GRADO EN INGENIERÍA EN TECNOLOGÍAS DE TELECOMUNICACIÓN

TRABAJO FIN DE GRADO

Desarrollo de una Aplicación Web para la Detección de Mensajes de Phishing

Autor: María Begara Girón

Director: David Martín-Corral Calvo

Madrid

Declaración de originalidad

Declaro bajo mi responsabilidad que el Proyecto presentado con el título **Desarrollo de una Aplicación Web para la Detección de Mensajes de Phishing** de la ETS de Ingeniería – ICAI de la Universidad Pontificia Comillas en el curso académico 2025-2026 es de mi autoría y no ha sido presentado con anterioridad a otros efectos. El Proyecto no es plagio de otro, ni total ni parcialmente y la información que ha sido tomada de otros documentos está debidamente referenciada.

Uso de Inteligencia Artificial[1]

Declaro bajo mi responsabilidad que (indicar la opción correcta):

☐ No he utilizado Inteligencia Artificial en la elaboración del presente documento.

☒ He utilizado Inteligencia Artificial en la elaboración del presente documento y/o del Anexo B siempre en las condiciones permitidas por la Universidad Pontificia Comillas, es decir, aplicando el Nivel 2 de la Escala de Evaluación de Perkins et al. (2024): "*La IA puede utilizarse para actividades previas a la tarea, como la lluvia de ideas, la descripción y la investigación inicial. Este nivel se centra en el uso de la IA para la planificación, las síntesis y la generación de ideas, pero las evaluaciones deben hacer hincapié en la capacidad de desarrollar y refinar estas ideas de forma independiente*". En concreto, las Inteligencia Artificial ha sido empleada para:

Se ha utilizado la Inteligencia Artificial para definir la estructura de la memoria del proyecto, así como para digitalizar los esquemas.
Firmado (alumno): 
Fecha: 25/05/2026

[1] Esta declaración se refiere al uso de la Inteligencia Artificial generativa para realizar los documentos del Proyecto (Anexo B y Memoria). No aplica a Proyectos donde, por su naturaleza, deban emplear inteligencia artificial como parte de los mismos (aplicación de técnicas de aprendizaje automático, redes neuronales, análisis de datos…)

Autorización para la entrega del Proyecto

El Director del Proyecto
Fdo: MARTIN CORRAL CALVO DAVID - 03909768K Digitally signed by MARTIN CORRAL CALVO DAVID - 03909768K Date: 2026.05.28 10:12:40 +02'00'
Fecha:

GRADO EN INGENIERÍA EN TECNOLOGÍAS DE TELECOMUNICACIÓN

TRABAJO FIN DE GRADO

Desarrollo de una Aplicación Web para la Detección de Mensajes de Phishing

Autor: María Begara Girón

Director: David Martín-Corral Calvo

Madrid

Agradecimientos

En primer lugar, me gustaría agradecer a mi tutor David por acompañarme a lo largo de este proceso. Gracias a su ayuda para definir el proyecto y su alcance, a su disponibilidad para resolver dudas y a la confianza que ha depositado en mí, he podido desarrollar este trabajo en un área apasionante y que me interesa particularmente.

En segundo lugar, me gustaría agradecer a todos los profesores que me han proporcionado todos los conocimientos necesarios para poder plantear y realizar este proyecto. Este trabajo me ha permitido unir todos los conocimientos aprendidos durante estos años.

Por último, me gustaría agradecer a mi familia su apoyo incondicional a lo largo de toda mi etapa universitaria. Sin ellos no habría llegado hasta aquí.

DESARROLLO DE UNA APLICACIÓN WEB PARA LA DETECCIÓN DE MENSAJES DE PHISHING

Autor: Begara Girón, María.
Director: Martín-Corral Calvo, David.
Entidad Colaboradora: ICAI – Universidad Pontificia Comillas

RESUMEN DEL PROYECTO

El *phishing* representa una de las principales amenazas digitales en todo el mundo. Este proyecto presenta una aplicación web con un modelo híbrido que combina Machine Learning (Regresión Logística sobre TF-IDF) y un modelo de lenguaje generativo (LLaMA 3.3-70B) para analizar correos electrónicos sospechosos y proporcionar recomendaciones personalizadas. El sistema alcanza un F1 del 99,31% sobre el *dataset* de test y ha sido validado con 88 correos reales recopilados durante siete meses.

Palabras clave: phishing, machine learning, large language model, detección de fraude

1. Introducción

En 2024, el INCIBE gestionó 97.348 incidentes de ciberseguridad en España, de los cuales 21.571 fueron de *phishing*, siendo esta una de las amenazas más frecuentes y con mayor impacto económico [1]. Detrás de cada uno de estos incidentes hay una persona que recibió un mensaje aparentemente legítimo y no supo identificarlo como fraudulento.

Existen soluciones en diversos formatos y plataformas cuyo objetivo es revertir esta situación, pero la fragmentación en canales específicos, la falta de explicaciones y orientación al usuario provocan que no sean óptimas. Además, el auge de la Inteligencia Artificial ha incrementado notablemente la sofisticación de las campañas de *phishing*, personalizando los mensajes de manera incremental y dificultando su detección.

En el presente proyecto se presenta la herramienta PhishGuard que pretende dar respuesta a estas limitaciones combinando un modelo de Machine Learning (ML) para la detección eficiente del *phishing* con una capa de inteligencia artificial generativa que transforma el resultado técnico en orientación personalizada y comprensible para cualquier usuario.

2. Definición del Proyecto

PhishGuard tiene como objetivo proporcionar al usuario una estimación de la probabilidad de que un mensaje sea fraudulento junto con recomendaciones personalizadas, sin requerir conocimientos técnicos previos. La herramienta es de acceso libre, gratuita y desplegada íntegramente en la nube.

Se ha implementado una arquitectura de microservicios, creando un sistema modular y escalable. Durante el desarrollo se ha seguido una metodología ágil iterativa, trabajando en paralelo sobre las distintas capas del sistema.

3. Descripción del Sistema

El modelo de detección se basa en un clasificador de Regresión Logística sobre vectorización TF-IDF, utilizando un corpus combinado de tres *datasets* públicos, con un total de 50.324 correos, destinando un 80% al entrenamiento y un 20% a la validación. El umbral de clasificación se ajustó a 0,44 mediante el análisis de la curva *Precision-Recall*. Los artefactos del modelo se almacenan en Hugging Face Hub para su descarga automática al arrancar.

Las recomendaciones personalizadas se implementan mediante el modelo LLaMA 3.3-70B, añadiendo un *prompt* dinámico en el que se incluye la probabilidad estimada por el ML y si existe urgencia en el mensaje. De forma opcional se añade contexto de filtraciones recientes del dominio remitente mediante Tavily Search API.

La arquitectura del sistema se organiza en tres microservicios independientes: el *frontend* desarrollado en React/Vite y desplegado en Vercel, el *backend* en SpringBoot desplegado en Railway que actúa como orquestador central, y el servicio de inferencia en FastAPI, también desplegado en Railway.

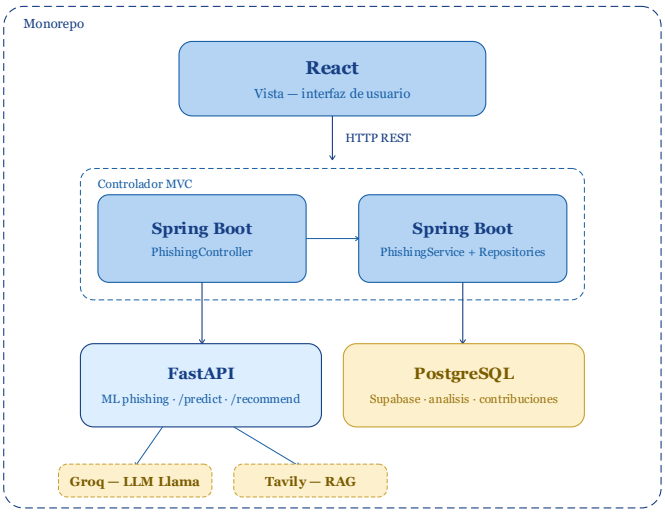

Figura 1. Esquema de la arquitectura del sistema.

La interfaz de usuario ofrece cuatro secciones: inicio, análisis de mensajes, estadísticas globales y recomendaciones generales de buenas prácticas y con contenido legal, basada en el Reglamento General de Protección de Datos (RGPD) [2] y la Directiva PSD2 [3]. La aplicación cuenta con diseño para escritorio y para móvil, así como un tour interactivo que guía al usuario por las funcionalidades principales.

4. Resultados

El modelo de Regresión Logística alcanzó una *accuracy* del 99,27% sobre el *dataset* de test, un *recall* del 99,40% y un *F1-score* del 99,31%, superando los resultados recogidos en la mayor parte de la literatura consultada para arquitecturas similares. La comparativa con tres modelos alternativos (Random Forest, SVM Lineal y Naive Bayes Multinomial) confirmó que la Regresión Logística es la opción más adecuada para este caso de uso, al ser el único clasificador que combina rendimiento con generación nativa de probabilidades calibradas e interpretabilidad de coeficientes por palabra.

Modelo	Accuracy	Precision (phishing)	Recall (phishing)	F1 (phishing)	T. entrenamiento	T. inferencia
Regresión Logística producción	99,27%	99,23%	99,40%	99,31%	4,25s	0,004s
SVM Lineal	99,69%	99,74%	99,68%	99,71%	0,39s	0,003s
Random Forest	98,96%	99,52%	98,49%	99,01%	45,35s	0,689s
Naive Bayes	98,66%	99,54%	97,91%	98,72%	0,03s	0,005s

Tabla 1. Comparativa de modelos

La validación con el *dataset* propio de 88 correos reales recopilados entre octubre de 2025 y mayo de 2026 reveló que el 92,2% de los correos de phishing fueron correctamente alertados como riesgo medio o alto. El análisis técnico de las cabeceras identificó cuatro niveles de sofisticación, siendo el más avanzado el abuso de plataformas legítimas de envío masivo, presente en el 33% del subconjunto de correos directos, y confirmando que el 38% de los correos de phishing superaron los filtros automáticos de los proveedores de correo sin ser detectados.

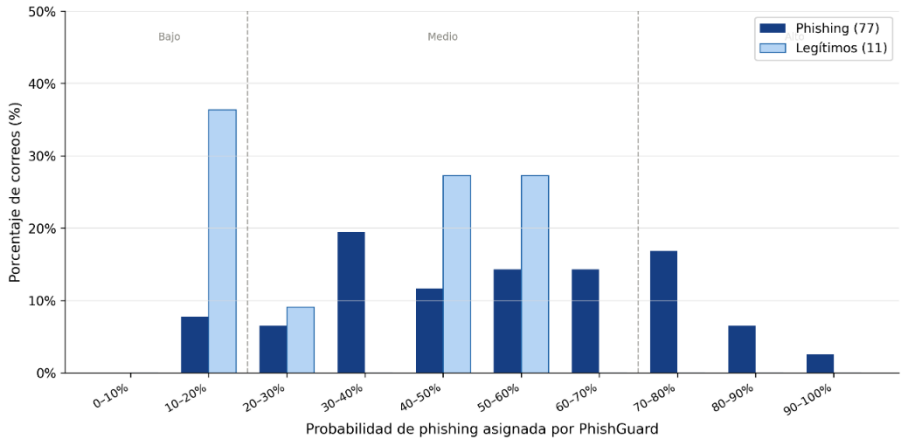

Figura 2. Histograma de probabilidades.

El análisis cualitativo de las recomendaciones generadas por el LLM demostró que el sistema adapta el tono y el contenido en función del nivel de riesgo, la urgencia detectada y el contexto del remitente. Tavily identificó filtraciones recientes en todos los correos con dominio corporativo real, enriqueciendo la recomendación con información verificable y referencias a fuentes externas.

5. Conclusiones

La herramienta demuestra que la combinación de un clasificador clásico de Machine Learning con una capa de inteligencia artificial generativa permite ofrecer al usuario no solo una estimación del riesgo sino una orientación personalizada y accionable, cubriendo un vacío que las soluciones existentes no abordan. El sistema alcanza métricas de rendimiento excelentes sobre datos de entrenamiento y mantiene una tasa de detección del 92,2% sobre correos reales actuales, con las limitaciones inherentes al idioma y a la desactualización de los corpus públicos disponibles.

Las líneas de trabajo futuro priorizan el reentrenamiento con contribuciones de la comunidad, el soporte multilingüe y la mejora del módulo de búsqueda en tiempo real, con el objetivo de convertir PhishGuard en una herramienta de referencia para la concienciación en ciberseguridad ciudadana. En un entorno en el que los ataques son cada

vez más sofisticados y dirigidos, la respuesta más efectiva no es solo técnica sino también educativa, y ese es el vacío que este proyecto pretende resolver.

6. Referencias

[1] INCIBE (2025). *Balance de ciberseguridad 2024*. Instituto Nacional de Ciberseguridad. Disponible en: https://www.incibe.es/incibe/sala-de-prensa/incibe-presenta-su-balance-de-ciberseguridad-2024-con-mas-de-97000-incidentes

[2] Parlamento Europeo y Consejo de la UE, "Reglamento (UE) 2016/679 relativo a la protección de las personas físicas en lo que respecta al tratamiento de datos personales (RGPD)," *Diario Oficial de la Unión Europea*, L 119, abr. 2016. [Online]. Disponible en: https://eur-lex.europa.eu/legal-content/ES/TXT/?uri=CELEX%3A32016R0679

[3] Jefatura del Estado, "Real Decreto-ley 19/2018, de 23 de noviembre, de servicios de pago y otras medidas urgentes en materia financiera," *BOE* núm. 284, 2018. [Online]. Disponible en: https://www.boe.es/buscar/doc.php?id=BOE-A-2018-16036

PHISHING DETECTION WEB APPLICATION

Author: Begara Girón, María.
Director: Martín-Corral Calvo, David.
Collaborating entity: ICAI – Universidad Pontificia Comillas

ABSTRACT

Phishing represents one of the main digital threats worldwide. This project presents a web application with a hybrid model combining Machine Learning (Logistic Regression over TF-IDF) and a generative language model (LLaMA 3.3-70B) to analyze suspicious emails and provide personalized recommendations. The system achieves an F1-score of 99.31% on the test dataset and has been validated with 88 real emails collected over seven months.

Key words: phishing, machine learning, large language model, fraud detection

1. Introduction

INCIBE managed 97,348 cybersecurity incidents in Spain, of which 21,571 were phishing-related, making it one of the most frequent threats with the greatest economic impact on citizens and businesses [1]. Behind each of these incidents is a person who received an apparently legitimate message and was unable to identify it as fraudulent.

Existing solutions in various formats and platforms aim to address this situation, but their fragmentation across specific channels, lack of explanations and insufficient user guidance make them deficient. Furthermore, the rise of Artificial Intelligence has significantly increased the sophistication of phishing campaigns, progressively personalizing messages and making their detection increasingly difficult.

This project presents PhishGuard, a tool designed to address these limitations by combining a Machine Learning (ML) model for efficient phishing detection with a generative AI layer that transforms technical results into personalized, comprehensible guidance for any user.

2. Project Definition

PhishGuard aims to provide users with an estimate of the probability that a message is fraudulent, along with personalized recommendations, without requiring any prior technical knowledge. The tool is freely accessible, free of charge, and fully deployed in the cloud.

A microservices architecture has been implemented, creating a modular and scalable system. Development followed an agile iterative methodology, working in parallel across the different layers of the system.

3. Description of the System

The detection model is based on a Logistic Regression classifier over TF-IDF vectorization, using a combined corpus of three public datasets summing 50,324 emails, with an 80/20 train-test split. The classification threshold was adjusted to 0.44 through Precision-Recall curve analysis. Model artefacts are stored in Hugging Face Hub and downloaded automatically on service startup.

Personalized recommendations are generated by the LLaMA 3.3-70B model through a dynamic prompt that incorporates the ML-estimated probability and whether urgency is detected in the message. Optionally, context on recent data breaches associated with the sender's domain is retrieved in real time via the Tavily Search API.

The system architecture is organized into three independent microservices: the frontend built in React/Vite and deployed on Vercel, the Spring Boot backend deployed on Railway acting as central orchestrator, and the FastAPI inference service, also deployed on Railway.

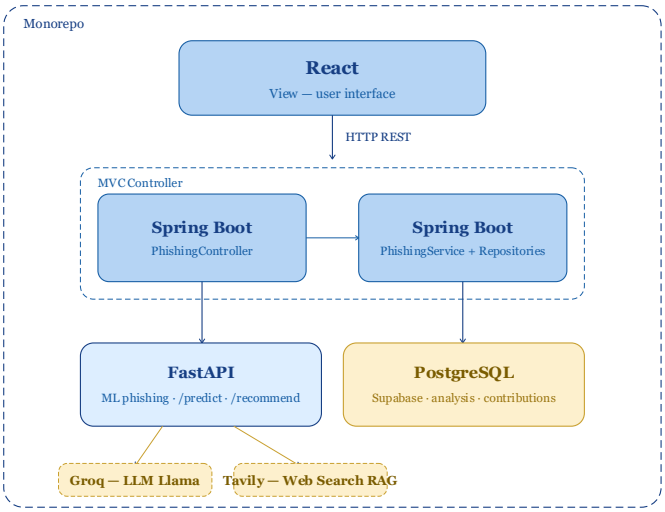

Figure 1. System architecture diagram.

The user interface offers four sections: home, message analysis, global statistics, and general best practice recommendations with legal guidance based on the GDPR and the PSD2 Directive — two key European regulations governing personal data protection and electronic payment security respectively. The application features a responsive design

for both desktop and mobile, as well as an interactive tour guiding users through the main functionalities.

4. Results

The Logistic Regression model achieved an accuracy of 99.27%, a recall of 99.40% and an F1-score of 99.31% on the test dataset, outperforming most results reported in the reviewed literature for similar architectures. A comparison with three alternative models — Random Forest, Linear SVM and Multinomial Naive Bayes — confirmed that Logistic Regression is the most suitable choice for this use case, as it is the only classifier that combines strong performance with native calibrated probability output and per-word coefficient interpretability.

Model	Accuracy	Precision (phishing)	Recall (phishing)	F1 (phishing)	Training time	Inference time
Logistic Regression production	99.27%	99.23%	99.40%	99.31%	4.25s	0.004s
Linear SVM	**99.69%**	**99.74%**	**99.68%**	**99.71%**	0.39s	0.003s
Random Forest	98.96%	99.52%	98.49%	99.01%	45.35s	0.689s
Naïve Bayes	98.66%	99.54%	97.91%	98.72%	0.03s	0.005s

Table 1. Model comparison.

Validation using a proprietary dataset of 88 real emails collected between October 2025 and May 2026 showed that 92.2% of phishing emails were correctly flagged as medium or high risk. Technical header analysis identified four sophistication levels, the most advanced being the abuse of legitimate bulk email platforms, present in 33% of the direct email subset, and confirming that 38% of phishing emails bypassed providers' automatic spam filters undetected.

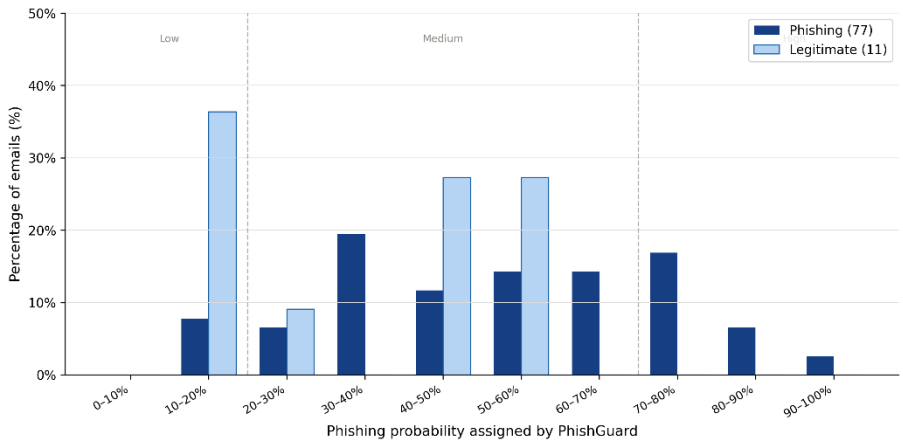

Figure 2. Probability histogram.

Qualitative analysis of the LLM-generated recommendations demonstrated that the system adapts its tone and content according to the risk level, detected urgency and sender context. Tavily identified recent data breaches for all emails with real corporate domains, enriching the recommendation with verifiable information and references to external sources.

5. Conclusions

PhishGuard demonstrates that combining a classical Machine Learning classifier with a generative AI layer makes it possible to offer users not only a risk estimate but also personalized, actionable guidance — addressing a gap that existing solutions do not cover. The system achieves excellent performance metrics on training data and maintains a 92.2% detection rate on current real-world emails, with the inherent limitations of language dependency and the outdated nature of publicly available corpora.

Future work priorities include retraining with community contributions, multilingual support and improvement of the real-time search module, with the ultimate goal of establishing PhishGuard as a reference tool for cybersecurity awareness among citizens. In an environment where attacks are becoming increasingly sophisticated and targeted, the most effective response is not only technical but also educational — and that is the gap this project aims to fill.

6. References

[1] INCIBE (2025). *Cybersecurity Balance 2024*. Spanish National Cybersecurity Institute. Available at: https://www.incibe.es/incibe/sala-de-prensa/incibe-presenta-su-balance-de-ciberseguridad-2024-con-mas-de-97000-incidentes

[2] European Parliament and Council of the EU, "Regulation (EU) 2016/679 on the protection of natural persons with regard to the processing of personal data (GDPR)," *Official Journal of the European Union*, L 119, Apr. 2016. [Online]. Available at: https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX%3A32016R0679

[3] Jefatura del Estado, "Real Decreto-ley 19/2018, de 23 de noviembre, de servicios de pago y otras medidas urgentes en materia financiera" [Royal Decree-Law 19/2018, of November 23, on payment services and other urgent financial measures], *BOE* no. 284, 2018. [Online]. Available at: https://www.boe.es/buscar/doc.php?id=BOE-A-2018-16036

Índice de la memoria

Índice de Figuras

Índice de Tablas

Capítulo1. Introducción

En la última década, el desarrollo de la tecnología ha traído innumerables ventajas en diferentes servicios digitales como la banca, la Administración pública, el comercio electrónico o las comunicaciones. Sin embargo, estas mejoras también han propiciado la multiplicación de los puntos de entrada a los diferentes sistemas, aumentado el riesgo de ataques, tanto masivos como personalizados. Entre las amenazas más extendidas se encuentra el *phishing*: una técnica de fraude que imita mensajes legítimos, suplantando la identidad de entidades oficiales con el objetivo de obtener credenciales, datos personales o perpetrar diferentes fraudes financieros.

El phishing no se limita al correo electrónico, sino que han aparecido variantes que se han extendido prácticamente a todos los canales de comunicación digital: el *smishing* se envía a través de SMS y aplicaciones de mensajería instantánea; el *vishing* es mediante llamadas telefónicas, pudiendo llegar a imitar la voz de una entidad de confianza mediante sistemas de inteligencia artificial, como el ataque al CEO; y el *spear phishing* es una variante personalizada del *phishing*, diseñada específicamente para un individuo u organización concreta, incorporando en muchos casos datos reales de la víctima para aumentar su credibilidad. Esta diversificación de canales es uno de los motivos por los que las soluciones de detección actuales, centradas en un único canal, resultan insuficientes.

El *phishing* es una forma de ingeniería social: una categoría de ataques que, en lugar de explotar vulnerabilidades técnicas de los sistemas, manipula el comportamiento humano mediante el engaño. Lo que distingue al *phishing* de otros tipos de ciberataque es que su objetivo no es el sistema, sino la persona, tratando de explotar vulnerabilidades psicológicas en lugar de técnicas: la urgencia, el miedo, la confianza en una marca conocida o la voluntad de ayudar. Esta naturaleza hace que ningún parche técnico pueda eliminar la amenaza por completo, y que la detección automática, por muy precisa que sea, deba ir acompañada de orientación al usuario para ser verdaderamente efectiva. Un sistema que detecta un mensaje como *phishing,* pero no incluye una explicación u orientación sobre qué hacer a continuación traslada al usuario la responsabilidad de interpretar y actuar, precisamente en el momento en que es más vulnerable.

Esta tipología de estafa ha evolucionado con el tiempo. Muchos de los ataques actuales no son únicamente mensajes con errores evidentes que permiten una detección sencilla. Con el boom de la inteligencia artificial se ha acelerado esta mejoría en la creación de textos persuasivos con imágenes, logos y formatos prácticamente idénticos a

los originales, haciéndolos especialmente difíciles de detectar incluso para usuarios con experiencia.

El incremento de estos fraudes ha motivado una respuesta regulatoria en la Unión Europea, por ejemplo, a través de la Directiva de Servicios de Pago PSD2 y sus guías de reporte de fraude, que refuerzan la protección del usuario frente a operaciones no autorizadas, y el Reglamento General de Protección de Datos (RGPD), que establece los derechos del ciudadano frente al uso indebido de su información personal. Sin embargo, muchos ciudadanos siguen sin saber cómo actuar cuando sospechan haber sido víctimas de phishing, ni qué pasos pueden dar para minimizar el daño.

En este contexto se enmarca el presente trabajo, que propone una aplicación web capaz de analizar el contenido de mensajes sospechosos, estimar la probabilidad de que se trate de *phishing*, mostrar los indicadores que han influido en la decisión del modelo en mayor medida, y ofrecer pautas de actuación y orientación legal básica adaptada al marco normativo vigente en España y la Unión Europea.

1.1. Motivación

En los últimos años, el aumento de fraudes digitales ha puesto de manifiesto dos problemas principales. Por un lado, la dificultad de muchos usuarios para reconocer mensajes de *phishing* cada vez más sofisticados, que imitan con gran precisión el estilo, diseño y contenidos de las comunicaciones legítimas. Por otro lado, la dispersión de las herramientas existentes, que suelen estar integradas en servicios concretos (banca, correo electrónico, mensajería) y no ofrecen un punto de acceso unificado ni explicaciones sobre el riesgo detectado.

Estos dos problemas no son independientes: la fragmentación de las soluciones obliga al usuario a gestionar múltiples herramientas en diferentes canales, precisamente en momentos en los que los atacantes generan urgencia para limitar la capacidad de respuesta de la víctima. El resultado es que la detección técnica, por sí sola, no es suficiente. El usuario necesita entender por qué un mensaje es sospechoso y qué debe hacer a continuación, y esa orientación no está disponible de forma centralizada ni accesible para el usuario general.

Este proyecto nace de la convicción de que el bien intrínseco de la ingeniería no se limita a desarrollar sistemas funcionales, sino a desarrollar sistemas que resuelvan problemas reales y aporten valor a las personas. En el caso de la ciberseguridad se traduce en que el usuario, especialmente aquel con menos conocimientos tecnológicos, es la parte más expuesta, por lo que debe ser una parte fundamental del diseño. No se puede presuponer que el usuario va a detectar amenazas por sí mismo ni que va a actuar siempre

de forma segura: la responsabilidad de que el sistema funcione no puede recaer únicamente en él.

Partiendo de estos pilares, la motivación de este proyecto es doble. La primera es técnica: desarrollar un sistema híbrido que combine la eficiencia de un clasificador de aprendizaje automático con la capacidad explicativa de un modelo de lenguaje de gran tamaño, permitiendo presentar una solución que combina ambos enfoques para cubrir las limitaciones que cada una presenta por separado. La segunda es social: ofrecer una herramienta de forma accesible y gratuita, acompañada de orientación legal y recomendaciones personalizadas, contribuyendo a construir una sociedad más resiliente y consciente ante este tipo de ataques. La concienciación no sustituye a la seguridad por diseño, pero sí la complementa: transmitir conocimiento sobre los riesgos es también una responsabilidad del ingeniero, ya que contribuye a largo plazo a reducir la vulnerabilidad del usuario frente a la ingeniería social.

1.2. Objetivos generales

El objetivo principal de este proyecto es desarrollar una herramienta web unificada que permita analizar mensajes sospechosos de cualquier origen y asistir al usuario en su decisión, combinando detección automática mediante aprendizaje automático con recomendaciones personalizadas generadas por inteligencia artificial generativa.

Para alcanzar este objetivo, se han definido una serie de objetivos generales y específicos que definen el alcance del proyecto: la accesibilidad y simplicidad de la herramienta, la automatización y transparencia, la centralización de la información en un único punto de acceso, la actualización continua del modelo mediante contribuciones voluntarias de los usuarios y la privacidad por diseño. Estos objetivos se describen en detalle, junto con los objetivos técnicos específicos, en la sección 4.2.

1.3. Estructura del documento

El presente documento se organiza en ocho secciones. En la primera sección se introduce el contexto, la motivación y los objetivos del proyecto. En la segunda, se describen las herramientas y tecnologías utilizadas para el desarrollo del sistema. En el capítulo 3 se revisa el estado del arte del *phishing* y las soluciones existentes, mientras que en la cuarta se define el trabajo junto a la justificación técnica y de mercado, los objetivos generales y específicos, la metodología y el marco normativo en el que se enmarca el trabajo. El capítulo 5 describe en detalle el sistema desarrollado, desde la

arquitectura global hasta cada uno de los componentes: el módulo de aprendizaje automático (*machine learning*), el servicio de inferencia, el *backend*, la base de datos, el *frontend* y el despliegue en la nube. En el capítulo 6 se presenta un análisis de los resultados, incluyendo la evaluación del modelo, el análisis de caso de uso reales con mensajes propios y el análisis cualitativo de las recomendaciones generadas por el LLM. En el capítulo 7 se recogen las conclusiones del proyecto, así como las líneas de trabajo futuras. El capítulo 8 contiene la bibliografía completa del trabajo. En último lugar, se incluyen cuatro anexos, en los que se recogen los Objetivos de Desarrollo Sostenible (ODS), fragmentos relevantes de código y el enlace al repositorio del proyecto.

Capítulo 2. Descripción de las Tecnologías

En esta sección se describe qué es cada tecnología, para qué se usa en el desarrollo del proyecto y por qué se ha elegido.

El control de versiones del proyecto se ha gestionado mediante Git, con un único repositorio en GitHub para los tres servicios: *frontend*, *backend* y servicio ML, facilitando el desarrollo paralelo y el seguimiento de cambios de forma centralizada.

2.1. Lenguajes y frameworks

El sistema desarrollado en este proyecto está compuesto por tres capas diferenciadas: servicio de inferencia, *backend* y *frontend*. Esta separación garantiza que cada componente pueda actualizarse o modificarse de forma independiente sin afectar al resto del sistema.

Python es el lenguaje seleccionado para el servicio de inferencia por ser el ecosistema natural del aprendizaje automático: las principales bibliotecas de entrenamiento (*scikit-learn*), vectorización e inferencia de modelos, en concreto la versión 3.12, ya que las versiones más recientes (3.14) no soportan todavía algunas de ellas. Además, se ha usado el *framework* FastAPI para exponer el modelo como una API REST, ya que es asíncrono, genera documentación interactiva que facilita el desarrollo y la depuración de los *endpoints*; y es el estándar actual para el despliegue de microservicios de ML en Python.

Java es el lenguaje seleccionado para el *backend* principal del sistema. En concreto se ha utilizado por su compatibilidad con Spring Boot y JPA/Hibernate. Spring Boot es el *framework* que facilita el desarrollo de APIs REST con arquitectura Modelo-Vista-Controlador (MVC) sobre Java, gestionando de forma automática la inyección de dependencias, la configuración del servidor y la conexión con la base de datos mediante JPA/Hibernate. En el contexto de este proyecto, Spring Boot actúa como orquestador central: recibe las peticiones del *frontend*, gestiona las llamadas al microservicio FastAPI y la persistencia en PostgreSQL, como se describe con más detalle en la sección 5.4.

