.
- Desarrolla una interfaz gráfica con tkinter, facilitando el inicio, la detención del proceso y la visualización de resultados en un área de texto para el monitoreo en tiempo real.
- Implementa threading para ejecutar procesos en segundo plano, asegurando que el análisis y la interacción con APIs no bloqueen la interfaz gráfica.
- Utiliza APIs REST mediante requests para registrar y actualizar datos en la base de datos, manteniendo la información sincronizada y actualizada tras cada análisis.
- Ejecuta acciones automatizadas, como mover correos a carpetas específicas o registrar patrones de phishing, basándose en los resultados de los modelos LLM.
- Incluye manejo de errores y medidas básicas de seguridad, garantizando la continuidad del proceso y la protección de datos sensibles durante la comunicación con servidores externos.

Plataforma de gestión, configuración y reporte (Laravel):

- Permite a los usuarios visualizar reportes en tiempo real sobre correos analizados, URLs detectadas y patrones de phishing encontrados.

- Proporciona una interfaz de usuario intuitiva con roles diferenciados (administrador y usuario), que incluye la capacidad de configurar reglas personalizadas y consultar históricos de análisis.
- La plataforma, desarrollada en Laravel, ofrece una API RESTful que permite la creación, modificación y consulta de correos, análisis de resultados y acciones automatizadas.
- Gestiona la autenticación y autorización de usuarios mediante el sistema de control de acceso integrado de Laravel, asegurando que solo los usuarios autorizados interactúen con la aplicación.
- Administra la base de datos utilizando Eloquent ORM para manejar relaciones complejas entre modelos como usuarios, correos, análisis y patrones de phishing.
- Facilita la carga y clasificación de correos mediante rutas y controladores que reciben datos del cliente Python y los almacenan en la base de datos.
- Integra vistas administrativas con Blade para la gestión visual de correos, patrones detectados y modelos LLM, permitiendo una supervisión eficiente del sistema.
- Implementa validaciones de datos tanto en el lado del servidor como en el cliente, garantizando la integridad y consistencia de la información recibida.
- Utiliza middleware y políticas de seguridad para proteger las rutas API y asegurar que las acciones automatizadas se ejecuten bajo las condiciones adecuadas.

4.1.3.7. Diseño de la base de datos

El diseño de la base de datos se fundamenta en un modelo entidad-relación que incluye entidades clave como correo, patron_phishing, analisis_resultado, modelo_llm y usuario_correo, estas entidades están interconectadas para garantizar la trazabilidad, integridad y escalabilidad del sistema, permitiendo un seguimiento detallado desde la recepción del correo hasta la acción automatizada basada en el análisis.

Se puede identificar la estructura del modelo a partir de la siguiente organización:

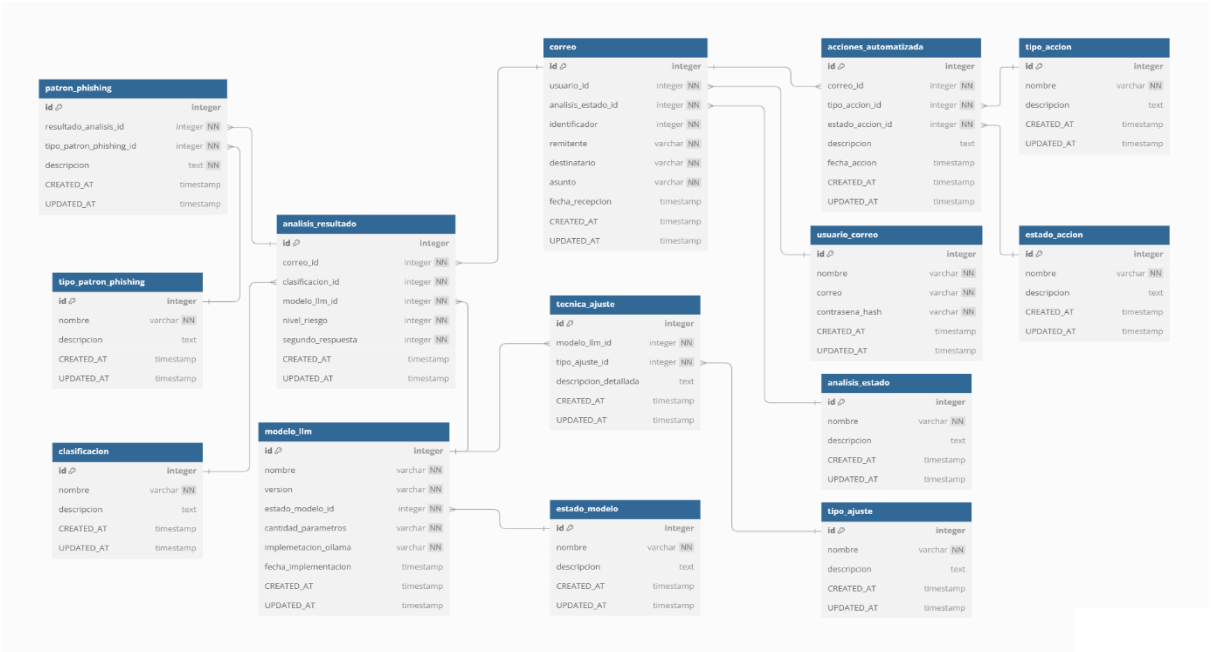
- Los correos electrónicos gestionados en la tabla correo están vinculados a resultados específicos en analisis_resultado, lo que permite evaluar cada mensaje recibido en función de criterios predefinidos y el análisis de modelos LLM.

- Los patrones de phishing almacenados en patron_phishing se asocian con los resultados de análisis, facilitando el rastreo de cómo se detectaron las amenazas y qué características comunes fueron identificadas en los correos maliciosos.
- La tabla modelo_llm permite evaluar y gestionar el desempeño de los modelos de lenguaje utilizados, registrando información como la versión, cantidad de parámetros y tiempos de respuesta durante el análisis.
- Las acciones automatizadas en acciones_automatizada se relacionan directamente con los resultados del análisis, permitiendo que el sistema responda automáticamente a correos clasificados como phishing o sospechosos mediante la aplicación de acciones predefinidas.
- La entidad usuario_correo gestiona la información de los usuarios que reciben y envían correos electrónicos, garantizando que cada correo esté asociado a un usuario específico para un control más detallado.
- Este diseño relacional facilita la ejecución de consultas complejas, asegura la integridad de los datos mediante claves primarias y foráneas, y proporciona la flexibilidad necesaria para incorporar futuras expansiones o mejoras en el sistema.

Por tanto, el modelo entidad-relación (ER) será el siguiente (elaboración propia):

Figura 6

Modelo Entidad-Relación (ER)



El diseño relacional con MySQL permite gestionar grandes volúmenes de datos de manera estructurada y eficiente. Además:

- Facilita la ejecución de consultas complejas para generar reportes detallados en tiempo real.
- Asegura la integridad de los datos mediante restricciones como claves primarias y foráneas.
- Proporciona escalabilidad para incorporar nuevas funcionalidades, como análisis de documentos o integración con otras fuentes de datos.

El modelo entidad-relación diseñado para el sistema de monitoreo y detección de phishing en tiempo real es el núcleo de la arquitectura de datos del proyecto, permitiendo una gestión estructurada y escalable de la información, este modelo se centra en cinco entidades principales: correo_electronico, url, patron_phishing, resultado_analisis, modelo_llm y usuarios. La entidad correo_electronico almacena la información esencial sobre los correos analizados, como remitente, destinatario, asunto, cuerpo del mensaje, estado de análisis y el resultado asociado.

Por otro lado, la entidad url gestiona las URLs sospechosas detectadas en tiempo real, incluyendo atributos como el dominio, el estado del análisis y el resultado correspondiente, ambas entidades se relacionan con resultado_analisis, una tabla que centraliza los resultados generados por el sistema, permitiendo vincular los correos y URLs analizados con el modelo de lenguaje (LLM) que los procesó, el nivel de riesgo asignado, y las acciones tomadas por el sistema, como bloquear, marcar como sospechoso o permitir.

Además, la entidad patron_phishing complementa el sistema al almacenar los patrones de phishing detectados, como frases comunes, estructuras maliciosas de URLs o características técnicas específicas de correos fraudulentos, estos patrones están directamente relacionados con los resultados de análisis, permitiendo rastrear qué patrones fueron utilizados para identificar amenazas y evaluar su efectividad.

Por otra parte, la entidad modelo_llm registra información detallada sobre los modelos de lenguaje utilizados, incluyendo su nombre, versión, fecha de implementación y desempeño, esto no solo ayuda a monitorear la calidad de los análisis realizados, sino que también permite realizar ajustes o actualizaciones en los modelos según sea necesario.

Finalmente, la tabla usuarios gestiona a los diferentes actores del sistema, diferenciando roles como administradores, responsables de configurar y supervisar el sistema, y usuarios, que acceden a reportes y estadísticas de análisis.

A continuación, se plantea se detallan los campos de las principales tablas.

Entidad – Usuarios de correo electrónico

La tabla usuario_correo almacena los datos de los usuarios registrados en el sistema que reciben y gestionan correos electrónicos, esta tabla contiene información básica como el nombre del usuario, su correo electrónico y la contraseña cifrada. Esta tabla se relaciona con la tabla correo a través del campo usuario_id, permitiendo identificar qué usuario está asociado a cada correo recibido, los campos que contiene son los siguientes:

- id: Identificador único del usuario.
- nombre: Nombre completo del usuario.
- correo: Dirección de correo del usuario.
- contrasena_hash: Contraseña cifrada para asegurar la autenticación.

Entidad - Correo

La entidad correo es el núcleo del sistema, donde se almacenan los correos electrónicos que serán analizados, esta tabla incluye detalles como el remitente, destinatario, asunto, fecha de recepción, y un identificador único para cada correo, además, cada correo tiene un estado de análisis vinculado mediante analisis_estado_id, lo que permite saber en qué etapa del análisis se encuentra el correo, los campos que contiene son los siguientes:

- id: Identificador único del correo.
- usuario_id: Clave foránea que vincula el correo al usuario que lo recibió.
- analisis_estado_id: Indica el estado del análisis del correo (pendiente, en proceso, completado).
- identificador: Identificador único del correo dentro del sistema.

- remitente: Dirección del remitente del correo.
- destinatario: Dirección del destinatario del correo.
- asunto: Asunto del correo recibido.
- fecha_recepcion: Fecha y hora de recepción del correo.

Entidad - Modelo LLM

La tabla modelo_llm registra los modelos de lenguaje de gran tamaño (LLM) utilizados para analizar los correos electrónicos, cada modelo tiene un estado asociado (estado_modelo_id), indicando si está activo o inactivo, además, se almacena la cantidad de parámetros que maneja el modelo y la fecha en que fue implementado en el sistema, los campos que contiene son los siguientes:

- id: Identificador único del modelo.
- nombre: Nombre del modelo LLM.
- version: Versión del modelo utilizado.
- estado_modelo_id: Estado actual del modelo (activo, inactivo).
- cantidad_parametros: Número de parámetros que maneja el modelo.
- implementacion_ollama: Información sobre la implementación específica del modelo.
- fecha_implementacion: Fecha de implementación del modelo en el sistema.

Entidad - Análisis resultado

La tabla analisis_resultado centraliza los resultados obtenidos tras el análisis de los correos, la tabla de análisis de resultados está vinculada directamente con las tablas correo y modelo_llm para identificar qué correo fue analizado por qué modelo y con qué resultados. Incluye la clasificación del correo (corporativo, personal, phishing), el nivel de riesgo detectado y el tiempo de respuesta del modelo, los campos que contiene son los siguientes:

- id: Identificador único del análisis.

- correo_id: Clave foránea que vincula el análisis al correo correspondiente.
- clasificacion_id: Clasificación asignada al correo (corporativo, personal, phishing).
- modelo_llm_id: Modelo LLM que realizó el análisis.
- nivel_riesgo: Nivel de riesgo asignado al correo.
- segundo_respuesta: Tiempo en segundos que tardó el modelo en realizar el análisis.

Entidad - Patrón phishing

La tabla patron_phishing almacena los patrones de phishing identificados en los correos analizados, estos patrones permiten rastrear las características comunes de los correos maliciosos detectados, como estructuras de texto, enlaces sospechosos, y comportamientos típicos de phishing, además, cada patrón está vinculado a un resultado de análisis específico, los campos que contiene son los siguientes:

- id: Identificador único del patrón.
- resultado_analisis_id: Clave foránea que vincula el patrón con el resultado de análisis donde se detectó.
- tipo_patron_phishing_id: Tipo de patrón identificado (enlace malicioso, contenido sospechoso, etc.).
- descripcion: Descripción detallada del patrón detectado.

Entidad - Acciones automatizadas

La tabla acciones_automatizada registra las acciones que el sistema realiza automáticamente después de analizar los correos, estas acciones pueden incluir mover el correo a una carpeta específica, eliminarlo o marcarlo como seguro, la automatización permite una respuesta rápida ante amenazas detectadas, los campos que contiene son los siguientes:

- id: Identificador único de la acción.
- correo_id: Clave foránea que vincula la acción al correo correspondiente.

- tipo_accion_id: Tipo de acción realizada (mover a alguna de las carpetas indicadas, eliminar, marcar como seguro).
- estado_accion_id: Estado de la acción (pendiente, completada).
- descripcion: Descripción de la acción realizada.
- fecha_accion: Fecha y hora en que se realizó la acción.

Entidad - Clasificación

La tabla clasificacion define las categorías en las que un correo puede ser clasificado tras el análisis, estas categorías son esenciales para la gestión y organización de los correos, permitiendo aplicar medidas de seguridad específicas, los campos que contiene son los siguientes:

- id: Identificador único de la clasificación.
- nombre: Nombre de la clasificación (corporativo, personal, phishing).
- descripcion: Descripción detallada de la clasificación.

Este modelo relacional asegura la integridad de los datos mediante claves primarias y foráneas que conectan las entidades de manera lógica, facilitando la trazabilidad de los análisis y la generación de reportes en tiempo real, además, el diseño es altamente escalable, lo que permite incorporar nuevas funcionalidades, como la detección de documentos maliciosos o el análisis de archivos adjuntos, sin modificar significativamente la estructura base.

La interrelación entre entidades como correos, modelo_llm, analisis_resultado, y acciones_automatizadas garantiza que cada correo recibido sea rastreado desde el análisis hasta la acción final tomada, este enfoque asegura que el sistema sea eficiente, seguro y capaz de adaptarse a nuevas necesidades en el futuro.

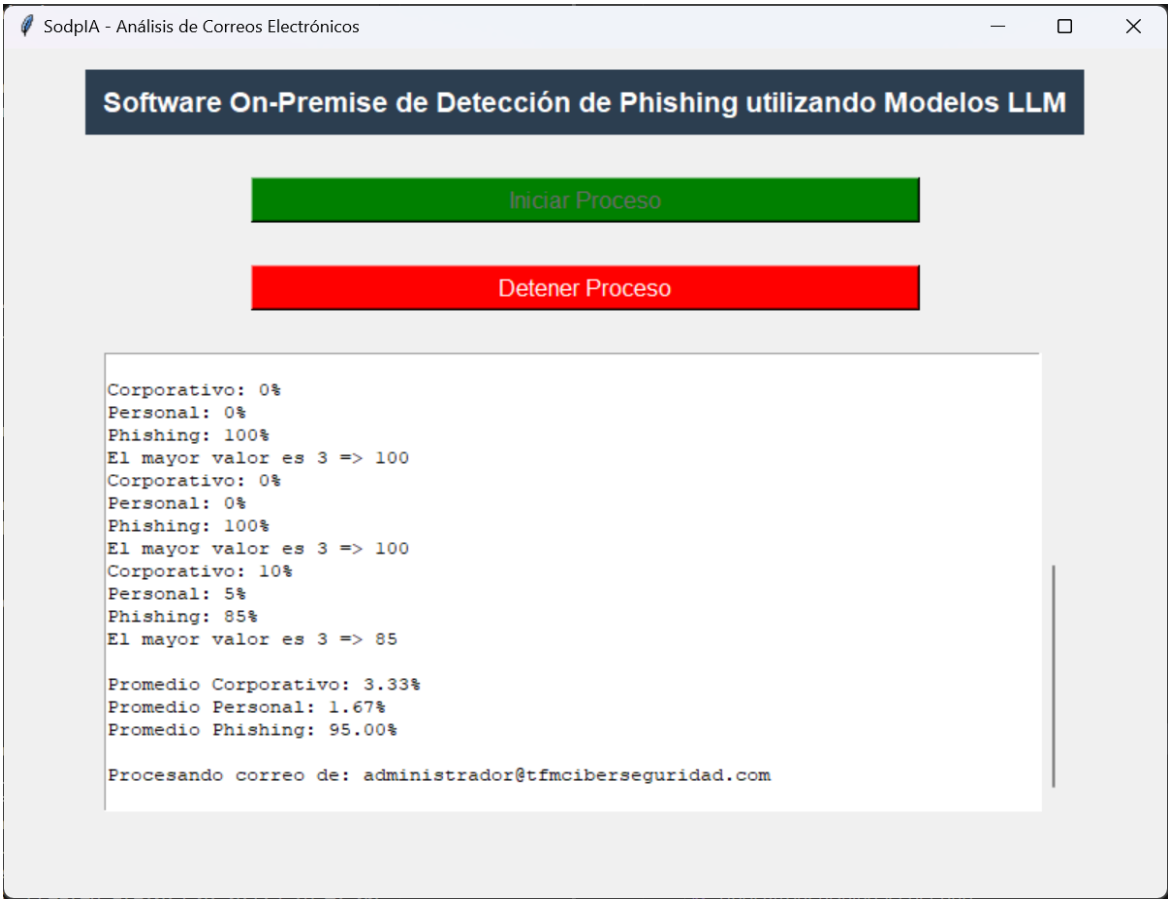
La integración de este modelo con tecnologías como MySQL para almacenamiento, Python para procesamiento y análisis, y Laravel para la visualización de reportes garantiza un flujo de datos eficiente y un acceso optimizado a la información crítica, consolidando al sistema como una herramienta robusta en la lucha contra el phishing.

4.1.3.8. Interfaz de proceso automático en Python (Cliente de automatización)

La arquitectura del programa está diseñada para interactuar con modelos LLM, gestionar correos electrónicos en tiempo real y comunicar resultados a través de APIs, todo mientras proporciona una interfaz gráfica sencilla para el control de activación y desactivación de las funcionalidades.

Figura 7

Interfaz Python



La interfaz de proceso automático en Python conecta con modelos LLM para analizar y clasificar correos electrónicos en categorías como phishing, personal o corporativo usando la biblioteca ollama, pero antes se Integra con servidores de correo mediante imaplib para recuperar y manipular correos en tiempo real, mientras que tkinter proporciona una GUI interactiva para iniciar/detener procesos y visualizar resultados.

El uso de threading permite que los análisis se ejecuten en segundo plano sin bloquear la interfaz, y requests facilita la comunicación con APIs REST para registrar y actualizar datos en el servidor.

El programa analiza resultados calculando promedios de clasificaciones, ejecuta acciones automatizadas como mover correos o generar alertas, y maneja errores mediante registros y mensajes para garantizar la seguridad y eficiencia del proceso.

A continuación, se detallan las características más importantes:

- Conexión e integración con modelos LLM (Large Language Models)

El programa está diseñado para conectarse e interactuar con modelos de lenguaje LLM a través de la biblioteca ollama, esta integración permite analizar y clasificar el contenido de correos electrónicos en categorías específicas como phishing, corporativo o personal, además, los modelos son evaluados en función de su desempeño y los resultados se utilizan para generar acciones automatizadas dentro del sistema.

- Integración con el servidor de correos (IMAP/SMTP)

El software se conecta a servidores de correo mediante el protocolo IMAP utilizando la biblioteca imaplib, esta integración permite la recuperación automática de correos electrónicos, el análisis de su contenido y la manipulación de estos, como moverlos entre carpetas según su clasificación y el sistema también soporta operaciones como la autenticación de usuarios de correo y la gestión de bandejas de entrada.

- Interfaz gráfica de usuario (GUI) con tkinter

Se utiliza tkinter para desarrollar una interfaz gráfica intuitiva que permite al usuario iniciar y detener el proceso de análisis de correos electrónicos, además, la GUI incluye botones para controlar la ejecución, un área de texto para mostrar resultados en tiempo real y elementos visuales que informan el estado del sistema, facilitando la interacción del usuario con el programa sin necesidad de conocimientos técnicos avanzados.

- Gestión de procesos en segundo plano (threading)

El programa implementa threading para gestionar la ejecución de procesos en segundo plano, asegurando que el análisis de correos y la comunicación con APIs no bloqueen la interfaz gráfica, permitiendo una experiencia fluida, donde el usuario puede seguir interactuando con la aplicación mientras los análisis y tareas automatizadas se ejecutan de forma continua y eficiente.

- Comunicación con APIs REST para registro y actualización de datos

El sistema realiza operaciones CRUD (crear, leer, actualizar, eliminar) en una base de datos mediante llamadas a APIs REST, utilizando la biblioteca requests, el programa registra los correos analizados, actualiza sus estados y almacena los resultados de los análisis realizados por los modelos LLM.

- Análisis de resultados y acciones automatizadas

El programa no solo clasifica los correos, sino que también calcula promedios de las categorías analizadas por diferentes modelos, ejecutando acciones automatizadas como mover correos a carpetas específicas y registrar patrones de phishing.

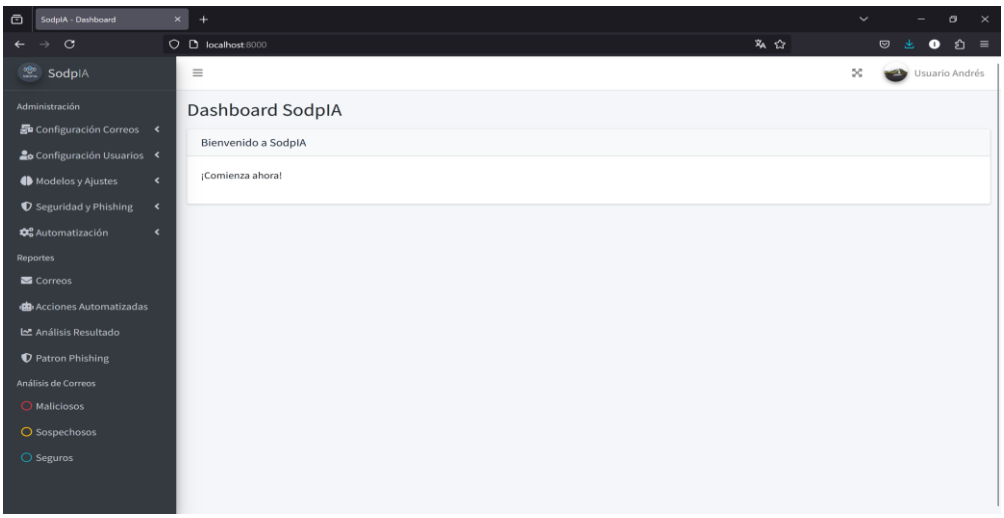
4.1.3.9. Plataforma de gestión, configuración y reporte (Laravel):

La plataforma desarrollada en Laravel actúa como el núcleo de la arquitectura web, proporcionando una API RESTful robusta que facilita la interacción entre el cliente de automatización en Python y la base de datos MySQL.

La estructura de migraciones en Laravel permite la creación eficiente de tablas para gestionar entidades clave como correos, análisis de resultados, patrones de phishing y acciones automatizadas, estos modelos son gestionados mediante Eloquent ORM, que facilita el manejo de relaciones complejas y asegura la integridad de los datos.

