PROTOTIPO PARA EL RECONOCIMIENTO DE FIRMAS MANUSCRITAS EN DOCUMENTOS DIGITALES APLICABLE A EPROCESS COMPANY S.A.S



UNIVERSIDAD DISTRITAL
FRANCISCO JOSÉ DE CALDAS

LINA MARIA MANCERA PEÑA

20211678036

UNIVERSIDAD DISTRITAL FRANCISCO JOSÉ DE CALDAS

FACULTAD TECNOLÓGICA

INGENIERÍA EN TELEMÁTICA

Bogotá D.C.

2025

PROTOTIPO PARA EL RECONOCIMIENTO DE FIRMAS MANUSCRITAS EN DOCUMENTOS DIGITALES APLICABLE A EPROCESS COMPANY S.A.S

LINA MARIA MANCERA PEÑA

20211678036

MODALIDAD

PASANTÍA

PRESENTADO PARA OPTAR AL TÍTULO DE: INGENIERO EN TELEMÁTICA.

TUTOR:

ISMAEL ANTONIO ARDILA SANCHEZ

UNIVERSIDAD DISTRITAL FRANCISCO JOSÉ DE CALDAS

FACULTAD TECNOLÓGICA

INGENIERÍA EN TELEMÁTICA

BOGOTÁ D.C.

2025

TABLA DE CONTENIDO

ÍNDICE DE TABLAS..... 6
ÍNDICE DE FIGURAS..... 6
RESUMEN..... 7
ABSTRACT..... 8
INTRODUCCIÓN..... 9
CAPÍTULO 1: FASE DE ORGANIZACIÓN, DEFINICIÓN Y PLANEACIÓN..... 10
1.1 TÍTULO..... 10
1.2 PLANTEAMIENTO DEL PROBLEMA..... 10
1.3 PROCESO ACTUAL DE VALIDACIÓN DE FIRMAS EN EPROCESS COMPANY S.A.S..... 15
1.4 OBJETIVOS..... 16
1.4.1 OBJETIVOS GENERAL..... 16
1.4.2 OBJETIVOS ESPECÍFICOS..... 17
1.5 JUSTIFICACIÓN..... 17
1.6 ALCANCE Y DELIMITACIÓN DEL PROYECTO..... 18
1.6.1 ALCANCE DEL PROYECTO..... 18
1.6.2 DELIMITACIONES DEL PROYECTO..... 19
1.6.2.1 DELIMITACIÓN TEMPORAL..... 19
1.6.2.2 DELIMITACIÓN ESPACIAL..... 19
1.6.2.3 DELIMITACIÓN TECNOLÓGICA..... 19
1.6.2.4 DELIMITACIÓN FUNCIONAL..... 20
1.7 MARCO TEÓRICO Y CONCEPTUAL..... 20
1.7.1 ESTADO DEL ARTE..... 20
1.7.2 MARCO METODOLÓGICO..... 23
1.8 SOLUCIÓN TECNOLÓGICA..... 25
1.9 PRESUPUESTO Y FINANCIACIÓN..... 26

1.9.1 FACTIBILIDAD ECONÓMICA....26
1.10 CRONOGRAMA....28
CAPÍTULO 2: ANÁLISIS DE TECNOLOGÍAS PARA EL RECONOCIMIENTO DE FIRMAS MANUSCRITAS....28
2.1 INTRODUCCIÓN....28
2.2. TECNOLOGÍA 1: YOLO (YOU ONLY LOOK ONCE)....29
2.2.1 DESCRIPCIÓN GENERAL....29
2.2.2 FUNCIONAMIENTO TÉCNICO....30
2.2.3 COSTOS....32
2.2.4 LICENCIAMIENTO....33
2.2.5 FACILIDAD DE INTEGRACIÓN....34
2.2.6 ACURA....35
2.2.7 TIEMPO DE DESARROLLO E IMPLEMENTACIÓN....36
2.2.8 REQUERIMIENTOS COMPUTACIONALES....37
2.2.9 COMPATIBILIDAD CON BALANCEO DE CARGA....38
2.2.10 SOPORTE Y DOCUMENTACIÓN....38
2.2.11 VENTAJAS....39
2.2.12 DESVENTAJAS....39
2.3 TECNOLOGÍA 2: MISTRAL....39
2.3.1 DESCRIPCIÓN GENERAL....39
2.3.2 FUNCIONAMIENTO TÉCNICO....40
2.3.3 COSTOS....42
2.3.4 LICENCIAMIENTO....43
2.3.5 FACILIDAD DE INTEGRACIÓN....44
2.3.6 ACURA....45
2.3.7 TIEMPO DE DESARROLLO E IMPLEMENTACIÓN....45
2.3.8 REQUERIMIENTOS COMPUTACIONALES....46
2.3.9 COMPATIBILIDAD CON BALANCEO DE CARGA....46
2.3.10 SOPORTE Y DOCUMENTACIÓN....47
2.3.11 VENTAJAS....48
2.3.12 DESVENTAJAS....48

2.4 TECNOLOGÍA 3: GTP-4.1 MINI....48
2.4.1 DESCRIPCIÓN GENERAL....48
2.4.2 FUNCIONAMIENTO TÉCNICO....49
2.4.3 COSTOS....51
2.4.4 LICENCIAMIENTO....53
2.4.5 FACILIDAD DE INTEGRACIÓN....53
2.4.6 ACURA....54
2.4.7 TIEMPO DE DESARROLLO E IMPLEMENTACIÓN....55
2.4.8 REQUERIMIENTOS COMPUTACIONALES....56
2.4.9 COMPATIBILIDAD CON BALANCEO DE CARGA....56
2.4.10 SOPORTE Y DOCUMENTACIÓN....56
2.4.11 VENTAJAS....57
2.4.12 DESVENTAJAS....57
2.5 COMPARACIÓN ENTRE TECNOLOGÍAS....57
2.6. SELECCIÓN DE LA TECNOLOGÍA BASE DEL PROTOTIPO....59
2.6.1 CRITERIOS DE EVALUACIÓN CONSIDERADOS....59
2.6.2 VENTAJAS DE GPT-4.1 MINI FRENTE A YOLO Y MISTRAL....59
2.6.3 FACTORES ECONÓMICOS Y DE LICENCIAMIENTO....62
2.6.4 JUSTIFICACIÓN TÉCNICA DE LA SELECCIÓN....63
2.6.5 CONCLUSIÓN DE LA SELECCIÓN....64
CAPÍTULO 3: DISEÑO PROTOTIPO....65
3.1 COMPONENTES PRINCIPALES DEL SISTEMA....65
3.2 FLUJO DEL PROTOTIPO....66
3.2.1 FLUJO DE AUTENTICACIÓN....67
3.2.2. FLUJO DE IDENTIFICACIÓN DE FIRMAS MANUSCRITAS....67
3.3 DIAGRAMAS DE SECUENCIA....69
3.4 GPT-4.1 MINI....72
3.5 MANEJO DE IMÁGENES CODIFICADAS EN BASE64....73
3.6 ENDPOINTS....74
3.7 PROCESAMIENTO DE DOCUMENTOS DIGITALES (PDF, JPEG, PNG)....74
3.8 Autenticación JWT....75
3.9 DISEÑO DE LA ARQUITECTURA DE BALANCEO DE CARGA....77

3.9.1 NGINX....77
3.9.2 FLASK Y GUNICORN....78
3.9.3 PLATAFORMA DE DESPLIEGUE RENDER....78
CAPÍTULO 4: IMPLEMENTACIÓN Y DESARROLLO DEL PROTOTIPO....79
4.1 CONFIGURACIÓN DEL ENTORNO DE DESARROLLO....79
4.2 TECNOLOGÍAS Y HERRAMIENTAS UTILIZADAS....80
4.3 IMPLEMENTACIÓN DEL SERVICIO DE AUTENTICACIÓN....81
4.4 PROCESAMIENTO DE DOCUMENTOS DIGITALES....81
4.5 INTEGRACIÓN CON OPENAI GPT-4.1....82
4.6 ENDPOINT DE VERIFICACIÓN DE FIRMAS....83
CAPÍTULO 5: PRUEBAS PROTOTIPO....84
5.1 METODOLOGÍA DE PRUEBAS....84
5.2 PRUEBAS FUNCIONALES....85
5.2.1 PRUEBA DE AUTENTICACIÓN....85
5.2.2 PRUEBA DE VERIFICACIÓN DE FIRMAS CON IMÁGENES....87
5.2.3 PRUEBA DE VERIFICACIÓN CON DOCUMENTOS PDF....94
5.3 PRUEBAS DE SEGURIDAD....98
5.3.1 ACCESO SIN TOKEN....98
5.3.2 TOKEN INVÁLIDO O MODIFICADO....99
5.3.3 TOKEN EXPIRADO....100
5.4 PRUEBAS DE BALANCEO DE CARGA....101
5.4.1 DISTRIBUCIÓN DE PETICIONES ENTRE INSTANCIAS....101
5.4.2 PRUEBA DE PERSISTENCIA DE SESIÓN (STATELESS JWT)....105
CONCLUSIONES....106
RECOMENDACIONES....107
BIBLIOGRAFÍA....108
ANEXOS....113

ÍNDICE DE TABLAS

Tabla 1 Factibilidad Económica Recursos Humanos..... 26
Tabla 2 Factibilidad Económica Recursos Técnicos..... 27
Tabla 3 Factibilidad Económica Costo Total.....27
Tabla 4 Comparación entre tecnologías.....58
Tabla 5 Resultado de distribución entre instancias..... 103

ÍNDICE DE FIGURAS

Figura 1. Proceso actual de validación de firmas en Eprocess Company S.A.S. (AS-IS)..... 16
Figura 2. Diagrama de cómo funciona Scrum.....24
Figura 3. Cronograma de actividades del proyecto..... 28
Figura 4. Diagrama de Gantt.....28
Figura 5. End-to-end YOLO Implementation.....31
Figura 6. Pixtral-12B evaluation protocol..... 45
Figura 7. Vision Transformer (Vit)..... 50
Figura 8. Visión..... 54
Figura 9. Comparación de desempeño entre GPT-4.1 Mini y Pixtral-12B en diferentes benchmarks (IFEval, MMLU, MMMU y MathVista)..... 62
Figura 10. Diagrama de secuencia del flujo de procesamiento de autenticación.....70
Figura 11. Diagrama de secuencia del flujo de verificación de firma.....72
Figura 12. Diagrama de secuencia del flujo del proceso de manejo de errores en comunicación..... 73
Figura 13. Login: credenciales correctas..... 87
Figura 14. Login: credenciales incorrectas..... 88
Figura 15. Verificar firma: JPG 1..... 89
Figura 16. Verificar firma: JPG 2..... 89
Figura 17. Verificar firma: JPG 3..... 90
Figura 18. Verificar firma: PNG 1..... 90
Figura 19. Verificar firma: PNG 2..... 91
Figura 20. Verificar firma: imagen 1 sin firma.....92
Figura 21. Verificar firma: imagen 2 sin firma.....93
Figura 22. Verificar firma: archivo con formato no válido: xlsx.....94
Figura 23. Verificar firma: archivo con formato no válido: docx..... 94
Figura 24. Verificar firma: archivo con formato no válido: docx..... 95
Figura 25. Verificar firma: PDF con firma en primera página..... 96
Figura 26. Verificar firma: PDF con firma en página intermedia..... 97
Figura 27. Verificar firma: PDF sin firma..... 98
Figura 28. Verificar firma: sin token.....99
Figura 29. Verificar firma: token invalido..... 100
Figura 30. Verificar firma: token expirado..... 101

Figura 31. Verificar firma: instancia A..... 104
Figura 32. Verificar firma: instancia B..... 105
Figura 33. Verificar firma: instancia C..... 105
Figura 34. Verificar firma: instancia C..... 106

RESUMEN

El presente proyecto desarrolla un prototipo para el reconocimiento de firmas manuscritas en documentos digitales, aplicable a Eprocess Company S.A.S., con el propósito de reducir la dependencia de la verificación manual. La problemática identificada se relaciona con alto volumen de documentos a validar, los errores frecuentes derivados del proceso manual, y también al uso de muchos recursos para llevar a cabo está tarea.

Se llevó a cabo un estudio comparativo de tres tecnologías que tienen el potencial de utilizarse para identificar firmas, analizando los aspectos técnicos, económicos e integradores. Se eligió la opción más apropiada a partir de esta investigación, sobre la cual se construyó el prototipo final.

El entregable corresponde a un servicio web, diseñado para detectar y validar firmas manuscritas en documentos digitales. La solución es respaldado por una infraestructura distribuida en la nube que asegura una distribución equitativa de la carga y así como un esquema de seguridad mediante JSON Web Tokens (JWT) que garantiza la confidencialidad y autenticidad de la información procesada.
En conclusión, el proyecto brinda una respuesta novedosa para automatizar la validación de firmas manuscritas en documentos digitales, disminuyendo así la dependencia de métodos manuales y contribuyendo a que Eprocess Company S.A.S. mejore su eficiencia y transformación digital.

ABSTRACT

This project develops a prototype for the recognition of handwritten signatures in digital documents, applicable to Eprocess Company S.A.S., with the purpose of reducing dependence on manual verification. The identified problem is related to the high volume of documents to be validated, the frequent errors derived from manual processes, and the excessive use of resources required to perform this task.

A comparative study of three technologies with potential application in signature recognition was carried out, analyzing technical, economic, and integration aspects. Based on this evaluation, the most appropriate option was selected, and the final prototype was built upon it.

The deliverable corresponds to a web service designed to detect and validate handwritten signatures in digital documents. The solution is supported by a distributed cloud infrastructure that ensures balanced load distribution, as well as a security scheme based on JSON Web Tokens (JWT) to guarantee the confidentiality and authenticity of the processed information.

In conclusion, this project provides an innovative response to automate the validation of handwritten signatures in digital documents, thus reducing reliance on manual methods and contributing to improved efficiency and digital transformation within Eprocess Company S.A.S.

INTRODUCCIÓN

Hoy en día, tanto entidades públicas como privadas siguen verificando de manera manual las firmas manuscritas en documentos digitalizados, sobre todo en procedimientos administrativos, legales y financieros. Este procedimiento, llevado a cabo por personal humano, tiene restricciones en cuanto a precisión y eficacia porque requiere que cada documento sea revisado visualmente de forma individual. Esto provoca retrasos, sobrecarga operativa y una alta posibilidad de equivocaciones. Según ProgressSoft Corporation[1], aspectos como la presión laboral, la interpretación subjetiva y la experiencia del personal aumentan los peligros, ocasionando incongruencias en el proceso de verificación de firmas, ante esta problemática se suma el desconocimiento sobre tecnologías emergentes como la inteligencia artificial y la visión computacional, así como la resistencia al cambio organizacional, lo que perpetúa la dependencia de métodos tradicionales y dificulta la adopción de soluciones innovadoras.

Ante lo anterior, el proyecto se plantea como una respuesta a la necesidad encontrada en Eprocess Company S.A.S., la cual se enfrenta con dificultades en la validación manual de firmas por el alto volumen de documentos y a los errores frecuentes del proceso. La solución consistió entonces en el desarrollo de un prototipo fundamentado en un análisis comparativo de tecnologías como YOLO, Mistral y GPT-4.1 Mini, del cual se eligió la alternativa más adecuada para la construcción del prototipo el cual es respaldado por una infraestructura distribuida en la nube y mecanismos de seguridad mediante JSON Web Tokens (JWT), lo que garantiza la confidencialidad e integridad de la información procesada. Esto contribuye a mejorar la gestión documental digital de la compañía y a reducir el grado de dependencia de las verificaciones manuales, aumentando así la eficiencia en los procesos internos de la organización.

---

[1] PROGRESSSOFT CORPORATION. Is manual signature verification obsolete?. [en línea]. Ammán: ProgressSoft Corporation, 2023. [citado el 12 de agosto de 2025]. Disponible en: https://www.progressoft.com/blogs/is-manual-signature-verification-obsolete

CAPÍTULO 1: FASE DE ORGANIZACIÓN, DEFINICIÓN Y PLANEACIÓN

1.1 TÍTULO

Prototipo para el Reconocimiento de Firmas Manuscritas en Documentos Digitales aplicable a Eprocess Company S.A.S

1.2 PLANTEAMIENTO DEL PROBLEMA

Actualmente, varias instituciones tanto gubernamentales como privadas siguen recurriendo al proceso de validar manualmente firmas manuscritas en documentos digitalizados, particularmente en procedimientos jurídicos, administrativos o financieros. Esta tarea, tradicionalmente realizada por personal humano, implica distintos obstáculos importantes en términos de eficacia y exactitud. Esta verificación manual exige que el equipo revise visualmente cada documento, uno por uno, lo que puede resultar ineficiente cuando existe un gran número de documentos que deben ser procesados en tiempos restringidos o periodos de crítica importancia. Además, varios elementos como la diferencia entre la experiencia de los empleados, la presión y el estrés en el trabajo influyen en la subjetividad del proceso, impactando la consistencia y coherencia en las decisiones y elevando los riesgos asociados, lo que puede provocar demoras, fallos y exceso de carga en operativa según ProgressSoft Corporation[2]. A pesar de las limitaciones de este proceso, este sigue siendo utilizado debido a la necesidad de verificar la autenticidad del documento mediante una firma física. Este proceso no tan solo retrasa el procesamiento de la documentación, sino que se convierte en un cuello de botella para procesos que requieren autenticación por urgencia por ejemplo autorización, contratos o procesos de tramitación jurídicos.

Ante esta realidad, crear un prototipo que permita el reconocimiento automático de las firmas manuscritas en documentos digitales significa una solución plausible aunque tecnológicamente desafiante. Una imagen similar a una firma no alcanza;

[2] Ibid

hay que enfrentar variación en el trazo, orientación, calidad del escaneo, e incluso presencia de ruido visual en las documentación. Además, la solución también tiene que poder trabajar con diferentes tipos de archivo, tales como imagen JPEG, PNG o documento PDF. Esto convierte la solución en un desafío multidisciplinario que demanda conocimiento en visión por computador, inteligencia artificial y procesamiento de imágenes.

Uno de los principales factores que dificulta la automatización del reconocimiento de firmas manuscritas en documentos digitales es la ausencia de herramientas tecnológicas especializadas en muchas instituciones. Esta carencia no solo se debe a la falta de inversión, sino también al desconocimiento técnico por parte del personal encargado de los procesos de validación documental sobre tecnologías emergentes, como inteligencia artificial o visión computacional, lo que impide la toma de decisiones informadas en cuanto a modernización tecnológica. Este retraso tecnológico impacta de manera directa la agilidad de la operación de instituciones que procesan volúmenes importantes de documentos. Al mismo tiempo, el problema sigue vigente por la resistencia al cambio organizacional, fenómeno bien documentado fenómeno en procesos de transformación digital. Las instituciones prefieren mantener hábitos tradicionales por miedo al error, gasto inicial o desconocimiento del funcionamiento de soluciones automatizadas. Dicha resistencia aumenta en ausencia de lineamientos institucionales o gubernativos que promuevan la adopción tecnológica para la gestión documental.

Según un informe publicado por ProgressSoft Corporation[3], muchas instituciones bancarias siguen recurriendo al proceso manual para autenticar firmas, lo que causa retrasos, errores y también sobrecargos operativos. Esto limita el poder de respuesta de las instituciones en relación con volúmenes elevados de documentación e impacta la eficacia de los servicios que estos ofertan. Además, la validación manual está sujeta a la interpretación subjetiva del personal, lo que implica que dos verificadores podrían llegar a conclusiones distintas sobre una misma firma. La persistencia de procesos manuales para validar firmas manuscritas está impulsada por el hecho que también existe escasez de integración entre las

[3] Ibid

distintas áreas de una institución, por ejemplo, el área tecnológico-administrativa. Aunque en muchos casos existen departamentos de TI, estos operan de forma aislada y no diseñan soluciones alineadas con las necesidades del flujo documental. Esta falta de integración impide el desarrollo de plataformas automatizadas que detecten o controlen firmas de manera efectiva. A esta falta hay que añadir una escasa cultura institucional digital en la que predomina el cumplimiento operativo diario por encima de la mejora continua sobre la base de la innovación. Así pues, mantener esta perspectiva convencional obstaculiza la inversión en investigación aplicada, así como en la utilización de herramientas de inteligencia artificial.

La automatización del reconocimiento de firmas manuscritas no solo representa un progreso tecnológico, sino que también satisface una necesidad de la realidad empresarial y gubernamental: la rapidez y la seguridad en los procedimientos de verificación documental. Un prototipo que permita de manera automatizada la identificación de firmas ahorraría tiempo valioso, disminuiría la carga operativa de los funcionarios y contribuiría al proceso de la transformación digitales de las organizaciones, un objetivo importante en la actualidad. La solución presentada cobra gran relevancia en sectores donde el estricto cumplimiento normativo y la integridad de los documentos son imprescindibles, como banca, salud o justicia. En el estudio presentado por Cordero-Pérez, M. I., y Salas-Duarte, P. E[4] , evidencian que la aplicación de técnicas de Inteligencia Artificial (IA) ha generado mejoras sustanciales en eficiencia y seguridad mediante la implementación de sistemas automatizados en procesos complejos. En este sentido, el reconocimiento automatizado de firmas manuscritas puede beneficiarse de algoritmos avanzados de procesamiento y aprendizaje automático para replicar dichos beneficios en la autenticación documental.

El área de reconocimiento de firmas manuscritas también ha evolucionado significativamente en la última década debido al gran avance que ha tenido la inteligencia artificial. Investigaciones tales como las de Santos et al. (2021) han

[4] CORDERO-PÉREZ, M. I.; SALAS-DUARTE, P. E. Inteligencia Artificial en la Gestión de Redes Telemáticas: Avances, Tendencias y Aplicaciones Actuales. En: Revista Vínculos. [en línea]. Bogotá. Vol. 21, No. 1 (2024). [citado el 12 de agosto de 2025]. Disponible en: https://revistas.udistrital.edu.co/index.php/vinculos/article/view/22565

resaltado que el empleo de redes convolucionales (CNN) hace que sea posible detectar firmas con una elevada tasa de precisión incluso en resolución baja o en papeles deteriorados. Estos hechos justifican la viabilidad del desarrollo de soluciones automatizadas que son aplicables en escenarios del mundo real, siempre cuando se utilicen modelos entrenados con bases de datos representativas. Además, la utilización de bibliotecas y frameworks tales como TensorFlow, Keras y OpenCV permiten construir prototipos funcionales para tareas de reconocimiento visual. Ahora bien, con este contexto, el presente proyecto cuenta con el apoyo de los avances teóricos realizados hasta el momento, permitiendo al igual integrarse con las mejores prácticas tecnológicas disponibles para resolver un problema concreto y actual.

