
Máster en Big Data

Trabajo Fin de Máster

# LLM-Based Email Management

Autor
Javier Arroyo García

Dirigido por
Andrés Jiménez Arocha

Madrid
Junio 2026

**Resumen**

El correo electrónico continúa siendo uno de los principales canales de entrada de información en las grandes organizaciones del sector asegurador, donde la gestión manual de altos volúmenes de mensajes con adjuntos heterogéneos representa un cuello de botella operativo significativo. El presente Trabajo de Fin de Máster aborda el diseño e implementación de una solución end-to-end para la gestión automatizada de buzones corporativos basada en Large Language Models (LLMs), desplegada sobre una arquitectura serverless en AWS dentro del entorno corporativo de MAPFRE.

El sistema implementa un pipeline desacoplado de funciones Lambda especializadas que cubre la ingesta de correos, la extracción y procesamiento de adjuntos mediante una estrategia híbrida que combina técnicas clásicas de OCR con las capacidades multimodales del modelo, la clasificación documental contra un catálogo definido por negocio, la extracción del número de referencia del siniestro y el reenvío al sistema transaccional. La adaptación del modelo se ha resuelto íntegramente mediante prompt engineering iterativo, evitando el coste y la rigidez del fine-tuning y permitiendo ajustar el comportamiento del sistema en ciclos cortos de validación conjunta con el equipo de tramitadores. El diseño contempla la generalización a nuevos buzones corporativos mediante un loader pattern, mecanismos de resiliencia que garantizan la no pérdida de información, un plan de contingencia para escenarios de fallo total del servicio, y monitorización en dos capas — técnica mediante Amazon CloudWatch y funcional mediante dashboards en Power BI.

La evaluación del sistema sobre batches reales de correos del buzón piloto evidencia una mejora sustancial frente a la solución basada en NLP clásico que se sustituye, validando tanto la viabilidad técnica como el potencial impacto operativo de la aproximación propuesta.

**Abstract**

Email remains one of the primary channels through which information enters large organizations in the insurance sector, where the manual handling of high volumes of messages with heterogeneous attachments represents a significant operational bottleneck. This Master's Thesis addresses the design and implementation of an end-to-end solution for the automated management of corporate mailboxes based on Large Language Models (LLMs), deployed on a serverless architecture on AWS within MAPFRE's corporate environment.

The system implements a decoupled pipeline of specialized Lambda functions covering email ingestion, the extraction and processing of attachments through a hybrid strategy that combines classical OCR techniques with the multimodal capabilities of the model, document classification against a business-defined catalogue, the extraction of the claim reference number, and forwarding to the transactional system. Model adaptation has been resolved entirely through iterative prompt engineering, avoiding the cost and rigidity of fine-tuning and allowing the system's behaviour to be adjusted in short cycles of joint validation with the claims-handling team. The design considers generalization to new corporate mailboxes through a loader pattern, resilience mechanisms that guarantee no loss of information, a contingency plan for total service failure scenarios, and two-layer monitoring — technical through Amazon CloudWatch and functional through Power BI dashboards.

The evaluation of the system on real batches of emails from the pilot mailbox shows a substantial improvement over the classical NLP-based solution it replaces, validating both the technical feasibility and the potential operational impact of the proposed approach.

Agradecimientos

A mis padres, por haberme dado la oportunidad de estudiar este máster y por haberme apoyado incondicionalmente en cada decisión que he tomado, gracias por ser los mejores padres del mundo.

A mis hermanos, por hacer que sienta que tengo siempre un lugar seguro a donde volver esté donde esté.

A Eva, por darme todo el amor que necesitaba y más durante todo este tiempo y por ser la mejor compañera de vida que alguien podría pedir.

A mis amigos y amigas, por haber hecho de este año un camino mucho más divertido y haberme hecho sentir en casa cuando estaba lejos de ella.

A mis compañeros y compañeras de Mapfre, por haberme acogido tan bien desde el primer día y por haberme dado la oportunidad de formar parte como uno más del equipo y de aprender de profesionales tan brillantes y cercanos.

Y por último a mí mismo, por haber sido capaz de superar otro reto más y demostrarme que puedo con todo lo que me propongo.

# Autorización
## de entrega del Trabajo Fin de Máster

El autor del Trabajo Fin de Máster y los responsables que se indican a continuación autorizan su entrega y presentación.

**El autor del Trabajo Fin de Máster**

ARROYO GARCIA JAVIER - 77447862P
Firmado digitalmente por ARROYO GARCIA JAVIER - 77447862P
Fecha: 2026.06.12 12:04:27 +02'00'

Javier Arroyo García

**El responsable de la empresa**

Firmado por JIMENEZ AROCHA ANDRES - ***8209** el día 12/06/2026 con un certificado emitido por AC FNMT Usuarios

Andrés Jiménez Arocha

**El responsable del máster**

Carlos Morrás Ruiz-Falcó

Firmado por MORRAS RUIZ-FALCO CARLOS - ***0907** el día 15/06/2026 con un certificado emitido por AC FNMT Usuarios

# Índice general

# Índice de figuras

# Índice de cuadros

# Capítulo 1

# Introducción

## 1.1. Motivación y contexto empresarial

MAPFRE es una de las principales aseguradoras a nivel mundial, con presencia en más de 35 países y una operativa diaria que genera un volumen masivo de comunicaciones escritas con clientes, mediadores, peritos, talleres, abogados y otros actores del ecosistema asegurador. En el contexto de la gestión de siniestros y prestaciones, el correo electrónico constituye uno de los canales de entrada de información más críticos: a pesar del avance de los formularios web y las aplicaciones digitales, el correo electrónico sigue siendo, por su antigüedad y por la inercia de los sistemas y procesos empresariales, la vía principal de comunicación entre organizaciones. Tanto los sistemas internos como los operarios están ampliamente acostumbrados a su uso, lo que dificulta su sustitución y refuerza la necesidad de soluciones que optimicen su gestión.

La criticidad de la información que se recibe a través de este canal, combinada con el elevado volumen de correos y la heterogeneidad de los adjuntos asociados como facturas, informes periciales, certificados médicos, declaraciones de accidente, comunicaciones judiciales, fotografías de daños, entre muchos otros, ha convertido la gestión manual en un cuello de botella operativo. En el marco de este proyecto, cada correo recibido debe ser revisado, sus adjuntos clasificados y renombrados conforme al criterio del sistema transaccional utilizado, y la información clave (el número de referencia del siniestro) extraída e introducida en el flujo correspondiente. Ejecutado de forma manual, este proceso tiene un impacto directo sobre la eficiencia operativa, los costes asociados a la gestión y, en última instancia, sobre la experiencia del cliente, cuya solicitud puede verse demorada por tareas que no aportan valor diferencial.

En este escenario, la automatización inteligente de la gestión de correos electrónicos se presenta como una oportunidad estratégica para liberar tiempo de los tra-

mitadores hacia tareas de mayor valor añadido, reducir costes operativos y acortar los tiempos de respuesta al cliente. Bajo este marco, el presente Trabajo de Fin de Máster aborda el diseño e implementación de una solución basada en modelos de lenguaje de gran escala (Large Language Models, LLMs) para la gestión automatizada de buzones de correo corporativos, con el objetivo de transformar un proceso tradicionalmente manual en un flujo de trabajo inteligente, escalable y adaptable a las necesidades cambiantes de la organización.

Cabe destacar que MAPFRE no parte de cero en este ámbito. La compañía dispone actualmente de una solución en producción para la gestión automatizada de correos electrónicos, basada en técnicas de Procesamiento del Lenguaje Natural (NLP) clásico. Sin embargo, esta solución ha demostrado ser insuficiente para abordar la complejidad y diversidad reales del problema: la variabilidad lingüística de los mensajes, el lenguaje informal y los errores tipográficos habituales en las comunicaciones, la naturaleza heterogénea de los adjuntos y la aparición de nuevos formatos documentales rebasan las capacidades de los algoritmos tradicionales, diseñados para escenarios con vocabulario controlado y categorías estables. Durante el año 2025, el sistema procesó 1.237.947 correos provenientes de 8 buzones diferentes, alcanzando una tasa de éxito por correo del 25 %. Esto implica que tres de cada cuatro correos requerían intervención manual por parte de los gestores, con el consiguiente coste operativo y la pérdida del valor que una automatización efectiva podría aportar.

El ya comentado bajo rendimiento, unido a la creciente sofisticación de los documentos recibidos y a la necesidad de incorporar nuevos buzones al ámbito de la automatización, deja clara la necesidad de evolucionar hacia una solución basada en modelos de lenguaje de gran escala. A diferencia del NLP clásico, los LLMs son capaces de comprender el contenido semántico de los correos y sus adjuntos de forma robusta, generalizable y adaptable a nuevos contextos sin necesidad de reentrenamientos específicos, lo que los convierte en candidatos naturales para abordar los retos que las soluciones anteriores no han podido resolver.

El proyecto se enmarca dentro del equipo de Transformación Operativa del área de Data & IA de MAPFRE y toma como caso de uso concreto el buzón de prestaciones de autos, que actúa como piloto de validación y el cual está directamente relacionado con el sistema transaccional conocido como TRAMES. Este buzón ha sido seleccionado por su alto volumen y por la diversidad de los adjuntos recibidos, condiciones que lo convierten en un escenario representativo y exigente sobre el que validar la viabilidad de la solución. No obstante, el diseño de la herramienta está orientado a ser transversal: cualquier nuevo buzón corporativo debería poder incorporarse al sistema con un esfuerzo mínimo de configuración, sin necesidad de rediseñar la arquitectura subyacente. Esta orientación a la generalización es, junto con el rendimiento de la clasificación, uno de los pilares principales del trabajo

y condiciona buena parte de las decisiones de diseño descritas en los capítulos posteriores.

Tras describir las acciones que deberán ejecutarse para alcanzar los objetivos planteados, se ha llevado a cabo un análisis de tiempos medios de operación (TMO) sobre el proceso manual, identificando así las siguientes operaciones como las de mayor impacto en el tiempo de gestión de cada correo y que por tanto se han priorizado a lo largo del proyecto:

- **Extracción del número de referencia del siniestro** cuando no figura en el asunto del correo: es la operación individualmente más costosa en tiempo, ya que requiere que el tramitador localice manualmente el número de referencia en el cuerpo del mensaje o en los adjuntos.

- **Transformación de formatos**: especialmente la descompresión de archivos ZIP y la extracción de correos encapsulados en formato `.eml`, que implican múltiples pasos manuales para acceder al contenido real de los adjuntos.

- **Transformación de imágenes** en formatos no soportados por TRAMES, como HEIC, que requieren conversión previa antes de poder adjuntarse al expediente.

- **Clasificación individual de adjuntos**: cada adjunto debe ser identificado, categorizado según el catálogo de tipos documentales y renombrado conforme al esquema que interpreta TRAMES.

- **Otras transformaciones**: cualquier otra operación de renombrado, reordenación o adaptación de formato necesaria para que los adjuntos sean procesables por TRAMES.

## 1.2. Objetivos del proyecto

El objetivo principal de este Trabajo de Fin de Máster es el diseño e implementación de un sistema robusto, escalable y generalizable de gestión automática de correos electrónicos basado en LLMs, desplegado sobre una arquitectura serverless en AWS. Dicho sistema tiene el nombre de Email Management y será referenciado a lo largo del documento como **EMM**.

El sistema desarrollado debe ser capaz de recibir correos electrónicos, extraer y procesar la información contenida en ellos y reenviar el correo con dicha información procesada a un buzón de destino. Para alcanzar este objetivo, se han definido los siguientes objetivos específicos:

- **Clasificación automática de documentos**: identificar y categorizar los adjuntos incluidos en cada correo según un catálogo de tipos documentales definido, extrayendo información clave como el número de referencia del siniestro.
- **Gestión del reenvío**: a partir de la clasificación obtenida, determinar automáticamente si los adjuntos deben reenviarse en su formato original o procesado, según la lógica definida por el grupo de acción y control. En nuestro caso de uso (prestaciones de autos), el correo procesado se reenvía a un buzón de destino del que lee el sistema transaccional **TRAMES**, completando así el flujo de entrada de información sin intervención manual.
- **Monitorización y control**: supervisar el funcionamiento del sistema tanto a nivel técnico, mediante AWS CloudWatch, como a nivel funcional a través de dashboards en Power BI, permitiendo hacer un seguimiento exhaustivo del estado del procesamiento y actuar ante incidencias.
- **Resiliencia y plan de contingencia**: garantizar la continuidad operativa ante fallos en cualquier componente del sistema, minimizando la pérdida de correos.
- **Generalización**: diseñar el sistema de forma que su implantación en un nuevo buzón requiera el mínimo esfuerzo de adaptación, sentando las bases para su extensión a otros ramos de negocio.
- **Integración futura con sistemas transaccionales**: aunque fuera del alcance de este trabajo, la arquitectura está concebida para permitir la conexión vía API con los sistemas transaccionales de MAPFRE, completando así el flujo de automatización extremo a extremo.
- **Seguridad y cumplimiento normativo**:el sistema se ha diseñado bajo un marco de seguridad, gobierno del dato y cumplimiento normativo, alineado con los requerimientos aplicables del EU AI Act.

## 1.3. Estructura del documento

El presente documento se organiza en los siguientes capítulos:

- **Capítulo 2 - Estado del arte**: revisión del estado actual de la gestión automatizada de correo electrónico, los fundamentos y panorama actual de los LLMs, los modelos multimodales, la comparativa entre fine-tuning y prompt engineering, las técnicas de prompt engineering, los métodos de extracción de texto mediante OCR y las arquitecturas serverless en AWS.

- **Capítulo 3 - Análisis de requisitos**: descripción de los requisitos funcionales del sistema, incluyendo la definición del grupo de acción y control que permite la monitorización del sistema.
- **Capítulo 4 - Diseño de la arquitectura**: visión general de la arquitectura propuesta, justificación de las principales decisiones de diseño arquitectónico adoptadas y descripción de la configuración del entorno de IA sobre el que se apoya el sistema.
- **Capítulo 5 - Implementación**: desarrollo detallado de los distintos componentes del sistema desplegado sobre AWS, incluyendo las funciones de extracción de texto y clasificación documental (`extract_text`) y reenvío (`email_forward`), el plan de contingencia y la monitorización técnica y funcional.
- **Capítulo 6 - Evaluación**: metodología de evaluación adoptada, resultados de la validación técnica inicial, análisis de las dos pruebas de concepto (POC 1 y POC 2) ejecutadas sobre el buzón de prestaciones de autos y comparativa de los resultados obtenidos frente a la solución previa basada en NLP clásico.
- **Capítulo 7 - Conclusiones y trabajo futuro**: conclusiones extraídas del proyecto, limitaciones identificadas y líneas de trabajo futuro para la evolución del sistema.

# Capítulo 2

# Estado del arte

Este capítulo recoge los fundamentos teóricos y tecnológicos sobre los que se asienta el sistema propuesto. Se parte de una revisión del estado actual de la gestión automatizada del correo electrónico, para a continuación introducir los modelos de lenguaje de gran escala, su panorama actual y sus variantes multimodales. Posteriormente se comparan las estrategias de adaptación de estos modelos (fine-tuning frente a prompt engineering) y se profundiza en las técnicas concretas de prompt engineering empleadas. Finalmente, se describen las soluciones de extracción de texto mediante OCR y las arquitecturas serverless en AWS que conforman el sustrato tecnológico del sistema.

## 2.1. Gestión automatizada de correo electrónico

El correo electrónico sigue siendo uno de los canales de comunicación empresarial más utilizados a nivel mundial. En entornos corporativos de alta demanda, como el sector asegurador, sabemos que la gestión manual de estos correos representa un reto operativo significativo, especialmente cuando los mensajes incluyen adjuntos de naturaleza heterogénea que requieren interpretación y clasificación.

Las primeras aproximaciones a la automatización de la gestión del correo electrónico se basaron en reglas deterministas: filtros por remitente, palabras clave en el asunto o expresiones regulares sobre el cuerpo del mensaje [Malone et al., 1987]. Si bien estas soluciones son simples y predecibles, presentan una escalabilidad limitada y requieren un mantenimiento continuo ante cualquier variación en el formato o vocabulario de los correos [Mackay, 1988, Sahami et al., 1998].

Posteriormente, la incorporación de técnicas de Procesamiento del Lenguaje Natural (NLP) clásico permitió avanzar hacia sistemas capaces de clasificar correos en función de su contenido semántico. Algoritmos como Naive Bayes, Support Vector Machines (SVM) o modelos basados en TF-IDF demostraron

ser efectivos en escenarios con vocabulario controlado y categorías bien definidas [AlShaikh et al., 2025]. Sin embargo, su rendimiento se degrada ante la variabilidad lingüística, el lenguaje informal, los documentos adjuntos en múltiples formatos o los correos multilingües, limitaciones que los hacen insuficientes para entornos empresariales complejos [Li et al., 2020, AlShaikh et al., 2025].

La irrupción de los modelos de lenguaje de gran escala ha abierto una nueva etapa en la automatización del correo electrónico. Su capacidad para comprender el contexto, razonar sobre el contenido de documentos adjuntos y generar respuestas estructuradas los convierte en grandes candidatos para mejorar aquello que las soluciones anteriores no han podido resolver.

## 2.2. Large Language Models: arquitectura y capacidades

Los modelos de lenguaje de gran escala (LLMs) son sistemas de inteligencia artificial entrenados sobre corpus masivos de texto con el objetivo de modelar la distribución del lenguaje natural. Su arquitectura está basada en el mecanismo de atención (*Transformer*) propuesto por [Vaswani et al., 2017], que permite capturar dependencias a larga distancia entre tokens de forma eficiente mediante operaciones de atención multi-cabeza (*multi-head attention*).

A diferencia de los modelos de NLP clásico, los LLMs no requieren un entrenamiento específico para cada tarea. Mediante el paradigma de *pretraining* seguido de *fine-tuning* o *prompting*, un único modelo puede adaptarse a tareas de clasificación, extracción de información, resumen, traducción o generación de texto [Sahoo et al., 2024]. Esta generalidad es precisamente lo que los hace especialmente valiosos en contextos donde la variabilidad de las entradas es alta.