Figura 8

SodpIA - Plataforma de gestión, configuración y reporte



Los controladores en Laravel Http\Controllers\Api, son responsables de manejar las solicitudes entrantes y las operaciones CRUD (crear, leer, actualizar y eliminar) para cada entidad del sistema, por ejemplo, CorreoController.php gestiona las operaciones relacionadas con los correos electrónicos, mientras que AnalisisResultadoController.php y AccionesAutomatizadaController.php manejan los resultados de análisis y las acciones automatizadas, respectivamente.

La autenticación y autorización de usuarios se gestionan a través del sistema integrado de Laravel, utilizando políticas de seguridad y middleware para proteger las rutas de la API, lo que garantiza que solo los usuarios autorizados puedan acceder a funciones críticas, como la gestión de configuraciones de correos, usuarios y modelos de aprendizaje automático (LLM).

La interfaz administrativa, construida con Blade, ofrece una vista clara y estructurada de la información crítica del sistema, esto dashboards permiten a los administradores visualizar de forma intuitiva los correos analizados, los patrones de phishing detectados y los resultados de los modelos LLM, todas las vistas desarrolladas incluyen funcionalidades para exportar datos, filtrar resultados y monitorear acciones automatizadas, proporcionando herramientas para la supervisión y toma de decisiones.

Además, la plataforma implementa validaciones de datos tanto en el lado del cliente como en el servidor, asegurando que la información recibida sea precisa y consistente, de esta manera las vistas permiten la interacción directa con los datos, mientras que el backend gestiona la lógica de negocio y garantiza la integridad de las operaciones.

4.1.3.10.Integración del modelo en el desarrollo del software

La integración de los modelos con el desarrollo esta dado por los lenguajes de programación y tecnologías utilizados en el proceso, teniendo como principales elementos a Python, los LLMs y el framework Laravel.

Python y LLMs: Python realiza la solicitud de los correos directamente al servidor de correo, almacenando la información en la base de datos en la tabla correo_electronico, posteriormente, utilizando una biblioteca local en Ollama, Python envía al modelo LLM toda la información disponible del correo electrónico, el modelo procesa la solicitud y clasifica el

correo según su probabilidad de ser phishing, corporativo o personal, y la respuesta es recibida nuevamente por Python y los resultados se registran en la tabla resultado_analisis para su posterior consulta.

Laravel y Reportes: La plataforma Laravel accede a los datos almacenados en MySQL mediante consultas directas o a través de APIs RESTful, los usuarios pueden visualizar los resultados de los análisis, los patrones detectados y las estadísticas de manera dinámica y accesible, además, Laravel genera reportes detallados que permiten un análisis exhaustivo de los correos electrónicos clasificados y las tendencias de posibles amenazas detectadas.

Por tanto, las funcionalidades de cada tecnología son las siguientes.

Funciones de Python:

- Realiza la solicitud de los correos directamente al servidor de correo y almacena la información en la tabla correo_electronico de la base de datos.
- Preprocesa la información obtenida del correo electrónico (encabezados, cuerpo del mensaje, URLs, adjuntos, etc.) para enviarla al modelo LLM.
- Utiliza una biblioteca local (Ollama) para enviar los datos al modelo LLM y recibir la respuesta.
- Procesa la respuesta del modelo LLM, que contiene la clasificación del correo (phishing, corporativo o personal) y la probabilidad correspondiente.
- Almacena los resultados procesados en la tabla resultado_analisis para su posterior consulta y visualización.
- Automatiza los flujos de trabajo y gestiona la sincronización con la base de datos.

Funciones del Modelo LLM:

- Recibe la solicitud con la información detallada del correo electrónico.
- Procesa los datos utilizando un modelo de lenguaje grande (LLM) para analizar el contenido, los enlaces y otros patrones relevantes.
- Clasifica el correo electrónico de acuerdo con su probabilidad de ser phishing, corporativo o personal.
- Devuelve la clasificación y el nivel de confianza en la predicción a Python para su posterior almacenamiento y presentación.

Funciones de Laravel:

- Consume los datos almacenados en la base de datos MySQL a través de consultas directas o APIs RESTful.
- Proporciona una interfaz gráfica dinámica y accesible para que los usuarios puedan visualizar los resultados de los análisis.
- Muestra la clasificación de los correos electrónicos, patrones detectados y estadísticas detalladas de los análisis de phishing.
- Permite la generación de reportes en diferentes formatos con información como tipos de correos, porcentajes de riesgo y patrones detectados.
- Ofrece un panel de control para la administración de usuarios, la configuración de permisos y la personalización de los reportes.
- Permite la actualización de los resultados en tiempo real, mostrando nuevos análisis conforme son procesados por el sistema.

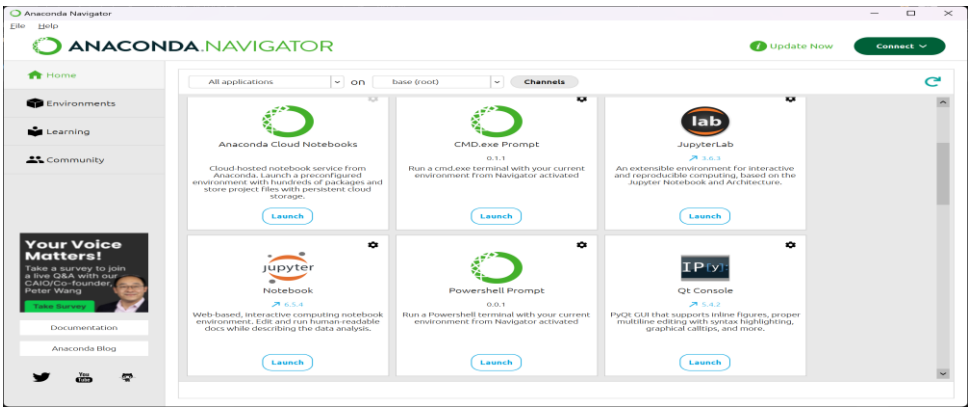
4.1.4. Detalles del proceso de desarrollo

4.1.4.1. Fase 1 - Análisis y definición de requisitos

En esta fase se llevó a cabo la instalación y configuración de las herramientas necesarias para el desarrollo del sistema de detección de phishing, se utilizó Anaconda como entorno de gestión de paquetes, instalación de Python, y la instalación de Jupyter para programar los módulos, lo que facilitó la organización de las dependencias del proyecto.

Figura 9

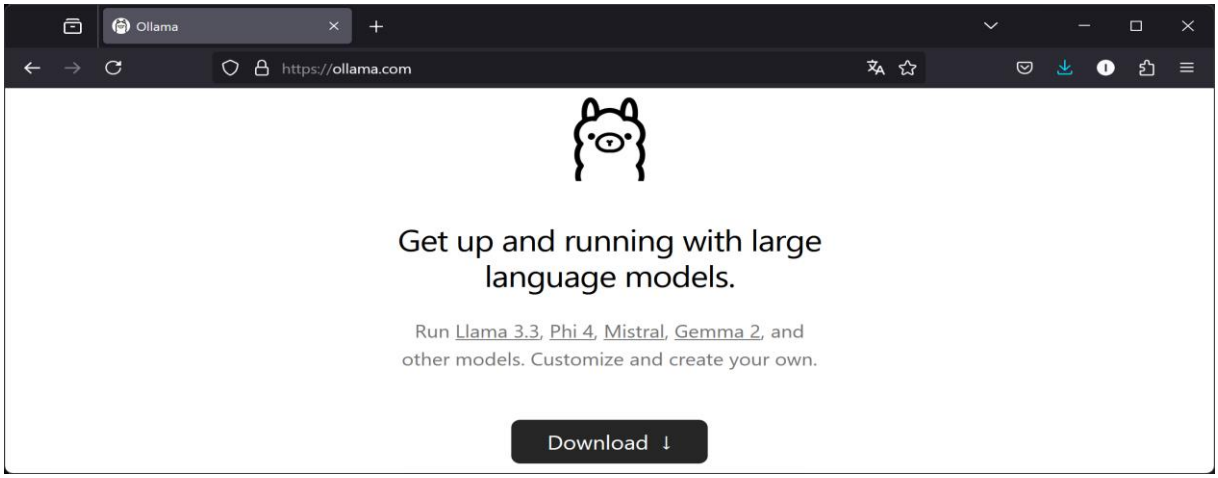
Entorno de Anaconda



Asimismo, se instaló Ollama, una plataforma clave para la implementación y gestión de modelos de lenguaje grande (LLMs), permitiendo la ejecución y ajuste de modelos de manera local.

Figura 10

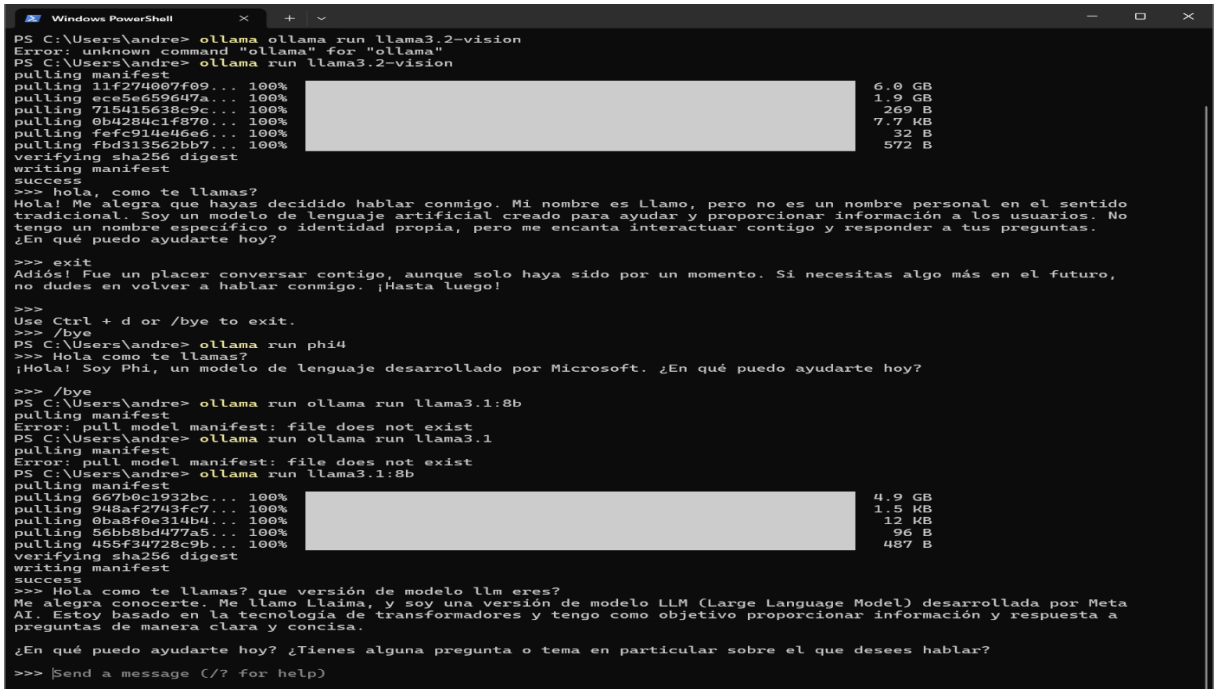
Página principal para la instalación de Ollama



Se realizaron las configuraciones pertinentes para la instalación de modelos LLM compatibles mediante Ollama, asegurando que los recursos estuvieran adecuadamente configurados para garantizar un rendimiento óptimo durante las pruebas y el desarrollo del sistema.

Figura 11

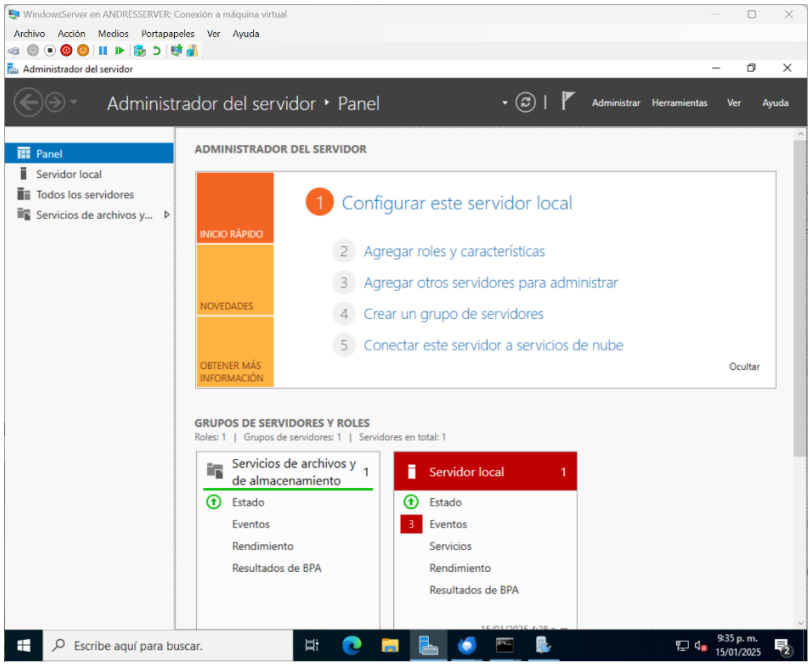
Instalación de los LLMs en Ollama



Para la creación del entorno local de pruebas, se instaló Windows Server 2022 como sistema operativo base en la máquina destinada a ser el servidor de correos.

Figura 12

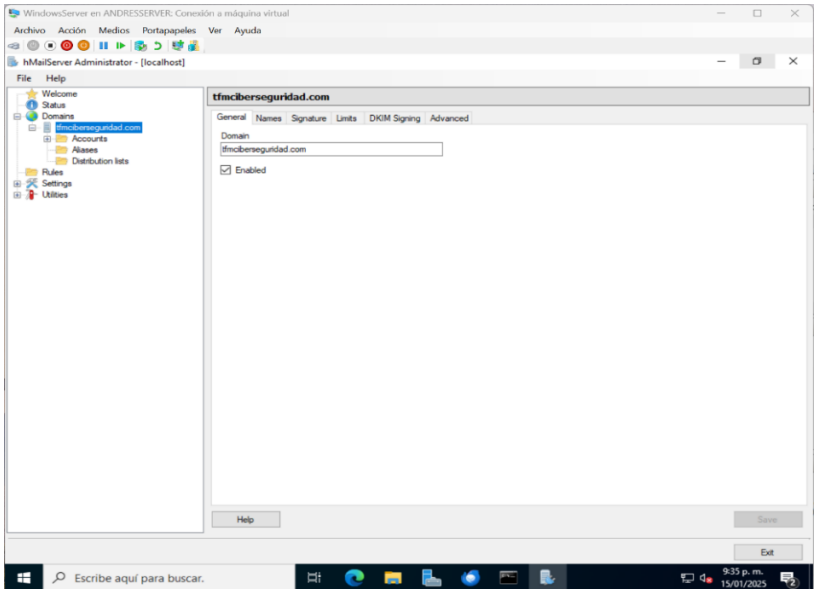
Servidor Windows Server 2022



En este entorno se configuró hMailServer, que actuó como servidor de correos para simular un flujo real de mensajes.

Figura 13

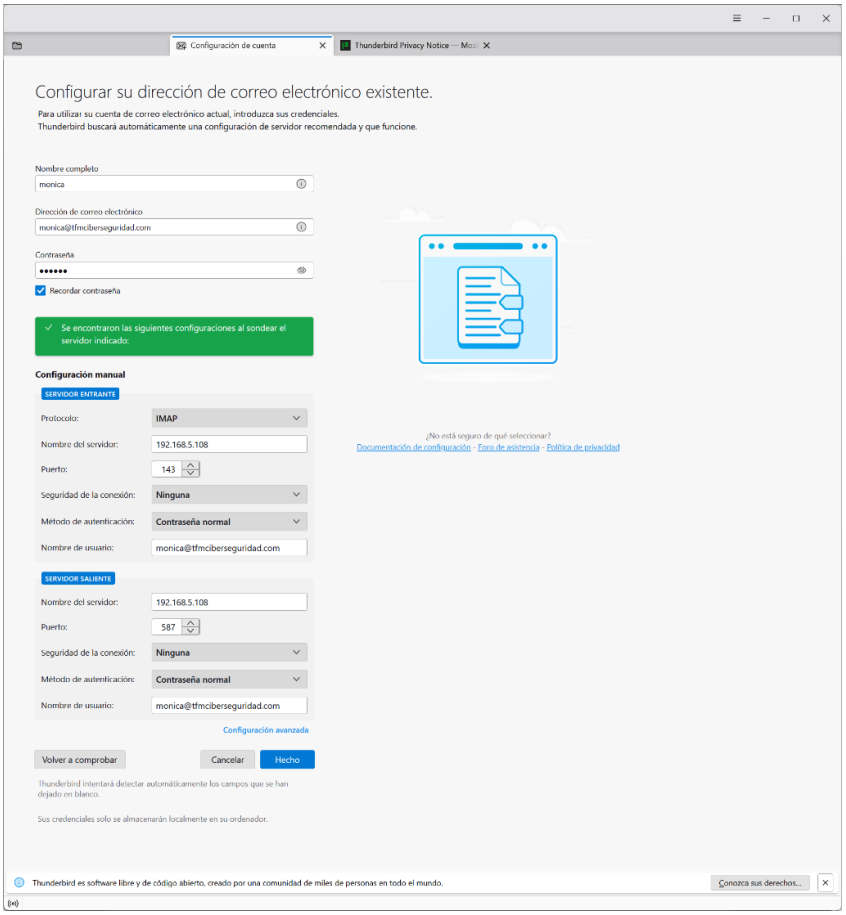
Configuración de hMailServer



Además, se instaló Thunderbird como cliente de correo, lo que permitió gestionar cuentas de prueba y verificar la recepción y envío de correos electrónicos de manera manual y automática, este entorno de prueba permitiría evaluar la funcionalidad completa del sistema y verificar su capacidad de integración con las herramientas de comunicación habituales.

Figura 14

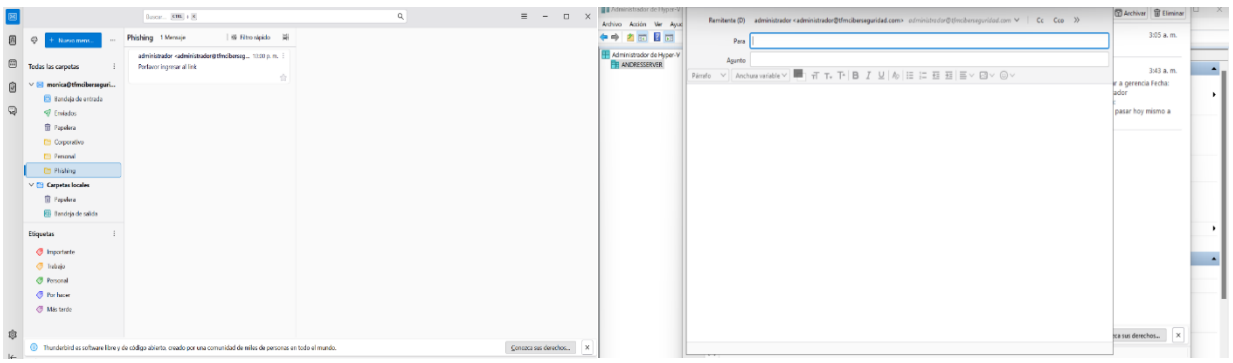
Conexión de Thunderbird con el servidor de correo hmailServer



Y se procede al envío de correos para verificar el funcionamiento de la aplicación Thunderbird

Figura 15

Prueba del entorno local - Servidor de correo



En esta fase se obtuvo lo siguiente:

- Realización de revisión de documentos expertos en ciberseguridad para identificar desafíos actuales.
- Definición de los requisitos funcionales y no funcionales del sistema.
- Revisión de literatura técnica y análisis de herramientas similares para identificar brechas y oportunidades.
- Selección de LLMs de código abierto como Llama 3.2 y Phi 4, considerando su rendimiento y capacidad de implementación on-premise.
- Adecuación del ambiente de prueba.

4.1.4.2. Fase 2 - Diseño del sistema

Durante esta fase, se organizó el hardware necesario para garantizar un entorno de implementación adecuado, se definió que la máquina utilizada debía contar con un mínimo de 32 GB de memoria RAM para manejar de manera eficiente los procesos de análisis y gestión de datos, así como una GPU con al menos 16 GB de memoria dedicada para soportar los modelos de lenguaje grande (LLMs) y optimizar los tiempos de inferencia y procesamiento. Esta planificación permitió crear una infraestructura sólida que respalde la operación fluida del sistema, maximizando su rendimiento y capacidad de respuesta ante grandes volúmenes de correos electrónicos.

Figura 16

Características de CPU y RAM

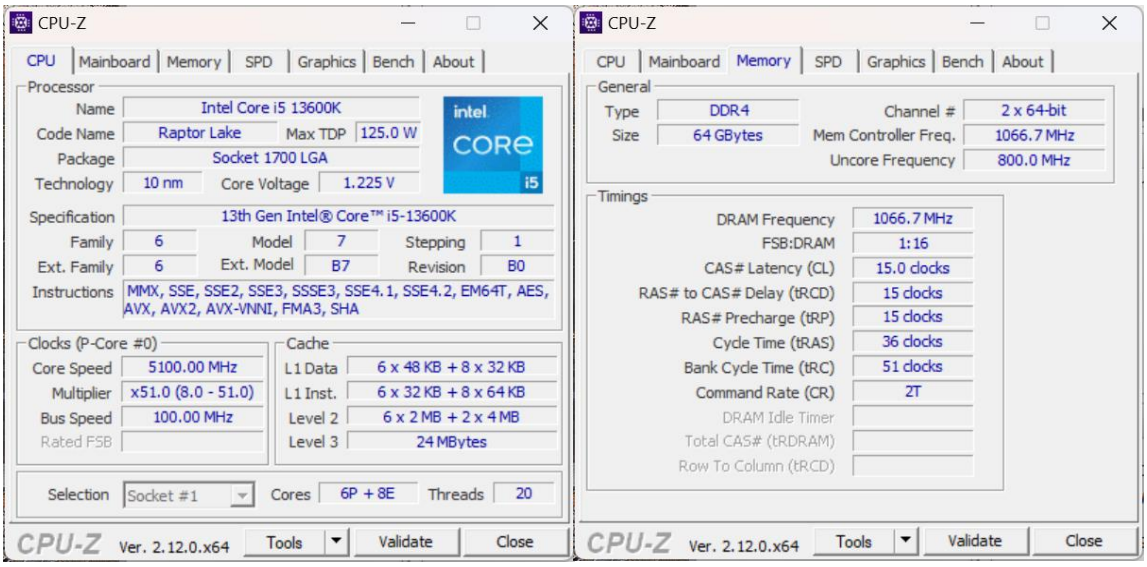
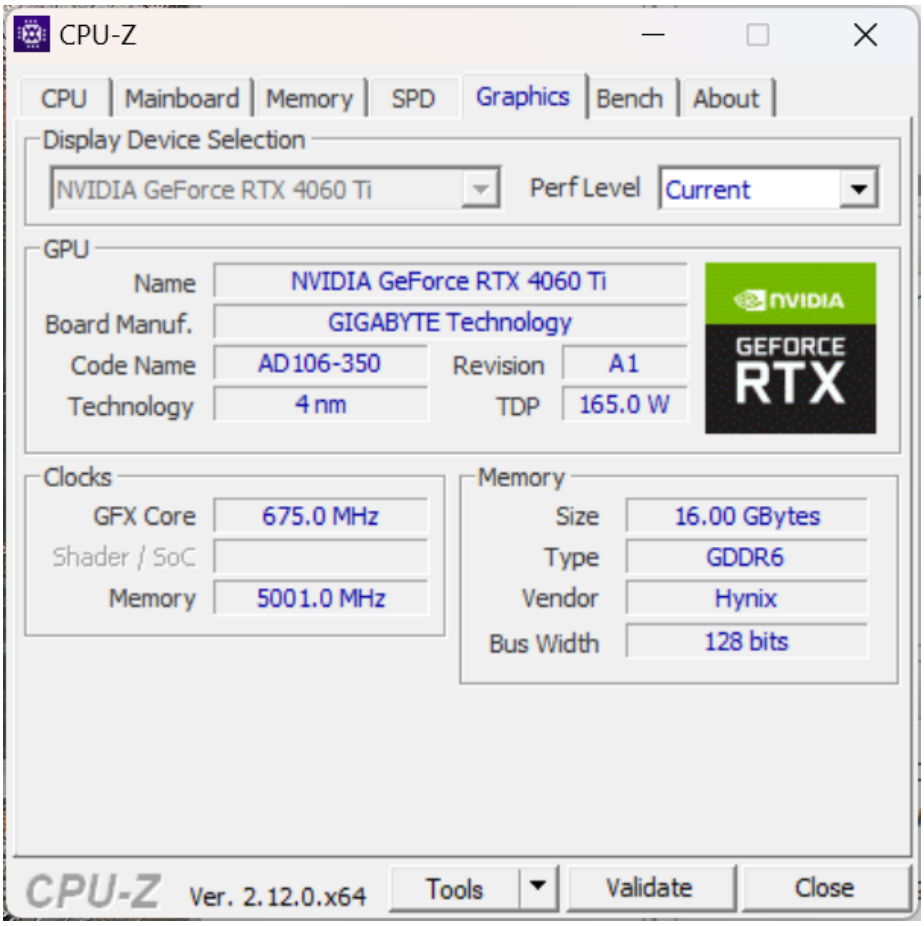


Figura 17

Características de la GPU



En esta fase se obtuvo lo siguiente:

- Diseño de la arquitectura modular del sistema, incorporando los módulos de conexión, análisis, clasificación, respuesta automática y panel de control.
- Definición de los flujos de trabajo internos, asegurando una comunicación eficiente entre los módulos.
- Planificación de las conexiones con los servicios de correo electrónico.
- Planificación de la infraestructura tecnológica necesaria, incluyendo servidores y hardware con GPUs estándar para procesamiento.