La protección de la información también es un elemento crucial, por lo que es esencial implementar mecanismos que aseguren la privacidad, integridad y accesibilidad de los datos compartidos entre usuarios y sistema. Como señalan Bernardis, E., Bernardis, H., Berón, M., & Montejano, G. A. [5], la seguridad informática debe anticiparse a amenazas potenciales y minimizar las vulnerabilidades de los sistemas. Por ello , se hace necesaria la aplicación de mecanismos de ofuscación y cifrado sobre los identificadores y otras secciones vitales del programa , ya que dificultan la manipulación y el acceso no autorizado a la información . Por este motivo , el prototipo desarrollado estará protegido mediante mecanismos de autenticación y autorización mediante JSON Web Token (JWT), permitiendo un control seguro y eficaz del acceso al sistema . Según BBVA Labs & BBVA[6], los JWT son compactos y autónomos , lo que los hace fáciles de utilizar dentro de sistemas distribuidos , así como en arquitecturas multiservicio sin almacenar sesiones en el servidor. Esto permite un mayor rendimiento y experiencia del usuario además de mejorar la seguridad general de la solución .

---

[5] BERNARDIS, E.; BERNARDIS, H.; BERÓN, M.; MONTEJANO, G. A. Seguridad en servicios web. [en línea]. La Plata: Universidad Nacional de La Plata, SEDICI, 2017. [citado el 12 de agosto de 2025]. Disponible en: http://sedici.unlp.edu.ar/handle/10915/62726

[6] BBVA LABS; BBVA. JSON Web tokens (JWT): claves para usarlos de manera segura. En: BBVA NOTICIAS. [en línea]. Madrid, 22 de marzo de 2023. [citado el 12 de agosto de 2025]. Disponible en: https://www.bbva.com/es/innovacion/json-web-tokens-jwt-claves-para-usarlos-de-manera-segura/

El objetivo general del actual proyecto es crear entonces un prototipo para el reconocimiento automático de firmas manuscritas en documentos digitales, que se integre en una arquitectura con balanceo de carga para su posterior despliegue. Este objetivo responde directamente a la necesidad que se logro detectar en varios sectores de mejorar los procesos de validación documental y reducir la dependencia de la verificación manual. Como complemento al propósito general, también se han definido cuatro objetivos específicos: (1) analizar tecnologías que permitan el reconocimiento de firmas manuscritas en documentos digitales, considerando criterios técnicos, funcionales y operativos con el fin de seleccionar la más adecuada para el desarrollo del prototipo, (2) desarrollar un prototipo que permita el reconocimiento de firmas manuscritas en documentos digitales basado en la tecnología seleccionada durante la etapa de análisis, (3) implementar el prototipo mediante una arquitectura que contemple mecanismos de balanceo de carga y (4) implementar un mecanismo de autenticación y autorización para el prototipo basado en tokens. Logrando así dividir el proyecto en etapas claras y cuantificables que permitan el desarrollo gradual de la solución propuesta.

La pregunta que guía esta investigación es: ¿Qué herramientas y tecnologías se pueden utilizar para desarrollar un prototipo que permita detectar firmas manuscritas en documentos digitales? Esta pregunta busca saber las diversas tecnologías disponibles para la detección de firmas y establece claramente el foco del proyecto, centrándose en la identificación de firmas manuscritas mediante soluciones tecnológicas modernas. Además, invita a explorar distintos caminos. Responder a esta cuestión no solo proporcionará una base sólida para el desarrollo del prototipo, sino que también contribuirá al avance en la automatización de este proceso de validación que actualmente dependen de la validación manual de documentos.

Adicional a todo lo anterior y como parte del proceso de levantamiento de información para el desarrollo del prototipo y verificar que la necesidad es real, se llevó a cabo una entrevista con un representante de la empresa Eprocess S.A.S., quien manifestó el interés de la organización en implementar una solución automatizada para el reconocimiento de firmas manuscritas en documentos digitales. Durante la conversación, se identificó que esta necesidad surge a partir de

enfrentarse a dificultades en la validación manual de firmas debido al alto volumen de documentos y a los errores frecuentes derivados del proceso manual, también al uso de muchos recursos para llevar a cabo está tarea. La entrevista permitió validar la pertinencia del proyecto y su aplicabilidad en un contexto empresarial real, lo que refuerza el valor práctico del prototipo propuesto. El acta de la entrevista se incluye como anexo 1 para evidenciar la interacción con el actor involucrado.

1.3 PROCESO ACTUAL DE VALIDACIÓN DE FIRMAS EN EPROCESS COMPANY S.A.S.

Según lo expresado por María Isabel Cuellar Izquierdo, Coordinadora de Proyectos de Software en Eprocess Company S.A.S., la compañía está llevando a cabo en la actualidad un proceso totalmente manual para comprobar la presencia de firmas manuscritas en documentos digitales. Este procedimiento exige que cada documento sea revisado visualmente por el personal administrativo, uno por uno, para verificar la existencia de la firma; este método es propenso a errores humanos y consume mucho tiempo. La compañía procesa alrededor de 5000 documentos por mes, cifra que aumenta durante algunos periodos internos de gestión. Los documentos que requieren validación incluyen contratos, autorizaciones de pago, certificaciones laborales y solicitudes internas. Durante la entrevista, la coordinadora expresó que "el proceso es lento, especialmente cuando hay grandes volúmenes de documentos. Además, la posibilidad de errores en la revisión puede generar problemas retrasos en trámites importantes"[7].

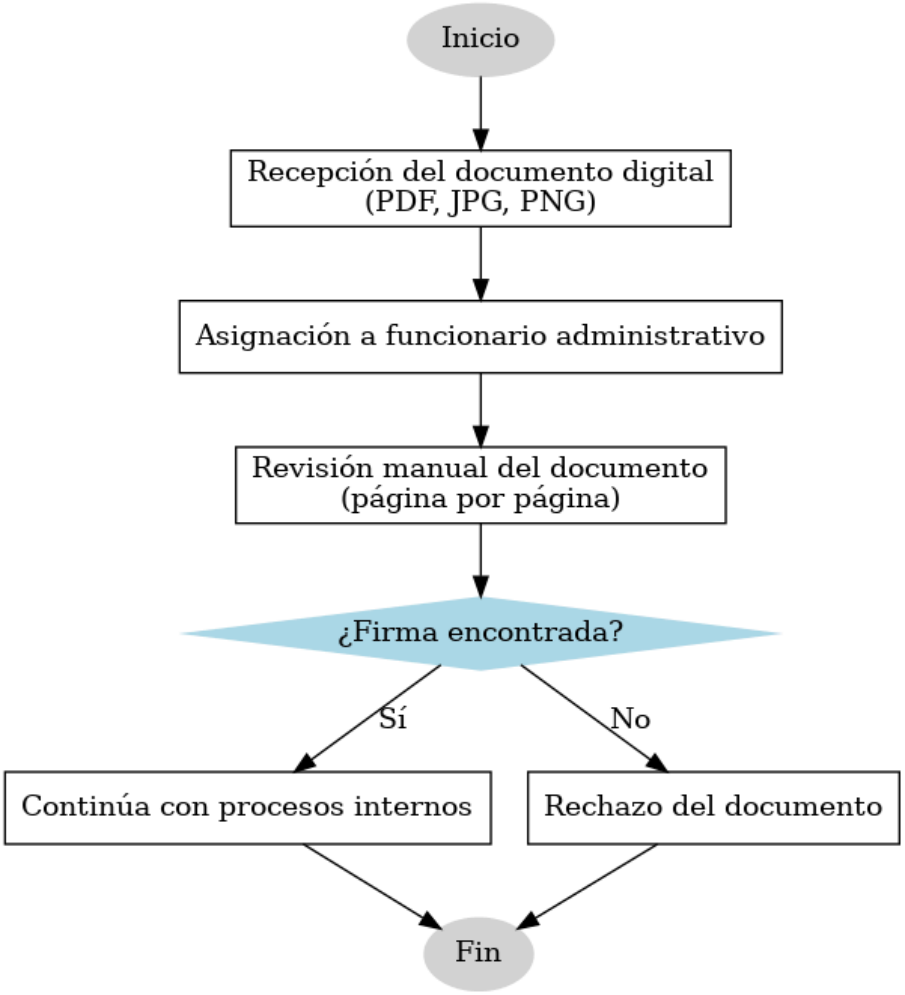
Los obstáculos más importantes que se han detectado son la ineficiencia del proceso con volúmenes documentales grandes, la tendencia a cometer errores por el análisis manual y visual, los costos operativos altos debido a que necesita personal exclusivo para esta función y el hecho de que el proceso actual no puede escalarse. La compañía expresó su interés en automatizar este procedimiento, con la observación de que "contar con un prototipo que verifique automáticamente si un

[7] CUÉLLAR, MARÍA ISABEL. Prototipo para el Reconocimiento de Firmas Manuscritas en Documentos Digitales. [Entrevista]. En: MANCERA, LINA. Proyecto de Grado: Prototipo para el Reconocimiento de Firmas Manuscritas en Documentos Digitales aplicable a Eprocess Company S.A.S. Bogota, 2025, Anexo 1.

documento contiene o no una firma manuscrita antes de pasar al siguiente paso del flujo sería muy útil"[8].

En la Figura 1 se muestra el diagrama de procesos actual (AS-IS), construido a partir de la información recolectada en la entrevista realizada

Figura 1. Proceso actual de validación de firmas en Eprocess Company S.A.S. (AS-IS).



Fuente: Elaboración propia.

[8] Ibid

1.4 OBJETIVOS

1.4.1 OBJETIVOS GENERAL

Desarrollar un prototipo para el reconocimiento automático de firmas manuscritas en documentos digitales, integrándose en una arquitectura que implemente balanceo de carga para su posterior despliegue.

1.4.2 OBJETIVOS ESPECÍFICOS

1. Analizar tecnologías que permitan el reconocimiento de firmas manuscritas en documentos digitales, considerando criterios técnicos, funcionales y operativos con el fin de seleccionar la más adecuada para el desarrollo del prototipo.
2. Desarrollar un prototipo que permita el reconocimiento de firmas manuscritas en documentos digitales basado en la tecnología seleccionada durante la etapa de análisis.
3. Implementar el prototipo mediante una arquitectura que contemple mecanismos de balanceo de carga.
4. Implementar un mecanismo de autenticación y autorización para el prototipo basado en tokens.

1.5 JUSTIFICACIÓN

El reconocimiento automático de firmas manuscritas en documentos digitales se justifica en la necesidad de optimizar los procesos administrativos, legales y financieros en organizaciones que aún dependen de la validación manual de documentos. Este proceso, aparte de requerir recursos humanos y tiempo, aumenta la posibilidad de equivocaciones y aplaza las decisiones estratégicas. Según una

investigación de Deloitte[9], el 67% de las organizaciones reconoce que la gestión manual de documentos genera cuellos de botella en sus procesos, lo cual resulta en demoras para tomar decisiones y en una disminución de la calidad del servicio.

Al implementar un prototipo de reconocimiento automático de firmas, se colabora con la digitalización empresarial, lo que propicia la eficacia, la seguridad y la disminución de costos. Esto al apoyarse con los avances recientes de la inteligencia artificial y la visión computacional, los cuales han alcanzado un nivel de madurez que permite abordar el reconocimiento de patrones visuales complejos con altos grados de precisión. según lo afirman Hafemann, Sabourin y Oliveira [10] las redes neuronales convolucionales (CNN) tienen la capacidad de identificar firmas manuscritas, con niveles de precisión que alcanzan o sobrepasan la habilidad humana en determinadas circunstancias controladas; esta evolución tecnológica combinada con la disponibilidad de modelos preentrenados y APIs especializadas como GPT-4.1 Mini, Mistral Vision y YOLO, facilita la implementación de soluciones automatizadas sin requerir el desarrollo desde cero de algoritmos complejos.

Para la empresa beneficiaria directa del proyecto; Eprocess Company S.A.S., la automatización le representaría una disminución significativa en el tiempo dedicado a la validación manual de firmas, liberando recursos humanos para actividades de mayor valor agregado, de acuerdo con investigaciones recientes demuestran que la automatización en la gestión documental puede incrementar la productividad hasta en un 50 % y reducir los costos en un 30 %[11] . De igual manera, el prototipo se alinea con los lineamientos de seguridad informática, integrando mecanismos de autenticación robusta como JSON Web Tokens (JWT), lo que asegura la confidencialidad e integridad de la información procesada[12]

---

[9] DELOITTE. Digital Document Management Trends 2022. [en línea]. [Ciudad de publicación no especificada]: Deloitte, 2022. [citado el 13 de agosto de 2025]. Disponible en: https://www2.deloitte.com/

[10] HAFEMANN, L. G.; SABOURIN, R.; OLIVEIRA, L. S. Learning Features for Offline Handwritten Signature Verification using Deep Convolutional Neural Networks. [Preprint]. En: arXiv preprint. [en línea]. [Localidad no especificada]: arXiv, 2017. [citado el 13 de agosto de 2025]. Disponible en: https://arxiv.org/abs/1705.05787

[11] DELOITTE ,op. cit.

[12] BBVA LABS; BBVA, ,op. cit.

En conclusión, un prototipo de reconocimiento automático de firmas manuscritas se presenta como una alternativa eficiente para ir más allá de las restricciones que supone la validación manual de firmas en documentos. Esto posibilita mejorar los recursos, disminuir los plazos operativos y contribuir a la transformación digital de Eprocess Company S.A.S., lo que repercute positivamente en la productividad y en el nivel del servicio.

1.6 ALCANCE Y DELIMITACIÓN DEL PROYECTO

1.6.1 ALCANCE DEL PROYECTO

El presente proyecto tiene como alcance el desarrollo de un prototipo funcional para el reconocimiento automático de firmas manuscritas en documentos digitales, específicamente aplicable a Eprocess Company S.A.S. El alcance comprende los siguientes entregables:

1. Análisis comparativo de tecnologías: Investigación y evaluación de tres tecnologías (YOLO, Mistral Vision y GPT-4.1 Mini) para el reconocimiento de firmas manuscritas, considerando criterios técnicos, funcionales y operativos.
2. Desarrollo del prototipo: Creación de un servicio web que detecte firmas manuscritas en documentos digitales, basado en la tecnología seleccionada durante la etapa de análisis con implementación de mecanismos de seguridad basados en JSON Web Tokens (JWT)
3. Implementación en arquitectura distribuida: Despliegue del prototipo mediante una arquitectura que contemple mecanismos de balanceo de carga,
4. Pruebas funcionales: Evaluación del rendimiento y funcionalidad del prototipo desarrollado.
5. Presentación final: Presentación del prototipo con información general sobre el proyecto con sus respectivos resultados y conclusiones.

1.6.2 DELIMITACIONES DEL PROYECTO

El proyecto presenta las siguientes delimitaciones específicas.

1.6.2.1 DELIMITACIÓN TEMPORAL

El desarrollo del proyecto se ejecuta durante el periodo académico 2025, con una duración estimada según el cronograma establecido en la sección 1.9.

1.6.2.2 DELIMITACIÓN ESPACIAL

La aplicación del prototipo está dirigida específicamente a Eprocess Company S.A.S., aunque la solución desarrollada puede ser adaptable a otras organizaciones con necesidades similares de validación documental.

1.6.2.3 DELIMITACIÓN TECNOLÓGICA

1. El prototipo se enfoca exclusivamente en el reconocimiento de firmas manuscritas, no contemplando otros elementos como sellos, rúbricas o texto impreso de manera diferenciada.
2. Los formatos de entrada soportados se limitan a documentos digitales como archivos PDF e imágenes escaneadas (JPEG, PNG).
3. La solución se basa en la tecnología seleccionada y su integración mediante API REST.

1.6.2.4 DELIMITACIÓN FUNCIONAL

1. El prototipo se concentra en la detección y validación de la presencia de firmas manuscritas, proporcionando información sobre la existencia o ausencia de firma en el documento.

2. No incluye funcionalidades de verificación de autenticidad o comparación con bases de datos de firmas de referencia.

1.7 MARCO TEÓRICO Y CONCEPTUAL

1.7.1 ESTADO DEL ARTE

La firma manuscrita es el medio más común para verificar la identidad de una persona. Se utiliza ampliamente en la sociedad para indicar la autoría de un documento o expresar conformidad en distintos tipos de transacciones. Es un rasgo personal y único, considerado una característica biométrica confiable, ya que sintetiza la personalidad del firmante y actúa como un sello representativo. Debido a su importancia legal, es también uno de los métodos biométricos más susceptibles a intentos de falsificación[13]. “las firmas manuscritas siguen siendo reconocidas como prueba de consentimiento y autenticidad en una variedad de jurisdicciones, incluso cuando existen opciones electrónicas disponibles”[14]

Tradicionalmente, este proceso ha sido manual, lo que implica largos tiempos de revisión, posibilidad de errores humanos y costos operativos elevados. Por ello, el reconocimiento automático de firmas manuscritas se presenta como una solución estratégica para agilizar y asegurar la eficiencia operativa. Según Hafemann[15], los sistemas de verificación automática de firmas se convierten en una alternativa viable para operaciones a gran escala, brindando tanto seguridad como eficiencia a instituciones que manejan documentos firmados en grandes volúmenes.

Actualmente, existen tecnologías altamente efectivas para el reconocimiento de firmas, El presente proyecto se centra en el reconocimiento fuera de línea, que

---

[13] MANUEL, F. R. E. La firma como un método biométrico de identificación. [en línea]. Ciudad de México: Instituto Politécnico Nacional (IPN), 2008. [citado el 14 de agosto de 2025]. Disponible en: https://www.repositoriodigital.ipn.mx/handle/123456789/8502

[14] AMERICAN BAR ASSOCIATION. Understanding Electronic Signatures and Their Legal Validity. [en línea]. Chicago (Sede): American Bar Association, 2023. [citado el 14 de agosto de 2025]. Disponible en: https://www.americanbar.org/groups/business_law/publications/blt/2023/01/electronic-signatures-legal/

[15] HAFEMANN, L. G.; SABOURIN, R.; OLIVEIRA, L. S, op. cit.

consiste en el análisis de firmas escaneadas o presentes en documentos digitales. Las tecnologías más destacadas incluyen:

OpenCV (Open Source Computer Vision Library) es una biblioteca de código abierto ampliamente utilizada en el procesamiento de imágenes y visión por computadora. Su potencial para la detección y reconocimiento de firmas radica en su capacidad de realizar operaciones como filtrado de ruido, segmentación de contornos, detección de bordes y análisis morfológico de imágenes, todos ellos pasos fundamentales para identificar patrones únicos en una firma manuscrita. Gracias a su eficiencia y versatilidad, OpenCV ha sido adoptado como una herramienta clave en proyectos de biometría y autenticación visual, incluyendo el reconocimiento de escritura y firmas, según Bradski[16].

Mistral Vision es un modelo de inteligencia artificial especializado en visión computacional y análisis multimodal, disponible en plataformas como Hugging Face. Está diseñado para tareas avanzadas como reconocimiento óptico de caracteres (OCR), detección de objetos y análisis contextual de imágenes. En aplicaciones de reconocimiento de firmas, Mistral Vision puede identificar firmas en documentos escaneados, diferenciarlas de otros elementos visuales, y compararlas con bases de datos para validar su autenticidad. Su diseño permite una integración eficaz en sistemas de flujo documental automatizado.

Phi-4-Multimodal-Instruct es un modelo de lenguaje e imagen que sigue un enfoque instruccional, lo cual lo hace especialmente útil en tareas donde se requiere seguir órdenes o analizar documentos bajo un criterio específico. Gracias a su capacidad de procesamiento multimodal, puede interpretar tanto el contenido visual de una firma como las instrucciones asociadas a su validación. Este modelo ha sido utilizado para tareas como revisión de documentos, extracción de información y verificación visual, siendo adaptable a contextos como la firma de contratos, formularios institucionales y documentos legales.

---

[16] BRADSKI, G. The OpenCV Library. En: Dr. Dobb's Journal of Software Tools. [Localidad no especificada]: Dr. Dobb's Journal, 2000. [citado el 16 de agosto de 2025].

Las Redes Neuronales Convolucionales (CNN, por sus siglas en inglés) son un tipo de arquitectura de aprendizaje profundo ampliamente utilizada en tareas de visión por computadora, debido a su capacidad para aprender automáticamente características espaciales jerárquicas de las imágenes. En el contexto del reconocimiento de firmas manuscritas, las CNN han demostrado ser altamente efectivas para extraer patrones únicos y discriminativos de las firmas, incluso cuando presentan variaciones en forma, tamaño o inclinación. Estas redes aprenden a identificar características como la presión del trazo, la curvatura, y las conexiones entre caracteres, lo cual permite una clasificación precisa entre firmas genuinas y falsificadas. Gracias a su precisión y adaptabilidad, las CNN se han convertido en una herramienta clave para sistemas biométricos de autenticación basados en firmas digitales[17].

JWT es un estándar abierto (RFC 7519) que facilita la transmisión de datos seguros y verificables entre cliente y servidor a través de tokens firmados digitalmente, lo que resulta óptimo para salvaguardar servicios web y APIs. Cuando introduce JWT en tu prototipo: Se garantiza que únicamente usuarios autenticados tengan acceso a los recursos resguardados. Se suprime la obligación de conservar el estado de sesión en el servidor, lo que favorece la escalabilidad y el equilibrio de carga. Es posible verificar la autenticidad e integridad del token en cada petición, evitando así accesos no permitidos. Se potencia la protección global del sistema mediante el uso de firmas digitales y, opcionalmente, el encriptado de datos delicados. Este sistema se utiliza extensamente para salvaguardar servicios web actuales y APIs REST, y se incorpora con facilidad a arquitecturas que aplican balance de carga[18]. Para incrementar la seguridad en los servicios web, es fundamental aplicar técnicas de ofuscación y encriptación sobre los identificadores y otros componentes críticos del software. De acuerdo con Bernardis[19], al reducir el nivel de comprensibilidad mediante estos métodos, se dificulta la manipulación y el robo de información, fortaleciendo así la protección de los sistemas frente a posibles ataques.