En el contexto de este proyecto, se ha utilizado **Gemini 2.5 Flash**, modelo multimodal de Google DeepMind optimizado para tareas de razonamiento con una ventana de contexto extensa y soporte nativo para documentos e imágenes, lo que lo hace especialmente adecuado para el procesamiento de correos con adjuntos heterogéneos.

Una característica clave de los LLMs modernos es su capacidad **multimodal**: además de texto, modelos como Gemini 2.5 Flash son capaces de procesar imágenes y documentos directamente, lo que reduce la necesidad de pipelines de extracción intermedios en muchos casos.

## 2.3. Modelos multimodales

Los primeros LLMs operaban exclusivamente sobre texto. Sin embargo, la creciente necesidad de procesar información en múltiples formatos ha impulsado el desarrollo de los modelos multimodales (*Multimodal Large Language Models*, MLLMs), capaces de integrar de forma nativa texto, imágenes, audio y documentos en un mismo modelo [Yin et al., 2024].

La arquitectura de estos modelos extiende el Transformer original incorporando encoders específicos para cada modalidad, habitualmente un encoder visual basado en Vision Transformer (ViT), cuya salida se proyecta al espacio de representación del LLM mediante capas de alineamiento [Cornia et al., 2024]. Esto permite al modelo razonar conjuntamente sobre el contenido textual y visual de una entrada, sin necesidad de pipelines separados para cada tipo de dato.

Esta capacidad resulta especialmente relevante en el contexto de la gestión de correos electrónicos corporativos, donde los adjuntos pueden presentarse en formatos heterogéneos: documentos PDF, imágenes de facturas o fotografías de daños, entre otros. Un modelo multimodal puede procesar directamente estos adjuntos sin depender exclusivamente de extractores de texto intermedios, reduciendo la complejidad del pipeline y los posibles puntos de fallo.

## 2.4. Panorama actual de LLMs

El ecosistema de LLMs disponibles actualmente puede clasificarse en dos grandes categorías: modelos propietarios y modelos de código abierto.

Entre los modelos propietarios más relevantes destacan la familia **GPT** de OpenAI, con GPT-5.5 como su versión fundacional más avanzada en el momento de redacción de este trabajo, la familia **Gemini** de Google DeepMind con Gemini 2.5 Flash y Gemini 3.1 Pro y la familia **Claude** de Anthropic. Estos modelos se caracterizan por su alto rendimiento en *benchmarks* estándar, su soporte multimodal nativo y su accesibilidad a través de APIs. En el ámbito del código abierto, la familia **LLaMA** de Meta ha democratizado el acceso a modelos de gran escala, permitiendo su despliegue en infraestructura propia sin dependencia de proveedores externos. Esta opción resulta atractiva en entornos corporativos con requisitos estrictos de privacidad de datos, aunque a costa de una mayor complejidad operativa, ya que requiere una gestión especializada de los recursos y la infraestructura subyacente.

La elección entre estas opciones depende de un conjunto de factores técnicos y operativos: el rendimiento en la tarea concreta, el coste por token, la latencia de respuesta, la ventana de contexto disponible, el soporte multimodal nativo y las restricciones de privacidad y residencia de datos del entorno de despliegue.

En este trabajo, aplicado al entorno corporativo de MAPFRE, la combinación de estos factores motivó la elección de **Gemini 2.5 Flash** [Comanici et al., 2025]. La justificación técnica y económica de esta decisión frente a las alternativas evaluadas (como GPT-5.5, Gemini 3.1 Pro, Claude y LLaMA) se sustenta sobre tres argumentos clave:

1. **Capacidad de contexto ultra-larga y viabilidad de recuperación:** El reporte oficial de Google DeepMind confirma que Gemini 2.5 Flash ofrece de forma nativa una ventana de contexto de 1 millón de tokens [Comanici et al., 2025]. Si bien lanzamientos recientes como GPT-5.5 de OpenAI han igualado esta magnitud alcanzando un límite de 1.05 millones de tokens, y Gemini 3.1 Pro ofrece soporte de 1 a 2 millones de tokens, la ventaja decisiva de Gemini 2.5 Flash en este ámbito es que mantiene una tasa de recuperación casi perfecta (superior al 99.7 %) en tareas de tipo *needle-in-a-haystack* (extracción de datos en grandes volúmenes de información) con un volumen de memoria operativa (1M) que ya resulta completamente suficiente para la tarea propuesta. Para MAPFRE, esta capacidad permite procesar un correo electrónico completo junto con múltiples archivos adjuntos de gran extensión en una única llamada, evitando asumir los altos sobrecostes de modelos frontera solo por su ventana de memoria bruta.

2. **Soporte multimodal nativo:** Gemini 2.5 Flash procesa de manera nativa entradas de texto, imágenes y PDF estructurados [Comanici et al., 2025]. A diferencia de arquitecturas históricas que exigían un software de Reconocimiento Óptico de Caracteres (OCR) como paso intermedio, los modelos frontera de última generación ya comentados asimilan la información directamente desde la entrada visual y documental. Emplear esta versión de Gemini (2.5 Flash) permite ejecutar esta característica multimodal de forma ultraligera, reduciendo significativamente la latencia del sistema sin la penalización de utilizar un modelo de razonamiento agentivo pesado para tareas de extracción directa.

3. **Coste operativo y eficiencia de escala:** Como opción específicamente optimizada para coste y velocidad en producción, Flash maximiza el rendimiento a una fracción de los requisitos de cómputo habituales [Comanici et al., 2025]. En términos financieros, el impacto es muy notable: la API de Gemini 2.5 Flash tiene un coste de $0.30 USD por millón de tokens de entrada y $2.50 USD por millón de tokens de salida. En comparación, GPT-5.5 impone un coste estándar de $5.00 USD en entrada y $30.00 USD en salida (tarifas que se duplican y multiplican al superar los 272 K tokens), mientras que Gemini 3.1 Pro presenta un coste de $2.00 USD en entrada y $12.00 USD en salida

(valores que ascienden para contextos mayores a 200 K tokens). Esto convierte a la opción seleccionada (2.5 Flash) en una alternativa casi 7 veces más barata en la fase de entrada frente a Gemini 3.1 Pro ($0.30 frente a $2.00), y hasta 16 veces más económica frente a GPT-5.5 para correos convencionales. Esta ventaja en ahorro es el factor decisivo que hace viable un proyecto de automatización corporativa como el de nuestro caso de uso, con un alto rendimiento y gran volumen.

## 2.5. Fine-tuning vs. prompt engineering

A la hora de adaptar un LLM a una tarea específica, existen dos aproximaciones principales: el *fine-tuning* y el *prompt engineering*.

El *fine-tuning* consiste en reentrenar el modelo, total o parcialmente, sobre un conjunto de datos etiquetados específico de la tarea, es decir, partimos de un modelo preentrenado con un corpus de datos general y lo adaptamos a la tarea específica. Esta aproximación puede lograr un alto rendimiento cuando se dispone de suficientes datos de entrenamiento, tiempo de ejecución y además la tarea presenta características muy particulares que difieren del comportamiento general del modelo [Vangibhurathachhi, 2025].

El *prompt engineering*, por el contrario, opera exclusivamente sobre el texto de entrada sin modificar los pesos del modelo, es decir, no se basa en un reentrenamiento sino que utiliza el modelo base. Su principal ventaja es la rapidez de iteración: un cambio en el prompt puede evaluarse de inmediato sin ningún proceso de entrenamiento, lo que lo convierte en la opción preferida cuando el tiempo de desarrollo es limitado o el modelo del que partimos se adecúa en complejidad a la tarea asignada [Sahoo et al., 2024].

En el contexto de este proyecto, la necesidad de iterar rápidamente sobre los requisitos de negocio, junto con la complejidad reducida de la tarea de clasificación de correos, hizo del prompt engineering la aproximación más adecuada.

## 2.6. Prompt engineering

El *prompt engineering* es la disciplina que estudia cómo formular las instrucciones que se proporcionan a un LLM para obtener respuestas precisas, consistentes y estructuradas [Sahoo et al., 2024]. Como ya comentamos, a diferencia del *fine-tuning*, que requiere reentrenar el modelo con datos etiquetados, el *prompt engineering* opera exclusivamente sobre el texto de entrada (denominado contexto), lo que lo convierte en una técnica de bajo coste computacional y alta iterabilidad.

Entre las técnicas más relevantes se encuentran:

- **Zero-shot prompting**: consiste en proporcionar al modelo únicamente la instrucción y el texto a procesar, sin ejemplos previos. Aunque es la aproximación más simple y suficiente cuando la tarea está bien definida [Sahoo et al., 2024], presenta una elevada sensibilidad a variaciones sutiles en la formulación de la instrucción y sufre de sesgos intrínsecos del modelo , lo que a menudo requiere técnicas de calibración contextual para estabilizar sus predicciones [Zhao et al., 2021].

- **Few-shot prompting** (o *In-Context Learning*): se incluyen uno o varios ejemplos de entrada-salida (*demonstrations*) dentro del contexto para guiar al modelo hacia el comportamiento esperado. Dado que el rendimiento final es bastante sensible a la selección, cantidad y orden de estos ejemplos [Zhao et al., 2021], las aproximaciones avanzadas proponen el uso de módulos de recuperación (*retrievers*) para seleccionar dinámicamente aquellos ejemplos que guarden mayor similitud semántica con la consulta del usuario [Liu et al., 2022, Dong et al., 2024].

- **Chain-of-Thought (CoT)**: instruye al modelo para que genere una secuencia de pasos de razonamiento intermedios antes de emitir la respuesta final, mejorando drásticamente el rendimiento en tareas complejas [Wei et al., 2022]. Esta técnica ha evolucionado hacia variantes optimizadas como *Self-Consistency*, que muestrea múltiples caminos de razonamiento y selecciona la respuesta más frecuente por votación mayoritaria [Wang et al., 2022], o *Auto-CoT*, que automatiza la generación de estas cadenas de razonamiento sobre los ejemplos para evitar la curación manual [Zhang et al., 2022].

- **Structured output prompting**: se especifica en el prompt un esquema de datos rígido (por ejemplo, en formato JSON o XML) para facilitar su procesamiento posterior por sistemas externos. Sin embargo, dado que las instrucciones en lenguaje natural no garantizan por sí solas la validez sintáctica del esquema bajo un flujo continuo de consultas [Sahoo et al., 2024], el estado del arte combina esta técnica con la decodificación restringida (*constrained decoding*), un método que intercepta las probabilidades de los tokens durante la generación mediante autómatas de estado finito o gramáticas independientes del contexto (CFG) para garantizar un cumplimiento estructural del 100 % [Willard and Louf, 2023].

En el contexto de este proyecto, el prompt engineering ha sido una de las palancas principales de mejora del rendimiento del sistema, siendo objeto de múltiples iteraciones a lo largo del desarrollo, como se describe en detalle en el Capítulo 4, en la sección 4.3. Para llevar a cabo esta metodología de trabajo, se han combinado

técnicas de zero-shot prompting, few-shot prompting y structured output prompting, con el objetivo de maximizar la precisión de las predicciones del modelo y garantizar la validez estructural de las salidas para su posterior procesamiento.

## 2.7. Extracción de texto en documentos: OCR

El Reconocimiento Óptico de Caracteres (*Optical Character Recognition*, OCR) es la tecnología que permite extraer texto legible a partir de imágenes o documentos escaneados. En el contexto de la gestión automatizada de correos electrónicos, el OCR constituye un componente esencial del pipeline cuando los adjuntos recibidos no contienen texto digital nativo, como ocurre con fotografías de documentos, PDFs escaneados o imágenes incrustadas.

Tesseract es el motor de OCR de código abierto más utilizado en la industria y la investigación [Smith, 2007]. Desarrollado originalmente por Hewlett-Packard (HP) entre 1984 y 1994 y posteriormente mantenido por Google, este motor fundamenta su arquitectura en algoritmos novedosos de segmentación de líneas (*line finding*), clasificación de características estructurales (*features*) y un clasificador adaptativo dinámico [Smith, 2007].

No obstante, Tesseract presenta limitaciones cuantitativas severas frente a alternativas modernas del estado del arte. Estudios comparativos empíricos revelan que su rendimiento se degrada de forma crítica ante layouts complejos, imágenes de baja calidad, texto sobre fondos no uniformes o escrituras complejas. Por ejemplo, se ha registrado una tasa de error de caracteres (CER, *Character Error Rate*) del 18.2 % para Tesseract en comparación con un destacado 4.5 % obtenido por herramientas avanzadas como Paddle OCR, lo que representa un margen de error aproximadamente cuatro veces superior [Sharma et al., 2025]. Asimismo, la literatura contemporánea en comprensión de documentos libre de OCR (*OCR-free document understanding*) resalta que los métodos de segmentación clásica e indexación de caracteres introducen fallas acumulativas en el pipeline, lo que compromete la extracción precisa de entidades semánticas [Park et al., 2024].

A pesar de estas limitaciones de precisión léxica, Tesseract conserva ventajas operativas críticas que justifican su presencia en entornos de producción: su tamaño compacto (aproximadamente 10 MB), su alta velocidad de ejecución en CPU (que requiere un tiempo sub-segundo por imagen) y su capacidad para operar de manera local y en el *edge* sin depender de APIs de red. Estas propiedades lo convierten en un mecanismo de contingencia (*fallback*) y redundancia sumamente coste-efectivo cuando los modelos multimodales masivos sufren de latencia o inestabilidad, cuando agotan su cuota de peticiones, o al procesar documentos triviales y maquetados de forma sencilla, donde un MLLM representaría una solución sobredimensionada e ineficiente.

En el sistema descrito en este trabajo, Tesseract se integra dentro de una estrategia de extracción híbrida en la que coexiste y se complementa con las capacidades multimodales nativas del LLM. Esta estructura está ampliamente respaldada por la literatura científica reciente; estudios como el de Greif et al. [Greif et al., 2024] demuestran que la combinación de un motor de OCR clásico con una fase de post-corrección de texto ruidoso ejecutada por un modelo de lenguaje multimodal (como Gemini 2.5 Flash) produce las tasas de error más bajas en documentos heterogéneos, superando tanto al uso exclusivo del OCR como al procesamiento directo y puro del MLLM. Esta evidencia justifica directamente la arquitectura híbrida adoptada en el caso de uso de MAPFRE, donde el modelo multimodal actúa como el motor de procesamiento principal, apoyándose en Tesseract cuando es necesario.

## 2.8. Arquitecturas serverless en AWS

El paradigma *serverless* hace referencia a un modelo de computación en la nube en el que el desarrollador no gestiona directamente la infraestructura subyacente. En lugar de aprovisionar y mantener servidores, las funciones se ejecutan bajo demanda en respuesta a eventos, y el proveedor cloud se encarga del escalado, la disponibilidad y el mantenimiento [Jonas et al., 2019]. De acuerdo con Jonas et al. [Jonas et al., 2019], la adopción de *serverless* equivale al salto que se dio en su día del lenguaje ensamblador a los lenguajes de alto nivel, y se perfila como el estándar del futuro en la nube.

AWS Lambda es el servicio de computación *serverless* de tipo FaaS (*Function-as-a-Service*) de Amazon Web Services, introducido en 2014 como la primera plataforma comercial de este tipo [Jangda et al., 2019]. Permite ejecutar código en respuesta a eventos, como la llegada de un mensaje a una cola, una petición HTTP o un evento programado, sin necesidad de gestionar servidores. El análisis llevado a cabo por el equipo ya mencionado en Jangda et al. ([Jangda et al., 2019]) resulta especialmente útil para razonar sobre la corrección del código en entornos de producción, donde influyen fenómenos operacionales de bajo nivel como la reutilización de entornos de ejecución y los arranques en caliente (*warm starts*).

Las ventajas de este modelo para un sistema de gestión de correos electrónicos son evidentes. El flujo de mensajes entrantes en un entorno corporativo es inherentemente variable e impredecible: presenta picos de tráfico durante las horas laborables y valles prolongados por las noches y fines de semana. Un modelo de aprovisionamiento clásico basado en instancias siempre activas significaría costes ociosos innecesarios. En su lugar, AWS Lambda ofrece un esquema de escala-a-cero (coste nulo en ausencia de carga) con facturación fraccionada al milisegundo de ejecución real. Además, el servicio cuenta de manera predeterminada con un límite

de concurrencia de 1,000 ejecuciones simultáneas por región y una capacidad de escalado en ráfaga (*burst scaling*) de 1,000 instancias adicionales cada 10 segundos, garantizando la absorción inmediata de picos masivos de correspondencia.

Para construir un pipeline *event-driven* robusto, Lambda se combina con un ecosistema de servicios auxiliares categorizados como *Backend-as-a-Service* (BaaS) [Jonas et al., 2019]. En la arquitectura propuesta, se integra **Amazon S3** para el almacenamiento duradero de objetos (correos y adjuntos) y **Amazon DynamoDB** para la persistencia de metadatos. La elección de DynamoDB frente a bases de datos relacionales o documentales tradicionales se justifica por su naturaleza *serverless-nativo* (sin gestión de instancias), latencias de lectura/escritura en el rango de un solo dígito de milisegundo a cualquier escala, y la capacidad de reaccionar en tiempo real mediante disparadores automáticos a través de DynamoDB Streams. El sistema se complementa con la suite de monitorización **Amazon CloudWatch** para observabilidad integral y **AWS Secrets Manager** para el almacenamiento y rotación segura de credenciales, siguiendo las directrices de seguridad y resiliencia del marco de buenas prácticas *AWS Well-Architected Framework*.

Finalmente, para evitar el fenómeno de la deriva de configuración o *configuration drift* (fenómeno mediante el cual la configuración real de un entorno de TI se desvía gradualmente del estado definido y documentado originalmente) en entornos de producción, la infraestructura del sistema ha sido modelada e implementada en su totalidad mediante **Terraform**, una herramienta declarativa de Infraestructura como Código (IaC) de gran adopción en la industria [Verdet et al., 2024, Morris, 2020]. Frente a herramientas propietarias como AWS CloudFormation o AWS CDK, Terraform aporta un valor estratégico clave sustentado en:

- **Agnosticismo cloud:** Su arquitectura basada en proveedores permite definir recursos de múltiples nubes (AWS, GCP, Azure) bajo una misma base de código, garantizando la portabilidad del pipeline ante eventuales estrategias multi-cloud.