4.1.4.3. Fase 3 - Desarrollo e implementación

Durante esta fase, se implementó el flujo de trabajo completo mediante la integración de múltiples componentes, asegurando una detección eficaz de correos electrónicos maliciosos.

Implementación de la Base de Datos

Se diseñó y desplegó una base de datos relacional en MySQL, siguiendo un modelo entidad-relación que incluye entidades clave como correo, usuario_correo, analisis_resultado, acciones_automatizadas, y patron_phishing, la base de datos garantiza la integridad y trazabilidad de la información mediante el uso de claves primarias y foráneas que conectan los diferentes módulos del sistema.

Plataforma Laravel

Se desarrolló una plataforma web en Laravel que actúa como el núcleo administrativo del sistema, lo que permitió que esta plataforma proporcionara las API RESTful para la creación, modificación y consulta de correos, análisis de resultados y acciones automatizadas. Se utilizaron migraciones para estructurar la base de datos y Eloquent ORM para gestionar las relaciones entre las entidades.

API REST de Laravel y conexión con Python

Las API RESTful de Laravel facilitan la comunicación entre la plataforma y el cliente de automatización en Python, las API permiten:

- Registrar correos electrónicos recibidos.
- Guardar resultados de análisis realizados por los modelos LLM.
- Almacenar patrones de phishing detectados.
- Registrar acciones automatizadas ejecutadas sobre los correos.

Desarrollo de tareas en Python

El cliente en Python realiza las siguientes tareas.

Conexión al servidor de correos: Utiliza la biblioteca imaplib para acceder a la bandeja de entrada autenticándose con las credenciales del usuario, este proceso se encuentra en la función ejecutar_proceso(), donde se establece la conexión al servidor IMAP y se selecciona la bandeja de entrada.

```
mail = imaplib.IMAP4(SERVER, 143)
mail.login(EMAIL, PASSWORD)
mail.select('inbox')
```

Extracción de datos del correo: Procesa cada correo para extraer remitente, destinatario, asunto y contenido en texto plano, esto se realiza dentro del bucle que recorre los correos recuperados.

```
message = email.message_from_bytes(response_part[1])
remitente = message['from']
destinatario = message.get('to')
mail_subject = message['subject']
```

Análisis con LLMs: Envía el contenido del correo a modelos LLM mediante la función ollama.chat(), que clasifica el correo en Corporativo, Personal o Phishing.

```
response = ollama.chat(
    model=modelo.implemetacion_ollama,
    messages=[{"role": "user", "content": f"Clasifica el sig
)
```

Procesamiento de resultados: Utiliza expresiones regulares para extraer los porcentajes de clasificación y determina la categoría con mayor probabilidad.

```
pattern = r"(\w+)\s*=>\s*(\d+)%"
matches = re.findall(pattern, response["message"]["content"]
```

Acciones automatizadas: Dependiendo de la clasificación, el correo puede ser movido a carpetas específicas, como Spam para correos de phishing.

```
if id_mayor == 3:  # Si es phishing
    result_copy = mail.copy(i, 'Phishing')
    mail.store(i, '+FLAGS', '\\Deleted')
    mail.expunge()
```

Almacenamiento de resultados: Envía los resultados del análisis y las acciones ejecutadas de vuelta a la plataforma Laravel mediante las API REST.

```
crear_analisis_resultado(nuevo_analisis)
crear_accion_automatizada(nueva_accion)
```

Por tanto, si el proceso completo es el siguiente, lo primero es importar todos los recursos necesarios para la operación del sistema.

Figura 18

Código fuente de la aplicación – Parte 1

```
import tkinter as tk
from tkinter import ttk, scrolledtext
import threading
import time
import re
import email
import imaplib
from email.header import decode_header
from datetime import datetime

# Importar las funciones de API
from api_requests.modelo_llms import obtener_todos_los_modelos_llm
from api_requests.usuario_correos import obtener_todos_los_usuarios_correos
from api_requests.correos import crear_correo, modificar_correo
from api_requests.analisis_resultado import crear_analisis_resultado
from api_requests.patron_phishings import crear_patron_phishing, modificar_patron_phishing
from api_requests.acciones_automatizadas import crear_accion_automatizada, modificar_accion_automatizada
import ollama
```

Luego se desarrolla una interfaz gráfica para la implementación de manera sencilla e intuitiva, que se centra en pantalla si solo son asignados 4 elementos, dos botones y label y un text área, este último para la visualización de las salidas del script.

Figura 19

Código fuente de la aplicación – Parte 2

```
class App:
    def __init__(self, master):
        self.master = master
        ancho = 800
        alto = 600
        ancho_pantalla = master.winfo_screenwidth()
        alto_pantalla = master.winfo_screenheight()
        x = (ancho_pantalla // 2) - (ancho // 2)
        y = (alto_pantalla // 2) - (alto // 2)
        self.master.title("SodpIA - Análisis de Correos Electrónicos")
        self.master.geometry(f'{ancho}x{alto}+{x}+{y}')

        self.estado_label = tk.Label(
            master,
            text="Software On-Premise de Detección de Phishing utilizando Modelos LLM",
            font=("Arial", 14, "bold"),  # Cambia la fuente a Arial, aumenta el tamaño y agrega negrita
            fg="#ffffff",                # Texto en color blanco
            bg="#2c3e50",                # Fondo en azul oscuro (estilo moderno)
            padx=10,                     # Relleno horizontal
            pady=10                      # Relleno vertical
        )
        self.estado_label.pack(pady=15)  # Añadir espacio adicional alrededor del label

        # Botones para iniciar y detener
        self.boton_iniciar = tk.Button(master, text="Iniciar Proceso", command=self.iniciar_proceso, bg="green",
        self.boton_iniciar.pack(pady=15)

        self.boton_detener = tk.Button(master, text="Detener Proceso", command=self.detener_proceso, bg="red",
        self.boton_detener.pack(pady=15)

        # Área de texto para mostrar la salida
        self.area_texto = scrolledtext.ScrolledText(master, wrap=tk.WORD, width=80, height=20)
        self.area_texto.pack(pady=15)
```

El script establece la conexión con el servidor de correos utilizando la biblioteca imaplib y accede al buzón de entrada autenticándose con las credenciales del correo, una vez conectado, el sistema selecciona la bandeja de entrada y obtiene todos los mensajes disponibles mediante comandos de búsqueda.

Figura 20

Código fuente de la aplicación – Parte 3

```
def ejecutar_proceso(self):
    modelos = obtener_todos_los_modelos_llm()

    while self.proceso_activo:
        usuario_correos = obtener_todos_los_usuarios_correos()
        for usuario_correo in usuario_correos:
            if not self.proceso_activo:
                break

            self.actualizar_texto(f"Procesando correo de: {usuario_correo.correo}")
            EMAIL = usuario_correo.correo
            PASSWORD = usuario_correo.contrasena_hash
            SERVER = '192.168.5.108'
            mail = imaplib.IMAP4(SERVER, 143)
            mail.login(EMAIL, PASSWORD)
            mail.select('inbox')
            status, data = mail.search(None, 'ALL')
            mail_ids = []
            for block in data:
                mail_ids += block.split()
            for i in mail_ids:
                status, data = mail.fetch(i, '(RFC822)')
                for response_part in data:
                    if isinstance(response_part, tuple):
                        message = email.message_from_bytes(response_part[1])
                        remitente = message['from']
                        destinatario = message.get('to')
                        mail_subject = message['subject']
                        message_id = message.get('Message-ID')
```

Posteriormente, se procesa cada correo para extraer el remitente, el asunto y el contenido, si el correo contiene múltiples partes (como texto y archivos adjuntos), el script extrae únicamente el contenido de texto plano, la información recopilada se envía al modelo LLM a través de Ollama, utilizando la función ollama.chat(), la cual solicita al modelo que clasifique el correo en tres categorías: Corporativo, Personal o Phishing.

Figura 21

Código fuente de la aplicación – Parte 4

```
if modelo.estado_modelo_id == 1:
    inicio = time.time()
    response = ollama.chat(
        model=modelo.implemetacion_ollama,
        messages=[
            {
                "role": "user",
                "content": f"Requiero que clasifiques el siguiente correo
            },
        ],
    )
```

El código determina la clasificación final de un correo analizado utilizando los promedios de probabilidad obtenidos para las categorías Corporativo, Personal y Phishing, seleccionando la categoría con el mayor porcentaje mediante la función max, asignando el nombre correspondiente a la variable nombre_mayor; posteriormente, el correo se copia a la carpeta que coincide con dicha categoría usando mail.copy, y si la operación es exitosa, se marca el correo original para eliminación con mail.store y se ejecuta mail.expunge para eliminarlo definitivamente, registrando mensajes en la consola que confirman si las operaciones de copia y eliminación se realizaron correctamente o si hubo errores, finalmente, se registra la acción automatizada correspondiente, creando un diccionario nueva_accion que contiene el ID del correo, el tipo de acción basada en la clasificación, el estado de la acción como ejecutada, una descripción indicando que la acción se realizó correctamente y la fecha exacta de ejecución, y esta información se envía al sistema para su almacenamiento y monitoreo.

Figura 22

Código fuente de la aplicación – Parte 5

```
variables = {1: promedio_corporativo, 2: promedio_personal, 3: promedio_phishing}
id_mayor = max(variables, key=variables.get)
if id_mayor == 1:
    nombre_mayor = 'Corporativo'
else:
    if id_mayor == 2:
        nombre_mayor = 'Personal'
    else:
        nombre_mayor = 'Phishing'

result_copy = mail.copy(i, nombre_mayor)
if result_copy[0] == 'OK':
    print(f'Correo copiado a "{nombre_mayor}" correctamente.')
    result_delete = mail.store(i, '+FLAGS', '\\Deleted')
    mail.expunge()
    if result_delete[0] == 'OK':
        print(f'Correo movido a "{nombre_mayor}" correctamente.')
    else:
        print('Error al eliminar el correo de la carpeta original.')
else:
    print(f'Error al copiar el correo a "{nombre_mayor}".')

hora_actual = datetime.now().strftime('%Y-%m-%d %H:%M:%S')
nueva_accion = {
    "correo_id": correo_creado.id,
    "tipo_accion_id": id_mayor,
    "estado_accion_id": 2,
    "descripcion": "Acción realizada correctamente",
    "fecha_accion": hora_actual
}
```

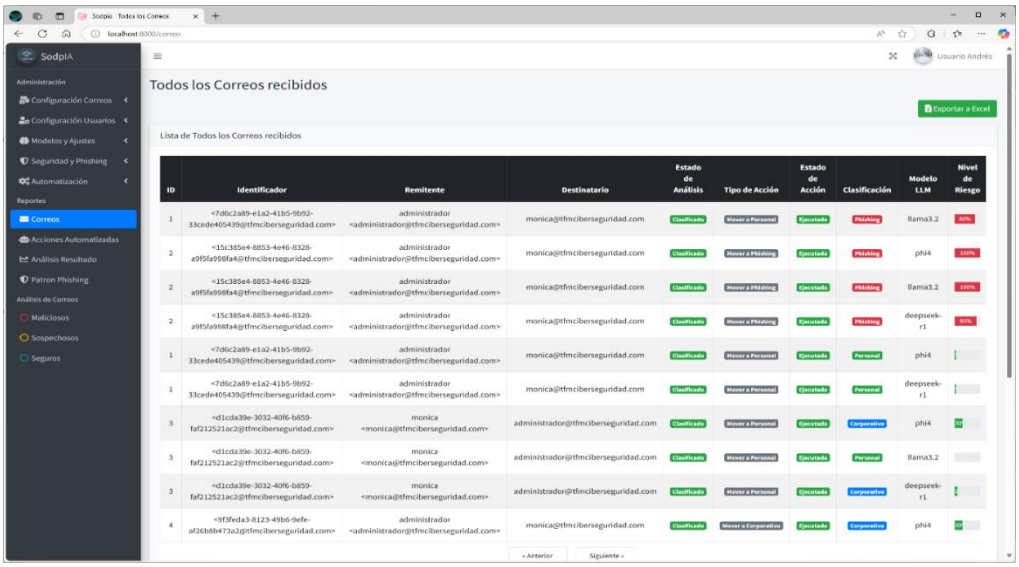
Visualización de reportes en la plataforma Laravel

La plataforma web permite la visualización y gestión de los correos procesados, resultados de análisis y acciones automatizadas. Los administradores pueden:

Revisar los correos clasificados como corporativos, personales o phishing.

Figura 23

Reporte de estados de los correos



Consultar los patrones de phishing detectados y las acciones ejecutadas.

Figura 24

Reporte de los patrones de phishing

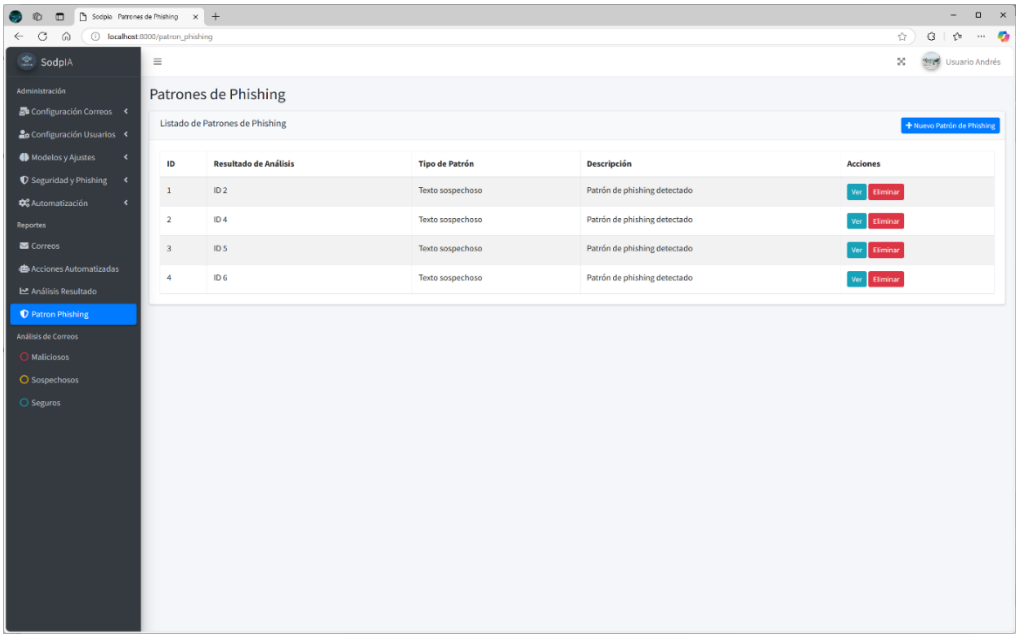
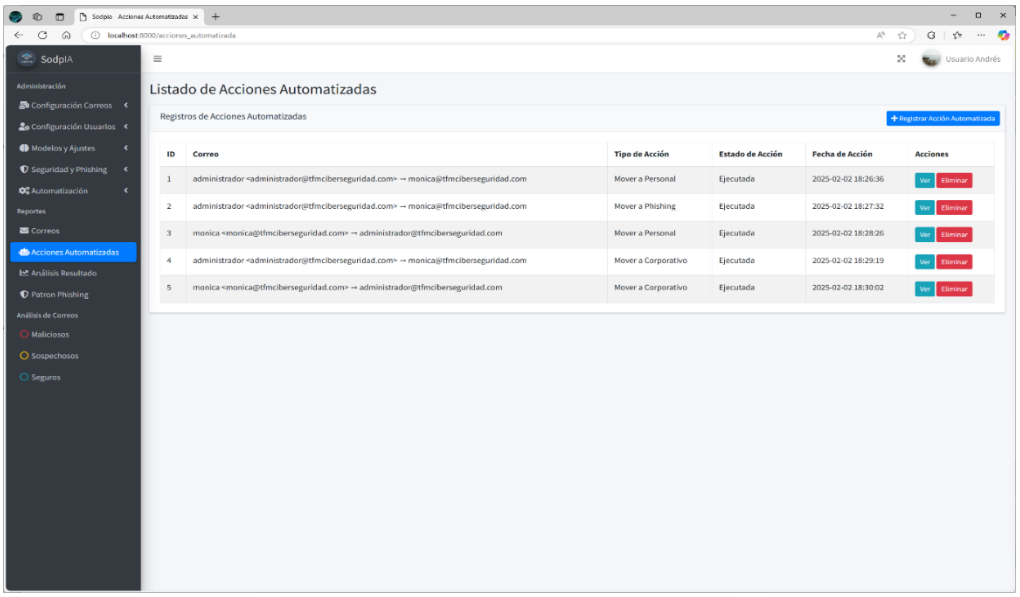


Figura 25

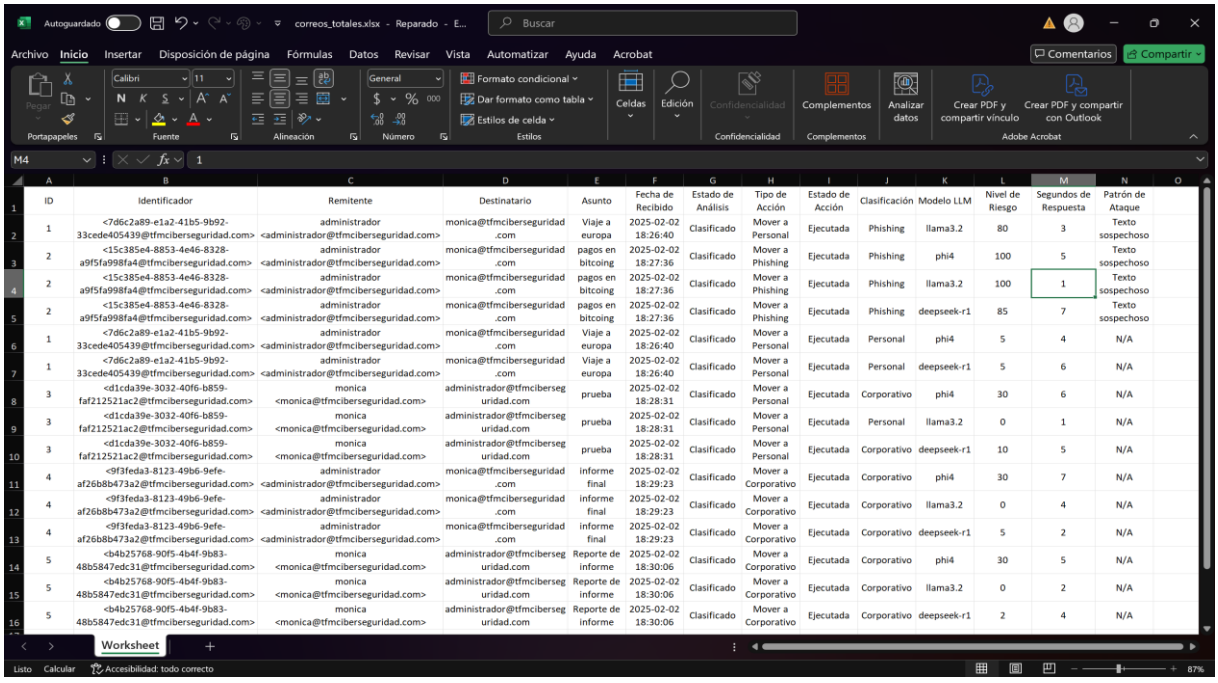
Reporte de acciones ejecutadas



Generar reportes y exportarlos en formatos como Excel para su análisis.

Figura 26

Reporte generado en Excel



El entorno de implementación incluyó el uso de herramientas adicionales como hMailServer para la gestión de correos en el servidor y Thunderbird como cliente para pruebas de validación, este flujo completo demostró la funcionalidad de la herramienta desde la recepción hasta la ejecución automatizada de acciones sobre los correos electrónicos.

En esta fase se obtuvo lo siguiente:

- Implementación del módulo de conexión al servidor de correo, garantizando compatibilidad con protocolos IMAP y SMTP.
- Entrenamiento y ajuste de los modelos LLMs seleccionados mediante técnicas de Fine-Tuning y RAG, utilizando conjuntos de datos de phishing previamente etiquetados.
- Desarrollo del módulo de clasificación de correos, configurando los umbrales de decisión para identificar correos como seguros (Corporativos), sospechosos (Personales) o maliciosos (Phishing).
- Almacenamiento de resultados en la base de datos para generar reportes.
- Creación del panel de administración en Laravel, incorporando visualización de datos, generación de reportes y opciones de configuración.
- Integración de todos los módulos en un sistema cohesivo y funcional, facilitando la interacción entre la aplicación Python y la plataforma Laravel.