---

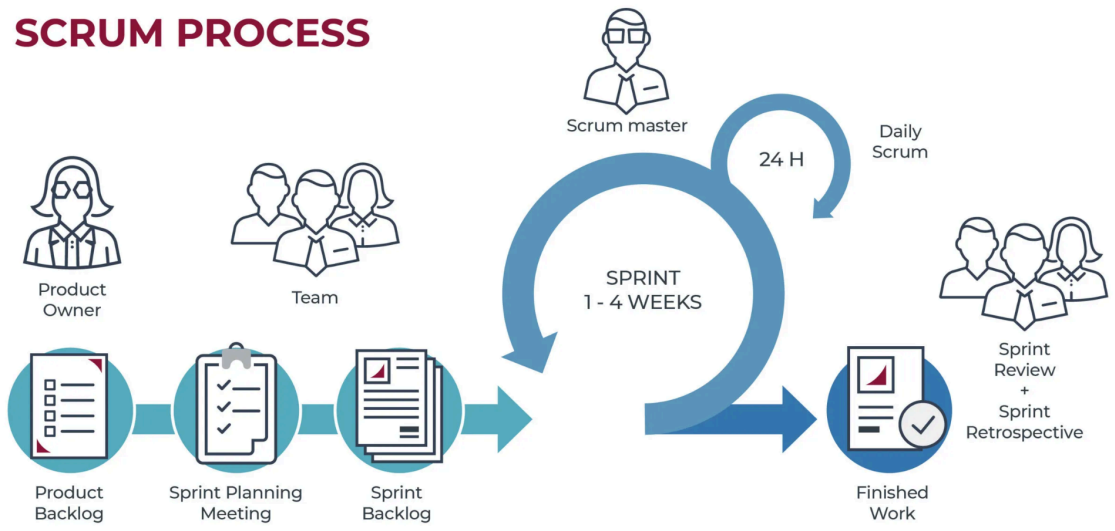
[17] HAFEMANN, L. G.; SABOURIN, R.; OLIVEIRA, L. S, op. cit.
[18] BBVA LABS; BBVA, op. cit.
[19] BERNARDIS, E.; BERNARDIS, H.; BERÓN, M.; MONTEJANO, G. A, op. cit.

1.7.2 MARCO METODOLÓGICO

Según Guevara Bolaños, J. C., y Pinzón, S. (2013), la gestión eficiente y la aplicación adecuada de metodologías en el desarrollo de software son fundamentales para mejorar la calidad del producto y optimizar los procesos organizacionales. Por lo cual para el desarrollo del prototipo para el reconocimiento de firmas manuscritas en documentos digitales, se implementará la metodología Agil SCRUM. Esto facilita un proceso iterativo, incremental y colaborativo que puede adaptarse al cambio y ofrecer valor continuamente a lo largo del ciclo de vida del proyecto.

Scrum es una metodología ágil para la gestión y desarrollo de proyectos que facilita la organización del trabajo en intervalos cortos denominados sprints, los cuales normalmente duran de una a cuatro semanas. Su meta fundamental es suministrar de manera progresiva productos útiles que sean evaluados y perfeccionados de forma continua. Scrum se fundamenta en la transparencia, la revisión y la flexibilidad para asegurar que los equipos puedan adaptarse rápidamente a los cambios y mejorar la calidad del producto a lo largo del proyecto (Schwaber, K., & Sutherland, J. 2020).

Figura 2. Diagrama de cómo funciona Scrum.



Fuente: Atenos (s.f.).

Scrum trabaja con roles definidos como lo son el Product Owner que da la prioridad a las tareas, el Scrum Master que simplifica el proceso y el equipo de desarrollo que es responsable de poner en marcha las funcionalidades. El proceso se estructura en reuniones periodicas donde se podria llevar acabo, la programación del sprint, encuentros diarios para la sincronización que son llamados Daily Scrum, evaluaciones para exhibir progresos y retrospectivas para optimizar el procedimiento. Los instrumentos que respaldan el desarrollo incluyen el Backlog Product, que alberga las necesidades prioritarias, el Backlog Sprint que especifica las actividades a finalizar en cada sprint y el Incremento que representa el resultado funcional proporcionado al concluir cada ciclo. Estos componentes facilitan la comunicación continua y un enfoque iterativo que incrementa la productividad y la capacidad de adaptación del equipo.

En conclusión se elige Scrum para el desarrollo del proyecto ya que este es apropiado por sus características, que promueve una adaptación constante a los descubrimientos y modificaciones que se presenten durante la investigación y selección de la tecnología. Gracias a su naturaleza iterativa e incremental permite la incorporación gradual de las distintas funcionalidades, lo que ayuda a que se tenga una validación constante y temprana de los diferentes resultados, logrando asi reducir los riesgos y potenciando la calidad del proyecto. Adicional que este permite tener un dialogo continuo atraves de las reuniones lo que potencializa la cooperación entre los integrantes del equipo, asegurando entonces una coordinación y solución rápida de problemas los cuales son elementos esenciales para desarrollar eficaz y exitosamente el proyecto.

1.8 SOLUCIÓN TECNOLÓGICA

La utilización de procesos manuales para confirmar la presencia de firmas manuscritas en documentos escaneados o creados digitalmente ocasiona demoras, incrementa los gastos de operación y es propenso a equivocaciones humanas y

limita la capacidad de respuesta de las organizaciones que gestionan altos volúmenes de documentación.

En respuesta a esta dificultad, la solución tecnológica propuesta es crear un prototipo para el reconocimiento de firmas manuscritas en documentos digitales. La solución definitiva se concretará en un servicio web que admitirá como entrada un documento digital (como un archivo en PDF o una imagen escaneada) y proporcionará como resultado datos acerca de la existencia o falta de una firma manuscrita, este servicio estará asegurado mediante un esquema de autenticación y autorización basado en JSON Web Tokens (JWT), con el propósito de garantizar el control de acceso al mismo, buscando también la trazabilidad de las peticiones y la protección de la información procesada. En última instancia, se desplegará el prototipo en una arquitectura con balance de carga, lo que facilitará la distribución de las peticiones entre diversas etapas del servicio, asegurando su disponibilidad y escalabilidad.

En conclusión, esta solución tecnológica contempla los siguientes entregables fundamentales:

1. Investigación de 3 tecnologías para reconocimiento de firmas manuscritas.
2. Prototipo que detecta firmas manuscritas en documentos digitales.
3. Servicio web seguro basado en JWT.
4. Despliegue en infraestructura distribuida con balanceo de carga.
5. Pruebas funcionales
6. Presentación final del prototipo

1.9 PRESUPUESTO Y FINANCIACIÓN

1.9.1 FACTIBILIDAD ECONÓMICA

El presente proyecto de grado para la carrera de Ingeniería en Telemática, titulado “Prototipo para el Reconocimiento de Firmas Manuscritas en Documentos Digitales

aplicable a Eprocess Company S.A.S", es completamente viable desde el punto de vista económico.

La factibilidad económica del proyecto es alta, ya que los recursos financieros necesarios para su desarrollo son mínimos y accesibles. Los principales requerimientos son:

- Un equipo de cómputo con características adecuadas para el procesamiento de datos.
- Asesorías académicas por parte de los tutores del proyecto.
- Acceso permanente a Internet para investigación, pruebas y colaboración.

En las siguientes tablas se detallan los elementos necesarios, agrupados en tres categorías: hardware, software y recursos humanos, junto con sus respectivos costos estimados. Esta evaluación permite concluir que el proyecto es económicamente factible y puede ejecutarse dentro del entorno académico sin requerir grandes inversiones

Tabla 1 Factibilidad Económica Recursos Humanos

Tipo	Descripción	Valor-Hora	Cantidad	Total
Tutor 1	Asesorías para la realización del proyecto, referente a la metodología.	$60.000	80	$4.800.000
Desarrollador	Desarrollo, pruebas y documentación	$40.000	300	$12.000.000
Asesor Eprocess	Dos (2) asesores de Eprocess para la realización del proyecto, referente a la metodología.	$80.000	100	$8.000.000
Total Recursos Humanos				$24.800.000

Nota. La tabla muestra el presupuesto en pesos colombianos que implicaría la ejecución de este proyecto, considerando la asignación de recursos humanos necesarios. Fuente: Elaboración propia.

Tabla 2 Factibilidad Económica Recursos Técnicos

Recurso	Descripción	Valor Unitario	Cantidad	Total
Computador portátil	Equipos portátiles para el desarrollo y las pruebas del sistema.	$ 4.700.000	1	$ 4.700.000
Conectividad	Servicio de internet de alta velocidad para el equipo	$100.000	2(meses)	$100.000
Equipos de comunicación	Plan teléfono	$80.000	2(meses)	$140.000
Gestión proyecto	Jira	$31.453	2(meses)	$64.360
Total Recursos Técnicos				$5.004.360

Nota. Esta tabla proporciona una evaluación detallada de los aspectos financieros relacionados con la viabilidad de los recursos técnicos necesarios para el proyecto. Elaboración propia. Fuente: Elaboración propia.

Tabla 3 Factibilidad Económica Costo Total

Recurso	Valor
Total Recursos Humanos	$24.800.000
Total Recursos Técnicos	$5.004.360

Total Otros recursos	$0
Costos imprevistos (10%)	$3.013.436
TOTAL COSTO	$32.817.796

Nota. En esta tabla se incluye el costo total de la asignación de recursos humanos, así como los costos asociados con los equipos. **Fuente**: Elaboración propia.

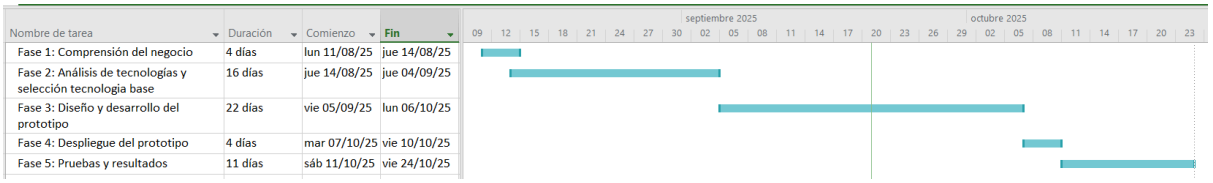
1.10 CRONOGRAMA

Figura 3. Cronograma de actividades del proyecto.

Modo de	Nombre de tarea	Duración	Comienzo	Fin
	Fase 1: Comprensión del negocio	4 días	lun 11/08/25	jue 14/08/25
	Fase 2: Análisis de tecnologías y selección tecnologia base	16 días	jue 14/08/25	jue 04/09/25
	Fase 3: Diseño y desarrollo del prototipo	22 días	vie 05/09/25	lun 06/10/25
	Fase 4: Despliegue del prototipo	4 días	mar 07/10/25	vie 10/10/25
	Fase 5: Pruebas y resultados	11 días	sáb 11/10/25	vie 24/10/25

Fuente: Elaboración propia.

Figura 4. Diagrama de Gantt



Nota: El eje horizontal exhibe las fechas correspondientes a las distintas etapas del proyecto, mientras que en el eje vertical se indica la duración estimada en semanas para cada fase. Fuente: Elaboración propia.

CAPÍTULO 2: ANÁLISIS DE TECNOLOGÍAS PARA EL RECONOCIMIENTO DE FIRMAS MANUSCRITAS

2.1 INTRODUCCIÓN

El reconocimiento de firmas manuscritas en documentos digitales constituye un problema técnico dentro de la visión computacional e inteligencia artificial. No solamente se refiere al localizar la firma en un documento, sino también a distinguir la firma de otros símbolos gráficos como sellos, rúbricas o texto impreso.

En esta etapa del proyecto se realizará un estudio comparativo de diversas tecnologías posibles que puedan implementarse para el desarrollo del prototipo de reconocimiento automático de firmas manuscritas. Lo que se pretende con este estudio comparativo es encontrar la tecnología con la mejor relación entre la precisión, facilidad de integración, y costo, así como tener en cuenta su capacidad de adaptación a una arquitectura distribuida con balanceo de carga, entre otros.

Para este propósito, se han seleccionado tres tecnologías que se diferencian entre sí, en su técnica, su grado de procesado y su nivel interpretativo visual, con el objetivo de estudiar soluciones que abarquen diferentes contextos en el reconocimiento de la firma manuscrita

1. YOLO (You Only Look Once): un algoritmo de detección de objetos de tiempo real basado en redes neuronales convolucionales.
2. Mistral visión (Pixtral-12B-2409): un modelo de lenguaje de vanguardia con capacidad multimodal, valioso para análisis semántico visual y contexto de las imágenes.
3. OpenAI Vision: set de herramientas basado en modelos de visión artificial con capacidad de comprender imágenes de manera contextual, integrados con procesamiento de lenguaje natural.

Cada una de las tecnologías se describe con detalle, incluyendo su funcionamiento técnico, ventajas/desventajas, licenciamiento, costos y otros parámetros pertinentes

como precisión, requerimientos computacionales, compatibilidad con balanceo de carga y facilidad de integración entre otros. El estudio servirá de fundamento más adelante para la selección de la tecnología más apropiada para la creación del prototipo.

2.2. TECNOLOGÍA 1: YOLO (YOU ONLY LOOK ONCE)

2.2.1 DESCRIPCIÓN GENERAL

YOLO, cuyas siglas significan You Only Look Once, es una propuesta la cual ha revolucionado la arquitectura de la red neuronal convolucional en el área de detección de objetos en visión de computadora ya que a diferencia de alternativas como R-CNN, donde el primer paso es generar regiones propuestas y el segundo es clasificar estas propuestas. YOLO en cambio propone un enfoque integral, donde en una única pasada sobre la imagen completa haga la detección. Este modelo trata la detección como un problema de regresión, prediciendo simultáneamente las cajas delimitadoras y las clases de los objetos. Desde su introducción por Redmon et al. en 2016, YOLO ha destacado por su eficiencia y velocidad, lo que lo convierte en una herramienta fundamental para aplicaciones en tiempo real como vigilancia, conducción autónoma y análisis de video. Su capacidad para procesar imágenes de forma rápida y precisa ha marcado un hito en el desarrollo de sistemas inteligentes de visión artificial.

Entre las operaciones de visión por computadora que pueden ser tratadas con YOLO se encuentran las siguientes: la detección de objetos, que es la identificación simultánea y localización al mismo tiempo de varios elementos de distintas categorías en una imagen o secuencia de video; la segmentación de instancias, que es la discriminación de objetos individuales al nivel de píxel, incluso si son parte de la misma clase; la clasificación de imágenes y el reconocimiento de objetos, orientado a asignar etiquetas o categorías a imágenes completas o a regiones específicas; y el seguimiento de objetos, que es mantener la identificación de cada

uno de los objetos detectados en el espacio-tiempo en secuencias dinámicas, lo cual es crucial para aplicaciones en video vigilancia, análisis de tráfico, entre otros[20].

2.2.2 FUNCIONAMIENTO TÉCNICO

YOLO tiene un funcionamiento en el que se caracteriza un enfoque directo y unificado en la detección de objetos, evitando la complejidad de los métodos tradicionales como R-CNN, que dependen de múltiples etapas. En lugar de ello, YOLO utiliza una red convolucional completamente integrada que procesa la imagen de entrada y, en una sola pasada, genera tanto las coordenadas de los cuadros delimitadores como las probabilidades de clase correspondientes. De esta forma la detección en la imagen se plantea como un problema de regresión que asocia directamente los píxeles de la imagen con la ubicación y clasificación de los objetos presentes[21].

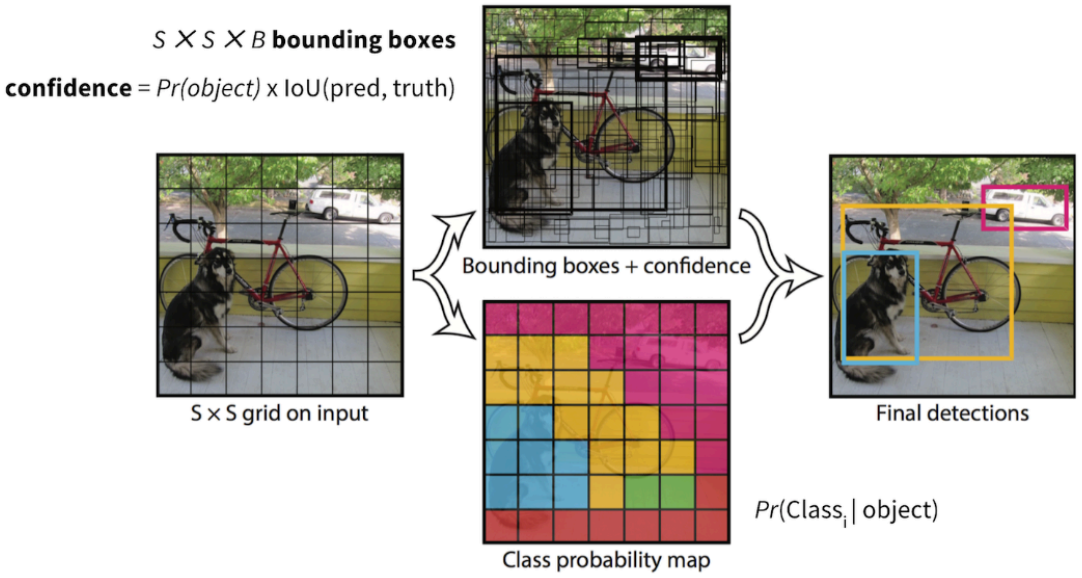
El algoritmo divide la imagen de entrada en una cuadrícula S × S, donde para cada celda, se predicen B cuadros delimitadores, sus respectivas puntuaciones de confianza, y C probabilidades de clase, donde C es el número de categorías a detectar. Entonces cada predicción incluye las coordenadas del centro del objeto (x, y), el ancho y la altura del cuadro (w, h), así como una puntuación de confianza que indica la certeza de que el cuadro contiene un objeto y su precisión estimada (IOU) respecto a la verdad de terreno (ground truth).

---

[20]JOCHER, G.; CHAURASIA, A.; STOKEN, A.; BOROVEC, J.; ULTRALYTICS. YOLOv5 by Ultralytics. [Código fuente en línea]. [Localidad no especificada]: GitHub, 2023. [citado el 20 de agosto de 2025]. Disponible en: https://github.com/ultralytics/yolov5

[21]REDMON, J.; DIVVALA, S.; GIRSHICK, R.; FARHADI, A. You Only Look Once: Unified, Real-Time Object Detection. En: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). [en línea]. [Localidad no especificada], 2016; p. 779–788. [citado el 20 de agosto 2025]. Disponible en: https://doi.org/10.1109/CVPR.2016.91

Figura 5. End-to-end YOLO Implementation



Fuente: Desai y otros (2024).

Las versiones más modernas de YOLO, como YOLOv3, YOLOv4 y las implementaciones recientes de YOLOv5 y YOLOv8 se introducen avances a nivel de arquitectura que son significativos. Se incluyen el uso de redes neuronales profundas como Darknet-53 y CSPDarknet para la extracción de características, la introducción de FPN (Feature Pyramid Networks) para detectar objetos en múltiples escalas, y la utilización de técnicas como la asignación de anclas (anchor boxes), NMS (Non-Maximum Suppression), y aumentos de datos específicas como Mosaic y MixUp para mejorar la generalización del modelo.[22]

El diseño de YOLO es orientado especificamente a la optimizacion de inferencia en aplicaciones que requieren procesamiento en tiempo real. Su arquitectura compacta y la eliminación de cálculos redundantes permiten que el modelo procese imágenes a velocidades superiores a 30 FPS en hardware con GPU modernas, haciéndolo especialmente adecuado para sistemas embebidos, aplicaciones móviles y transmisión de video en vivo.

[22]BOCHKOVSKIY, A.; WANG, C.-Y.; LIAO, H.-Y. M. YOLOv4: Optimal Speed and Accuracy of Object Detection. [Preprint]. En: arXiv preprint. [en línea]. [Localidad no especificada]: arXiv, 2020. [citado el 20 de agosto de 2025]. Disponible en: https://arxiv.org/abs/2004.10934

Respecto a los resultados, YOLO solo proporciona las predicciones con mayor probabilidad, utilizando un límite de confianza y métodos de supresión no máxima para prevenir repeticiones. Así, el sistema consigue un reconocimiento exacto, eficaz y uniforme.[23]

2.2.3 COSTOS

Utilizar YOLO (You Only Look Once) implica un conjunto de costos que cambian según la modalidad de implementación y los recursos utilizados, en el ámbito de desarrollo independiente,existen distintas versiones como YOLOv5, YOLOv7 y YOLOv8, que son distribuidas bajo licencias de código abierto, permitiendo su utilización sin costo directo asociado al software. A pesar de ello, es necesario contar con la infraestructura computacional adecuada, particularmente una unidad de procesamiento gráfico (GPU) de alto rendimiento. Tal infraestructura puede estar disponible de forma local o puede ser adquirida a través de servicios en la nube como Google Colab. A pesar de ofrecer el acceso inicial gratuito a través de sus plataformas, el uso intensivo o profesional suele requerir la contratación de planes pagos que aseguren la disponibilidad y el rendimiento requeridos. Si bien aunque el software propiamente dicho pueda ser gratuito, los gastos asociados a la computación y el almacenamiento representan un aspecto relevante a considerar en la planificación de proyectos basados en YOLO[24].

2.2.4 LICENCIAMIENTO

La licencia de modelos YOLO (You Only Look Once) puede variar con la versión específica del modelo y el autor. Las versiones iniciales, tales como YOLOv1 a YOLOv4, fueron creadas por Joseph Redmon y otros colaboradores, y se encuentran accesibles bajo la licencia pública general de GNU, versión 2.0 (GNU GPLv2). Esta licencia, publicada en 1991 por la Fundación de Software Libre, es una licencia de software libre que otorga a los usuarios finales la libertad para ejecutar, examinar, distribuir y cambiar el software. cuya característica primordial es

[23] REDMON, J.; DIVVALA, S.; GIRSHICK, R.; FARHADI, A, op. cit.
[24] JOCHER, G.; CHAURASIA, A.; STOKEN, A.; BOROVEC, J.; ULTRALYTICS, op. cit.

el principio del "copyleft", que exige que toda obra derivada o fusionada con el software original también esté bajo la misma licencia, esto significa que si un desarrollador modifica un software licenciado bajo GPLv2 o lo integra en otro sistema, el código fuente completo del sistema resultante debe ser igualmente publicado bajo GPLv2[25].