- **Determinismo y estado unificado:** La gestión mediante un archivo de estado (*state file*) unificado facilita la detección de discrepancias mediante planes de ejecución (*terraform plan*) deterministas previos a la aplicación real de cambios.

- **Ecosistema y mantenibilidad:** Utiliza el lenguaje declarativo HCL (*HashiCorp Configuration Language*), el cual resulta sustancialmente más legible y mantenible que las representaciones JSON o YAML de CloudFormation, mitigando la introducción de fallos de seguridad [Verdet et al., 2024].

## 2.9. Conclusión

En síntesis, este capítulo ha establecido la base técnica y conceptual que permite abordar la automatización de la gestión del correo electrónico en el sector asegurador desde una perspectiva de vanguardia. La convergencia entre modelos multimodales de gran escala, como **Gemini 2.5 Flash**, y arquitecturas *serverless* en AWS, permite superar las limitaciones de los sistemas legados y los enfoques de NLP tradicional.

A través de esta revisión, se ha justificado la viabilidad del *prompt engineering* como motor de adaptación frente al *fine-tuning*, así como la pertinencia de una arquitectura híbrida donde el LLM actúa como procesador principal, apoyado por mecanismos clásicos de OCR como Tesseract para garantizar la resiliencia del sistema. Finalmente, la adopción de un enfoque *Infrastructure as Code* (IaC) mediante Terraform asegura que la arquitectura propuesta no solo sea eficiente y escalable, sino también auditable y resistente a la deriva de configuración.

Con estos cimientos definidos, el proyecto cuenta con el respaldo teórico y tecnológico necesario para proceder, en los siguientes capítulos, al diseño detallado y la implementación práctica de la solución propuesta.

# Capítulo 3

# Análisis de Requisitos

Este capítulo recoge el proceso de análisis del sistema de gestión automatizada de correo electrónico desarrollado para MAPFRE. El punto de partida es el contexto operativo real: un buzón corporativo del área de prestaciones de autos que recibe un volumen elevado y variable de correos con adjuntos heterogéneos, cuya gestión manual consume un tiempo significativo de los tramitadores.

Cabe comentar que dicho buzón de correo es de naturaleza on premise, lo que implica ciertas limitaciones técnicas y de integración que llevaron a diseñar una solución que introduce un buzón proxy en la nube como punto de entrada del sistema, al cual llegan los correos a procesar desde el buzón on premise y desde el cual se reenvían al buzón principal tras su procesamiento. Esta arquitectura híbrida es un aspecto clave que condiciona tanto los requisitos como las decisiones de diseño del sistema.

A partir de ese contexto, se identificaron y formalizaron los requisitos funcionales del sistema, organizados en cuatro grupos que cubren desde el procesamiento del correo y la integración con el sistema transaccional TRAMES, hasta la monitorización y el plan de continuidad de negocio. Estos requisitos constituyen la base sobre la que se asientan las decisiones de diseño arquitectónico recogidas en el Capítulo 4.

## 3.1. Requisitos funcionales y no funcionales

A continuación se recogen los requisitos identificados durante la fase de análisis del proyecto, organizados en cuatro grupos funcionales (G001–G004), así como las decisiones de diseño derivadas de dichos requisitos.

### 3.1.1. G001 – Proceso DATA&IA

**RF_001 – Lectura del Inbox y envío de la información.**
El sistema debe leer de forma continua el buzón proxy de entrada y procesar cada correo recibido. Se establecen las siguientes reglas de filtrado previo:

- Los correos cuyo remitente sea diferente al buzón principal de prestaciones de autos serán redirigidos a dicho buzón y eliminados del buzón proxy, ya que se asume que únicamente deben llegar correos procedentes de ese origen.
- Los correos que ya hayan sido procesados por el sistema, identificados por la etiqueta `[data_process_id]` en el asunto, no deben volver a procesarse.
- Los correos con un tamaño superior a 25.000 KB quedan fuera del procesamiento automático y se gestionan mediante una alarma interna con reenvío manual. Esto se debe a limitaciones técnicas de los servicios de correo y para evitar bloqueos del sistema ante casos excepcionales de correos con adjuntos extremadamente pesados.

**RF_002 – Procesamiento del correo.** Una vez recogido el correo del Inbox del buzón proxy, el sistema ejecuta el siguiente flujo de procesamiento:

1. **Asignación del identificador único.** Se genera un `data_process_id` único por correo, compuesto por caracteres alfanuméricos de máximo 6 caracteres seguidos de una letra (ejemplo: `260421a143055a`). El formato está diseñado deliberadamente para evitar que TRAMES lo interprete como número de referencia, ya que este sistema rechaza cadenas de 8 dígitos consecutivos.
2. **Extracción de adjuntos.** Se recogen el asunto, cuerpo y todos los adjuntos del correo, incluyendo imágenes incrustadas en el cuerpo y documentos anidados (correos encapsulados, ZIPs). Para los casos de recursividad, se itera hasta extraer todos los documentos posibles. Las imágenes de resolución inferior a 128x128 píxeles son descartadas automáticamente por considerarse firmas o logos. Las que superan este umbral pasan por un filtro adicional basado en IA generativa para detectar y descartar logos y firmas antes de la clasificación.
3. **Validación y transformación de formatos.** Se comprueban los formatos aceptados por TRAMES. Los documentos en formato no válido que puedan transformarse son convertidos (por ejemplo, HEIC a JPG). Los documentos cifrados o que generen error durante la extracción son preservados para el reenvío sin clasificar.

4. **Clasificación de adjuntos.** Para cada adjunto válido se realiza una llamada al modelo de IA generativa, proporcionándole las descripciones del catálogo de tipos documentales definido y proporcionado por el equipo de negocio. Dicha clasificación consta de un nombre de categoría (por ejemplo, `INFORME PERICIAL`) que viene asociado a un código numérico (por ejemplo, `001`) que es el que se utiliza posteriormente para el renombrado de los documentos ya que es el que realmente interpreta TRAMES. El modelo devuelve la categoría más probable para cada documento y en caso de no asociarse a ninguna categoría o producirse un fallo del modelo, el adjunto recibe la categoría `DESCONOCIDO`.

5. **Renombrado de documentos.** Cada documento clasificado es renombrado siguiendo el esquema `#código#-data_ia-nombrefichero.ext`, donde el código corresponde a la tipología asignada y es leído posteriormente por TRAMES. Los documentos clasificados como `DESCONOCIDO` siguen el esquema `DESCONOCIDO-data_ia-nombrefichero.ext`. La etiqueta `data_ia` permite a los tramitadores identificar qué documentos han pasado por el sistema.

6. **Extracción del número de referencia.** Se busca el número de referencia del siniestro en el asunto, cuerpo y adjuntos clasificados, aplicando dos patrones: 10 dígitos comenzando por 049/050/051/052/053/054/055/056, o 9 dígitos comenzando por 49/50/51/52/53/54/55/56. La búsqueda de dicho número de referencia se hace en dos fases:

   - **Fase 1.** Búsqueda directa mediante expresiones regulares en el asunto y cuerpo del correo.
   - **Fase 2.** Búsqueda en los adjuntos clasificados mediante una llamada al modelo de IA generativa, que analiza el contenido de los documentos y devuelve el número de referencia si lo encuentra.

   Una vez realizada esta búsqueda, en caso de que se encuentren varios números de referencia, se da prioridad a aquellos encontrados en el asunto y cuerpo del correo sobre los encontrados en los adjuntos, debido a reglas de negocio.

7. **Composición del asunto.** El asunto del correo reenviado incluye, en orden: el `data_process_id`, el número de referencia (si se ha encontrado) y el asunto original. El asunto resultante se trunca a 255 caracteres ya que el servicio de Microsoft 365 no acepta asuntos de más de dicha longitud. Este truncamiento se hará preservando siempre el número de referencia.

8. **Detección de SPAM.** El sistema evalúa si el correo debe marcarse como SPAM según criterios definidos por negocio, añadiendo en ese caso la etiqueta `[SPAM]` al asunto.

9. **Reenvío.** El correo se reenvía mediante la acción de "responder" al buzón principal de prestaciones de autos, con todos los adjuntos clasificados y no clasificados incluidos. Esta acción de responder permite que el correo reenviado conserve el mismo ID de mensaje en el buzón principal, lo que facilita su trazabilidad y evita problemas de duplicados en TRAMES. El correo reenviado se etiqueta con `[data_process_id]` en el asunto para evitar reprocesamientos. En el caso del grupo de control, se reenvía el correo en su formato original del correo y todo el hilo de conversación que pudiese venir del correo sin clasificar ni renombrar los adjuntos.

10. **Eliminación.** Tras el reenvío exitoso, el correo es eliminado permanentemente del Inbox del buzón proxy.

**RF_003 – Generación del Grupo de Control.**

Por requerimientos normativos del área DATA&IA y para medir el ROI (retorno de la inversión) del proyecto, se establece un grupo de control. El **10 %** de los correos entrantes no serán categorizados por el sistema de inteligencia artificial, siendo etiquetados únicamente con el identificador `[data_process_id]` en el asunto y sin clasificación de adjuntos, para que los gestores los procesen manualmente. El **90 %** restante sigue el flujo automatizado completo. Esta división permite cuantificar el tiempo real ahorrado a los gestores y validar el impacto del sistema de forma objetiva.

**RF_004 – Generación de reglas de filtrado en el buzón proxy.**

El sistema requiere la configuración de reglas en el buzón proxy que garanticen que únicamente lleguen al Inbox los correos susceptibles de ser procesados por DATA&IA, redirigiendo al buzón principal aquellos que no cumplan los criterios definidos.

**RF_005 – Control de llenado del buzón proxy (alarmas).**

El sistema debe monitorizar el nivel de ocupación del buzón proxy y generar alarmas ante situaciones de llenado que puedan comprometer la recepción de nuevos correos.

**RF_006 – Control de llenado del buzón proxy (eliminación de correos).**

Los correos ya procesados deben ser eliminados del buzón proxy de forma periódica para evitar su acumulación y garantizar la disponibilidad de espacio.

### 3.1.2. G002 – Funcionamiento de TRAMES, buzón principal e incidencias

**RF_007 – Reglas en el buzón principal.**

El buzón principal de prestaciones de autos debe disponer de reglas de enrutamiento que dirijan cada correo entrante a la carpeta o destino correspondiente según su naturaleza.

**RF_008 – Funcionamiento de TRAMES en el buzón principal.**

TRAMES es el sistema transaccional que lee el Inbox del buzón principal. Para que un correo sea procesado correctamente por TRAMES deben cumplirse las siguientes condiciones:

- El número de referencia del siniestro debe aparecer en el asunto del correo.
- Dicho número de referencia debe existir previamente en TRAMES.
- Todos los adjuntos deben estar en formatos digitalizables aceptados: `.txt`, `.doc`, `.pdf`, `.tif`, `.tiff`, `.jpeg`, `.jpg`, `.gif`, `.bmp`, `.rtf`, `.docx`, `.png`.

**RF_009 – Control de llenado del buzón principal.**

De forma análoga al buzón proxy, el buzón principal debe ser monitorizado para evitar situaciones de llenado que impidan la recepción de nuevos correos.

### 3.1.3. G003 – Monitorización del producto

**RF_010 – Monitorización de TRAMES.**

El sistema debe proporcionar visibilidad sobre el estado del procesamiento en TRAMES, incluyendo el volumen de correos procesados correctamente, los casos de error y los duplicados detectados.

**RF_011 – Monitorización de DATA&IA.**

El sistema debe exponer métricas técnicas y funcionales del pipeline de clasificación, incluyendo tiempos de procesamiento, tasas de clasificación por tipo documental, errores producidos y estado de los componentes. Esta monitorización se articula en dos capas: una técnica mediante AWS CloudWatch y una funcional mediante dashboards en Power BI.

### 3.1.4. G004 – Plan de continuidad de negocio

**RF_012 – Excepciones inesperadas tras recibir el correo en DATA&IA.**

El sistema debe gestionar de forma controlada cualquier excepción producida durante el procesamiento de un correo, garantizando que ningún correo se pierda ante un fallo inesperado del pipeline.

**RF_013 – Fallo del servicio en DATA&IA.**

En caso de fallo total del servicio de DATA&IA, debe existir un plan de contingencia que permita la continuidad operativa del negocio, manteniendo la capacidad de gestión de correos aunque sea de forma manual o degradada.

## 3.2. Conclusión

A través del análisis detallado en este capítulo, se ha definido el marco funcional necesario para que el sistema de gestión de correos se integre de forma transparente en el flujo de trabajo de MAPFRE. Los requisitos establecidos (G001–G004) no solo garantizan la automatización técnica mediante IA, sino que protegen la continuidad operativa y la calidad de la información entregada a lo largo del flujo.

La definición del grupo de control y las estrategias de monitorización (explicados en profundidad en el capítulo 4) proporcionan las herramientas necesarias para evaluar, en capítulos posteriores, el impacto real de la solución tanto en términos de eficiencia como de fiabilidad. Con estos requisitos formalizados, el sistema está preparado para la fase de diseño arquitectónico, donde se concretarán los componentes tecnológicos necesarios para materializar este flujo lógico en una infraestructura técnica robusta y escalable.

# Capítulo 4

# Diseño de la arquitectura

Este capítulo recoge las decisiones de diseño adoptadas a partir de los requisitos identificados en el Capítulo 3. Se describen la arquitectura general del sistema, los componentes que forman el pipeline de procesamiento y las decisiones tecnológicas que condicionan su estructura y comportamiento.

Antes de describir los componentes técnicos del sistema, la figura 4.1 ilustra el flujo completo de procesamiento desde la perspectiva del negocio, mostrando la integración entre los tres entornos principales que intervienen en el sistema.

El flujo se inicia cuando un usuario externo envía un correo al buzón on-premise de prestaciones de autos (`prestacionesautos@mapfre.com`). En este buzón, unas reglas de filtrado previas a la bandeja de entrada determinan qué correos deben ser procesados por DATA&IA, redirigiendo al resto directamente al flujo manual. Los correos seleccionados se mueven al buzón proxy en la nube (`emmproxyllodprod@mapfre.com`), desde donde el pipeline de DATA&IA los procesa: lee y analiza el correo, invoca las capacidades de IAGEN para la categorización de adjuntos y la extracción del número de referencia, y reenvía el correo procesado al buzón de destino. Desde allí, TRAMES lee el inbox, localiza el número de referencia en el asunto y asocia automáticamente los adjuntos clasificados al expediente correspondiente. Los correos que no pueden procesarse correctamente ya sea por formato no soportado, ausencia de número de referencia válido u otras incidencias se derivan al buzón de revisión manual (`incidenciasMAPFREods@mapfre.com`).

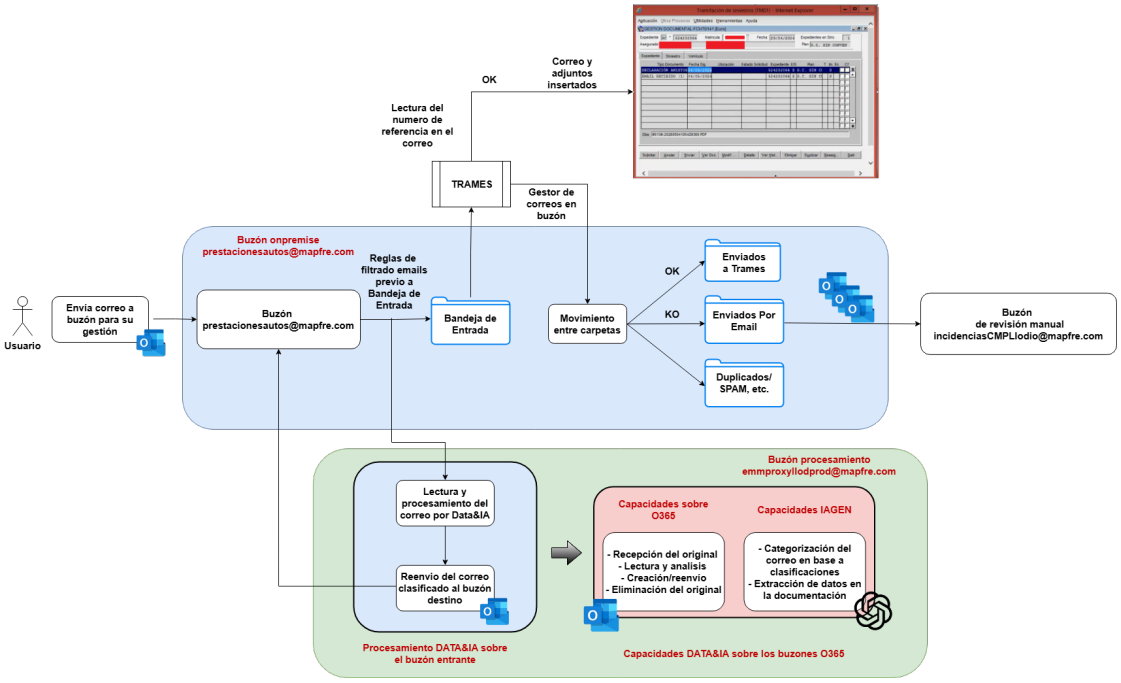


Figura 4.1: Flujo completo de procesamiento del sistema EMM

## 4.1. Visión general de la arquitectura

La figura 4.2 muestra la visión general de la arquitectura del sistema.