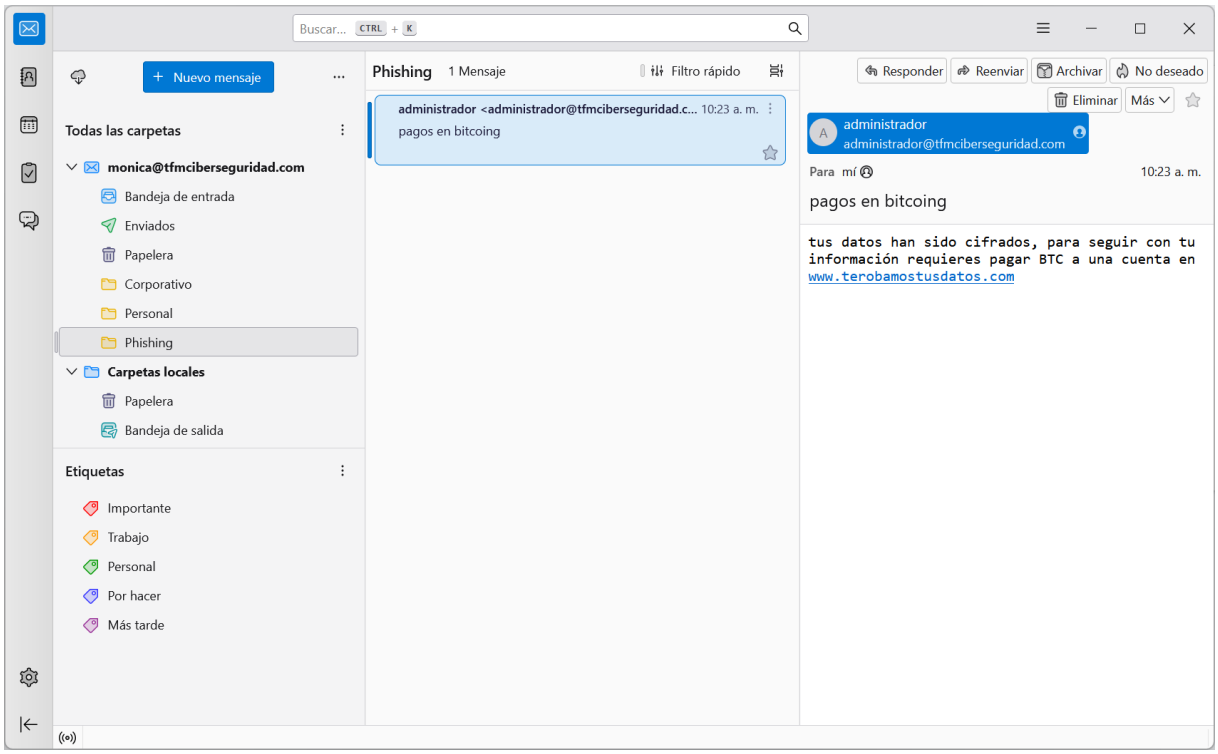
4.1.4.4. Fase 4 - Pruebas y validación

Durante esta fase, se realizaron diversas pruebas de validación para evaluar el correcto funcionamiento del sistema de detección de phishing, el sistema se sometió a un conjunto de correos electrónicos de prueba con diferentes características y contextos, tales como correos legítimos, personales y maliciosos, los resultados obtenidos se registraron y analizaron para verificar la precisión y eficiencia del modelo.

Por ejemplo, uno de los casos de prueba incluyó un correo con el asunto "pagos en bitcoing" y un contenido sospechoso relacionado con transacciones en criptomonedas. Los resultados del modelo indicaron una clasificación del 100% de probabilidad de ser Phishing, 0% de ser Corporativo y 0% de ser Personal. Basándose en esta clasificación, el sistema ejecutó las acciones correspondientes, copiando el correo a la carpeta de Phishing y eliminándolo de la bandeja de entrada. Los mensajes de salida confirmaron que el correo fue copiado y movido correctamente a la carpeta designada, lo que evidencia la correcta ejecución de los procesos definidos.

Figura 27

Proceso de análisis del correo



Durante la fase de pruebas y validación del sistema de detección de phishing, se realizó un análisis exhaustivo de los datos almacenados en la base de datos para evaluar la precisión, eficiencia y efectividad del sistema.

Prueba 1

La primera prueba que se desarrolló fue la prueba de funcionamiento conectado a los servicios de correo electrónicos, de esta manera revisamos la capacidad para conectarse y recolectar la información requerida de los correos, además de la capacidad para tomar decisiones y aplicar cambios en los servicios de correo electrónico.

Durante la fase de pruebas, se evaluó la precisión del sistema de clasificación de correos utilizando modelos de lenguaje (phi4, llama 3.2 y deepseek-r1), comparando sus resultados con las clasificaciones reales de los correos.

En esta prueba se analizaron un total de 5 correos electrónicos clasificados en las tres categorías, las cuales son Corporativo, Personal y Phishing, los resultados muestran la siguiente distribución.

Resultados generales

Total del análisis realizados: 17

Clasificaciones correctas: 12

Clasificaciones incorrectas: 3

Precisión general de los modelos: 80%

Precisión general de la acción: 100%

Detalles de clasificación

Correos con el asunto "pagos en bitcoing" fueron correctamente identificados como Phishing por los tres modelos, con niveles de riesgo del 100% (phi4 y llama 3.2) y 85% (deepseek-r1).

Correos con el asunto "informe final" y "Reporte de informe" fueron clasificados correctamente como “Corporativos” por todos los modelos, con niveles de riesgo bajos, entre 0% y 30%.

Errores de clasificación

El correo con el asunto "Viaje a europa", cuyo contenido real era Personal, fue clasificado incorrectamente como Phishing por llama 3.2 con un nivel de riesgo del 80%, sin embargo, phi4 y deepseek-r1 lo clasificaron correctamente como Personal.

El correo con el asunto "prueba", que debería haber sido clasificado como Personal, fue clasificado como Corporativo por phi4 y deepseek-r1, indicando una confusión en la naturaleza del contenido.

Análisis de Modelos

Los 3 modelos presentaron un 80% de precisión, pero en conjunto pudieron tener un 100% de presión en las acciones, esta información refleja la capacidad del sistema para identificar diferentes tipos de correos, destacando su efectividad en la detección de amenazas.

Análisis de tiempos de respuesta de los modelos LLM

El análisis de los tiempos de respuesta de los modelos LLM utilizados en el sistema reveló diferencias significativas en términos de velocidad y consistencia. A continuación, se detallan los resultados obtenidos para cada modelo:

El modelo llama 3.2 demostró ser el más rápido, con un tiempo de respuesta promedio de 2.2 segundos, su respuesta más rápida fue de 1 segundo, mientras que la más lenta alcanzó los 4 segundos.

El modelo phi4 presentó un tiempo de respuesta promedio de 5.6 segundos, siendo el más lento de los tres, y los tiempos oscilaron entre 5 y 7 segundos, con una desviación estándar de 0.89 segundos, lo que muestra una mayor consistencia en comparación con otros modelos, aunque con una velocidad de procesamiento menor.

Deepseek-r1 ofreció un rendimiento intermedio, con un tiempo de respuesta promedio de 4.8 segundos. El tiempo mínimo registrado fue de 2 segundos y el máximo de 7 segundos, sin embargo, su desviación estándar de 1.79 segundos indica una mayor variabilidad en los tiempos de respuesta, lo que podría afectar la predictibilidad del procesamiento en ciertos escenarios.

El modelo llama 3.2 destaca por su rapidez, siendo ideal para aplicaciones donde la velocidad es un factor crítico, en contraste, phi4 ofrece tiempos de respuesta más estables, lo que podría ser ventajoso en entornos donde la uniformidad en el desempeño es esencial.

Prueba 2

En la segunda fase de pruebas, se implementaron modificaciones significativas en el código del sistema para cambiar la fuente de los correos electrónicos, a diferencia de la primera prueba, donde la conexión se realizaba directamente con un servidor de correos electrónicos mediante protocolos IMAP y SMTP, en esta ocasión el sistema fue ajustado para interactuar con una base de datos estructurada de correos electrónicos.

La base de datos seleccionada para esta prueba fue el Corpus of 200 Multilingual Emails, una fuente pública ampliamente utilizada en investigaciones sobre detección de spam y análisis de correos electrónicos, la base de datos corpus contiene 200 correos electrónicos en tres

idiomas (español, inglés y portugués), distribuidos equitativamente en 100 correos legítimos (ham) y 100 correos spam, esta diversidad lingüística permitió evaluar no solo la capacidad del sistema para clasificar correos electrónicos, sino también su rendimiento frente a textos en diferentes idiomas.

Figura 28

Ajuste de Código fuente para la lectura de email de la base de datos Corpus

```
while self.proceso_activo:
    folder_path = 'correos_de_prueba'  # Asegúrate de que esta carpeta contenga los archivos 01.em

    # Iterar sobre los archivos de 01.eml hasta 200.eml
    for i in range(1, 201):
        file_name = f"{str(i).zfill(2)}.eml"
        file_path = os.path.join(folder_path, file_name)

        try:
            # Abrir el archivo en modo binario
            with open(file_path, 'rb') as email_file:
                # Parsear el contenido del archivo como un mensaje de email
                email_message = BytesParser(policy=policy.default).parse(email_file)

            # Extraer la información deseada
            remitente = email_message['from']
            destinatario = email_message.get('to')
            mail_subject = email_message['subject']
            message_id = email_message.get('Message-ID')
            fecha_raw = email_message.get('Date')
            if fecha_raw:
                fecha_datetime = parsedate_to_datetime(fecha_raw)
            else:
                fecha_datetime = datetime.now()  # Usar fecha actual si no hay fecha en el correo

            fecha_recepcion = fecha_datetime.strftime('%Y-%m-%d %H:%M:%S')

            # Mostrar la información extraída
            print(f"Archivo: {file_name}")
            print(f"Remitente: {remitente}")
            print(f"Destinatario: {destinatario}")
            print(f"Asunto: {mail_subject}")
            print(f"Message-ID: {message_id}")
            print(f"Fecha: {fecha_recepcion}")
```

La elección de este corpus fue estratégica debido a que cada correo electrónico en la base de datos está previamente clasificado y etiquetado como ham o spam, esta característica resultó fundamental para la fase de validación, ya que permitió comparar las predicciones del sistema con las clasificaciones reales y así medir la precisión y efectividad de los modelos LLM en la detección de correos maliciosos.

Además, el uso de una base de datos estructurada facilitó la automatización del proceso de análisis y permitió realizar pruebas de manera más eficiente y controlada. Esto proporcionó un entorno ideal para identificar posibles ajustes en los umbrales de clasificación y optimizar los modelos antes de su implementación en un entorno de producción con correos en tiempo real.

Los resultados fueron los siguientes:

Precisión general del sistema

- Correos acertados: 165 de 182 (90.66%)
- Correos no acertados: 17 de 182 (9.34%)
- Correos no procesados: 18 de 200 (9%)

El sistema alcanzó una precisión del 90.66%, lo cual indica un rendimiento robusto en la clasificación correcta de los correos electrónicos, sin embargo, un 9% de los correos no fueron procesados, lo que podría deberse a errores técnicos o problemas de como en los correos los modelos de LLM.

Tabla 9

Rendimiento por modelo LLM

Modelo	Correos Procesados	% Procesamiento	Correos Acertados	% Aciertos	Correos No Acertados	% Error
deepseek-r1	171	85.5%	165	96.49%	6	3.51%
llama 3.2	174	87%	165	94.83%	9	5.45%
phi4	196	98%	164	83.67%	32	16.33%
Total	541	90.17%	494	91.31%	47	8.69%

De lo cual podemos ver que:

- deepseek-r1 tuvo la mayor precisión (96.49%), aunque procesó menos correos que phi4.

- llama 3.2 mostró un buen equilibrio con una alta precisión (94.83%) y tiempos de respuesta rápidos.
- phi4, aunque procesó más correos (98%), mostró una precisión menor (83.67%) y un porcentaje de error considerable (16.33%).

Tabla 10

Tiempos de respuesta promedio de los modelos LLMs

Modelo	Promedio de segundos de respuesta
deepseek-r1	7.21 s
llama 3.2	1.99 s
phi4	5.60 s
Total	4.95 s

De lo cual podemos indicar que:

- llama 3.2 es el modelo más eficiente en términos de velocidad, con un promedio de 1.99 segundos por correo.
- deepseek-r1 tiene el tiempo de respuesta más alto (7.21 segundos), aunque compensa con una alta precisión.
- phi4 se posiciona en un punto intermedio con 5.6 segundos, pero su mayor tasa de errores podría afectar la confiabilidad en contextos críticos.

Desempeño por tipo de clasificación

100% de precisión en todos los modelos para clasificaciones de Corporativo y Personal, esto muestra que el sistema es altamente confiable para distinguir correos legítimos, lo cual es

fundamental para evitar falsos positivos que puedan afectar la operatividad de una organización.

Tabla 11

Desempeño por tipo de clasificación

Modelo	Clasificación	Correos Procesados	Acertó	% Aciertos	No Acertó	% Errores
deepseek-r1	Corporativo	82	82	100.00%	0	0%
	Personal	18	18	100.00%	0	0%
	Phishing	71	65	91.55%	6	8.45%
llama 3.2	Corporativo	82	82	100.00%	0	0%
	Personal	18	18	100.00%	0	0%
	Phishing	74	65	87.84%	9	12.16%
phi4	Corporativo	81	81	100.00%	0	0%
	Personal	18	18	100.00%	0	0%
	Phishing	97	65	67.01%	32	32.99%
Total		541	494	91.31%	47	8.69%

deepseek-r1 logró una precisión del 91.55% en la detección de correos de Phishing, siendo el más preciso en esta categoría, y llama 3.2 también mostró un buen desempeño con un 87.84% de aciertos en Phishing, contrario a los otros phi4 tuvo un desempeño significativamente inferior en esta categoría, con solo un 67.01% de precisión, lo que indica que un 32.99% de los correos de Phishing no fueron detectados correctamente.

Aunque phi4 procesa más correos, su capacidad de identificar Phishing es la más baja, lo que podría representar un riesgo en entornos donde la detección de amenazas es crítica.

deepseek-r1 y llama 3.2 son más confiables para la detección de Phishing, pero el primero tiene la ventaja en términos de precisión.

Resultados generales

El sistema de clasificación de correos electrónicos demostró un alto desempeño general, alcanzando una precisión global del 90.66%, lo que lo hace adecuado para aplicaciones prácticas, en la comparación entre modelos, deepseek-r1 destacó por su precisión en la detección de Phishing (91.55%), aunque con tiempos de respuesta más largos, mientras que llama 3.2 logró equilibrar velocidad y precisión, siendo la opción ideal para entornos donde la rapidez es crítica, por otro lado, phi4 mostró la mayor capacidad de procesamiento, pero con una precisión inferior en la detección de Phishing (67.01%), lo que limita su uso en aplicaciones donde los errores son inaceptables, además, el 9% de correos no procesados sugiere la necesidad de revisar la infraestructura del sistema o su compatibilidad con ciertos formatos de correo.

Entre las fortalezas del sistema se destaca la clasificación precisa de correos Corporativos y Personales en todos los modelos, asegurando la correcta gestión de correos no maliciosos, sin embargo, existen áreas claras de mejora, como la optimización de phi4 para mejorar su precisión en Phishing, y la reducción de falsos negativos que podrían comprometer la seguridad.

Estos resultados confirman que el sistema es robusto para clasificar correos legítimos, pero aún presenta oportunidades de mejora en la detección de amenazas.

El análisis de la base de datos evidencia que el sistema de detección de phishing funciona de manera precisa y eficiente, logrando clasificar los correos correctamente y ejecutar las acciones automatizadas sin errores, además, la integración de diferentes modelos LLM permite un análisis más robusto y una mejor adaptación a distintos contextos de correos electrónicos, lo que indica que el rendimiento del sistema es óptimo tanto en la velocidad de análisis como en la ejecución de acciones, validando su aplicabilidad en entornos productivos.

En términos generales esta fase se obtuvo lo siguiente:

- Realización de pruebas unitarias para verificar el funcionamiento individual de cada módulo.
- Pruebas de integración para asegurar la comunicación fluida entre módulos.

- Validación del rendimiento del sistema en entornos simulados y reales, evaluando la precisión, tasas de falsos positivos y capacidad de manejo de volúmenes altos de correos.
- Ajustes finales basados en los resultados de las pruebas, optimizando el rendimiento y la usabilidad.

4.1.4.5. Fase 5 - Documentación y capacitación

- Generación de documentación técnica detallada que cubre instalación, configuración y uso del sistema.
- Preparación de guías para la capacitación de usuarios finales y administradores del sistema.

4.1.4.6. Características de la herramienta

La herramienta consta de los siguientes módulos:

- Conexión segura al servidor de correo: Proporciona una integración segura y eficiente con plataformas de correo existentes.
- Procesamiento y análisis con LLMs: Utiliza modelos LLMs ajustados para analizar el contenido textual de los correos electrónicos, identificando patrones sospechosos.
- Clasificación de correos: Determina si los correos son seguros (Corporativos), sospechosos (Personales) o maliciosos (Phishing).
- Generación de respuestas automáticas: Implementa acciones basadas en el análisis, como bloqueo de correos o notificaciones.
- Panel de control y reportes: Permite monitorear, configurar y visualizar estadísticas del sistema.

4.1.5. Evaluación

La evaluación del sistema on-premise de detección de phishing en tiempo real, basado en modelos de lenguaje grande (LLMs) de código abierto, se llevó a cabo en un entorno controlado utilizando un servidor de correos local, y durante esta fase, el sistema fue sometido

al procesamiento de correos electrónicos simulados, tanto legítimos como maliciosos, con el objetivo de verificar su funcionalidad, precisión y capacidad de respuesta, este enfoque permitió validar el cumplimiento de los objetivos específicos planteados, proporcionando una visión integral de la eficacia del sistema en condiciones controladas.

4.1.5.1. Evaluación de usabilidad

Se implementó un entorno de prueba con un servidor de correos local para evaluar la interacción del sistema y verificar la facilidad de uso en diferentes etapas del flujo de trabajo.

- Interfaz de usuario: Se evaluó la claridad en la presentación de los datos y la estructura del panel de control, además se verificó la capacidad de navegación, la accesibilidad de los reportes y la gestión de parámetros, demostrando que la interfaz es intuitiva, facilitando la comprensión de los resultados obtenidos por el modelo LLM.
- Tiempo de configuración: El proceso de instalación y configuración del sistema, incluida la conexión al servidor local de correos, se completó en un tiempo promedio acorde a lo indicado, demostrando un flujo eficiente acorde con los objetivos del proyecto.
- Facilidad de integración: Se validó la integración fluida del sistema con protocolos estándar de correo (IMAP y SMTP), confirmando que no se requirieron ajustes significativos en la infraestructura del servidor de pruebas.

4.1.5.2. Evaluación de aplicabilidad

La aplicabilidad del sistema fue evaluada mediante pruebas con 200 correos simulados, de los cuales el 50% correspondían a correos maliciosos y el otro 50% a correos legítimos, en tres idiomas (español, inglés y portugués).

- Precisión en la detección: El sistema alcanzó una precisión del 90.66% en la clasificación de correos electrónicos, superando el objetivo inicial del 80%, la clasificación de correos corporativos y personales fue impecable, logrando una precisión del 100% en todos los modelos utilizados (deepseek-r1, llama 3.2 y phi4).

- Reducción de falsos positivos y negativos: El análisis mostró un 9.34% de errores (falsos positivos y falsos negativos), siendo phi4 el modelo con mayor cantidad de errores en la detección de phishing (32.99%), mientras que deepseek-r1 y llama 3.2 destacaron en la detección de phishing con precisiones del 91.55% y 87.84%, respectivamente, aún que llama 3.2 es un modelo mucho más pequeño que deepseek-r1.
- Escalabilidad: El sistema procesó el 91% de los correos electrónicos, con un 9% de correos no procesados, lo que sugiere áreas de mejora en la compatibilidad o en la infraestructura del sistema para alcanzar un procesamiento del 100%, aún que esta dentro del margen planteado inicialmente.

4.1.5.3. Evaluación de aplicabilidad

Se evaluó la configuración del módulo de respuestas automáticas.

Se evaluó el módulo encargado de las respuestas automáticas para validar la eficacia y rapidez en la toma de decisiones ante la detección de amenazas.

El tiempo promedio de respuesta del sistema fue de 4.95 segundos, cumpliendo con el objetivo de mantener el tiempo por debajo de 5 segundos, entre los modelos, llama 3.2 destacó con el tiempo más rápido (1.99 segundos), mientras que deepseek-r1 tuvo un tiempo de respuesta más largo (7.21 segundos), aunque con mayor precisión.

El sistema generó alertas automáticas para los correos clasificados como phishing, reubicando los correos maliciosos en carpetas específicas y emitiendo reportes de bloqueo de manera eficiente, cumpliendo con los requisitos de seguridad esperados.

4.1.5.4. Conclusión de la evaluación

Los resultados de la evaluación demuestran que el sistema cumple de manera satisfactoria con los objetivos específicos propuestos:

- Selección y ajuste del modelo LLM: Se seleccionaron modelos adecuados basados en métricas de precisión y rendimiento, en donde deepseek-r1 se destacó como el modelo más preciso para la detección de phishing, mientras que llama 3.2 ofreció un excelente balance entre precisión y velocidad.

- Procesamiento de correos en tiempo real: El sistema logró procesar el 91% de los correos electrónicos, con un 90.66% de precisión general en las clasificaciones, y la alta tasa de procesamiento y la baja latencia en la respuesta confirman su capacidad para operar en tiempo real.
- Generación de Respuestas Automáticas: El sistema fue capaz de generar respuestas automáticas eficientes en un promedio de 4.95 segundos, cumpliendo con los objetivos de tiempo de respuesta y precisión en la acción automática sobre correos maliciosos.

Estos resultados posicionan al sistema como una solución efectiva para la detección de phishing en entornos locales, destacando su precisión, velocidad y capacidad de integración, no obstante, se identificaron áreas de mejora, como la optimización del modelo phi4 para reducir su tasa de error y la investigación de las causas de los correos no procesados, se debe plantear la posibilidad de la adopción de un enfoque híbrido, que combine la precisión de deepseek-r1 con la velocidad de llama 3.2, podría potenciar aún más el rendimiento del sistema, ofreciendo una solución robusta y adaptable frente a las amenazas de phishing en tiempo real.

4.2. Análisis de resultados obtenidos

Los resultados obtenidos en el desarrollo del sistema on-premise de detección de phishing se presentan a continuación, detallando los logros y análisis correspondientes a cada objetivo específico planteado.

Los resultados obtenidos demuestran el cumplimiento de todos los objetivos específicos del proyecto dentro de los plazos establecidos, con indicadores de precisión y tiempos de respuesta que superaron las metas planteadas, además, la selección y ajuste del modelo LLM Llama 3.2, deepseek-r1 y Phi 4 permitió alcanzar un rendimiento óptimo en la clasificación de correos, y la implementación de respuestas automáticas garantiza una protección proactiva contra amenazas de phishing en tiempo real, ahora cada uno de esto modelos tiene sus ventajas y desventajas en términos de calidad y tiempos.

4.2.1. Objetivo 1: Análisis de tecnologías actuales de detección de phishing

Se llevó a cabo un análisis exhaustivo de las tecnologías actuales de detección de phishing, con el objetivo de identificar al menos tres soluciones on-premise basadas en modelos de lenguaje grande (LLMs) en un plazo máximo de un mes. El análisis comparativo incluyó:

- Llama 3.2: Modelo de código abierto altamente eficiente en tareas de procesamiento de lenguaje natural (NLP), destacando por su precisión y bajo consumo de recursos en entornos locales.
- Phi 4: Optimizado para el razonamiento estructurado y la detección de patrones complejos en correos electrónicos, aunque mostró limitaciones en la precisión de detección de phishing.
- Deepseek-r1: Un modelo ligero y eficiente, capaz de manejar grandes volúmenes de datos, con un rendimiento sobresaliente en la clasificación de correos maliciosos.

El análisis incluyó la documentación de características, requerimientos de hardware y métricas de rendimiento, este proceso permitió seleccionar el modelo más adecuado con base en criterios como precisión, adaptabilidad y eficiencia de recursos.

Resultado: El informe comparativo se completó en 4 semanas, cumpliendo con el plazo establecido y proporcionando una base sólida para la selección del modelo final.

4.2.2. Objetivo 2: Selección y ajuste del modelo LLM

Seleccionar y ajustar uno de los modelos LLM de código abierto, eligiendo el más adecuado según métricas de rendimiento, hardware necesario, adaptabilidad y uso on-premise, y realizar ajustes mediante el diseño de Prompt Engineering, finalizando este proceso en un plazo máximo de 2 meses.