React es la librería de JavaScript seleccionada para el desarrollo del *frontend* por su modelo basado en componentes reutilizables, que permite construir interfaces reactivas en las que el estado de cada componente determina lo que se muestra al usuario en cada momento, sin necesidad de recargar la página completa. Esto es especialmente relevante en este proyecto, donde el resultado del análisis se muestra en la misma pantalla que el

formulario de entrada sin necesidad de navegación adicional. Vite actúa como *bundler* y servidor de desarrollo, empaquetando todos los ficheros del proyecto en un conjunto optimizado para producción y proporcionando tiempos de compilación muy reducidos durante el desarrollo local.

2.2. Machine Learning

El módulo de aprendizaje automático del sistema está desarrollado íntegramente con *scikit-learn*, la biblioteca de referencia para aprendizaje automático en Python. Se ha elegido teniendo en cuenta la necesidad del modelo entrenado, descartando opciones como TensorFlow o PyTorch, ya que scikit-learn proporciona las herramientas necesarias con una complejidad computacional y de despliegue menores.

TF-IDF (Term Frequency-Inverse Document Frequency) es la técnica utilizada para transformar el texto de los mensajes en una representación numérica que el clasificador pueda procesar. Su principal ventaja para este proyecto es su rendimiento y la interpretabilidad: los pesos asignados a cada término permiten identificar y mostrar al usuario las palabras que más han influido en la decisión del modelo. Se explica más en detalle en la sección 5.2.

Se ha seleccionado el algoritmo de clasificación Regresión Logística porque, al contrario que otros clasificadores, estima una probabilidad de pertenencia a cada clase mediante la función sigmoide, lo que permite mostrar al usuario un porcentaje de riesgo en lugar de un veredicto binario. Sus coeficientes son además directamente interpretables, lo que facilita la generación de explicaciones comprensibles. En la sección 6.2 se hace un comparativa del rendimiento de este modelo con respecto a otros.

2.3. Inteligencia Artificial Generativa

Los modelos de lenguaje de gran tamaño (Large Language Models, LLM) son modelos de inteligencia artificial entrenados sobre corpus de texto con el objetivo de comprender y generar lenguaje natural. De esta forma, los LLM son capaces de generar texto contextualizado, razonar sobre el contenido semántico de un mensaje y producir respuestas adaptadas a una situación concreta.

Groq es la plataforma de inferencia utilizada para ejecutar el LLM. A diferencia de las plataformas de inferencia convencionales basadas en GPU, Groq utiliza hardware especializado denominado LPU (Language Processing Unit), diseñado específicamente para la inferencia de modelos de lenguaje. Esto se traduce en velocidades de generación

de texto significativamente superiores a las de plataformas alternativas, lo que es especialmente relevante en este proyecto dado que la respuesta del LLM forma parte del flujo principal de análisis y el usuario espera el resultado en tiempo real.

LLaMA-3.3-70B es el modelo seleccionado dentro de la plataforma Groq. Su elección frente a modelos propietarios como GPT-4 responde a tres motivos: el acceso gratuito a través de Groq, su rendimiento documentado en tareas de análisis y generación de texto en español e inglés, y la alineación con el principio de transparencia, al ser un modelo de código abierto.

La combinación de Groq como plataforma y LLaMA-3.3-70B como modelo permite obtener recomendaciones personalizadas de alta calidad con una latencia reducida y sin coste adicional para el usuario.

2.4. Búsqueda aumentada

La Generación Aumentada por Recuperación (RAG por sus siglas en inglés) es una técnica que enriquece el *prompt* de un LLM con información recuperada en tiempo real de fuentes externas, permitiendo que el modelo genere respuestas contextualizadas con datos actuales que no forman parte de su entrenamiento. En concreto, se utiliza Tavily Search API, que permite realizar búsquedas y devuelve las fuentes consultadas, asegurando la trazabilidad de los datos obtenidos. Además, permite que sus respuestas sean consumidas por el LLM directamente, sin necesidad de procesamiento adicional, lo que facilita su implementación y uso, además de contar con un plan gratuito con un volumen de consultas más que suficiente para el alcance actual del proyecto.

En este proyecto, Tavily se utiliza para incorporar información sobre el dominio del remitente como filtraciones de datos e incidentes de seguridad recientes directamente en el prompt de generación de recomendaciones.

2.5. Infraestructura y despliegue

El sistema se despliega íntegramente en servicios en la nube, sin infraestructura propia. Vercel es la plataforma utilizada para el despliegue del *frontend* React/Vite, sirviendo la aplicación como una *Single Page Application* (SPA), una aplicación web en la que toda la interfaz se carga una única vez y las actualizaciones se realizan dinámicamente sin recargar la página, desde una CDN global. Railway aloja los dos servicios del *backend*: el servidor Spring Boot y el microservicio FastAPI, desplegados como servicios independientes dentro del mismo proyecto. Supabase proporciona la

instancia de PostgreSQL en la nube, gestionando la base de datos de producción con conexión segura mediante JDBC/SSL desde Spring Boot. Hugging Face Hub actúa como repositorio de los artefactos del modelo de ML *tfidf.joblib* y *phishing_model.joblib*, desde donde el microservicio FastAPI los descarga al arrancar si no están disponibles localmente. La configuración detallada de cada servicio y las variables de entorno asociadas se describen en la sección 5.7.

2.6. Base de datos

PostgreSQL es un sistema gestor de bases de datos relacional de código abierto ampliamente utilizado. Su modelo relacional organiza la información de forma que se garantiza la integridad de los datos, facilitando asimismo las consultas agregadas.

Se ha elegido para este proyecto por tres motivos. En primer lugar, los datos gestionados por el sistema tienen una estructura fija y relaciones claras entre tablas, lo que hace que el modelo relacional sea el más adecuado. En segundo lugar, PostgreSQL es el motor compatible de forma nativa con Supabase, la plataforma de base de datos en la nube utilizada en el despliegue, y con Spring Boot mediante JPA/Hibernate, lo que simplifica la integración entre capas. En tercer lugar, es el estándar de facto en la industria para aplicaciones web con requisitos de integridad y consistencia de datos.

Capítulo 3. Estado de la Cuestión

El *phishing* es una de las amenazas más extendidas en el ecosistema digital, representando más del 20% de los ciberataques registrados en España en 2025, según estadísticas publicadas por el Instituto Nacional de Ciberseguridad (INCIBE) []6. De acuerdo con registros de ese mismo año de la Agencia de la Unión Europea para la ciberseguridad, llamada European Union Agency for Cybersecurity (ENISA), fue el principal punto de entrada, representando el 60% de todos los casos []4. Este tipo de técnicas explotan la vulnerabilidad social, lo que hace que la detección automática y la concienciación sean especialmente relevantes para resolver estos problemas.

En esta sección se investiga sobre el estado actual del problema del *phishing* y de las soluciones existentes, con el objetivo de presentar el contexto en el que se enmarca el presente trabajo. En primer lugar, se analiza la dimensión y alcance de este tipo de ataque, así como su evolución global y también en España. En segundo lugar, se analizan diferentes soluciones adoptadas en la industria, finalizando con la identificación de las limitaciones que plantean las soluciones existentes.

3.1. Panorama actual del phishing: estadísticas y evolución

Para analizar de forma cuantitativa el impacto del *phishing* de forma global, se hace una distinción necesaria entre las estadísticas a nivel mundial y a nivel español.

3.1.1. Visión global

En primera instancia se analiza la dimensión global. Una de las entidades que recopila datos más completos sobre *phishing* es el Anti-Phishing Working Group (APWG), una asociación internacional sin ánimo de lucro fundada en 2003. Durante 2025, el APWG registró un total de 3,8 millones de ataques de *phishing*, dato ligeramente superior al recogido el año anterior [2]. El segundo trimestre fue en el que se recopiló una mayor actividad, experimentando un incremento de hasta el 13% con respecto al trimestre anterior, con un total de 1.130.393 ataques, lo que se traduce en aproximadamente 565.000 ataques al mes y casi 19.000 ataques diarios.

Además del número de ataques recopilados, se debe analizar otros factores relevantes en este tipo de fraudes, en este caso el impacto económico que supone. En Estados Unidos, el Federal Bureau of Investigation (FBI) reportó que los ataques de *phishing* y *spoofing*, a pesar de haber disminuido en número en casi un 40% respecto a 2023, generaron unas pérdidas totales superiores a 215 millones de dólares americanos

en 2025, superando en doce veces los costes reportados en 2023 [3]. De estas pérdidas, casi un 50% (111 millones de dólares americanos) estuvieron relacionados con criptomonedas. Por lo tanto, a pesar de que el número de denuncias de este tipo de ataques haya disminuido, el impacto económico en las víctimas es mucho más significativa, pasando en dos años de $62,50 de media por incidente, a $1.125. Esto ilustra cómo las amenazas se vuelven más precisas en lugar de hacer ataques masivos.

En el ámbito europeo, ENISA publicó en octubre de 2025 el *Threat Landscape 2025*, donde se recopilan todos los ciberataques registrados en ese año. En el informe se determina que el 60% de los casos de una primera intrusión es mediante mensajes de *phishing*, donde el 73% de los casos registrados no derivaron en una intrusión confirmada, mientras que el 27% restante, sí [4]. Esto resalta la relevancia y magnitud del problema, pero a la vez pone en valor la detección temprana en este tipo de ataques. Otro dato que destacar es que el 42,4% de todos los ataques registrados, fueron en dispositivos móviles, lo que subraya la importancia de disponer de una herramienta adaptada en formato a estos dispositivos.

ENISA también hace referencia a la consolidación de un sistema criminal entorno al *Phishing-as-a-Service* (PhaaS), que permite desplegar campañas complejas y a gran escala con pocos conocimientos técnicos. Se pone como ejemplo la plataforma Darcula, empleada para suplantar más de 200 organizaciones en más de 100 países. En el plano técnico, se documenta la aparición de FlowerStorm, un kit de ataque *Adversary-in-the-Middle* (AiTM) que suplanta portales de Microsoft 365 y que es capaz de eludir autenticación en varios factores. También se señala que más del 80% de las campañas de ingeniería social del año 2025 incluían contenido generado o mejorado por inteligencia artificial.

3.1.2. Visión en España

En España, el INCIBE registró 122.223 incidentes de ciberseguridad en 2025, un 26% más que el año anterior, y el *phishing* fue la técnica más recurrente, con 25.133 incidentes, más de uno de cada cinco incidentes registrados en conjunto [5]. Además, se atendieron 142.767 consultas, de las cuales el 28% fueron por usuarios que denunciaban haber recibido algún intento de *phishing, vishing* o *smishing*. Estos datos demuestran que las soluciones técnicas actuales no son suficientes por sí solas y que la detección temprana y la orientación a los usuarios sigue siendo necesarias.

Desde el punto de vista policial, el Ministerio del Interior registró 464.801 ciberdelitos en 2024, de los cuales el 89% correspondió a estafas informáticas, con 350.795 víctimas y 19.322 personas detenidas o investigadas, un 14% más que el año anterior [6]. Sin embargo, 2024 fue el primer año en el que se observó un descenso en el

número de ciberdelitos en su conjunto. lo que apoya la idea de que hay menos ataques, pero están más dirigidos.

Analizando diferentes zonas geográficas, se observa un mismo patrón: el *phishing* es una amenaza global, estructural y en proceso de sofisticación y personalización, provocando un impacto cada vez más significativo en las víctimas; a pesar de que hay menos ataques, están más dirigidos.

3.2. Soluciones existentes

En el marco de los sistemas actuales de filtrado y detección de *phishing* existen bastantes herramientas, aunque con una fragmentación estructural: cada una se centra en un canal o contexto concreto, y ninguna ofrece una respuesta integrada, explicativa y accionable ante un mensaje sospechoso independientemente del origen. A continuación, se analizan las principales categorías de soluciones accesibles para el usuario.

3.2.1. Filtros integrados en proveedores de correo electrónico

Los principales proveedores de correo electrónico integran sistemas de filtrado de *spam* y *phishing* basados en listas negras, reglas heurísticas y, en algunos casos, modelos de aprendizaje automático (*Machine Learning*). Estos filtros trabajan sobre el texto y los metadatos del mensaje y, en muchos casos, obtienen altas tasas de detección para amenazas conocidas.

La principal limitación es estructural, ya que únicamente protegen frente a mensajes transmitidos por esa plataforma. Si un usuario recibe un mensaje de *phishing* desde otro proveedor o por SMS, este mecanismo no podría utilizarse para detectar posibles fraudes. Además, estas herramientas no ofrecen una explicación al usuario sobre por qué un mensaje ha sido clasificado como *phishing*, ni qué pasos se deben tomar, en el caso de que fuera necesario; el mensaje es trasladado a una carpeta de *spam* o se muestra un aviso genérico.

3.2.2. Herramientas web de análisis bajo demanda

Existen páginas y plataformas web en las que el usuario puede analizar URLs, ficheros o dominios concretos, que son contrastadas con grandes bases de datos de amenazas conocidas para determinar su legitimidad.

A pesar de estar disponibles únicamente teniendo acceso a Internet, presentan varias limitaciones, como que están limitadas a análisis seccionado y no al análisis del contenido completo del mensaje. Además, la eficacia depende de que el contenido que se está consultando ya se haya registrado y aparezca en la base de datos de la plataforma,

por lo que algunas campañas nuevas pueden no ser detectadas correctamente. Además, es necesario extraer información o metadatos concretos de un mensaje, lo que puede suponer una dificultad o barrera en algunos casos.

3.2.3. Aplicaciones y extensiones de navegador

Existen herramientas descargables en forma de aplicación o de extensión del navegador, que amplían la seguridad del usuario además de los filtros por defecto en los proveedores de correo. Estas extensiones previenen el acceso a URLs maliciosas en tiempo real. También algunas aplicaciones incorporan detección de SMS o llamadas sospechosas basadas en modelos de clasificación

Estas soluciones comparten las mismas limitaciones descritas anteriormente, cada una analiza el contenido en un medio y, además, las versiones gratuitas suelen tener limitaciones en las capacidades con respecto a las de pago y, para estar más seguro se deben configurar varias de manera simultánea, lo que la mayor parte de usuarios no están dispuestos a asumir, especialmente en dispositivos móviles, donde apenas un 49% tiene una aplicación antivirus [7].

3.2.4. Iniciativas colaborativas y *crowdsourcing*

Diversas plataformas y comunidades en línea permiten a los usuarios compartir mensajes sospechosos, enlaces fraudulentos y capturas de pantalla para recibir la valoración de otros miembros de la comunidad o de expertos. Son recursos accesibles que, en ocasiones, permiten identificar campañas de *phishing* antes de que aparezcan en bases de datos oficiales.

La principal limitación de estas iniciativas es la latencia en la respuesta ya que, para que el usuario pueda recibir una respuesta, es necesario que otro miembro de la comunidad lo vea, lo analice y responda, lo que puede prolongarse en el tiempo, al contrario que una herramienta automatizada. En un contexto en el que se requiere que el usuario actúe de manera inmediata, esa tardanza elimina la eficacia de la herramienta como mecanismo de detección temprana, teniendo más valor a nivel informativo o preventivo.

3.2.5. Marco legal y orientación al usuario

En el plano normativo, las dos normas europeas especialmente relevantes son la Directiva (PSD2) [8], transpuesta en España mediante el Real Decreto-ley 19/2018 [9], para definir un mercado de servicios de pago adecuado y regular estos servicios, especialmente el pago en línea y el Reglamento General de Protección de Datos (RGPD), que define cómo deben tratarse los datos personales y qué derechos tiene el usuario con respecto a su información [10].

A pesar de existir un marco normativo esencial, la orientación práctica sobre los pasos que un usuario puede seguir tras sufrir un ataque tras haber sido víctima de un ataque de *phishing* es suficiente en cuanto al contenido, pero se encuentran dispersos entre múltiples fuentes, sin un punto de acceso único para el usuario, lo que complica en gran medida las gestiones. Este proyecto recoge ese contexto normativo como referencia para elaborar una guía sintética y práctica que acompañe al usuario después de la detección técnica del posible *phishing*.

3.3. Investigación Académica en Detección de Phishing con ML

La detección de *phishing* mediante aprendizaje automático es una de las líneas de investigación más activas dentro de la ciberseguridad aplicada. Las decisiones metodológicas clave que aparecen en la literatura, incluyendo la elección del corpus [1], la representación del texto y el clasificador utilizado, son las mismas que se han tomado en el desarrollo de este proyecto, por lo que este apartado sirve también como justificación de las decisiones técnicas descritas en el capítulo 5.

3.3.1. Conjuntos de Datos de Referencia

La investigación de detección de *phishing* por correo electrónico dispone de un conjunto de corpus, o *datasets*, públicos que han servido como referencia para un gran número de proyectos de investigación. Entre los más utilizados destacan el *Enron*, con una gran cantidad de correos legítimos, el CEAS-2008 y las colecciones de correos de fraude nigeriano, compuestas íntegramente por mensajes maliciosos.

Es habitual unir varias fuentes, ya que los correos en cada corpus presentan bastante heterogeneidad, y este proceso permite obtener un conjunto de datos más completo y equilibrado. Por este motivo, se ha decidido combinar y depurar los tres corpus mencionados, proceso que se explica con mayor detalle en la sección 5.2.1.

También se debe tener en cuenta que existe un gran número de corpus en los que se recogen correos electrónicos y mensajes SMS, sin embargo, las técnicas de *phishing* han ido evolucionando con el tiempo, por lo que se debe buscar bases de datos lo más recientes posibles.

3.3.2. Vectorización del texto: TF-IDF frente a otras representaciones

Para poder clasificar correos electrónicos mediante aprendizaje automático, el texto debe transformarse en una representación numérica. Los tres enfoques más frecuentes en la literatura son TF-IDF, Word2Vec y BERT, y la elección entre ellos

implica un compromiso entre rendimiento, coste computacional e interpretabilidad. Un estudio comparativo publicado en *Computers, Materials & Continua* en 2024 evaluó cuatro clasificadores (Regresión Logística, Árbol de Decisión, Random Forest y Perceptrón Multicapa) sobre vectorizaciones TF-IDF, Word2Vec y BERT, obteniendo métricas muy competitivas con la representación en TF-IDF y la clasificación Perceptrón Multicapa, obtenido las métricas: precisión, *recall* y F1-score de 0,98 [11]. Todos estos parámetros serán definidos en la sección 5.2.

Además de su rendimiento, TF-IDF presenta una ventaja directamente relevante para este proyecto: los pesos asignados a cada término son interpretables, lo que permite identificar las palabras con mayor influencia en cada clasificación y mostrarlas al usuario como indicadores de la decisión del modelo, funcionalidad implementada en el *endpoint /predict* descrito en la sección 5.3.1.

3.3.3. Algoritmos de Clasificación

Uno de los algoritmos más citados en investigaciones recientes en detección de *phishing* es Random Forest debido a su alto rendimiento, que se encuentra entre el 94% y el 99% en función del conjunto de datos [12]. Este algoritmo crea un gran número de árboles de decisión, donde cada uno está entrenado con una muestra aleatoria del *dataset* y un subconjunto de características aleatorias. Al recibir un nuevo correo, cada uno de los árboles decide si el correo es o no legítimo y la clasificación final es la que obtiene más votos. Produce probabilidades nativas, pero no es interpretable a nivel de palabras individuales y además es lento en el entrenamiento.

Otro algoritmo que aparece de manera recurrente es el Support Vector Machine lineal (LinearSVC), que es una variante que usa un hiperplano de separación lineal, adecuado para espacios de alta dimensionalidad creado por TF-IDF. Se busca encontrar el hiperplano que mejor separa las dos clases maximizando el margen entre correos de *phishing* y legítimos. Funciona bien con grandes dimensiones, pero no produce probabilidades de forma nativa, teniendo que calibrarlo posteriormente y añadiendo complejidad [13].

El algoritmo de Naive Bayes (MultinomialNB), variante del Naive Bayes diseñado específicamente para características de frecuencia como TF-IDF, es la más utilizada en clasificación de texto. Asume independencia entre las palabras, simplificando el análisis y obteniendo buenos resultados en este tipo de análisis [14]. Produce probabilidades de forma nativa pero no están calibradas, ya que parte de la suposición de la independencia entre palabras.

Por otro lado, el algoritmo de Regresión Logística ofrece resultados competitivos, además de contar con dos ventajas: es un algoritmo más simple, que permite un menor

coste computacional, y que sus coeficientes son interpretables, lo que facilita poder generar explicaciones comprensibles para el usuario, que es uno de los objetivos principales de la herramienta.

A diferencia de los clasificadores anteriores, basados en representaciones estáticas del texto, los modelos de lenguaje BERT y RoBERTa son clasificadores preentrenados. Han documentado tasas de detección de hasta 98,99% y 99,08%, respectivamente, superando a los clasificadores clásicos con un margen medio del 4,7% [15]. Sin embargo, estos algoritmos implican una mayor latencia de respuesta, más consumo de memoria y mayor complejidad de despliegue, por lo que no se consideran como alternativa inicial para desarrollar el proyecto.

3.3.4. El sesgo de idioma

Una limitación actual de la mayor parte de los corpus, y por lo tanto también de este proyecto, es el idioma; la práctica totalidad de los corpus públicos están compuestos por correos en inglés []17. Es por este motivo que la extensión a otros idiomas se plantea como una de las principales líneas de trabajo futuras listadas en el capítulo 7.

3.4. Sistemas híbridos ML y LLM en ciberseguridad

La incorporación de modelos de lenguaje de gran tamaño (LLM) en sistemas de detección de phishing es otra línea de investigación más reciente y activa en la literatura. A diferencia de los enfoques descritos en el apartado anterior, donde el LLM actúa como clasificador principal, en este proyecto el LLM no contribuye a la detección, sino que actúa como una capa complementaria que transforma la decisión técnica del modelo en una respuesta comprensible y accionable para el usuario. Este enfoque tiene respaldo directo en investigaciones recientes relacionadas con la ciberseguridad.

3.4.1. LLM para explicabilidad

La falta de explicabilidad es un problema ampliamente documentado en sistemas de aprendizaje automático: el modelo produce una etiqueta o una probabilidad, pero el usuario no entiende por qué. EXPLICATE es un *framework* publicado en 2025 que aborda este problema, llegando a alcanzar una precisión del 98,4% combinando un clasificador de ML con un LLM, donde el LLM traduce los indicadores técnicos de la detección a explicaciones en lenguaje natural comprensibles para cualquier usuario [16].

La integración del LLM no mejora la detección en sí, sino la comprensión del resultado por parte del usuario, que es precisamente uno de los objetivos principales de esta herramienta, definidos en la sección 4.2.

3.4.2. LLM para recomendaciones defensivas contextuales

Además de la explicabilidad, también se ha estudiado el uso de LLM para generar recomendaciones defensivas personalizadas en función del contenido específico del mensaje analizado. Un sistema presentado en ESORICS 2025 propone un *framework* que permite analizar el contenido del correo y las URLs incluidas con un 98% y un 97% respectivamente [17]. Esto se consigue utilizando un LLM para generar recomendaciones defensivas contextuales basadas en el *framework* MITRE D3FEND, un estándar de contramedidas de ciberseguridad.

Este planteamiento es similar al que se expone en este proyecto, donde además se tiene en cuenta el perfil del usuario, en concreto si tiene cuenta en la empresa remitente, y si se ha detectado urgencia en el mensaje o alguna filtración reciente del remitente mediante consultas con Tavily

3.4.3. Limitaciones del uso de LLMs en producción

El uso de LLMs en sistemas de detección de phishing presenta limitaciones prácticas. La principal es la latencia, ya que un LLM introduce un tiempo de respuesta significativamente mayor que un clasificador clásico, lo que hace inviable utilizarlo como primer filtro en sistemas de alto volumen.

Estas limitaciones han sido tenidas en cuenta al desarrollar el proyecto y se han utilizado como criterio de diseño, ya que solo se realiza la llamada al LLM tras obtener la clasificación del modelo ML, mediante la separación de dos *endpoints* independientes, pudiendo determinar cuándo solicitar una recomendación personalizada. De esta forma se elude realizar llamadas evitables que consumirían tokens y aumentarían la latencia de la herramienta sin aportar valor significativo al usuario.

3.5. Limitaciones de las soluciones actuales

A lo largo de esta sección se han analizado las principales soluciones existentes en el ámbito de la detección de phishing, tanto desde el punto de vista de la industria como de la investigación académica. En todos los casos se observan limitaciones recurrentes que ninguna solución actual resuelve de forma completa, y que justifican el desarrollo del sistema presentado en este trabajo.

La primera limitación es la fragmentación por canal. Las soluciones existentes —filtros de correo, aplicaciones de detección de SMS, extensiones de navegador— protegen un único canal de comunicación y no ofrecen al usuario un punto de análisis centralizado e independiente del origen del mensaje. Un usuario que recibe un mensaje sospechoso

por un canal no cubierto por su herramienta habitual no tiene una alternativa inmediata y accesible.

La segunda es la ausencia de explicaciones al usuario. La mayoría de los sistemas de detección, tanto comerciales como académicos, producen una etiqueta o un porcentaje de riesgo sin explicar al usuario qué elementos del mensaje han activado la alerta. Esta falta de transparencia puede reducir la confianza en el sistema y no contribuye a que el usuario aprenda a identificar este tipo de ataques por sí mismo, suponiendo un riesgo potencial para el usuario, ya que todas las herramientas con falibles.

La tercera es la falta de orientación accionable. Detectar un mensaje como phishing no es suficiente si el usuario no sabe qué hacer a continuación. Como se ha visto en la sección 3.2.5, la información sobre los pasos a seguir existe, pero está dispersa entre múltiples fuentes sin un punto de acceso único.

La cuarta es la dependencia de amenazas conocidas. Los sistemas basados en listas negras y bases de datos de indicadores de compromiso presentan una ventana de vulnerabilidad ante campañas nuevas que utilizan infraestructura no catalogada previamente, como se ha documentado en los informes de ENISA y el APWG.

El sistema desarrollado en este trabajo aborda estas cuatro limitaciones de forma directa. La arquitectura híbrida de ML y LLM permite ofrecer tanto una clasificación eficiente del contenido del mensaje como una explicación personalizada en lenguaje natural y recomendaciones accionables adaptadas al contexto específico del usuario. La integración con Tavily añade una capa de inteligencia en tiempo real sobre el dominio del remitente, reduciendo parcialmente la dependencia de amenazas previamente catalogadas. Y el módulo de orientación legal recoge en un único punto accesible la información normativa y los pasos a seguir en caso de haber sido víctima de un ataque, eliminando la necesidad de que el usuario busque esa información de forma dispersa.

Todo ello se ofrece además como una herramienta de acceso libre, sin barreras económicas ni requisitos de configuración, lo que la diferencia de las soluciones comerciales analizadas en la sección 3.2 y la acerca al perfil del usuario particular que, según los datos de INCIBE recogidos en la sección 3.1.2, representa la mayor parte de las víctimas de phishing en España.

Capítulo 4. Definición del Trabajo

El *phishing* constituye una amenaza global que supone un riesgo real para la población. Sin embargo, a pesar de que existen múltiples soluciones en el mercado, se mantiene la tendencia en la que los mensajes son cada vez más difíciles de identificar en estilo, diseño y contenido, ya que en muchas ocasiones incluyen datos personales, lo que aumenta su credibilidad y dificultan la detección con filtros automáticos.

Las soluciones existentes presentan una limitación estructural común: cada herramienta cubre un canal concreto, ninguna ofrece explicaciones comprensibles al usuario y la orientación sobre qué hacer tras recibir un mensaje sospechoso está dispersa entre múltiples fuentes. PhishGuard nace para cubrir ese vacío: una herramienta centralizada, explicable y orientada al usuario que combina detección automática con orientación personalizada

4.1. Justificación técnica y de mercado

Con este proyecto se han buscado principalmente dos motivaciones que justifican tanto el desarrollo técnico como su relevancia en el contexto actual.

La primera es de carácter técnico. Las soluciones de detección basadas exclusivamente en aprendizaje automático producen una etiqueta o un porcentaje de riesgo, pero carecen de explicaciones sobre qué elementos del mensaje han activado la alerta ni qué debe hacer el usuario a continuación. Al mismo tiempo, los sistemas basados en LLM ofrecen capacidad explicativa, pero presentan una latencia y un coste de inferencia que los hace inviables como primer filtro. La arquitectura híbrida ML + LLM desarrollada en este proyecto pretende responder a esta limitación: el clasificador proporciona una respuesta eficiente y trazable, y el LLM actúa como capa complementaria que transforma esa respuesta en una recomendación personalizada y comprensible.

La segunda motivación es de accesibilidad y mercado. En los datos de INCIBE recogidos en la sección 3.1.2, el 33% de las consultas recibidas por su línea telefónica de consultas 2024 procedían de usuarios que habían recibido un intento de *phishing* y no sabían cómo actuar. Las soluciones comerciales más completas no están pensadas para los usuarios particulares, sino que están orientadas a entornos empresariales, ya que presentan barreras económicas y técnicas. Este proyecto cubre ese espacio: una herramienta de acceso libre, sin requisitos de registro ni instalación, que combina detección técnica con orientación para cualquier perfil de usuario.

4.2. Objetivos

El objetivo principal de este proyecto es desarrollar una herramienta unificada que permita la detección de mensajes fraudulentos provenientes de cualquier fuente o servidor de mensajería, así como ayudar al usuario a generar una mentalidad crítica para poder ayudar a crear una sociedad más consciente y resiliente ante este tipo de estafas.

Se ha querido crear una única herramienta para detección de mensajes de *phishing* por dos motivos principales. En primer lugar, la simplicidad y facilidad que proporciona tener una única herramienta se considera primordial. Uno de los problemas de las herramientas actuales es que se presentan muy fraccionadas, donde cada una trabaja en una sección concreta, lo que puede complicar su uso, especialmente en momentos de urgencia como los que tratan de generar muchos mensajes fraudulentos. En segundo lugar, se observa que los proveedores de correo electrónico han implementado mayores medidas anti-*phishing*, ya que ha sido durante mucho tiempo el medio utilizado por los ciberdelincuentes. Sin embargo, actualmente existen un sinfín de plataformas y aplicaciones en las que se pueden recibir este tipo de mensajes, por lo que este aspecto también se pone como pilar de la intención de la herramienta presentada.

En ataques de ingeniería social, como el *phishing*, se considera un problema complejo de solucionar únicamente con herramientas técnicas, ya que el objetivo es la persona, no el sistema. Es por este motivo por el que la segunda parte del objetivo principal de esta herramienta es asistir al usuario en su decisión, por lo que se ha decidido dar importancia al modelo técnico, pero también a las recomendaciones personalizadas para cada usuario, así como las fuentes de las que proviene esa información para que toda la información pueda ser verificable. De esta manera lo que se busca es darle al usuario una herramienta puramente técnica y estadística junto con recomendaciones personalizadas que le ayuden tomar una decisión, añadiendo sugerencias genéricas preventivas, de comprobación y reactivas ante una situación de duda tras recibir un mensaje sospechoso.

Además de estos dos objetivos generales, también se ha querido conseguir los siguientes:

- Crear una herramienta accesible y sencilla de utilizar, con independencia de los conocimientos previos del usuario y tratando de hacer que se muestre como una herramienta útil e informativa.

- Crear una herramienta automatizada que proporcione resultados inmediatos e independientes de la intervención de terceros, a diferencia de las iniciativas de *crowdsourcing* analizadas en la sección 3.2.4.
- Garantizar la transparencia en todo el proceso, informando debidamente al usuario de que los resultados no son asesoramiento legal ni técnico en ningún caso, y que es el propio usuario el que tiene la última decisión.
- Proporcionar información relevante y centralizada en un único punto de acceso, en forma de recomendaciones personalizadas, sugerencias, estadísticas y buenas prácticas, a diferencia de la orientación dispersa entre múltiples fuentes analizada en la sección 3.2.5, y con el objetivo de ayudar a crear una sociedad más concienciada y preparada para enfrentarse a estos ataques.
- Crear una herramienta actualizada y en constante evolución, tanto en la base de datos como en el modelo predictivo, utilizando las contribuciones de la comunidad de usuarios.