Con la llegada de YOLOv5, mantenido por la empresa Ultralytics, el modelo fue licenciado originalmente bajo la GNU GPLv3, una edición más actual y rigurosa que la anterior. Esta licencia es de software libre y fue elaborada por la Free Software Foundation con el propósito de garantizar al usuario final la libertad de usar, estudiar, modificar y redistribuir el software. Lanzada en 2007, la GPLv3 amplía los principios del copyleft presentes en versiones anteriores, pidiendo que toda obra derivada conserve la misma licencia y sea, por ende, también software libre. Entre sus características distintivas, esta versión introduce cláusulas específicas para proteger a los usuarios frente a restricciones impuestas por tecnologías de gestión digital (DRM), aborda el uso del software en entornos de software como servicio (SaaS), e incluye disposiciones relacionadas con litigios de patentes, lo cual brinda más seguridad legal a los desarrolladores. En contextos comerciales, la GPLv3 impone la condición de que el código fuente completo del producto que utilice o derive el software licenciado bajo esta licencia sea el mismo, lo que puede impedir que se incluya en soluciones cerradas o propietarias.[26]

Por otro lado, Ultralytics adopta una licencia propia para YOLOv8 denominada Ultralytics Open-Source License , con esta licencia se busca un equilibrio entre la apertura del código y la flexibilidad de uso en aplicaciones comerciales. Se permite utilizar, modificar y distribuir el software, incluyendo su uso en productos comerciales, siempre que se cumpla con el reconocimiento de autoría (atribución) y con cualquier otra condición impuesta por Ultralytics. A diferencia de la GPL, esta licencia no exige que las obras derivadas sean distribuidas bajo la misma licencia, lo

[25]FREE SOFTWARE FOUNDATION. GNU General Public License version 2. [Documento legal en línea]. Boston, Massachusetts: Free Software Foundation, 2007. [citado el 22 de agosto de 2025]. Disponible en: https://www.gnu.org/licenses/old-licenses/gpl-2.0.html

[26]FREE SOFTWARE FOUNDATION. GNU General Public License version 3. [Documento legal en línea]. Boston, Massachusetts: Free Software Foundation, 2007. [citado el 22 de agosto de 2025]. Disponible en: https://www.gnu.org/licenses/gpl-3.0.html

cual la hace más permisiva y adecuada para entornos empresariales que requieren mantener partes del software como propiedad privada[27].

Debe considerarse que los modelos como YOLOv5 y YOLOv8 se emplean sin ningun costo alguno, siempre que se respeten los términos indicados en sus licencias, pero el uso de plataformas comerciales relacionadas para el entrenamiento, anotación y la implementación de modelos como por ejemplo: Ultralytics HUB o Roboflow, está sometido a licencias comerciales concretas, que incluyen o pueden incluir restricciones adicionales y tarifas por uso, almacenamiento o acceso a funcionalidades avanzadas, y que deben ser tenidas en consideración a la hora de planificar el desarrollo de un servicio que requiera estos servicios.

2.2.5 FACILIDAD DE INTEGRACIÓN

Los modelos que se basan en la arquitectura YOLO (You Only Look Once) se destacan debido a su sencilla adaptación en diferentes entornos de desarrollo, tanto para propósitos educativos como en la industria. Esta arquitectura fue diseñada para ofrecer una detección de objetos en tiempo real mediante una sola pasada por la red neuronal, lo que reduce significativamente la latencia del procesamiento (Redmon et al., 2016). En las últimas versiones, como YOLOv5 y YOLOv8, la compatibilidad con ambientes de desarrollo conocidos como Python y PyTorch, acompañada con la presencia de scripts para entrenamiento e inferencia, hace que sea mas simple su integración en sistemas locales, servidores o plataformas en la nube. Adicional, herramientas como Ultralytics HUB y Roboflow optimizan aún más la integración de modelos YOLO gracias a que automatizan procesos complejos relacionados con el entrenamiento, gestión de datos y despliegue, mediante interfaces gráficas accesibles y APIs que simplifican la interacción incluso para usuarios con conocimientos técnicos limitados. Estas plataformas lo que hacen es permiter gestionar flujos de trabajo completos sin necesidad de configurar entornos complejos o escribir código extenso, lo que contribuye a acelerar la adopción de

[27]ULTRALYTICS. Ultralytics Open-Source License. [Documento legal en línea]. [Localidad no especificada]: GitHub, 2023. [citado el 22 de agosto de 2025]. Disponible en: https://github.com/ultralytics/ultralytics/blob/main/LICENSE

tecnologías de visión por computador. El soporte activo de documentación oficial y una comunidad amplia también favorecen la rápida solución de problemas y la actualización continua de los modelo[28].

2.2.6 ACURA

YOLO11 según la documentación oficial el modelo alcanza un 54.7% de mAP (mean Average Precision) en su variante YOLO11x cuando se evalúa en el dataset COCO que constituye el estándar de la industria para medir el rendimiento de modelos de detección de objetos. Esta medida muestra entonces la exactitud media del modelo al identificar y clasificar varios objetos en imágenes complejas de manera simultánea, teniendo en cuenta tanto la precisión de localización (IoU - Intersection over Union) como la clasificación adecuada de las categorías identificadas. El mAP se determina al valorar la exactitud del modelo en distintos niveles de confianza y promediar los resultados entre todas las categorías de objetos que se encuentran en el dataset; de este modo, se ofrece una medida integral del desempeño general del sistema de detección[29].

YOLO11 se presenta en cinco versiones que brindan diversas proporciones entre velocidad, precisión y requisitos computacionales. YOLO11n (nano) logra un mAP del 39.5% y está optimizado para dispositivos de recursos escasos, por ejemplo sistemas embebidos o teléfonos inteligentes. YOLO11s (small) alcanza un 47.0% mAP, lo que indica un balance entre la velocidad de procesamiento y la capacidad para detectar, por lo que es apto para aplicaciones que necesitan respuestas veloces sin renunciar en demasía a la precisión. YOLO11m (medium) obtiene 51.5% mAP y constituye la opción intermedia recomendada para la mayoría de aplicaciones industriales, mientras que YOLO11l (large) consigue 53.4% mAP ofreciendo alta precisión con velocidad moderada apropiado para aplicaciones donde la exactitud es prioritaria pero se mantienen restricciones de tiempo real, por ultimo se tiene a YOLO11x (extra-large) representa la máxima precisión disponible

---

[28] JOCHER, G.; CHAURASIA, A.; STOKEN, A.; BOROVEC, J, op. cit.
[29] ULTRALYTICS. YOLOv11 Models: Deployment on Edge Devices. [Documentación en línea]. [Localidad no especificada]: Ultralytics, 2025. [citado el 23 de agosto de 2025]. Disponible en: https://docs.ultralytics.com/models/yolo11/#can-yolo11-be-deployed-on-edge-devices

con 54.7% mAP, diseñado para escenarios donde se privilegia la exactitud sobre la velocidad de procesamiento[30].

2.2.7 TIEMPO DE DESARROLLO E IMPLEMENTACIÓN

El tiempo requerido para la integración y puesta en marcha de YOLO comprende de varias etapas que abarcan desde la configuración inicial hasta la integración final:

1. Preparación del entorno: Esta etapa incluye seleccionar y arreglar el espacio de desarrollo, ya sea con Ultralytics (al instalarlo localmente o en Google Colab) o con una plataforma simple como Roboflow. En los dos casos, es necesario necesario instalar dependencias como Python, PyTorch y los paquetes asociados al framework YOLO. También se realiza la validación del entorno para garantizar que las versiones de librerías, el acceso a GPU (si corresponde) y la conectividad con los datos están correctamente configurados ; esto puede tomar entre uno a dos días, dependiendo también de la experiencia del programador.

2. Carga y verificación del dataset: Para el caso de conjuntos de datos ya estructurados y etiquetados, como el dataset de firmas manuscritas disponible en la documentación oficial de Ultralytics, esta fase consiste en la descarga del archivo .yaml que define las rutas y clases del dataset, la verificación de su integridad, y su adecuación al entorno de entrenamiento (por ejemplo, asegurarse de que las rutas son accesibles desde el entorno local o en la nube). Esta etapa es relativamente sencilla, y puede tardar entre 3 a 4 horas aunque es sencilla es esencial para garantizar una ejecución fluida del entrenamiento.

3. Entrenamiento del modelo (6 - 48 horas): El entrenamiento es la fase más intensiva en recursos. Utilizando una GPU moderna (por ejemplo, Nvidia RTX 3060 o superior), entrenar un modelo con un dataset de tamaño moderado puede tomar entre 6 y 24 horas para obtener resultados iniciales aceptables.

[30] Ibid.

Para un entrenamiento completo con ajuste fino, el tiempo puede extenderse hasta 48 horas o más.

2.2.8 REQUERIMIENTOS COMPUTACIONALES

Los recursos computacionales disponibles tienen una fuerte conexión con el desempeño de los modelos YOLO (You Only Look Once), en particular durante las fases de ejecución en tiempo real y entrenamiento. Se aconseja utilizar unidades de procesamiento gráfico (GPU) con arquitectura CUDA compatibles para el entrenamiento, como las de la serie NVIDIA RTX o más avanzadas, ya que estas posibilitan la aceleración del cálculo de operaciones matriciales intensivas propias del aprendizaje profundo. Los modelos más nuevos, como YOLOv5 o YOLOv8, tienen la capacidad de ejecutarse en máquinas con GPU de gama media o alta y al menos 8 GB de memoria de video dentro de los entornos productivos, aunque versiones más ligeras (por ejemplo, YOLOv5s o YOLO-Nano) están optimizadas para dispositivos con recursos limitados, incluyendo CPU o edge devices como Raspberry Pi o NVIDIA Jetson (Ultralytics, 2023). Además, se requiere un entorno de desarrollo compatible, como Python 3.8 o superior, junto con bibliotecas como PyTorch, OpenCV y cuDNN para el manejo de operaciones de red neuronal y procesamiento de imágenes. La cantidad de memoria RAM sugerida varía entre 8 y 32 GB, en función de lo complejos que sean los modelos y del tamaño del conjunto de datos. En entornos de formación en la nube o distribuidos, existen plataformas como Microsoft Azure, Amazon SageMaker o Google Colab que cuentan con configuraciones predeterminadas que cumplen con los requerimientos técnicos para poner en marcha YOLO sin que se requiera contar con una infraestructura local especializada.

2.2.9 COMPATIBILIDAD CON BALANCEO DE CARGA

Los modelos YOLO (You Only Look Once) son compatibles con esquemas de balanceo de carga en arquitecturas distribuidas. Esta compatibilidad se consigue separando el modelo en servicios independientes. Los servicios se pueden desplegar como contenedores (por ejemplo, usando Docker) o microservicios, permitiendo su integración en entornos que utilizan balanceadores de carga y

orquestadores como Kubernetes. Gracias a esta capacidad, se pueden distribuir las solicitudes de inferencia entre muchas instancias del modelo, ya sea en servidores locales o en la nube. Un balanceo es esencial para mantener un rendimiento constante ante cargas variables de procesamiento. Adicional se tienen frameworks como TorchServe, ONNX Runtime o TensorRT permiten servir modelos YOLO en paralelo, habilitando el escalado horizontal para aplicaciones en tiempo real. La integración con soluciones de balanceo de carga hace más fácil el despligue de sistemas robustos, tolerantes a fallos y con alta disponibilidad, especialmente en aplicaciones críticas como vigilancia inteligente, análisis de vídeo en streaming o monitoreo industrial.[31]

2.2.10 SOPORTE Y DOCUMENTACIÓN

Los modelos YOLO (You Only Look Once) reciben amplio soporte técnico y documentación, lo que las vuelve mucho más sencillas de implementarse en proyectos industriales y académicos. Sus versiones más recientes, como YOLOv5 e YOLOv8, mantenidas por la organización Ultralytics, tienen documentación oficial en línea con guías de instalación, ejemplos de entrenamiento e inferencia, líneas directrices para personalización de modelos, despliegue sobre diferentes entornos y resolución de problemas comunes, se complementa también con repositorios públicos en plataformas como GitHub donde se encuentran tanto el código fuente como actualizaciones periódicas, permitiendo la participación activa de la comunidad en el desarrollo del software. Aparte , se puede contar con tutoriales interactivos, notebooks en Google Colab y foros de discusión donde usuarios y desarrolladores intercambian soluciones. Igualmente el soporte no oficial también es generoso, se cuenta con publicaciones académicas, videos educativos, artículos técnicos donde se detallan casos de uso, buenas prácticas y comparativas con otros modelos de detección de objetos.[32] Toda esta disponibilidad de documentación y otras fuentes contribuyen a reducir significativamente la barrera de iniciación de nuevos usuarios y facilita la integración del modelo en distintos contextos tecnológicos.

---

[31]ULTRALYTICS. Computer Vision Tasks supported by Ultralytics YOLO11. [en línea]. [Localidad no especificada]: Ultralytics, 30 de marzo de 2025. [citado el 26 de agosto de 2025]. Disponible en: https://docs.ultralytics.com/es/tasks/

[32] Ibid.

2.2.11 VENTAJAS

- Alta velocidad y eficiencia para procesamiento en tiempo real
- Compatible con múltiples entornos y lenguajes.
- Modelo gratuito y de código abierto.

2.2.12 DESVENTAJAS

- Entrenamiento intensivo en GPU.
- Requiere datasets bien etiquetados.
- No tiene comprensión semántica ni contexto textual.
- Menor rendimiento en imágenes muy complejas o confusas.

2.3 TECNOLOGÍA 2: MISTRAL

2.3.1 DESCRIPCIÓN GENERAL

Mistral Vision, especialmente su modelo Pixtral-12B-2409, es una tecnología de inteligencia artificial creada por la compañía francesa Mistral AI, experta en modelos de lenguaje e imagen fundacionales. Este modelo fusiona la habilidad para entender imágenes con la gestión del lenguaje natural, facilitando el estudio de imágenes con un entendimiento semántico profundo. Dentro del ámbito del reconocimiento de firmas manuscritas, Mistral Vision proporciona un método multimodal que facilita la identificación, categorización y descripción de firmas en documentos digitalizados o escaneados[33].

En contraposición a otros métodos que solo se enfocan en la ubicación visual, Mistral Vision incluye una comprensión contextual del documento, lo que facilita la distinción entre una firma, un texto escrito o un gráfico. Esta comprensión se consigue a través de su arquitectura de gran escala y formación en variados

[33] MISTRAL AI. Announcing Pixtral 12B. [Anuncio en línea]. [Localidad no especificada]: Mistral AI, 17 de septiembre de 2024. [citado el 26 de agosto de 2025]. Disponible en: https://mistral.ai/news/pixtral-12b

conjuntos de datos, tanto visuales como lingüísticos. Esto transforma a Mistral Vision en un instrumento con gran potencial para aplicaciones que necesitan interpretación precisa en documentos complejos o desorganizados.

Mistral AI ha elegido un método de código abierto, lo que facilita a entidades académicas y programadores el acceso al modelo, su adaptación y evaluación en situaciones concretas como la identificación de autenticidad en documentos. Su accesibilidad y adaptabilidad lo establecen como una opción viable y contemporánea en comparación con otras tecnologías estrictamente especializadas o cerradas. Adicionalmente, su conexión con herramientas y librerías de Inteligencia Artificial populares permite su integración en sistemas actuales para labores como el análisis forense, la automatización de procedimientos legales o la digitalización de documentos históricos.

2.3.2 FUNCIONAMIENTO TÉCNICO

El Pixtral-12B-2409 es un sistema de inteligencia artificial multimodal que une la habilidad para el procesamiento del lenguaje natural con la interpretación visual en una solo sistema integrado. Esto es posible mediante una arquitectura que fusiona un codificador de visión con un decodificador de lenguaje, lo cual permite al modelo procesar simultáneamente entradas tanto textuales como visuales. el modelo fracciona una imagen en parches pequeños, que suelen tener dimensiones de 16x16 píxeles, convierte cada parche en una representación tokenizada. Siguiente, estos tokens de imagen se alinean luego con los tokens textuales para formar una secuencia coherente que es procesada por el decodificador multimodal[34].

Con 400 millones de parámetros, el codificador de visión está configurado para registrar detalles visuales significativos y relaciones espaciales entre distintas áreas de la imagen. El modelo emplea embeddings posicionales rotatorios bidimensionales (2D RoPE) con el fin de mantener la información posicional en la imagen. Este método codifica las posiciones relativas en dos dimensiones, lo que le brinda al decodificador la capacidad de analizar la estructura espacial sin

[34] Ibid.

comprometer la integridad del contenido visual. Esto es crucial para aplicaciones donde la interpretación del contexto visual y su relación con el texto es fundamental, como en la comprensión de documentos gráficos complejos o imágenes acompañadas de texto explicativo[35].

En cuanto al decodificador, tiene 12 mil millones de parámetros y se fundamenta en la arquitectura Mistral Nemo, que es famosa por su capacidad para procesar secuencias extensas y producir textos coherentes. Esta capacidad le permite manejar secuencias de hasta 128,000 tokens, lo que representa un avance significativo en comparación con modelos previos que tenían limitaciones más estrictas en cuanto a la longitud de contexto. Esta amplia ventana de contexto facilita el procesamiento conjunto de múltiples imágenes y textos extensos, integrándolos en una única consulta que el modelo puede analizar de forma conjunta[36].

Pixtral-12B-2409 usa tokens específicos para señalar el comienzo y el final de bloques de imagen, como `[IMG BREAK]` y `[IMG END]`, con el fin de administrar la interacción entre texto e imagen en una misma secuencia. Estos tokens actúan como separadores que posibilitan que el modelo distinga entre cuándo está analizando texto y cuándo está procesando contenido visual, lo que permite una estructura clara y facilita la generación de respuestas precisas y contextualmente adecuadas. Esta estructura posibilita que el modelo comprenda preguntas que incluyen referencias cruzadas entre texto e imágenes, ofreciendo respuestas más integradas y complejas[37].

En cuanto a su funcionamiento, este diseño permite que Pixtral-12B-2409 tenga una versatilidad particular para aplicaciones de lenguaje y visión, incluyendo análisis documental, sistemas de preguntas y respuestas con soporte visual, interpretación de gráficos y datos visuales complejos, así como la generación de descripciones detalladas y coherentes basadas en contenido mixto. Al combinar la potencia de un gran modelo de lenguaje con un codificador de visión optimizado, Pixtral-12B-2409

[35] Ibid.
[36] Ibid.
[37] Ibid.

representa un paso importante hacia modelos de inteligencia artificial más completos, capaces de entender y generar conocimiento a partir de fuentes multimodales complejas.[38]

2.3.3 COSTOS

El modelo Pixtral-12B-2409 posibilita que las imágenes sean procesadas a través de un método de tokenización fundamentado en bloques de píxeles. Cada imagen se divide internamente en bloques de 16×16 píxeles, y cada bloque es considerado como un token. Según la documentación técnica oficial, se puede determinar el total aproximado de tokens que produce una imagen con la siguiente ecuación[39]:

$$número\ de\ tokens \approx \frac{Resolución\ X \times Resolución\ Y}{256}$$

Donde:

Resolución X: número de píxeles horizontales de la imagen.

Resolución Y: número de píxeles verticales de la imagen.

Es importante señalar que todas las imágenes con una resolución superior a 1024×1024 píxeles son automáticamente redimensionadas por el modelo, manteniendo la proporción de aspecto. Esta redimensión asegura que el eje mayor de la imagen no supere los 1024 píxeles, con el objetivo de optimizar el uso de tokens y mantener la consistencia computacional[40].

De acuerdo con la política de precios vigente para Pixtral-12B-2409, el procesamiento de tokens tiene los siguientes valores: Costo por millón de tokens de entrada: $0.15 USD y Costo por millón de tokens de salida: $0.15 USD[41].

Con base en ello, el costo de procesamiento de una imagen se estima como:

$$Costo\ (usd) \approx (\frac{Resolución\ X \times Resolución\ Y}{256})\ x\ 0.00000015$$

---

[38] Ibid.
[39] Ibid.
[40] Ibid
[41] LLM STATS. Pixtral-12B: Pricing, Context Window, Benchmarks, and More. [Página web en línea]. [Localidad no especificada]: LLM Stats, 2024. [citado el 26 de agosto de 2025]. Disponible en: https://llm-stats.com/models/pixtral-12b-2409

Se debe tener en cuenta que este valor corresponde únicamente al costo al costo por tokens de entrada. Si la operación involucra una respuesta generada por el modelo (tokens de salida), el costo de salida debe calcularse adicionalmente utilizando la misma tarifa por token.

2.3.4 LICENCIAMIENTO

Pixtral-12B-2409 es Apache 2.0 usa la licencia Apache 2.0 la cual es una licencia de software libre y de código abierto ampliamente utilizada que permite a los usuarios utilizar, modificar y distribuir software bajo ciertas condiciones. Esta licencia otorga una gran libertad al usuario, incluyendo la posibilidad de incorporar el software en proyectos propios, ya sean comerciales o no comerciales, sin la obligación de liberar el código fuente derivado. Sin embargo, la licencia impone requisitos importantes, como la inclusión de un aviso de derechos de autor, la preservación de la licencia original y la prohibición de utilizar marcas registradas sin permiso. Además, Apache 2.0 proporciona una cláusula explícita de concesión de patentes, lo que protege a los usuarios frente a litigios relacionados con patentes que puedan surgir por el uso del software[42].

2.3.5 FACILIDAD DE INTEGRACIÓN

El modelo Pixtral-12B-2409 puede ser integrado utilizando dos métodos fundamentales: por medio de la interfaz de programación de aplicaciones (API) que brinda La Plateforme, el espacio para acceder a Mistral AI en la nube o mediante la descarga directa del modelo para ejecutarlo localmente. En el primer caso, el modelo es accesible mediante solicitudes HTTP estructuradas, lo que permite enviar imágenes como entradas y recibir respuestas tokenizadas. Para acceder a esto, se necesita una clave de autenticación (API key), que se otorga a los usuarios que están registrados en el entorno de La Plateforme. Las respuestas del modelo

[42] PERENS, B. Understanding Open Source Licenses: Apache 2.0 and Beyond. En: Open Source Initiative. [Documento en línea]. [Localidad no especificada]: Open Source Initiative, 2023. [citado el 26 de agosto de 2025]. Disponible en: https://opensource.org/licenses/Apache-2.0

siguen el formato JSON, lo cual facilita su incorporación en aplicaciones desarrolladas en lenguajes como Python, JavaScript, o frameworks orientados a servicios web.

Como otra opción, los usuarios tienen la posibilidad de bajar el modelo desde los repositorios que Mistral AI ha publicado, bajo licencias abiertas que posibilitan su funcionamiento en infraestructuras locales, con la condición de tener a disposición los recursos computacionales requeridos para modelos de gran escala. Para aplicaciones que requieren control completo del entorno de inferencia o para integraciones personalizadas fuera del servicio en la nube, este tipo de integración requiere un conocimiento más alto.