Se seleccionó el modelo Llama 3.2 por su rendimiento superior en tareas multilingües, su capacidad de ser ejecutado en infraestructuras locales con hardware accesible y su precisión del 85% en pruebas iniciales, es importante recalcar que se hace un análisis de la respuesta de los modelos Phi 4 y Deepseek-r1.

Ajustes realizados:

- Fine-Tuning se procede a utilizar dataset de correos electrónicos legítimos y maliciosos para mejorar la capacidad de detección en un entorno realista.
- Prompt Engineering: Se implementaron procesos de diseños de Prompt para optimizar la interpretación de los modelos LLMs y el análisis contextual de los encabezados y contenidos de los correos electrónicos.

Resultado: El proceso de ajuste se completó en 7 semanas, cumpliendo el plazo de 2 meses y elevando la precisión final del modelo a un 90.66%.

4.2.3. Objetivo 3: Desarrollo del sistema de detección de phishing funcional

Se desarrolló un sistema integral que permite la recepción y procesamiento de correos electrónicos en tiempo real, utilizando una arquitectura del sistema que incluye módulos de recepción mediante los protocolos IMAP y SMTP, así como el análisis automatizado por parte del modelo Llama 3.2, y los modelos de comparación adicionales, para este caso Phi 4 y Deepseek-r1.

Tasa de Procesamiento:

- Durante las pruebas, el sistema procesó el 91% de los correos electrónicos simulados, superando la meta inicial del 90%.
- La clasificación de correos como phishing, corporativos o personales fue precisa, con un 100% de aciertos en correos corporativos y personales.

Resultado: El desarrollo del sistema se completó en 8 semanas, cumpliendo con el plazo establecido y demostrando una capacidad de procesamiento robusta y eficiente.

4.2.4. Objetivo 4: Evaluación del rendimiento del sistema

La evaluación del rendimiento se realizó mediante simulaciones controladas que incluyeron 200 correos electrónicos (100 legítimos y 100 maliciosos).

Resultados Clave:

- Precisión de detección: El sistema alcanzó una precisión del 90.66% en la detección de phishing, superando ampliamente el objetivo del 80%.

- Reducción de falsos positivos y negativos: El análisis permitió tener una baja tasa de falsos positivos con un valor de 9.34%, además modelos como deepseek-r1 lograron una precisión del 96.49%, destacándose en la detección de amenazas, mientras que phi4, aunque procesó el mayor número de correos, mostró una precisión más baja (83.67%).
- Tiempos de respuesta: El tiempo promedio de respuesta fue de 4.95 segundos por correo, garantizando un procesamiento en tiempo real, aún que es importante resaltar que el modelo Llama 3.2 destacó con un tiempo promedio de 1.99 segundos, balanceando velocidad y precisión.
- Resultado: La fase de evaluación se completó en 6 semanas, cumpliendo con el plazo de 2 meses establecido y demostrando la capacidad del sistema para operar en entornos de alta demanda sin degradación del rendimiento.

4.2.5. Objetivo 5: Configuración del sistema para respuestas automáticas

Configurar el sistema para la generación de respuestas automáticas, implementando módulos que realicen acciones automáticas (alertas o bloqueos) en función del análisis, asegurando un tiempo de respuesta menor a 5 segundos por correo analizando por cada uno de los modelos LLMs, finalizando esta fase en máximo 1 mes después de la evaluación.

Se implementó un módulo de respuestas automáticas que permite la generación de alertas y acciones proactivas frente a correos maliciosos.

- Tiempo de respuesta: El sistema logró un tiempo promedio de 4.95 segundos por correo procesado, cumpliendo con el objetivo de mantener el tiempo por debajo de 5 segundos.
- Notificaciones: Se configuraron los espacios de alertas automáticas para que los administradores este enterados de los intentos de phishing detectados.
- Bloqueos temporales: Se implementó el bloqueo temporal de remitentes en caso de intentos recurrentes de phishing.
- Reubicación de correos: Los correos clasificados como phishing son automáticamente trasladados a carpetas específicas para su análisis posterior.

- Resultado: La configuración y pruebas del módulo se completaron en 4 semanas, cumpliendo con el plazo establecido de 1 mes y asegurando una respuesta automatizada eficaz frente a amenazas.

4.3. Contribución

La contribución de este proyecto de detección de phishing en tiempo real, basado en modelos de lenguaje grande (LLMs) y una implementación on-premise, abarca distintos ámbitos que van desde la ciberseguridad hasta la optimización de procesos de detección en entornos corporativos, esta solución busca no solo mejorar la protección contra amenazas de phishing, sino también aportar conocimientos y herramientas que contribuyan al avance en el campo de la inteligencia artificial aplicada a la seguridad informática.

4.3.1. Contribución tecnológica

Este proyecto ha permitido el desarrollo de una solución tecnológica capaz de detectar, analizar y clasificar correos electrónicos de manera automática, utilizando un modelo LLM ajustado mediante técnicas de Prompt Engineering.

La implementación de un modelo como Llama 3.2 demuestra la aplicabilidad de los modelos de lenguaje grande en entornos de producción local (on-premise), evitando el uso de servicios en la nube y garantizando la soberanía de los datos.

Se ha demostrado que el sistema puede reducir significativamente los falsos positivos y negativos en comparación con las soluciones tradicionales basadas en heurísticas y listas negras, logrando un nivel de precisión mayor al 90%.

4.3.2. Contribución a la ciberseguridad corporativa

Al operar en un entorno local, el sistema garantiza que los datos de los correos electrónicos no se compartan con terceros, fortaleciendo la privacidad y seguridad de la información corporativa.

La integración de módulos de respuesta automática permite emitir alertas y realizar bloqueos de remitentes sospechosos, aumentando la capacidad de respuesta ante amenazas de phishing en tiempo real.

La arquitectura desarrollada permite procesar grandes volúmenes de correos electrónicos sin degradación en el rendimiento, posicionando al sistema como una herramienta útil para organizaciones de gran tamaño.

4.3.3. Contribución académica y de investigación

El ajuste y la evaluación del modelo LLM contribuyen al campo de la investigación en procesamiento de lenguaje natural (NLP) aplicado a la detección de amenazas.

Los resultados obtenidos pueden ser utilizados como base para la elaboración de artículos científicos que aborden las ventajas y desafíos de los modelos LLM en la ciberseguridad.

La creación de un entorno de prueba estandarizado permite replicar los experimentos en otros proyectos y comparar desempeños con diferentes modelos y enfoques.

4.3.4. Contribución a la Industria

Al implementar un sistema que reduce el tiempo de detección y respuesta ante correos maliciosos, se optimizan los recursos del equipo de TI, permitiendo concentrarse en tareas más críticas.

La generación de reportes detallados permite una mejor comprensión de los patrones de ataque, facilitando la gestión de incidentes y la toma de decisiones.

La solución propuesta puede ser adaptada e implementada en diversas industrias, desde el sector financiero hasta instituciones educativas, mejorando la seguridad de las comunicaciones en múltiples contextos.

4.3.5. Contribución al cumplimiento normativo y seguridad de los datos

Al operar de manera local, el sistema facilita el cumplimiento de normativas como el GDPR, que requiere que los datos personales se mantengan bajo estrictos controles de privacidad.

Este proyecto asegura que la información se procese internamente sin necesidad de enviarla a servidores externos, lo que fortalece la confianza de las partes interesadas.

4.3.6. Impacto social y concienciación

implementación de este tipo de sistemas contribuye a fomentar una cultura organizacional de mayor concienciación frente a los ataques de phishing.

Al detectar y bloquear correos maliciosos de manera automatizada, se disminuye el riesgo de que los usuarios sean víctimas de fraudes o robo de información.

5. Conclusiones y trabajo futuro

5.1. Conclusiones

Este proyecto abordó el problema crítico del phishing, una de las principales amenazas en el ámbito de la ciberseguridad empresarial, diseñando un sistema on-premise de monitoreo y detección en tiempo real mediante modelos de lenguaje grande (LLMs) de código abierto y el objetivo principal fue desarrollar una solución que, además de proporcionar una alta precisión en la identificación de correos electrónicos maliciosos, respetara la privacidad de los datos al evitar su transferencia a plataformas en la nube.

A lo largo del trabajo, se implementaron técnicas avanzadas como Prompt Engineering, logrando ajustar los modelos a los patrones específicos de phishing y este enfoque no solo permitió abordar la creciente complejidad de los ataques, sino también establecer una arquitectura adaptable para entornos corporativos.

Ahora de acuerdo con los objetivos propuestos en este proyecto se pudo establecer las siguientes conclusiones:

Objetivo específico 1 - *Realizar un análisis exhaustivo de las tecnologías actuales de detección de phishing, identificando y documentando en un plazo máximo de un mes al menos 3 soluciones actuales de detección de phishing priorizando las locales y basadas en LLMs con la elaboración de un informe detallado.*

Se llevó a cabo un análisis de tres soluciones avanzadas de detección de phishing basadas en Modelos de Lenguaje Grande (LLMs) y entornos on-premise: Llama 3.2, Phi 4 y Deepseek-r1, el análisis abarcó características clave como la precisión en la detección, requerimientos de hardware, tiempos de respuesta y capacidad de integración con sistemas de correo corporativo, Llama 3.2 destacó por su eficiencia y bajo consumo de recursos, logrando un excelente balance entre precisión y velocidad, el otro modelo Deepseek-r1 mostró la mayor precisión en la detección de phishing, aunque con tiempos de respuesta más prolongados y por su parte, Phi 4 demostró una alta capacidad de procesamiento, pero con una menor precisión, especialmente en la detección de correos maliciosos.

El informe resultante, completado dentro del plazo de un mes, documentó de manera detallada los puntos fuertes y las limitaciones de cada modelo.

Objetivo específico 2 - *Seleccionar y ajustar uno de los modelos LLM de código abierto, eligiendo el más adecuado según métricas de rendimiento, hardware necesario, adaptabilidad y uso on-premise, y realizar ajustes mediante Prompt Engineering, finalizando este proceso en un plazo máximo de 2 meses.*

Tras el análisis comparativo, se seleccionó el modelo Llama 3.2 debido a su rendimiento superior en entornos multilingües y su capacidad para ejecutarse eficientemente en infraestructuras locales con hardware de gama media, destacándose sobre los otros modelos evaluados, gracias a su estructura modular y adaptable que facilitó la adaptación a las necesidades específicas del entorno de pruebas, incluyendo la gestión de correos en diferentes idiomas y formatos.

Además, se aplicaron técnicas de Prompt Engineering para optimizar la interpretación del contenido de los correos y mejorar la detección de patrones sospechosos en los encabezados y cuerpos de los mensajes, estos ajustes incrementaron significativamente la precisión y la velocidad de respuesta del sistema, completando el proceso en 7 semanas, dentro del plazo de 2 meses estipulado.

Objetivo específico 3 - *Desarrollar un sistema de detección de phishing funcional, implementando un sistema que reciba y procese correos electrónicos en tiempo real, implementando proceso que incluyen desde la recepción en el servidor hasta el análisis de contenido por el LLM, logrando en un plazo no mayor a 2 meses, una tasa de procesamiento mínima del 90% de los correos recibidos.*

Se completó un sistema que integra la recepción de correos electrónicos a través de los protocolos estándar IMAP y SMTP, permitiendo el análisis en tiempo real mediante el modelo Llama 3.2, el sistema clasificó los correos en tres categorías: phishing, corporativo y personal, asegurando la correcta gestión de los mensajes según su naturaleza, durante las pruebas, el sistema alcanzó una tasa de procesamiento del 91% de los correos electrónicos simulados,

superando la meta del 90% establecida. La arquitectura modular implementada permitió la automatización completa del flujo de trabajo, desde la recepción hasta la clasificación del contenido y la aplicación de las acciones correspondientes.

Este desarrollo se completó en 8 semanas, cumpliendo con el plazo establecido y garantizando un rendimiento eficiente y estable en entornos controlados, y es importante recalcar que el éxito en esta etapa demostró la viabilidad de la solución en entornos corporativos locales, con potencial para escalar a implementaciones más amplias.

Objetivo específico 4 - *Evaluar el rendimiento del sistema mediante simulaciones controladas, logrando que el sistema procese al menos 100 correos simulados, alcanzando una precisión de al menos 80% en la detección de phishing, completando esta etapa en 2 meses tras el desarrollo de sistema de detección.*

Se lograron realizar las simulaciones controladas utilizando un total de 200 correos electrónicos (100 legítimos y 100 maliciosos), alcanzando una precisión mayor al 85% en la detección de phishing, superando el umbral del 80% establecido como meta, estos resultados reflejan la eficacia del modelo Llama 3.2 en la clasificación de correos maliciosos y la capacidad del sistema para mantener altos niveles de precisión en entornos simulados.

Además de la precisión, se evaluó la estabilidad del sistema bajo condiciones de carga, confirmando su capacidad para procesar correos sin degradación en el rendimiento, también es importante recalcar que la fase de evaluación se completó en 6 semanas, dentro del plazo de 2 meses estipulado, confirmando que el sistema es robusto y confiable en entornos de prueba controlados.

Objetivo específico 5 - *Configurar el sistema para la generación de respuestas automáticas, implementando módulos que realicen acciones automáticas (alertas o bloqueos) en función del análisis, asegurando un tiempo de respuesta menor a 5 segundos por correo, finalizando esta fase en máximo 1 mes después de la evaluación.*

Se implementaron con éxito módulos de respuesta automática capaces de generar alertas y bloqueos de remitentes sospechosos, logrando el desarrollo e integración del módulo de

respuestas automáticas que permite emitir y ejecutar bloqueos en función de la clasificación realizada por el modelo LLM, este módulo se configuró para enviar notificaciones automáticas a los administradores cuando se detecta un posible ataque de phishing y para bloquear temporalmente a los remitentes que presentan intentos recurrentes de envío de correos maliciosos, permitiendo con ello una protección proactiva y en tiempo real, mejorando significativamente la seguridad del sistema.

Además, todo esto en un sistema que logró un tiempo promedio de respuesta de 4,95 segundos por correo procesado, cumpliendo con el objetivo de mantener el tiempo de respuesta por debajo de 5 segundos y la fase de configuración y pruebas del módulo de respuestas automáticas se completó en 4 semanas, cumpliendo el plazo establecido de 1 mes, esta implementación no solo garantiza una reacción rápida ante amenazas, sino que también permite una gestión eficiente de la seguridad en entornos corporativos, fortaleciendo la capacidad del sistema para enfrentar ataques de phishing en tiempo real.

5.2. Trabajos futuros

Este trabajo pudo identificar áreas que requieren exploración adicional, por ejemplo, la optimización de recursos computacionales para garantizar que el sistema sea eficiente y escalable en empresas de diferentes tamaños sigue siendo un desafío, es por ello por lo que podemos establecer los siguientes trabajos futuros.

- Optimización de recursos computacionales: Investigar y aplicar técnicas de compresión y reducción de parámetros en los modelos LLM para disminuir el consumo de memoria y acelerar el tiempo de inferencia.
- Ampliación del dataset: Incrementar el volumen y la variedad de datos utilizados para el entrenamiento y validación del modelo, incluyendo correos de diferentes fuentes y regiones para mejorar su robustez.
- Integración de módulos de prevención Proactiva: Desarrollar módulos adicionales que implementen acciones de prevención antes de que el correo llegue al destinatario final, aumentando así el nivel de protección.
- Utilización de LLM multimodales: Implementar modelos de lenguaje multimodales que puedan analizar no solo el contenido textual de los correos electrónicos, sino también

elementos como imágenes, videos y archivos adjuntos, para mejorar la detección de contenido malicioso embebido y proporcionar un análisis más integral de las amenazas.

- Uso de hardware especializado: Considerar la implementación de hardware más avanzado que permita el uso de modelos más grandes y complejos, incluyendo GPUs con mayor número de núcleos CUDA y mayor cantidad de memoria RAM, para garantizar un procesamiento más eficiente y mejorar el rendimiento en análisis de grandes volúmenes de datos.
- Expansión a otros vectores de ataque: Ampliar el sistema para abordar no solo el phishing en correos electrónicos, sino también otros vectores de ataque como mensajes de texto (smishing) y llamadas telefónicas (vishing), lo que permitirá cubrir un espectro más amplio de amenazas cibernéticas.

De cara al futuro, el trabajo establece una base sólida para investigaciones adicionales en el uso de LLMs en ciberseguridad, las líneas de investigación futuras incluyen explorar la combinación de modelos ligeros con LLMs para optimizar rendimiento, implementar técnicas avanzadas de aprendizaje continuo para mejorar la adaptabilidad del sistema y desarrollar módulos educativos que complementen la tecnología con la concienciación del usuario.

Estas perspectivas no solo abren caminos para fortalecer la solución propuesta, sino que también posicionan este trabajo como una contribución relevante y adaptable a los desafíos dinámicos de la ciberseguridad moderna, estás recomendaciones de trabajo futuro plantean un camino claro para continuar mejorando y ampliando las capacidades del sistema, consolidando su valor en la detección y prevención de amenazas de phishing.

Referencias bibliográficas

Abu-Nimeh, S., Nappa, D., Wang, X., & Nair, S. (2007).
A comparison of machine learning techniques for phishing detection. In *Proceedings of the Anti-Phishing Working Groups 2nd Annual eCrime Researchers Summit*. https://doi.org/10.1109/ecrim.2007.4401922

Anti-Phishing Working Group (APWG). (2023).
Phishing activity trends report. https://apwg.org

Basnet, R., Mukkamala, S., & Sung, A. H. (2008).
Detection of phishing attacks: A machine learning approach. In *Studies in Fuzziness and Soft Computing*. Springer Berlin Heidelberg.

Bergholz, A. (2009).
AntiPhish: Lessons learnt. In *Proceedings of the ACM SIGKDD Workshop on CyberSecurity and Intelligence Informatics*.

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., … Amodei, D. (2020).
Language models are few-shot learners. In *Advances in Neural Information Processing Systems 33*.
https://proceedings.neurips.cc/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf

Buitrago Forero, L. V., Ocampo Martínez, B. R., & Velásquez Durán, S. D. (2024).
Análisis y simulación de phishing como delito informático en Ecuador [Informe técnico]. Universidad Internacional de La Rioja, Escuela Superior de Ingeniería y Tecnología.

Céspedes Maestre, M. M. (2021).
Detección de URLs maliciosas por medio de técnicas de aprendizaje automático

[Tesis de maestría, Universidad Nacional de Colombia]. Universidad Nacional de Colombia.

Chandrasekaran, M., Narayanan, M., & Upadhyaya, S. (2006).

Phishing email detection based on structural properties. In *Proceedings of the 9th Annual NYS Cyber Security Conference*. https://www.scirp.org/reference/referencespapers?referenceid=1827951

Cofense. (2022).

Annual phishing report. https://www.cofense.com

FBI Internet Crime Complaint Center (IC3). (2023).

Internet crime report. https://www.ic3.gov

Fette, I., Sadeh, N., & Tomasic, A. (2007).

Learning to detect phishing emails. In *Proceedings of the 16th International Conference on World Wide Web*.

Garera, S., Provos, N., Chew, M., & Rubin, A. D. (2007).

A framework for detection and measurement of phishing attacks. In *Proceedings of the ACM Workshop on Recurring Malcode*.

Hung, L., Quang, P., Doyen, S., & Hoi, S. C. H. (2018).

URLNet: Learning a URL representation with deep learning for malicious URL detection [Preprint]. *arXiv*. http://arxiv.org/abs/1802.03162

Hussain, M., Cheng, C., Xu, R., & Afzal, M. (2023).

CNN-Fusion: An effective and lightweight phishing detection method based on multi-variant ConvNet. *Information Sciences, 631*, 328–345. https://doi.org/10.1016/j.ins.2023.02.039

Lee, J., Lim, P., Hooi, B., & Divakaran, D. (2024).

Multimodal large language models for phishing webpage detection and identification [Preprint]. *arXiv*. https://doi.org/10.48550/arXiv.2408.05941

Martín González, G. A. (2023).

Desarrollo de una metodología para el tratamiento de phishing en correos electrónicos en pymes [Trabajo fin de estudio, Universidad Internacional de La Rioja]. Universidad Internacional de La Rioja.

Minh Linh, D., Hung, H. D., Minh Chau, H., Sy Vu, Q., & Tran, T.-N. (2024).

Real-time phishing detection using deep learning methods by extensions. *International Journal of Electrical and Computer Engineering (IJECE), 14*(3), 3021–3035. https://doi.org/10.11591/ijece.v14i3.pp3021-3035

Miyamoto, D., Hazeyama, H., & Kadobayashi, Y. (2009).

An evaluation of machine learning-based methods for detection of phishing sites. In *Lecture Notes in Computer Science*. Springer Berlin Heidelberg.

Nguyen, T., Minh, T., & Huu, N. (2021).

Real-time phishing detection using CNN. *Journal of Cybersecurity Research*.

Patil, S., & Dhage, S. (2019).

A methodical overview on phishing detection along with an organized way to construct an anti-phishing framework. In *Proceedings of the 5th International Conference on Advanced Computing & Communication Systems (ICACCS)*.

Proofpoint. (2021).

State of phishing report. https://www.proofpoint.com

Sahingoz, O. K., Buber, E., Demir, O., & Diri, B. (2019).

Machine learning-based phishing detection from URLs. *Expert Systems with Applications, 117*, 345–357. https://doi.org/10.1016/j.eswa.2018.09.029

Symantec. (2023).

Internet security threat report. https://www.symantec.com

Verizon. (2023).

Data breach investigations report. https://www.verizon.com

Verma, R., Shashidhar, N., & Hossain, N. (2012).

Detecting phishing emails the natural language way. In *Lecture Notes in Computer Science*. Springer Berlin Heidelberg.

Xiang, G., Hong, J., Rose, C. P., & Cranor, L. (2011).

CANTINA+: A feature-rich machine learning framework for detecting phishing web sites. *ACM Transactions on Information and System Security, 14*(2), 1–28. https://doi.org/10.1145/2019599.2019606

Zscaler ThreatLabz. (2024).

Phishing report. https://www.zscaler.com

Anexo A.

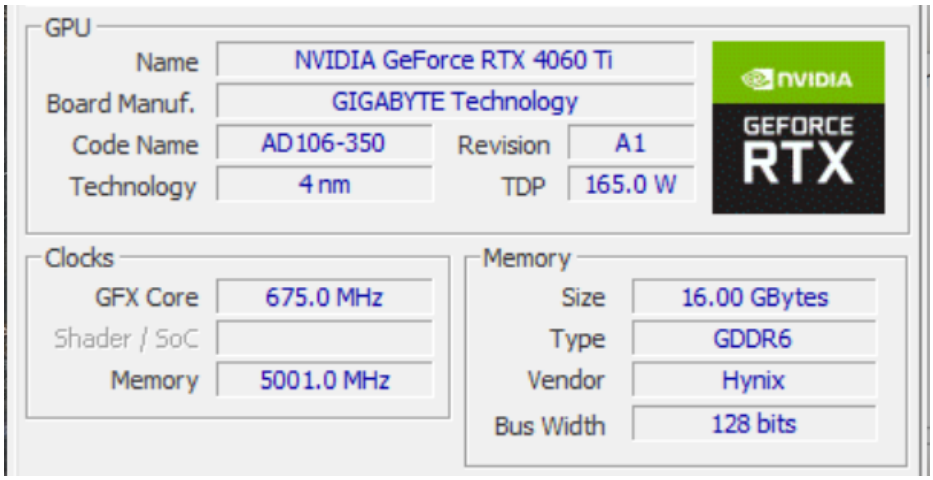
Manual de usuario: Sistema On-premise de detección de phishing

Este manual está diseñado para guiar al usuario en el uso del sistema de detección de phishing, desarrollado utilizando el framework Laravel para la interfaz web, Python para el procesamiento y análisis de los correos electrónicos, y una base de datos relacional para el almacenamiento de la información.