Se busca conseguir estos objetivos a través del cumplimiento de los siguientes objetivos técnicos específicos:

- Diseñar un modelo predictivo en Python: entrenar y evaluar un clasificador supervisado que, a partir del asunto y el texto del mensaje introducido, estime la probabilidad de que el mensaje sea *phishing*.
- Diseñar un sistema de recomendaciones personalizadas: utilizando un modelo previamente entrenado y con acceso a Internet para poder investigar el remitente del mensaje.
- Desarrollar una arquitectura web basada en microservicios: crear una infraestructura escalable e independiente, donde el código se encuentre perfectamente diferenciado entre la parte visual de la aplicación (*frontend*), la parte del *backend* y el modelo predictivo.
- Desplegar íntegramente la aplicación en la nube: tanto el *frontend*, como *backend*, la API y la base de datos en diferentes servicios, para asegurar escalabilidad, disponibilidad, resiliencia y mantenimiento eficiente del sistema.
- Diseñar y completar una base de datos propia: crear una base de datos que almacena las contribuciones de los usuarios que den su permiso expreso, pudiendo reentrenar el modelo en un futuro.
- Ofrecer una interfaz accesible: diseñar un *frontend* limpio, con formato adaptado al dispositivo y que sea visualmente explicativa, incluyendo también una guía de uso simple, concisa y útil.
- Elaborar un módulo de recomendaciones generales básicas: integrar una guía resumida y actualizada que recoja: cómo actuar de manera preventiva para evitar

ser víctima de un mensaje de *phishing*, cómo actuar en el momento de recibir cualquier mensaje y cómo actuar en el caso de que se sospeche de haber sido víctima.

- Implementar mecanismos de transparencia y trazabilidad: incluir en cada análisis las fuentes consultadas por el módulo de búsqueda y añadir avisos legales explícitos que delimiten el alcance informativo de los resultados, diferenciando claramente entre la estimación estadística del modelo y el asesoramiento profesional.
- Mostrar estadísticas y contribuciones en directo: publicar estadísticas que permitan al usuario observar las tendencias en las contribuciones realizadas por toda la comunidad, así como contribuciones recientes aportadas por otros usuarios.
- Implementar un sistema de contribución voluntaria de datos: permitir que los usuarios cedan de forma anónima y con consentimiento explícito los mensajes analizados, construyendo un corpus de entrenamiento propio que permita el reentrenamiento y la actualización del modelo predictivo en futuras versiones del sistema.

4.3. Metodología

La metodología seguida durante el proyecto ha sido una metodología ágil iterativa. El proceso se ha dividido en varias fases que no se han ejecutado de manera estrictamente secuencial, sino que se ha trabajado en paralelo en varias de ellas, incorporando nuevas decisiones de diseño actualizadas a las mejoras y avances en el desarrollo.

La división del desarrollo se ha organizado en las siguientes fases:

4.3.1. Fase 1: Investigación y planificación inicial

La primera fase consistió en la revisión del estado del arte en detección de phishing, la selección de los *datasets* de entrenamiento y la definición de la arquitectura inicial del sistema.

En esta fase se estableció un diseño preliminar en el que la base de datos se alojaría en una máquina virtual local, decisión que posteriormente se modificó para realizar un despliegue completo en la nube.

4.3.2. Fase 2: Desarrollo del modelo de Machine Learning

En paralelo con la configuración de la infraestructura inicial, se desarrolló el modelo de aprendizaje automático en un *notebook* de Python. Esta fase incluyó la unión

y depuración de los tres corpus seleccionados, el preprocesamiento y vectorización TF-IDF, el entrenamiento del clasificador de Regresión Logística y el análisis del umbral óptimo, todo ello documentado en la sección 5.2. El desarrollo simultáneo del modelo y de la infraestructura permitió validar desde el principio que los artefactos generados eran compatibles con el resto de los módulos de la herramienta.

Los artefactos resultantes se almacenaron en Hugging Face Hub para su posterior uso en el microservicio de inferencia.

4.3.3. Fase 3: Backend, base de datos y frontend

El diseño del esquema de la base de datos PostgreSQL comenzó con una estructura básica que fue evolucionando a lo largo del desarrollo, incorporando nuevas tablas y dependencias a medida que aparecían nuevas funcionalidades. Inicialmente la base de datos se alojó en una máquina virtual local, lo que permitió validar el esquema y las consultas de forma ágil antes de la migración definitiva a Supabase.

El desarrollo del *backend* en Spring Boot y la construcción del *frontend* en React/Vite avanzaron en paralelo durante la mayor parte del desarrollo del proyecto: cada nuevo *endpoint* definido en el *backend* se verificaba de forma independiente y antes de añadir su elemento correspondiente en el *frontend*. De esta forma se garantizaba el progreso parejo y coherente entre los componentes del proyecto.

4.3.4. Fase 4: Integración del servicio ML y LLM

En la primera versión, el modelo de ML se ejecutaba directamente desde el *backend* de Spring Boot, sin embargo, más adelante se separó en un microservicio independiente de Python, desde el que se hacía la descarga de los modelos desde Hugging Face Hub al arrancar. Esta decisión, aunque no estaba en el diseño inicial, mejoró la modularidad y la escalabilidad del sistema.

La integración del servicio de inferencia en FastAPI, junto con los módulos de Groq y Tavily, se realizó de manera iterativa después de verificar el funcionamiento de extremo a extremo de la aplicación, ya que este módulo requería que otras funcionalidades ya hubieran sido implementadas para poder hacer pruebas. El *prompt engineering* del módulo Groq y la integración con Tavily también se mejoraron de forma iterativa durante esta fase.

4.3.5. Fase 5: Despliegue en cloud

Una vez validado el sistema en local, se procedió a la migración completa a servicios en cloud: el *frontend* se desplegó en Vercel, el *backend* y el microservicio FastAPI en Railway, y la base de datos se migró de la máquina virtual a Supabase. De

esta forma, el despliegue en la nube fue íntegro, eliminando cualquier dependencia de infraestructura local.

4.3.6. Fase 6: Evaluación y análisis de resultados

La última fase consistió en la evaluación del modelo mediante métricas de clasificación, el análisis de casos de uso reales con correos propios y el análisis cualitativo de las recomendaciones generadas por el LLM, incluyendo la verificación de las fuentes proporcionadas por Tavily en cada caso.

4.4. Planificación y estimación económica

El sistema desarrollado en este proyecto se ofrece de forma completamente gratuita y sin modelo de negocio asociado en su versión actual. El objetivo principal es la accesibilidad: cualquier usuario puede analizar mensajes sin necesidad de registro, suscripción ni pago de ningún tipo.

Los costes de infraestructura se mantienen en cero gracias al uso de los planes gratuitos de los servicios de despliegue utilizados (Vercel y Supabase), suficientes para el volumen de uso actual del proyecto, y exceptuando Railway, que requiere de un pago mensual de un dólar para el alcance actual del proyecto, pudiendo aumentar al plan *Hobby* por $5 o al plan *Pro* por $20 al mes en función de los requisitos de la web. Los servicios de terceros con límite de uso gratuito, como la API de Groq y la API de Tavily, cubren igualmente las necesidades del sistema en su estado actual.

En caso de que el proyecto escalara en número de usuarios o funcionalidades, podrían explorarse vías de financiación que mantuvieran el acceso gratuito para el usuario final, como la incorporación de publicidad no intrusiva o un modelo *freemium* con funcionalidades avanzadas opcionales. En cualquier caso, la gratuidad del acceso básico se considera un principio no negociable del proyecto, dado que su propósito principal no es comercial.

4.5. Marco legal y normativo (PSD2, RGPD)

Para el desarrollo de las sugerencias generales de la aplicación, se ha estudiado el marco legal y normativo vigente en España, analizando en detalle las dos normativas más relevantes para el proyecto y su relevancia para el contexto del proyecto: la Directiva PSD2, que regula los servicios de pago electrónico y establece medidas de seguridad frente al fraude, y el Reglamento General de Protección de Datos (RGPD), que define las

obligaciones de cualquier sistema que recoja, almacene o trate datos personales de usuarios.

4.5.1. Reglamento General de Protección de Datos (RGPD)

El Reglamento General de Protección de datos (RGPD) 2016/679, regula el tratamiento de datos personales de ciudadanos europeos y es de aplicación directa en todos los estados miembros. Su objetivo principal es garantizar que el usuario tenga control sobre sus propios datos y que cualquier organización que los trate lo haga de forma lícita, transparente y proporcional [22].

Los siete principios que articulan el RGPD son especialmente relevantes para este proyecto. El principio de licitud exige que exista una base jurídica para el tratamiento de datos: en este caso, el consentimiento explícito del usuario, que se recoge mediante los *checkboxes* obligatorios del formulario de análisis descritos en la sección 5.6.2. El principio de transparencia obliga a informar al usuario de forma clara sobre qué se hace con sus datos, cuándo y para qué: esto se materializa en los avisos legales incluidos en la interfaz y en la distinción explícita entre el análisis estadístico del modelo y el asesoramiento profesional. El principio de minimización exige recoger únicamente los datos estrictamente necesarios: el sistema no solicita ningún dato de identificación personal, y el campo de remitente es opcional. El principio de limitación de finalidad implica que los datos cedidos voluntariamente para el reentrenamiento del modelo no pueden utilizarse para ningún otro propósito, lo que queda reflejado en el diseño de la tabla *contribuciones_modelo* descrita en la sección 5.5.1. El séptimo principio es el de integridad y confidencialidad, que exige que los datos se traten de forma segura frente a accesos no autorizados. En este proyecto se materializa mediante tres medidas: el cifrado SSL en la conexión con la base de datos, la activación de *Row Level Security* (RLS) en Supabase para bloquear accesos directos no autorizados a las tablas, y el almacenamiento de credenciales y claves de API en los servicios de despliegue, nunca en el código fuente.

Dos principios complementarios del RGPD también han guiado el diseño del sistema. El principio de *privacy by design* establece que la privacidad debe integrarse desde las primeras fases del diseño, no añadirse a posteriori: el sistema no almacena el contenido de los mensajes analizados salvo consentimiento explícito, y las recomendaciones generadas por el LLM son *transient*, es decir, no se persisten en la base de datos. El principio de *privacy by default* exige que la configuración inicial del producto sea la más restrictiva posible: el *checkbox* de contribución al corpus de entrenamiento es opcional y está desmarcado por defecto.

4.5.2. Directiva PSD2 y su Transposición en España

La Segunda Directiva de Servicios de Pago (PSD2, Directiva UE 2015/2366), transpuesta en España mediante el Real Decreto-ley 19/2018, regula los servicios de pago electrónico en Europa con el objetivo de aumentar la seguridad, la transparencia y la competencia en el sector financiero [23]. Su medida más relevante en el contexto de este proyecto es la Autenticación Reforzada de Cliente (SCA por sus siglas en inglés), que obliga a los proveedores de servicios de pago a verificar la identidad del usuario mediante al menos dos factores independientes: algo que sabe (contraseña o PIN), algo que tiene (dispositivo móvil o token) o algo que es (biometría).

La relevancia de la PSD2 para este proyecto no es de cumplimiento directo, ya que el sistema no procesa pagos, sino de contexto: una proporción significativa de los ataques de phishing documentados en el capítulo 3 suplantan a entidades bancarias y proveedores de pago, precisamente porque el usuario asocia esos contextos con operaciones de alto valor. Conocer el marco que regula esos servicios permite al sistema ofrecer una orientación más precisa si el mensaje analizado hace referencia a operaciones de pago, bloqueos de cuenta o verificaciones bancarias, que son patrones de urgencia frecuentes en este tipo de ataques.

El módulo de orientación legal de la sección 5.6.4 recoge de forma resumida y accesible los derechos del usuario recogidos tanto en el RGPD como las protecciones descritas en la PSD2, con el objetivo de que cualquier persona, independientemente de su formación jurídica, pueda entender qué puede exigir y a quién dirigirse en caso de haber sido víctima de un ataque.

Capítulo 5. Sistema Desarrollado

En esta sección se describe en detalle el sistema desarrollado, denominado PhishGuard, que constituye el núcleo técnico del proyecto. Se trata de una aplicación web de acceso libre que permite al usuario analizar el contenido de cualquier mensaje sospechoso y obtener, de forma inmediata, una estimación probabilística de si se trata de un intento de *phishing*, una explicación de los indicadores que han activado la alerta y un conjunto de recomendaciones personalizadas adaptadas a su situación concreta.

Las siguientes secciones describen cada uno de estos componentes del sistema, comenzando por la visión general de la arquitectura y el flujo de datos, y continuando con el módulo de ML, el servicio de inferencia, el *backend*, la base de datos, el *frontend* y el despliegue en cloud.

5.1. Visión general de la arquitectura

El flujo de datos comienza introduciendo el correo electrónico en la página web y acto seguido el controlador envía la petición de predicción al modelo de *Machine Learning* (ML). Teniendo el porcentaje y los datos adicionales añadidos por el usuario, se envía una nueva petición de recomendación al modelo *Large Language Model* (LLM). En caso de que el campo *remitente* no esté vacío, se envía esa información a través de Tavily para poder hacer una búsqueda de filtraciones o noticias recientes, pudiendo de esta forma personalizar en mayor medida las recomendaciones enviadas al usuario. En la base de datos se almacena tanto el porcentaje estimado, como los datos introducidos por el usuario y las palabras más utilizadas para las estadísticas globales (y en directo) de la aplicación.

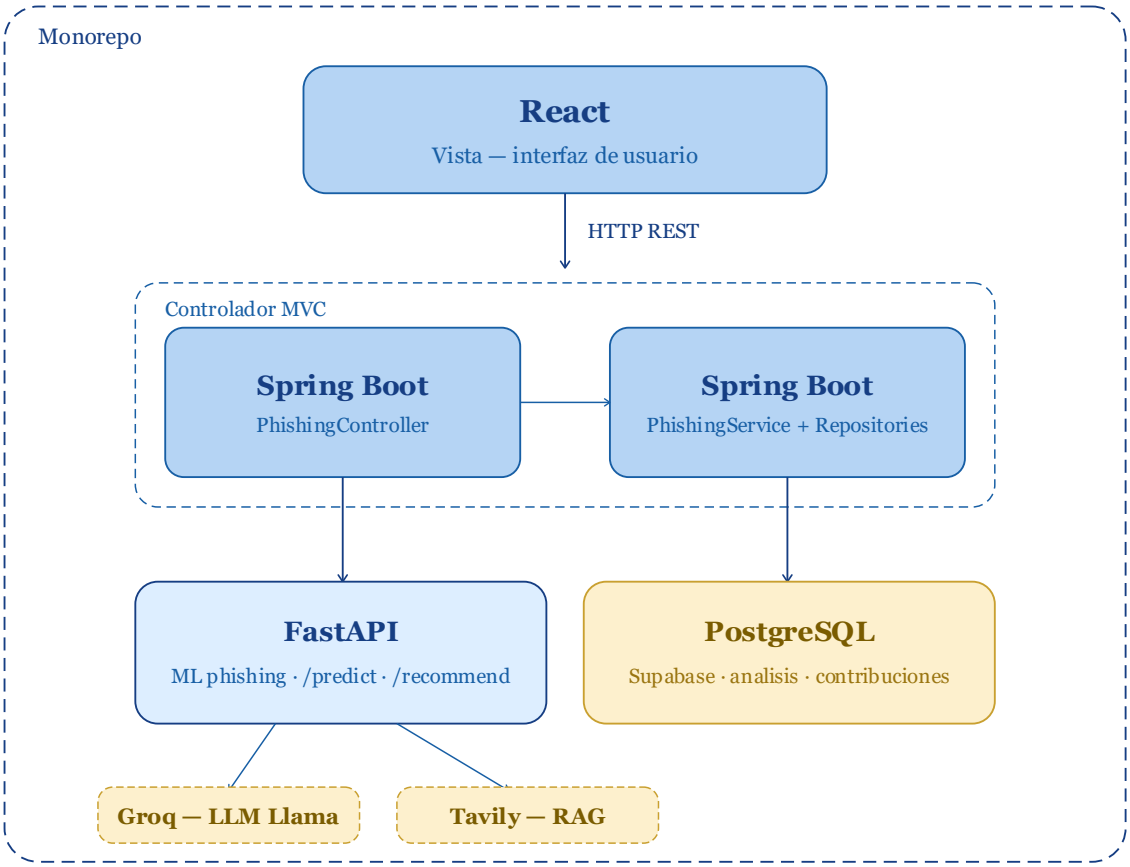

Figura 1. Arquitectura del sistema en formato monorepo siguiendo el patrón MVC. React (vista) se comunica con Spring Boot (controlador), que delega en FastAPI para la inferencia del modelo de phishing. FastAPI integra Groq (LLM Llama) y Tavily (RAG) para la generación de recomendaciones, mientras que PostgreSQL gestiona la persistencia.

5.2. Módulo de Machine Learning

El módulo de Machine Learning (ML) es el componente encargado de analizar el contenido del mensaje introducido por el usuario y estimar la probabilidad de que se trate de un intento de *phishing*. Está desarrollado íntegramente en Python, haciendo uso de la biblioteca *scikit-learn*, y sigue un *pipeline* clásico de aprendizaje supervisado: preprocesamiento del texto, vectorización TF-IDF y clasificación mediante Regresión Logística. Las siguientes secciones describen en detalle cada una de las decisiones tomadas durante su desarrollo, desde la selección y unión de los datasets hasta el versionado de los artefactos finales en Hugging Face Hub.

5.2.1. Datasets utilizados

Para el entrenamiento y evaluación del modelo de *Machine Learning* (ML) se ha contado con un conjunto de tres *datasets* públicos de correos electrónicos, tanto legítimos

como fraudulentos, complementados con un dataset propio recopilado para la validación con casos reales.

Datasets públicos de entrenamiento

Para el entrenamiento y el test del modelo de ML se ha contado con un conjunto de tres *datasets* diferentes de correos electrónicos, tanto legítimos como ilegítimos.

El primero de ellos es CEAS_08, una base de datos en formato CSV con información del remitente, el receptor, la fecha, el asunto y el cuerpo del correo. También aparece la etiqueta en la que se determina si el correo es (1) o no es (0) *phishing*, así como un *boolean* en el que se indica si hay o no hay incluida una *URL* en el correo electrónico. En esta base de datos se han incluido tanto mensajes de *phishing* como de *no phishing* y contiene un total de 39.153 correos.

El segundo *dataset* es Nigerian_Fraud, también en formato CSV con la misma estructura de campos que el anterior: remitente, receptor, fecha, asunto, cuerpo, etiqueta determinando si el correo es (1) o no es (0) *phishing,* así como si hay o no un *URL* incluida en el cuerpo del correo. En este caso todos los correos electrónicos incluidos han sido clasificados como *phishing*, con un total de 3.331 correos.

El tercero es Enron, que recoge únicamente el asunto, el cuerpo y la etiqueta determinando si es (1) o no es (0) *phishing*. Todos los correos electrónicos registrados han sido clasificados como *no phishing*, sumando un total de 29.766 correos.

De cada uno de los *datasets* se han utilizado los parámetros de asunto y cuerpo, concatenándolos como un único campo de texto, ya que son los parámetros que contienen el contenido textual analizable. El otro parámetro utilizado es la etiqueta de *phishing* o *no phishing*, mientras que el resto de los parámetros se han descartado, ya que el alcance del modelo llega únicamente al contenido del correo. Además, el análisis de parámetros como el remitente o la *URL* incluida en el correo se plantean como desarrollos futuros del proyecto.

Tras unir todos los *datasets*, se han eliminado todos los correos duplicados que pudieran existir, teniendo finalmente un *dataset* con un total de 50.324 correos, siendo 26.527 (52,71% del total) correos clasificados como *phishing* y 23.797 (47,29%) correos clasificados como *no phishing*.

Todos los *datasets* registran tanto el correo electrónico como el resto de los parámetros en inglés, por lo que el modelo está optimizado para la detección de *phishing* en ese idioma. El soporte en español y en otros idiomas se plantea como línea de trabajo futuro.

La división de entrenamiento y test se ha hecho con *random_state=25* fijo para asegurar que los resultados obtenidos son siempre los mismos. Además, para poder garantizar que el porcentaje entre mensajes de *phishing* y legítimos sea la misma en los conjuntos de entrenamiento y test, se ha utilizado el parámetro *stratify=y*. De esta forma, se mantiene que el 52% de los mensajes en test es *phishing* y el 48% es legítimo.

Dataset propio de validación

Con el objetivo de evaluar el comportamiento del modelo con mensajes actuales, se recopiló un *dataset* propio de 88 correos electrónicos recibidos, también por los colaboradores del proyecto entre octubre de 2025 y mayo de 2026. Este *dataset* se compone de dos fuentes principales: 53 correos recopilados mediante reenvío desde las cuentas de los colaboradores, de los cuales 46 son fraudulentos, y 35 correos recibidos directamente, siendo 30 de ellos *phishing*.

Los correos cubren una amplia variedad de entidades suplantadas, entre las que se encuentran entidades bancarias españolas (Santander, BBVA, ING, CaixaBank), organismos oficiales (Agencia Tributaria, DGT, Correos), servicios tecnológicos globales (Spotify, Netflix, Microsoft, MetaMask, TrustWallet) y servicios de mensajería (DHL, FedEx, USPS, Correos); así como correos de spam comercial no solicitado. Si bien esta última categoría no se categoriza como *phishing*, se han incorporado para poder evaluar la capacidad del modelo para diferenciar entre estas dos categorías con características similares, como ofertas o urgencia artificial.

Adicionalmente se incluyeron 12 correos legítimos de las mismas marcas suplantadas en el *dataset* fraudulento, con el objetivo de evaluar la tasa de falsos positivos del modelo y disponer de casos de referencia para el análisis comparativo de cabeceras desarrollado en la sección 6.3. Esta incorporación permite mantener un rigor metodológico, al permitir medir la capacidad de detección de *phishing* y también la precisión del modelo ante correos legítimos.

Dado que los *datasets* públicos de entrenamiento están íntegramente en inglés, todos los correos del *dataset* propio fueron traducidos al inglés antes de su análisis por el modelo, proceso descrito en detalle en la sección 6.3.

A diferencia de los *datasets* públicos, este *dataset* propio no se utilizó para el entrenamiento del modelo sino exclusivamente para su validación con casos reales, permitiendo evaluar su comportamiento ante correos actuales y originalmente en español. El análisis técnico de los correos se desarrolla en detalle en la sección 6.3.

5.2.2. Preprocesamiento y vectorización TF-IDF

Por un lado, el *TF* (*Term Frequency*) contabiliza el número de veces que aparece una palabra en un correo determinado, por lo que cuantas más veces aparece en el mismo, más peso tiene. Por otro lado, el *IDF* (*Inverse Document Frequency*), mide las palabras que aparecen en un gran número de correos del *dataset*, independientemente de si son o no *phishing*, dándoles un menor peso a dichas palabras; son términos que no ayudan a determinar la legitimidad del correo en cuestión. Por lo tanto, la combinación TF-IDF permite hacer un análisis en el que se le da un mayor peso a palabras frecuentes en un correo determinado, pero menos frecuentes en el resto de los correos; estas palabras serán las más determinantes y permitirán discernir la legitimidad de los mensajes introducidos a posteriori.

Para poder vectorizar las palabras dentro de los correos, es esencial definir una serie de parámetros para determinar cómo se tratarán.

- *Stop_words*: permite identificar y eliminar palabras comunes de una lista predefinida en el idioma indicado, ya que son palabras que son muy utilizadas, pero carecen de significado y que, por lo tanto, no aportarán valor al determinar si son o no legítimos. En este caso los correos electrónicos de los *datasets* seleccionados son en inglés, por lo que se ha determinado '*stop_words=english*'. Algunas de las palabras descartadas son: *it, is, I, at, on*, etc.
- *Max_df*: elimina palabras que, a pesar de tener significado, se repiten en una gran cantidad de correos, tanto de *phishing* como de *no phishing*, lo que se traduce en que no son determinantes para poder discernir entre ambas categorías. El parámetro seleccionado ha sido que las palabras que aparezca en más de un 95% de los correos, serán descartadas.
- *Min_df*: elimina palabras que, a pesar de tener significado, aparecen únicamente en una cantidad reducida de correos electrónicos. Siendo el tamaño total del *dataset* final 50.324 correos, se ha determinado que las palabras que aparezca en menos de 5 correos no se tendrán en cuenta para entrenar al modelo, ya que incluirlas añade dimensionalidad del modelo, pero el valor predictivo que pueden añadir es despreciable en comparación con la carga computacional que puede implicar. Este tipo de palabras pueden ser nombres propios, errores tipográficos, etc.

Este preprocesamiento se debe aplicar únicamente al *dataset* de entrenamiento, ya que el modelo aprenderá el vocabulario y pesos IDF del conjunto de entrenamiento y transforma los textos, utilizando *fit_tranform* con los datos de entrenamiento.

Por otro lado, al conjunto de test se debe aplicar este mismo vocabulario y pesos sin recalcular nada, utilizando *transform* con los datos de test. De esta forma se verifica

que el modelo no llega a ver nunca los datos de test, pudiendo hacer una evaluación correcta del modelo con estos nuevos correos.

5.2.3. Entrenamiento del clasificador (Regresión Logística)

Se ha seleccionado el algoritmo Regresión Logística para realizar la clasificación binaria (*phishing*/ *no phishing*). Este algoritmo no se limita a asignar el correo a una de las dos categorías, sino que produce una probabilidad de pertenencia a cada clase mediante la función sigmoide, lo que hace que sea idóneo para el proyecto, ya que la aplicación muestra al usuario un porcentaje de riesgo en lugar de un veredicto binario.

Matemáticamente, la probabilidad de que un correo sea phishing se obtiene mediante la función sigmoide aplicada sobre una combinación lineal de los pesos TF-IDF:

$$P(phishing) = \frac{1}{(1 + e^{-z})}$$

donde $z = w_1x_1 + w_2x_2 + \ldots + w_nx_n + b$, siendo cada x_i el peso TF-IDF de la palabra i y cada w_i su coeficiente aprendido.

Para clasificar los correos correctamente se deben tener en cuenta:

- *Max_iter*: se ha seleccionado un número de iteraciones máxima de *1000* para garantizar la convergencia del modelo, ya que se tiene un *dataset* relativamente grande con muchas características y la vectorización TF-IDF genera muchas columnas: una columna por palabra diferente incluida en el diccionario creado.
- Regularización L2: se utiliza para evitar el sobreajuste del modelo, impidiendo que el modelo se focalice en unos parámetros concretos, perdiendo de esta manera su capacidad de generalización y disminuyendo su porcentaje de acierto en correos nuevos nunca vistos previamente por el modelo (datos de test). La regulación L2 permite paliar este peligro mediante la penalización de los coeficientes más grandes que, de aparecer, determinarían por sí solos el resultado proporcionado por el modelo. Por lo tanto, fuerzan al modelo a distribuir el peso entre más características en lugar de depender demasiado de unas pocas palabras. Además, es la opción por defecto al usar la biblioteca *scikit-learn*.

Durante el entrenamiento, el modelo recibe los correos de entrenamiento (40.259 correos electrónicos en total) como un vector de pesos TF-IDF. Para cada correo multiplica esos pesos por unos coeficientes iniciales y aplica la sigmoide, obteniendo la

probabilidad. Después se compara esa probabilidad con la etiqueta real del correo en concreto (*phishing/ no phishing*) y ajusta los coeficientes para reducir ese error. Este proceso se repite de forma iterativa hasta que los coeficientes convergen.

Tras el entrenamiento, el resultado final es una lista de coeficientes fijos, donde cada uno se asocia a una palabra del vocabulario creado; el número de coeficientes vendrá determinado por el número de palabras que aparecen tras la vectorización. Las palabras que aparecen frecuentemente, pero únicamente en correos de *phishing*, tendrán asociados unos coeficientes positivos altos, mientras que las que únicamente aparecen en correos legítimos tendrán coeficientes negativos. Al llegar un nuevo correo, el modelo utiliza los coeficientes obtenidos durante el entrenamiento para calcular la probabilidad en los nuevos correos.

5.2.4. Evaluación: métricas, curvas ROC, análisis del umbral óptimo

Se aplica el modelo con el umbral estándar (0,5) para poder analizar una primera aproximación de la efectividad del modelo. Los parámetros que determinan la eficiencia del modelo y los que se usarán para analizarlo serán los siete parámetros descritos a continuación.

- *Precision*: permite determinar el porcentaje de correos correctamente clasificados por el modelo dentro de una clase: *de todos los correos que el modelo ha determinado que son phishing, cuántos realmente lo son*. Se obtiene un porcentaje de *precision* por clase, donde el más relevante en este caso será la precisión en la clase de correos ilegítimos, ya que implica que, si el modelo determina que un correo es *phishing*, raramente se equivoca.
- *Recall:* mide el porcentaje de correos clasificados correctamente por el modelo dentro de una misma clase. El *recall* de la clase *phishing* es el más crítico del modelo, ya que un valor bajo implica que el modelo no detecta los correos fraudulentos y los clasifica como legítimos.
- *F1-score:* media armónica de las dos métricas anteriores: *precison* y *recall*. Es un parámetro útil que permite analizar la relación entre ambos, donde un valor de F1 alto implica que existe un equilibrio entre ambos parámetros y que el modelo no deja pasar demasiadas amenazas ni genera muchas falsas alarmas.
- *Support*: representa el número de correos reales de cada clase presentes en el conjunto de test, indicando el tamaño de cada clase sobre el que se calculan el resto de las métricas.
- *Accuracy*: porcentaje de correos clasificados correctamente en ambas clases. En este caso el *dataset* está relativamente balanceado, por lo que es una métrica representativa.

- *Macro average:* media aritmética no ponderada de cada métrica anterior entre las dos clases; no se tiene en cuenta el tamaño de cada clase.
- *Weighted average:* media ponderada de cada métrica anterior entre las dos clases; sí se tiene en cuenta el tamaño de cada clase. Al estar el *dataset* prácticamente equilibrado, los valores obtenidos son muy similares a los anteriores.

Para $umbral = 0{,}5$, se obtienen los siguientes resultados:

Clase / Métrica	Precision	Recall	F1-Score	Support
0	0.9900	0.9947	0.9923	4,759
1	0.9953	0.9910	0.9931	5,306
Accuracy	—	—	0.9927	10,065
Macro avg	0.9926	0.9929	0.9927	10,065
Weighted avg	0.9928	0.9927	0.9927	10,065

Tabla 1. Accuracy 99,27%. Clase legítimo (0): precision 99,00%, recall 99,47%, F1 99,23%, support 4.759. Clase phishing (1): precision 99,53%, recall 99,10%, F1 99,31%, support 5.306.

Destaca especialmente la precisión del 99,53% en la clase phishing, lo que indica que cuando el modelo alerta de un correo fraudulento, lo hace con una fiabilidad muy alta.