2.3.6 ACURA

El modelo Pixtral 12B de Mistral AI fue evaluado junto a otros modelos abiertos y cerrados utilizando un protocolo común de pruebas multimodales. En tareas como comprensión de documentos, gráficos y preguntas visuales, Pixtral 12B superó a modelos como Claude-3 Haiku y Gemini Flash 8B, e incluso igualó el rendimiento de modelos más grandes como LLaVA-OV 72B[43].

Figura 6. Pixtral-12B evaluation protocol.

Model	MMMU (CoT)	Mathvista (CoT)	ChartQA (CoT)	DocVQA (ANLS)	VQAv2 (VQA Match)	MM MT-Bench
Pixtral 12B	**52.5**	**58.0**	**81.8**	**90.7**	**78.6**	**6.05**
Claude-3 Haiku	50.4	44.8	69.6	74.6	68.4	5.46
Gemini-1.5 Flash 8B (0827)	50.7	56.9	78.0	79.5	65.5	5.93
LLaVA-OV 72B	54.4	57.2	66.9	91.6	83.8	4.95
GPT-4o	68.6	64.6	85.1	88.9	77.8	7.72
Claude-3.5 Sonnet	68.0	64.4	87.6	90.3	70.7	7.50

Fuente: Mistral AI (2024).

[43] MISTRAL AI, *op. cit.*

2.3.7 TIEMPO DE DESARROLLO E IMPLEMENTACIÓN

El tiempo requerido para la integración y puesta en marcha del modelo Pixtral-12B-2409 depende del modo de uso que se seleccione:

mediante el API en la nube: En el caso de integración a través de la API ofrecida por La Plateforme (plataforma en línea de Mistral AI), el proceso de implementación puede ser relativamente rápido. Esta modalidad requiere la obtención de una clave de acceso (API key) como fue explicado anteriormente, la configuración de los encabezados de autenticación en las solicitudes HTTP, y el tratamiento de respuestas en formato JSON. En un entorno donde ya se disponga de una infraestructura compatible con llamadas a servicios web RESTful, está puede implementarse en un total de 2 días (sin realizar algún ajuste adicional)

Implementación local: la descarga y ejecución del modelo en infraestructura local implica mayores tiempos de preparación. Este enfoque requiere la provisión de recursos de cómputo adecuados (como GPUs con memoria suficiente), la configuración del entorno de ejecución, es muy dependiente también por la experiencia del técnico por lo que el tiempo estimado para una integración funcional puede extenderse a 1 o 2 semanas o incluso más.

2.3.8 REQUERIMIENTOS COMPUTACIONALES

El uso del modelo vía API, proporcionada por el entorno La Plateforme, no requiere infraestructura computacional local para la inferencia del modelo. La ejecución se realiza en los servidores administrados por Mistral AI. En este caso, los requerimientos se reducen al entorno del cliente para consumir servicios web.

Por otro lado, descargar el modelo y usarlo localmente requiere que se despliegue el modelo en una unidad de procesamiento gráfico (GPU), la ejecución local permite mayor control sobre el comportamiento del modelo, pero requiere infraestructura especializada y conocimientos técnicos avanzados para su instalación y mantenimiento.

2.3.9 COMPATIBILIDAD CON BALANCEO DE CARGA

Pixtral-12B-2409 tiene la posibilidad de integrarse en esquemas de balanceo de carga, dependiendo de la modalidad de uso seleccionada. En La Plateforme, si se accede a través de la API, la infraestructura en la nube de Mistral AI es responsable del equilibrio entre carga y escalabilidad, por lo que el usuario no necesita configurar nada adicionalmente. Las llamadas a la API se pueden distribuir de forma natural entre múltiples instancias cliente, sin generar colisiones ni afectar la integridad del servicio, lo cual permite operar en entornos distribuidos o multiusuario.

Por otro lado, cuando el modelo se ejecuta en entornos locales, la compatibilidad con balanceo de carga depende de la arquitectura implementada por el usuario. La ejecución concurrente del modelo requiere asegurar que los recursos computacionales (particularmente GPU) sean suficientes y estén correctamente aislados o coordinados para evitar sobrecargas y garantizar la estabilidad del servicio. Por lo que se puede concluir que se puede desplegar en una arquitectura de balanceo de carga pero se requiere más conocimientos técnicos[44].

2.3.10 SOPORTE Y DOCUMENTACIÓN

El modelo Pixtral-12B-2409 tiene a su disposición soporte documental oficial, que Mistral AI brinda y se puede acceder por medio de su portal de documentación en línea. proporciona indicaciones sobre cómo utilizar el modelo tanto mediante la API alojada en La Plateforme como para su descarga y ejecución en entornos locales. Los recursos que se tienen disponibles cubren temas como la estructura de las peticiones HTTP, formatos de entrada y salida, autenticación mediante clave API, manejo de errores, y ejemplos de integración en distintos lenguajes de programación. Adicional, se registra el procedimiento de inferencia sobre imágenes, con detalles acerca de los tamaños y formatos permitidos, así como la conducta del modelo frente a diferentes clases de entrada visual.

[44] MISTRAL AI, op. cit.

Respecto al soporte técnico, Mistral AI brinda vías oficiales para solucionar problemas e interrogantes técnicas, los cuales varían según el tipo de usuario (comunidad, desarrolladores registrados o clientes con acuerdos empresariales). Adicional, se brinda acceso a repositorios públicos con modelos pre entrenados y scripts de uso, así como foros comunitarios y sistemas de seguimiento de errores para la retroalimentación y mejora continua del sistema. Pero se debe a tener en cuenta que la disponibilidad y nivel de soporte personalizado pueden estar condicionados por el tipo de suscripción o licencia adquirida.

La documentación se actualiza con frecuencia e incorpora versiones históricas del modelo, lo que posibilita conservar la compatibilidad con las versiones precedentes y facilita los procesos de migración o actualización en sistemas productivos.[45]

2.3.11 VENTAJAS

- Alta velocidad y eficiencia para procesamiento en tiempo real.
- Compatible con múltiples entornos y lenguajes.
- Modelo gratuito y de código abierto.
- Flexible en despliegue (local o nube).
- Gran comunidad y documentación disponible.

2.3.12 DESVENTAJAS

- Entrenamiento intensivo en GPU.
- Requiere datasets bien etiquetados.
- No tiene comprensión semántica ni contexto textual.
- Menor rendimiento en imágenes muy complejas o confusas.
- El peso límite del archivo es de 10MB, solo permite: PNG (.png), JPEG (.jpeg and .jpg), WEBP (.webp) Non-animated GIF with only one frame (.gif)

[45] MISTRAL AI. op. cit.

2.4 TECNOLOGÍA 3: GTP-4.1 MINI

2.4.1 DESCRIPCIÓN GENERAL

GPT-4.1 Mini es una versión más pequeña y optimizada del modelo multimodal que OpenAI creó para la familia GPT-4. Esta versión fue concebida con el objetivo de proporcionar habilidades de análisis automático y razonamiento más complejas, utilizando menos recursos computacionales y tiempo en contraste con las versiones completas de GPT. GPT-4.1 Mini tiene habilidades multimodales, es decir, puede analizar y entender texto, imágenes y videos. Esto lo hace adecuado para tareas que necesitan la interpretación de imágenes junto con el entendimiento del lenguaje natural y una capacidad contextual ampliada de hasta un millón de tokens.[46]

Este modelo se diseñó para balancear eficiencia y potencia, resultando perfecto para aplicaciones que necesitan respuestas veloces y procesamiento en tiempo real sin renunciar a la calidad. Así pues, GPT-4.1 Mini fue seleccionado en comparación con sus otras versiones debido a su óptimo balance entre rendimiento, eficiencia y costos. La identificación de firmas requiere el análisis de patrones visuales complejos, lo que requiere un modelo que pueda manejar grandes cantidades de información con gran exactitud y velocidad, y este modelo lo consigue con un costo significativamente inferior a sus otras versiones, sin renunciar a su detenimiento y exactitud.

2.4.2 FUNCIONAMIENTO TÉCNICO

Aunque OpenAI no ha publicado detalles exhaustivos sobre la arquitectura exacta, se sabe que GPT-4.1 Mini utiliza la misma base de transformadores que los modelos mayores de la familia de modelos de lenguaje desarrollados por OpenAI, está arquitectura de transformadores es un tipo de red neuronal diseñada para procesar secuencias de datos, como texto o imágenes, mediante un mecanismo

[46] OPENAI. Introducing GPT-4.1 in the API. [Documento en línea]. [Localidad no especificada]: OpenAI, 14 de abril de 2025. [citado el 27 de agosto de 2025]. Disponible en: https://openai.com/index/gpt-4-1/

llamado autoatención que permite al modelo evaluar la relevancia de cada elemento en la secuencia respecto a los demás simultáneamente. Esta capacidad supera las limitaciones de los modelos anteriores que secuencialmente procesaron la información, lo que permite un entrenamiento más eficiente y una mejor captura de relaciones a largo plazo en los datos. La estructura básica de estos mecanismos comprende capas de codificadores y decodificación que trabajan con representaciones vectorizadas de los tokens y en donde la capa calcula pesos de atención y contexto para destilar información precisa de términos cercanos. Matemáticamente, la atención escalada de producto escalar se usa para calcular las relaciones entre tokens y realizar el proceso de manera optimizada mediante paralelismo en bloque operaciones matriciales.[47]

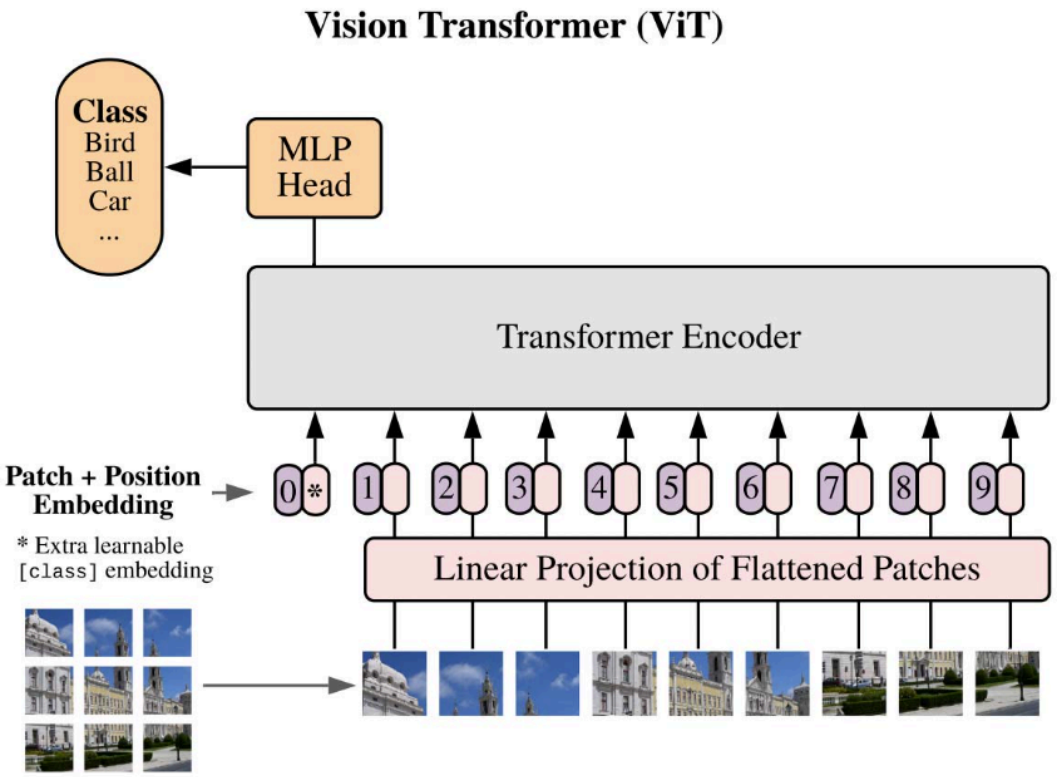
Adicionalmente se cree que GTP-4 utilice como base para su componente multimodal, Vision Transformer (ViT) que es una adaptación del modelo transformer explicado anteriormente, pero ahora pero aplicados a la clasificación de imágenes. En lugar de analizar palabras, ViT comienza dividiendo la imagen de entrada en pequeños bloques denominados patches, los cuales se aplanan y proyectan linealmente a vectores numéricos. A esta secuencia de vectores se le añade un class token (un vector especial que resume la información global de la imagen) y se incorporan embeddings posicionales para preservar la ubicación de cada patch. La secuencia completa se introduce en un codificador transformer que aplica mecanismos de atención para relacionar globalmente los patches entre sí. Finalmente, el class token procesado se envía a una red neuronal multicapa (MLP Head), que determina la clase asignada a la imagen como ave o automóvil[48]. Esto permite que el modelo pueda aprender qué secciones de la imagen son más importante para una tarea, como identificar objetos o categorizar imágenes.

---

[47]DATACAMP. Cómo funcionan los transformadores: Una exploración detallada. [en línea]. [Localidad no especificada]: DataCamp, 11 de septiembre de 2024. [citado el 27 de agosto de 2025]. Disponible en: https://www.datacamp.com/es/tutorial/how-transformers-work

[48]WAGH, A. What's new in GPT-4: Architecture and Capabilities. En: Medium. [Artículo en línea]. [Localidad no especificada]: Medium, 2 de julio de 2023. [citado el 27 de agosto de 2025]. Disponible en:https://medium.com/@amol-wagh/whats-new-in-gpt-4-an-overview-of-the-gpt-4-architecture-and-capabilities-of-next-generation-ai-900c445d5ffe

Figura 7. Vision Transformer (Vit)



Fuente: Speechmatics Blog

2.4.3 COSTOS

El modelo GPT-4.1-mini, desarrollado por OpenAI, permite el procesamiento de imágenes mediante la conversión de información visual en tokens, se basa en una estructura de codificación basada en "patches" o fragmentos fijos de 32 × 32 píxeles. El costo del uso de imágenes se calcula en función del número de estos bloques requeridos para cubrir la imagen por completo[49]. Para obtener dicha cantidad, se aplica la fórmula:

$$Patches\ W: \frac{(W + 32 - 1)}{32}$$

$$Patches\ H: \frac{(H + 32 - 1)}{32}$$

[49]OPENAI. Pricing. [Página web en línea]. [Localidad no especificada]: OpenAI, 2024. [citado el 27 de agosto de 2025]. Disponible en: https://openai.com/pricing

Donde:

- W: ancho original de la imagen en píxeles.
- H: alto original de la imagen en píxeles.

Esta fórmula garantiza halla un redondeo hacia arriba, aun cuando las dimensiones de la imagen no son múltiplos exactos de 32, como por ejemplo, una imagen de 1800 píxeles de ancho se traduce en (1800 + 32 - 1) // 32 = 57 bloques, ya que se requieren 57 divisiones de 32 píxeles para cubrir totalmente la anchura. De forma análoga, una altura de 2400 píxeles se convierte en 75 bloques, resultando en un total de 57 × 75 = 4275 patches[50].

Se debe tener en cuenta que el sistema establece un límite máximo de 1536 tokens por imagen. Si se llega al caso donde la cantidad calculada sobrepasa ese límite establecido, la imagen se escala proporcionalmente conservando su relación de aspecto, esto con tal de que pueda ser representada por un máximo de 1536 patches. Este procedimiento conlleva la implementación de un factor de reducción, cuyo valor se establece a través de la fórmula:

$$Factor\ de\ reducción: \sqrt{\frac{Tmax \cdot P^2}{W \cdot H}}$$

Donde:

- $Tmax$: el número máximo permitido de patches, que es, 1536.
- P^2: el tamaño del patch en píxeles, que es, 32.
- W: ancho original de la imagen en píxeles
- H: alto original de la imagen en píxeles

Este valor indica cuánto debe reducirse proporcionalmente cada dimensión para que la imagen total no supere el límite de procesamiento. Una vez escaladas las

[50] OPENAI. Ibid.

dimensiones, se redondean hacia abajo para asegurar un número entero de bloques por dimensión, garantizando una cobertura adecuada[51]:

$$H' = H \cdot Factor\ de\ reducción$$
$$W' = W \cdot Factor\ de\ reducción$$

$$Patches\ W = \frac{W'}{32}$$
$$Patches\ H = \frac{H'}{32}$$

Así, los tokens finales que representarán la imagen serán:

tokens imagen = patches W · patches H

En el caso de GPT-4.1-mini, este valor se multiplica por un factor de ajuste de 1.62 para obtener los tokens facturados, que se computan y cobran al mismo ritmo que los tokens de texto procesados:

tokens facturados = tokens imagen · 1.62

Ahora bien dichos tokens se facturan de acuerdo con una tarifa fija establecida por OpenAI. A la fecha de publicación, el costo por cada millón de tokens de entrada procesados (Envio pregunta) es de USD 0.40 y por cada millón de tokens de salida (Respuesta) el valor es de USD 1.60.[52]

2.4.4 LICENCIAMIENTO

GPT-4.1-mini opera bajo la licencia de consumo bajo demanda, también llamada "uso basado en tokens", en la que se define un marco legal y operativo que controla la relación entre el usuario y la tecnología. Primero, la licencia otorgada es

[51] OPENAI. Ibid.
[52] OPENAI. Ibid.

restringida, no exclusiva y revocable, lo que implica que el usuario adquiere el derecho a emplear el modelo mediante la API, pero sin obtener la propiedad del software o del código que lo acompaña. Esta licencia permite la integración del modelo en aplicaciones propias siempre que se respeten las políticas y restricciones definidas por OpenAI. El uso de este modelo se factura según la cantidad de tokens procesados, donde el usuario es el responsable de pagar las tarifas vigente según el consumo, estas tarifas pueden ser modificadas por OpenAI sin previo aviso, las suscripciones mensuales no son obligatorias se puede cancelar en cualquier momento.[53]

2.4.5 FACILIDAD DE INTEGRACIÓN

El modelo GPT-4.1-mini se puede integrar a través de una API REST ofrecida por openIA, la que facilita su integración en diversos ambientes y lenguajes de programación. La interfaz admite peticiones en formato JSON, facilitando la transmisión conjunta de texto e imágenes codificadas en base64. La instalación local del modelo no es necesaria, dado que el procesamiento se lleva a cabo totalmente en los servidores de OpenAI. La documentación técnica oficial ofrece directrices precisas acerca de la estructura de las peticiones, autenticación a través de claves API, gestión de errores y restricciones de uso. Esta forma de integración es compatible con arquitecturas de software modernas , que incluyen servicios en la nube, microservicios y aplicaciones basadas en la web.[54]

---

[53]OPENAI. Terms of use. [Documento legal en línea]. [Localidad no especificada]: OpenAI, 2024. [citado el 28 de agosto de 2025]. Disponible en: https://openai.com/terms

[54] OPENAI. Vision models and image inputs. [Documentación en línea]. [Localidad no especificada]: OpenAI Developer Documentation, 2024. Disponible en: https://platform.openai.com/docs/guides/images-vision

2.4.6 ACURA

Figura 8. Visión.

Visión

Categoría	GPT-4.1	GPT-4.1 mini	GPT-4.1 nano	GPT-4o (2024-11-20)	GPT-4o mini	OpenAI o1 (alto)	OpenAI o3-mini (alto)	GPT-4.5
MMMU	74,8%	72,7%	55,4%	68,7%	56,3%	77,6%	-	75,2%
MathVista	72,2%	73,1%	56,2%	61,4%	56,5%	71,8%	-	72,3%
CharXiv-R	56,7%	56,8%	40,5%	52,7%	36,8%	55,1%	-	55,4%
CharXiv-D	87,9%	88,4%	73,9%	85,3%	76,6%	88,9%	-	90.0%

Fuente: OpenAI (2025).

La tabla mostrada muestra los grados de exactitud obtenidos por distintos modelos de lenguaje multimodal, incluyendo GPT-4.1 mini, en exámenes estandarizados de interpretación visual. Estas evaluaciones se agrupan en cuatro métricas o benchmarks: MMMU, MathVista, CharXiv-R y CharXiv-D, cada una centrada en competencias particulares vinculadas con la interpretación de imágenes en conjunto con texto.

En la categoría de Comprensión Multimodal Multitarea Massiva (MMMU), los modelos deben responder a interrogantes de múltiples opciones que demandan el análisis de imágenes técnicas o académicas. Esta evaluación es especialmente rigurosa debido a la diversidad de tareas que implica. De acuerdo con la tabla, GPT-4.1 mini alcanza una precisión del 72,7 %, un porcentaje similar al de GPT-4.1 (74,8 %) y superior al rendimiento del modelo GPT-4o (68,7 %) y GPT-4o mini (56,3 %).

Por otro lado, MathVista es un conjunto de datos creado para valorar la capacidad de razonamiento matemático visual. En este contexto, las imágenes contienen expresiones matemáticas, diagramas o fórmulas que necesitan ser analizadas para solucionar un problema. GPT-4.1 mini llega al 73,1%, sobrepasando incluso a

GPT-4.1 (72,2 %) y demostrando un avance significativo en comparación con modelos previos como GPT-4o mini (56,5 %).

En la métrica CharXiv-R se evalúa la habilidad del modelo para inferir razonamientos basándose en gráficos científicos auténticos. En otras palabras, no es suficiente con leer datos: el modelo tiene que extraer conclusiones a partir de representaciones visuales como esquemas de líneas, barras o zonas. En esta situación, GPT-4.1 mini alcanza un 56,8 %, mostrando un desempeño similar al de GPT-4.1 (56,7 %) y superior al de GPT-4o mini (36,8 %).

Por otro lado, en CharXiv-D se analiza una destreza más técnica, la obtención de información exacta de gráficos. Aquí se evalúa la precisión con la que el modelo puede reconocer valores, etiquetas, unidades o aspectos esenciales en una visualización científica. GPT-4.1 mini alcanza el 88,4 %, lo que es uno de los resultados más destacados entre todos los modelos, incluso un poco más alto que GPT-4.1 (87,9 %) y similar a GPT-4.5 (90,0 %).

2.4.7 TIEMPO DE DESARROLLO E IMPLEMENTACIÓN

La única forma para implementar el modelo GPT-4.1-mini es mediante la API en la nube que proporciona OpenAI. Esta forma de acceso permite el envío de entradas en formato texto o imágenes (codificadas en base64) a los servidores de OpenAI, recibiendo una respuesta en tiempo real a través de peticiones HTTP. No es necesario instalar el modelo en un lugar específico ni implementarlo en infraestructura propia, lo que simplifica de manera significativa el proceso de implementación.