El diagrama refleja la separación entre los dos entornos en los que opera el sistema. En la parte inferior se sitúa el entorno **AWS Cloud**, una VPC aislada dentro de la cuenta corporativa de MAPFRE que aloja el pipeline principal: una secuencia de Lambdas especializadas que cubre desde la lectura del buzón proxy (`mailbox_reader`) hasta el reenvío al buzón principal (`email_forward`) y la posterior eliminación del correo del proxy (`email_deleter`). El pipeline se apoya en cuatro servicios gestionados de AWS: **Amazon S3** como almacén de adjuntos procesados, **DynamoDB** como registro estructurado de metadatos, **CloudWatch Logs** como capa de observabilidad técnica que centraliza los logs de ejecución de todas las Lambdas, y **Secrets Manager** para la custodia segura de las credenciales de acceso a servicios externos, evitando cualquier referencia a claves en el código fuente.

En la parte superior se representa la **AWS Net Enterprise**, la capa de red corporativa por la que transita todo el tráfico saliente del sistema hacia servicios externos. La comunicación atraviesa un *firewall* Palo Alto, garantizando que el

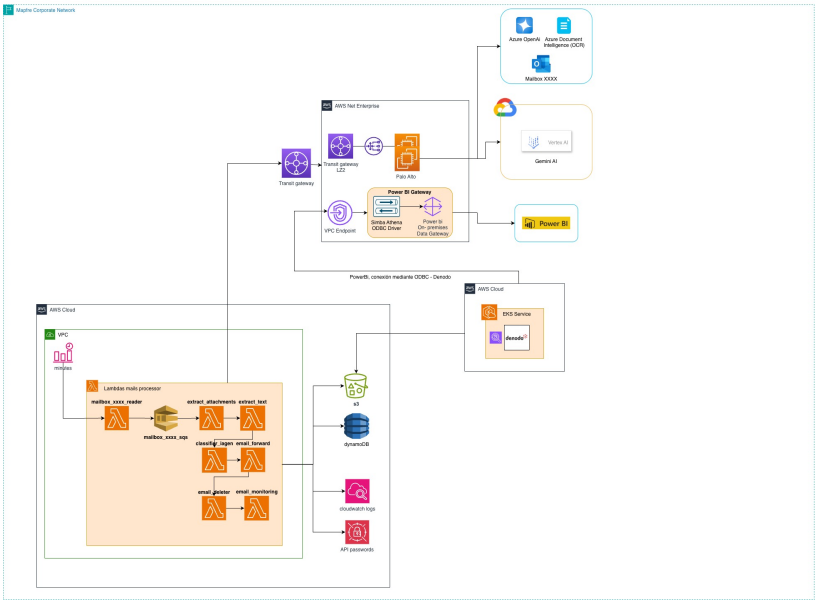


Figura 4.2: Arquitectura general del sistema EMM

sistema cumple con las políticas de seguridad perimetral de la organización, y sale hacia dos destinos principales: **Microsoft 365**, que aloja el buzón proxy y es accedido vía API Graph para la lectura de correos entrantes y el reenvío al buzón de destino; y **Vertex AI** de Google Cloud, plataforma a través de la cual se consume el modelo Gemini 2.5 Flash, encargado de la clasificación de adjuntos y la extracción del número de referencia. La extracción de texto de documentos escaneados e imágenes por medio de OCR se resuelve internamente mediante Tesseract, embebido como dependencia de la lambda `extract_text`, sin necesidad de servicios externos adicionales para esta tarea. La conexión entre ambos entornos se realiza mediante un **Transit Gateway**, que articula el enrutamiento entre la VPC del sistema, la red corporativa y los servicios externos.

Finalmente, para el reporting y seguimiento del impacto del sistema, **Power BI** se conecta a los datos almacenados en AWS a través de una capa de virtualización de datos basada en **Denodo**, desplegada sobre Amazon EKS. La conexión se realiza mediante un *gateway* ODBC expuesto a través de un VPC Endpoint, permitiendo construir cuadros de mando sobre la información del sistema sin exponer directamente la infraestructura interna.

## 4.2. Decisiones de diseño arquitectónico

Las decisiones de diseño del sistema vienen condicionadas por tres factores principales: el entorno corporativo de MAPFRE, los requisitos de escalabilidad y la necesidad de generalización a múltiples buzones.

### 4.2.1. Pipeline desacoplado por Lambdas especializadas

El pipeline sigue un patrón de **orquestación**. La ingesta se desacopla mediante una cola Amazon SQS (`mailbox_sqs`), que actúa como buffer frente a picos de carga y garantiza que ningún correo se pierda ante un fallo de las Lambdas consumidoras. La llegada de un mensaje a la cola dispara el flujo de procesamiento, orquestado mediante una **AWS Step Function** que coordina la ejecución secuencial de las Lambdas especializadas, cada una activada como un estado de la máquina de estados.

De este modo, cada Lambda conserva una responsabilidad única y completamente desacoplada a nivel de lógica de negocio, mientras que es la máquina de estados la que gestiona las transiciones del flujo. Este diseño facilita la escalabilidad, el mantenimiento y la evolución independiente de cada componente. Las principales Lambdas del pipeline son las siguientes, cada una con una breve explicación de su función que será ampliada en el capítulo de implementación:

- `mailbox_xxxx_reader`: se conecta a Microsoft 365 mediante la API Graph, obtiene los correos del buzón proxy y los encola en Amazon SQS (`mailbox_sqs`) un servicio gestionado de colas que nos permite desacoplar la ingesta del procesamiento posterior, actuando como buffer frente a picos de carga y garantizando que ningún correo se pierda en caso de fallo transitorio de las Lambdas consumidoras. Esta lambda está configurada para ejecutarse de forma periódica mediante EventBridge (un servicio de programación de eventos), con una frecuencia ajustable según las necesidades de procesamiento.

- `extract_attachments`: esta lambda se encarga de extraer el asunto y cuerpo de cada correo, así como todos los adjuntos y los metadatos asociados (nombre, tipo, tamaño). Para los casos de adjuntos anidados, como correos encapsulados o archivos ZIP, se itera recursivamente hasta extraer todos los documentos posibles. Las imágenes incrustadas en el cuerpo del correo también se extraen como adjuntos, pero, como ya se ha comentado previamente, aquellas con resolución inferior a 128x128 píxeles son descartadas automáticamente por considerarse firmas o logos. Una vez llevada a cabo esta extracción, se encarga de persistir los adjuntos en un bucket de Amazon S3 y los metadatos en DynamoDB, para su posterior procesamiento.

- `extract_text`: esta lambda es la encargada de extraer el contenido de los adjuntos obtenidos por `extract_attachments`. Para llevar a cabo dicha tarea, se ha implementado una estrategia de extracción híbrida, utilizando tanto técnicas clásicas de OCR con Tesseract como directamente el LLM, el cual es capaz de analizar el contenido de adjuntos como imágenes gracias a su capacidad multimodal (véase sección 2.3). Este procesamiento híbrido permite alcanzar mejores resultados al no depender exclusivamente de la calidad de los adjuntos ni de las limitaciones de cada técnica, sino combinando ambas para maximizar la tasa de extracción de información relevante. Una vez extraído el contenido de los adjuntos, se realizan varias llamadas a IAGen para clasificar correctamente según el catálogo de categorías establecido por negocio y se persiste lo obtenido tanto en DynamoDB como en el bucket de S3.

- `classifier_iagen`: en nuestro caso de uso y pese a su nombre, esta lambda no realiza una llamada a un LLM, sino que obtiene como evento de entrada la información extraída por la lambda `extract_text` de los adjuntos clasificados y se encarga de renombrar los archivos según lo estipulado en RF_002 (véase paso 5). Una vez que se han renombrado los archivos, se persisten en el bucket de S3 con una lógica subyacente que los organiza según se trate de grupo de acción o grupo de control, y que permitirá a la siguiente lambda (`email_forward`) abstraerse de dicha lógica al hacer la función de reenvío.

- `email_forward`: reenvía el correo al buzón de destino con los adjuntos procesados o en su formato original, según la lógica del grupo de acción y control. A su vez establece mecanismos de seguridad que aseguran que no se pierde información en ningún caso, por ejemplo, en caso de fallo durante el reenvío, el correo se reintenta automáticamente hasta un número máximo de intentos, y si se supera dicho número, se genera una alarma para que el equipo de soporte revise el caso manualmente. También se encarga de etiquetar el asunto del correo con el identificador `[data_process_id]` para evitar reprocesamientos y asegura que nunca se pierda el número de referencia en el asunto, incluso en casos de truncamiento por exceso de longitud. En caso de que el correo pertenezca al grupo de control, se reenvía sin clasificación ni renombrado de adjuntos, pero igualmente se etiqueta con `[data_process_id]` para su seguimiento.

- `email_deleter`: elimina permanentemente el correo del Inbox del buzón proxy tras el reenvío exitoso, conforme a lo establecido en RF_002 (véase paso 10) y RF_006 (3.1.1). Esta eliminación es necesaria para evitar la acumulación de correos ya procesados y garantizar la disponibilidad de espacio

en el buzón, lo que de no controlarse podría impedir la recepción de nuevos correos.

- `email_monitoring`: registra el resultado de cada ejecución del pipeline para dar cumplimiento a RF_010 (3.1.3) y RF_011 (3.1.3). Expone métricas técnicas y funcionales como tiempos de procesamiento, tasas de clasificación por tipo documental, errores producidos y estado de los componentes, que se articulan en dos capas: una técnica mediante AWS CloudWatch y una funcional mediante dashboards en Power BI.

### 4.2.2. Loader pattern para la generalización a múltiples buzones

La generalización a nuevos buzones es uno de los objetivos centrales del sistema (véase sección 1.2). Para materializarla a nivel de implementación, se ha adoptado un *loader pattern*: cada buzón o caso de uso se define como un proyecto independiente dentro de la lambda `extract_text`, con su propio *summarizer* (encapsula la lógica del proyecto), sus propios prompts y sus propios esquemas de salida. En tiempo de ejecución, la lambda carga dinámicamente el *summarizer* correspondiente al proyecto configurado mediante variable de entorno, sin necesidad de modificar el código base.

Este diseño en cadena de responsabilidad única facilita el mantenimiento y el testeo independiente de cada componente, y permite sustituir o modificar cualquier pieza del pipeline sin afectar al resto. La incorporación de un nuevo buzón al sistema se reduce, así, a desplegar una nueva instancia del pipeline con configuración propia, sin modificar la lógica existente ni la arquitectura subyacente.

### 4.2.3. Gestión de credenciales y configuración

Todas las claves de acceso a servicios externos se almacenan en AWS Secrets Manager, evitando cualquier referencia a credenciales en el código fuente y centralizando su rotación y auditoría.

### 4.2.4. Persistencia y observabilidad

El sistema utiliza Amazon S3 para el almacenamiento de adjuntos procesados y DynamoDB para el registro estructurado de metadatos de cada correo procesado. Los logs de ejecución de las Lambdas se centralizan en CloudWatch Logs, desde donde son accesibles para monitorización en tiempo real. Para el reporting y seguimiento del impacto del sistema, Power BI se conecta a los datos mediante un gateway ODBC a través de Denodo, permitiendo construir cuadros de mando

sobre los datos almacenados en AWS sin exponer directamente la infraestructura interna.

## 4.3. Configuración del entorno de IA

Tras la evaluación técnica detallada en el Capítulo 2, se ha procedido a la integración de **Gemini 2.5 Flash** como motor de razonamiento del sistema. Para efectos del diseño detallado, la configuración operativa se ha estandarizado bajo los siguientes parámetros de ejecución:

- **Endpoint y Gobernanza:** Consumo mediante *Vertex AI*, garantizando el cumplimiento de las políticas de seguridad y residencia de datos de MAPFRE.
- **Gestión de cuotas:** Ajuste de los límites de concurrencia en *Vertex AI* para sincronizarse con las capacidades de escalado de las funciones Lambda y evitar errores de *rate limiting*.
- **Estrategia de reintento (*Exponential Backoff*):** Ante errores transitorios de red o saturación, el cliente de invocación implementa una política de reintentos con *backoff* exponencial y *jitter*, mitigando la posibilidad de fallo en el procesamiento de correos durante picos de tráfico.

**Estrategia de adaptación: prompt engineering.** La adaptación del modelo a la variabilidad documental de MAPFRE se ha realizado mediante *prompt engineering*. Esta elección, justificada técnicamente en la Sección 2.5 de este trabajo se debe principalmente a la necesidad de **agilidad iterativa**. Debido a que los criterios de clasificación han evolucionado en respuesta a las necesidades de negocio durante el desarrollo, el *prompting* ha permitido reconfigurar el comportamiento del sistema de forma inmediata, evitando los tiempos de reentrenamiento.

### 4.3.1. Infraestructura como código

Toda la infraestructura AWS del proyecto está definida mediante Terraform, lo que garantiza la reproducibilidad del entorno y elimina la configuración manual. Esto es especialmente relevante para el objetivo de generalización: desplegar el sistema sobre un nuevo buzón se reduce a parametrizar y aplicar los módulos Terraform existentes, sin necesidad de provisionar recursos manualmente. Dicho despliegue automatizado es posible gracias a pipelines de CICD implementados con Jenkins sobre el repositorio de código alojado en Bitbucket.

### 4.3.2. Definición del grupo de acción y control

El RF_003 (sección 3.1.1) establece la necesidad de una división entre grupo de acción y grupo de control con el objetivo de medir el impacto real de la automatización. Esta sección describe cómo se materializa dicha división a nivel de diseño y cómo habilita el cálculo del retorno de la inversión del proyecto.

El **grupo de acción** comprende el 90 % de los correos entrantes, que son procesados de forma completa por el pipeline de DATA&IA: clasificación de adjuntos, extracción del número de referencia y reenvío al buzón de destino con los adjuntos procesados o en su formato original según la lógica del sistema.

El **grupo de control** comprende el 10 % restante, seleccionados de forma aleatoria. Estos correos son marcados con el identificador `[data_process_id]` en el asunto pero no reciben clasificación de adjuntos, siendo derivados al flujo manual tradicional para que los gestores los procesen como lo harían sin el sistema.

Esta separación permite construir una comparativa objetiva y estadísticamente representativa entre ambos grupos. El indicador principal es el **tiempo de gestión por correo**: el tiempo medio registrado en el grupo de control actúa como línea base, representando el esfuerzo que habría requerido cada correo sin automatización. La diferencia respecto al tiempo invertido en los correos del grupo de acción refleja directamente el ahorro operativo unitario que aporta el sistema.

Agregando ese ahorro al volumen total de correos procesados y poniéndolo en relación con el coste de operación del sistema, infraestructura AWS y llamadas a la API del modelo, es posible calcular el retorno de la inversión de forma rigurosa y basada en datos reales de uso.

# Capítulo 5

# Implementación

Este capítulo detalla la implementación de los componentes del pipeline sobre los que se ha tenido mayor responsabilidad de desarrollo: `extract_text`, encargada de la extracción de contenido y clasificación de adjuntos, y `email_forward`, responsable del reenvío controlado al buzón de destino, así como las componentes auxiliares de monitorización y contingencia. El resto de componentes del pipeline fueron descritos en el Capítulo 4 a nivel de diseño; aquí se amplía únicamente aquello que requiere detalle de implementación adicional.

## 5.1. extract_text

Como se describió en la sección 4.2.1, `extract_text` se ejecuta una vez por adjunto, de forma independiente y en paralelo para cada uno de los ficheros extraídos por `extract_attachments`. Cada invocación recibe un único adjunto, lo procesa de forma completamente aislada y persiste su resultado en S3 y DynamoDB sin ningún estado compartido con el resto de invocaciones del mismo correo. Este diseño aprovecha el paralelismo nativo de AWS Lambda para escalar el procesamiento de forma proporcional al volumen recibido, tal y como ilustra la figura 5.1.

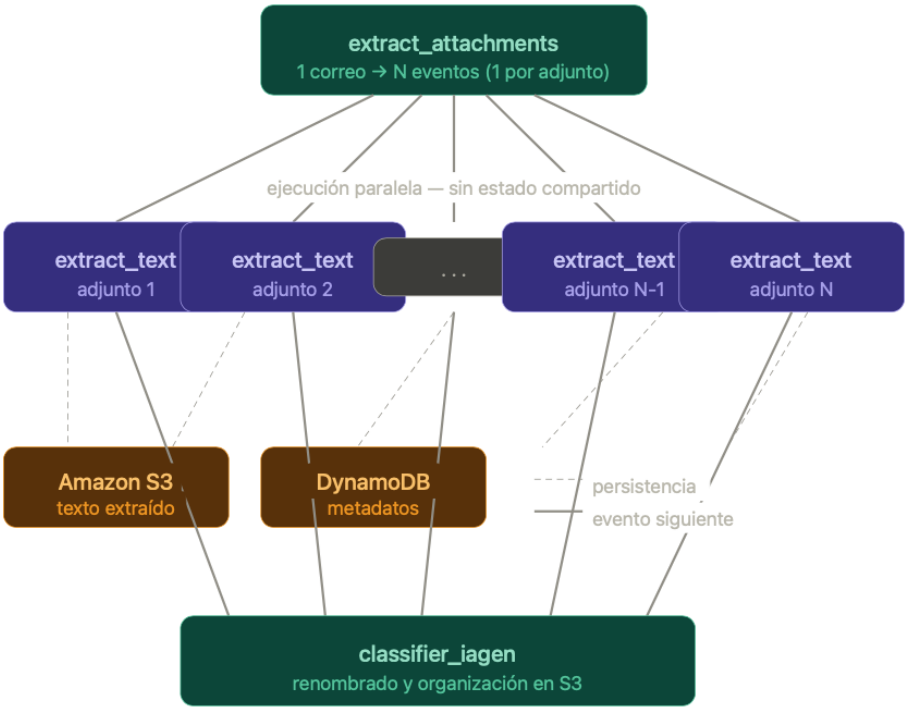


Figura 5.1: Ejecución paralela de `extract_text`: cada adjunto genera una invocación independiente.

Internamente, la lambda sigue un flujo de dos fases: en la primera se extrae el contenido textual del adjunto mediante una estrategia híbrida; en la segunda se clasifica el adjunto contra el catálogo de tipos documentales, se extrae el número de referencia y se persiste el resultado.