La matriz de confusión en la que se determina el número de aciertos y fallos separados por categoría es la siguiente:

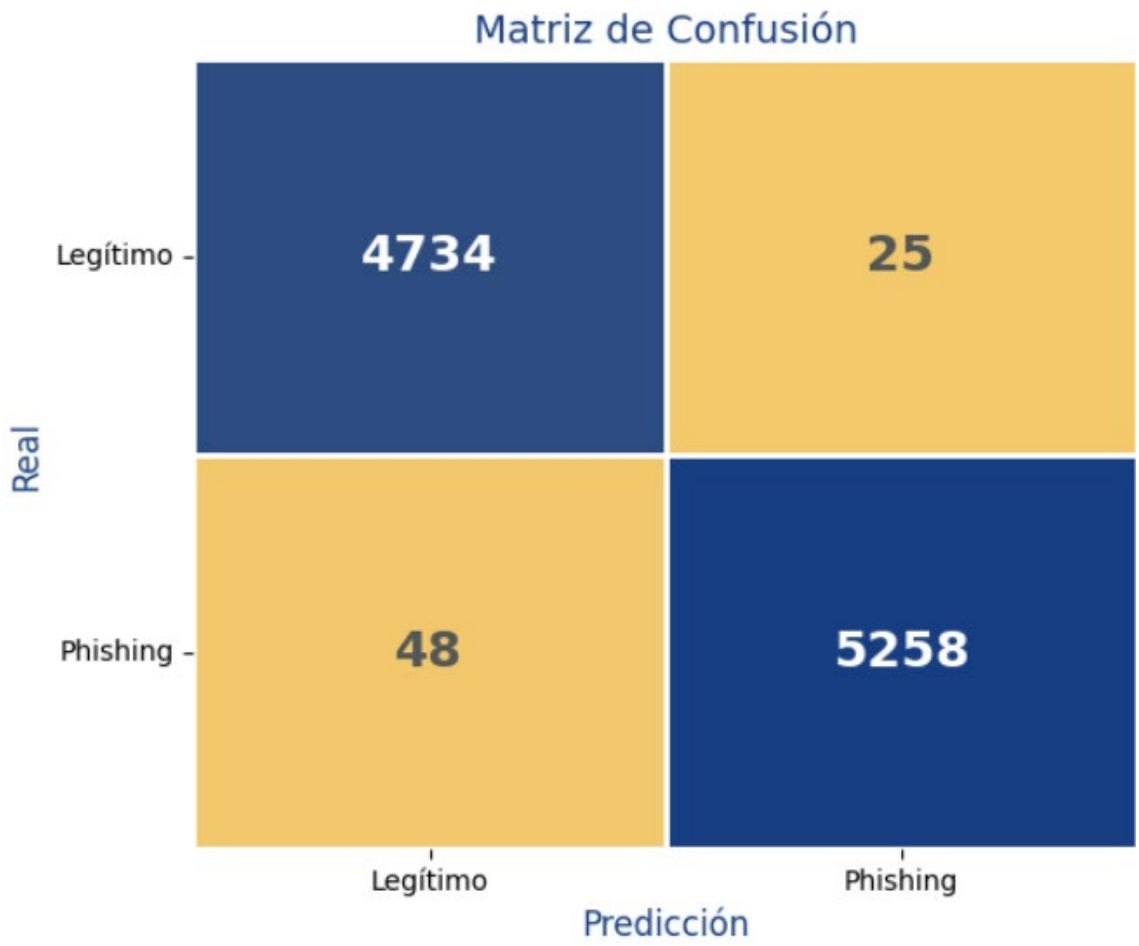

Figura 2. Matriz de confusión del modelo con umbral 0.5, donde se observan 4.734 verdaderos negativos (correos legítimos correctamente clasificados), 5.258 verdaderos positivos (correos de phishing correctamente clasificados), 25 falsos positivos y 48 falsos negativos.

El acierto en correos legítimos aparece en 4.734 correos, mientras que hay 25 falsos positivos, y el acierto en correos de phishing se da en 5.258 correos, mientras que hay 48 falsos negativos.

Se observa que los resultados anteriores son aceptables, pero el objetivo en este contexto es minimizar los 48 falsos negativos, ya que el mayor riesgo se asume cuando un correo que es *phishing* se categoriza como legítimo. Por lo tanto, se decide analizar cómo otros umbrales pueden afectar a estos parámetros, con el objetivo de determinar el umbral óptimo. De esta forma, se ha dibujado la siguiente gráfica, en la que se compara cómo varían la *precision* y el *recall* al variar el umbral del modelo:

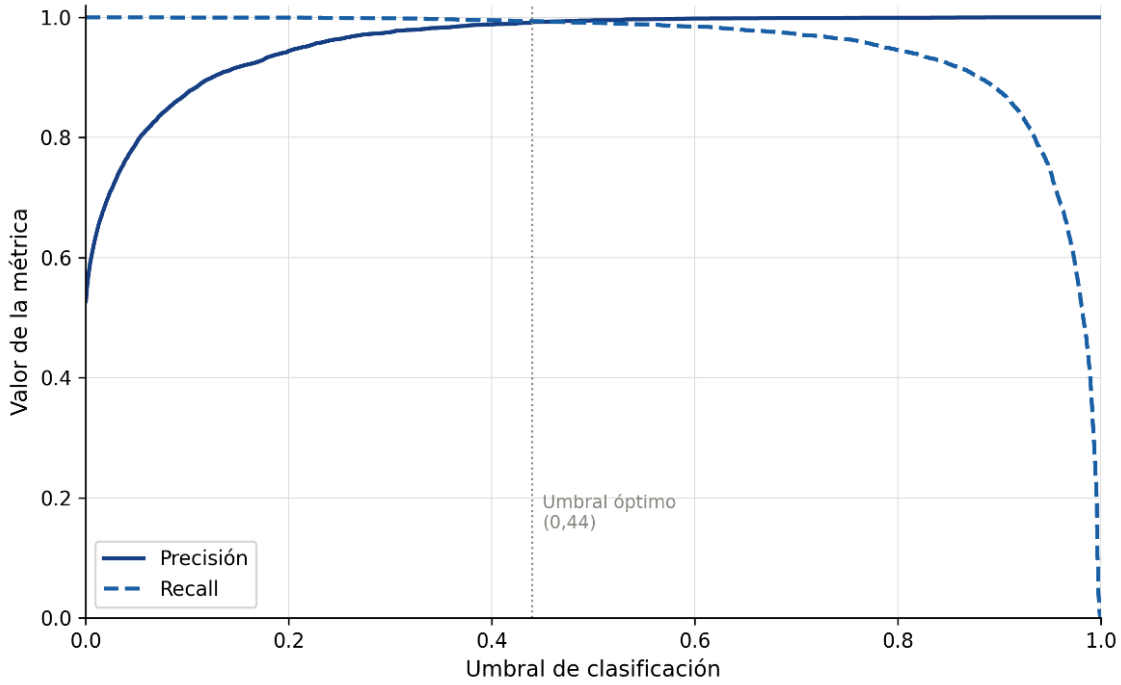

Figura 3. Curva Precision-Recall en función del umbral de clasificación, donde se observa el compromiso entre precisión y sensibilidad al variar el umbral entre 0 y 1.

En primer lugar, se observa como las curvas de precisión y *recall* tienen diferentes formas: para un umbral muy bajo, el modelo clasifica todos los correos como *phishing,* haciendo que la precisión disminuya hasta el 0,5 mientras que el *recall* es 1: todos los mensajes son clasificados como *phishing*, que representan el 0,52 de todo el *dataset*. Por otro lado, para un umbral muy alto, ninguno de los correos se consideran *phishing*, por lo que la precisión es prácticamente 1 mientras que el *recall* desciende a 0.

El objetivo para encontrar el umbral óptimo es encontrar un *threshold* que tenga una precisión alta sin permitir que el *recall* descienda demasiado. Se observa cómo el *recall* se mantiene hasta legar prácticamente al umbral 0,6, mientras que la precisión empieza a aumentar en el umbral 0,35, aproximadamente. Por estos motivos, se han analizado los resultados de aplicar varios umbrales diferentes, determinado como óptimo el umbral 0,44, valor muy similar al 0,5 predeterminado. Los nuevos parámetros para el umbral ajustado son:

Clase / Métrica	Precision	Recall	F1-Score	Support
0	0.9933	0.9914	0.9923	4,759
1	0.9923	0.9940	0.9931	5,306
Accuracy	—	—	0.9927	10,065
Macro avg	0.9928	0.9927	0.9927	10,065
Weighted avg	0.9927	0.9927	0.9927	10,065

Tabla 2. Accuracy idéntica 99,27%. Clase legítimo (0): precision 99,33%, recall 99,14%, F1 99,23%, support 4.759. Clase phishing (1): precision 99,23%, recall 99,40%, F1 99,31%, support 5.306.

En la Tabla 2 se observa que, en este caso, que la precisión ha aumentado para casos de *no phishing*, mientras que ha disminuido ligeramente con respecto a los casos de *phishing*. Por otro lado, ocurre al contrario con el *recall*, ya que aumenta en los casos legítimo y disminuye ligeramente en los casos ilegítimos. El resto de los parámetros se mantienen prácticamente idénticos. Además, se observa la mejora del modelo en la matriz de confusión:

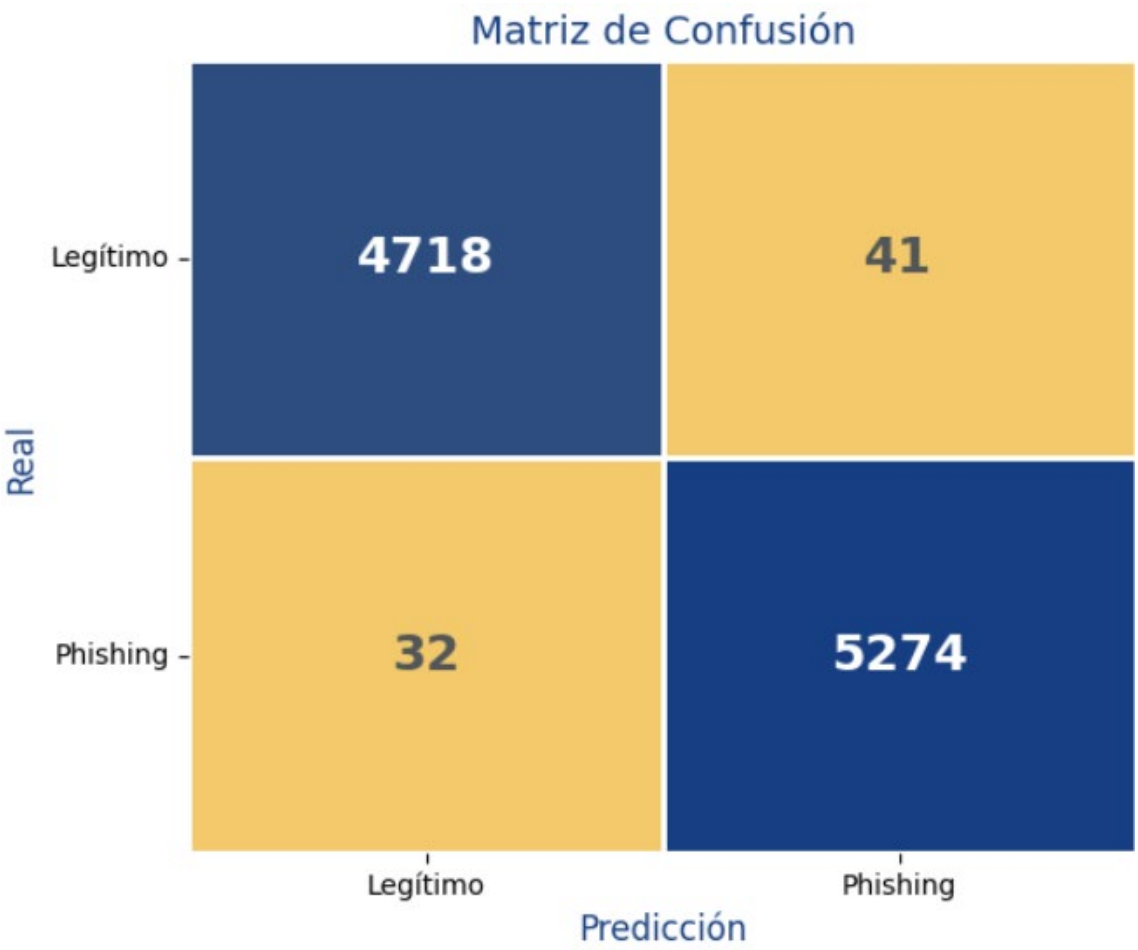

Figura 4. Matriz de confusión del modelo con umbral 0.4, donde se observan 4.718 verdaderos negativos (menos que con el umbral original), 5.274 verdaderos positivos (más que con el umbral 0,5), 41 falsos positivos y 32 falsos negativos.

Se observa que los falsos positivos han aumentado, pero los falsos negativos, que eran los que se pretende minimizar en este caso, han disminuido un tercio.

Los cambios no han sido muy significativos entre los dos modelos (modificando el umbral por defecto inicial por el umbral óptimo) y que la diferencia entre los correos clasificados correctamente representa la inmensa mayoría en comparación con los clasificados correctamente en ambas clases, por lo que se analiza la distribución de probabilidades de las decisiones tomadas por el modelo con los datos de test.

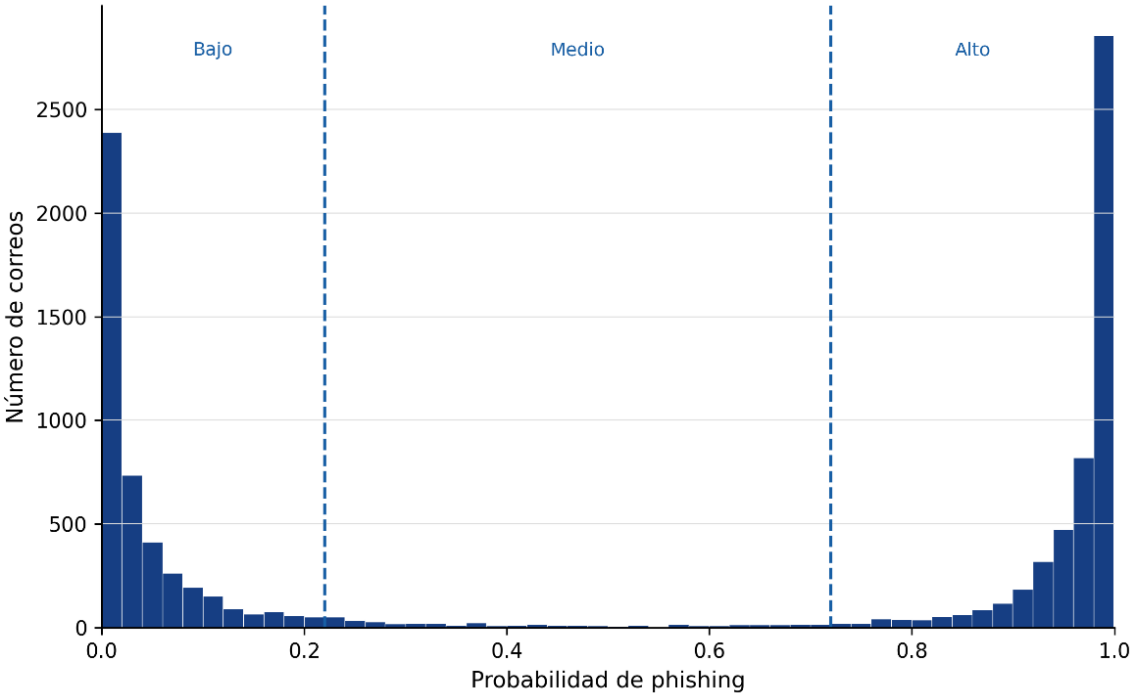

Figura 5. Distribución de probabilidades generadas por el modelo sobre el conjunto de test, donde se observan tres zonas diferenciadas: alta concentración por debajo de 0,2 (44,18% de los casos), zona intermedia entre 0,2 y 0,8 (5,94%) y alta concentración por encima de 0,8 (49,87%).

Se observa en la Figura 5 que el modelo está bastante seguro al diferenciar entre mensajes legítimos e ilegítimos. Al diferenciar tres zonas claras en esta gráfica, finalmente se determina optar por un modelo con tres franjas de decisión: bajo riesgo, riesgo medio y alto riesgo. También se analiza diseñar un modelo con cinco niveles, pero al tener casi el 97% de los datos entre [0-0,2] y [0,7-1] y apenas el 6% de los datos se encuentran en la franja de riesgo medio, determinando que estas divisiones adicionales no aportan valor significativo a la predicción. Finalmente, los umbrales seleccionados fueron 22% y 72% en cada caso, donde se eligió el límite inferior por ser ligeramente más conservador que la zona de baja densidad, y el límite superior con el que se garantiza que únicamente el 4% de los correos se encuentran en la franja de riesgo medio, ya que se busca que el modelo sea lo menos ambiguo posible y permita determinar, con una cierta incertidumbre, que el correo es o no legítimo. Estos niveles de riesgo se emplean para determinar el grado de certeza del modelo sobre la naturaleza de *phishing* del correo, así como el tipo de recomendaciones personalizadas que recibirá el usuario.

5.2.5. Versionado de modelos en Hugging Face Hub

Se han generado dos artefactos: *tfidf.joblib* y *phishing_model.joblib* en el *notebook*. El primero es un fichero binario que contiene el vectorizador TF-IDF

entrenado, es decir, el vocabulario completo de las palabras aprendidas por el modelo durante el entrenamiento y los pesos IDF asociados a cada una. El segundo es un fichero que contiene el clasificador de Regresión Logística entrenado, es decir, los coeficientes aprendidos.

Es esencial mantener el mismo vocabulario para que las palabras mantengan sus índices correspondientes y se mantengan las predicciones esperadas.

Ambos archivos ocupan un espacio considerable, por lo que no se suben al repositorio de Github, sino que se almacenan en el repositorio phishing-detection-model-v1 de Hugging Face Hub. Para recuperar los modelos completos se comprueba si los ficheros existen localmente al arrancar el microservicio FastAPI. De no ser así, se descargan desde Hugging Face Hub usando un *token* de autenticación, almacenado como variable de entorno en Railway y evitando así su exposición en el repositorio de código.

Una vez descargados, se cargan en memoria, quedando disponibles para las peticiones futuras.

La versión actual del modelo (v1) no contempla un proceso automatizado de reentrenamiento. En futuras implementaciones se plantea automatizar tanto el reentrenamiento del modelo como la subida de los nuevos artefactos al repositorio de Hugging Face Hub, de forma que el despliegue en producción se actualice sin intervención manual.

5.3. Servicio de inferencia (Python/FastAPI)

El servicio de inferencia es el componente encargado de ejecutar el modelo de ML, generar las recomendaciones personalizadas y consultar Tavily en caso de que el usuario haya proporcionado el remitente. Está desarrollado en Python con FastAPI, exponiéndose como un microservicio independiente del backend de Spring Boot, permitiendo actualizar o modificar cualquier componente del sistema sin afectar al resto.

5.3.1. Diseño de la API REST

La API REST tiene dos *endpoints*: *predict* y *recommend*. Se han diseñado dos diferentes para poder decidir cuándo llamar a cada uno. Por ejemplo, el *endpoint /recommend* consume tokens de Groq y de Tavily, por lo que no se hace ninguna petición en el caso de que el riesgo del correo sea bajo, únicamente se añaden recomendaciones estándar. Además, esta separación permite modificar cada módulo de forma independiente y sin modificar el otro, para poder hacer mejoras en futuras implementaciones.

El *endpoint /predict* recibe el texto del mensaje (*text*) y el asunto (*subject*) introducido por el usuario. La respuesta del modelo incluye la probabilidad (*probability*) entre 0 y 1 de que el mensaje sea *phishing* y las cinco palabras con un mayor coeficiente que han sido identificadas en el mensaje (*word_indicators*). Si los modelos no están cargados, se considera un error crítico del servidor: sin modelos la aplicación no funciona. Por lo tanto, no se genera un *fallback*.

Por otro lado, el *endpoint /recommend* recibe tanto el texto del mensaje como el asunto. También recibe la probabilidad de que el correo sea *phishing* calculada con */predict*. Además, de manera opcional, se puede incluir el remitente del mensaje y si el usuario tiene o no tiene cuenta en esa empresa.

La respuesta en este caso es una recomendación personalizada en formato texto en función de los datos introducidos. Además, se incluye una lista de fuentes consultadas por Tavily. Solo se hace la llamada a Tavily en el caso de que se incluya el remitente del correo ya que, de no incluirse, la llamada se hace únicamente a Groq, que genera recomendaciones estándar añadiendo un aviso para que el usuario considere añadir el remitente del correo para mejorar el análisis. Esto permite identificar si la empresa ha sufrido filtraciones o ataques recientes. Por lo tanto, la información variará en el caso de añadir o no el remitente del correo, ya que es el criterio utilizado actualmente para la búsqueda de información relevante. En implementaciones futuras, la conexión con Tavily se utilizará para obtener más información sobre la empresa, como datos de contacto o noticias concretas para informar y dar opciones al usuario sobre posibles decisiones.

5.3.2. Pipeline de análisis: de texto a predicción

En primer lugar, se envía el texto y el asunto desde el *frontend* hasta el *endpoint /predict*. A continuación, se concatena el texto del correo junto con el asunto para contar con un único *string*. Se ha decidido unir ambas secciones para formar un único texto que se procesa con el mismo modelo, evitando la necesidad de entrenar un modelo separado para el asunto.

Para vectorizar el mensaje, se aplica la función *transform* con el TF-IDF ya cargado en memoria, y se genera la matriz dispersa (*sparse*) – muchos valores del diccionario serán cero para la mayoría de los correos, por lo que en esta matriz se almacenan los valores TD-IDF distintos de cero, reduciendo el consumo de memoria de forma significativa.

Tras la vectorización del cuerpo y del asunto, el modelo calcula la probabilidad de que el mensaje introducido sea *phishing* mediante la función *predict_proba*. También se extraen las cinco palabras indicadoras con mayores coeficientes y se construye la respuesta final que se envía al *frontend*.

La función *detectar_urgencia* en este *endpoint* se plantea como una implementación futura que podría incluirse tras un análisis más exhaustivo de los datos iniciales para poder enriquecer el análisis de la predicción final.

5.3.3. Integración con Groq (LLaMA-3.3-70B): prompt engineering y generación de recomendaciones personalizadas

Dentro del *endpoint /recommend* en primer lugar se detecta la posible urgencia del correo mediante patrones *regex*, donde se ha generado una lista con palabras que indican urgencia tanto en inglés como en español. Dado que los patrones de urgencia son similares en ambos idiomas, se ha definido una única lista de expresiones regulares que cubre los términos más comunes en inglés y español. Este análisis podría hacerse con otra capa del LLM (Groq en este caso), aunque se ha decidido hacer de esta forma para no consumir tantos tokens con la versión actual de Groq y Tavily. Sin embargo, se plantea como una mejora futura del proyecto.

A continuación, se hace una construcción dinámica del *prompt* de Groq, incorporando datos como la probabilidad, el nivel de riesgo asignado, si tiene cuenta, la urgencia y el contexto de Tavily (parámetro opcional). Es importante adaptar el *prompt* a los valores concretos de cada correo electrónico, y añadir la capa LLM como un complemento del análisis ML realizado con el modelo entrenado. Además, se ha adaptado el tono de la respuesta (recomendaciones) en función del nivel del riesgo: más tranquilizador para riesgo bajo y más firme para riesgo alto, aunque en ningún caso con tono alarmista. La función y el *prompt* completo se incluyen en el Anexo II.

Finalmente, una vez se ha construido el *prompt*, se hace la llamada a Groq con LLaMA-3.3-70b con los siguientes parámetros.

- *Temperatura*: parámetro que controla la aleatoriedad de las respuestas proporcionadas por el modelo. El rango va de 0 a 1, siendo 0 la respuesta más probable y predecible, y 1 las respuestas más variadas. Para mantener un cierto rigor y estructura se selecciona *temperatura=0.4*.
- *Max_tokens*: parámetro que limita el número de tokens que el modelo puede emplear por respuesta. Siendo la equivalencia, aproximadamente, un token por una palabra, el valor máximo elegido han sido 350. Se busca que las 2 ó 3 recomendaciones proporcionadas por el modelo sean concisas.

5.3.4. Integración con Tavily: búsqueda de filtraciones por dominio

La búsqueda con Tavily se activa solo si el usuario proporciona remitente. En primer lugar, se extrae el dominio del remitente con la función *extraer_dominio*.

A continuación, se construye una *query* en la que se construye el nombre del dominio junto con los términos como '*data breach*', '*phishing attack*' o '*cybersecurity incident*'. Las búsquedas se han limitado a resultados posteriores a *2020*. Se ha seleccionado la búsqueda básica, que permite obtener tres resultados por cada consulta, por considerarse suficiente para el alcance actual del proyecto; se alcanza un compromiso entre respuestas obtenidas y tokens consumidos.

Para mostrar el resultado al usuario, se concatenan un resumen de 300 caracteres por cada uno de los tres resultados y se extraen las *URLs* como fuentes. Este texto formado por los tres resúmenes se incorpora al *prompt* de Groq y las fuentes se muestran al usuario.

Si hubiera algún fallo en la llamada a Tavily o se devolviera un mensaje vacío, la llamada a Groq se haría de igual manera, pero sin contexto de filtraciones.

5.4. Backend (Java/Spring Boot)

El *backend* de Java actúa como orquestador: recibe las peticiones del *frontend* y llama a FastAPI de forma interna. El esquema global del *backend* es el siguiente:

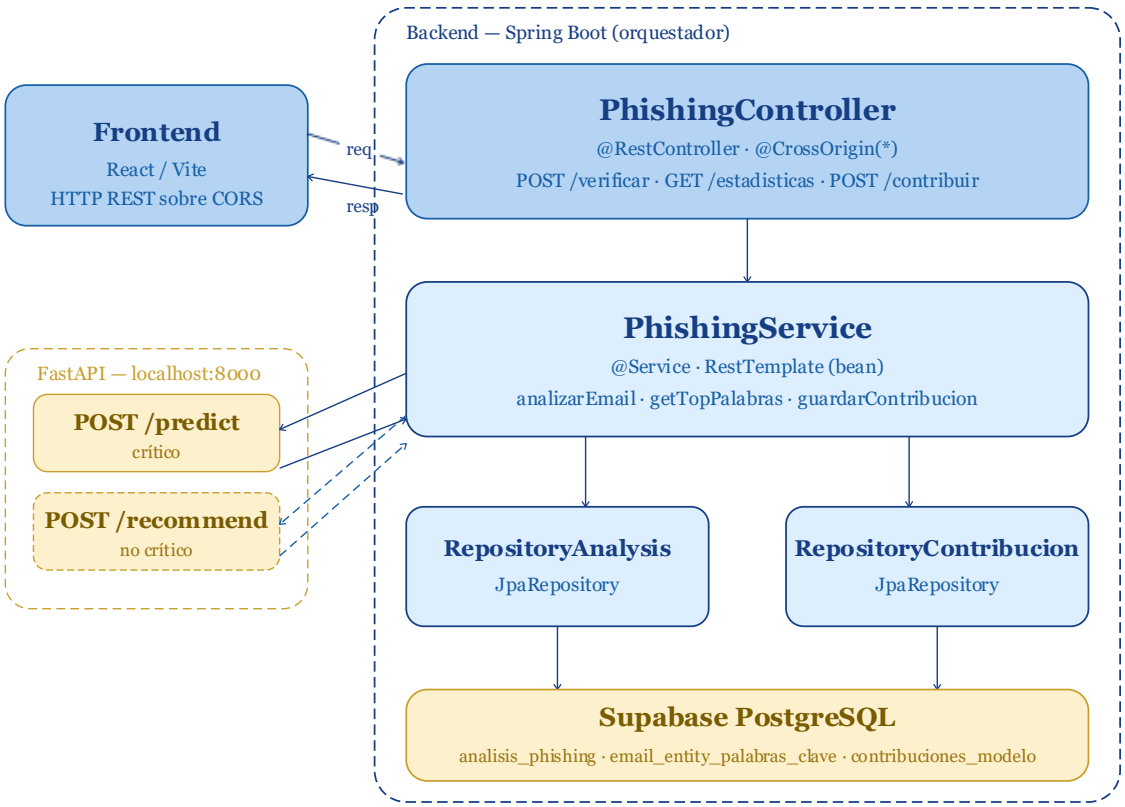

Figura 6. Arquitectura global del backend. Spring Boot actúa como orquestador centralizado entre el frontend y los servicios internos: FastAPI (detección de phishing y recomendación) y Supabase PostgreSQL (persistencia). El frontend no accede directamente a ninguno de los dos.

5.4.1. Arquitectura MVC y diseño de la API REST

Se ha elegido implementar una arquitectura Modelo-Vista-Controlador (MVC) porque organiza la lógica y responsabilidad del *backend* en tres capas: el Controlador recibe la petición HTTP desde el usuario y la envía al servicio correspondiente; el Servicio tiene la lógica y se encarga de realizar las funciones correspondientes haciendo peticiones al Repositorio, que es el que tiene acceso a la base de datos. Este modelo es un estándar en la industria porque permite una buena mantenibilidad y escalabilidad del código: permite limitar los cambios en el código en el caso de modificar la API o la base de datos. Además, facilita realizar pruebas de manera independiente.

Además, el diseño de la API Rest cuenta con tres *endpoints*: */api/verificar*, /api/*estadisticas* y */api/contribuir*.

/api/verificar es esencial en el funcionamiento de la aplicación, es el *endpoint* principal, ya que es el punto de entrada de toda la funcionalidad principal de la aplicación. Cuando el *frontend* envía la petición, el *backend* llama a FastAPI */predict* para obtener la probabilidad y las palabras clave del modelo ML. A continuación, guarda ese resultado en la base de datos utilizando JPA y después llama a FastAPI de nuevo con */recommend* para obtener la recomendación personalizada para el usuario. Finalmente, devuelve al *frontend* un objeto con toda la información: la probabilidad, las palabras clave, la recomendación y las fuentes consultadas por Tavily. Por ello, es un *endpoint* crítico sin el que la aplicación perdería su funcionalidad por completo: la aplicación no podría analizar ningún correo.

Por otro lado, el *endpoint /api/estadisticas* se emplea para poder acceder a las estadísticas generadas por el *backend* recopilando toda la información devuelta por el modelo. En concreto, se muestran las diez palabras que aparecen con mayor frecuencia en los mensajes enviados por los usuarios. Es un apartado interesante pero no esencial, donde se muestra una parte de análisis global a los usuarios con aportaciones de todos los correos y con actualizaciones en tiempo real, sin embargo.

Finalmente, el último *endpoint* es también opcional, ya que solo se llama en el caso de que el usuario haya dado su consentimiento expreso para que el mensaje que ha introducido pueda ser utilizado en un futuro para entrenar al modelo de ML.

5.4.2. Comunicación con el servicio ML

Para comunicarse con el microservicio FastAPI, el *backend* Java utiliza *RestTemplate*, el cliente HTTP síncrono de Spring Boot. Al ser síncrono y bloqueante, el *backend* espera la respuesta de FastAPI antes de continuar con el flujo. La URL base del microservicio se almacena como variable de configuración en *application.properties*, evitando *hardcodearla* en el código y facilitando futuros cambios de despliegue.

Por otro lado, para el flujo de *analizarEmail(),* se llama a */predict* y, en el caso de que falle se lanza un error 503 y se corta el flujo. Este fallo sería crítico porque incluye la lógica principal de la aplicación, aunque en el *frontend* existe un *fallback* en este caso, en el que se muestra un porcentaje aproximado. Tras la consulta, se guarda la probabilidad junto a las palabras en las tablas correspondientes de la base de datos. Finalmente, se llama a */recommend* y, si algo falla, se devuelve el resultado sin recomendación personalizada.

5.4.3. Gestión de la base de datos

Spring Boot interactúa con PostgreSQL mediante JPA/Hibernate (Java Persistence API), que es la interfaz que extienden los repositorios en Spring Boot. Proporciona las operaciones básicas de base de datos sin necesidad de escribir SQL (buscar, guardar, eliminar, etc.), todo de manera automática, donde solo se tiene que definir la interfaz y Spring Boot genera la implementación. Una de las principales ventajas de usar ORM (*Object-Relational Mapping*) es que permite trabajar con clases Java en lugar de con tablas. Para hacer esto se definen clases de entidad como *EmailEntity* con los atributos que luego se quieren como filas de la tabla. Hibernate hace la traducción automática de ese código Java a formato SQL, rellenando las filas de la base de datos. Otro de los beneficios es que se reducen los posibles errores comunes y hace el código más legible.