En entornos que ya posean capacidades para hacer peticiones a servicios web RESTful, la integración básica del modelo, que incluye la autenticación con claves API, la gestión de peticiones y la lectura de respuestas en formato JSON puede finalizarse en un periodo previsto de 1 a 2 días laborables. Este rango implica una aplicación estándar, sin adición de procesamiento ni validaciones complicadas.

2.4.8 REQUERIMIENTOS COMPUTACIONALES

El modelo GPT-4.1-mini funciona únicamente desde la infraestructura en la nube de OpenAI, lo que significa que los requerimientos computacionales locales para su utilización son mínimos. Como la gestión de peticiones se lleva a cabo totalmente en los servidores remotos de OpenAI, el sistema del usuario o desarrollador sólo requiere habilidad para enviar peticiones HTTP, gestionar respuestas en formato JSON y, en caso de procesamiento multimodal, transformar imágenes a formato base64.

2.4.9 COMPATIBILIDAD CON BALANCEO DE CARGA

El modelo GPT-4.1-mini, al funcionar únicamente mediante la API de OpenAI en la nube, es compatible con sistemas de balanceo de carga a nivel de cliente o aplicación. Ya que las peticiones se transmiten como peticiones HTTP a un punto final (endpoint) remoto proporcionado por OpenAI, los programadores tienen la capacidad de repartir estas peticiones entre diversas versiones de su aplicación cliente, ya sea en ambientes locales o en la nube, a través de estrategias habituales de balanceo como el round-robin, la distribución por carga actual o el enrutamiento geográfico.

2.4.10 SOPORTE Y DOCUMENTACIÓN

El modelo GPT-4.1-mini dispone con soporte técnico y documentación oficial directamente suministrada por OpenAI. La documentación se encuentra disponible en línea a través del sitio oficial para desarrolladores de openAI, en la que se especifican elementos técnicos fundamentales como la autenticación a través de claves API, la organización de peticiones y respuestas, los formatos de entrada permitidos (texto e imágenes), restricciones de uso, políticas de facturación, y ejemplos de código en diferentes lenguajes de programación.

Además, OpenAI proporciona una sección específica sobre el uso de modelos con capacidades de visión, que abarca el método de cálculo de los tokens para imágenes, su codificación en base64 y su vinculación con mensajes multimodales. Esta documentación se renueva con regularidad para mostrar modificaciones en las versiones del modelo, incorporación de nuevas funcionalidades y consideraciones técnicas pertinentes.

En cuanto al soporte, los usuarios tienen la posibilidad de utilizar herramientas comunitarias como los foros de discusión y la comunidad oficial de OpenAI, además de enviar peticiones de asistencia a través del centro de asistencia técnica. Aunque el acceso a soporte personalizado puede depender del tipo de suscripción contratada, la base de datos pública abarca de manera adecuada la mayoría de los casos de uso y escenarios de integración.

2.4.11 VENTAJAS

- Rendimiento competitivo en tareas visuales complejas: Adquiere elevados grados de exactitud en índices como MMMU, MathVista, CharXiv-R y CharXiv-D, demostrando habilidades sólidas de razonamiento, interpretación y obtención de datos a partir de imágenes.
- Fácil integración mediante API
- Reducción de costos y tiempos de desarrollo
- Actualizaciones continuas
- Compatibilidad con arquitecturas distribuidas

2.4.12 DESVENTAJAS

- Costo por tokens de entrada y salida relativamente alto.
- Es un modelo cerrado (no se puede modificar ni ejecutar localmente).
- Depende 100% de OpenAI y su disponibilidad de servicio.
- Puede presentar limitaciones para tareas muy específicas si no se plantea bien la consulta.

2.5 COMPARACIÓN ENTRE TECNOLOGÍAS

Tabla 4. Comparación entre tecnologías

Criterio	YOLO (You Only Look Once)	Mistral Vision (Pixtral-12B-2409)	GPT-4.1 Mini (OpenAI)
Tipo de modelo	Detector de objetos (visión por computadora)	Modelo multimodal visión + lenguaje natural	Modelo multimodal visión + lenguaje natural
Licenciamiento	GPLv2/GPLv3 / Ultralytics License	Apache 2.0 (open source)	Licencia cerrada basada en uso (API OpenAI)
Costo	Gratuito el modelo; costo por infraestructura	$0.15/M tokens (entrada/salida)	$0.40/M tokens entrada y $1.60/M tokens salida
Infraestructura requerida	GPU local o nube (entrenamiento)	GPU local o API en nube (La Plateforme)	Solo acceso a API en la nube
Facilidad de integración	Media - baja (Python, PyTorch, Colab, conocimiento para entrenamiento)	Media-Alta (API o ejecución local con configuración)	Alta (API REST JSON sencilla y bien documentada)
Requisitos computacionales	GPU ≥ 8GB VRAM, Python, cuDNN, RAM 8–32 GB	GPU potente para local; nada en uso vía API	Mínimos (solo conexión HTTP y JSON)
Balanceo de carga	Compatible con Docker, Kubernetes, TorchServe	Sí, especialmente con API	Natural en la nube (distribución cliente vía HTTP)

Exactitud (Acura)	Alta (detección de objetos en tiempo real con precisión)	Alta en comprensión contextual y semántica	Muy alta (más de 88% en benchmarks científicos visuales)
Tiempo de implementación	2–3 días instalación; 1–2 días integración; 1–2 días entrenamiento	2 días vía API / 1–2 semanas local	1–2 días vía API REST
Soporte y documentación	Amplia comunidad, GitHub, tutoriales, foros	Documentación oficial, foros, ejemplos	Extensa y actualizada por OpenAI, foros activos

Nota: La tabla presenta la comparación de las tres tecnologías presentadas anteriormente a modo de resumen. Fuente: Elaboración Propia.

2.6. SELECCIÓN DE LA TECNOLOGÍA BASE DEL PROTOTIPO

Elegir la tecnología más apropiada para crear el prototipo de reconocimiento de firmas manuscritas es una decisión esencial para asegurar la factibilidad técnica, financiera y operativa del proyecto. Tras el análisis detallado de las alternativas evaluadas —YOLO (You Only Look Once), Mistral Vision (Pixtral-12B-2409) y GPT-4.1 Mini (OpenAI) —, se establecieron unos criterios de seleccion que considere no sólo las capacidades técnicas individuales de cada alternativa, sino también su alineación específica con los objetivos del proyecto, las necesidades operativas de Eprocess Company S.A.S., y los requerimientos de implementación en una arquitectura distribuida con balanceo de carga.

2.6.1 CRITERIOS DE EVALUACIÓN CONSIDERADOS

Los criterios de evaluación se definieron a partir del análisis detallado que se presento en la sección 2.4, con el fin de establecer parámetros objetivos que permitieran comparar las tecnologías evaluadas y seleccionar la más adecuada para el desarrollo del prototipo. Estos criterios incluyen elementos, económicos y

operativos, asegurando que la decisión final esté alineada con los objetivos del proyecto y con las capacidades de implementación de Eprocess Company S.A.S.

2.6.2 VENTAJAS DE GPT-4.1 MINI FRENTE A YOLO Y MISTRAL

Del análisis comparativo realizado señala varias ventajas que posicionan a GPT-4.1 Mini como la tecnología más idónea para el desarrollo del prototipo de reconocimiento de firmas manuscritas. A continuación, se detallan sus principales fortalezas frente a YOLO y Mistral Visión:

Mayor precisión en tareas visuales complejas

Pese a que YOLO presenta un rendimiento sobresaliente en la detección de objetos en tiempo real, su enfoque se limita a la localización y clasificación, sin incluir un razonamiento semántico profundo. por su parte, Mistral Visión agrega un componente multimodal que mejora la comprensión del contexto; sin embargo, sus resultados en pruebas especializadas no logran el mismo grado de precisión en detalles sutiles. Mientras que GPT-4.1 Mini destaca en benchmarks estandarizados como MathVista (73,1 %) y CharXiv-D (88,4 %), rebasando inclusive a modelos de dimensiones más grandes. Esta precisión es particularmente importante en la identificación de firmas manuscritas, ya que la existencia de ruido, la calidad del escaneo y las diferencias en los trazos requieren un modelo con una fuerte capacidad de análisis visual y contextual.

Menor dependencia de infraestructura especializada

Para utilizar YOLO es necesario contar con una GPU que sea de alto rendimiento (≥8 GB VRAM) y entornos configurados con librerías como PyTorch y cuDNN, lo que implica inversión en hardware o contratación de servicios de cómputo en la nube. En cuanto Mistral Vision este también demanda GPU potentes y entornos técnicos especializados en su entorno local, aunque tambien brinda la posibilidad de funcionar a través de API con una carga más ligera. Por otro lado, GPT-4.1 Mini opera solamente a través de la infraestructura en la nube de OpenAI; así, el cliente

únicamente tiene que manejar solicitudes HTTP y respuestas en formato JSON. Esto hace innecesario el uso de hardware especializado, disminuye los gastos de mantenimiento y permite que instituciones sin recursos avanzados de computación puedan acceder sin mayores complicaciones.

Consideraciones de Mantenimiento y Escalabilidad

La infraestructura administrada por OpenAI garantiza actualizaciones automáticas, mantenimiento continuo y escalabilidad transparente, reduciendo la carga operativa para el equipo de desarrollo. Esta característica es particularmente relevante para Eprocess Company S.A.S., que busca una solución que minimice la complejidad técnica interna.

Rendimiento en Benchmarks Especializados, los resultados de evaluación demuestran la superioridad de GPT-4.1 Mini en tareas visuales específicas, como se detalla en 2.4.8. ACURA:

- MMMU (Comprensión Multimodal Multitarea): 72.7%, superando a GPT-4o (68.7%)
- MathVista (Razonamiento matemático visual): 73.1%, superando incluso a GPT-4.1 (72.2%)
- CharXiv-D (Extracción de datos de gráficos): 88.4%, el rendimiento más alto registrado

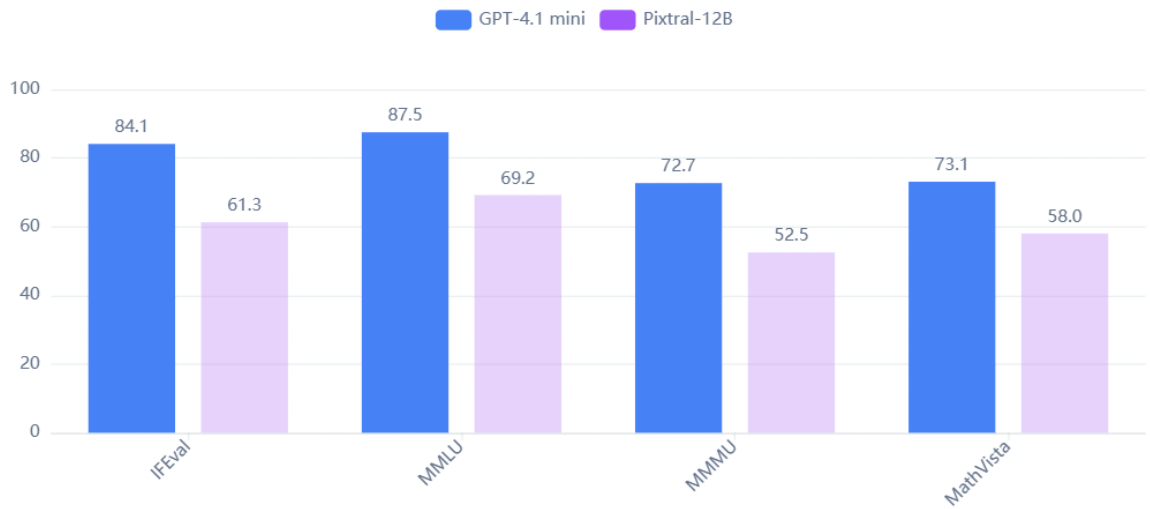
Comparación directa GPT-4.1 Mini vs. Pixtral-12B

Para emplear YOLO, se necesita obligatoriamente una GPU de alto rendimiento (≥8 GB VRAM) y entornos que estén configurados con bibliotecas como cuDNN y PyTorch. Esto significa que es necesario invertir en hardware o contratar servicios de computación en la nube. Mistral Vision, en modalidad local, requiere también de GPU potentes y entornos técnicos especializados. Sin embargo, se puede operar a través de API con una carga inferior. GPT-4.1 Mini, por otro lado, opera exclusivamente a través de la infraestructura en la nube de OpenAI; así, el cliente

solo tiene que manejar las solicitudes HTTP y las respuestas en JSON. Esto hace innecesario el uso de hardware especializado, disminuye los gastos de mantenimiento y permite que instituciones sin recursos avanzados de computación puedan acceder.

Aunque Pixtral-12B-2409 tiene un costo unitario por token reducido cuando se utiliza a través de una API, su menor exactitud genera un riesgo más elevado de cometer errores y, en consecuencia, se requiere más iteraciones para lograr resultados fiables. Esto significa que se procesan más tokens y que los costos operativos aumentan de manera indirecta. Por otro lado, la superioridad técnica de GPT-4.1 Mini posibilita que se consigan respuestas válidas con menos intentos, lo que contrarresta su costo más alto por token y lo convierte en una opción más eficaz en situaciones donde son cruciales la exactitud y la disminución de reprocesos, como sucede con Eprocess.

Figura 9. Comparación de desempeño entre GPT-4.1 Mini y Pixtral-12B en diferentes benchmarks (IFEval, MMLU, MMMU y MathVista).



Fuente: LLM-Stats.

2.6.3 FACTORES ECONÓMICOS Y DE LICENCIAMIENTO

GPT-4.1 Mini tiene un modelo de facturación basado en tokens procesados ($0.40/M tokens entrada, $1.60/M tokens salida) permite control granular de costos y escalabilidad según demanda real, evitando inversiones iniciales en infraestructura especializada, en comparacion a YOLO, que al ser de código abierto, no implica costos directos de licencia, sí requiere una inversión significativa en infraestructura de cómputo, particularmente en GPUs de alto rendimiento, así como en personal especializado para el entrenamiento y mantenimiento del modelo. Este escenario incrementa los costos ocultos asociados a su adopción.

En cuanto a Pixtral-12B-2409, si se usa vía API, el costo por token es más bajo, lo que puede indicar una rentabilidad inicial más alta. Sin embargo, los resultados de los benchmarks comparativos indican que GPT-4.1 Mini llega a un grado de exactitud más alto, incluso en comparación con modelos de mayor escala. Esto se puede ver en la figura 3: GPT-4.1 Mini supera de manera constante a Pixtral-12B en evaluaciones como MMLU, IFEval, MMMU y MathVista, con diferencias de entre el 10% y el 20%. Esta ventaja técnica se manifiesta en una eficiencia operativa superior y en una disminución de los costos ocultos generados por reprocesos, lo que es crucial para la viabilidad del prototipo en Eprocess.

En lo que respecta al licenciamiento, YOLO es gratuito y abierto bajo GPL; sin embargo, esto implica que se deben liberar los desarrollos derivados, lo cual no es favorable para Eprocess, pues necesita proteger su propiedad intelectual. Pixtral-12B-2409, bajo Apache 2.0, brinda flexibilidad en términos comerciales; sin embargo, si se utiliza de manera local, requiere que la empresa asuma la gestión técnica, lo cual eleva su carga operativa. Por otro lado, GPT-4.1 Mini funciona bajo un esquema de consumo a través de API, lo que hace innecesario lidiar con la complejidad de las licencias y asegura el soporte, las actualizaciones y el cumplimiento normativo.

2.6.4 JUSTIFICACIÓN TÉCNICA DE LA SELECCIÓN

Se escoge a GPT-4.1 Mini como tecnología fundamental para la creación del prototipo, teniendo en cuenta los elementos técnicos y de implementación que

aseguran su capacidad de sostenerse y crecer dentro de un ambiente productivo. En esta dirección, se reconocen las siguientes propiedades técnicas esenciales:

1. Compatibilidad con balanceo de carga: La posibilidad de repartir las solicitudes entre varias instancias de clientes sin tener que establecer manualmente servidores locales es una ventaja al funcionar en la nube. Esto se traduce en una capacidad de respuesta ante cargas de trabajo cambiantes y en una escalabilidad natural.

2. Escalabilidad tanto vertical como horizontal: El rendimiento no se ve afectado cuando varias aplicaciones y usuarios utilizan GPT-4.1 Mini a la vez. El escalado vertical se gestiona en la nube de OpenAI, pero el horizontal se logra mediante algoritmos de balanceo o enrutamiento que distribuyen las solicitudes a través de la capa cliente.

3. Bajo mantenimiento y simplicidad operativa: la ejecución remota elimina la necesidad de instalar, actualizar o entrenar modelos localmente. Esto reduce significativamente la carga del equipo técnico y asegura que siempre se trabaje de manera mas optimizada.

4. Seguridad integrada: El modelo funciona en la infraestructura de OpenAI, que contiene métodos avanzados para la gestión de datos y seguridad, autenticación por medio de claves API y cumplimiento de las normativas internacionales en cuanto a protección informativa.

5. Portabilidad e interoperabilidad: El uso de API REST para la integración, con entradas y salidas en JSON, posibilita la conexión con varios lenguajes de programación, frameworks web y arquitecturas distribuidas. Esto garantiza que el sistema tenga interoperabilidad con los sistemas de las empresas que ya están en funcionamiento.

6. Implementación: El modelo se puede poner en funcionamiento solo configurando las credenciales de acceso y utilizando el endpoint de la API, lo cual significa que la implementación toma entre uno y dos días, un periodo mucho más corto que el que necesitan los modelos que requieren un entrenamiento intensivo o infraestructura dedicada.

2.6.5 CONCLUSIÓN DE LA SELECCIÓN

Al comparar Pixtral-12B-2409, YOLO y GPT-4.1 Mini lleva a la conclusión de que esta última es la alternativa más apropiada para crear el prototipo en Eprocess. A pesar de que YOLO no conlleva ningún gasto en licencias, su dependencia de infraestructura especializada y la necesidad de liberar desarrollos subsecuentes bajo GPL restringen su uso en un ambiente corporativo. Aunque Pixtral-12B-2409 tiene un precio por token más bajo y puede usarse a través de API con una licencia Apache 2.0 más flexible, muestra un rendimiento técnico inferior en comparación con GPT-4.1 Mini en indicadores importantes como IFEval, MMLU, MMMU y MathVista. Esto significa que hay un riesgo mayor de cometer errores y que se requieren iteraciones extra, lo cual hace que los costos operativos reales puedan aumentar.

En cambio, GPT-4.1 Mini combina un desempeño mejor en benchmarks con una integración sencilla a través de API y un marco de consumo administrado que asigna al proveedor la responsabilidad de las actualizaciones, el soporte y la seguridad. A pesar de que su costo por unidad es mayor, la disminución de reprocesos y la fiabilidad superior de sus resultados lo hacen ser la opción más eficaz y sustentable. Para Eprocess, esto implica una implementación que es ágil, segura y escalable. No solo mejora los recursos existentes, sino que también asegura la estabilidad y continuidad del prototipo en el ámbito empresarial.

CAPÍTULO 3: DISEÑO PROTOTIPO

3.1 COMPONENTES PRINCIPALES DEL SISTEMA

El prototipo para el reconocimiento de firmas manuscritas está conformado por componentes que trabajan de manera coordinada para garantizar el procesamiento de documentos digitales con los mecanismos de seguridad, escalabilidad e integración establecidos en los objetivos del proyecto. Dichos componentes son:

Servicio de autenticación: Este servicio es responsable de proporcionar tokens JWT a los usuarios autorizados. Cuando un usuario envía sus credenciales válidas a este servicio, recibe un token JWT que posteriormente utilizará para acceder al servicio de verificación de firmas. Este componente implementa la generación de tokens según el estándar JWT.

Método de validación de token : Este verifica que el token JWT recibido en cada solicitud sea correcto, vigente y no haya sido alterado, asegurando que solo usuarios autorizados accedan al servicio.

Conversión a base 64 y validación: Este componente se encarga de preparar el documento para que pueda ser analizado. En primer lugar, asegúrate de que el archivo recibido sea uno de los formatos admitidos: PNG, JPEG o PDF. Si se recibe un archivo PDF, el módulo lo transforma a formato de imagen (PNG o JPEG), cada una de sus paginas. Este, despues codifica cada imagen final en formato Base64 para hacer más fácil su transmisión por medio de la API de OpenAI GPT-4.1 Mini, sin tener en cuenta el formato original.

Componente de conexión y procesamiento con OpenAI (GPT-4.1 Mini): Este Componente gestiona lo que es la comunicación con la API de GPT-4.1 Mini. Elaborando entonces el formato JSON según las especificaciones de OpenAI, incluyendo la imagen codificada en Base64, el prompt específico para reconocimiento de firmas manuscritas y utiliza la clave API de OpenAI para autenticar las peticiones. Una vez que se enviada la petición, este recibe y procesa la respuesta de la API, extrayendo la información relevante sobre la presencia o

ausencia de firma manuscrita, y transformándola al formato de respuesta del prototipo.

Componente de verificacion de firmas manuscritas: Este es el componente principal del prototipo donde se presenta el endpoint para validar la presencio o no de una firma manuscrita. En la solicitud, se recibe el archivo que se va a examinar (en formato JPEG, PDF o PNG) y el token JWT para poder autenticar. Este servicio organiza la ejecución de los diversos componentes de procesamiento y validación, y devuelve una respuesta en formato JSON que señala si se halló o no una firma manuscrita en el documento; si no es así, proporciona un mensaje de error descriptivo.

Infraestructura distribuida con balanceo de carga: Se asegura que el sistema sea capaz de manejar múltiples peticiones simultáneamente, repartiendo la carga entre varias instancias para garantizar la escalabilidad, la tolerancia a fallos y la disponibilidad.

3.2 FLUJO DEL PROTOTIPO

El flujo del prototipo para el reconocimiento de firmas manuscritas en documentos digitales aplicable a Eprocess Company S.A.S describe la secuencia de operaciones que se ejecutan desde que un usuario envía un documento para detectar o no la presencia de una firma manuscrita hasta que recibe la respuesta indicando la presencia o ausencia de esta. Este flujo incorpora todos los elementos que se describieron antes y se compone de dos procesos fundamentales: la verificación del usuario y el reconocimiento de firmas.