La figura 5.2 muestra el diagrama de clases del módulo, ilustrando las relaciones entre los componentes principales: el `TextExtractorRegistry` para la extracción de contenido, la jerarquía de clases centrada en `PrestAutoSummarizer` para la lógica de clasificación, los esquemas Pydantic que validan las respuestas del modelo, y el `GenAIGeminiRequestHandler` que encapsula la comunicación con Vertex AI.

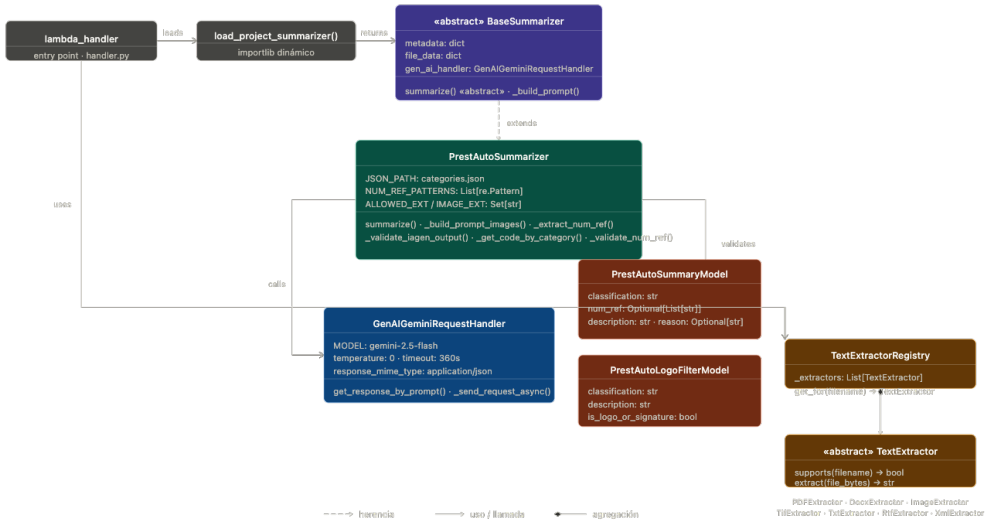


Figura 5.2: Diagrama de clases del módulo `extract_text`

### 5.1.1. Estrategia de extracción híbrida

El primer paso de cada invocación es extraer el contenido textual del adjunto. Para ello se ha implementado un `TextExtractorRegistry`, un registro de extractores especializados por formato que selecciona automáticamente el extractor adecuado en función de la extensión del fichero. Los formatos soportados son: PDF, DOCX, DOC, RTF, TIF/TIFF, TXT, imágenes (JPG, PNG, BMP, GIF) y XML.

Para la mayoría de formatos, la extracción produce texto plano que se incorpora directamente al contexto del modelo. Sin embargo, existen dos casos que requieren un tratamiento específico.

El primero son los **documentos escaneados e imágenes**: cuando el adjunto es una imagen o un PDF cuyo texto extraído es vacío o insuficiente, se asume que el contenido no es digital nativo sino visual. En ese caso, el adjunto se envía directamente al modelo como imagen, aprovechando su capacidad multimodal nativa (véase sección 2.3) para analizar el contenido visualmente sin depender de la calidad del OCR.

El segundo caso son los **PDFs con contenido mixto**: documentos que combinan páginas con texto digital y páginas escaneadas. Para estos casos, el PDF se convierte a una secuencia de imágenes PNG, una por página, que se envían al modelo junto con el texto extraído. Para evitar sobrecargar el contexto del modelo, se limita el envío a las primeras diez páginas.

Esta estrategia (texto cuando está disponible, imagen cuando no) implemen-

ta la arquitectura híbrida descrita en la sección 2.7: el modelo multimodal actúa como motor principal de procesamiento, con Tesseract como mecanismo de contingencia para documentos donde la llamada al modelo no es posible o resulta sobredimensionada.

### 5.1.2. Clasificación de adjuntos

Una vez extraído el contenido, la lambda ejecuta la lógica de clasificación en cuatro pasos.

#### Filtro previo de logos y firmas

Antes de invocar al modelo para la clasificación documental, se aplica un filtro específico para imágenes. Los correos corporativos incluyen con frecuencia imágenes incrustadas que corresponden a logos de empresa o firmas digitales; clasificar estos elementos como documentos introduciría ruido en el resultado. Para evitarlo, cuando el adjunto es una imagen se realiza una primera llamada al modelo con un prompt específico que determina si se trata de un logo o firma. Si el modelo lo confirma, la invocación termina devolviendo la categoría `Desconocido` con código TRAMES vacío, sin realizar ninguna llamada adicional, ahorrando coste y latencia. Solo si el modelo descarta que sea un logo o firma se procede a la clasificación documental completa.

Nótese que este filtro actúa como una segunda capa de seguridad: la primera es el descarte por resolución inferior a 128×128 píxeles realizado en `extract_attachments`, que elimina la mayoría de firmas y logos antes de que lleguen a esta lambda.

La separación entre las dos llamadas al modelo queda reflejada en los esquemas Pydantic que validan cada respuesta. El filtro de logos y firmas utiliza `PrestAutoLogoFilterModel`, que únicamente requiere que el modelo indique si el adjunto es un logo o firma junto con una justificación. La clasificación documental completa utiliza `PrestAutoSummaryModel`, que extiende el anterior añadiendo los campos de clasificación y número de referencia. Esta separación garantiza que el modelo no realiza trabajo innecesario cuando el adjunto es descartado en el primer filtro.

```python
class PrestAutoLogoFilterModel(BaseModel):
    classification: str
    description: str
    is_logo_or_signature: bool

class PrestAutoSummaryModel(BaseModel):
    classification: str
    description: str
    num_ref: Optional[List[str]] = None
    is_logo_or_signature: bool = False
```

Listing 1: Esquemas Pydantic para el filtro de logos y la clasificación documental

### Llamada al modelo y validación de la salida

La clasificación se realiza mediante una llamada al modelo con el prompt descrito en la sección 5.1.4, que recibe el contenido extraído, el nombre del fichero, el tipo MIME y el catálogo completo de categorías. El modelo devuelve un JSON validado contra `PrestAutoSummaryModel` con los campos: la categoría asignada (`classification`), el razonamiento explícito del modelo (`description`), el número de referencia encontrado (`num_ref`) y un indicador de logo o firma (`is_logo_or_signature`).

La respuesta del modelo no se acepta directamente sino que pasa por un proceso de validación en dos niveles. En el primer nivel, la respuesta se valida contra el esquema Pydantic que comprueba que los campos obligatorios están presentes y tienen el tipo correcto. En el segundo nivel, la categoría devuelta se contrasta contra el catálogo de tipos documentales cargado desde un fichero JSON externo (`categories.json`), verificando que coincide literalmente (incluyendo mayúsculas, tildes y espacios) con alguna de las categorías válidas. Si la categoría no supera esta validación, se sustituye automáticamente por `Desconocido`, evitando que el modelo invente o renombre categorías fuera del catálogo definido por negocio.

### Asignación del código TRAMES

Una vez validada la categoría, se asigna el código numérico TRAMES correspondiente, también cargado desde `categories.json`. Este código es el que realmente interpreta TRAMES para el enrutamiento del documento dentro del expediente del siniestro, por lo que su asignación se realiza localmente a partir del catálogo y nunca se delega al modelo. Los documentos clasificados como `Desconocido`, ya sea por fallo de validación o por haber sido identificados como

logo o firma en el filtro previo, reciben un código vacío y se reenvían sin renombrar para que los gestores los procesen manualmente.

### Extracción y validación del número de referencia

La extracción del número de referencia sigue la estrategia de dos fases definida en el RF 002 (véase sección 3.1.1). En la primera fase se aplican expresiones regulares sobre el asunto y cuerpo del correo, buscando los dos patrones definidos por negocio: 10 dígitos con prefijo 049/050/051/052/053/054/055/056, o 9 dígitos con prefijo 49/50/51/52/53/54/55/56. En la segunda fase, el número de referencia devuelto por el modelo desde el contenido del adjunto se valida contra esos mismos patrones antes de aceptarlo, descartando cualquier valor que no cumpla el formato esperado.

Cuando se encuentran varios números de referencia, se aplica la regla de prioridad de negocio: los encontrados en asunto y cuerpo tienen precedencia sobre los extraídos de adjuntos. El resultado final (categoría validada, código TRAMES, número de referencia y razonamiento del modelo (`description`)) se persiste en DynamoDB y S3 para su consumo por las lambdas posteriores del pipeline. El campo `description` no interviene en el flujo de procesamiento posterior, pero resulta de especial utilidad durante el proceso iterativo de mejora del prompt: permite identificar con precisión por qué el modelo toma una decisión incorrecta en casos ambiguos, facilitando la redacción de reglas de desempate específicas sin necesidad de inferir el razonamiento a partir únicamente de la categoría devuelta. A modo ilustrativo:

```
{
    "classification": "INFORME PERICIAL",
    "code": "001",
    "num_ref": ["0491234567"],
    "description": "Se identifica membrete de empresa pericial y sección
                de conclusiones con valoración económica. Se descarta
                Comunicación de Terceros porque el contenido predominante
                es técnico-pericial y no una comunicación genérica."
}
```

Listing 2: Ejemplo de registro persistido en DynamoDB tras la clasificación de un adjunto

### 5.1.3. Invocación del modelo

Las llamadas al modelo se centralizan en la clase `GenAIGeminiRequestHandler`, que encapsula toda la lógica de comunicación con Vertex AI y abstrae al resto del sistema de los detalles de la API. El cliente se inicializa con credenciales de cuenta de servicio GCP recuperadas en tiempo de ejecución desde AWS Secrets Manager, garantizando que ninguna clave aparece en el código fuente.

Los parámetros de generación más relevantes son:

- **Modelo**: `gemini-2.5-flash`, consumido a través de Vertex AI para garantizar la residencia de datos y el cumplimiento de las políticas de seguridad corporativas de MAPFRE.
- **Temperature**: `0`, eliminando la aleatoriedad en las respuestas y maximizando el determinismo de las clasificaciones en producción.
- **Response MIME type**: `application/json`, forzando una salida JSON nativa en combinación con el esquema Pydantic definido para cada proyecto, lo que implementa a nivel de API el *structured output prompting* descrito en la sección 2.6.
- **Timeout**: 360 segundos por invocación, suficiente para absorber la latencia de documentos de gran tamaño o picos de carga en la API.

La estrategia de resiliencia ante fallos transitorios implementa un *exponential backoff* con reintentos en `[5s, 15s, 40s]` antes de declarar el fallo definitivo, tal y como se describe en la sección 4.3. Adicionalmente, el handler registra el número de tokens de entrada y salida de cada llamada mediante `count_tokens`, lo que permite monitorizar el consumo real de la API y detectar anomalías en el volumen de tokens procesados.

Por último, el método `_safe_json_parse` implementa una capa de limpieza defensiva sobre la respuesta: aunque la combinación de `response_mime_type` y esquema Pydantic garantiza en la práctica una salida JSON válida, el parser elimina posibles prefijos de markdown antes de deserializar, evitando fallos en casos extremos donde el modelo no respete completamente el formato solicitado.

### 5.1.4. Prompt engineering

El prompt de clasificación es el componente central de `extract_text` y el que más ha evolucionado a lo largo del proyecto. Esta sección describe su estructura y las técnicas aplicadas. El proceso iterativo de mejora y sus resultados se recogen en el Capítulo 6.

### Estructura del prompt

El prompt está implementado con plantillas Jinja2, lo que permite separar la lógica de renderizado del contenido y reutilizar la misma estructura para distintos casos de uso, alineándose con el loader pattern descrito en la sección 4.2.2. El catálogo de tipos documentales se mantiene en un fichero JSON externo (`categories.json`), desacoplado del prompt, de forma que cualquier modificación en las categorías como añadir un tipo documental nuevo, modificar su descripción o cambiar su código TRAMES, no requiere tocar el prompt sino únicamente el fichero de configuración.

```
{# System prompt - fragmento de categorías #}
{% for cat in categories %}
- Tipo Documental: "{{ cat.tipo_documental }}"
  Descripcion: "{{ cat.descripcion }}"
{% endfor %}

{# User prompt - fragmento del adjunto #}
Archivo: {{ filename }}
Tipo MIME: {{ mime_type }}

Contenido extraido:
{{ content }}
```

Listing 3: Fragmento de la plantilla Jinja2 del prompt de clasificación

El prompt se compone de dos partes. El *system prompt* define el rol del modelo como experto en gestión documental de MAPFRE, proporciona el catálogo completo de categorías con sus descripciones, instruye sobre los patrones válidos del número de referencia y especifica el esquema exacto de la respuesta esperada en formato JSON. El *user prompt* aporta el contexto del adjunto concreto: el contenido extraído, el nombre del fichero y el tipo MIME. En el caso de adjuntos visuales, se acompaña además de la imagen o secuencia de imágenes del documento.

### Técnicas aplicadas

La clasificación combina las técnicas de prompt engineering descritas en la sección 2.6. El *structured output prompting* garantiza que el modelo devuelva siempre un JSON con los campos `classification`, `description` y `num_ref`, validado posteriormente contra un esquema Pydantic. (ver listado 1).

El *zero-shot prompting* es la base del prompt: el modelo clasifica cada adjunto sin ejemplos previos, apoyándose únicamente en las descripciones del catálogo. Pa-

ra los casos ambiguos, se aplica una lógica de *chain-of-thought* implícita mediante el campo `description`, que obliga al modelo a justificar su decisión antes de emitir la categoría final, reduciendo las clasificaciones arbitrarias.

Además, el prompt incorpora reglas de desempate explícitas para las categorías que generaban más confusión durante la validación técnica inicial. Estas reglas se añaden como instrucciones `IMPORTANTE` en ambos niveles del prompt para maximizar su cumplimiento. Las más relevantes son:

- **Comunicación de Terceros vs. categoría específica**: el modelo tendía a clasificar como *Comunicación de Terceros* cualquier documento con membrete externo, cuando debía priorizar la categoría específica si el contenido lo justificaba.
- **Documentos con estructura económica**: facturas, minutas y liquidaciones deben clasificarse por su tipología específica, independientemente del emisor.
- **Documentos mixtos**: adjuntos que combinan varios tipos documentales deben clasificarse por el tipo predominante en volumen de contenido y objetivo del documento, justificándolo en `description`.
- **Etiquetas de paquetería y albaranes**: se clasifican como *Desconocido* solo cuando constituyen el contenido predominante, no cuando acompañan a otro documento con contenido sustantivo.

La incorporación progresiva de estas reglas fue uno de los principales factores de mejora del rendimiento entre iteraciones, como se evidencia en los resultados de la POC recogidos en el Capítulo 6.

## 5.2. email_forward

La lambda `email_forward` es el último paso del pipeline de procesamiento antes de la eliminación del correo del buzón proxy. Su responsabilidad es devolver al buzón principal de prestaciones de autos un correo que conserve toda la información del original (asunto, cuerpo, hilo de conversación y adjuntos) pero enriquecido con el resultado del procesamiento: adjuntos clasificados y renombrados según el catálogo TRAMES y un asunto compuesto que incluye el `data_process_id` y el número de referencia del siniestro.

El diseño de esta lambda contempla dos estrategias de reenvío intercambiables, seleccionables mediante la variable de entorno `FORWARD_POLICY`:

- **ENCAPSULATE_EMAIL**: genera un correo nuevo en Microsoft 365 que encapsula el MIME original como adjunto `.eml`. Esta estrategia es adecuada para casos de uso donde el buzón de destino no requiera preservar el hilo de conversación, o como mecanismo de fallback cuando no sea posible realizar el reenvío. Su naturaleza genérica la convierte en la opción de referencia para nuevos buzones corporativos que se incorporen al sistema con requisitos de integración distintos a los de prestaciones de autos.

- **REPLY_EMAIL**: localiza el correo original en el buzón de destino mediante su `Internet-Message-ID` y responde sobre él con los adjuntos procesados y el asunto construido. Al tratarse de una respuesta al mensaje original, el correo reenviado conserva el mismo identificador de conversación, lo que garantiza la trazabilidad en TRAMES y elimina el riesgo de duplicados.

La estrategia `REPLY_EMAIL` es la aportación central de este proyecto en lo que respecta al reenvío. Su implementación requirió resolver varios problemas no triviales: la localización del correo original por `Internet-Message-ID` mediante la API de Microsoft Graph, la gestión del ciclo de vida de un borrador de respuesta, el filtrado diferenciado de adjuntos según el grupo de acción o control, y el truncado inteligente del asunto para preservar el número de referencia cuando se supera el límite de 255 caracteres impuesto por Microsoft 365. La clase `EmailReplyForwarder` y su hija `PrestAutoEmailReplyForwarder`, desarrollada específicamente para el buzón piloto de prestaciones de autos, extiende la implementación base con esta última lógica de truncado, ilustrando cómo el loader pattern descrito en la sección 4.2.2 permite adaptar el comportamiento del sistema a las particularidades de cada buzón sin modificar el núcleo de la arquitectura.

### 5.2.1. Jerarquía de clases y patrón Strategy

La figura 5.3 muestra el diagrama de clases del módulo. La arquitectura se organiza en torno a dos niveles de abstracción. El primero es `EmailProcessorBase`, clase base común a toda operación sobre un correo en el pipeline, tanto el reenvío como la eliminación que implementa el orquestador `execute()` con la lógica de reintentos, actualización en DynamoDB y cierre de sesión con Microsoft 365, y declara `_perform_action()` como método abstracto que cada subclase debe implementar. El segundo es `EmailForwarderBase`, que extiende la base añadiendo los atributos específicos del reenvío: destinatario, asunto construido por el clasificador, estado del procesamiento, indicador de grupo de control y ruta base en S3.

Sobre esta jerarquía se implementan las dos estrategias de reenvío descritas en la introducción. `EmailNewGenerator` materializa `ENCAPSULATE_EMAIL` descargando

el MIME original desde S3 y enviándolo como correo nuevo con el contenido original encapsulado como adjunto con formato `eml`. `EmailReplyForwarder` materializa `REPLY_EMAIL` localizando el correo original en el buzón de destino y respondiendo sobre él con los adjuntos procesados. `PrestAutoEmailReplyForwarder` extiende esta última clase sobreescribiendo únicamente el método de truncado del asunto para preservar el número de referencia, sin modificar ningún otro comportamiento. De nuevo está presente el principio de responsabilidad única y modularidad: cada clase está pensada para resolver el problema asociado al caso de uso concreto, sin mezclar lógica de negocio o detalles de implementación.