Además, se implementa DDL (*Data Definition Language*) con *ddl-auto=update*: se compara de manera automática si hay algún cambio entre las clases Java y las tablas de PostgreSQL y se hace la modificación de manera automática. Esta configuración es especialmente relevante en este proyecto porque al estar en continuo desarrollo, el esquema puede cambiar y, utilizando Hibernate estos cambios se ven reflejados en PostgreSQL de manera automática, facilitando modificaciones y evitando la intervención manual.

Existe una *query* personalizada para las estadísticas, ya que se necesita hacer el recuento de la frecuencia de las palabras, por lo que no puede hacerse de manera automática, sino que requiere de una lógica extra que haga este cálculo. Se ha redactado con una *query* JPQL, pudiendo escribir en términos de clases de Java en lugar de tablas. Esta *query* devuelve las diez palabras más frecuentes entre todos los análisis almacenados, mostradas en el panel de estadísticas de la aplicación.

Utilizar JPA/Hibernate y JPQL facilita una posible migración a otro motor de base de datos y mejora la legibilidad del código.

5.5. Base de datos (PostgreSQL/Supabase)

Se ha planteado un diseño de una base de datos relacional, ya que este tipo de estructuras permiten organizar la información en tablas independientes relacionadas entre sí, evitando duplicar datos y garantizando la integridad de la información.

5.5.1. Diseño de la base de datos

En la base de datos hay un total de tres tablas: analisis_phishing, contribuciones_modelo y email_entity_palabras_clave.

La tabla *analisis_phishing* almacena cada análisis realizado por el usuario, guardando el asunto, el texto y la probabilidad asignada por el modelo. La recomendación personalizada por Groq y las fuentes consultadas por Tavily son *@Transient,* lo que implica que no se almacenan en la base de datos ya que solo se envían en la respuesta JSON al *frontend,* generándose en cada análisis. El campo *id* es de tipo *BIGSERIAL* para mantener la escalabilidad de la aplicación, ya que es la clave primaria (*primary key*) de la tabla y no puede repetirse ni ser un valor nulo. Además, se ha añadido *auto-increment*, por lo que la numeración de los identificadores la gestiona la base de datos automáticamente.

Campo	Tipo	Restricciones
id PK	BIGSERIAL	Auto-increment, NOT NULL
asunto	VARCHAR	Opcional
texto	TEXT	Opcional
probabilidad	DOUBLE PRECISION	Opcional

Tabla 3. tabla analisis_phishing, que registra cada análisis realizado por un usuario, almacenando el texto del correo, el asunto y la probabilidad asignada por el modelo.

La tabla *contribuciones_modelo* almacena los mensajes introducidos que los usuarios han decidido ceder para el futuro reentrenamiento del modelo. Únicamente se

almacena el correo en el caso de que el usuario dé su consentimiento explícito a través del *endpoint /api/contribuir*. Con respecto a la tabla anterior, en esta se añade el campo *remitente*, que también es opcional, y el campo *fecha*, que se guarda automáticamente en el momento del registro y que no se puede modificar, ya que tiene la característica *updatable=false*. Este último campo no puede estar vacío. El identificador *id* de esta tabla es independiente del identificador de la tabla *analisis_phishing*, ya que esta tabla es independiente del resto para poder analizarla en un futuro para el reentrenamiento del modelo.

Campo	Tipo	Restricciones
id PK	BIGSERIAL	Auto-increment, NOT NULL
asunto	VARCHAR	Opcional
texto	TEXT	Opcional
remitente	VARCHAR	Opcional
probabilidad	DOUBLE PRECISION	Opcional
fecha	TIMESTAMP	NOT NULL, auto-generado

Tabla 4. Tabla que almacena los correos que los usuarios han cedido voluntariamente para el futuro reentrenamiento del modelo. Incluye el campo fecha generado automáticamente en el momento del registro.

La tabla *email_entity_palabras_clave* es una tabla auxiliar generada automáticamente por hibernate mediante *@ElementCollection*. Las cinco palabras almacenadas en esta tabla se consultan cuando el usuario accede a la pestaña *Estadísticas*, donde el *frontend* realiza una petición al *backend* para recuperar las palabras más frecuentes de todos los análisis almacenados hasta ese momento. De esta forma, las estadísticas reflejan siempre el estado actual de la base de datos, actualizándose con cada nuevo análisis realizado. El identificador *email_entity_id* es una clave foránea (*foreign key*) del identificador del correo electrónico, repitiéndose cinco veces el mismo *id* del mensaje, una por cada palabra, para mantener la relación entre cada palabra y su correo correspondiente. Es un campo obligatorio porque las palabras clave no pueden almacenarse sin estar asociadas a un análisis.

Campo	Tipo	Restricciones
email_entity_id FK	BIGINT	FK → analisis_phishing.id, NOT NULL
palabras_clave	VARCHAR	Opcional

Tabla 5. Tabla que almacena las palabras clave identificadas por el modelo para cada análisis, relacionadas con su correo correspondiente mediante la clave foránea email_entity_id.

5.5.2. Modelo entidad-relación

El diseño de la base de datos se ha realizado con las tres tablas ya mencionadas. En lugar de almacenar todas las palabras clave de un análisis en la misma fila que el correo, se ha creado una tabla auxiliar vinculada mediante una clave foránea, que relaciona un mismo mensaje con cinco palabras clave: si un análisis tiene cinco palabras clave, solo se almacena una vez y se crean cinco filas en la tabla auxiliar, cada una referenciando el mismo análisis. Este enfoque reduce el espacio de almacenamiento y facilita las consultas agregadas como el recuento de frecuencias para las estadísticas.

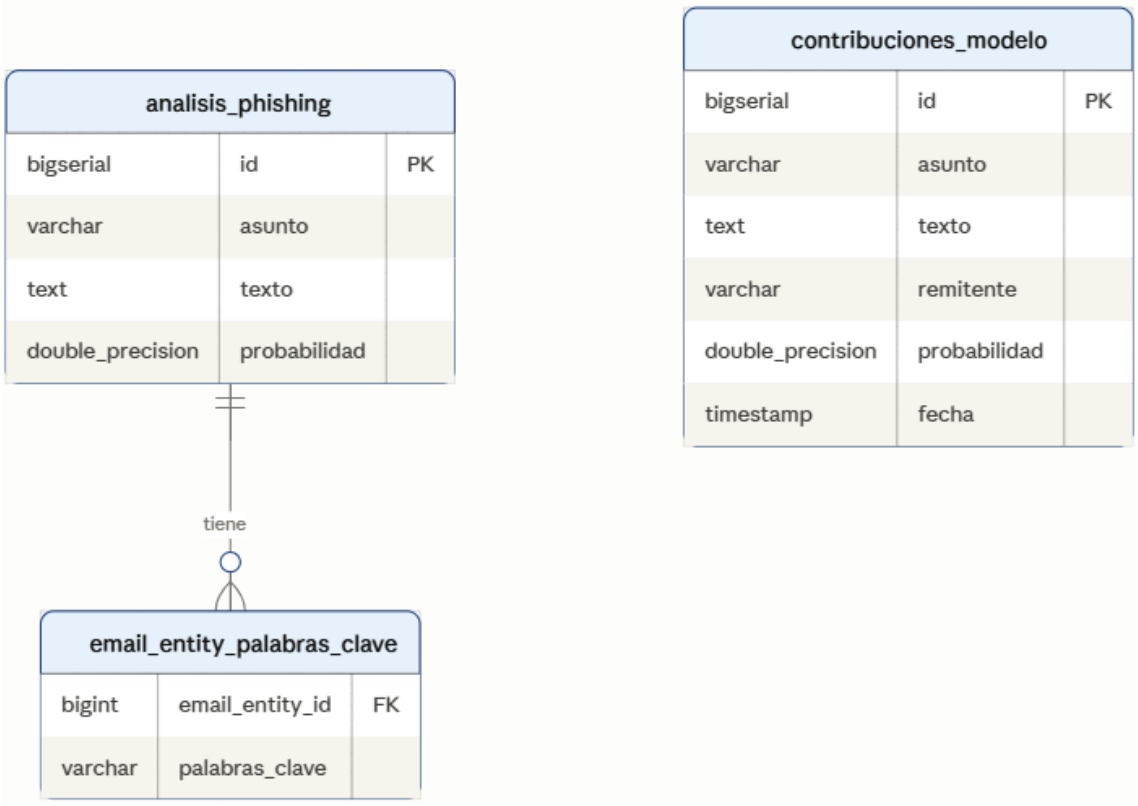

Figura 7. Diagrama Entidad-Relación de la base de datos, mostrando la relación 1:N entre analisis_phishing y email_entity_palabras_clave mediante la clave foránea email_entity_id, y la tabla independiente contribuciones_modelo.

El diagrama muestra las tres tablas de la base de datos y sus relaciones. La tabla *analisis_phishing* mantiene una relación uno a muchos (1:N) con *email_entity_palabras_clave*, ya que un análisis puede tener asociadas hasta cinco palabras clave, pero cada palabra pertenece a un único análisis, identificado mediante la clave foránea *email_entity_id*. La tabla *contribuciones_modelo* no tiene relación con las otras dos, ya que registra correos cedidos voluntariamente por los usuarios de forma independiente al flujo principal de análisis.

5.5.3. Seguridad de la Base de Datos

Por motivos de seguridad, se ha activado *Row Level Security* (RLS) en todas las tablas de *Supabase*. RLS es un mecanismo de seguridad de PostgreSQL que permite controlar el acceso a los datos a nivel de fila, es decir, al contenido y datos recopilados en la tabla. Sin esta restricción, cualquier persona con acceso a la base de datos podría leer, escribir o modificar cualquier dato de la tabla.

La base de datos se encuentra desplegada en *Supabase*, que expone una API REST pública que permite interactuar con la base de datos desde el navegador, no solo desde el *backend*. En el plan seleccionado en *Supabase*, las tablas son accesibles públicamente a través de esa API y activando RLS se bloquea este acceso.

En este caso se ha activado RLS sin definir políticas estrictas, bloqueando el acceso de los roles *anon* (usuario anónimo) y *authenticated* (usuario autenticado) de *Supabase*. Al hacer la conexión mediante el *Spring Boot* con el rol *postgres* (dueño de la base de datos) se puede seguir accediendo a todas las tablas sin restricción y sin realizar ninguna modificación en el código. Este es el único tipo de usuario que puede modificar, leer y escribir en la base de datos.

Se ha implementado ese sistema de seguridad como recomendación de la propia aplicación de *Supabase.* En futuras implementaciones se definirán políticas RLS explícitas para determinar qué roles tienen acceso a lectura y cuáles a escritura en cada tabla. Esta mejora es especialmente relevante al añadir autenticación de usuarios o si se expone la API de Supabase directamente al *frontend*, por ejemplo, si en el futuro el *frontend* consultara directamente a Supabase para mostrar el historial de análisis de un usuario autenticado. Este tipo de peticiones reducen la latencia, al evitar pasar por *Spring Boot*, y simplifican la arquitectura.

5.6. Frontend (React/Vite)

El frontend de la aplicación se ha desarrollado con React y Vite, siguiendo una arquitectura de componentes que organiza la interfaz en cuatro vistas principales. React permite construir interfaces reactivas mediante componentes reutilizables, donde el estado de cada componente determina lo que se muestra al usuario en cada momento; se actualiza la interfaz de forma automática sin recargar la página, lo que ahorra recursos y latencia. Vite actúa como *bundler*, empaquetando todos los ficheros en un conjunto optimizado. También funciona como servidor de desarrollo, proporcionando tiempos de compilación muy rápidos y actualizando la aplicación en el navegador tras los cambios de forma sencilla durante el desarrollo local.

La aplicación cuenta con diseños diferentes para escritorio y para dispositivos móviles, haciendo uso de clases de *Tailwind CSS* para poder adaptar la organización (*layout*) a una columna en móvil y dos columnas en escritorio. En la versión móvil se sustituye la barra de navegación superior por una inferior fija.

5.6.1. Arquitectura de componentes

En la Figura 8 se muestra la arquitectura de componentes del *frontend*. *App.jsx* es el componente raíz de la aplicación. Gestiona la navegación mediante un estado *view* con un renderizado condicional, que varía en función de dónde hace clic el usuario: *Inicio*, *Analizar*, *Estadísticas* y *Recomendaciones*. Se ha elegido este sistema de navegación en lugar de utilizar React Router por la simplicidad de la página web, no siendo necesaria una *URL* diferente en cada página con la funcionalidad actual. El diseño actual simplifica el código y las dependencias.

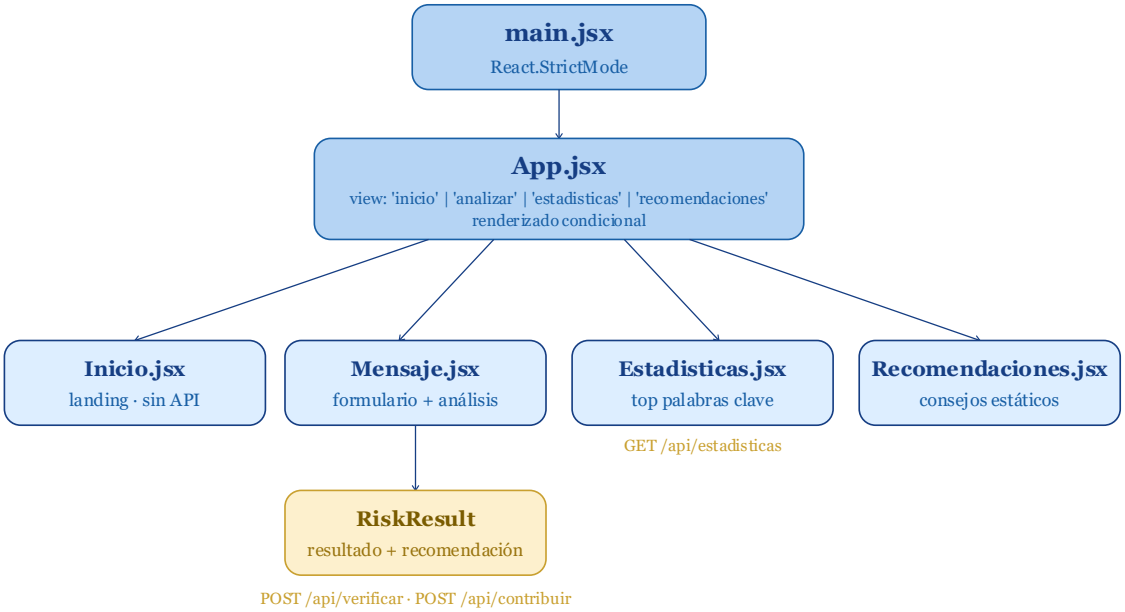

Figura 8. Diagrama de componentes del frontend React, mostrando App.jsx como componente raíz con los cuatro componentes de página gestionados mediante el estado view con renderizado condicional.

El estado de cada componente se gestiona de manera local con *useState*, ya que es la forma de realizar esta gestión de la forma más simple posible: no existe ningún estado que necesite compartirse entre componentes, por lo que *useState* es suficiente para el alcance actual.

Cada petición *fetch* está definida dentro de su propio componente (*inline*), por lo que se lanzan las peticiones al *backend* de manera aislada; no existe una capa de servicios centralizada. Las peticiones HTTP se realizan mediante *fetch* nativo del navegador, sin dependencias externas, lo que reduce el número de bibliotecas del proyecto. La *URL* del *backend* se configura mediante la variable de entorno VITE_API_URL que en desarrollo corresponde con un proxy de Vite y en producción es la *URL* absoluta de Railway.

Existen componentes *legacy* que no se encuentran activos en la versión actual, que se conservan para poder realizar la implementación de la página web bilingüe (español e inglés).

5.6.2. Flujo de usuario: análisis de un correo

El usuario entra en la aplicación sin necesidad de hacer ningún tipo de registro. En la pestaña '*Inicio*', mostrada en la Figura 9, aparece una breve explicación de *qué es el phishing* y de cómo funciona la herramienta en términos generales.

Figura 9. Captura de pantalla del inicio de PhishGuard en el navegador de un ordenador.

En la sección de la derecha aparecen tres opciones: analizar un correo, ver las estadísticas globales y ver algunas recomendaciones generales. En la parte inferior, aparece un aviso en el que se indica al usuario que el modelo actual se ha entrenado con mensajes en inglés, por lo que se recomienda traducir los mensajes que se quieran analizar a este idioma, haciendo que el análisis sea lo más veraz posible. También se observa un aviso legal en el que se informa al usuario de que los resultados y las recomendaciones proporcionadas han sido creadas con el objetivo de ser herramientas informativas y ser un filtro más para determinar la legitimidad de un mensaje, y que en ningún momento se puede considerar como asesoramiento legal y siempre debe primar el criterio del propio usuario, quien actúa bajo su propio riesgo.

Haciendo clic en '*Analizar*', aparece el formulario para analizar el mensaje seleccionado. El único campo obligatorio es el cuerpo del mensaje, ya que es esencial para poder hacer el análisis predictivo. El botón para analizar el mensaje se encuentra deshabilitado hasta que el usuario completa esta sección. Por otro lado, los campos de asunto y remitente son opcionales, aunque añadir este último puede mejorar las recomendaciones, pudiendo analizar si la empresa que envía dicho mensaje ha sufrido filtraciones recientemente.

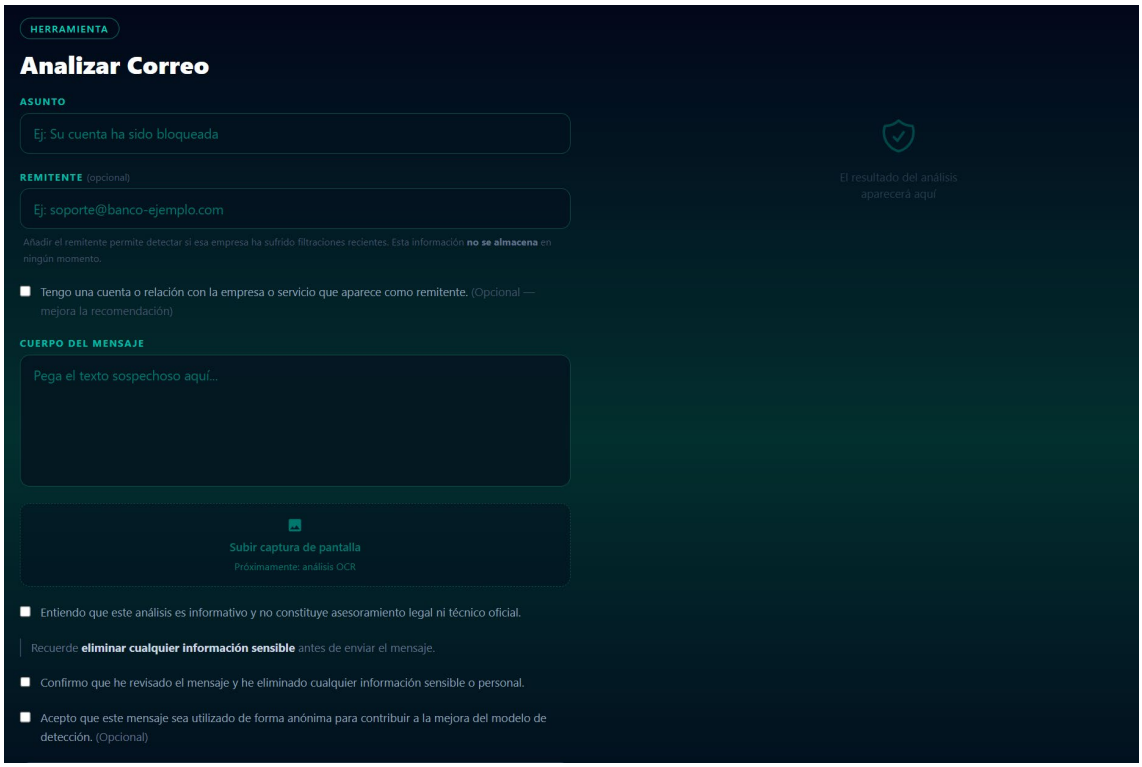

Figura 10. Captura de pantalla de la página de análisis de mensajes de escritorio, incluyendo los diferentes campos y la sección para la futura implementación de 'Subir captura de pantalla' mediante OCR.

Además, existen dos *checkboxes* que son obligatorios. El primero se asegura de que el usuario entiende que el contenido del análisis no es asesoramiento oficial, sino que su fin es meramente informativo. El segundo se asegura de que el usuario haya eliminado cualquier información sensible o personal. Finalmente, existe un *checkbox* opcional para informar si el usuario tiene o no cuenta con la empresa que envía el mensaje, cuyo fin es mejorar la recomendación personalizada, y otro que permite al usuario aportar la información introducida de forma opcional para futuras mejoras del modelo predictivo. En el caso de la versión móvil, el aviso sobre la recomendación de idioma del mensaje aparece al abrir por primera vez la pantalla *Análisis*, como se muestra en la Figura 11.

Figura 11. Captura de pantalla de la página del aviso de recomendación de idioma en la página del análisis de móvil.

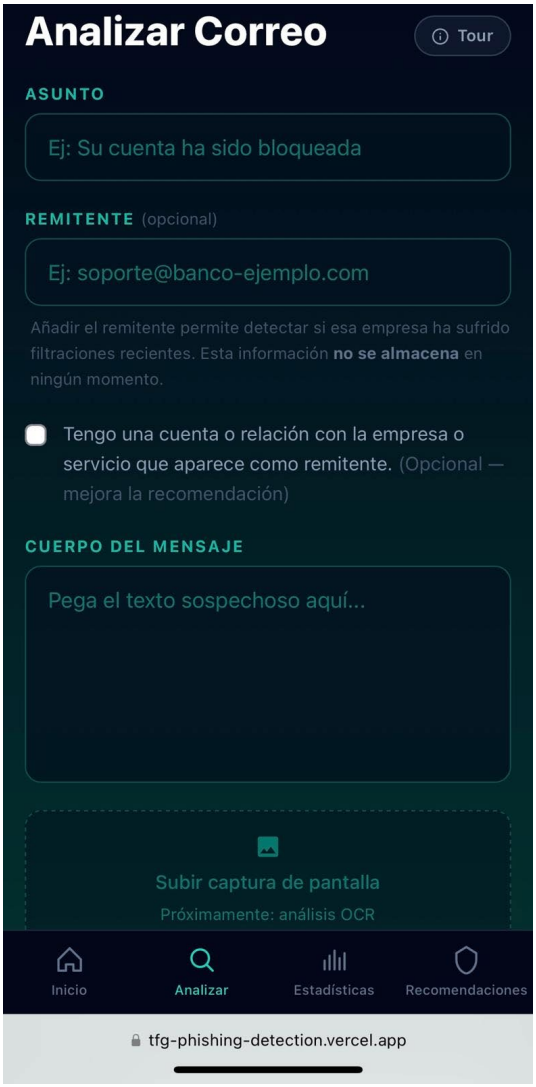

Figura 12. Captura de pantalla del formulario de análisis de mensajes de móvil.

En la parte inferior (en ambas versiones), aparece de nuevo un aviso indicando que los resultados obtenidos no son vinculantes e indicando al usuario que no haga clic en enlaces, aunque el riesgo mostrado sea bajo.

Tras analizar el mensaje enviado, el resultado se muestra a la derecha. En la parte superior se muestra el porcentaje estimado por el modelo y, debajo, aparecen las recomendaciones personalizadas en función de los parámetros introducidos.

En el caso de que el riesgo sea alto, se informa al usuario de que hay patrones que pueden implicar que el mensaje sea *phishing* y, en función de los parámetros introducidos, modifica estas recomendaciones. En el ejemplo mostrado en la Figura 13 se ha analizado un mensaje fraudulento que suplanta la identidad de una plataforma de reservas, solicitando un pago urgente para no perder la reserva [24]. Además, al incluir el remitente de la empresa se ha encontrado una filtración de datos reciente, por lo que la recomendación se ha adaptado a esta información. En la parte inferior aparecen las fuentes consultadas para que el usuario pueda verificar toda la información proporcionada.

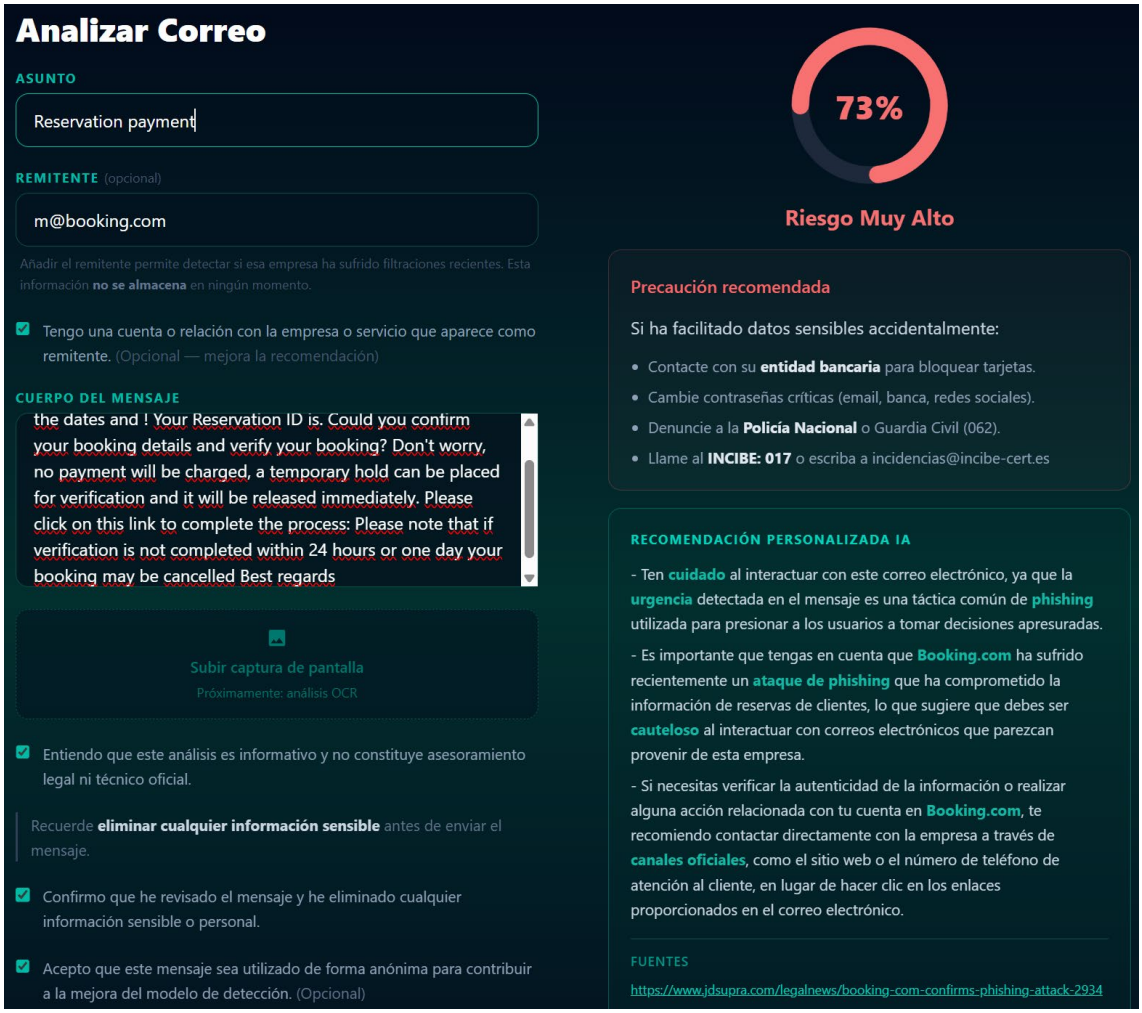

Figura 13. Captura de pantalla de un análisis de un mensaje ilegítimo, con un 73% de riesgo de ser phishing y recomendaciones personalizadas que hacen referencia a la función de detección de urgencia y a una filtración reciente (consulta Tavily), así como las fuentes empleadas.

En el caso de que el riesgo sea bajo, se informa al usuario de que no se han encontrado patrones típicos de fraude. Además, la recomendación personalizada tranquiliza al usuario, pero manteniendo las recomendaciones generales, como que, en caso de duda, es mejor tratar de verificar la información por medios oficiales.

Figura 14. Captura de pantalla de un resultado de un mensaje legítimo con riesgo bajo de ser phishing, incluyendo el porcentaje (20%) y las recomendaciones personalizadas.

En el caso de que el servidor no esté disponible, la aplicación aplica un mecanismo de *fallback offline*: detecta palabras clave en el texto introducido por el usuario y asigna una probabilidad aproximada según los patrones encontrados, buscando términos de urgencia o nombres de entidades financieras. El resultado se muestra igualmente al usuario acompañado de un aviso explícito indicando que el modelo no está disponible y que la estimación mostrada es orientativa, garantizando así la transparencia con el usuario en todo momento. El flujo de usuario para analizar un correo se muestra al completo en el siguiente esquema:

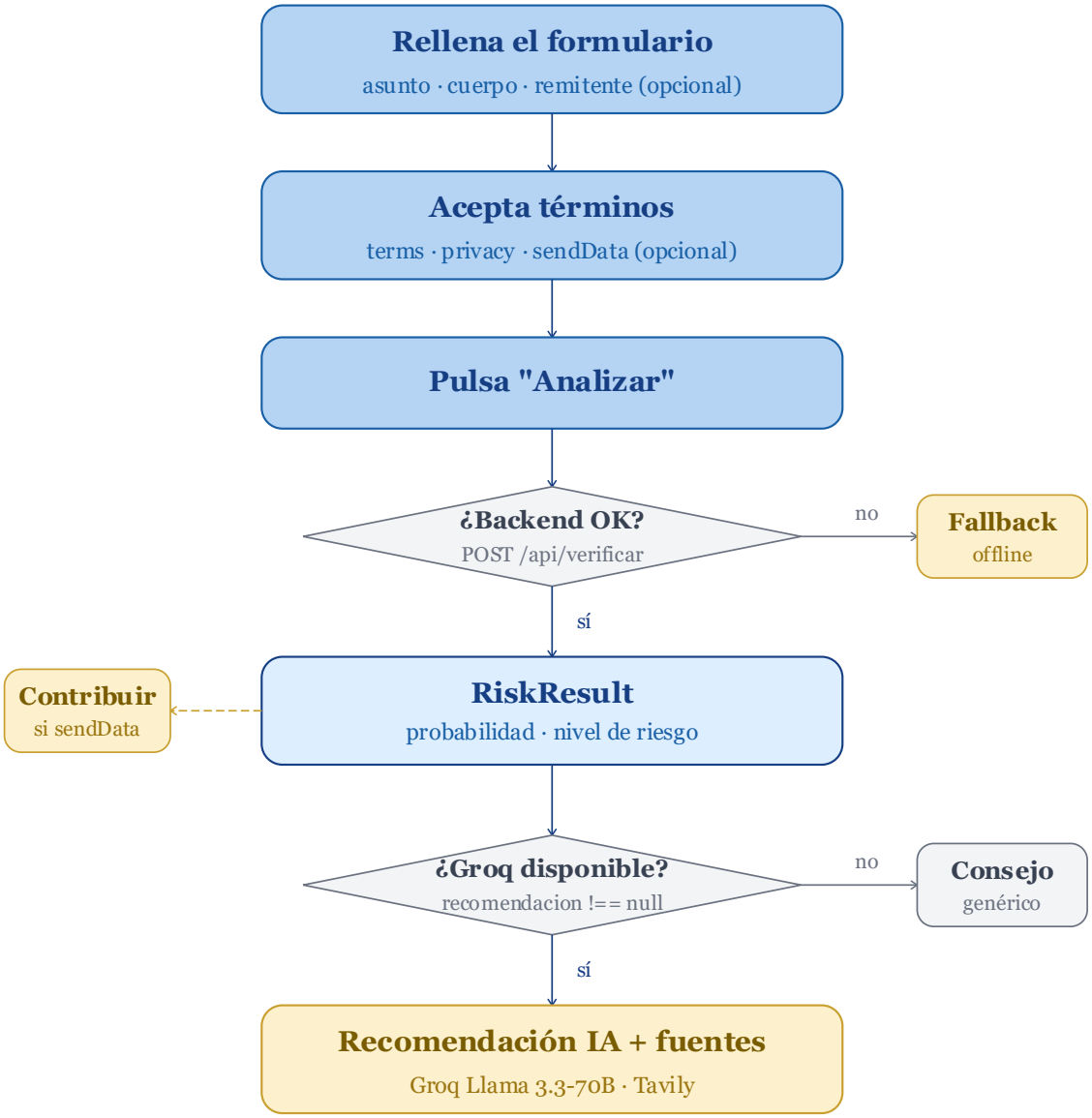

Figura 15. Diagrama de flujo del proceso de análisis de un mensaje en PhishGuard, desde la introducción del mensaje hasta la obtención del resultado, incluyendo el mecanismo de fallback offline.