3.2.1 FLUJO DE AUTENTICACIÓN

Para que un usuario tenga acceso al servicio de verificación de firmas, el primer paso que debe seguir es el proceso de autenticación. Este flujo se lleva a cabo de la siguiente forma:

- Envío de credenciales: El usuario envía una petición, HTTP POST el cual seria el servicio de autenticación donde este debe incluir el nombre de usuario y contraseña en el cuerpo de la petición en formato JSON.
- Verificación credenciales: Se comparan las credenciales introducidas por el usuario con las configuradas en código para determinar si estas son válidas o no.
- Generación de token JWT: En caso de que las credenciales sean válidas se crea un token JWT que contiene datos del usuario y un tiempo de caducidad establecido.
- Respuesta al usuario: El servicio devuelve una respuesta HTTP 200 que incluye el token JWT, presentado en formato JSON. Este token debe ser añadido en todas las solicitudes que se realicen al servicio de reconocimiento de firmas.
- Manejo de errores: Si las credenciales no son válidas, el servicio entonces devuelve un mensaje explicativo que señala que la autenticación no fue exitosa y un código de error HTTP 401 (Unauthorized).

3.2.2. FLUJO DE IDENTIFICACIÓN DE FIRMAS MANUSCRITAS

Una vez que el usuario ha obtenido un token JWT válido, puede proceder a enviar el documento para su análisis. El flujo de identificación de firma manuscrita ejecuta de la siguiente manera:

- Recepción de la petición: El balanceador de carga recibe la petición HTTP POST del usuario y la distribuye a una de las instancias disponibles del servicio de verificación de firmas.
- Validación del token JWT: La instancia seleccionada invoca el método de validación de token para verificar que el JWT incluido en los headers de la petición sea válido, no haya expirado y no haya sido alterado. Si el token no es válido, el proceso se detiene y se retorna un error HTTP 401 con el mensaje correspondiente.
- Recepción del documento: Si el token es válido, el servicio extrae el archivo enviado por el usuario desde el cuerpo de la petición (multipart/form-data).

- Validación del formato: El módulo de validación y procesamiento de archivos verifica que el documento recibido corresponda a uno de los formatos soportados: PDF, JPEG o PNG. Si el formato no es válido, se retorna un error HTTP 400 (Bad Request) indicando que el formato no es soportado.
- Conversión de PDF a imagen: Si el archivo es PDF, el módulo extrae el número total de páginas y procede a convertir cada página individualmente a formato de imagen (PNG o JPEG). Para archivos JPEG o PNG, este paso se omite y se procesa directamente la imagen única.
- Procesamiento iterativo por página: Para documentos PDF con múltiples páginas, el sistema procesa cada página de manera secuencial. Para cada página (o imagen única en caso de JPEG/PNG):
- Codificación en Base64: El módulo codifica la imagen resultante en formato Base64, preparándola para su transmisión a la API de OpenAI.
- Construcción de la petición a OpenAI: El módulo de integración con OpenAI construye la petición HTTP en formato JSON incluyendo:
    - La imagen codificada en Base64
    - El prompt específico para reconocimiento de firmas manuscritas
    - Los parámetros de configuración del modelo
    - La clave API en el header de autorización (Bearer Token)
- Envío a la API de GPT-4.1 Mini: El módulo envía la petición HTTPS a los servidores de OpenAI y espera la respuesta.
- Procesamiento de la respuesta de OpenAI: Una vez recibida la respuesta de cada página analizada, el módulo extrae la información relevante sobre la detección de firma manuscrita. Si se detecta una firma en alguna página, el procesamiento de páginas restantes se detiene para optimizar el consumo de tokens. Si no se encuentra firma, se continúa con la siguiente página hasta procesar todas las páginas del documento.
- Estructuración de la respuesta: El servicio de verificación de firmas estructura la respuesta final en formato JSON, incluyendo:
    - hasSignature: Indicador booleano de presencia de firma; true/false
    - Detalle: No/Si hay presencia de firma manuscrita, Token vencido..
    - Estado: Satisfactorio, Error

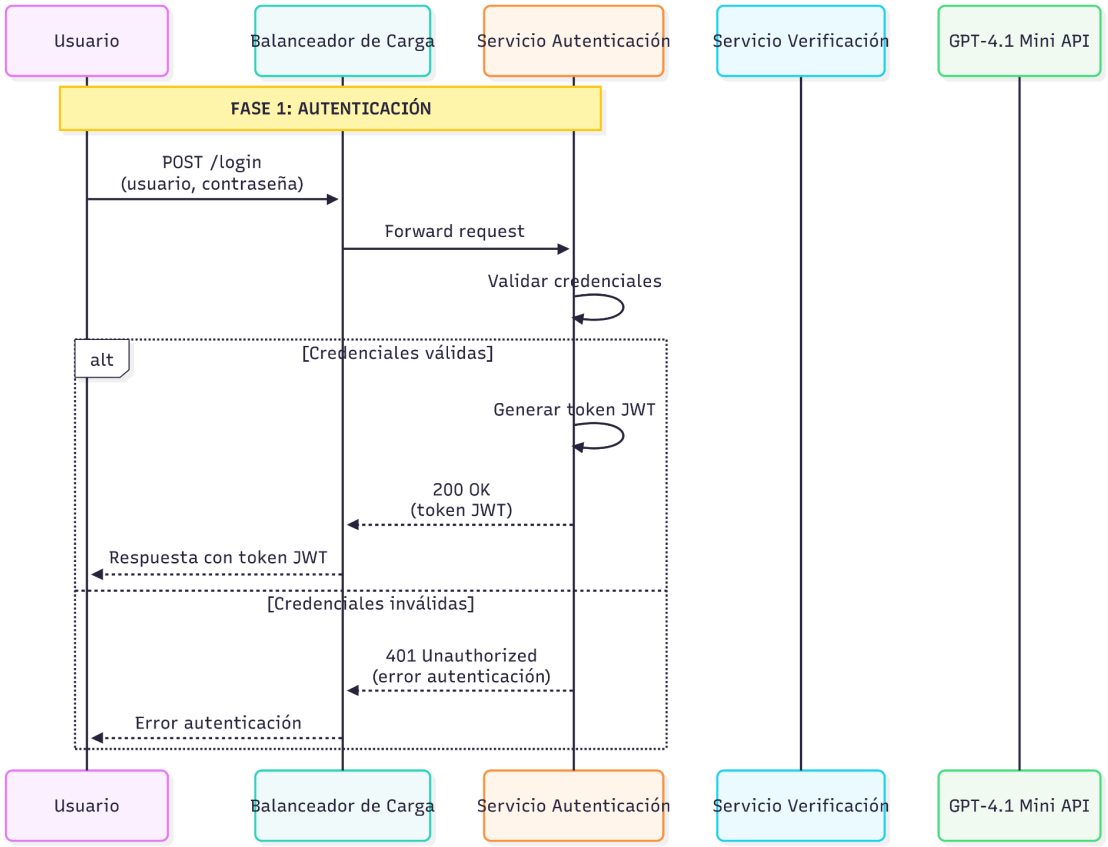
- Retorno al usuario: La respuesta JSON se envía de vuelta al usuario a través del balanceador de carga, completando el ciclo de procesamiento.
- Manejo de errores: En caso de que ocurra algún error durante el procesamiento (error de comunicación con OpenAI, timeout, formato inválido, etc.), el servicio captura la excepción y retorna una respuesta de error estructurada en el parámetro de respuesta; Detalle, junto con el código de estado; Error.

3.3 DIAGRAMAS DE SECUENCIA

Las Figuras 1, 2 y 3 representan gráficamente la secuencia de interacciones entre los diferentes componentes del sistema; la autenticación , verificación de firma y manejo de errores en comunicación.

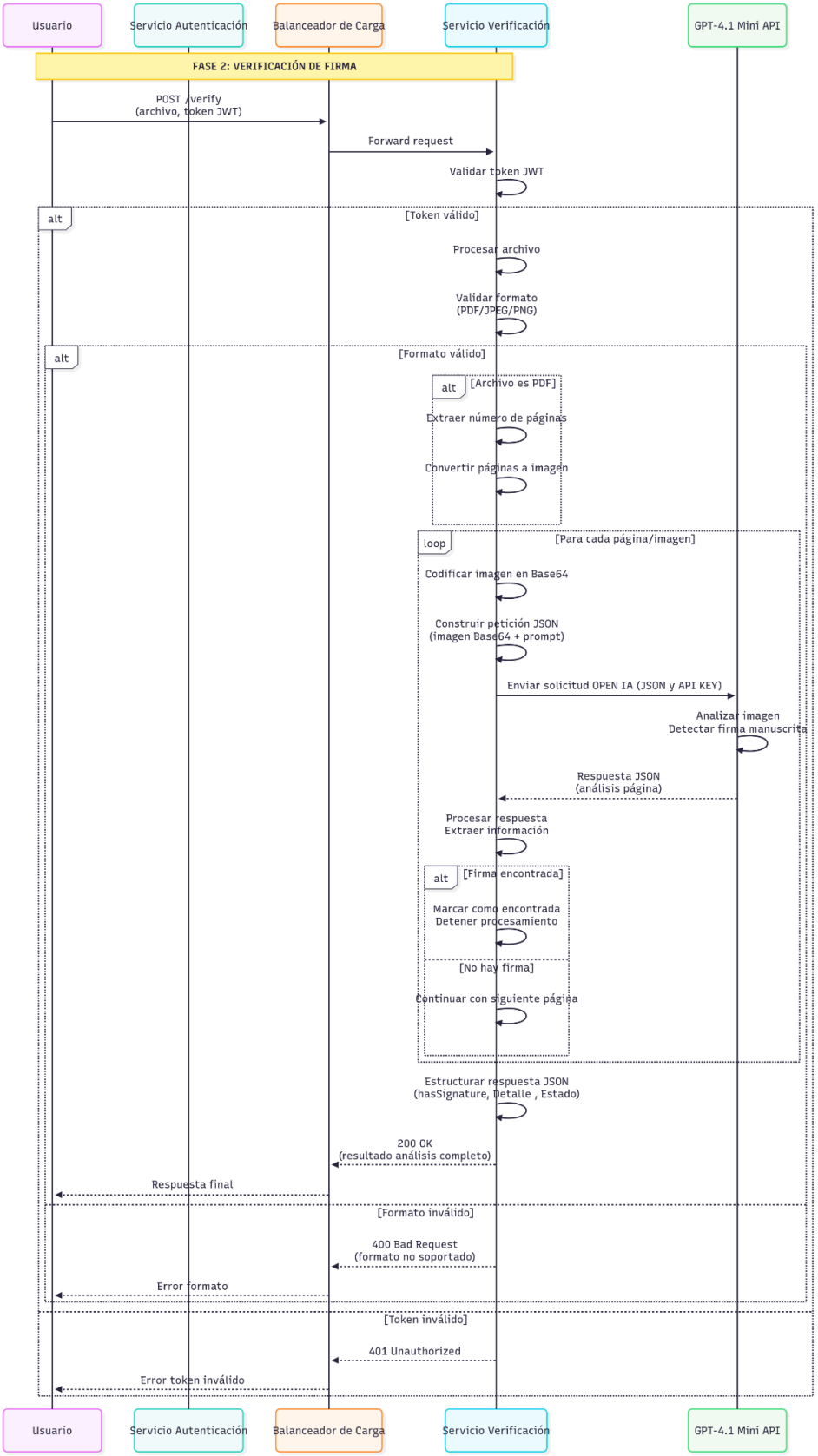
Con dichos flujos se garantiza que cada documento sea procesado de manera segura, eficiente y confiable, cumpliendo con los objetivos establecidos para el prototipo y proporcionando a Eprocess Company S.A.S. Una solución automatizada para el reconocimiento de firmas manuscritas en documentos digitales.

Figura 10. Diagrama de secuencia del flujo de procesamiento de autenticación



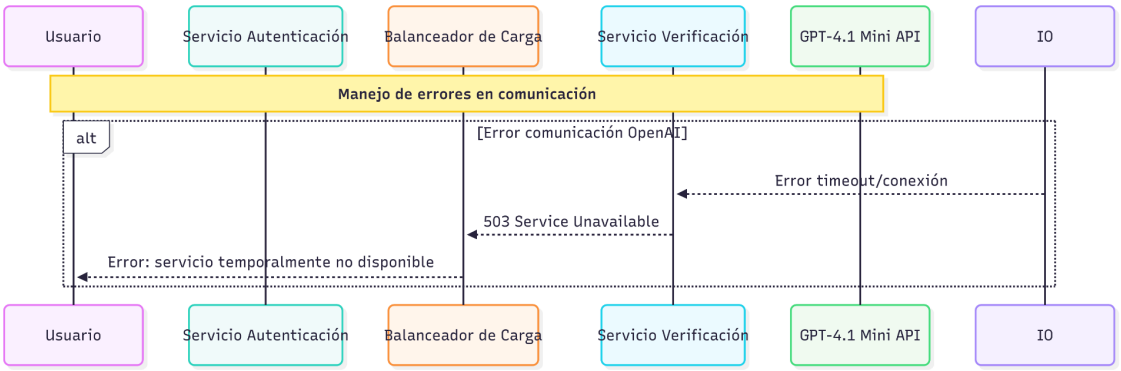
Fuente: Elaboración propia.

Figura 11. Diagrama de secuencia del flujo de verificación de firma



Fuente: Elaboración propia.

Figura 12. Diagrama de secuencia del flujo del proceso de manejo de errores en comunicación.



Fuente: Elaboración propia.

3.4 GPT-4.1 MINI

Lo primero que se debe hacer para lograr una comunicación entre el modelo GPT-4.1 Mini y el prototipo es configurar la API de OpenAI, este método asegura que la infraestructura en la nube de OpenAI sea capaz de autenticar y gestionar correctamente las peticiones enviadas.

El primer paso es conseguir la clave de API, que es un identificador único que permite acceder a los servicios de la plataforma. Esta clave tiene que ser creada en la cuenta oficial del creador en el sitio web de OpenAI.[55].

Las solicitudes a la API se organizan de acuerdo con el formato JSON que OpenAI estableció para modelos multimodales Cada solicitud tiene que contener:

- Identificador del modelo a utilizar (gpt-4.1-mini)

[55]OPENAI. Vision: Image inputs. En: OpenAI Platform Documentation. [Documentación en línea]. [Localidad no especificada]: OpenAI, 2024. [citado el 1 de septiembre de 2025]. Disponible en: https://platform.openai.com/docs/guides/vision

- Rol del mensaje (user)
- Contenido multimodal que combina
- Texto con las instrucciones de análisis
- Imagen codificada en Base64 con prefijo de tipo MIME

La biblioteca oficial openai de Python proporciona el cliente necesario para construir y enviar estas peticiones de manera estructurada, abstrayendo la complejidad del protocolo HTTP subyacente.

3.5 MANEJO DE IMÁGENES CODIFICADAS EN BASE64

El proceso de codificar las imágenes en base64 representa una etapa importante, la cual se debe realizar antes del consumo de la API de OpenAI, entonces, lo que se logra mediante está codificación es convertir la información binaria de la imagen a una representación textual utilizando un algoritmo que agrupa los bytes en bloques de 3 (24 bits) y los transforma en 4 caracteres ASCII de 6 bits cada uno[56]. La biblioteca estándar base64 de Python simplifica está tarea, ya que ofrece la función b64encode la cual transforma los bytes extraídos del archivo imagen a su representación en Base64 para luego decodificar a UTF-8 usando el método .decode("utf-8") para así conseguir una cadena de texto manipulable[57].

La secuencia Base64 que surge tras codificar la imagen tiene que ser adaptada de acuerdo con el esquema de URI de datos (data URI scheme) que especifica el RFC 2397.[58] Este formato necesita, según el tipo de imagen, añadir primero el prefijo [datos_codificados], es la que finalmente

---

[56] JOSEFSSON, S. The Base16, Base32, and Base64 Data Encodings. RFC 4648 [en línea]. Internet Engineering Task Force (IETF), octubre 2006. [Consultado: 16 septiembre 2025]. Disponible en: https://www.rfc-editor.org/rfc/rfc4648

[57] PYTHON SOFTWARE FOUNDATION. base64 — Base16, Base32, Base64, Base85 Data Encodings. En: Python Documentation [en línea]. 2024. [Consultado: 16 septiembre 2025]. Disponible en: https://docs.python.org/3/library/base64.html

[58] MASINTER, L. The "data" URL scheme. RFC 2397 [en línea]. Internet Engineering Task Force (IETF), agosto 1998. [Consultado: 16 septiembre 2025]. Disponible en: https://www.rfc-editor.org/rfc/rfc2397

se incluye en el campo image_url de la petición JSON enviada a la API, permitiendo que OpenAI decodifique y procese correctamente la imagen para su análisis visual.

3.6 ENDPOINTS

El servicio web expone dos endpoints principales que gestionan las funcionalidades de autenticación y verificación de firmas: Endpoint de autenticación: /api/login

- Método HTTP: POST
- Propósito: Autenticar usuarios y generar tokens JWT
- Ruta: http://[host]:[puerto]/api/login
- Formato de entrada: form-data
- Formato de salida: JSON (application/json)

Este endpoint recibe las credenciales del usuario (nombre de usuario y contraseña) y, tras validarlas, genera un token JWT que debe ser incluido en las peticiones posteriores al servicio de verificación de firmas. Endpoint de verificación de firmas: /api/verificarFirma

- Método HTTP: POST
- Propósito: Analizar documentos digitales y detectar firmas manuscritas
- Ruta: http://[host]:[puerto]/api/verificarFirma
- Formato de entrada: Multipart/form-data
- Formato de salida: JSON (application/json)
- Autenticación: Requerida mediante token JWT en header Authorization

Este endpoint recibe el documento a analizar como archivo adjunto y retorna una respuesta estructurada indicando la presencia o ausencia de firma manuscrita. En el encabezado de autorización, la solicitud tiene que incluir el token JWT adquirido con anterioridad en este formato: Authorization: Bearer [token].

3.7 PROCESAMIENTO DE DOCUMENTOS DIGITALES (PDF, JPEG, PNG)

El procesamiento de documentos digitales se diseña para manejar los tres formatos más comunes: PDF, JPEG y PNG. Antes de su envío a la API de OpenAI, cada formato necesita un procesamiento particular.

Procesamiento de imágenes JPEG y PNG: Para documentos en formato de imagen (JPEG o PNG), el procesamiento es directo, es decir; se valida que el archivo corresponda a uno de estos formatos mediante la verificación de la extensión, una vez validado, la imagen se codifica directamente en Base64 y se procesa mediante la API de OpenAI como una única consulta.

Procesamiento de documentos PDF: A causa de su condición de multipágina, los documentos en formato PDF necesitan un procesamiento extra. Para la gestión de archivos PDF, el sistema emplea bibliotecas especializadas de Python como PyMuPDF o PyPDF2, que posibilitan obtener información acerca del número de páginas y transformar cada página por separado en formato imagen, el flujo seria :

- Validación del formato y estructura del archivo PDF
- Conversión iterativa de cada página a formato de imagen (PNG o JPEG)
- Codificación de cada imagen resultante en Base64
- Análisis secuencial mediante la API de OpenAI
- Optimización mediante detección temprana: si se detecta firma en alguna página, se suspende el análisis de páginas restantes

Esta estrategia de procesamiento iterativo con detención temprana minimiza el consumo de tokens y optimiza los tiempos de respuesta, especialmente relevante para documentos extensos donde la firma suele ubicarse en las páginas iniciales o finales.

3.8 Autenticación JWT

El método de autorización y autenticación del prototipo se fundamenta en los Tokens Web JSON (JWT), una norma abierta dispuesta en el RFC 7519 que permite la transferencia de información entre entidades de manera segura por medio de un

objeto JSON firmado digitalmente[59]. JWT es un método autónomo y compacto para expresar declaraciones que son emitidas entre dos partes; se estructura como una cadena, la cual puede ser trasladada fácilmente a través de encabezados HTTP. Se elige JWT porque es stateless, lo que posibilita no tener que conservar sesiones en el servidor y facilita la puesta en marcha de arquitecturas distribuidas con balanceo de carga. Esto es así porque cualquier instancia del servicio puede validar el token sin necesidad de compartir estado con otras instancias[60]. Los JWT son compactos y autónomos, lo que los hace perfectos para sistemas distribuidos y arquitecturas multiservicio, según BBVA Labs & BBVA, esto mejora la experiencia del usuario y el rendimiento, además de robustecer la seguridad global de la solución[61].

Un token JWT está compuesto de tres secciones, las cuales están separadas entre sí por puntos: la carga útil (payload), el encabezado (header) y la firma (signature); todas ellas han sido codificadas a través de Base64URL. El encabezado señala la clase de token y el algoritmo de firma utilizado, el cual suele ser HS256 (HMAC con SHA-256) para el prototipo. La carga útil contiene afirmaciones (claims) sobre el usuario, por ejemplo, el identificador (sub), la hora de emisión (iat) y la fecha de caducidad (exp), que se define en una hora. La creación de la firma se realiza al implementar el algoritmo en el encabezado y en la carga útil codificados, utilizando una clave secreta que solo conocen los servidores: HMACSHA256(codificarBase64Url(header) + ".") + base64UrlEncode(payload), secret) [62]. Esta firma entonces garantiza la integridad del token, ya que cualquier modificación resultaría en una firma diferente que no coincidiría con la original.

El proceso de autenticación consta de dos fases, en primer lugar, el cliente manda sus credenciales a la ruta /api/auth/login; si estas son válidas, se genera un token JWT que incluye la información del usuario y se lo regresa al cliente, en segundo lugar, para acceder al endpoint protegido /api/verificarFirma, el cliente debe incluir el

---

[59] IETF (Internet Engineering Task Force). RFC 7519: JSON Web Token (JWT). [Norma en línea]. [Localidad no especificada]: IETF, 2015. [citado el 30 de septiembre de 2025]. Disponible en: https://www.rfc-editor.org/rfc/rfc7519

[60] Ibid.

[61] BBVA LABS; BBVA, op. cit.

[62] JONES, M. JSON Web Signature (JWS). RFC 7515 [en línea]. Internet Engineering Task Force (IETF), mayo 2015. [Consultado: 16 septiembre 2025]. Disponible en: https://www.rfc-editor.org/rfc/rfc7515

token en la cabecera Authorization: Portador [token]. En consecuencia, el servicio comprueba:

- La firma con la clave secreta
- El formato del token
- Que este no esté caducado al cotejar el timestamp actual con el claim exp
- Obtiene los datos del usuario. Si alguna validación no tiene éxito, se devuelve un código HTTP 401 (No autorizado).

si todas tienen éxito, la solicitud se procesa de forma habitual. Este sistema asegura que solo los usuarios validados tengan acceso al servicio, lo que protege el recurso y posibilita rastrear las operaciones.