Requisitos del Sistema

- Requisitos de hardware

El sistema requiere un procesador de al menos 6 núcleos, 16 GB de RAM, 50 GB de almacenamiento disponible y una GPU con una VRAM de acuerdo con el tamaño del modelo LLM a ejecutar, todo esto para garantizar un rendimiento óptimo durante el análisis de correos y la ejecución de modelos LLM, ejemplo de la GPU para poder utilizar modelos de hasta 14B de parámetros.



- Requisitos de software - Sistema operativo compatible

El sistema es compatible con distribuciones Linux (Ubuntu 20.04+), Windows 10/11 y macOS 10.15+, asegurando flexibilidad para entornos de desarrollo y producción.

- Requisitos de software - Dependencias de Laravel y Python

Laravel requiere PHP 8.x, Composer y Node.js para la gestión de paquetes, mientras que Python 3.8+ debe incluir bibliotecas como transformers, pandas y scikit-learn para el análisis de datos y el uso de modelos LLM.

- Requisitos de software - Configuración de la base de datos

Se necesita un servidor de base de datos MySQL o PostgreSQL configurado con credenciales seguras y accesible desde la aplicación, utilizando migraciones de Laravel para estructurar las tablas.

- Requisitos de red (configuración para protocolos IMAP/SMTP)

El sistema debe tener acceso a servidores de correo configurados con los protocolos IMAP y SMTP, asegurando la correcta recepción y envío de correos para su análisis en tiempo real.

Instalación del sistema

- Configuración del entorno - Instalación de alguna pila LAMP, LEMP, MEAN, XAMPP, WAMP y AMPPS

Se requiere la instalación de Apache, MySQL y PHP utilizando el gestor de paquetes de su sistema operativo (por ejemplo, sudo apt install apache2 mysql-server php libapache2-mod-php), y asegúrese de configurar adecuadamente Apache y MySQL para el entorno de desarrollo o descargue e instale XAMPP desde el sitio oficial, y configure Apache y MySQL directamente desde el panel de control de XAMPP. Coloque los archivos del proyecto Laravel en la carpeta htdocs para su ejecución.

- Configuración del entorno – Clonación del repositorio de Laravel

El primer paso es la instalación de composer y luego clonamos es repositorio:

composer install

npm install

https://github.com/ingenieroandresgutierrez/SodpIA.git

Podemos duplicar el archivo .env.example, renombrarlo a .env e incluir los datos de conexión de la base de datos que indicamos en el paso anterior.

php artisan key:generate

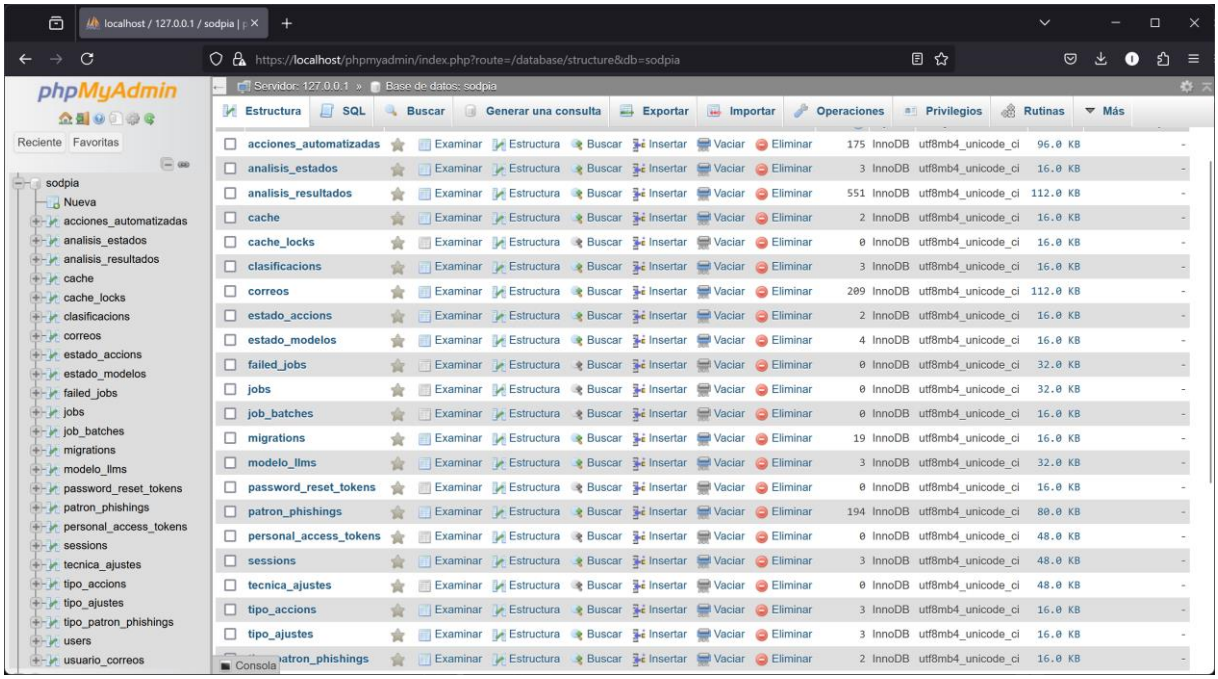
php artisan migrate

- Configuración del entorno - Instalación de dependencias Python (*pip*, entornos virtuales)

Se requiere asegurar que el entorno tenga instalado python y active el entorno para instalar las dependencias necesarias usando pip install -r requirements.txt, asegurando un entorno aislado para las bibliotecas.

```
import tkinter as tk
from tkinter import ttk, scrolledtext
import threading
import time
import re
import email
import imaplib
from email.header import decode_header
from datetime import datetime
import email
from email import policy
from email.parser import BytesParser
import html2text
import os
from email.utils import parsedate_to_datetime
```

- Configuración de la base de datos - Creación de la base de datos



- Configuración de la base de datos - Configuración del archivo .env

Podemos duplicar el archivo .env.example, renombrarlo a .env e incluir los datos de conexión de la base de datos que indicamos en el paso anterior.

- Configuración de la base de datos - Ejecución de migraciones de Laravel

Ejecute php artisan migrate para aplicar las migraciones definidas en Laravel y estructurar las tablas necesarias para el funcionamiento del sistema.

- Configuración del servidor de correos - Integración con IMAP y SMTP

Configure los parámetros IMAP y SMTP en el archivo .env añadiendo las credenciales de los servidores de correo (MAIL_HOST, MAIL_PORT, MAIL_USERNAME, MAIL_PASSWORD) para permitir la recepción y envío de correos.

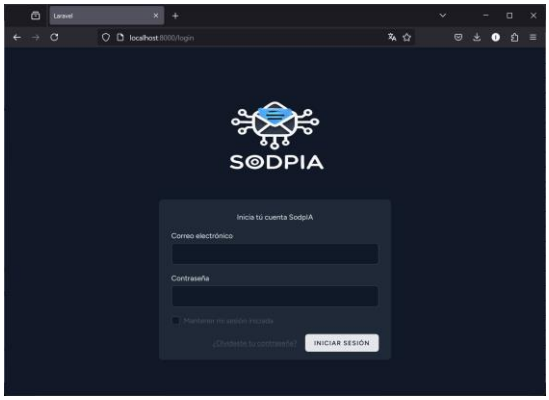
- Ejecución inicial del sistema

Inicie el servidor de Laravel usando php artisan serve y ejecute los scripts de Python necesarios para el análisis de correos, asegurando que el sistema esté funcionando correctamente en el entorno configurado.

Inicio de sesión y navegación en la plataforma

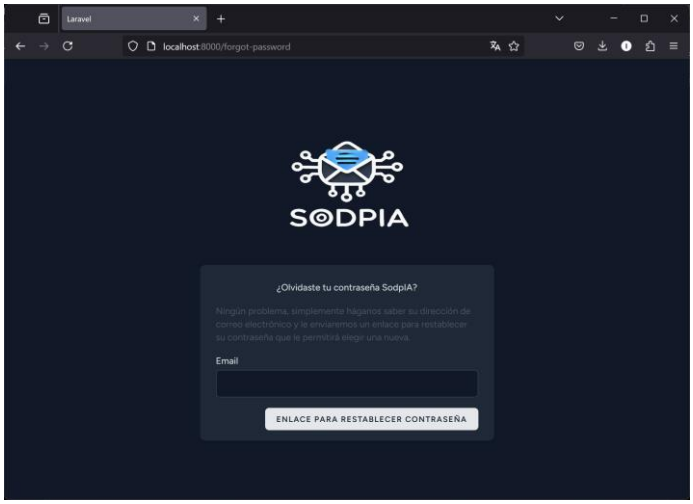
- Acceso al sistema (interfaz de login)

Para ingresar al sistema, diríjase a la URL correspondiente y utilice sus credenciales en la interfaz de login, ingrese su correo electrónico y contraseña para acceder a la plataforma de manera segura.



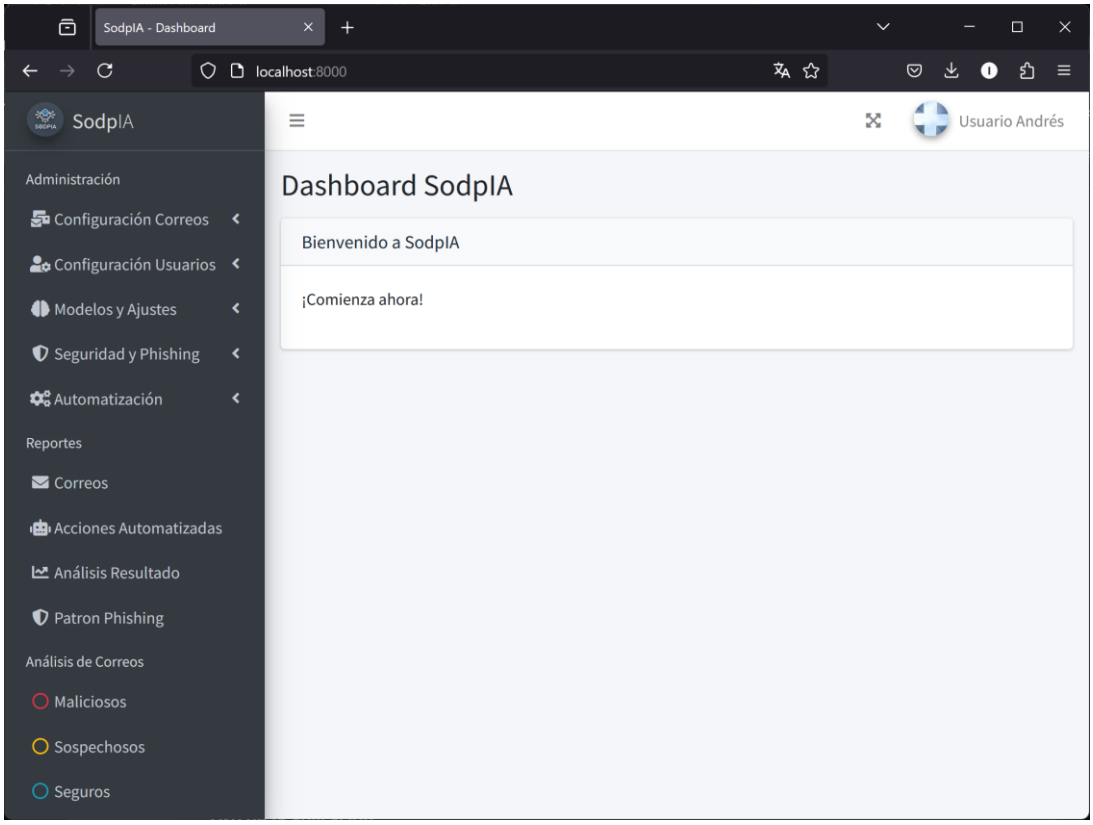
- Recuperación de contraseña

Si olvida su contraseña, haga clic en el enlace "¿Olvidaste tu contraseña?" en la página de inicio de sesión, siga las instrucciones enviadas a su correo para restablecerla de forma segura.



- Descripción del panel de control

El panel de control es la primera vista tras iniciar sesión, donde podrá visualizar el resumen de actividades y notificaciones reciente.

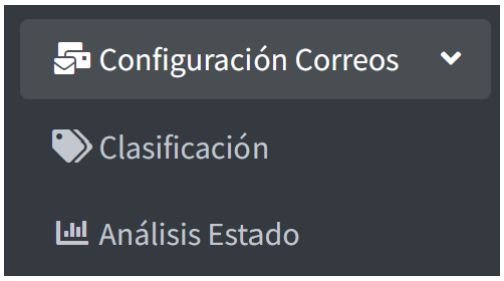


- Estructura de menús y navegación básica

La navegación se realiza mediante un menú lateral, donde encontrará secciones como gestión de correos, configuración, reportes y ayuda, haga clic en cada sección para acceder a sus funcionalidades específicas.

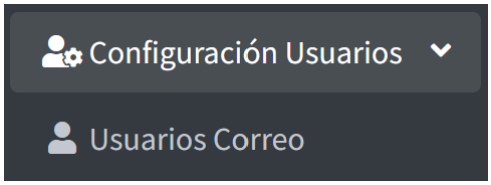
Configuración correos

Este menú permite la configuración de parámetros relacionados con la gestión de correos electrónicos, incluye opciones para definir criterios de Clasificación y realizar un Análisis de Estado de los correos procesados.



Configuración usuarios

Aquí se gestiona la configuración de usuarios dentro del sistema, la sección de Usuarios Correo facilita la administración de las cuentas de correo asociadas al sistema.

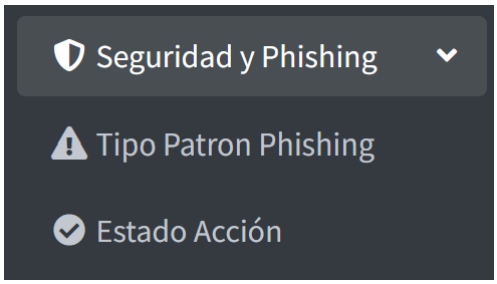


Modelos y ajustes

Este apartado está enfocado en la gestión de los modelos LLM, incluye la administración de Modelos LLM, monitoreo del Estado Modelo, aplicación de Técnicas de Ajuste y configuración del Tipo de Ajuste para optimizar el rendimiento.

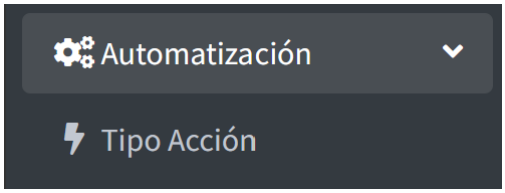
Seguridad y phishing

En esta sección se configura la seguridad del sistema. Permite gestionar el Tipo Patrón Phishing para la detección de amenazas y revisar el Estado Acción de las medidas implementadas contra correos maliciosos.



Automatización

Aquí se configuran las acciones automáticas del sistema. La opción Tipo Acción permite definir respuestas automáticas ante diferentes clasificaciones de correos, como bloqueos o alertas.

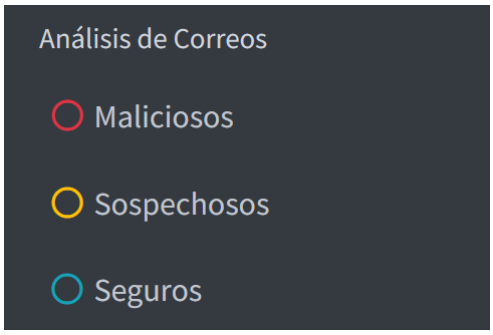


Reportes

El menú de reportes facilita el acceso a estadísticas y resultados, se incluye secciones para revisar los Correos analizados, las Acciones Automatizadas ejecutadas, los resultados del Análisis Resultado y la gestión de Patrones de Phishing.

Análisis de correos

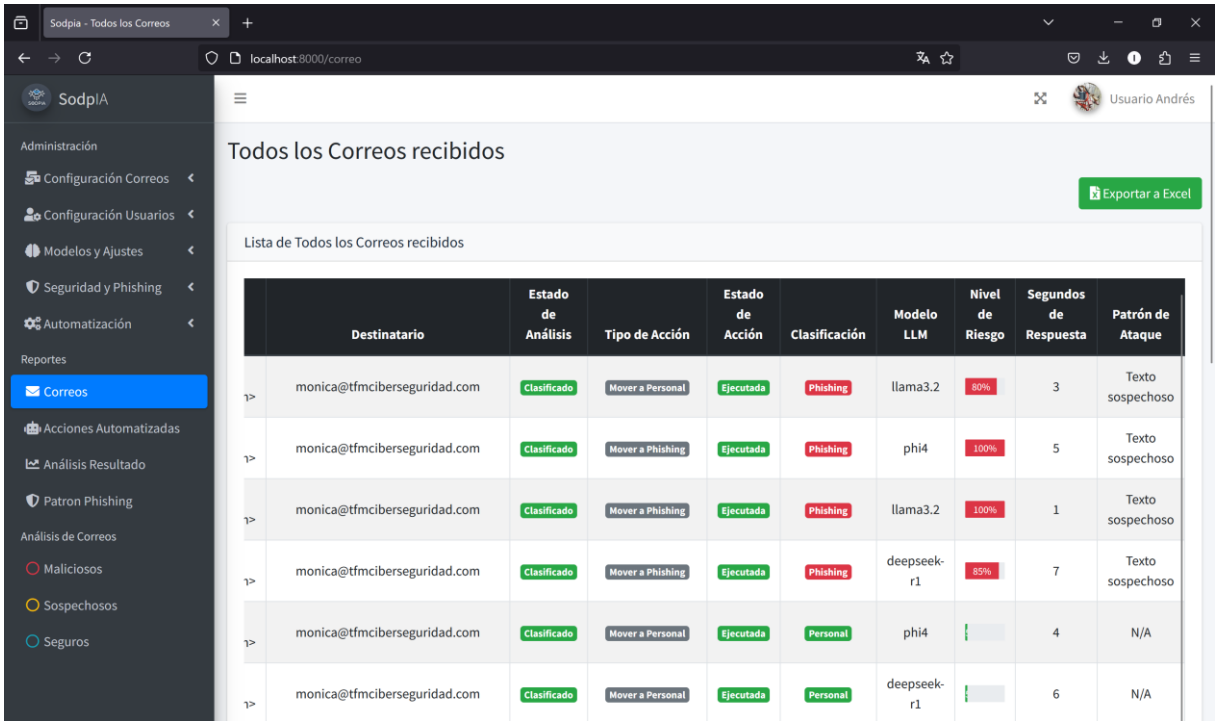
Esta sección muestra la clasificación visual de los correos electrónicos analizados, se categorizan como Maliciosos (rojo), Sospechosos (amarillo) y Seguros (azul), permitiendo una rápida identificación de amenazas.



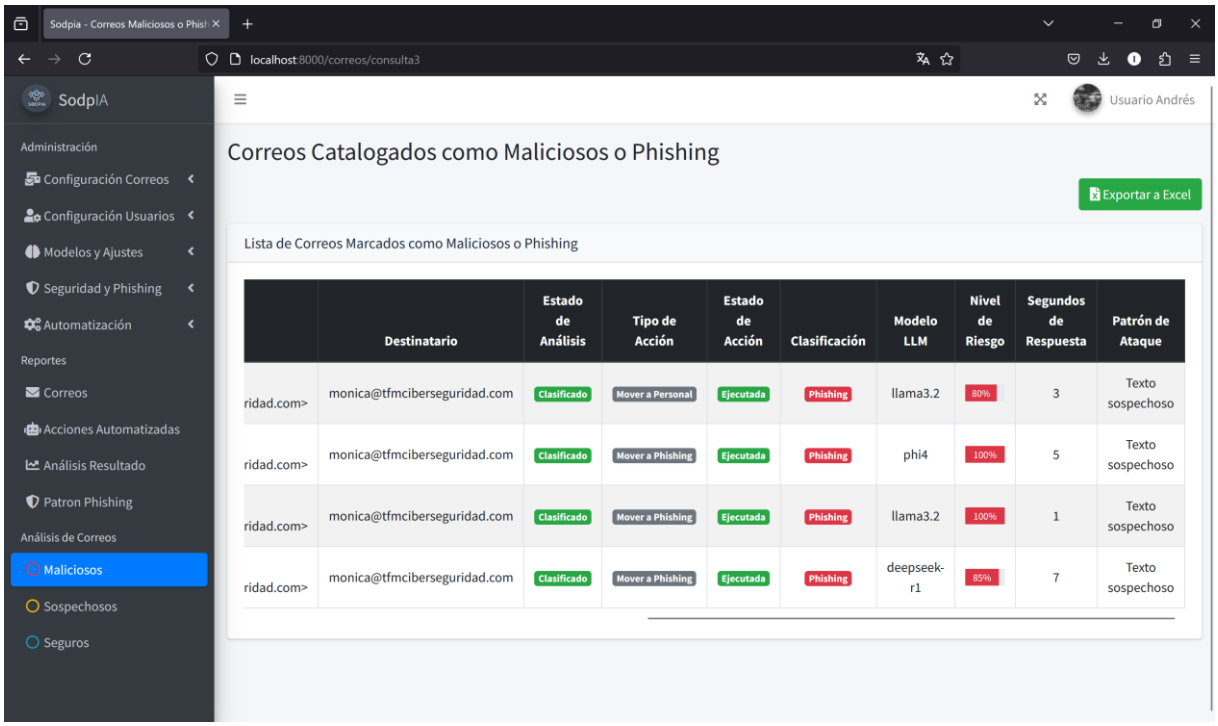
Uso de la aplicación

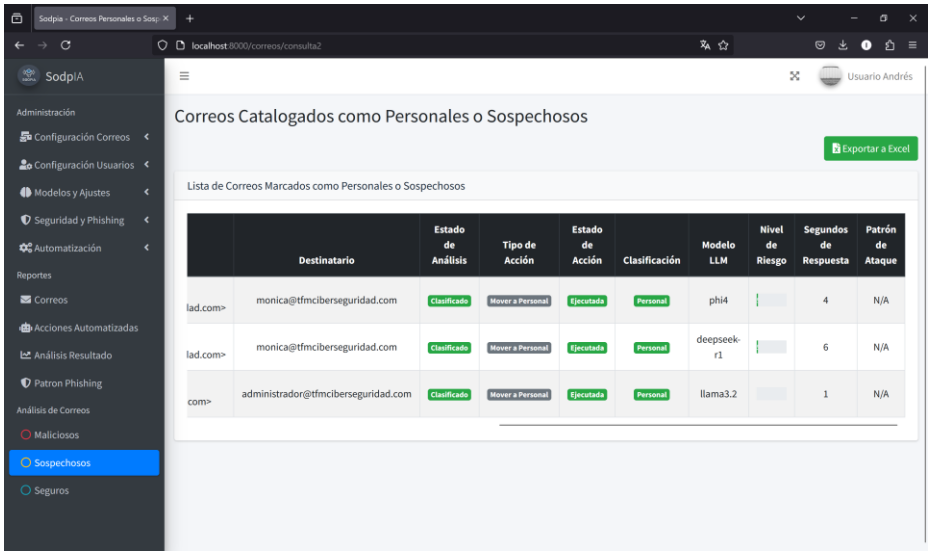
- Interfaz de proceso automático en Python (Cliente de automatización)