La selección de la estrategia concreta se realiza en el `lambda_handler` mediante un diccionario de políticas indexado por el valor de la variable de entorno `FORWARD_POLICY`. Cuando el proyecto es `prest_auto`, la entrada `REPLY_EMAIL` del diccionario se sobreescribe con `PrestAutoEmailReplyForwarder` antes de instanciar el forwarder, siguiendo el mismo loader pattern descrito en la sección 4.2.2:

```
_FORWARD_POLICY_MAP = {
    "ENCAPSULATE_EMAIL": EmailNewGenerator,
    "REPLY_EMAIL": EmailReplyForwarder,
}

def lambda_handler(event, context):
    forward_policy = get_env("FORWARD_POLICY", default="ENCAPSULATE_EMAIL")
    project_name   = get_env("PROJECT_NAME",   default="default_project")

    if project_name == "prest_auto":
        _FORWARD_POLICY_MAP["REPLY_EMAIL"] = PrestAutoEmailReplyForwarder

    forwarder_cls = _FORWARD_POLICY_MAP.get(forward_policy)
    email_forwarder = forwarder_cls(event)
    return email_forwarder.execute()
```

Listing 4: Selección dinámica de la estrategia de reenvío en `lambda_handler`

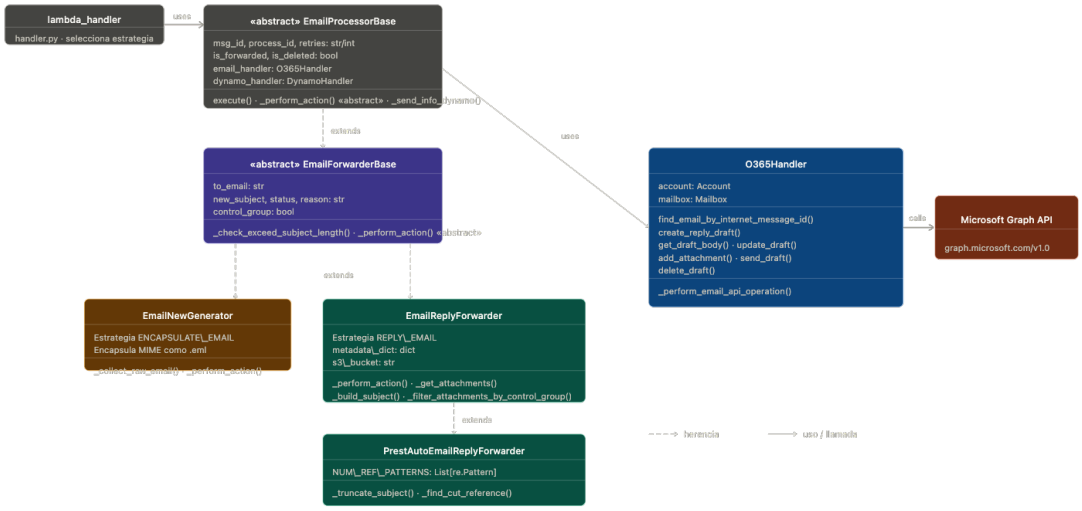


Figura 5.3: Diagrama de clases del módulo `email_forward`

### 5.2.2. Integración con Microsoft Graph API

La comunicación con Microsoft 365 se centraliza en la clase `O365Handler`, que encapsula todas las llamadas a la API de Microsoft Graph y abstrae al resto del sistema de los detalles del protocolo HTTP subyacente. La autenticación se realiza mediante credenciales de aplicación (client credentials flow) recuperadas en tiempo de ejecución desde AWS Secrets Manager (`client_id`, `client_secret` y `client_tenant`), garantizando que ninguna credencial aparece en el código fuente.

Una decisión de implementación relevante es el uso de llamadas directas a la API de Microsoft Graph en lugar de utilizar los métodos de alto nivel que ofrece la librería `O365` sobre la que se apoya el handler. Durante el desarrollo se comprobó que varios de estos métodos de alto nivel producían errores intermitentes e inconsistencias en operaciones críticas como la creación de borradores de respuesta o la adición de adjuntos, debido a diferencias entre la versión de la librería disponible y el comportamiento actual de la API. La solución adoptada fue implementar un wrapper propio, `_perform_email_api_operation`, que ejecuta las llamadas HTTP directamente sobre los endpoints de Graph API reutilizando únicamente la sesión autenticada que proporciona la librería. Este enfoque ofrece un control total sobre el payload, los códigos de respuesta y el manejo de errores, eliminando la dependencia de los abstracciones de la librería para las operaciones más sensibles del pipeline.

Los métodos utilizados específicamente por `EmailReplyForwarder` durante el flujo de reenvío son los recogidos en el cuadro 5.1:

| Método | Endpoint Graph API | Propósito |
|---|---|---|
| find_email_by_internet_message_id | GET /messages?$filter | Localizar el correo original por su ID |
| create_reply_draft | POST /messages/{id}/createReply | Crear borrador de respuesta |
| get_draft_body | GET /messages/{id}?$select=body | Obtener el cuerpo del borrador |
| update_draft | PATCH /messages/{id} | Actualizar asunto y cuerpo |
| add_attachment | POST /messages/{id}/attachments | Adjuntar fichero procesado |
| send_draft | POST /messages/{id}/send | Enviar el borrador |
| delete_draft | DELETE /messages/{id} | Eliminar borrador en caso de fallo |

Cuadro 5.1: Métodos de O365Handler utilizados en el flujo de reenvío

Todas las operaciones se ejecutan con reintentos automáticos mediante retry_operation, un helper que reintenta la llamada hasta tres veces ante fallos transitorios antes de elevar la excepción al orquestador de EmailProcessorBase, donde se gestiona el ciclo completo de reintentos descrito en la sección 5.2.6.

### 5.2.3. Flujo de reenvío

El método _perform_action de EmailReplyForwarder implementa el flujo de reenvío en cinco pasos secuenciales, cada uno dependiente del anterior, tal y como ilustra la figura 5.4.


Figura 5.4: Flujo de reenvío de `EmailReplyForwarder`: camino feliz

**1. Localización del correo original.** El primer paso es localizar el correo original en el buzón de destino mediante su `Internet-Message-ID`, el identificador único e inmutable que asigna el servidor de correo al mensaje en el momento de su creación y que se preserva a lo largo de toda su vida útil, independientemente de los movimientos entre carpetas. Esta localización se realiza mediante una consulta filtrada a la API de Graph (`find_email_by_internet_message_id`), que devuelve una lista de mensajes coincidentes.

**2. Creación del borrador de respuesta.** Una vez localizado el correo original, se crea un borrador de respuesta sobre él mediante `create_reply_draft`. Graph API genera automáticamente el borrador con el hilo de conversación citado, el destinatario del remitente original y el asunto prefijado con `RE:`. El identificador del borrador (`draft_id`) se conserva en memoria durante todo el flujo, lo que resulta clave para la gestión de errores descrita en la sección 5.2.6.

**3. Actualización del asunto y limpieza del cuerpo.** El borrador recién creado se actualiza con el asunto construido por el pipeline, que incluye el `data_process_id`, el número de referencia y el asunto original, y con el cuerpo del hilo de conversación original, del que se eliminan previamente las referencias a imágenes inline (`cid:`) mediante `_clean_inline_images`. Esta limpieza es necesaria porque las imágenes inline del correo original serán adjuntadas (clasificadas) en el nuevo correo y por

lo tanto deben eliminarse para evitar duplicados.

**4. Adición de adjuntos procesados.** Los adjuntos clasificados y renombrados se recuperan desde S3 y se añaden al borrador uno a uno mediante llamadas sucesivas a `add_attachment`. La selección de qué adjuntos incluir depende del grupo al que pertenezca el correo, como se detalla en la sección 5.2.4.

**5. Envío del borrador.** Una vez completados los pasos anteriores, el borrador se envía mediante `send_draft`. En este momento el correo abandona el estado de borrador y se deposita en el buzón de destino como respuesta al mensaje original, conservando el hilo de conversación y el identificador de mensaje. Esta es precisamente la ventaja clave de la estrategia `REPLY_EMAIL` frente a `ENCAPSULATE_EMAIL`: al responder sobre el mensaje original en lugar de crear uno nuevo, TRAMES recibe el correo con el mismo identificador de conversación y sin riesgo de duplicados.

### 5.2.4. Gestión de adjuntos según grupo de acción y control

Como se describió en la sección 4.3.2, el pipeline divide los correos entrantes en dos grupos: el grupo de acción (90 %) recibe el procesamiento completo de clasificación y renombrado de adjuntos, mientras que el grupo de control (10 %) se reenvía sin clasificar para que los gestores lo procesen manualmente y sirva de línea base para el cálculo del ROI.

Esta distinción se materializa en `EmailReplyForwarder` a través del método `_filter_attachments_by_control_group`, que selecciona qué adjuntos recuperar desde S3 en función del valor del flag `control_group`. En ambos casos los adjuntos se encuentran en S3, depositados previamente por la lambda `classifier_iagen`:

- **Grupo de acción** (`control_group = False`): se recuperan los ficheros con `level != 0`, es decir, todos los adjuntos procesados independientemente de su profundidad de anidamiento, clasificados y renombrados según el catálogo TRAMES.
- **Grupo de control** (`control_group = True`): se recuperan los ficheros con `level == 1`, los adjuntos directos del correo original, sin clasificar ni renombrar, tal y como llegaron en el correo. Este filtro evita reenviar contenido extraído recursivamente de ZIPs o correos encapsulados que el gestor no esperaría recibir.

El campo `level` actúa como mecanismo de seguimiento de la profundidad de anidamiento: `level = 0` identifica el correo raíz, `level = 1` los adjuntos directos del correo original, y valores superiores los adjuntos extraídos de forma recursiva

de adjuntos anidados. A modo ilustrativo, considérese un correo con la siguiente estructura:

En ambos casos se excluyen los adjuntos marcados con `is_logo = True`, es decir, aquellos que el filtro previo de `extract_text` identificó como logos o firmas digitales y que no deben incluirse en el correo reenviado independientemente del grupo.

```
Correo original                          (level 0)
├── informe_pericial.pdf                 (level 1)
├── documentacion.zip                    (level 1)
│   ├── factura.pdf                      (level 2)
│   └── fotos_danios.jpg                 (level 2)
└── correo_reenviado.eml                 (level 1)
    └── declaracion_accidente.pdf        (level 2)
```

Listing 5: Ejemplo de estructura jerárquica de adjuntos con campo `level`

El grupo de acción reenviará los seis adjuntos procesados (`level != 0`), mientras que el grupo de control reenviará únicamente los tres adjuntos directos del correo original (`level == 1`): `informe_pericial.pdf`, `documentacion.zip` y `correo_reenviado.e` sin descomprimir ni extraer su contenido.

### 5.2.5. Truncado inteligente del asunto

Microsoft 365 impone un límite de 255 caracteres en el asunto de los mensajes. El asunto construido por el pipeline puede superar fácilmente este límite, ya que concatena el `data_process_id`, el número de referencia del siniestro y el asunto original del correo, que en ocasiones es largo por sí solo.

La clase base `EmailReplyForwarder` implementa un truncado simple: si el asunto supera el límite, se corta por el carácter 255. Este comportamiento es suficiente para la mayoría de casos, pero presenta un problema crítico en el contexto de nuestro caso de uso: si el corte cae en mitad del número de referencia del siniestro, TRAMES no podrá leerlo y el correo no se asociará automáticamente al expediente correspondiente, obligando a intervención manual.

`PrestAutoEmailReplyForwarder` sobreescribe el método `_truncate_subject` para resolver este problema. En lugar de cortar siempre por el carácter 255, el algoritmo comprueba si el punto de corte cae dentro de un número de referencia válido. Si no hay ningún número de referencia en riesgo, aplica el truncado normal. Si lo hay, elimina caracteres del prefijo para que el número de referencia quepa ínte-

gro dentro del límite, garantizando que TRAMES pueda procesarlo correctamente independientemente de la longitud total del asunto.

### 5.2.6. Resiliencia y garantía de no pérdida de información

La premisa de diseño del sistema es que ningún correo puede perderse bajo ninguna circunstancia: cada correo que entra en el pipeline debe llegar al buzón de destino, aunque sea sin clasificar. Para materializar esta garantía, el manejo de errores opera en dos capas complementarias.

En la imagen 5.5 se ilustran los puntos de fallo posibles durante el flujo de reenvío y los mecanismos de recuperación asociados a cada uno, que se describen a continuación.

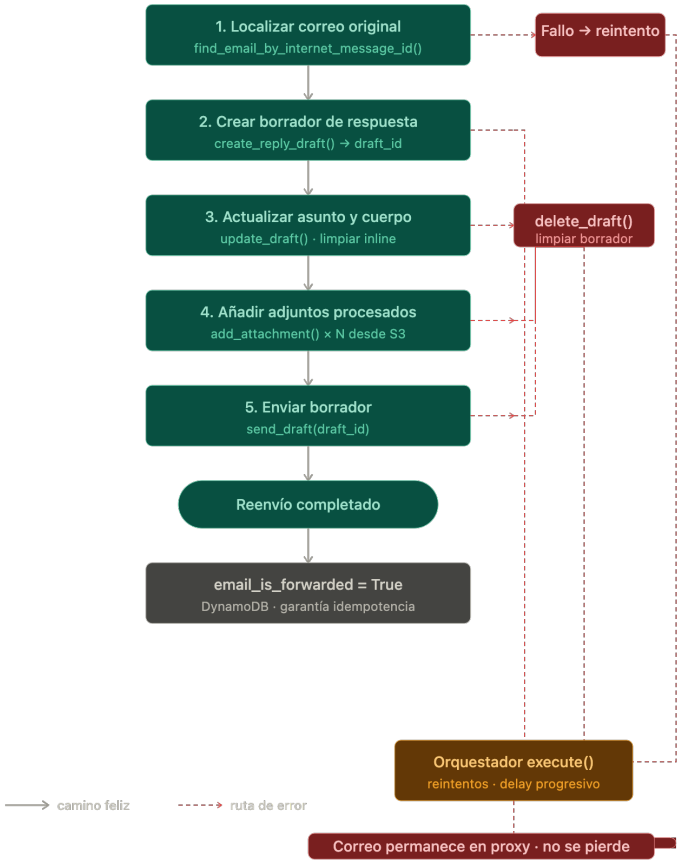


Figura 5.5: Flujo de reenvío de `EmailReplyForwarder`: puntos de fallo y mecanismos de recuperación

**Primera capa: limpieza local ante fallo durante el reenvío.** Si cualquiera de los pasos 3, 4 o 5 del flujo falla después de que el borrador ya ha sido creado en el paso 2, el bloque `except` intenta eliminar el borrador mediante `delete_draft` antes de propagar la excepción. Esta limpieza evita que queden borradores huérfanos en el buzón que pudieran confundirse con correos reales y sobre todo, evitan que se envíen correos con información parcial o incompleta. Se incluye además una salvaguarda explícita: si por algún motivo el `draft_id` coincidiera con el `object_id` del correo original, lo que indicaría un error grave en la API, la eliminación se cancela para evitar destruir el correo original. Si los pasos 1 o 2 fallan, no existe borrador que limpiar y la excepción se propaga directamente.

**Segunda capa: reintentos en el orquestador.** La excepción propagada por el flujo es recibida por el orquestador `execute()` de `EmailProcessorBase`, que implementa un mecanismo de reintentos con delay progresivo de `retries * 3 + 1` segundos entre intentos. Este delay creciente reduce la presión sobre la API de Microsoft Graph en situaciones de degradación transitoria y añade variación natural entre reintentos para evitar que múltiples invocaciones concurrentes saturen la API simultáneamente. Tras cinco reintentos sin éxito, el orquestador eleva el error como incidencia para revisión manual del equipo de soporte.

**Idempotencia mediante DynamoDB.** Una vez completado el reenvío con éxito, el orquestador actualiza el flag `email_is_forwarded` a `True` en DynamoDB. Si por cualquier motivo la lambda se reinvocara sobre el mismo correo, por ejemplo, ante un fallo en la actualización de DynamoDB después de un reenvío exitoso, el orquestador detecta que el flag ya está a `True` y omite el procesamiento sin volver a reenviar el correo, evitando duplicados en el buzón de destino y en TRAMES.

**Correo permanece en el buzón proxy.** En todos los escenarios de fallo, tanto si se agotan los reintentos como si el fallo es irrecuperable, el correo permanece en el buzón proxy sin ser eliminado. La lambda `email_deleter` solo actúa sobre correos cuyo flag `email_is_forwarded` está a `True`, por lo que un correo que no ha podido reenviarse nunca será eliminado automáticamente. Esta dependencia entre lambdas garantiza que, ante cualquier fallo no recuperable, el correo sigue disponible en el buzón proxy, para su gestión manual o para un nuevo intento una vez resuelta la incidencia.

## 5.3. Plan de contingencia

El plan de contingencia cubre el escenario de fallo total del servicio DATA&IA, es decir, aquella situación en la que el sistema deja de procesar los correos del

Inbox del buzón proxy. Este fallo puede producirse por múltiples causas: fallo en la conexión al Inbox, llenado inesperado del buzón, caída del servicio de clasificación, incluyendo una interrupción de la API de IA generativa o modificaciones de red, o fallo en el reenvío al buzón de destino.

Independientemente del motivo, el plan de actuación sigue siempre la misma secuencia, articulada en tres fases.

**Fase 1: Detección y aislamiento.** La incidencia puede ser detectada por DATA&IA mediante alertas internas de CloudWatch o bien por el equipo de Negocio, que observa que los correos del buzón principal dejan de recibir respuestas del sistema. En el momento de la detección se ejecutan de forma inmediata las siguientes acciones:

- Se detiene la regla del buzón `prestacionesautos@mapfre.com` que redirige los correos entrantes al buzón proxy, evitando que sigan acumulándose correos que no van a poder procesarse.