Además, se ha añadido un *tour* interactivo que está disponible en las dos versiones: en el escritorio se muestra en la barra de navegación superior, mientras que en la versión móvil aparece como un botón flotante en la esquina superior derecha de la pantalla. El *tour* guía al usuario por las cuatro secciones principales de la aplicación: la página de inicio, el formulario de análisis, el panel de estadísticas y las recomendaciones; facilitando la familiarización del usuario con la herramienta. Incluye tanto navegación

por teclado como mediante botones. Se ha incluido mediante el uso de *useState* y *useEffect*, sin liberías externas.

5.6.3. Panel de estadísticas

En la sección de '*Estadísticas*', se muestra un ranking con las cinco palabras que más aparecen en los correos analizados, listando las palabras con mayor peso en la decisión del modelo. Además, se muestra un diagrama de burbujas en la derecha de forma visual mostrando las 10 palabras que aparecen con mayor frecuencia. Dado que el modelo está entrenado en inglés y se anima a que el usuario introduzca mensajes traducidos a inglés, la mayoría de las palabras mostradas estarán en este idioma.

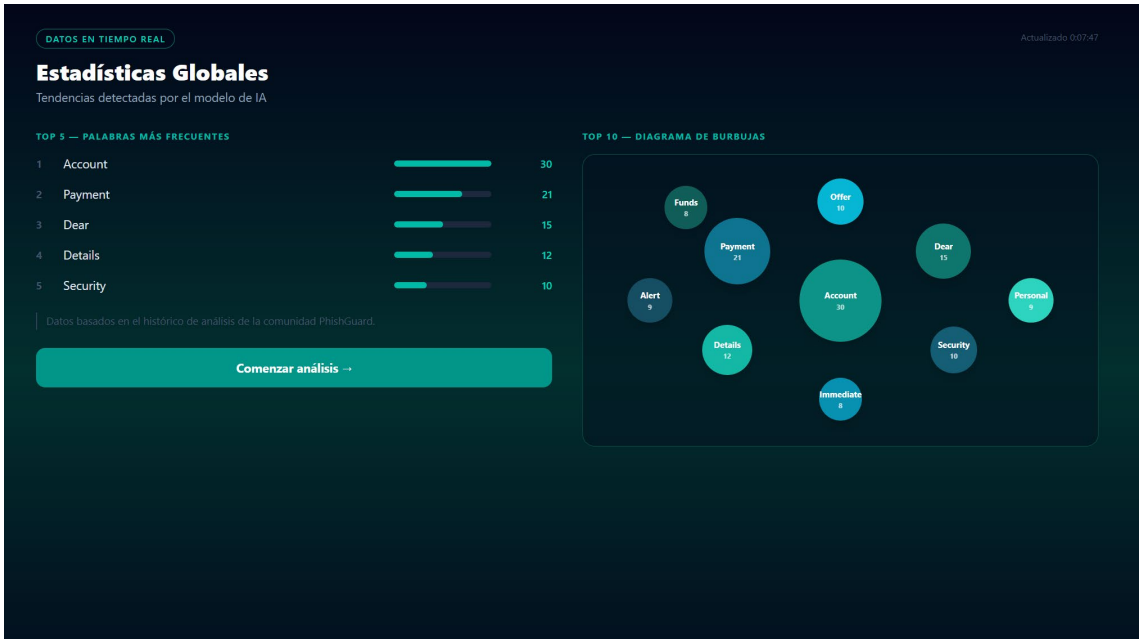

Figura 16. Panel de estadísticas globales mostrando el ranking de las cinco palabras más frecuentes y el diagrama de burbujas con las diez palabras con mayor peso acumulado en los análisis realizados por los usuarios.

Arriba a la derecha aparece la última fecha de actualización de los datos disponibles. Esta fecha se actualiza con la hora local del navegador en el momento en el que el *frontend* recibe los datos de *backend*, actualizándose de forma automática cada 5 minutos.

5.6.4. Módulo de orientación práctica y legal

Se ha añadido un breve módulo de orientación práctica y legal, donde se incluyen buenas prácticas en la prevención, comprobación y actuación ante posibles mensajes de *phishing*. El contenido es orientación legal basada en el marco normativo europeo vigente

y en los organismos competentes en España. Se ha optado por un formato visual de tarjetas para facilitar la lectura y que el contenido sea accesible para usuarios de cualquier perfil tecnológico. Las explicaciones se han mantenido concisas y completas, empleando un lenguaje directo para poder maximizar la usabilidad y utilidad para los usuarios. Se ha dividido el contenido en tres pestañas diferentes.

La primera sección informa al usuario de los derechos legales que tiene con respecto a sus datos personales, recogidos en el Reglamento General de Protección de Datos (RGPD). También se incluyen cuatro consejos de prevención, como activar la verificación en dos pasos siempre que sea posible, y no repetir contraseñas.

Figura 17. Captura de pantalla del módulo de recomendaciones y orientación legal de 'Prevención', incluyendo cuatro consejos concretos y aplicables.

En la segunda sección se explica el contexto del PSD2 (*Payment Services Directive* o Directiva de Servicios de Pago) y algunos consejos generales para poder comprobar si un mensaje puede ser fraudulento. Algunos consejos concretos son verificar siempre el remitente, desconfiar de mensajes que impliquen urgencia y no hacer clic en enlaces sospechosos. También se aconseja acceder a los sitios mediante sus páginas web oficiales de manera directa, en lugar de hacerlo a través de enlaces.

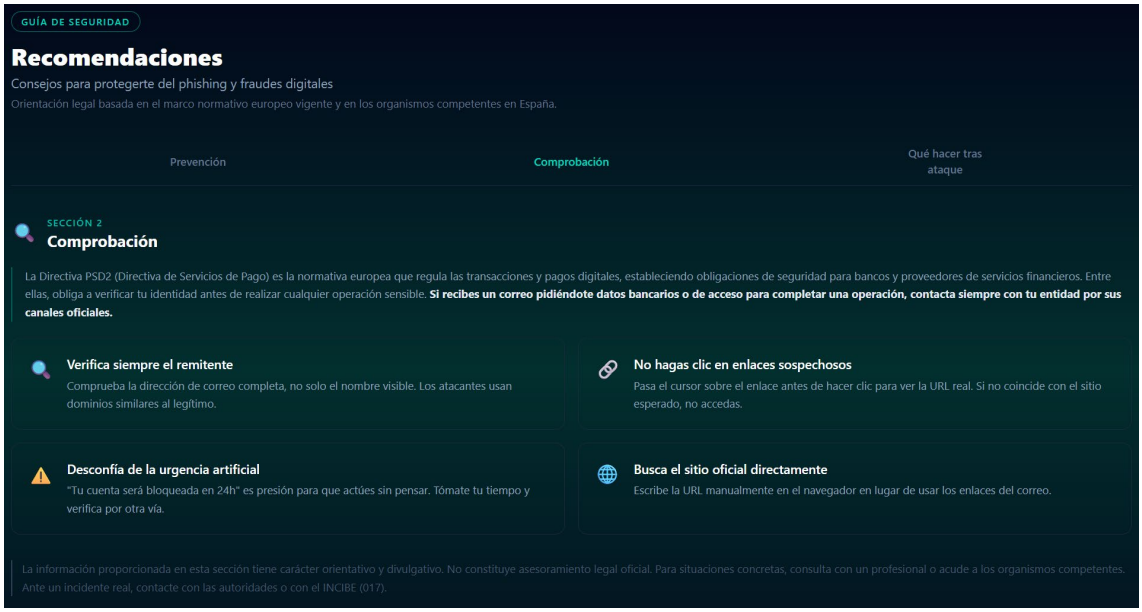

Figura 18. Captura de pantalla del módulo de recomendaciones y orientación legal de 'Comprobación', incluyendo cuatro consejos concretos y aplicables.

En la tercera y última sección, se informa al usuario cómo puede actuar en el caso de haber sido víctima de un ataque en España, ya que los organismos mencionados únicamente tienen competencia en este país. Algunas recomendaciones que se le dan al usuario son cambiar las contraseñas lo antes posible, incluyendo otros servicios en los que se use la misma contraseña; contactar con el banco para tratar de bloquear la tarjeta y para consultar posibles opciones de reclamación (si aplica); denunciar el incidente y contactar con el INCIBE a través de su línea de ayuda gratuita 017.

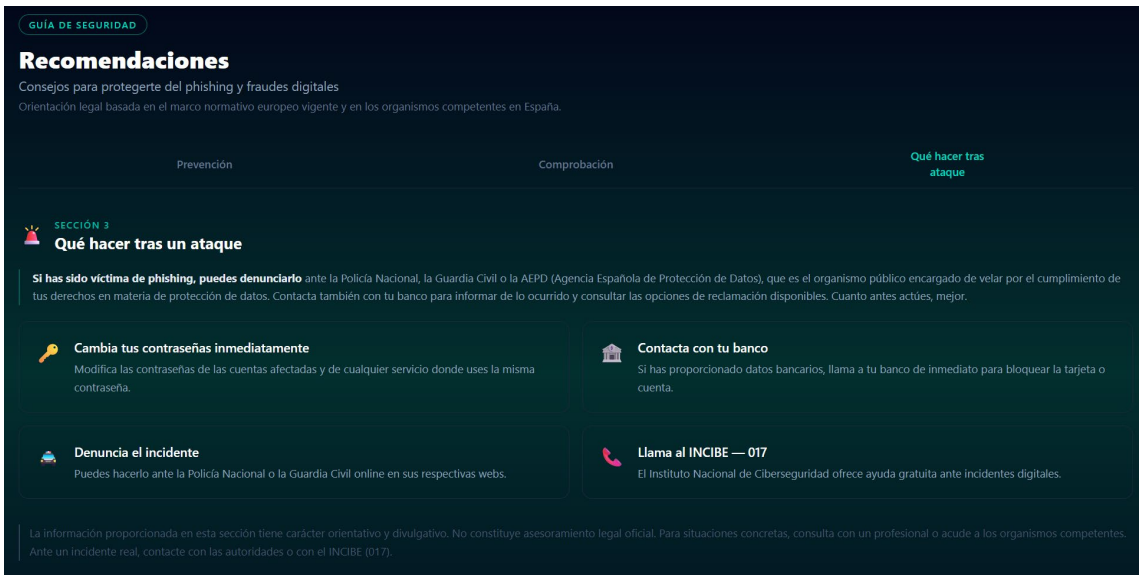

Figura 19. Captura de pantalla del módulo de recomendaciones y orientación legal de 'Qué hacer tras un ataque', incluyendo cuatro consejos concretos y aplicables.

En la parte inferior de la página se indica que las recomendaciones que aparecen en la página web no constituyen asesoramiento legal y que el objetivo principal es únicamente orientativo y divulgativo.

5.7. Despliegue e infraestructura cloud

El modelo de despliegue se ha realizado haciendo uso de tres servicios principales: Vercel para el *frontend*, Railway para el *backend* y ML y Supabase para la base de datos (BD). Es un despliegue completo en la nube, por lo que no tiene infraestructura propia. El esquema completo del despliegue en producción es el que se muestra en la Figura 20:

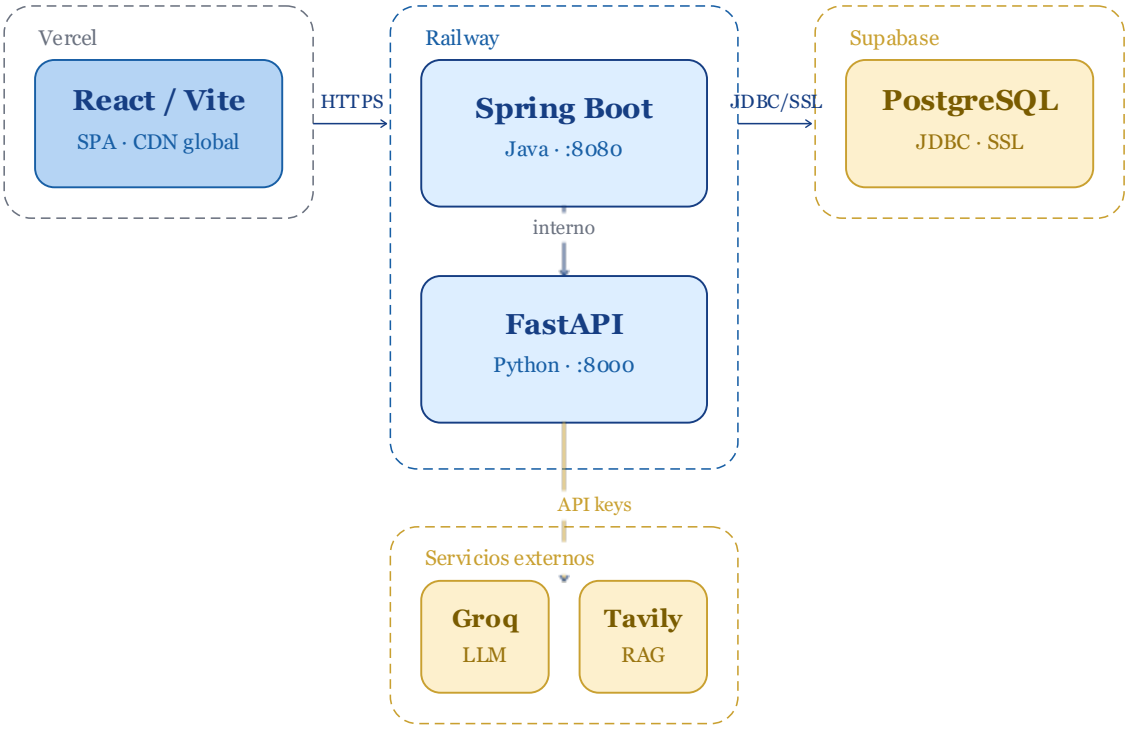

Figura 20. Diagrama de arquitectura del sistema en producción, mostrando los tres servicios cloud y sus conexiones.

El *frontend,* desarrollado en React/Vite, está desplegado en Vercel con la URL https://tfg-phishing-detection.vercel.app/. La variable de entorno utilizada para poder acceder al *backend* es VITE_API_URL.

Variable	Descripción	Desarrollo	Producción
VITE_API_URL	URL base del backend Java	Vacía — proxy Vite activo	URL pública de Railway

Tabla 6. Variable de entorno del frontend desplegado en Vercel, con su valor en desarrollo local y en producción.

El *backend* con Spring Boot y el ML con FastAPI están desplegados en el mismo proyecto en Railway en dos servicios separados, como se observa en la Figura 21. Spring Boot está desplegado en el puerto 8080, permitiendo peticiones HTTPS desde Vercel, mientras que FastAPI está en el puerto 8000.

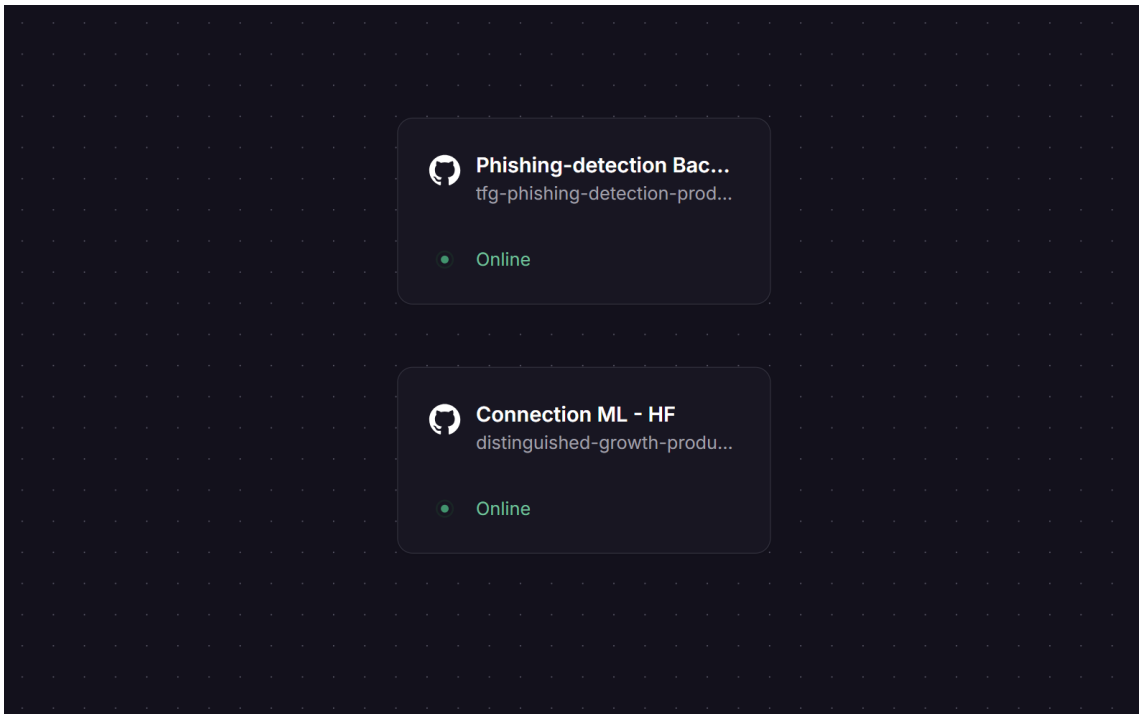

Figura 21. Dashboard de Railway con los dos servicios desplegados: Spring Boot en el puerto 8080 y FastAPI en el puerto 8000.

Variable	Descripción	Obligatoria
SUPABASE_DB_URL	URL de conexión JDBC a Supabase (con SSL)	Sí
DB_USERNAME	Usuario de la base de datos PostgreSQL	Sí
SUPABASE_DB_PASSWORD	Contraseña de la base de datos	Sí
ML_SERVICE_URL	URL interna del servicio FastAPI en Railway	Sí
PORT	Puerto del servidor	No — 8080 por defecto

Tabla 7. Variables de entorno del backend Java desplegado en Railway, todas obligatorias para el correcto funcionamiento del servicio en producción.

El *backend* se conecta y gestiona la conexión con los servicios externos Groq (LLM) y Tavily (RAG) mediante API *keys* almacenadas como variables de entorno en Railway para mayor seguridad, sin exponer las variables de forma pública en el repositorio. Por otro lado, Spring Boot se conecta con Supabase mediante Java Database Connectivity (JDBC) y con seguridad *Secure Sockets Layer* (SSL), ya que es el estándar de Java para conectarse a bases de datos relacionales. Para configurar la seguridad es necesario añadir ?*sslmode=require* al final de la URL de conexión JDBC en las variables de entorno. El resto de la gestión la realiza Supabase automáticamente.

Variable	Descripción	Comportamiento si no está definida
GROQ_API_KEY	Clave de acceso a la API de Groq (LLM)	/recommend devuelve HTTP 500
TAVILY_API_KEY	Clave de acceso a la API de Tavily	Búsqueda omitida — Groq sigue funcionando sin contexto web
HF_TOKEN	Token de autenticación en Hugging Face Hub	Descarga falla si el repositorio es privado

Tabla 8. Variables de entorno del servicio ML desplegado en Railway, con el comportamiento del sistema en caso de que alguna no esté definida.

Durante el proceso de desarrollo, la base de datos corría en Docker en una máquina virtual, simulando un servidor físico externo al resto del proyecto. Para producción se decidió migrar a Supabase para no depender de nada en local y que el despliegue se hiciera por completo en la nube. Para poder mantener la dirección dinámica y válida en ambas situaciones, el proxy de Vite redirige /api/* al *backend* local durante el desarrollo.

Cross-Origin Resource Sharing (CORS) es un mecanismo de seguridad del navegador que controla qué dominios pueden hacer peticiones a un servicio. Está configurado en FastAPI con *allow_origins*, restringiendo las peticiones a los dominios permitidos: el frontend en producción y localhost en desarrollo. Spring Boot no requiere configuración CORS explícita porque el frontend nunca lo llama directamente.

Actualmente se cuenta con las versiones gratuitas de todos los servicios utilizados, donde el límite de Railway es por horas de ejecución mensuales (con un crédito disponible para consumir de cinco dólares), Vercel es gratuito y no tiene ningún límite para el caso de uso, y Supabase tiene límite de almacenamiento y conexiones que son suficientes para el proyecto.

Capítulo 6. Análisis de Resultados

En esta sección se presentan los resultados del modelo de aprendizaje automático, evaluando las métricas obtenidas. Además, se estudian las respuestas ofrecidas por el modelo de manera cuantitativa y se analizan las recomendaciones personalizadas devueltas por el LLM en los casos de uso reales. Para que el análisis sea más riguroso se utilizan tanto mensajes de *phishing* como mensajes legítimos de las empresas o entidades suplantadas en los mensajes fraudulentos. Por último, se realiza una comparativa con los resultados de otras soluciones disponibles en el mercado.

6.1. Evaluación del modelo ML

A continuación, se muestra un análisis del modelo en producción con el fin de estudiar cómo se comporta en el uso real de la aplicación web. El flujo de análisis se inicia cuando el usuario introduce el texto del correo en el formulario de la sección Analizar y pulsa el botón de análisis. El *frontend* envía la petición al *backend* de Spring Boot, que llama al microservicio FastAPI: primero al *endpoint /predict* para obtener la probabilidad y las palabras clave identificadas por el modelo, y posteriormente a */recommend* para generar la recomendación personalizada. El resultado se muestra con el porcentaje estimado, el nivel de riesgo correspondiente y las recomendaciones personalizadas. El flujo completo de datos entre los distintos componentes del sistema se muestra en la Figura 22.

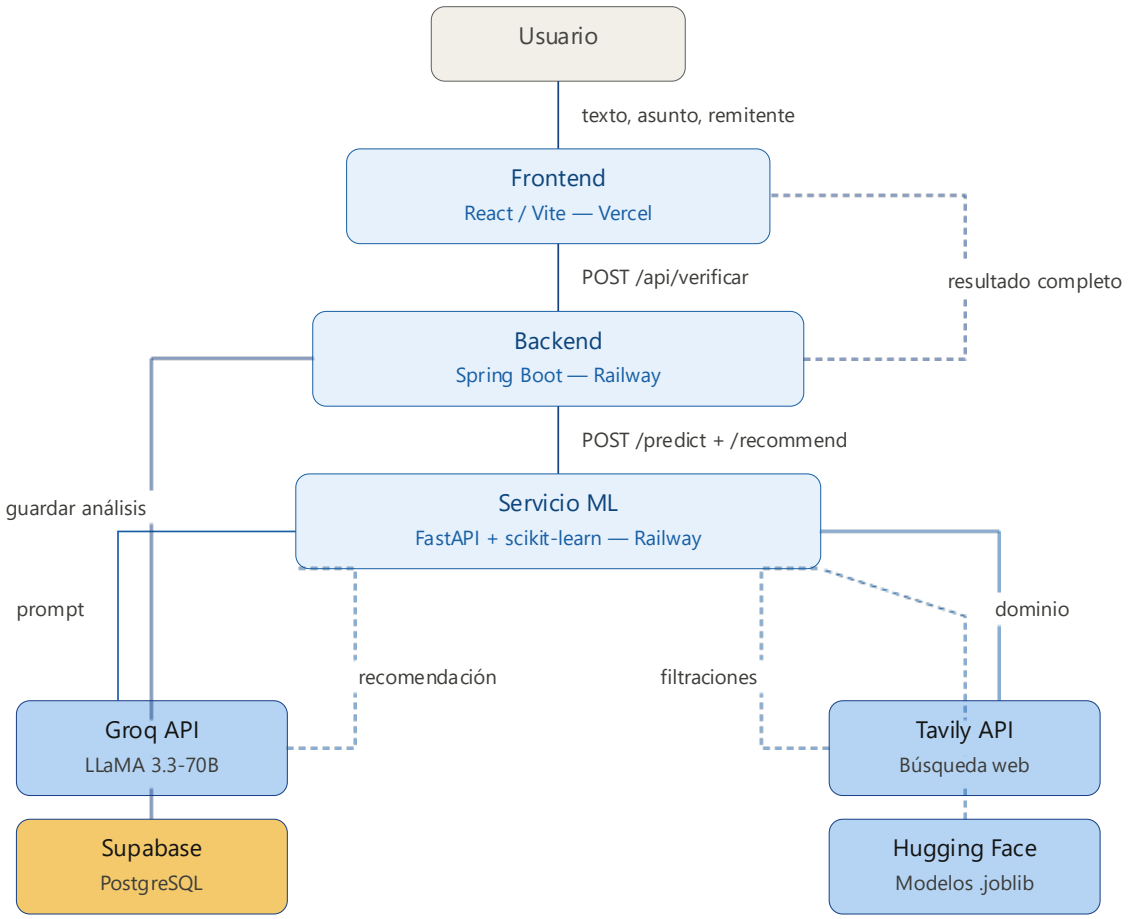

Figura 22. Diagrama de flujo de datos del sistema

Desde el punto de vista del usuario, en primer lugar, se accede a la pantalla de inicio de la aplicación mostrado en la Figura 23. A continuación, selecciona la pestaña "Análisis" o hace clic en el botón "Comenzar análisis →" para acceder al formulario de la aplicación.

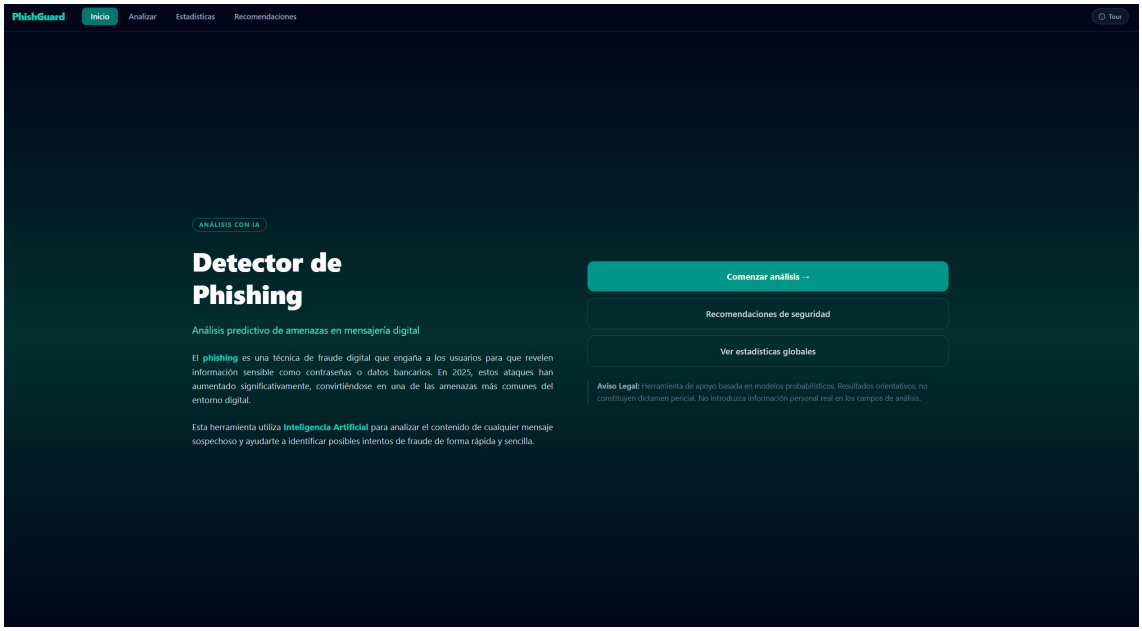

Figura 23. Página de onboarding del usuario de PhishGuard.

Al abrir esa pantalla, aparece un aviso sobre el idioma, recomendando al usuario traducir el mensaje antes de introducirlo en la herramienta para conseguir la máxima fiabilidad posible.

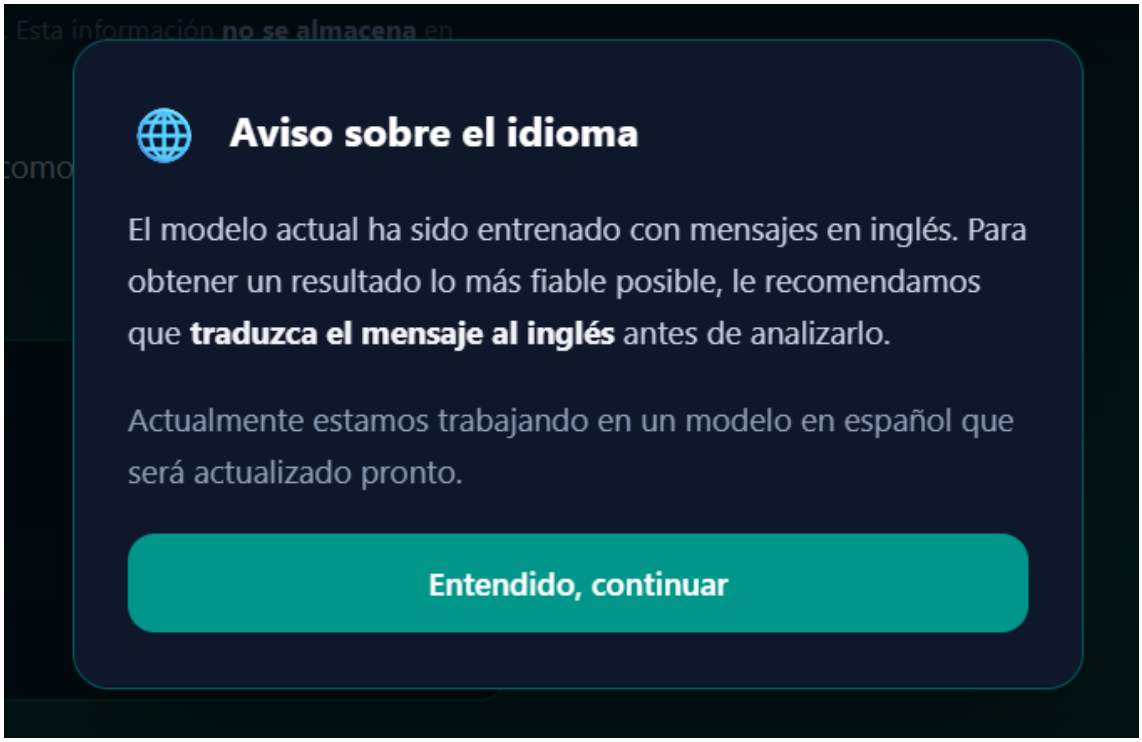

Figura 24. Aviso de traducción para el análisis.

A continuación, el usuario introduce todos los datos correspondientes al correo o mensaje que desea analizar, como se observa en la Figura 25.