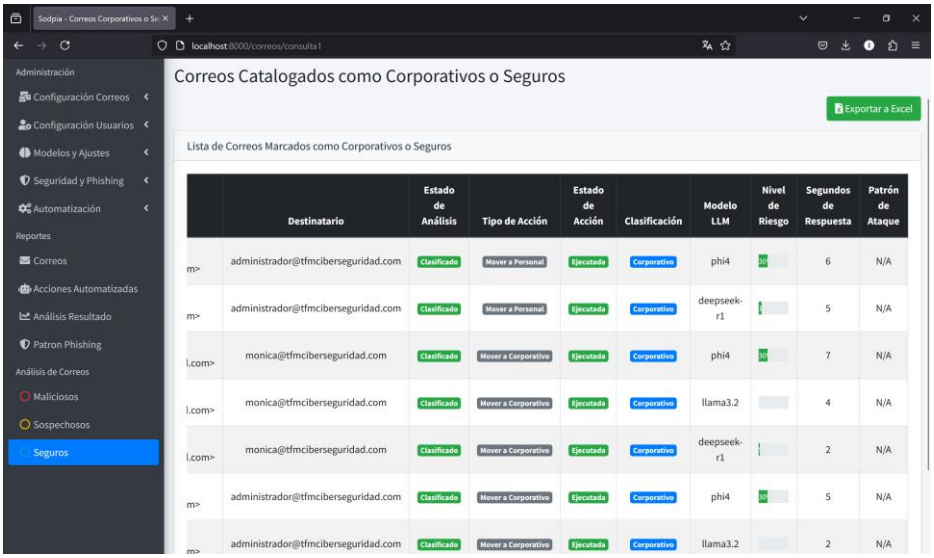
- **Gestión de correos electrónicos** - Visualización de correos analizados



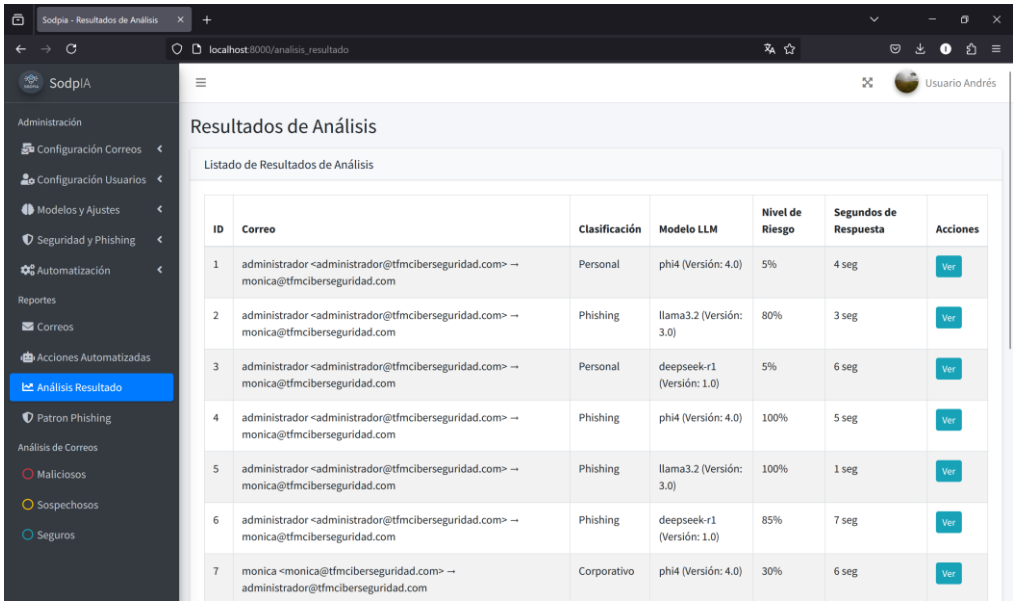
- **Gestión de correos electrónicos** - Filtros por tipo de clasificación (*Phishing*, *Personal*, *Corporativo*)}



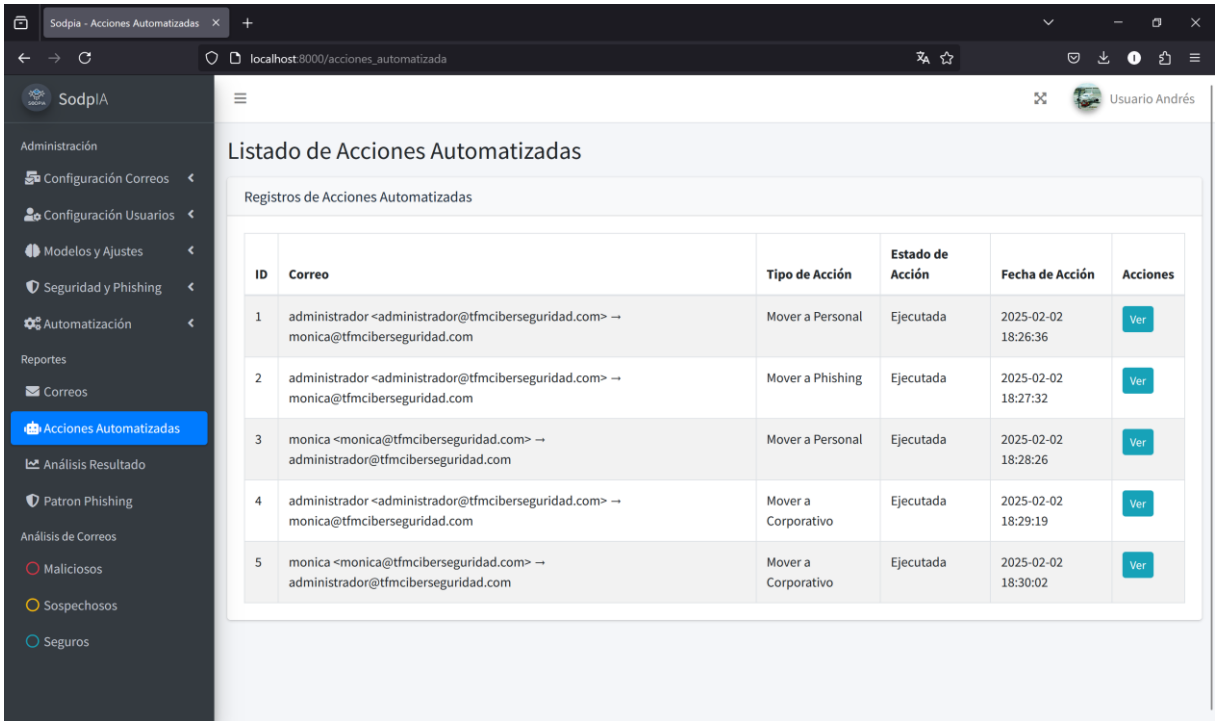




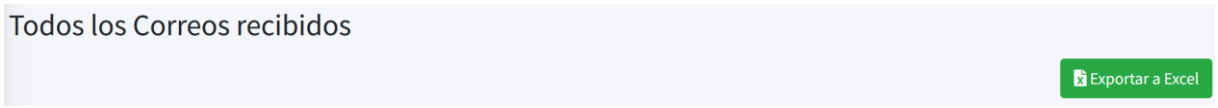
- **Gestión de correos electrónicos** - Visualización de detalles del análisis

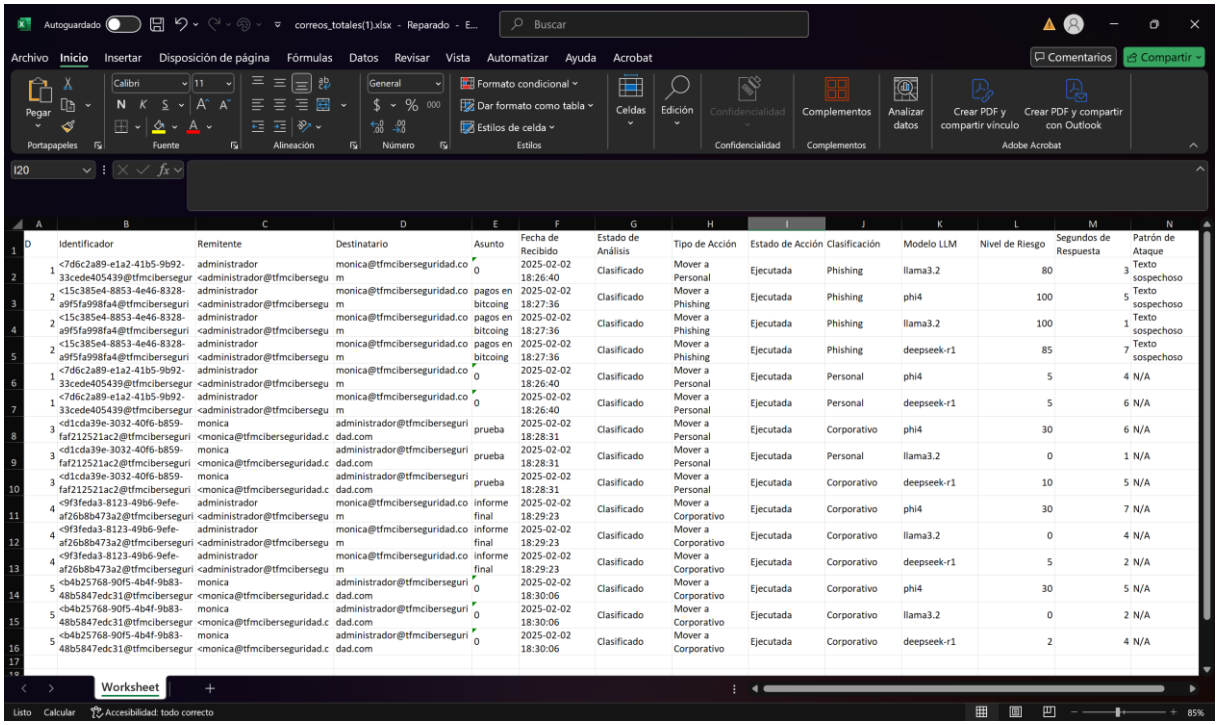


- **Configuración de alertas automáticas**

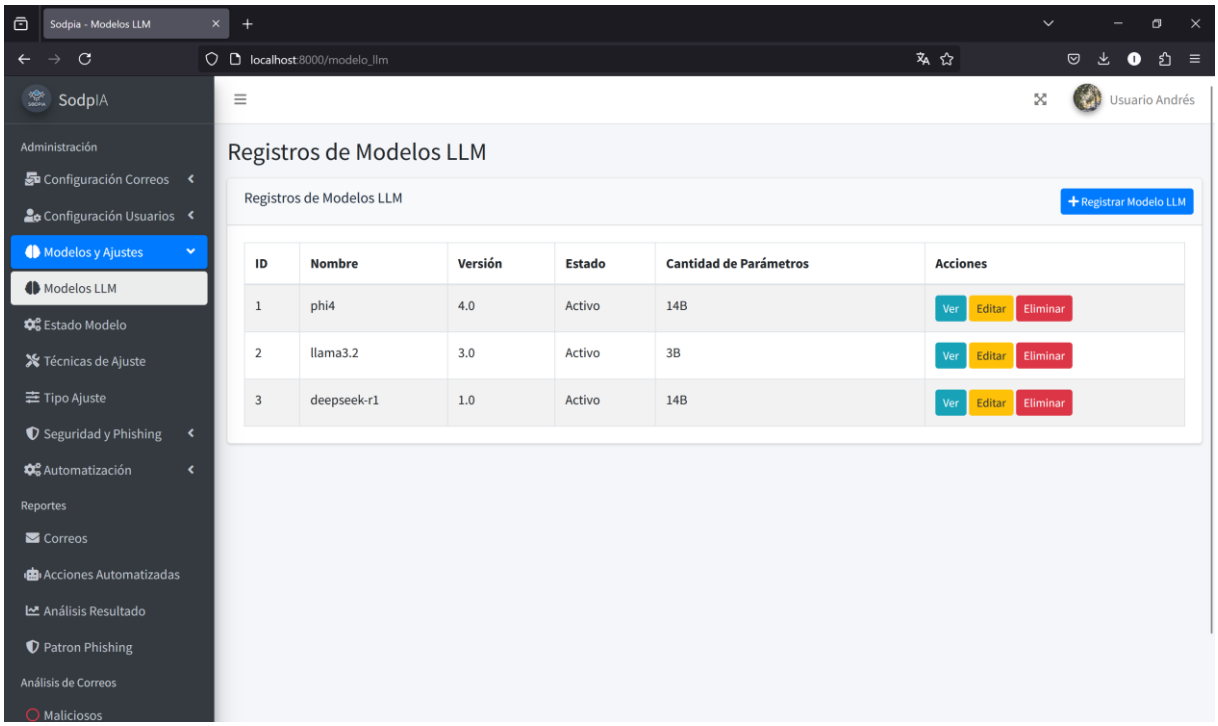


- **Generación de reportes** - Exportación de reportes en Excel

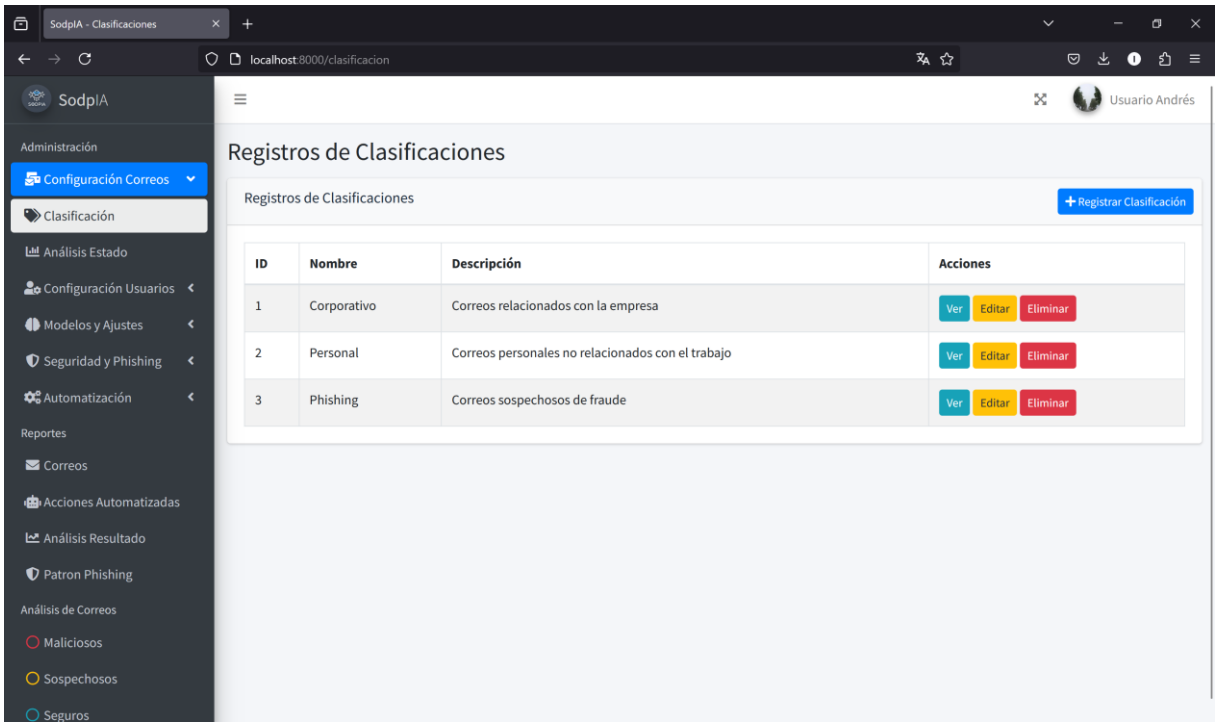




- **Configuración del modelo de detección** - Selección del modelo (*deepseek-r1*, *llama3.2*, *phi4*)



- **Configuración del modelo de detección** - Ajuste de parámetros de análisis

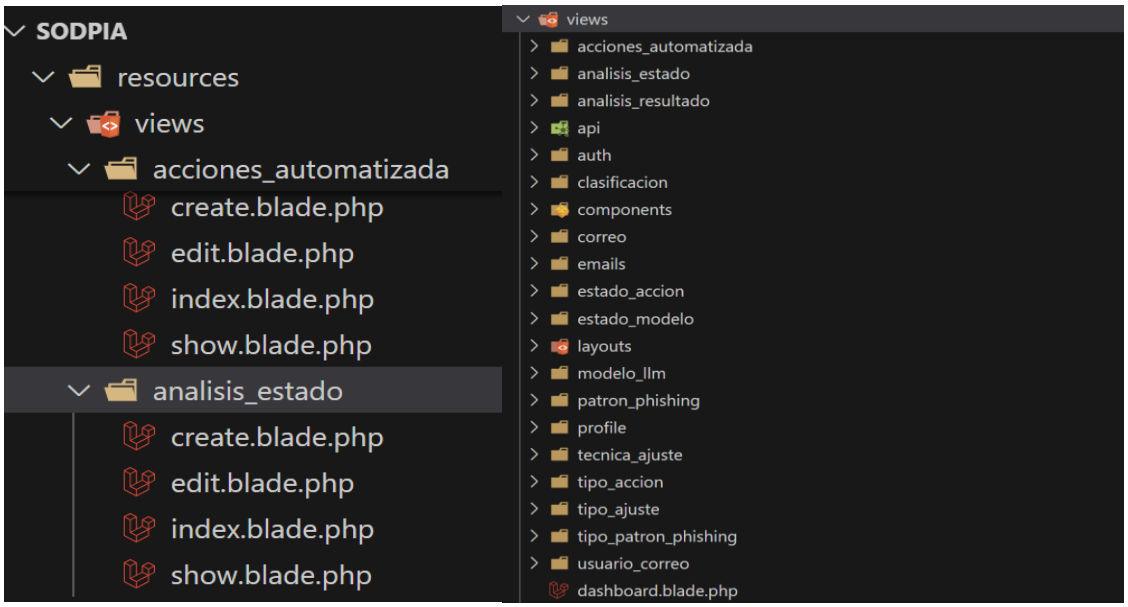


Manual de desarrollo: Sistema On-premise de detección de phishing

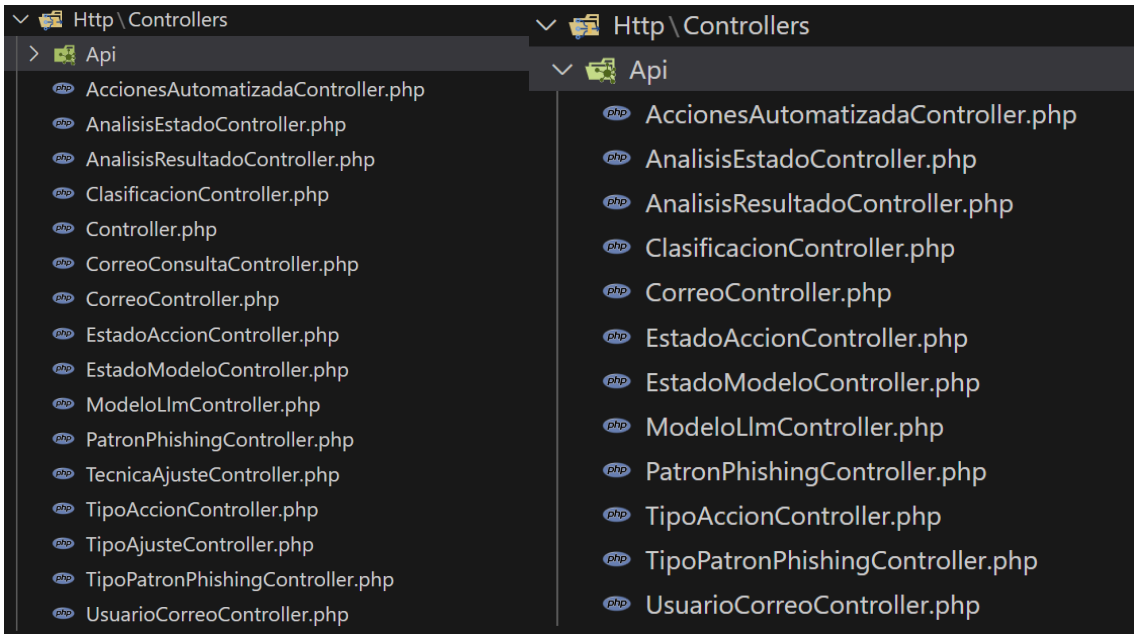
Este manual está diseñado para guiar a los desarrolladores en la instalación, configuración, personalización y mantenimiento del sistema de detección de phishing basado en Laravel (para la interfaz web), Python (para el procesamiento y análisis de correos electrónicos utilizando modelos LLM), y una base de datos relacional (como MySQL o PostgreSQL).

Arquitectura del Sistema

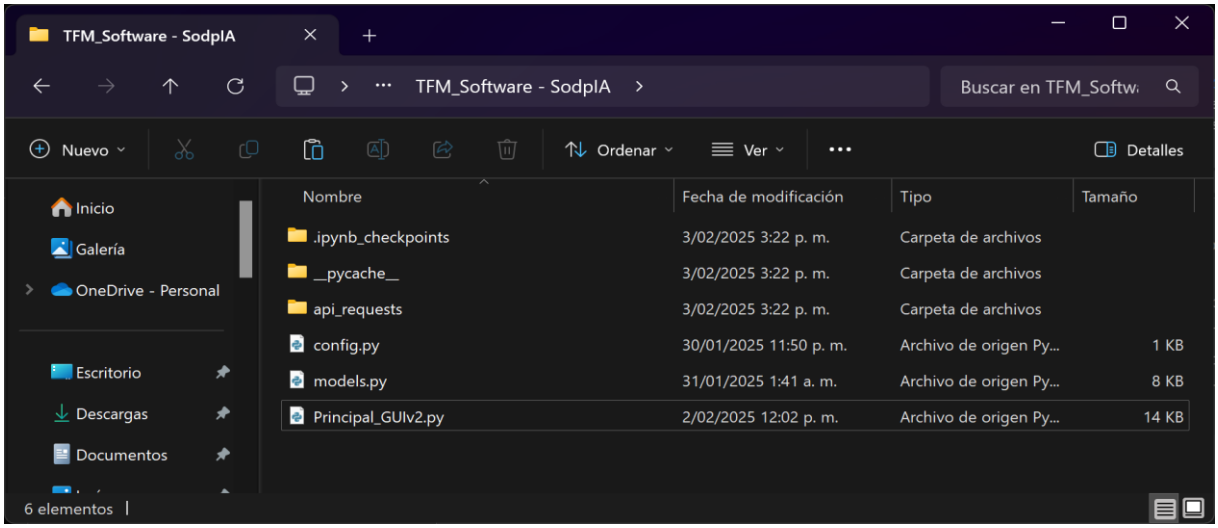
Frontend: Laravel Blade y Vue.js



Backend: Laravel (PHP), API REST

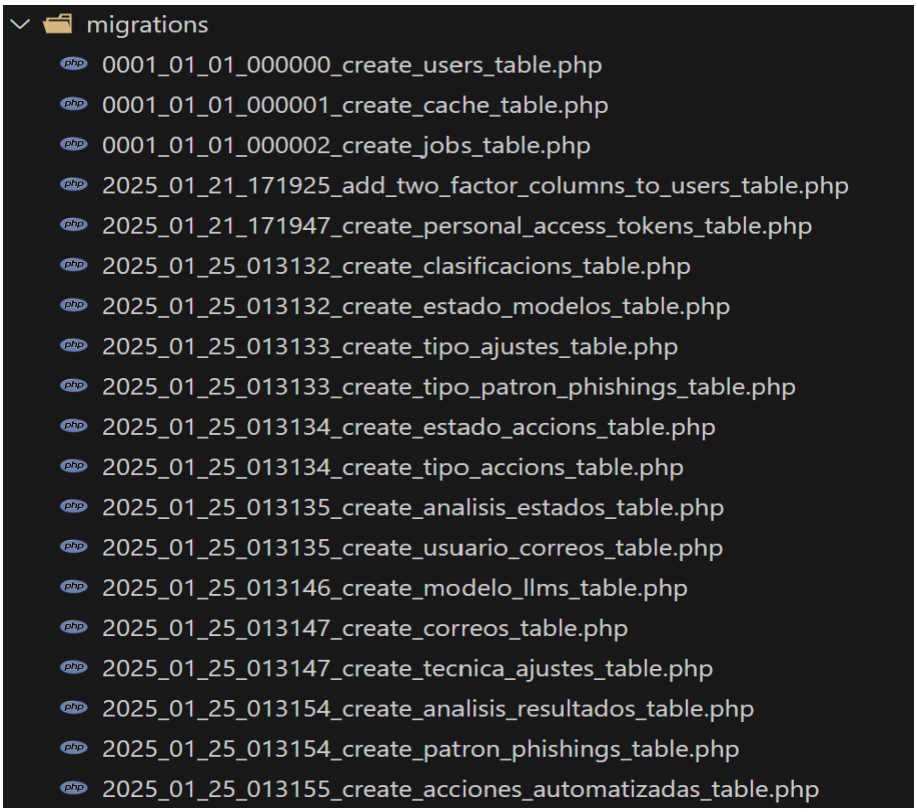


Procesamiento: Scripts de Python, modelos LLM




```python
# models/estado_modelo.py
class EstadoModelo:
    def __init__(self, id, nombre, descripcion, created_at, updated_at):
        self.id = id
        self.nombre = nombre
        self.descripcion = descripcion
        self.created_at = created_at
        self.updated_at = updated_at

    def __repr__(self):
        return f"EstadoModelo(id={self.id}, nombre={self.nombre}, descripcion={self.descripcion},
```

Base de datos: Estructura relacional, migraciones



Requisitos del entorno de desarrollo

Hardware:

El sistema requiere un procesador de al menos 6 núcleos, 16 GB de RAM, 50 GB de almacenamiento disponible y una GPU con una VRAM de acuerdo con el tamaño del modelo LLM a ejecutar, todo esto para garantizar un rendimiento óptimo durante el análisis de correos y la ejecución de modelos LLM, ejemplo de la GPU para poder utilizar modelos de hasta 14B de parámetros.