- DATA&IA detiene el servicio de lectura del Inbox del buzón proxy, garantizando que el sistema queda en estado detenido hasta resolver la incidencia.

- DATA&IA estima el RTO (*Recovery Time Objective*) en función del tipo de incidencia y comunica a Negocio si la resolución se prevé dentro del plazo máximo de 2 días laborables o si requerirá más tiempo.

Durante esta fase, es importante que el responsable del buzón proxy no elimine ni mueva correos del Inbox sin conocimiento previo del equipo DATA&IA, y que no marque ni desmarque correos como leídos, ya que el estado de lectura es el mecanismo que permite discriminar qué correos han sido ya reenviados.

**Fase 2: Gestión de los correos retenidos.** Los correos acumulados en el Inbox del buzón proxy durante la ventana de incidencia deben ser gestionados en función del volumen y del plazo de resolución estimado, según la tabla 5.2:

| Acción | Plazo | Condición |
|---|---|---|
| Esperar a que el servicio se restablezca | $\leq$ 2 días laborables | Recomendada por DATA&IA en la mayoría de casos |
| Movimiento manual de correos del buzón proxy a `prestacionesautos` (por Negocio) | $>$ 2 días laborables | $\leq$ 500 correos a mover |
| Movimiento automático mediante Power Automate (por DATA&IA) | $>$ 2 días laborables | $>$ 500 correos (lotes de 25–50 por ejecución) |

Cuadro 5.2: Acciones del plan de contingencia según volumen y plazo

La opción recomendada por DATA&IA es esperar a que el servicio se restablezca, ya que históricamente las incidencias del sistema se han resuelto en periodos cortos. Los correos retenidos serán clasificados automáticamente cuando el servicio se reactive, reduciendo al mínimo la intervención manual.

**Fase 3: Reactivación del servicio.** Una vez resuelta la incidencia, el servicio se restaura en el siguiente orden: DATA&IA activa el servicio de lectura del Inbox del buzón proxy, que procesará automáticamente los correos remanentes; una vez vaciado el Inbox, se notifica al responsable del buzón `prestacionesautos@mapfre.com` para que reactive la regla de redireccionamiento al buzón proxy, restableciendo el flujo normal de procesamiento.

**Power Automate como mecanismo de reenvío masivo.** Para los casos en que el volumen de correos retenidos supera los 500 y la espera no es asumible, DATA&IA dispone de un flujo de Power Automate que permite el reenvío masivo desde el buzón proxy al buzón principal. El flujo se ejecuta manualmente en lotes de 25–50 correos por ejecución y realiza las siguientes acciones: lee los correos marcados como no leídos de la carpeta `RecibidosContingencia`, los reenvía a `prestacionesautos@mapfre.com` conservando adjuntos, asunto y cuerpo originales, mueve el correo original a la carpeta `EnviadosPrestaciones` como copia de seguridad temporal, e introduce un retraso aleatorio de 2–6 segundos entre envíos para evitar bloqueos de seguridad. La figura 5.6 muestra el diagrama del flujo implementado.


Figura 5.6: Flujo de Power Automate para el reenvío masivo en situaciones de contingencia

Este mecanismo se recomienda únicamente en casos extremos en los que el ser-

vicio no pueda restablecerse en el corto-medio plazo, dado que presenta limitaciones inherentes a su naturaleza de herramienta genérica: puede producirse pérdida puntual de adjuntos o duplicación de imágenes en el cuerpo del mensaje, y su ejecución consume recursos del equipo DATA&IA que deberían dedicarse a resolver la incidencia principal. La decisión de activarlo debe tomarse de forma conjunta entre DATA&IA y Negocio valorando los riesgos en cada caso concreto.

## 5.4. Monitorización

La observabilidad del sistema se gestiona en dos capas complementarias: una capa técnica basada en Amazon CloudWatch, que supervisa el estado de los componentes de infraestructura y detecta anomalías en tiempo real, y una capa funcional basada en Power BI, que expone los KPIs de negocio descritos en la sección 6.1.1 para el seguimiento del impacto operativo del sistema.

### 5.4.1. Monitorización técnica: Amazon CloudWatch

Toda la infraestructura del sistema emite logs y métricas hacia Amazon CloudWatch, que actúa como capa central de observabilidad técnica. Sobre esta base se han definido tres tipos de alarmas: alarmas técnicas, que monitorizan el comportamiento de los componentes de AWS; alarmas de producto, que detectan situaciones de negocio anómalas a partir de patrones en los logs; y alarmas funcionales, que calculan ratios compuestos sobre métricas del pipeline.

Toda la infraestructura de alarmas está definida mediante Terraform, lo que garantiza su reproducibilidad y versionado junto con el resto de la infraestructura del sistema. Todas las alarmas notifican a través de un topic SNS que enruta las alertas al canal de soporte correspondiente según su prioridad.

**Alarmas técnicas.** Cada Lambda del pipeline dispone de una alarma sobre la métrica `Errors` de CloudWatch, con umbrales y prioridades calibrados en función de la criticidad de cada componente. La tabla 5.3 recoge las alarmas definidas con su umbral y prioridad.

| Componente | Métrica | Umbral | Prioridad |
|---|---|---|---|
| `receiver_mailbox` | Errors (30 min) | $\geq 1$ | P1 |
| `extract_attachments` | Errors (30 min) | $\geq 10$ | P3 |
| `extract_text` | Errors (30 min) | $\geq 15$ | P2 |
| `classifier_iagen` | Errors (30 min) | $\geq 15$ | P2 |
| `email_forward` | Errors (30 min) | $\geq 20$ | P1 |
| `email_deleter` | Errors (30 min) | $\geq 20$ | P3 |
| `contingency` | Errors (1 min) | $\geq 1$ | P1 |
| `monitor` | Errors (1 min) | $\geq 1$ | P1 |
| Step Function principal | ExecutionsFailed (1 min) | $\geq 1$ | P1 |
| Step Function monitorización | ExecutionsFailed (30 min) | $\geq 1$ | P1 |
| SQS — antigüedad mensaje | ApproximateAgeOfOldestMessage | $\geq$ 900s | P1 |
| SQS — mensajes acumulados | ApproximateNumberOfMessagesVisible | $\geq 100$ | P1 |

Cuadro 5.3: Alarmas técnicas de CloudWatch por componente

Las dos alarmas sobre la cola SQS merecen mención especial: la primera detecta mensajes que llevan más de 15 minutos sin procesarse, lo que puede indicar un bloqueo en las Lambdas consumidoras; la segunda detecta una acumulación anormal de mensajes pendientes, síntoma de que el pipeline no está procesando al ritmo de ingesta.

**Alarmas de producto.** El sistema define alarmas de producto que detectan situaciones de negocio anómalas mediante filtros (*metric filters*) sobre los logs de CloudWatch. Cada filtro extrae un patrón de texto específico de los logs de una Lambda y genera una métrica personalizada en el namespace del proyecto, sobre la que se define la alarma correspondiente. Entre las más relevantes:

- **Correos con tamaño excesivo**: detecta correos que superan el límite de 25.000 KB en `receiver_mailbox`, que requieren gestión manual fuera del pipeline automático.
- **Correos posiblemente duplicados**: detecta correos con estado `SUCCEEDED` en DynamoDB que vuelven a intentar procesarse, lo que puede indicar un problema de idempotencia.
- **Reintentos excesivos a la API de Gemini**: detecta cuando `extract_text` o `classifier_iagen` superan 10 reintentos a Vertex AI en 30 minutos, indicando posible saturación o caída del servicio de IA generativa.
- **Reintentos agotados en el reenvío**: detecta cuando `email_forward` supera el umbral de 5 reintentos para un correo, requiriendo revisión manual.

- **Activación del fallback de encapsulado**: detecta cuando `email_forward` recurre a la estrategia `ENCAPSULATE_EMAIL` como contingencia ante un fallo interno, indicando que el correo se ha reenviado sin clasificar.

**Alarmas funcionales.** A diferencia de las anteriores, las alarmas funcionales calculan ratios compuestos combinando múltiples métricas mediante *metric queries* de CloudWatch. El sistema implementa dos:

- **Ratio de adjuntos sobre correos** (`attach_vs_emails`): calcula el porcentaje de correos con adjuntos sobre el total de invocaciones de `extract_attachments`. Se activa cuando el ratio cae por debajo del 30 %, lo que podría indicar una anomalía en la extracción de adjuntos.
- **Ratio de reenvíos KO sobre total** (`emails_ko_vs_total`): calcula el porcentaje de correos con estado KO en `email_forward` sobre el total de invocaciones de `extract_attachments`. Se activa cuando supera el 15 %, señalando una degradación generalizada del pipeline de reenvío.

### 5.4.2. Monitorización funcional: Power BI

La capa funcional de monitorización se materializa en dashboards de Power BI que consumen los datos persistidos en DynamoDB y S3 a través de una capa de virtualización de datos basada en Denodo, desplegada sobre Amazon EKS y accesible mediante un gateway ODBC expuesto a través de un VPC Endpoint. Esta arquitectura permite construir cuadros de mando sobre los KPIs de negocio definidos en la sección 6.1.1 sin exponer directamente la infraestructura interna de AWS.

Los dashboards proporcionan visibilidad diaria sobre el volumen de correos procesados, las tasas de clasificación y extracción del número de referencia, los errores producidos por componente y el ahorro operativo acumulado. Esta capa está orientada al equipo de negocio y a los gestores del buzón, complementando la visibilidad técnica de CloudWatch con una perspectiva funcional que permite evaluar el impacto del sistema sobre el proceso de tramitación.

# Capítulo 6

# Evaluación

## 6.1. Metodología de evaluación

La evaluación del sistema EMM se articula en dos fases complementarias que responden a momentos distintos del ciclo de vida del producto.

La primera fase es la **validación técnica**, llevada a cabo durante el desarrollo mediante dos rondas de pruebas previas realizadas de manera individual tras las que se realizaron otras dos pruebas pero esta vez conjuntas con el equipo de negocio, denominadas POCs (Prove of Concept) 1 y 2, sobre batches reales de correos del buzón piloto de prestaciones de autos. El objetivo de esta fase es validar que el modelo alcanza un nivel de precisión suficiente en las dos tareas principales del sistema: la clasificación de adjuntos y la extracción del número de referencia del siniestro.

La segunda fase es la **validación de negocio**, que medirá el impacto operativo real del sistema una vez entre en producción. Esta validación se apoya en un conjunto de KPIs definidos conjuntamente con el equipo de negocio y en los tiempos medios de operación (TMOs) descritos en la sección 1.1, que actúan como línea base del coste manual de cada tarea automatizada. El mecanismo central de esta medición es el grupo de acción y control descrito en la sección 4.3.2: el 10 % de correos que se procesan sin clasificar sirve de referencia para cuantificar el ahorro real que aporta el sistema frente al flujo manual.

Dado que el sistema se encuentra en el momento de redacción de este trabajo en fase de validación previa a la puesta en producción, los resultados de negocio recogidos en este capítulo son los obtenidos durante las POCs. Los KPIs de negocio y el cálculo del ROI a partir de los TMOs se presentan como el marco de medición que se activará en producción, pero no disponen aún de datos reales medibles.

### 6.1.1. KPIs de negocio

Los KPIs definidos con el equipo de negocio para el seguimiento del sistema en producción se recogen en el cuadro 6.1. Se organizan en tres grupos: KPIs de volumen y procesamiento, que describen el flujo de correos a través del pipeline; KPIs de excepción, que identifican los casos que requieren intervención manual; y KPIs de rendimiento y ahorro, que cuantifican el impacto operativo del sistema.

| **KPI** | **Descripción** |
|---|---|
| *Volumen y procesamiento* | |
| KPI1 | Total de emails entrantes al buzón |
| KPI2 | Total de emails procesados por el pipeline |
| KPI3 | Emails con error en el procesamiento |
| KPI4 | Total de emails en alcance del sistema |
| KPI5 | Total de emails sin adjuntos |
| *Excepciones* | |
| KPI6 | Emails con número de expediente KO |
| KPI7 | Emails con número de expediente OK |
| KPI8 | Emails con adjuntos no soportados por TRAMES |
| KPI9 | Acierto únicamente en búsqueda de número de expediente |
| KPI10 | Acierto únicamente en catalogación |
| KPI11 | Sin número de expediente y sin catalogar |
| KPI12 | Número de expediente encontrado o catalogado por TRAMES |
| KPI17 | Total de emails sin número de expediente |
| KPI19 | Emails con error en el envío |
| *Rendimiento y ahorro* | |
| KPI20 | Ahorro operativo grupo de acción (CASO 1) |
| KPI21 | Ahorro operativo grupo de control (CASO 2) |
| KPI22 | Ahorro operativo combinado (CASO 1 y CASO 2) |
| KPI23 | Tiempo medio de procesamiento del email (TMO) |
| KPI24 | Volumen total de adjuntos recibidos |
| KPI25 | Volumen de adjuntos procesados por el sistema |

Cuadro 6.1: KPIs de negocio definidos para el seguimiento del sistema EMM en producción

Los KPIs de ahorro (KPI20, KPI21, KPI22) se calcularán a partir de los TMOs descritos en la sección 1.1, aplicando el coste unitario por minuto de tramitador al volumen de operaciones automatizadas por el sistema. La diferencia entre el tiempo registrado en el grupo de control y el tiempo invertido en el grupo de acción reflejará directamente el ahorro operativo unitario por correo, que agregado

al volumen total permitirá calcular el retorno de la inversión de forma rigurosa y basada en datos reales de uso.

## 6.2. Validación técnica inicial

Con carácter previo a la validación conjunta con el equipo de negocio, se realizaron dos batches de 100 correos cada uno, seleccionados de forma representativa del volumen real del buzón de prestaciones de autos. El objetivo de esta fase era alcanzar un nivel mínimo de calidad que hiciera productivas las reuniones conjuntas con negocio, evitando invertir tiempo de los tramitadores en corregir errores que podían detectarse y resolverse de forma autónoma.

### 6.2.1. Primer batch

El primer batch reveló varios problemas sistemáticos en el comportamiento del modelo. El más relevante fue la tendencia a clasificar documentos basándose en elementos secundarios del adjunto (membrete, firma, logotipo) en lugar del contenido principal, lo que producía una sobreclasificación de la categoría *Comunicación de Terceros*. También se detectaron errores recurrentes en documentos con contenido mixto, donde el modelo no era capaz de identificar la categoría del elemento más relevante. Un tercer patrón identificado fue la invención o renombrado de categorías: en ocasiones el modelo devolvía nombres ligeramente distintos a los definidos en el catálogo, lo que hacía fallar la validación posterior. Este problema se resolvió reforzando en el prompt la instrucción de coincidencia literal con el catálogo y añadiendo la capa de validación contra `categories.json` descrita en la sección 5.1.4.

A partir de los errores detectados se introdujeron las primeras versiones de las reglas de desempate descritas también en la sección 5.1.4 y se refinaron las descripciones de las categorías más problemáticas.

### 6.2.2. Segundo batch

El segundo batch de 100 correos sirvió para validar que las correcciones introducidas resolvían los patrones identificados sin generar nuevos errores en categorías que anteriormente funcionaban bien, lo cual sabíamos que era un riesgo habitual al modificar prompts, donde mejorar un caso puede degradar otro. Los resultados mostraron una mejora significativa en las categorías problemáticas identificadas, con una reducción notable de los errores de *Comunicación de Terceros* y documentos mixtos, lo que indicó que el prompt estaba en condiciones de ser validado con el equipo de negocio.

## 6.3. POC 1

Una vez alcanzado un nivel de precisión suficiente en la validación técnica inicial, se realizaron dos rondas de validación conjunta con el equipo de negocio, denominadas POC 1 y POC 2, sobre batches reales de correos del buzón piloto de prestaciones de autos (alrededor de 300 correos por batch). El objetivo de esta fase es validar con el equipo especializado que el sistema alcanza un nivel de precisión suficiente en las tareas principales del sistema.

Los conjuntos de datos sobre los que se realizaron las POCs fueron seleccionados de forma representativa del volumen real del buzón de prestaciones de autos, con el objetivo de validar el sistema en condiciones lo más cercanas posible a las reales. El incremento del tamaño del batch respecto a la validación técnica inicial responde a la necesidad de obtener una muestra estadísticamente más representativa de la distribución real de tipos documentales, incluyendo categorías poco frecuentes que difícilmente aparecen en muestras de 100 correos.

La primera ronda de validación conjunta con el equipo de negocio se realizó sobre un batch de 315 correos reales del buzón de prestaciones de autos, que generaron un total de 536 adjuntos a clasificar. Los resultados se recogen en el cuadro 6.2.

| **Métrica** | **Valor** | **Detalle** |
|---|---|---|
| Total correos evaluados | 315 | — |
| Total adjuntos procesados | 536 | — |
| NumRef encontrado (total) | 81,3 % | 256/315 |
| NumRef encontrado (relativo) | 100 % | 256/256 |
| Clasificación OK (total) | 70,3 % | 377/536 |
| Clasificación OK (modelo) | 99,4 % | 377/379 |
| Correos resueltos end-to-end | 80,6 % | 254/315 |

Cuadro 6.2: Resultados de la POC 1 (315 correos)

### 6.3.1. Extracción del número de referencia

La tasa total del 81,3 % refleja que 59 correos del batch no contenían número de referencia en ninguna parte del correo ni sus adjuntos, por lo que no era posible extraerlo independientemente del rendimiento del sistema. Excluyendo estos casos, es decir, considerando únicamente los correos de los que el sistema podía extraer el número de referencia, la tasa asciende al 100 %, lo que indica que el sistema no cometió ningún error de extracción cuando la información estaba disponible.

### 6.3.2. Clasificación de adjuntos

La tasa total del 70,3 % incluye adjuntos cifrados, en formatos no transformables u otras situaciones que escapan al control del modelo. La tasa de eficiencia del modelo, que excluye estos casos y mide únicamente los adjuntos que el sistema podía procesar, alcanza el 99,4 %, lo que supone un resultado técnicamente sólido.