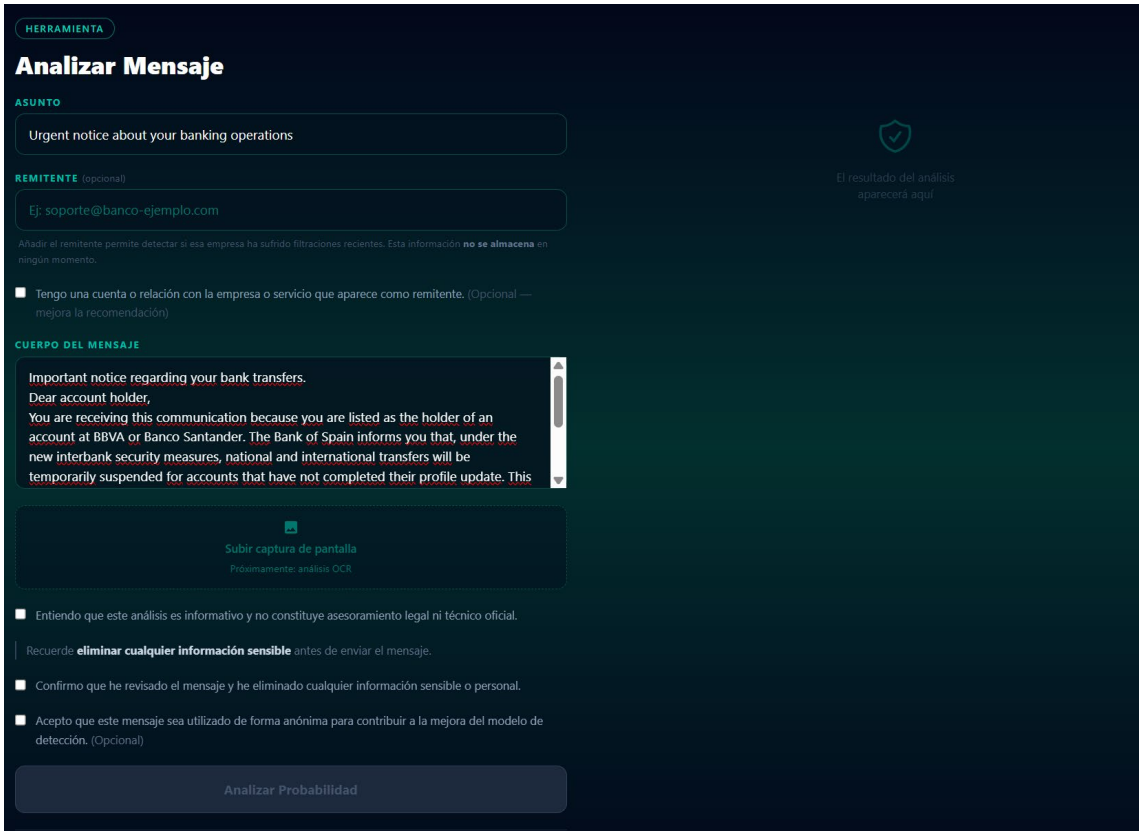

Figura 25. Formulario con los datos completados por el usuario.

Después, el usuario debe aceptar los *checkboxes* obligatorio y, opcionalmente, los adicionales, definidos como campos *opcionales*. En ese momento se activa el botón "Analizar Probabilidad" para poder hacer el análisis del mensaje, el contexto y los parámetros introducidos.

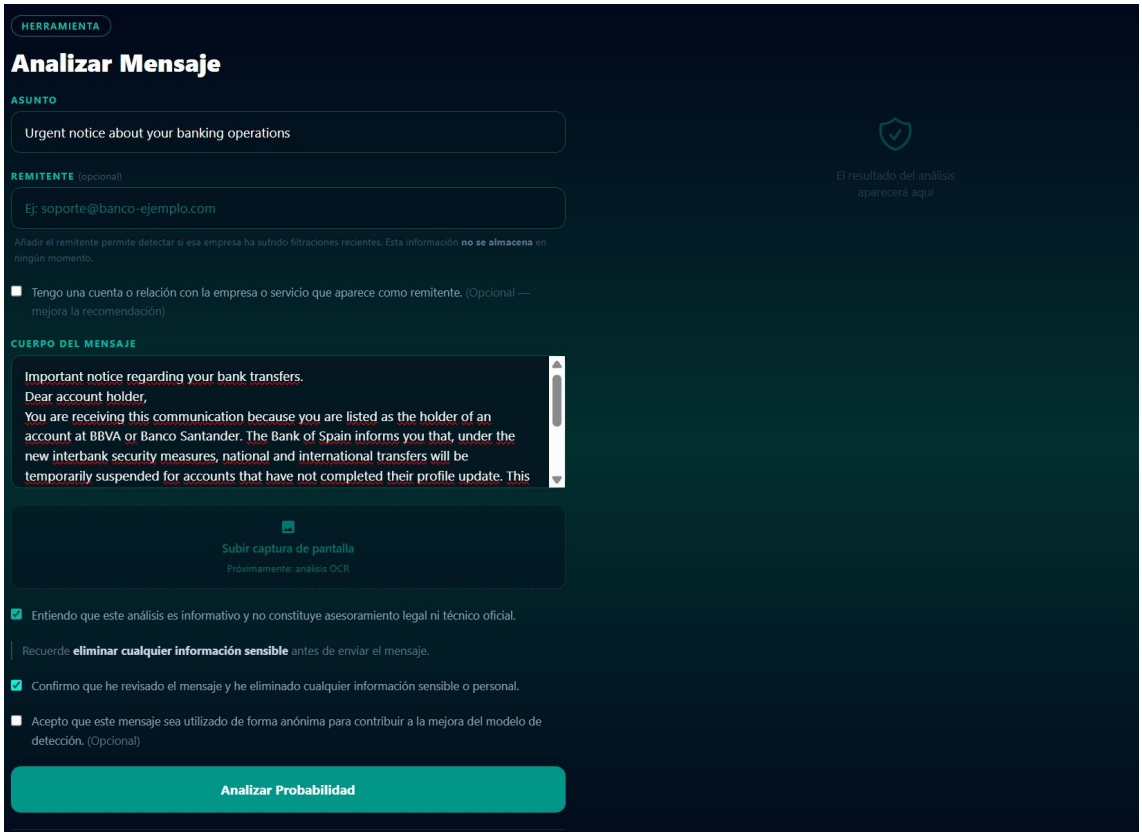

Figura 26. Captura de pantalla con los checkboxes obligatorios marcados.

Tras pulsar en ese botón, el usuario recibe en la misma página el porcentaje de riesgo asociado al mensaje en la parte superior derecha, mientras que las recomendaciones personalizadas aparecen en la parte inferior derecha.

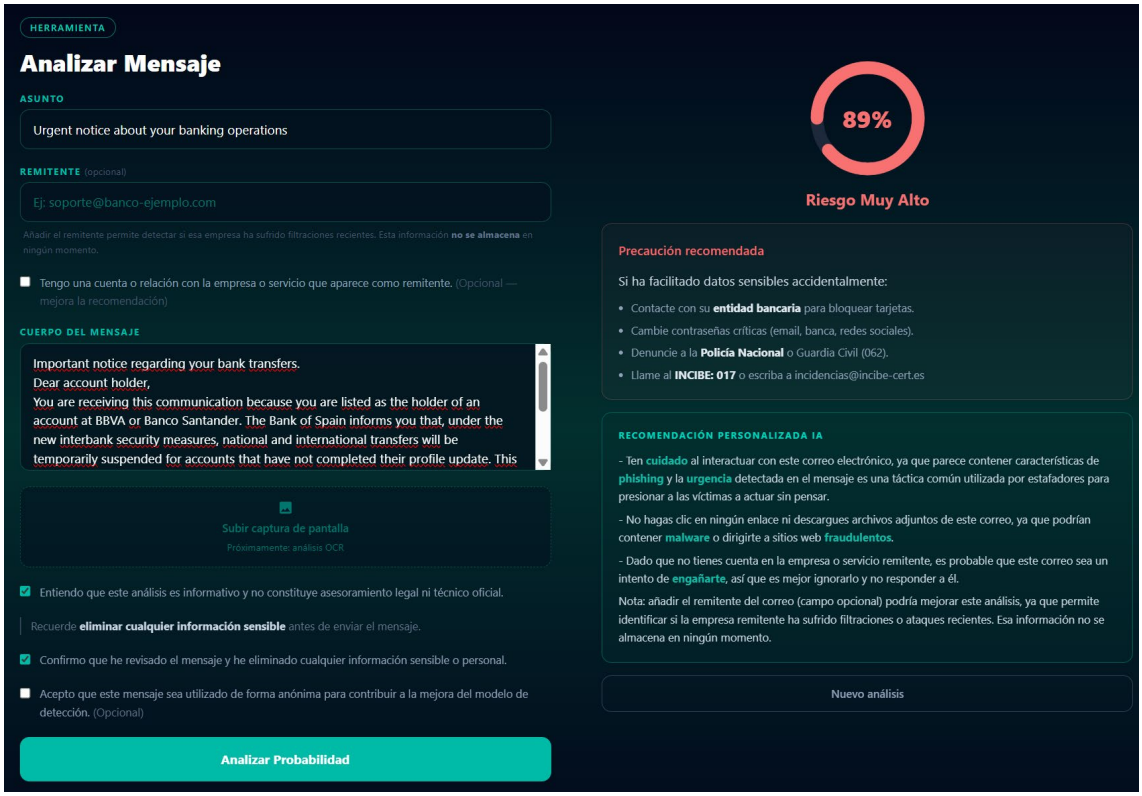

Figura 27. Captura de pantalla con la respuesta completa.

Si el usuario quiere analizar otro correo, puede realizar el mismo proceso descrito, o seleccionar la opción de “Nuevo Análisis”, limpiando el formulario.

Las métricas principales del modelo muestran un rendimiento muy elevado para un clasificador de texto basado en vectorización TF-IDF y Regresión Logística. Con el umbral ajustado a 0,44, Se obtiene una *accuracy* del 99,27% con un F1 de 99,31% para la clase *phishing*. Estos valores determinan que, de cada 1000 correos de *phishing*, solo 6 de ellos no son detectados. Además, se tiene un *recall* del 99,39%, con 32 falsos negativos sobre un total de 5.306 correos de *phishing*. Este margen en aceptable, ya que la herramienta desarrollada no es un filtro de bloqueo automático, sino una herramienta de ayuda a la decisión.

La elección del umbral 0,44 frente al estándar 0,5 reduce los falsos negativos en un tercio (de 48 a 32), manteniendo la *accuracy* constante, a costa de un ligero aumento en los falsos positivos. En el contexto de detección de *phishing*, donde un falso negativo tiene consecuencias más graves para el usuario que una falsa alarma, esta decisión está justificada.

Es relevante mencionar las limitaciones actuales del modelo, ya que se ha entrenado y validado con *datasets* públicos en inglés que, a pesar de ser ampliamente utilizados en la literatura, fueron recogidos entre los años 2008 y 2015, limitando el modelo actual a técnicas menos recientes y contextos diferentes de intentos de *phishing* [1]. Como se analiza en detalle en el apartado 6.3, las técnicas actuales de *phishing* más sofisticadas, como el abuso de plataformas legítimas o el *PDF-based phishing,* no están representadas en estos *datasets* de entrenamiento, lo que limita la capacidad del modelo para detectarlas.

6.2. Comparación con modelos alternativos

Se han entrenado tres modelos alternativos aparte de Regresión logística, partiendo del mismo *dataset* y verctorización TF-IDF para validar la elección del algoritmo de Regresión Logística: Random Forest, SVM Lineal y Naive Bayes Multinomial.

Modelo	Accuracy	Precision (phishing)	Recall (phishing)	F1 (phishing)	T. entrenamiento	T. inferencia
Regresión Logística producción	99,27%	99,23%	99,40%	99,31%	4,25s	0,004s
SVM Lineal	**99,69%**	**99,74%**	**99,68%**	**99,71%**	0,39s	0,003s
Random Forest	98,96%	99,52%	98,49%	99,01%	45,35s	0,689s
Naive Bayes	98,66%	99,54%	97,91%	98,72%	**0,03s**	**0,005s**

Tabla 9. Comparativa de métricas y tiempos de entrenamiento e inferencia de los cuatro modelos evaluados sobre el mismo dataset y vectorización TF-IDF. En negrita los mejores valores de cada métrica.

Se observa que los resultados obtenidos por el modelo SVM Lineal son los únicos que mejoran a Regresión Logística en todas las métricas por ligeras modificaciones. Random Forest es el modelo más lento y el *recall* es el más bajo de los cuatro. Naive Bayes es el modelo más rápido, pero menos preciso en este caso.

A pesar de que SVM Lineal obtiene las métricas más altas, no muestra probabilidades de forma nativa, mostrando únicamente el resultado binario sin porcentaje de riesgo asociado, funcionalidad esencial del modelo. Existe la posibilidad de obtener probabilidades mediante una calibración de *Platt scaling* [Niculescu-Mizil & Caruana, 2005], lo que aumenta la complejidad.

Regresión Logística produce directamente la probabilidad mediante la función sigmoide con coeficientes interpretables por palabra, funcionalidad esencial para mostrar al usuario los indicadores de la decisión. Esta interpretabilidad nativa justifica su elección frente a SVM a pesar de la ligera diferencia en métricas.

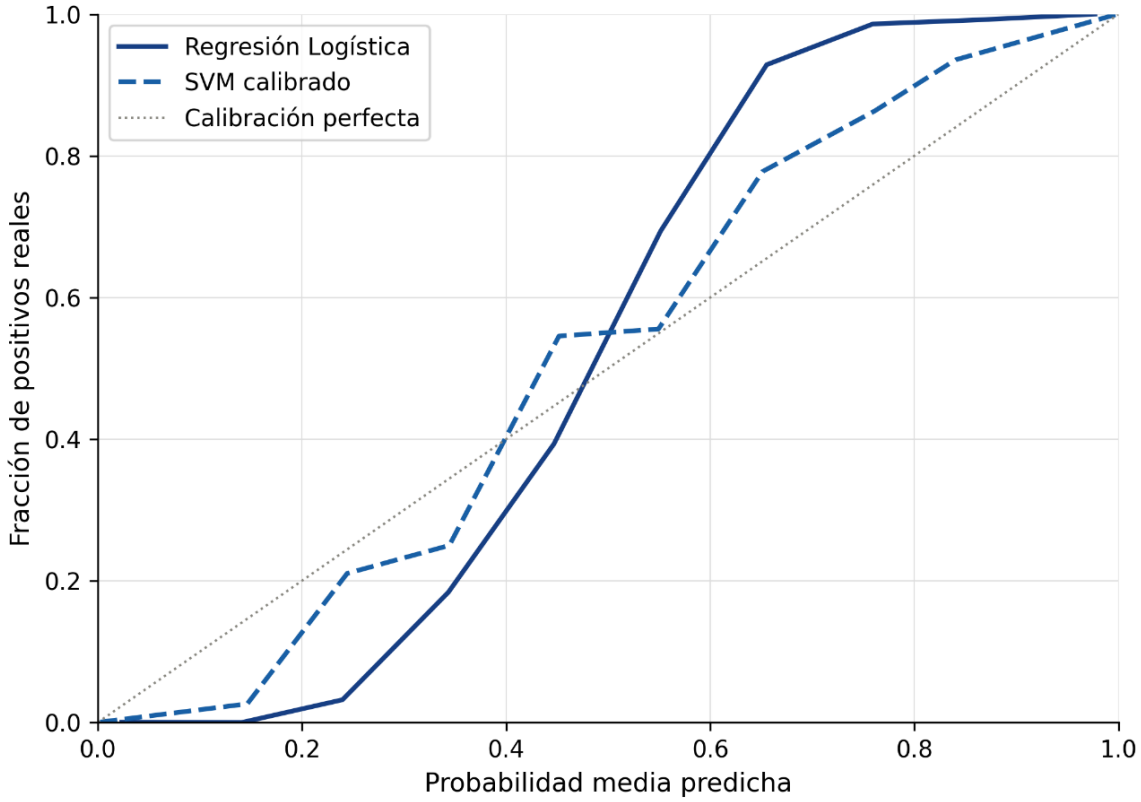

Figura 28. Curvas de calibración de Regresión Logística y SVM calibrado. Ambos modelos muestran desviaciones respecto a la calibración perfecta, con mayor concentración de probabilidades en los extremos en el caso de Regresión Logística, comportamiento coherente con la distribución observada en el apartado 5.2.4.

En conclusión, la elección de Regresión Logística como modelo en producción se hace buscando el equilibrio entre tres criterios: rendimiento, generación nativa de probabilidades e interpretabilidad de los coeficientes. Estos elementos son esenciales para poder mostrar al usuario un porcentaje de riesgo interpretable, así como identificar las palabras más determinantes en cada clasificación. Ninguno de los tres modelos alternativos cumple simultáneamente estos tres criterios con el mismo nivel de simplicidad y eficiencia.

6.3. Casos de uso reales: correos propios analizados

A continuación, se presenta el análisis técnico del *dataset* propio de validación, examinando las marcas suplantadas, las infraestructuras de envío identificadas y los niveles de sofisticación de los ataques detectados.

6.3.1. Descripción del dataset recopilado

Con el objetivo de evaluar el comportamiento del modelo ante correos actuales y reales, se diseñó un proceso de validación sobre el *dataset* propio descrito en la sección 5.2.1, compuesto por 88 correos electrónicos: 76 clasificados como *phishing* y 12 como legítimos. A diferencia del *dataset* de entrenamiento, formado por corpus públicos de hasta 2015, este *dataset* recoge campañas activas entre octubre de 2025 y mayo de 2026, permitiendo evaluar la efectividad del modelo ante técnicas de *phishing* actuales.

Para verificar que los correos eran clasificados correctamente en el *dataset*, se analizaron las cabeceras de los mensajes. Los mensajes recibidos de manera directa se analizaron mediante los parámetros SPF, DKIM, DMARC y BIMI a través del visor de cabeceras integrado en el proveedor de correos correspondiente. Por otro lado, los correos que fueron reenviados y de los que no se disponía de la cabecera original, se analizó el campo *References* y el contenido (remitente original, patrones de urgencia artificial, URLs).

Todos los correos recibidos estaban escritos en español, por lo que fueron traducidos al inglés mediante Google Translate antes de su análisis con PhishGuard.

6.3.2. Análisis de marcas suplantadas y técnicas detectadas

El análisis del *dataset* propio revela patrones de suplantación claramente diferenciados según el perfil del destinatario. En el subconjunto de correos recibidos directamente (subconjunto A), Spotify es la marca más atacada con diferencia, representando el 37% de los correos fraudulentos, seguida de Netflix con un 13% y las carteras de criptomonedas MetaMask y TrustWallet con un 7%. En el subconjunto de correos recopilados mediante reenvío (subconjunto B), el patrón es radicalmente distinto: predominan las entidades bancarias españolas, representando un 26% del total, seguidas de correos que suplantan organismos oficiales como la Agencia Tributaria, la DGT y Correos con un 11%, y una campaña de suplantación de Microsoft Defender que representa el 16%. Esta divergencia refleja la segmentación que realizan los atacantes según el perfil del destinatario, adaptando las marcas suplantadas en función del perfil del usuario.

Desde el punto de vista de la infraestructura de envío, el análisis de las cabeceras técnicas permitió clasificar los correos en cuatro categorías según su nivel de sofisticación, empezando por el más avanzado.

El primer tipo, el más peligroso y difícil de detectar, agrupa los correos enviados a través de plataformas legítimas de email o servicios cloud, como Amazon SES, Delighted, Constant Contact, Papershift, Microsoft Groups y Shopify Email, y representan aproximadamente el 33% del subconjunto A y el 18% del subconjunto B. Al operar bajo cuentas verificadas por estas plataformas legítimas, estos correos obtienen resultados válidos en SPF, DKIM y en ocasiones DMARC, dificultando su detección automática.

El segundo tipo recoge los correos enviados desde servidores universitarios comprometidos, identificándose un total de diez universidades en nueve países distintos (Argentina, Grecia, Turquía, India, Tailandia, Polonia, Honduras y España) y representa el 23% del subconjunto A.

El tercero engloba los correos enviados desde servidores propios registrados por los atacantes, como los dominios *sayerlack.info, kal.pl* o *flegeds.net*, y representa el 27% del subconjunto A y el 40% del subconjunto B de reenvíos.

Finalmente, el cuarto tipo agrupa los correos sin ningún tipo de autenticación configurada, todos ellos detectados correctamente como *spam* por los filtros de los proveedores, representando el 17% del *dataset* combinado.

Entre todas las infraestructuras identificadas, destaca el dominio *sayerlack.info* como la operación más activa, persistente y peligrosa del *dataset*. Se identificaron ocho correos atribuibles a esta infraestructura, distribuidos a lo largo de un periodo de al menos siete meses. La infraestructura presenta siempre la misma configuración técnica: *OpenDKIM v2.11.0*, selector *default_1024* y servidor *infosayer.sayerlack.info*, con SPF y DKIM correctamente configurados para su propio dominio. Esta campaña suplantó a las marcas Spotify y TrustWallet en el conjunto combinado. En dos de los ocho correos identificados se detectaron cabeceras propietarias de Johnson & Johnson, lo que indica que al menos parte de la campaña estaba dirigida específicamente a empleados de grandes corporaciones. Esto la sitúa en la categoría de *spear phishing* corporativo, significativamente más sofisticada que el *phishing* masivo e indiscriminado que representa la mayoría del *dataset* propio.

6.3.3. Efectividad de los filtros de spam existentes

El filtro de *spam* automático del proveedor de correos detectó en un 62% los mensajes de *spam*, dejando pasar a la bandeja de entrada el 38% de los correos del subconjunto A. No se ha registrado el porcentaje de mensajes detectados en el subconjunto B de correos.

Los correos no detectados fueron principalmente del primer y tercer tipo: se usaron plataformas legítimas o infraestructuras propias bien configuradas, haciendo que los parámetros SPF, DKIM y DMARC fueran válidos. En el caso de los correos del cuarto tipo, sin autenticación, fueron detectados en todos los casos.

Destaca el caso de *sayerlack.info*, ya que se detectaron solo tres de los ocho correos enviados. La configuración es idéntica en todos los mensajes, pero el contenido y el momento de envío varían, lo que sugiere que los filtros de los proveedores incluyen modelos dinámicos de reputación que mejoran progresivamente con el tiempo, aparte de los parámetros técnicos.

También destaca un caso reciente en el que se suplantaba a una entidad financiera en la que el correo no fue detectado como *spam*, ya que se incluyó todo el contenido fraudulento en un archivo PDF adjunto, evadiendo el análisis de texto del cuerpo del mensaje.

6.4. Resultados sobre el dataset propio

De los correos de *phishing* se detectaron correctamente el 92,2%, siendo clasificados como riesgo medio o alto. Solo seis correos fueron categorizados como riesgo bajo, por lo que fueron clasificados como falsos negativos. Además, la probabilidad media por cada región de riesgo fue: riesgo alto 80,5%, riesgo medio 46,8% y riesgo bajo 15,5%, con una probabilidad media del conjunto de 52,3%.

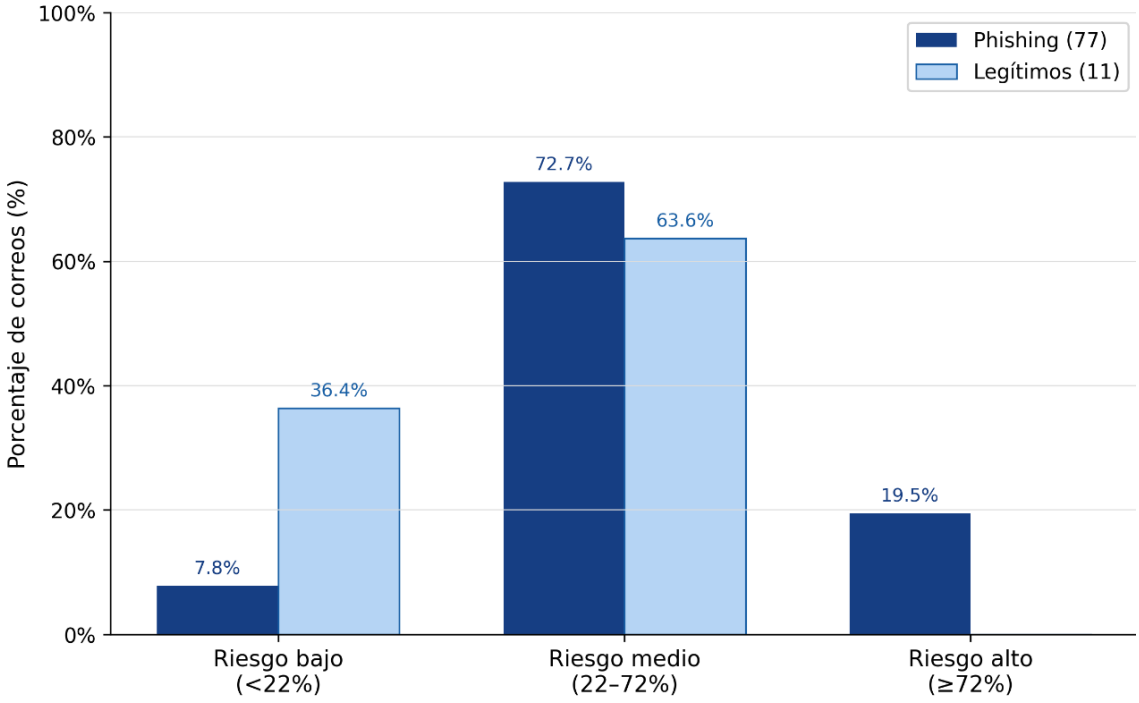

Figura 29. Distribución porcentual de los correos del dataset de validación por nivel de riesgo asignado por PhishGuard, separados por categoría (phishing y legítimos).

Los seis falsos negativos incluían *spam* de ofertas de almacenamiento en la nube y productos farmacéuticos. Analizando estos correos, no aparecen mensajes de urgencia ni vocabulario propio de mensajes de *phishing* "clásico", sino que se asemejan más a correos legítimos de publicidad. Estos fallos son coherentes con la limitación del *dataset* de entrenamiento.

Los correos clasificados como riesgo alto, que representan casi el 20% del conjunto de correos detectados como *phihing* por el modelo, son los que presentan las características propias de este tipo de mensajes. Algunos ejemplos son: suplantación de una entidad bancaria, detectado con un 80%; correo supuestamente enviado por la Dirección General de Tráfico (DGT), detectado con un 93%; y uno de los mensajes provenientes de servidores universitarios, detectado también con un 93% de riesgo.

Únicamente cuatro de los once correos legítimos fueron clasificados correctamente como riesgo bajo, mientras que los siete restantes se clasificaron con riesgo medio y ninguno como riesgo alto. El principal motivo de esta causa puede ser la limitación del idioma, ya que tras la traducción se puede perder parte del contexto. La probabilidad media de los mensajes legítimos es un 34% frente al 52,3% que presentaban los mensajes de *phishing*, lo que demuestra que hay una discriminación correcta entre las dos clases, a pesar de no ser suficiente para ser tan concluyente como con los datos originales del *dataset* de test.

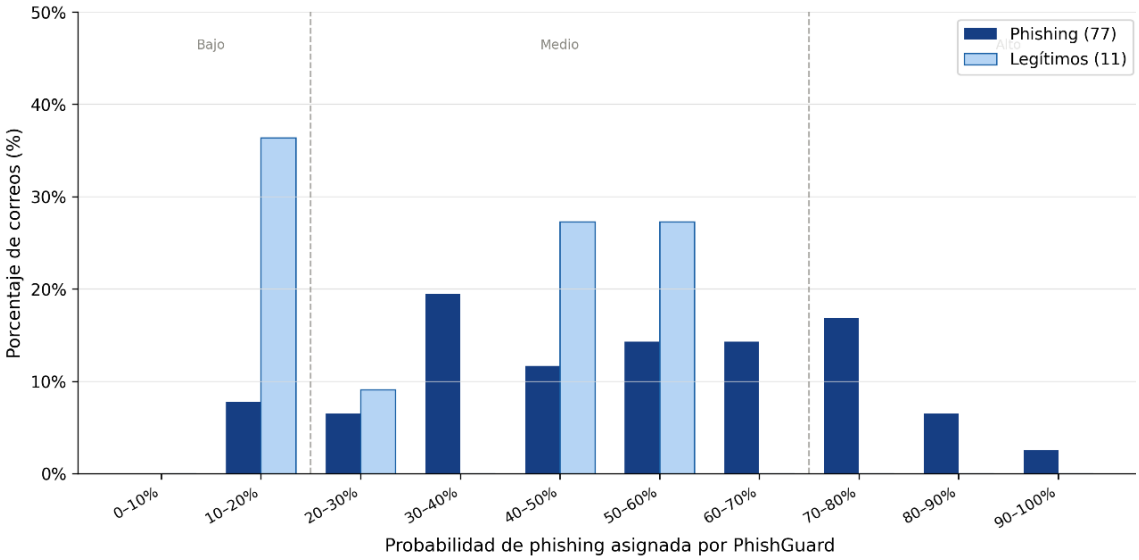

Figura 30. Histograma de las probabilidades asignadas por PhishGuard a los correos del dataset de validación, normalizado por categoría. Las líneas discontinuas indican los umbrales de riesgo bajo (22%) y riesgo alto (72%).

Esta diferencia entre la distribución observada en el *dataset* de test (donde el 94% de los casos se concentraba en los extremos) y la distribución del *dataset* de validación (donde predomina la zona intermedia) confirma que la traducción al inglés introduce ruido que reduce la capacidad discriminativa del modelo.

En conclusión, el modelo es efectivo como primera capa de detección para *phishing* "clásico", pero presenta limitaciones ante spam genérico y correos legítimos en español. Es por este motivo por el que la capa LLM con recomendaciones personalizadas aporta un valor esencial, complementando estas limitaciones para añadir contexto a la solución proporcionada.

6.5. Análisis cualitativo de las recomendaciones del LLM

Se analizaron cualitativamente las recomendaciones generadas en la totalidad de los correos del *dataset* propio. De todos ellos, 16 tenían dominio corporativo y en todos ellos Tavily encontró filtraciones, ya que los dominios de entidades conocidas (bancos, organismos oficiales, servicios de mensajería) tienen historial documentado de filtraciones. Los dominios genéricos no tienen filtraciones asociadas porque son proveedores de correo electrónico, no entidades suplantadas. La función de detección de urgencia funcionó correctamente en la mayoría de los casos, teniendo un único falso positivo.

A continuación, se analizan cuatro casos diferentes: los tres primeros de mensajes de *phishing* y el último de uno legítimo.

En este primer caso, se incluyó tanto asunto como remitente, indicando que no se tiene relación con la entidad. En esta ocasión, Tavily encontró un conjunto de campañas de *phishing* que tenían como objetivo suplantar a la DGT [40, 41 y 42]. En este caso se observa una personalización máxima, y con una orientación clara a no interactuar porque las probabilidades de que el mensaje no sea legítimo son muy altas: un riesgo alto clasificado por el modelo y una campaña de filtración.