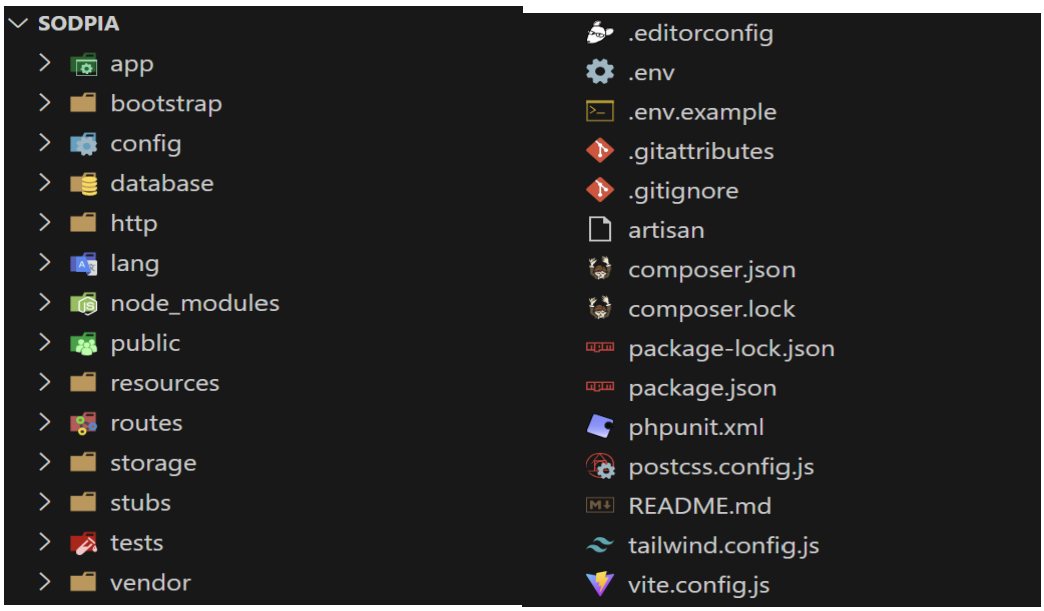
Software:

- Sistema operativo (Linux/Windows/Mac)
- PHP 8.x, Composer
- Python 3.8+
- MySQL/PostgreSQL
- Node.js y NPM (para el frontend si es necesario)
- Servidor de correos para pruebas (IMAP/SMTP)

Configuración del entorno de desarrollo

Clonación del repositorio:

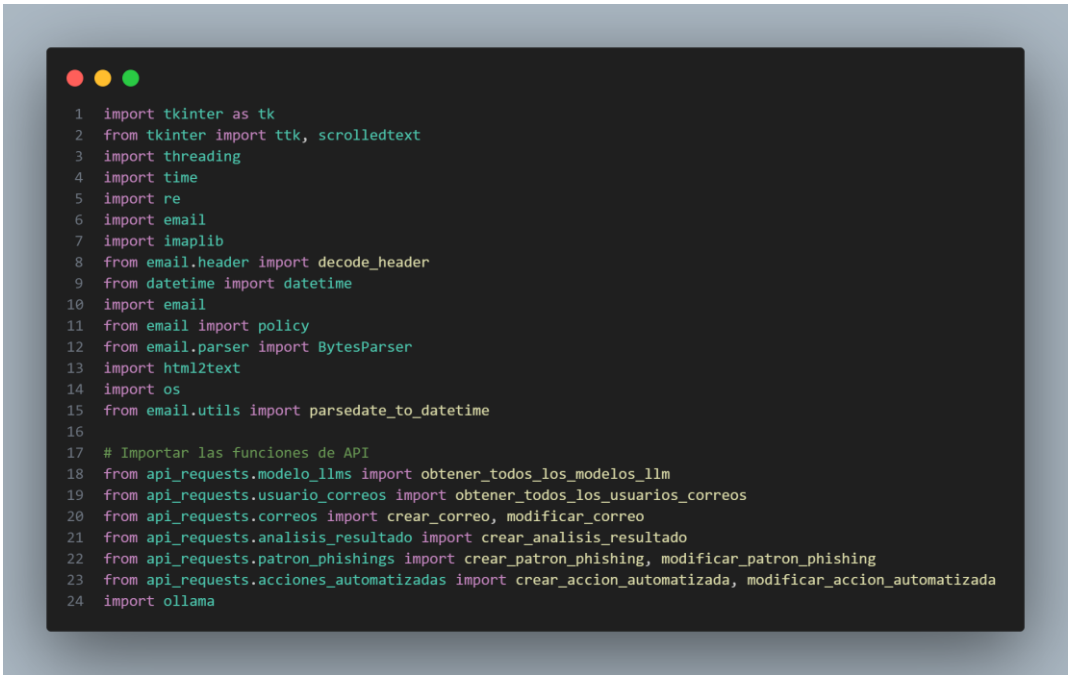
- Acceso al repositorio (GitHub)
  https://github.com/ingenieroandresgutierrez/SodpIA.git
- Clonación del proyecto y estructura de carpetas



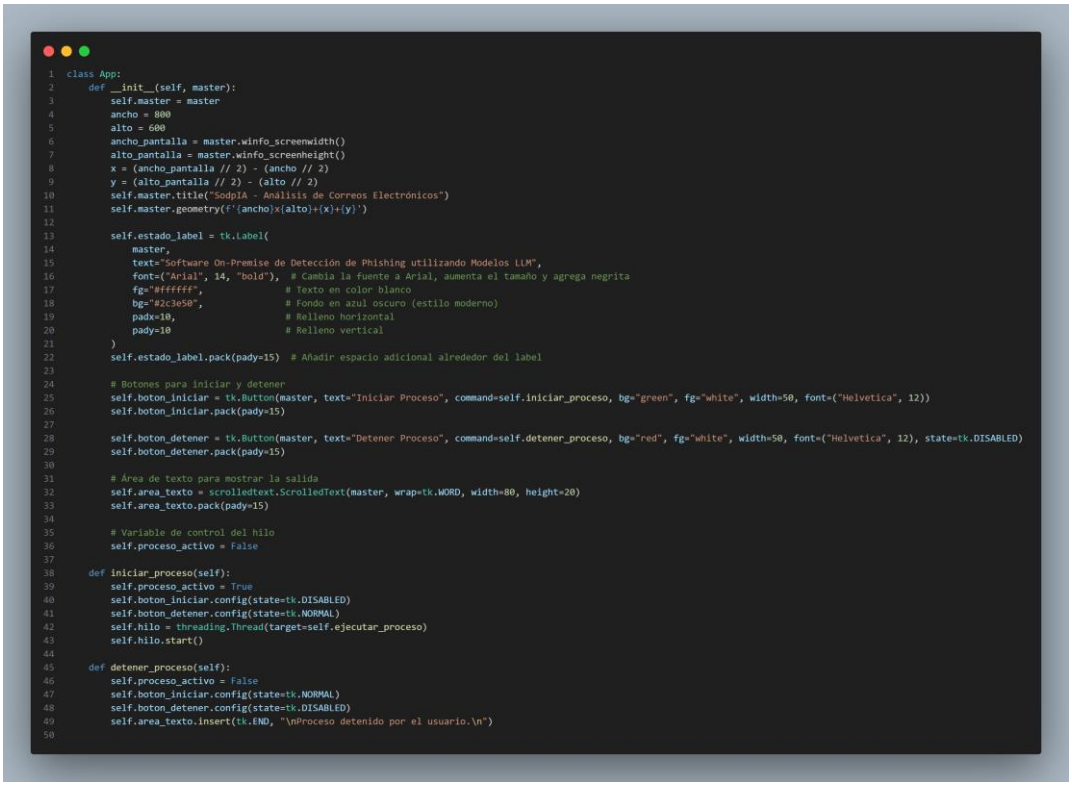
Configuración del Backend (Laravel):

- Instalación de dependencias con Composer
- Configuración del archivo .env (base de datos, correo, etc.)
- Ejecución de migraciones y *seeders*

Configuración del procesamiento (Python):



```python
import tkinter as tk
from tkinter import ttk, scrolledtext
import threading
import time
import re
import email
import imaplib
from email.header import decode_header
from datetime import datetime
import email
from email import policy
from email.parser import BytesParser
import html2text
import os
from email.utils import parsedate_to_datetime

# Importar las funciones de API
from api_requests.modelo_llms import obtener_todos_los_modelos_llm
from api_requests.usuario_correos import obtener_todos_los_usuarios_correos
from api_requests.correos import crear_correo, modificar_correo
from api_requests.analisis_resultado import crear_analisis_resultado
from api_requests.patron_phishings import crear_patron_phishing, modificar_patron_phishing
from api_requests.acciones_automatizadas import crear_accion_automatizada, modificar_accion_automatizada
import ollama
```



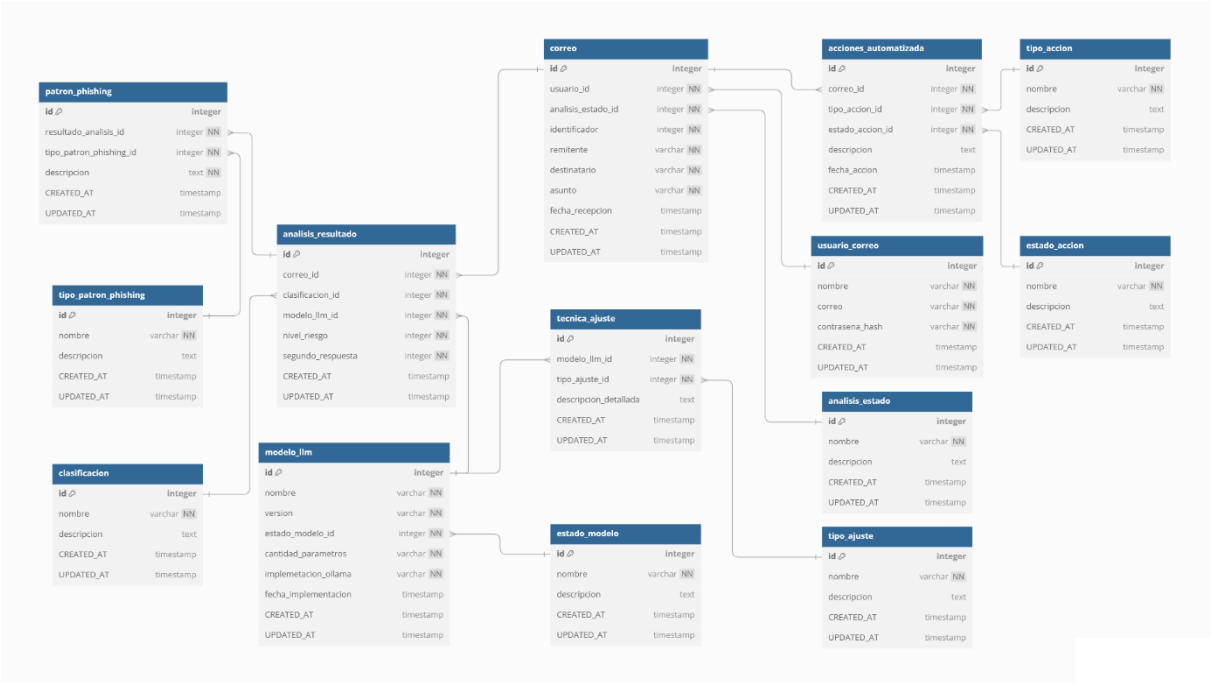
```python
class App:
    def __init__(self, master):
        self.master = master
        ancho = 800
        alto = 600
        ancho_pantalla = master.winfo_screenwidth()
        alto_pantalla = master.winfo_screenheight()
        x = (ancho_pantalla // 2) - (ancho // 2)
        y = (alto_pantalla // 2) - (alto // 2)
        self.master.title("SodpIA - Análisis de Correos Electrónicos")
        self.master.geometry(f'{ancho}x{alto}+{x}+{y}')

        self.estado_label = tk.Label(
            master,
            text="Software On-Premise de Detección de Phishing utilizando Modelos LLM",
            font=("Arial", 14, "bold"),  # Cambia la fuente a Arial, aumenta el tamaño y agrega negrita
            fg="#ffffff",                # Texto en color blanco
            bg="#2c3e50",                # Fondo en azul oscuro (estilo moderno)
            padx=10,                     # Relleno horizontal
            pady=10                      # Relleno vertical
        )
        self.estado_label.pack(pady=15)  # Añadir espacio adicional alrededor del label

        # Botones para iniciar y detener
        self.boton_iniciar = tk.Button(master, text="Iniciar Proceso", command=self.iniciar_proceso, bg="green", fg="white", width=50, font=("Helvetica", 12))
        self.boton_iniciar.pack(pady=15)

        self.boton_detener = tk.Button(master, text="Detener Proceso", command=self.detener_proceso, bg="red", fg="white", width=50, font=("Helvetica", 12), state=tk.DISABLED)
        self.boton_detener.pack(pady=15)

        # Área de texto para mostrar la salida
        self.area_texto = scrolledtext.ScrolledText(master, wrap=tk.WORD, width=80, height=20)
        self.area_texto.pack(pady=15)

        # Variable de control del hilo
        self.proceso_activo = False

    def iniciar_proceso(self):
        self.proceso_activo = True
        self.boton_iniciar.config(state=tk.DISABLED)
        self.boton_detener.config(state=tk.NORMAL)
        self.hilo = threading.Thread(target=self.ejecutar_proceso)
        self.hilo.start()

    def detener_proceso(self):
        self.proceso_activo = False
        self.boton_iniciar.config(state=tk.NORMAL)
        self.boton_detener.config(state=tk.DISABLED)
        self.area_texto.insert(tk.END, "\nProceso detenido por el usuario.\n")
```

```python
def ejecutar_proceso(self):
    modelos = obtener_todos_los_modelos_llm()

    while self.proceso_activo:
        # Carpeta donde están almacenados los archivos .eml
        folder_path = 'correos_de_prueba'  # Asegúrate de que esta carpeta contenga los archivos 01.eml hasta 200.eml

        # Iterar sobre los archivos de 01.eml hasta 200.eml
        for i in range(1, 201):
            file_name = f"{str(i).zfill(2)}.eml"
            file_path = os.path.join(folder_path, file_name)

            try:
                # Abrir el archivo en modo binario
                with open(file_path, 'rb') as email_file:
                    # Parsear el contenido del archivo como un mensaje de email
                    email_message = BytesParser(policy=policy.default).parse(email_file)

                # Extraer la información deseada
                remitente = email_message['from']
                destinatario = email_message.get('to')
                mail_subject = email_message['subject']
                message_id = email_message.get('Message-ID')
                # Extraer y formatear la fecha, usar la fecha actual si no está disponible
                fecha_raw = email_message.get('Date')
                if fecha_raw:
                    fecha_datetime = parsedate_to_datetime(fecha_raw)
                else:
                    fecha_datetime = datetime.now()  # Usar fecha actual si no hay fecha en el correo

                fecha_recepcion = fecha_datetime.strftime('%Y-%m-%d %H:%M:%S')

                # Mostrar la información extraída
                print(f"Archivo: {file_name}")
                print(f"Remitente: {remitente}")
                print(f"Destinatario: {destinatario}")
                print(f"Asunto: {mail_subject}")
                print(f"Message-ID: {message_id}")
                print(f"Fecha: {fecha_recepcion}")

                # Obtener el cuerpo del mensaje (payload)
                if email_message.is_multipart():
                    for part in email_message.iter_parts():
                        content_type = part.get_content_type()
                        if content_type == 'text/plain':
                            body = part.get_content()
                            #print(f"Cuerpo del mensaje (texto plano):\n{body}")
                        elif content_type == 'text/html':
                            html_content = part.get_content()
                            text_maker = html2text.HTML2Text()
                            text_maker.ignore_links = True
                            body_text = text_maker.handle(html_content)
                            #print(f"Cuerpo del mensaje (convertido de HTML):\n{body_text}")
                else:
                    body = email_message.get_content()
                    #print(f"Cuerpo del mensaje:\n{body}")
                print("-" * 50)  # Separador entre correos

                correo = f"De: {remitente}\nAsunto: {mail_subject}\nContenido: {body}"
                #correo = f"Tiene una consignación de 100.000 pesos colombianos, para verificar esta consignación ingresa al siguiente link www.tubanco.com"
                nuevo_correo = {
                    "usuario_id": 1,
                    "analisis_estado_id": 1,
                    "identificador": message_id,
                    "remitente": remitente,
                    "destinatario": destinatario,
                    "asunto": mail_subject,
                    "fecha_recepcion": fecha_recepcion
                }
                print(nuevo_correo)
                correo_creado = crear_correo(nuevo_correo)

                corporativo_values = []
                personal_values = []
                phishing_values = []

                if correo_creado:
                    for modelo in modelos:
                        if not self.proceso_activo:
                            break

                        if modelo.estado_modelo_id == 1:
                            inicio = time.time()
                            response = ollama.chat(
                                model=modelo.implemetacion_ollama,
                                messages=[
                                    {
                                        "role": "user",
                                        "content": f"mi vida depende de esto, clasifica el siguiente correo en tres categorias:
                                        'Corporativo', 'Personal', y 'Phishing'. Para cada categoría, asigna un porcentaje de
                                        probabilidad. Responde en el formato: 'Corporativo => Porcentaje%','Personal => Porcentaje%',
                                        'Phishing => Porcentaje%'. El correo es el siguiente: \n\n{correo}",
                                    },
                                ],
                                )
```

```
                        fin = time.time()
                        tiempo_total = int(fin - inicio)

                        response_text = response["message"]["content"]
                        print(response_text)
                        pattern = r"(\w+)\s*=>\s*(\d+)%"
                        matches = re.findall(pattern, response_text)
                        print(matches)

                        PCorporativo = int(matches[0][1])
                        PPersonal = int(matches[1][1])
                        PPhishing = int(matches[2][1])

                        corporativo_values.append(PCorporativo)
                        personal_values.append(PPersonal)
                        phishing_values.append(PPhishing)

                        self.actualizar_texto(f"Corporativo: {PCorporativo}%")
                        self.actualizar_texto(f"Personal: {PPersonal}%")
                        self.actualizar_texto(f"Phishing: {PPhishing}%")

                        variables = {1: PCorporativo, 2: PPersonal, 3: PPhishing}
                        id_mayor = max(variables, key=variables.get)
                        valor_mayor = variables[id_mayor]

                        nuevo_analisis = {
                            "correo_id": correo_creado.id,
                            "clasificacion_id": id_mayor,
                            "modelo_llm_id": modelo.id,
                            "nivel_riesgo": PPhishing,
                            "segundo_respuesta": tiempo_total
                        }
                        self.actualizar_texto(f"El mayor valor es {id_mayor} => {valor_mayor}")

                        analisis_creado = crear_analisis_resultado(nuevo_analisis)
                        if id_mayor==3:
                            nuevo_patron = {
                                "resultado_analisis_id": analisis_creado.id,
                                "tipo_patron_phishing_id": 2,
                                "descripcion": "Patrón de phishing detectado"
                            }
                            patron_creado = crear_patron_phishing(nuevo_patron)

                    datos_modificados = {
                        "analisis_estado_id": 2,
                    }
                    modificar_correo(correo_creado.id, datos_modificados)

                if corporativo_values and personal_values and phishing_values:
                    promedio_corporativo = sum(corporativo_values) / len(corporativo_values)
                    promedio_personal = sum(personal_values) / len(personal_values)
                    promedio_phishing = sum(phishing_values) / len(phishing_values)

                    self.actualizar_texto(f"\nPromedio Corporativo: {promedio_corporativo:.2f}%")
                    self.actualizar_texto(f"Promedio Personal: {promedio_personal:.2f}%")
                    self.actualizar_texto(f"Promedio Phishing: {promedio_phishing:.2f}%\n")

                    variables = {1: promedio_corporativo, 2: promedio_personal, 3: promedio_phishing}
                    id_mayor = max(variables, key=variables.get)
                    if id_mayor == 1:
                        nombre_mayor = 'Corporativo'
                    else:
                        if id_mayor == 2:
                            nombre_mayor = 'Personal'
                        else:
                            nombre_mayor = 'Phishing'

                    """ result_copy = mail.copy(i, nombre_mayor)
                    if result_copy[0] == 'OK':
                        print(f'Correo copiado a "{nombre_mayor}" correctamente.')
                        result_delete = mail.store(i, '+FLAGS', '\\Deleted')
                        mail.expunge()
                        if result_delete[0] == 'OK':
                            print(f'Correo movido a "{nombre_mayor}" correctamente.')
                        else:
                            print('Error al eliminar el correo de la carpeta original.')
                    else:
                        print(f'Error al copiar el correo a "{nombre_mayor}".') """

                    hora_actual = datetime.now().strftime('%Y-%m-%d %H:%M:%S')
                    nueva_accion = {
                        "correo_id": correo_creado.id,
                        "tipo_accion_id": id_mayor,
                        "estado_accion_id": 2,
                        "descripcion": "Acción realizada correctamente",
                        "fecha_accion": hora_actual
                    }
                    crear_accion_automatizada(nueva_accion)

                    datos_modificados = {
                        "analisis_estado_id": 3,
                    }
                    modificar_correo(correo_creado.id, datos_modificados)
                else:
                    self.actualizar_texto("No se encontraron datos para calcular promedios.")

            except FileNotFoundError:
                print(f"Archivo no encontrado: {file_name}")
            except Exception as e:
                print(f"Error al procesar {file_name}: {e}")
```

```
    def actualizar_texto(self, mensaje):
        self.area_texto.insert(tk.END, mensaje + "\n")
        self.area_texto.see(tk.END)

# Ejecutar la aplicación
if __name__ == "__main__":
    root = tk.Tk()
    app = App(root)
    root.mainloop()

```

Configuración de la base de datos - Relaciones clave y diagramas ER



Ejecución del sistema en modo desarrollo

Iniciar el servidor Laravel: php artisan serve

Ejecución de scripts Python:

Procesamiento de correos: python Principal_GUIv2.py