### 6.3.3. Análisis de errores y mejoras introducidas

La revisión conjunta con los tramitadores de los casos incorrectos puso de manifiesto que la mayoría de los errores restantes respondían a ambigüedades propias del dominio asegurador que solo los tramitadores podían resolver. A partir de este análisis se introdujeron nuevas reglas de desempate en el prompt y se refinaron las descripciones de las categorías más problemáticas, especialmente en los casos de documentos con estructura económica y documentos mixtos, con el objetivo de mejorar los resultados en la POC 2.

## 6.4. POC 2

La segunda ronda de validación conjunta con el equipo de negocio se realizó sobre otro batch diferente de 300 correos reales del buzón de prestaciones de autos, que generaron un total de 676 adjuntos a clasificar. Los resultados se recogen en el cuadro 6.3.

| **Métrica** | **Valor** | **Detalle** |
|---|---|---|
| Total correos evaluados | 300 | — |
| Total adjuntos procesados | 676 | — |
| NumRef encontrado (total) | 91,9 % | 275/300 |
| NumRef encontrado (relativo) | 98,9 % | 275/278 |
| Clasificación OK (total) | 94,8 % | 641/676 |
| Clasificación OK (modelo) | 98,9 % | 641/648 |
| Correos resueltos end-to-end | 88,0 % | 264/300 |

Cuadro 6.3: Resultados de la POC 2 (300 correos)

### 6.4.1. Extracción del número de referencia

La tasa total del 91,9 % refleja que 25 correos del batch no contenían número de referencia en ninguna parte del correo ni sus adjuntos, por lo que no era posible extraerlo independientemente del rendimiento del sistema. Excluyendo estos casos,

la tasa asciende al 98,9 %, lo que indica que el sistema prácticamente no falla cuando la información está disponible.

### 6.4.2. Clasificación de adjuntos

La tasa de eficiencia del modelo, que excluye estos casos y mide únicamente los adjuntos que el sistema podía procesar, alcanza el 98,9 %, resultado que implica continuidad con respecto al batch anterior. La comparación frente al sistema NLP clásico, que requiere contrastar métricas homologables, se analiza en detalle en la sección 6.5.

## 6.5. Análisis comparativo

### 6.5.1. Evolución entre POC 1 y POC 2

Los resultados de ambas rondas de validación se presentan en el cuadro 6.4.

| **Métrica** | **POC 1** | **POC 2** |
|---|---|---|
| NumRef encontrado (total) | 81,3 % | 91,9 % |
| NumRef encontrado (relativo) | 100 % | 98,9 % |
| Clasificación OK (total) | 70,3 % | 94,8 % |
| Clasificación OK (modelo) | 99,4 % | 98,9 % |
| Correos resueltos end-to-end | 80,6 % | 88,0 % |

Cuadro 6.4: Resultados de POC 1 y POC 2

El dato más relevante es la consistencia de la tasa de eficiencia del modelo en ambas rondas, que se mantiene por encima del 98 % en los dos casos. Esto confirma que el modelo clasifica correctamente prácticamente todos los adjuntos que puede procesar con independencia de la composición del batch evaluado. Las diferencias observadas en las tasas totales responden fundamentalmente a la distinta proporción de adjuntos en formatos no procesables y correos sin número de referencia en cada batch, factores inherentes a la variabilidad natural del buzón que escapan al control del modelo.

Por su parte, la tasa de resolución end-to-end por correo mejora de forma apreciable entre rondas, pasando del 80,6 % en la POC 1 al 88,0 % en la POC 2, lo que refleja el efecto de las reglas de desempate y refinamientos del prompt introducidos tras el análisis de errores de la primera ronda.

### 6.5.2. Comparativa normalizada frente al sistema NLP clásico

Para que la comparación sea justa, no solo vamos a usar la métrica global del sistema NLP anterior (ese 25 % de éxito sobre más de un millón de correos). Mezclar el rendimiento de distintos buzones corporativos (como patrimonial y autos) desvirtuaría la realidad del análisis.

Por lo tanto, hemos establecido una **línea base real**, comparando el nuevo sistema EMM exclusivamente contra los datos históricos del sistema antiguo en este mismo buzón de Prestaciones de Autos.

**El impacto real en los datos**

En nuestra prueba final (POC 2), el sistema EMM fue capaz de resolver de principio a fin el 88,0 % de los casos (264 de 300 correos). Matemáticamente, el margen de error nos garantiza que, una vez desplegado en producción, el rendimiento operativo del sistema se mantendrá por encima del 84,3 % incluso en los escenarios más conservadores.

Para entender de dónde viene este éxito, desglosamos los resultados en las tareas clave para el negocio:

- **Clasificación de documentos:** Si descartamos aquellos archivos que humanamente tampoco se podrían procesar (como documentos cifrados o formatos corruptos), el modelo de IA acertó la categoría exacta en el 98, 9 % de los casos (641 aciertos sobre 648 adjuntos procesables). Esto demuestra que el sistema entiende perfectamente la naturaleza del documento que tiene delante.

- **Localización del siniestro:** Cuando el número de referencia existía en el correo o en sus adjuntos, el sistema fue capaz de encontrarlo el 98, 9 % de las veces (275 aciertos de 278 casos posibles). Esto confirma que procesar el documento al completo con un modelo multimodal es mucho más fiable que buscar patrones de texto como hacía el sistema anterior.

- **Salto cualitativo:** La brecha entre el límite mínimo garantizado por el nuevo modelo (84,3 %) y el rendimiento histórico del sistema anterior es definitiva. La mejora es estructural y estadísticamente probada, descartando que los buenos resultados sean producto de la casualidad en la muestra elegida.

En definitiva, los datos confirman que la adopción de Gemini 2.5 Flash junto a la implementación de las mejoras en el pipeline transforman completamente el proceso. La capacidad para procesar información sin reentrenamientos (*zero-shot*) y de tratar los documentos de forma integral supone una solución real al cuello de botella del procesamiento documental.

# Capítulo 7

# Conclusiones y trabajo futuro

El presente Trabajo de Fin de Máster ha abordado el diseño e implementación de un sistema de gestión automatizada de correos electrónicos basado en modelos de lenguaje de gran escala, desplegado sobre una arquitectura serverless en AWS y validado sobre el buzón piloto de prestaciones de autos de MAPFRE.

A lo largo del proyecto se han alcanzado los objetivos planteados en la sección 1.2. El sistema implementa un pipeline completo que cubre desde la ingesta del correo hasta su reenvío al sistema transaccional TRAMES, pasando por la extracción de adjuntos, la clasificación documental y la extracción del número de referencia del siniestro. La estrategia híbrida de extracción, que combina las capacidades multimodales de Gemini 2.5 Flash con Tesseract como mecanismo de contingencia, ha demostrado ser eficaz ante la heterogeneidad de los adjuntos recibidos en un entorno real de producción. La adaptación del modelo mediante prompt engineering iterativo ha permitido ajustar el comportamiento del sistema a los requisitos de negocio en ciclos cortos de validación conjunta con el equipo de tramitadores, sin necesidad de recurrir al fine-tuning.

Los resultados de las dos pruebas de concepto realizadas confirman la viabilidad técnica de la aproximación. A nivel de clasificación de adjuntos, la tasa de eficiencia del modelo se mantiene por encima del 98 % en ambas rondas de validación. Más relevante para la comparación con el sistema anterior es la tasa de resolución end-to-end por correo , que alcanza el 88,0 % en la POC 2 frente al 25 % del sistema NLP clásico que el proyecto viene a sustituir, teniendo en cuenta las diferencias entre ambos sistemas que se explican en la sección 6.5. Estos resultados son especialmente relevantes si se considera que se han obtenido con un enfoque zero-shot, sin ejemplos de entrenamiento específicos.

Más allá de la clasificación, el proyecto ha resuelto problemas técnicos de integración que condicionaban la viabilidad del sistema en un entorno corporativo real: la estrategia de reenvío mediante reply sobre el correo original garantiza la trazabilidad en TRAMES, el truncado inteligente del asunto preserva el número

de referencia ante el límite de 255 caracteres de Microsoft 365, y el sistema de resiliencia en dos capas asegura que ningún correo se pierde bajo ninguna circunstancia. La monitorización técnica y funcional, articulada mediante CloudWatch y Power BI, y el plan de contingencia basado en Power Automate completan un sistema preparado para operar en producción con las garantías que exige un entorno corporativo.

El diseño orientado a la generalización, materializado a través del loader pattern, permite que la incorporación de un nuevo buzón al sistema se reduzca a la definición de un nuevo proyecto con su propio summarizer, sus prompts y su catálogo de categorías, sin modificar la arquitectura subyacente. Esta capacidad de extensión es una de las aportaciones más relevantes del proyecto, ya que convierte al sistema en una plataforma reutilizable y no en una solución ad hoc para un único buzón.

No obstante, el trabajo presenta limitaciones que deben reconocerse. Los resultados de las POCs validan el rendimiento técnico del modelo, pero el impacto operativo real, cuantificado mediante los KPIs de negocio y los TMOs definidos con el equipo de tramitadores, solo será medible una vez el sistema entre en producción y se disponga de datos reales del grupo de acción y control. Asimismo, la validación se ha realizado exclusivamente sobre el buzón piloto de prestaciones de autos, por lo que la generalización a otros buzones con distribuciones de tipos documentales distintas requerirá validaciones adicionales.

De cara al trabajo futuro, se identifican las siguientes líneas de evolución del sistema:

- **Puesta en producción y medición del ROI**: la activación del sistema en el entorno productivo permitirá medir el ahorro operativo real a partir de los TMOs y el grupo de acción y control, validando así el impacto de negocio que las POCs anticipan a nivel técnico.
- **Extensión a nuevos buzones**: la incorporación de buzones de otros ramos de negocio como vida, hogar, salud, aprovechando el loader pattern existente, lo que permitirá validar la generalización del sistema en dominios con vocabularios y tipos documentales distintos.
- **Transformación de formatos**: la ampliación de los formatos transformables por el sistema que ya contempla la descompresión de ZIPs, la extracción de correos encapsulados y la conversión de formatos de imagen no soportados por TRAMES, para reducir la proporción de adjuntos actualmente clasificados como no procesables.
- **Integración directa con TRAMES**: la conexión vía API con el sistema transaccional permitiría completar el flujo de automatización extremo a ex-

tremo, eliminando la dependencia del reenvío por correo electrónico como mecanismo de integración.

- **Gestión del cambio**: la introducción de un sistema de clasificación automática transforma el rol del tramitador, que pasa de ejecutar tareas manuales repetitivas a supervisar y validar los resultados del sistema. Esta transición requiere un plan de gestión del cambio que incluya la formación del equipo de tramitadores en el nuevo flujo de trabajo, la definición de protocolos de supervisión para los correos del grupo de acción, y la gestión de las expectativas y resistencias habituales ante la automatización de tareas. El éxito del sistema en producción dependerá en gran medida de la adopción efectiva por parte de los usuarios finales.
- **Evolución del modelo**: la evaluación periódica de nuevas versiones de Gemini u otros modelos multimodales que pudieran ofrecer mejoras en rendimiento, latencia o coste, manteniendo la arquitectura actual del sistema y sustituyendo únicamente el modelo consumido a través de Vertex AI.

# Bibliografía

[AlShaikh et al., 2025] AlShaikh, M., Alrajeh, Y., Alamri, S., Melhem, S., and Abu-Khadrah, A. (2025). Supervised methods of machine learning for email classification: a literature survey. *Systems Science & Control Engineering*, 13(1).

[Comanici et al., 2025] Comanici, G., Bieber, E., Schaekermann, M., Pasupat, I., Sachdeva, N., Dhillon, I., Blistein, M., Ram, O., Zhang, D., Rosen, E., Marris, L., Petulla, S., Gaffney, C., Aharoni, A., Lintz, N., Pais, T. C., Jacobsson, H., Szpektor, I., Jiang, N.-J., Haridasan, K., Omran, A., Calian, M., Huang, M., van den Oord, A., Goyal, N., Chen, T., Rawlani, P., Schallhart, C., Lokhande, S., Luo, X., Shan, J., Montgomery, C., vision, V., Piccinini, F., Barak, O., Cui, J., Jia, Y., Dektiarev, M., Kolganov, A., Huang, S., Chen, Z., Wang, X., Agarwal, M., Kwasiborski, S., Sandhu, P., Siegler, P., Iscen, A., Ben-David, E., Butt, S., Allamanis, M., Benjamin, S., Busa-Fekete, R., Hernandez-Campos, F., Goldshtein, S., Dibb, M., Zhang, W., Marsden, A., Radebaugh, C., Roller, S., Nayyar, A., Austin, J., and Tayfun (2025). Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *arXiv preprint arXiv:2507.06261*.

[Cornia et al., 2024] Cornia, M. et al. (2024). The revolution of multimodal large language models: A survey. *arXiv preprint arXiv:2402.12451*.

[Dong et al., 2024] Dong, Q., Li, L., Dai, D., Zheng, C., Wu, Z., Chang, B., Sun, X., Xu, J., and Sui, Z. (2024). A survey on in-context learning. *arXiv preprint arXiv:2301.00234 (v4, revised 2024)*.

[Greif et al., 2024] Greif, K. et al. (2024). Multimodal LLMs for OCR, OCR post-correction, and named entity recognition in historical documents. *arXiv preprint arXiv:2504.00414*.

[Jangda et al., 2019] Jangda, A., Pinckney, D., Brun, Y., and Guha, A. (2019). Formal foundations of serverless computing. *Proceedings of the ACM on Programming Languages*, 3(OOPSLA):1–26.

[Jonas et al., 2019] Jonas, E., Schleier-Smith, J., Sreekanti, V., et al. (2019). Cloud programming simplified: A berkeley view on serverless computing. Technical report, UC Berkeley.

[Li et al., 2020] Li, Q., Peng, H., Li, J., et al. (2020). A survey on text classification: From traditional to deep learning. *arXiv preprint arXiv:2008.00364*.

[Liu et al., 2022] Liu, J., Shen, D., Zhang, Y., Dolan, B., Carin, L., and Chen, W. (2022). What makes good in-context examples for GPT-3? In *Proceedings of the Third Workshop on Deep Learning for Low-Resource NLP (DeepLo)*, pages 100–114. Association for Computational Linguistics.

[Mackay, 1988] Mackay, W. E. (1988). Diversity in the use of electronic mail: A preliminary inquiry. *ACM Transactions on Information Systems (TOIS)*, 6(4):380–397.

[Malone et al., 1987] Malone, T. W., Grant, K. R., Turbak, F. A., Brobst, S. A., and Cohen, M. D. (1987). Intelligent information-sharing systems. *Communications of the ACM*, 30(5):390–402.

[Morris, 2020] Morris, K. (2020). *Infrastructure as Code: Dynamic Systems for the Cloud Age*. O'Reilly Media, 2nd edition.

[Park et al., 2024] Park, J. et al. (2024). Hierarchical visual feature aggregation for OCR-free document understanding. *arXiv preprint arXiv:2411.05254*.

[Sahami et al., 1998] Sahami, M., Dumais, S., Heckerman, D., and Horvitz, E. (1998). A bayesian approach to filtering junk e-mail. In *Learning for Text Categorization: Papers from the 1998 Workshop*, Madison, Wisconsin. AAAI Technical Report WS-98-05.

[Sahoo et al., 2024] Sahoo, P., Singh, A. K., Saha, S., Jain, V., Mondal, S., and Chadha, A. (2024). A systematic survey of prompt engineering in large language models: Techniques and applications. *arXiv preprint arXiv:2402.07927*.

[Sharma et al., 2025] Sharma, V., Jani, S., Jain, R., Hajela, K., Shah, A., and Patel, J. T. (2025). Paddle OCR vs. Tesseract: A comparative performance analysis on the Gujarati script. *International Journal of Research Publication and Reviews (IJRPR)*, 6(10):1224–1228.

[Smith, 2007] Smith, R. (2007). An overview of the Tesseract OCR engine. In *Proceedings of the Ninth International Conference on Document Analysis and Recognition (ICDAR)*, volume 2, pages 629–633. IEEE.

[Vangibhurathachhi, 2025] Vangibhurathachhi, S. K. (2025). Adapting large language models: A comparative study of prompt engineering and fine-tuning. *International Journal of Emerging Research in Engineering and Technology (IJERET)*, 6(3):9–15.

[Vaswani et al., 2017] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. In *Advances in Neural Information Processing Systems*.

[Verdet et al., 2024] Verdet, A., Hamdaqa, M., Da Silva, F., and Khomh, F. (2024). Assessing the adoption of security policies by developers in Terraform across different cloud providers. *Empirical Software Engineering*, 29(5):1–35.

[Wang et al., 2022] Wang, X., Wei, J., Schuurmans, D., Le, Q., Chi, E., Narang, S., Chowdhery, A., and Zhou, D. (2022). Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations (ICLR)*.

[Wei et al., 2022] Wei, J., Wang, X., Schuurmans, D., Bosma, M., Chi, E., Le, Q., and Zhou, D. (2022). Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems*, 35.

[Willard and Louf, 2023] Willard, B. T. and Louf, R. (2023). Efficient guided generation for large language models. *arXiv preprint arXiv:2307.09702*.

[Yin et al., 2024] Yin, S., Fu, C., Zhao, S., Li, K., Sun, X., Xu, T., and Chen, E. (2024). A survey on multimodal large language models. *National Science Review*, 11(12).

[Zhang et al., 2022] Zhang, Z., Zhang, A., Li, M., and Smola, A. (2022). Automatic chain of thought prompting in large language models. In *Advances in Neural Information Processing Systems (NeurIPS)*.

[Zhao et al., 2021] Zhao, T., Wallace, E., Feng, S., Klein, D., and Singh, S. (2021). Calibrate before use: Improving few-shot performance of language models. In *Proceedings of the 38th International Conference on Machine Learning (ICML)*, pages 12697–12706. PMLR.