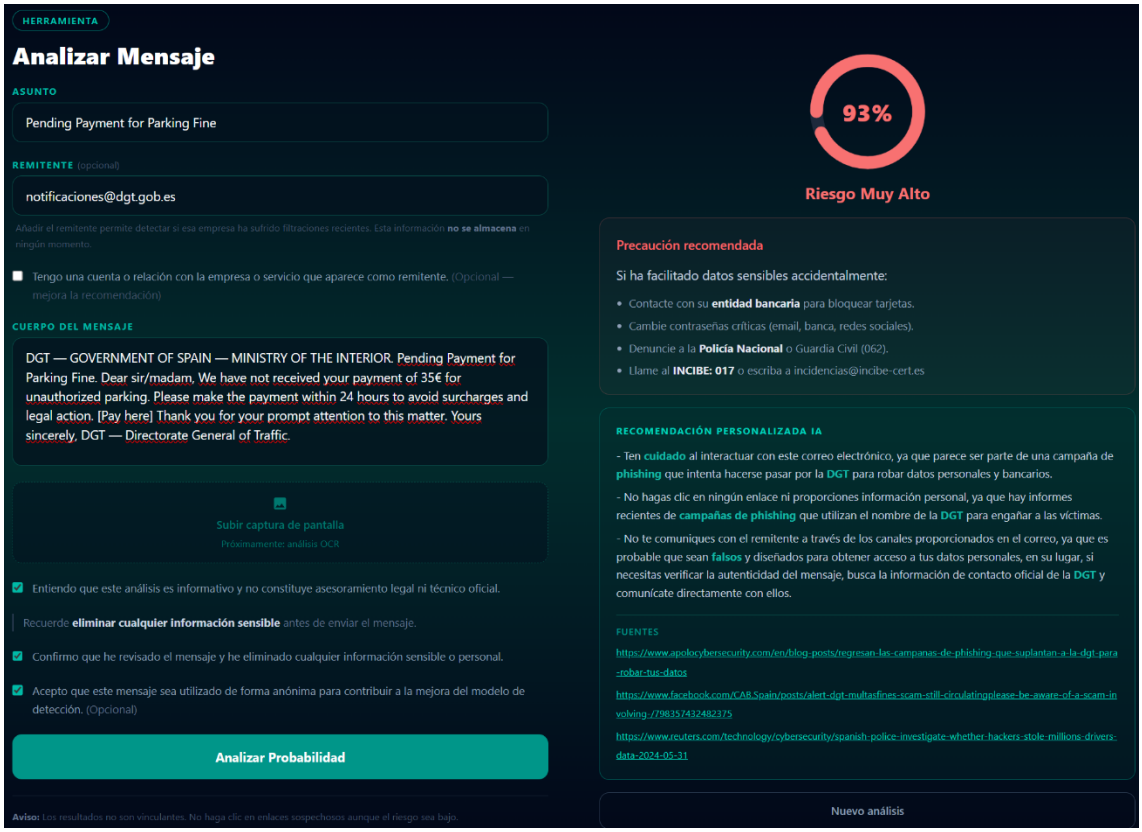

Figura 31. Resultado del análisis de un correo de riesgo muy alto (93%) con remitente, mostrando recomendación personalizada con contexto de filtraciones encontradas por Tavily.

En el segundo caso analizado, no se incluye el remitente, por lo que no se realiza ninguna búsqueda en Tavily. Sin embargo, la función de detección de urgencia ha identificado un patrón de urgencia artificial, ya que se indica al usuario que debe actuar antes de “48 horas”. La recomendación informa de esta urgencia y de que, para poder hacer un análisis más extenso, incluya el remitente del mensaje. Además, al informar de que se tiene relación previa con la empresa, otra de las sugerencias es contactar a la empresa por medios oficiales de forma directa.

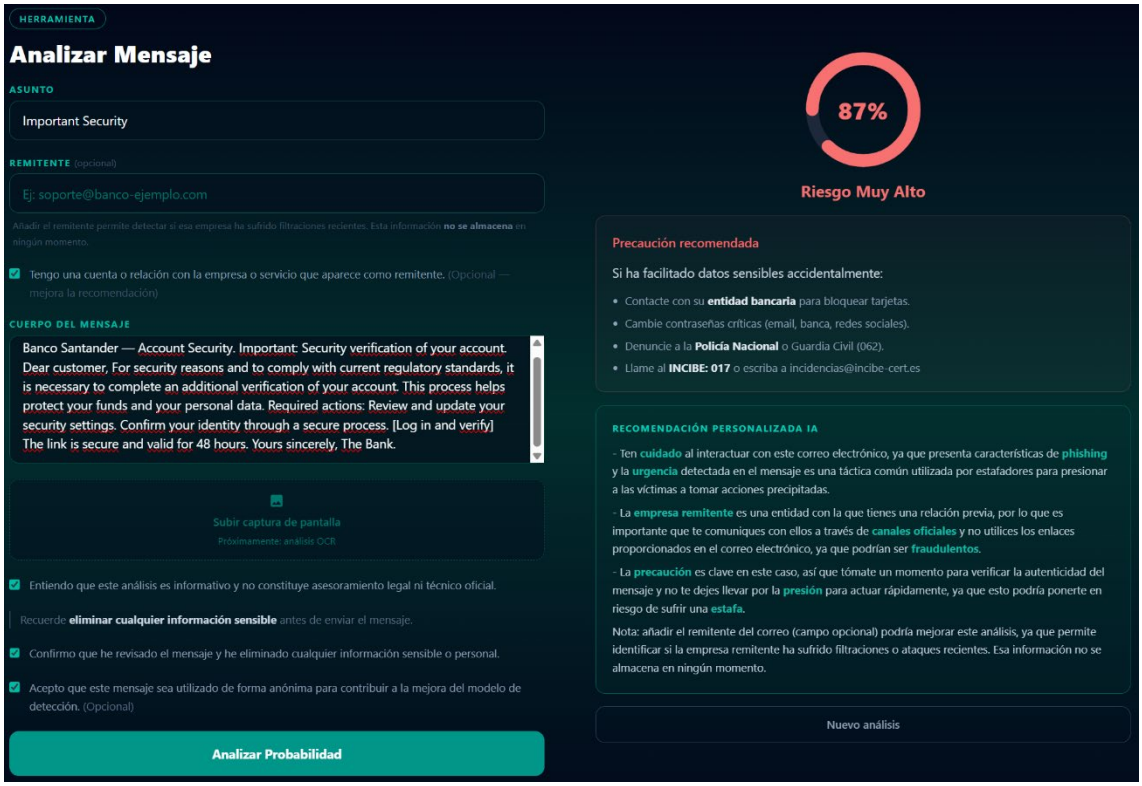

Figura 32. Resultado del análisis de un correo de riesgo muy alto (87%) sin remitente, mostrando recomendación basada en urgencia detectada y cuenta del usuario.

Para el siguiente caso se ha analizado un correo de riesgo moderado, en el que se han incluido todos los datos: asunto, remitente y cuerpo de mensaje. Además, en este caso se cuenta con relación previa con la entidad que supuestamente envía el correo. En este caso, Tavily encontró una filtración correspondiente con la empresa en cuestión [50, 51, 52], a lo que, sumado al riesgo estimado por el modelo, se recomienda precaución.

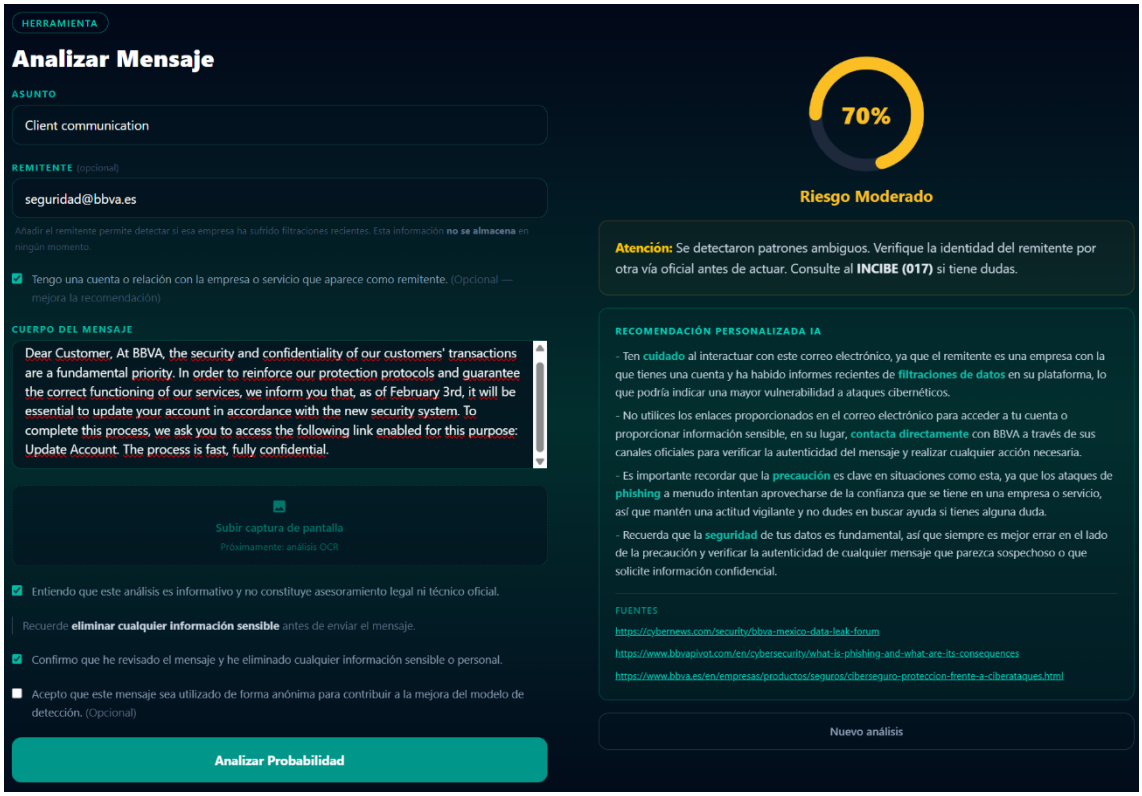

Figura 33. Resultado del análisis de un correo de riesgo moderado (70%) con remitente, mostrando recomendación equilibrada con contexto de filtraciones de BBVA

En el cuarto caso se ha analizado un correo legítimo de una empresa de mensajería. En este caso, no se ha añadido remitente, solo asunto y cuerpo del correo, por lo que aparecen recomendaciones genéricas, que pueden aplicar a cualquier correo, como que es recomendable no hacer clic en ningún enlace de un correo si no se cuenta con una autenticación del remitente. Además, se aprecia cómo el tono del mensaje es más tranquilizador que en el resto de los casos.

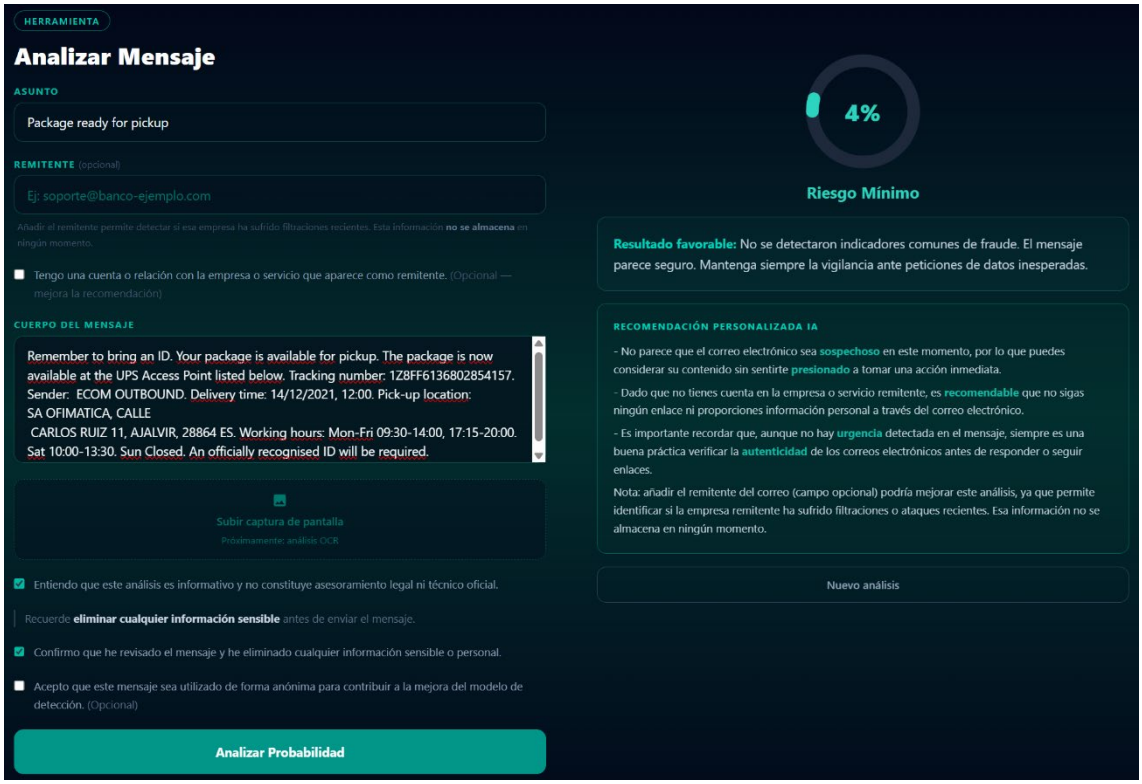

Figura 34. Resultado del análisis de un correo legítimo de riesgo mínimo (4%), mostrando recomendación tranquilizadora sin urgencia detectada.

Los cuatro casos analizados demuestran que el módulo de recomendaciones personaliza su respuesta de forma coherente en función de tres variables: el nivel de riesgo estimado por el modelo, la presencia o ausencia de remitente y si el usuario tiene relación previa con la entidad. La combinación de estas variables genera recomendaciones cualitativamente distintas: desde una orientación firme y específica ante riesgo alto con filtraciones documentadas, hasta un mensaje tranquilizador ante un correo legítimo sin indicadores de fraude.

Este comportamiento diferenciado confirma que la capa LLM aporta valor real como complemento al modelo de ML, añadiendo orientación accionable y comprensible a una probabilidad numérica.

Capítulo 7. Conclusiones

En esta sección se recogen las conclusiones extraídas del desarrollo y evaluación del proyecto, se analiza el grado de cumplimiento de los objetivos definidos en la sección 4.2 y se identifican las principales líneas de trabajo futuro.

7.1. Conclusiones

En el presente proyecto se ha creado una aplicación web completa con despliegue en la nube para la detección de *phishing*, combinando un modelo de Machine Learning entrenado con un conjunto de *datasets* públicos y una capa de inteligencia artificial generativa que convierte la predicción numérica en orientación personalizada y comprensible para cualquier usuario. La herramienta ha sido validada tanto con métricas estándar provenientes de la validación con el *dataset* de test, tanto como por un conjunto de 88 correos electrónicos reales, recopilados durante siete meses, añadiendo una comprobación extra del modelo con datos actuales y pudiendo verificar la calidad cualitativa de las recomendaciones según el caso tratado.

La aportación más relevante del proyecto es tanto técnica como de planteamiento: la combinación ML y LLM como diseño conjunto con funciones perfectamente definidas e independientes, a la vez que complementarias. En primer lugar, el modelo de Regresión Logística proporciona eficiencia, trazabilidad e interpretabilidad, mostrando las palabras que más han influido en la decisión. En segundo lugar, el LLM transforma esos resultados en una recomendación accionable personalizada al contexto concreto tanto del usuario como del mensaje. Se tienen en cuenta varios parámetros como la relación del usuario con la empresa remitente, la urgencia detectada en el mensaje, el porcentaje de riesgo determinado por el ML y las filtraciones recientes de la empresa remitente. Esta aproximación no se encuentra en ninguna otra herramienta analizada en el Estado del Arte del documento.

Las métricas obtenidas sobre el *dataset* de test son excelentes: *accuracy* del 99,27%, *F1-score* del 99,31% y *recall* del 99,4%, superando los resultados recogidos por la mayor parte de la literatura consultada en la que se analizan estructuras similares. Sin embargo, la validación con correos actuales y traducidos reveló dos limitaciones relevantes. La primera es el contenido desactualizado del *dataset* de entrenamiento y test, ya que los corpus públicos fueron recopilados entre 2005 y 2015, lo que no refleja las técnicas actuales de *phishing*, reduciendo la capacidad del modelo para detectar ataques de mayor sofisticación como los identificados en el *dataset* propio en la validación. La segunda es la limitación del idioma: el 63,6% de los correos legítimos, escritos

originalmente en español, fueron asociados erróneamente como riesgo moderado tras su traducción, demostrando que la traducción introduce ruido que disminuye la capacidad de clasificación del modelo.

Estas limitaciones hacen que la herramienta actual no sea infalible, pero de igual manera consigue alcanzar el objetivo principal del proyecto: proporcionar al usuario una herramienta accesible, transparente y accionable que no sustituye su criterio ni decisión, sino que lo complementa. El análisis del *dataset* propio demostró además que el 38% de los correos de phishing más sofisticados superaron los filtros automáticos de los proveedores de correo, confirmando la necesidad de herramientas complementarias orientadas al usuario final.

Con esta herramienta se busca la evolución continua a través de medidas futuras como el reentrenamiento con contribuciones de la comunidad, el soporte multilingüe y la mejora del módulo de búsqueda en tiempo real. El desarrollo de estas funcionalidades, junto con otras, tiene como objetivo último convertir esta herramienta en una referencia para la concienciación en ciberseguridad ciudadana. En un entorno en el que los ataques son cada vez más sofisticados, la respuesta más efectiva no es solo técnica, es también educativa, y ese es el vacío que pretende resolver este proyecto.

7.2. Objetivos cumplidos

Se han analizado el grado de cumplimiento de las tres categorías de objetivos. Los dos objetivos principales se han cumplido. Por otro lado, cuatro de los cinco objetivos generales adicionales se han cumplido, mientras que uno de ellos se ha cumplido parcialmente. En último lugar, se han cumplido ocho de los diez objetivos específicos, mientras que los dos restantes se han cubierto parcialmente.

El objetivo de crear y mostrar de estadísticas se ha cumplido parcialmente, ya que existen estadísticas globales actualizadas, pero no se muestran contribuciones recientes de otros usuarios, funcionalidad que requiere implementar la gestión de usuarios, que se define como una de las principales líneas de trabajo futuras.

El objetivo de crear una herramienta actualizada y en constante evolución también se ha cumplido parcialmente, ya que el sistema de contribución voluntaria está implementado y es funcional, pero el reentrenamiento automático del modelo con esas contribuciones está pendiente de desarrollo.

7.3. Líneas de trabajo futuro

A continuación, se han clasificado las líneas de trabajo futuro en tres grupos diferenciados: el primero relacionado con el modelo de Machine Learning, el segundo de funcionalidades y el tercero de mejoras en la arquitectura y experiencia de usuario.

7.3.1. Mejoras del modelo ML

- Incluir mensajes y correos electrónicos cedidos por los usuarios, para poder reentrenar el modelo.
- Explorar la detección de correos fraudulentos en español: valorar si añadir un filtro de traducción previa o entrenar un nuevo y diferente.
- Creación de un *honeypot* para obtener suficientes correos para comprobar las métricas del modelo.

7.3.2. Ampliación de funcionalidades

- Añadir OCR para hacer análisis de capturas de pantalla, y elementos de detección de estilo y colores para poder analizar más parámetros de los mensajes.
- Añadir análisis de *URL* fraudulentas, incluyéndolo en el análisis del remitente para poder verificarlo.
- Recomendaciones legales más específicas y más ayuda a los usuarios en la sección legal.
- Ampliación de la página de estadísticas y añadir estadísticas personales del usuario.
- Adaptar las recomendaciones al perfil del usuario, permitiendo seleccionar un nivel de conocimiento técnico (básico, intermedio, avanzado) que ajuste el lenguaje y el detalle de las explicaciones proporcionadas.

7.3.3. Mejoras de arquitectura y experiencia de usuario

- Añadir gestión segura de usuarios para poder analizar y hacer seguimiento de correos que hayan analizado.
- Crear un foro en el que los usuarios pueda compartir sus correos.
- Definición de políticas RLS explícitas para determinar qué roles tienen acceso a lectura y cuáles a escritura en cada tabla.
- Añadir soporte para mostrar la aplicación tanto en español como en inglés, estudiando la posible traducción a más idiomas.
- Centralización de la capa de servicios en el *frontend*.

Capítulo 8. Bibliografía

[1] A. Al-Subaiey, M. Al-Thani, N. A. Alam, K. F. Antora, A. Khandakar y S. A. U. Zaman, "Novel Interpretable and Robust Web-based AI Platform for Phishing Email Detection," *arXiv*, may. 2024. [Online]. Disponible en: https://arxiv.org/abs/2405.11619

[2] APWG, "Phishing Activity Trends Report, Q2 2025," Anti-Phishing Working Group, 2025. [Online]. Disponible en: https://docs.apwg.org/reports/apwg_trends_report_q2_2025.pdf

[3] FBI, "2025 Internet Crime Report," Internet Crime Complaint Center (IC3), 2025. [Online]. Disponible en: https://www.ic3.gov/AnnualReport/Reports/2025_IC3Report.pdf

[4] ENISA, "ENISA Threat Landscape 2025," European Union Agency for Cybersecurity, 2025. [Online]. Disponible en: https://www.enisa.europa.eu/publications/enisa-threat-landscape-2025

[5] INCIBE, "Balance de Ciberseguridad 2025," Instituto Nacional de Ciberseguridad, feb. 2026. [Online]. Disponible en: https://www.incibe.es/incibe/sala-de-prensa/incibe-detecto-mas-de-122000-incidentes-de-ciberseguridad-en-2025

[6] Ministerio del Interior, "Informe sobre la Cibercriminalidad en España 2024," Gobierno de España, nov. 2025. [Online]. Disponible en: https://www.lamoncloa.gob.es/serviciosdeprensa/notasprensa/interior/paginas/2025/211125-informe-cibercriminalidad.aspx

[7] SafetyDetectives, "Antivirus y ciberseguridad: estadísticas, tendencias, datos 2026." [Online]. Disponible en: https://es.safetydetectives.com/blog/antivirus-statistics-es/

[8] Parlamento Europeo y Consejo de la UE, "Directiva (UE) 2015/2366 sobre servicios de pago en el mercado interior (PSD2)," *Diario Oficial de la Unión Europea*, L 337, nov. 2015. [Online]. Disponible en: https://eur-lex.europa.eu/legal-content/ES/TXT/?uri=CELEX%3A32015L2366

[9] Jefatura del Estado, "Real Decreto-ley 19/2018, de 23 de noviembre, de servicios de pago y otras medidas urgentes en materia financiera," *BOE* núm. 284, 2018. [Online]. Disponible en: https://www.boe.es/buscar/doc.php?id=BOE-A-2018-16036

[10] Parlamento Europeo y Consejo de la UE, "Reglamento (UE) 2016/679 relativo a la protección de las personas físicas en lo que respecta al tratamiento de datos personales (RGPD)," *Diario Oficial de la Unión Europea*, L 119, abr. 2016. [Online]. Disponible en: https://eur-lex.europa.eu/legal-content/ES/TXT/?uri=CELEX%3A32016R0679

[11] A. A. Tawil, L. Almazaydeh, D. Qawasmeh, B. Qawasmeh, M. Alshinwan et al., "Comparative Analysis of Machine Learning Algorithms for Email Phishing Detection Using TF-IDF, Word2Vec, and BERT," *Computers, Materials & Continua*, vol. 81, no. 2, pp. 3395–3412, 2024. doi: 10.32604/cmc.2024.057279

[12] T. Koide et al., "Real-Time Phishing URL Detection Using Machine Learning," *Engineering Proceedings*, vol. 107, p. 108, 2025. [Online]. Disponible en: https://www.researchgate.net/publication/395874974

[13] T. Joachims, "Text Categorization with Support Vector Machines: Learning with Many Relevant Features," in *Proc. 10th European Conf. on Machine Learning (ECML)*, 1998, pp. 137–142. [Online]. Disponible en: https://www.researchgate.net/publication/28351286_Text_Categorization_with_Support_Vector_Machines

[14] H. Zhang, "The Optimality of Naive Bayes," in *Proc. 17th Int. Florida Artificial Intelligence Research Society Conf. (FLAIRS)*, 2004, pp. 562–567. [Online]. Disponible en: https://cdn.aaai.org/FLAIRS/2004/Flairs04-097.pdf

[15] S. Salloum et al., "In-Depth Analysis of Phishing Email Detection: Evaluating the Performance of Machine Learning and Deep Learning Models Across Multiple Datasets," *Applied Sciences*, vol. 15, no. 6, p. 3396, 2025. doi: 10.3390/app15063396

[16] B. Lim et al., "EXPLICATE: Enhancing Phishing Detection through Explainable AI and LLM-Powered Interpretability," *arXiv*, mar. 2025. [Online]. Disponible en: https://arxiv.org/pdf/2503.20796

[17] S. Khandan et al., "An Explainable Multimodal Framework for Phishing Attack Detection," in *Computer Security. ESORICS 2025 International Workshops*, Springer, 2026, pp. 75–91. doi: 10.1007/978-3-032-16165-9_5

[18] A. Niculescu-Mizil y R. Caruana, "Predicting Good Probabilities with Supervised Learning," in *Proc. 22nd Int. Conf. on Machine Learning (ICML)*, 2005. [Online]. Disponible en: https://www.cs.cornell.edu/~alexn/papers/calibration.icml05.crc.rev3.pdf

[19] Agencia Tributaria, "Phishing," Sede Electrónica de la Agencia Tributaria. [Online]. Disponible en: https://sede.agenciatributaria.gob.es/Sede/ayuda/Phishing.html

[20] Groq Inc., "Groq LPU Inference Engine," *Groq Technology*, 2024. [Online]. Disponible en: https://groq.com/technology/

[21] INCIBE, "Ingeniería social: técnicas utilizadas por los ciberdelincuentes y cómo protegerse." [Online]. Disponible en: https://www.incibe.es/empresas/blog/ingenieria-social-tecnicas-utilizadas-los-ciberdelincuentes-y-protegerse

[22] INCIBE, "¿Sabes cómo funciona un ciberataque que utiliza ingeniería social?" [Online]. Disponible en: https://www.incibe.es/empresas/blog/sabes-funciona-ciberataque-utiliza-ingenieria-social

[23] INCIBE, "Evita los engaños en la red: utiliza soluciones antifraude." [Online]. Disponible en: https://www.incibe.es/empresas/blog/evita-los-enganos-red-utiliza-soluciones-antifraude

[24] INCIBE, "Técnicas de ingeniería social," Infografía. [Online]. Disponible en: https://www.incibe.es/ciudadania/formacion/infografias/tecnicas-ingenieria-social

ANEXO I. Alineación con los Objetivos de Desarrollo Sostenible (ODS)

La Agenda 2030 de las Naciones Unidas establece 17 Objetivos de Desarrollo Sostenible (ODS) como marco global para el desarrollo sostenible. Este proyecto se alinea con cuatro de ellos, dos de forma primaria y dos de forma secundaria, como se recoge en la siguiente tabla:

Dimensión	ODS	Rol	Meta específica
Económica	ODS 9: Industria, innovación e infraestructura	Primario	Meta 9.c: Aumentar el acceso a las TIC
Social	ODS 16: Paz, justicia e instituciones sólidas	Primario	Meta 16.a: Prevenir la delincuencia
Social	ODS 4: Educación de calidad	Secundario	Meta 4.4: Competencias digitales
Social	ODS 10: Reducción de las desigualdades	Secundario	Meta 10.2: Inclusión digital

Tabla 10. Identificación y clasificación de los ODS alineados con el proyecto, con sus roles de impacto y metas específicas de la Agenda 2030.

ODS 9: Industria, innovación e infraestructura

El *phishing* debilita la confianza en los servicios digitales y genera pérdidas económicas significativas para ciudadanos y empresas. La construcción de infraestructuras digitales más seguras y accesibles es una de las metas del ODS 9, que reconoce el papel de la tecnología como motor del desarrollo económico sostenible.

Este proyecto contribuye directamente a esta meta mediante el desarrollo de una herramienta de ciberseguridad de acceso libre que combina aprendizaje automático e inteligencia artificial generativa. La arquitectura de microservicios desplegada íntegramente en la nube garantiza la disponibilidad y escalabilidad del sistema, mientras que el modelo de clasificación, con un F1 del 99,31% sobre el *dataset* de entrenamiento y un 92,2% de correos phishing alertados correctamente en el *dataset* de validación real, demuestra la viabilidad técnica de soluciones de ciberseguridad accesibles para cualquier usuario.

ODS 16: Paz, justicia e instituciones sólidas

El fraude digital y el *phishing* son formas de delincuencia económica que erosionan la confianza de los ciudadanos en las instituciones financieras, administrativas y digitales. El ODS 16 establece como meta la reducción de todas las formas de delincuencia y el fortalecimiento de instituciones que protejan a los ciudadanos.

PhishGuard actúa como mecanismo de defensa proactivo contra el cibercrimen, proporcionando al ciudadano una herramienta que le permite identificar mensajes fraudulentos que suplantan a entidades bancarias, organismos oficiales y servicios digitales, instituciones cuya credibilidad protege el ODS 16. El análisis del *dataset* propio de validación identificó campañas activas de suplantación de entidades como la DGT, la Agencia Tributaria y entidades bancarias españolas, confirmando la relevancia directa del proyecto en el contexto normativo español y europeo.

ODS 4: Educación de calidad

La alfabetización digital es una competencia fundamental en la sociedad actual. El ODS 4 establece como meta garantizar que todos los ciudadanos adquieran las competencias necesarias para participar en la economía digital, incluyendo la capacidad de identificar riesgos en entornos digitales.

PhishGuard contribuye a esta meta de forma indirecta al diseñarse no como un sistema de bloqueo automático sino como una herramienta de ayuda a la decisión: muestra al usuario las palabras que más han influido en la clasificación del modelo, genera recomendaciones explicativas en lenguaje natural y proporciona orientación sobre buenas prácticas de ciberseguridad. Este enfoque fomenta el pensamiento crítico del usuario y contribuye a que, con el tiempo, desarrolle mayor capacidad para identificar mensajes fraudulentos de forma autónoma.

ODS 10: Reducción de las desigualdades

Las soluciones avanzadas de ciberseguridad están mayoritariamente orientadas a entornos empresariales y presentan barreras económicas y técnicas que las hacen inaccesibles para el ciudadano particular. El ODS 10 establece como meta potenciar la inclusión de todas las personas, independientemente de su situación económica o nivel de formación.

PhishGuard reduce esta brecha al ofrecer una herramienta de acceso completamente gratuito, sin requisitos de registro ni instalación, diseñada para ser comprensible por cualquier perfil de usuario. El diseño responsive permite su uso tanto

en dispositivos de escritorio como en móvil, ampliando su alcance a usuarios con acceso limitado a tecnología. La ausencia de barreras de entrada convierte a PhishGuard en una herramienta de democratización de la ciberseguridad, alineada directamente con los principios de equidad e inclusión del ODS 10.

ANEXO II. Código fuente relevante del Modelo ML de la aplicación

```
from sklearn.metrics import precision_recall_curve

precision, recall, thresholds = precision_recall_curve(y_test, y_prob)

y_prob = clf.predict_proba(X_test_tfidf)[:,1]
precision, recall, thresholds = precision_recall_curve(y_test, y_prob)

precision_t = precision[:-1]
recall_t = recall[:-1]
th = thresholds

f1 = 2 * (precision_t * recall_t) / (precision_t + recall_t + 1e-12)
best = np.argmax(f1)

print("Umbral:", th[best], "precision:", precision_t[best], "recall:",
recall_t[best], "F1:", f1[best])

def get_word_phishing_indicators(vector_tfidf, tfidf_model,
classifier):

    feature_names = tfidf_model.get_feature_names_out()
    coefficients = classifier.coef_[0]

    word_list = []

    for idx, value in zip(vector_tfidf.indices, vector_tfidf.data):
        weight = coefficients[idx] * value
        word_list.append((feature_names[idx], weight))

    word_list.sort(key=lambda x: x[1], reverse=True)

    return [word for word, _ in word_list[:5]]
```

ANEXO III. Repositorio con el código completo de la aplicación

El resto del código base de la aplicación se encuentra en el siguiente repositorio público: https://github.com/MariaBegara/tfg-phishing-detection

Además, los corpus pueden descargarse: https://www.kaggle.com/datasets/naserabdullahalam/phishing-email-dataset