3.9 DISEÑO DE LA ARQUITECTURA DE BALANCEO DE CARGA

La arquitectura de balanceo de carga del prototipo está diseñada para repartir las solicitudes HTTP que llegan entre diversas instancias del servicio Flask, lo que asegura una gran disponibilidad, tolerancia a errores y la capacidad de escalar horizontalmente. La propuesta se basa en la interacción de tres elementos tecnológicos fundamentales: el equilibrador NGINX, el servidor de aplicaciones Flask administrado a través de Gunicorn y la plataforma para el despliegue Render. La elección de estos elementos se debió a su estabilidad, su compatibilidad y su amplia implementación en los ambientes de producción actuales.

3.9.1 NGINX

NGINX es un servidor web y proxy inverso de alto rendimiento, concebido para manejar una cantidad elevada de conexiones simultáneas utilizando el menor número posible de recursos. Igor Sysoev lo creó en 2004 y se ha establecido como una de las soluciones más empleadas para equilibrar la carga, hacer proxy y acelerar el contenido web[63].

[63] NGINX INC. HTTP Load Balancing. En: NGINX Documentation [en línea]. 2024. [Consultado: 2 octubre 2025]. Disponible en: https://docs.nginx.com/nginx/admin-guide/load-balancer/http-load-balancer/

NGINX, según el diseño propuesto, tiene la responsabilidad de recibir las solicitudes que los usuarios envían y distribuirlas equitativamente entre las distintas instancias del servicio Flask ubicadas en el bloque upstream. El principio del mecanismo de distribución es el algoritmo Round Robin, que asigna a un servidor diferente cada petición de manera secuencial y regresa al primero tras finalizar el ciclo. Este método facilita el uso balanceado de los recursos existentes y permite que el tráfico se reparta de forma equitativa[64]. La elección de NGINX se debió a su capacidad de concurrencia elevada, su escaso uso de memoria, su soporte nativo para balanceo de carga y proxy inverso, además que ofrece una facilidad de configuración mediante archivos declarativos y permite tambien la escalabilidad horizontal del sistema sin requerir modificaciones en la aplicación base.

3.9.2 FLASK Y GUNICORN

Flask es un microframework para el desarrollo web que se basa en Python y está orientado a la simplicidad y a la posibilidad de ampliación. Su diseño modular lo hace adecuado para la creación de servicios web RESTful. Flask es el componente lógico central de la aplicación en el diseño, ya que se ocupa de gestionar las solicitudes HTTP y elaborar las respuestas adecuadas. Gunicorn (Green Unicorn), para su parte, es un servidor de aplicaciones WSGI (Web Server Gateway Interface) que opera en sistemas Unix y se encarga de administrar varios procesos simultáneos que llevan a cabo la aplicación Flask. Gunicorn funciona como un intermediario entre Flask (nivel de aplicación) y NGINX (nivel de red), posibilitando la interacción eficaz entre los dos[65].

En el prototipo Gunicorn es el que supervisa cada instancia de Flask, y a través de él se levantan varios trabajadores que responden las solicitudes simultáneamente. Cuando NGINX obtiene una petición, esta es reenviada a alguna de las instancias que son administradas por Gunicorn. Cada worker procesa la solicitud y lleva a cabo las tareas de validación o análisis de documentos, devolviendo la respuesta al balanceador, entonces, Gunicorn posibilita que varios procesos trabajen al mismo

---

[64] Ibid.
[65] GUNICORN. Design. En: Gunicorn Documentation [en línea]. 2024. [Consultado: 2 octubre 2025]. Disponible en: https://docs.gunicorn.org/en/stable/design.html

tiempo, lo cual mejora el rendimiento y disminuye la latencia en ambientes con alta concurrencia.

3.9.3 PLATAFORMA DE DESPLIEGUE RENDER

Render es una plataforma en la nube que brinda servicios de PaaS, plataforma como servicio e IaaS, infraestructura como servicio. En el diseño prototipo, Render funciona como una plataforma de orquestación y es responsable de llevar a cabo las instancias Flask/Gunicorn y el balanceador NGINX. Permite la implementación automatizada de aplicaciones web, administrando entornos virtualizados que incluyen bases de datos, balanceadores de carga y servidores en un esquema de contenedores aislados. La infraestructura de Render posibilita la gestión de procesos en segundo plano, el establecimiento de variables de entorno y la ampliación dinámica de los servicios en función del número de usuarios. De este modo, la plataforma no solo aloja la aplicación, sino que además supervisa el estado de las instancias y reinicia automáticamente cualquier proceso que se detiene, lo cual ayuda a garantizar que el sistema esté siempre disponible[66].

CAPÍTULO 4: IMPLEMENTACIÓN Y DESARROLLO DEL PROTOTIPO

El capítulo detalla la ejecución del proceso de implementación del prototipo de verificación de firmas manuscritas, fundamentado en el diseño propuesto previamente. La implementación se centró en lograr las funciones del sistema planteadas, garantizando un correcto funcionamiento, la interacción entre módulos definidos y un intercambio de datos ágil con la API de IA de OpenAI.

4.1 CONFIGURACIÓN DEL ENTORNO DE DESARROLLO

Se utilizó Visual Studio Code (VS Code) como IDE (entorno de desarrollo integrado) y se configuró el entorno de desarrollo en un sistema operativo Windows 11. El proyecto se desarrolló en un ambiente virtual de Python, generado con la

[66] RENDER. Documentation. En: Render Docs [en línea]. 2024. [Consultado: 2 octubre 2025]. Disponible en: https://render.com/docs

herramienta venv. De esta manera, las dependencias pudieron permanecer aisladas y no hubo problemas con otras instalaciones del sistema.

Cuando se ejecutó el archivo principal proyectoFinal.py, el servidor Flask comenzó en modo depuración (debug=True), lo que posibilitó ver los mensajes de ejecución, las solicitudes registradas y las posibles excepciones durante la etapa de pruebas en la consola.

La estructura final del ambiente laboral consistió en:

- Un entorno virtual con las librerías instaladas mediante pip.
- El archivo principal proyectoFinal.py, que contiene toda la lógica del servicio.

4.2 TECNOLOGÍAS Y HERRAMIENTAS UTILIZADAS

El sistema fue creado en Python 3.11, debido a su flexibilidad y su extenso apoyo para bibliotecas de procesamiento de imágenes, gestión de documentos y comunicación con APIs externas. Python posibilita el desarrollo veloz y la conservación de un código comprensible, lo que lo hace perfecto para proyectos de investigación y prototipado rápido[67]. Utilizando además el microframework Flask, que es una herramienta liviana que facilita la creación de servicios REST y aplicaciones web. Flask facilita la integración con bibliotecas externas, el manejo de las solicitudes HTTP y la especificación de puntos finales, sin exigir que el proyecto esté estructurado de manera inflexible[68].

Se llevó a cabo la integración con el modelo de inteligencia artificial GPT-4.1 a través del cliente oficial de OpenAI (openai), que fue configurado directamente en el código fuente para hacer más sencillo el ambiente de ejecución y no recurrir a archivos externos de configuración.

---

[67] VAN ROSSUM, G.; DRAKE, F. The Python Language Reference Manual. [Manual técnico]. [Localidad no especificada]: Python Software Foundation, 2023. [citado el 30 de septiembre de 2025]
[68] PYTHON SOFTWARE FOUNDATION. Flask Documentation. [Documentación en línea]. [Localidad no especificada]: Python Software Foundation, 2024. [citado el 1 de octubre de 2025]. Disponible en: https://flask.palletsprojects.com

Adicionalmente, se emplearon las siguientes bibliotecas adicionales:

- Flask-CORS: posibilita activar las peticiones de origen cruzado, lo que asegura la comunicación con clientes web o aplicaciones externas.
- PyJWT: Se utiliza para crear y verificar los tokens de JSON Web Token (JWT), lo que asegura la autenticidad de las peticiones.
- PyMuPDF (fitz): Se encarga de la conversión y lectura de documentos PDF a imágenes, manteniendo la calidad visual del archivo.
- Pillow (PIL): biblioteca que simplifica la transformación y el manejo de formatos gráficos, permitiendo manipular imágenes.
- Base64 y BytesIO: Se utilizan para transmitir y codificar imágenes en formato de texto, lo cual es necesario para enviar datos a la API de OpenAI.
- OpenAI (cliente oficial): utilizada para conectarse con el modelo GPT-4.1 y enviar las imágenes codificadas junto con el texto de instrucción.

Se estableció el ambiente de desarrollo en un sistema operativo Windows 11, empleando GitHub como instrumentos para el control de versiones y la protección del código fuente.

4.3 IMPLEMENTACIÓN DEL SERVICIO DE AUTENTICACIÓN

Para efectuar el servicio de autenticacion se debe ejecutar el endpoint /api/login enviando las credenciales de usuario por medio de una solicitud POST en el formato application/x-www-form-urlencoded. Cuando el sistema recibe una contraseña y un nombre de usuario, verifica las credenciales en relación con unos valores previamente definidos en el código (username: "username", password: "password"). Se crea un token JWT con la biblioteca PyJWT y se firma usando el algoritmo HS256 y una clave secreta (SECRET_KEY = "EprocessCompany")². Esto sucede si las credenciales son válidas. El token creado contiene en su payload el nombre de usuario y una duración de 20 minutos desde que se expidió, que se determina a través de datetime.utcnow() + timedelta(minutes=20).

Se desarrolló un decorador llamado @token_required para salvaguardar los endpoints que requieren autenticación. Este se interpone en las solicitudes antes de ejecutar la función del endpoint. Este decorador obtiene el token del encabezado de autorización, que debe tener el formato "Bearer [token]", lo descodifica con la clave secreta de la aplicación y comprueba que no haya caducado. Permite que la solicitud prosiga si el token es válido; en caso contrario, devuelve un mensaje descriptivo con un error HTTP 401. Este sistema asegura que solo los usuarios validados tengan acceso al endpoint de verificación de firmas, cumpliendo así con las normas de seguridad definidas en la creación del prototipo.

4.4 PROCESAMIENTO DE DOCUMENTOS DIGITALES

El procesamiento de documentos digitales se implementó para manejar tres formatos: PDF, JPEG y PNG. Cuando se recibe un archivo a través del endpoint, el sistema primero verifica que la extensión del archivo sea una de las extensiones permitidas en el conjunto de extensiones válidas: {"pdf", "png", "jpg", "jpeg"}. La función pdf_to_images_in_memory(), que emplea PyMuPDF para abrir el documento en memoria sin necesidad de generar archivos temporales en el disco, necesita un procesamiento específico para los documentos PDF. Esta función recorre cada página del PDF, mejora la calidad al duplicar la resolución a través de page.get_pixmap(matrix=fitz.Matrix(2, 2)), y transforma los píxeles en un objeto Image de Pillow utilizando Image.frombytes()[69]. Como resultado, se obtiene una lista de imágenes PIL, una para cada hoja del documento.

Para todos los tipos de archivo, las imágenes se convierten a formato Base64 mediante la función image_to_base64(), que guarda la imagen en un buffer de memoria (BytesIO) en formato PNG, codifica los bytes resultantes con base64.b64encode(), y los decodifica a una cadena UTF-8. Esta representación en Base64 es el formato requerido por la API de OpenAI para el procesamiento de imágenes. En el caso de archivos JPEG o PNG recibidos directamente, se abren con Pillow mediante Image.open(BytesIO(file_bytes)) y se procesan individualmente.

---

[69] ARTIFEX SOFTWARE. PyMuPDF Documentation. [Documentación en línea]. [Localidad no especificada]: PyMuPDF, 2024. [citado el 1 de octubre de 2025]. Disponible en: https://pymupdf.readthedocs.io/

El procesamiento en memoria sin archivos temporales mejora el rendimiento y la seguridad del sistema, evitando problemas de concurrencia y eliminando la necesidad de limpiar archivos residuales.

4.5 INTEGRACIÓN CON OPENAI GPT-4.1

Se puso en funcionamiento la integración con OpenAI a través de la inicialización del cliente oficial con la clave API proporcionada: client = OpenAI(api_key="..."). La función procesar_imagen_base64() incluye la lógica de comunicación con el modelo GPT-4.1. Esta función genera una petición multimodal que combina instrucciones en texto y una imagen codificada en Base64. La estructura de la solicitud se adecúa al formato que ha determinado OpenAI, que comprende el modelo a utilizar (gpt-4.1), la función del mensaje (usuario) y dos elementos en el contenido: un aviso textual que solicita al modelo comprobar si la imagen tiene una firma manuscrita o digital, y la imagen codificada con el prefijo de tipo MIME data:image/jpeg;base64[70].

El sistema normaliza a minúsculas la respuesta del modelo, utilizando response.output_text.strip().lower(), para hacer más fácil la comparación. Se considera que hay firma si la respuesta incluye las palabras "sí" o "si". La función aplica un manejo sólido de errores, capturando excepciones concretas como los errores 401 (clave API inválida), 429 (límite de solicitudes superado) y timeouts de conexión; para cada situación, retorna mensajes descriptivos. Esta implementación posibilita que el sistema siga funcionando a pesar de que se presenten errores de comunicación con OpenAI, brindando al cliente información valiosa acerca del problema.

4.6 ENDPOINT DE VERIFICACIÓN DE FIRMAS

El flujo integral de verificación de firmas manuscritas se ejecuta a través del endpoint principal /api/verificarFirma, que está protegido por el decorador @token_required con el propósito de asegurar que solo los usuarios autenticados tengan acceso. Cuando el sistema recibe una solicitud, comprueba que el archivo

[70] OPENAI, Vision: Image inputs. op. cit

esté en el campo de archivo del formulario multipart, lo extrae y revisa su extensión. Después, sigue con el proceso apropiado según la clase de archivo. El sistema, para los documentos PDF, repasa cada página convertida a imagen de manera secuencial hasta que encuentra una firma o termina con todas las páginas. Cuando el sistema detecta una firma en alguna de las páginas, se detiene la ejecución de inmediato para hacer un uso más eficiente de los tokens de OpenAI y devuelve un mensaje que señala en qué página fue hallada la firma[71].

El procedimiento para archivos de imagen directa (PNG o JPEG) es más sencillo: se abre la imagen y se codifica en Base64, enviándose solamente una solicitud a OpenAI. El endpoint devuelve respuestas organizadas en formato JSON con códigos de estado HTTP adecuados, el codigo 200 para una gestión satisfactoria (con o sin firma identificada), 400 para errores de validación (archivo ausente o extensión prohibida), 401 para errores relacionados con la autenticación (administrados por el decorador) y 500 para fallos internos durante la gestión. Cada mensaje de error proporciona información precisa que facilita el diagnóstico de los problemas. La gestión de excepciones a través de bloques try-except asegura que el servicio no se detenga repentinamente frente a errores imprevistos, lo cual permite preservar la estabilidad del sistema.

CAPÍTULO 5: PRUEBAS PROTOTIPO

5.1 METODOLOGÍA DE PRUEBAS

Para validar el correcto funcionamiento del prototipo desarrollado, se implementó una estrategia de pruebas que combinó diferentes enfoques, permitiendo evaluar el comportamiento del sistema, estas pruebas fueron:

1. Funcionales: cuyo objetivo era validar el correcto desempeño del servicio de autenticación que produce tokens JWT y el endpoint para verificar firmas con varios formatos de archivo (JPEG, PNG, PDF) que se generara respuestas

[71] OPENAI, Vision: Image inputs. op. cit

coherentes según la presencia o ausencia de firmas y manejara apropiadamente los errores en las solicitudes.

2. Seguridad: para garantizar que los mecanismos de autenticación protegen los recursos del sistema apropiadamente, negando accesos no autorizados, tokens inválidos y expirados.

3. Rendimiento: Realizadas con un enfoque exploratorio, con la finalidad de analizar los tiempos de respuesta del servicio al procesar documentos con diferentes tamaños y número de páginas.

Para llevar a cabo las pruebas, se empleó Postman como herramienta principal. Esta facilita la creación de solicitudes HTTP personalizadas, la administración de colecciones de casos de prueba agrupadas por funcionalidad y el análisis de respuestas en formato estructurado, lo que simplifica la verificación de resultados. A través de la consola de comandos se supervisan en tiempo real los registros del servidor Flask para ver el procesamiento interno que se realiza con cada solicitud, como la validación de tokens, la conversión de archivos PDF a imágenes, la codificación Base64 y la comunicación con OpenAI. Las capturas de pantalla documentadas en los apartados subsiguientes se adquirieron directamente de Postman al momento de ejecutar cada caso de prueba. En ellas se pueden observar las solicitudes completas, que incluyen sus parámetros, encabezados de autenticación y las respuestas JSON enviadas por el servidor.

5.2 PRUEBAS FUNCIONALES

Las pruebas funcionales validan que cada parte del prototipo realiza de manera adecuada las funciones que les corresponde. Se probaron entonces los dos endpoints principales del sistema, el servicio de verificación de firmas: /api/verificarFirma que analiza documentos digitales e imágenes para identificar firmas manuscritas, y el servicio de autenticación /api/login que produce tokens JWT después de confirmar las credenciales del usuario. Con el fin de comprobar que el prototipo responde adecuadamente en todos los casos, se crearon entonces para

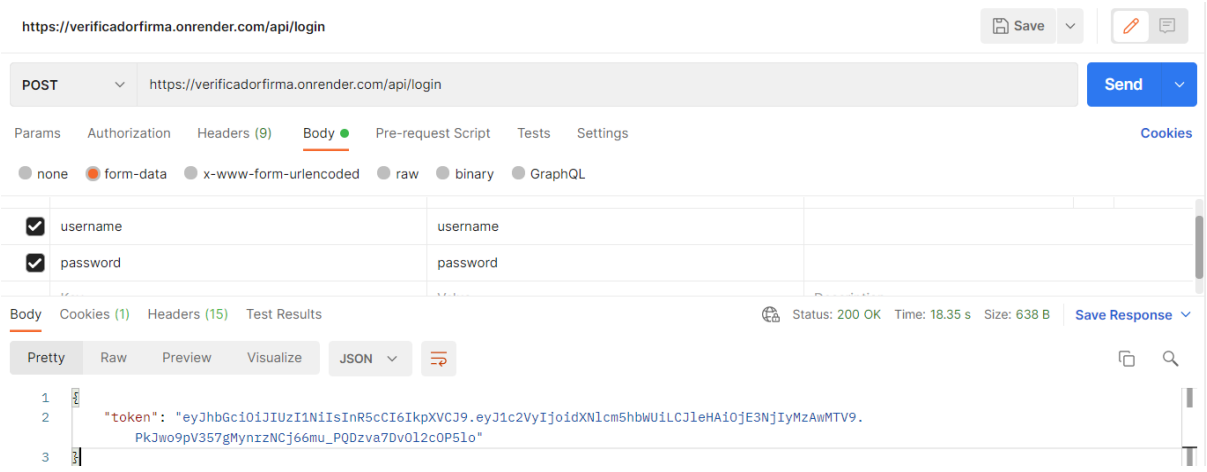
cada endpoint casos de prueba que abarcan situaciones de éxito y situaciones de error.

5.2.1 PRUEBA DE AUTENTICACIÓN

Caso #1: Verificar que el sistema genera un token JWT válido cuando se proporcionan credenciales correctas.

Se ejecutó la prueba para validar el proceso de autenticación exitosa. Se enviaron las credenciales válidas (username: "username", password: "password") mediante una petición HTTP POST al endpoint /api/login con formato application/x-www-form-urlencoded. El sistema validó correctamente las credenciales, generó un token JWT firmado con el algoritmo HS256 y la clave secreta "EprocessCompany", y respondió con código HTTP 200. El token retornado sigue el formato estándar JWT. La Figura 13 muestra la petición y respuesta capturada en Postman.

Figura 13. Login: credenciales correctas.

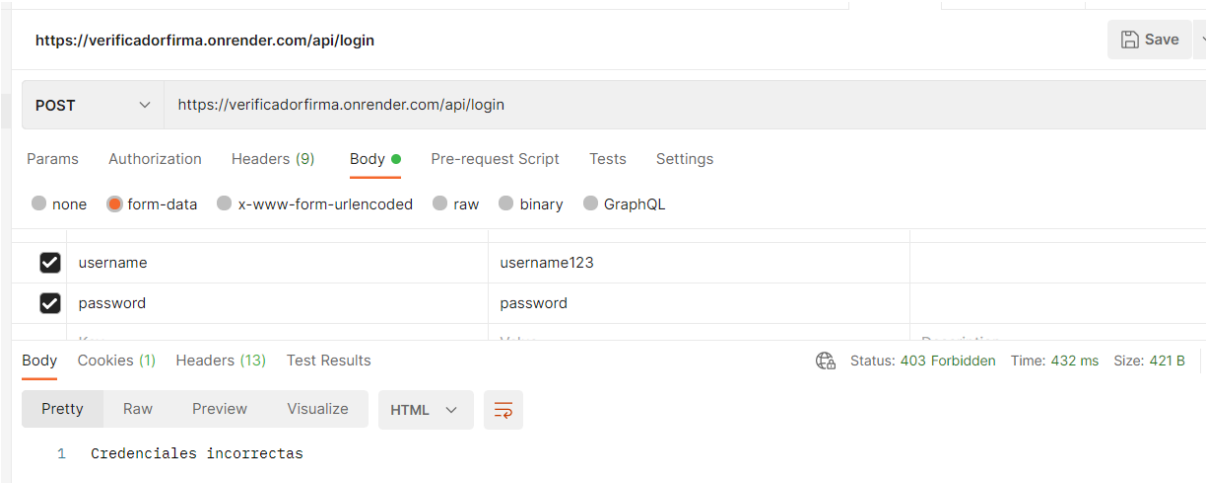


Nota: Captura postman del consumo del endpoint login con credenciales correctas. Fuente: Elaboración propia.

Caso #2: Verificar que el sistema rechaza credenciales inválidas

Se envió una petición POST al endpoint: /api/login, usando "username123" como nombre de usuario, que no coincide con las credenciales válidas establecidas en el sistema. El sistema rechazó la verificación sin crear un token y devolvió el mensaje de texto plano "Credenciales incorrectas" así como el código HTTP 403 (Forbidden). Este comportamiento es el que se previó de acuerdo con la implementación del endpoint. Esta prueba se registra en la Figura 14, donde se puede observar cómo el sistema protege el acceso cuando se intenta autenticar con credenciales inválidas. Es importante señalar que el sistema no brinda información concreta acerca de cuál de los dos campos (contraseña o usuario) es incorrecto, en cumplimiento con las buenas prácticas de seguridad que evitan ataques de enumeración de usuarios.

Figura 14. Login: credenciales incorrectas.



Nota: Captura postman del consumo del endpoint login con credenciales incorrectas. Fuente: Elaboración propia.

5.2.2 PRUEBA DE VERIFICACIÓN DE FIRMAS CON IMÁGENES

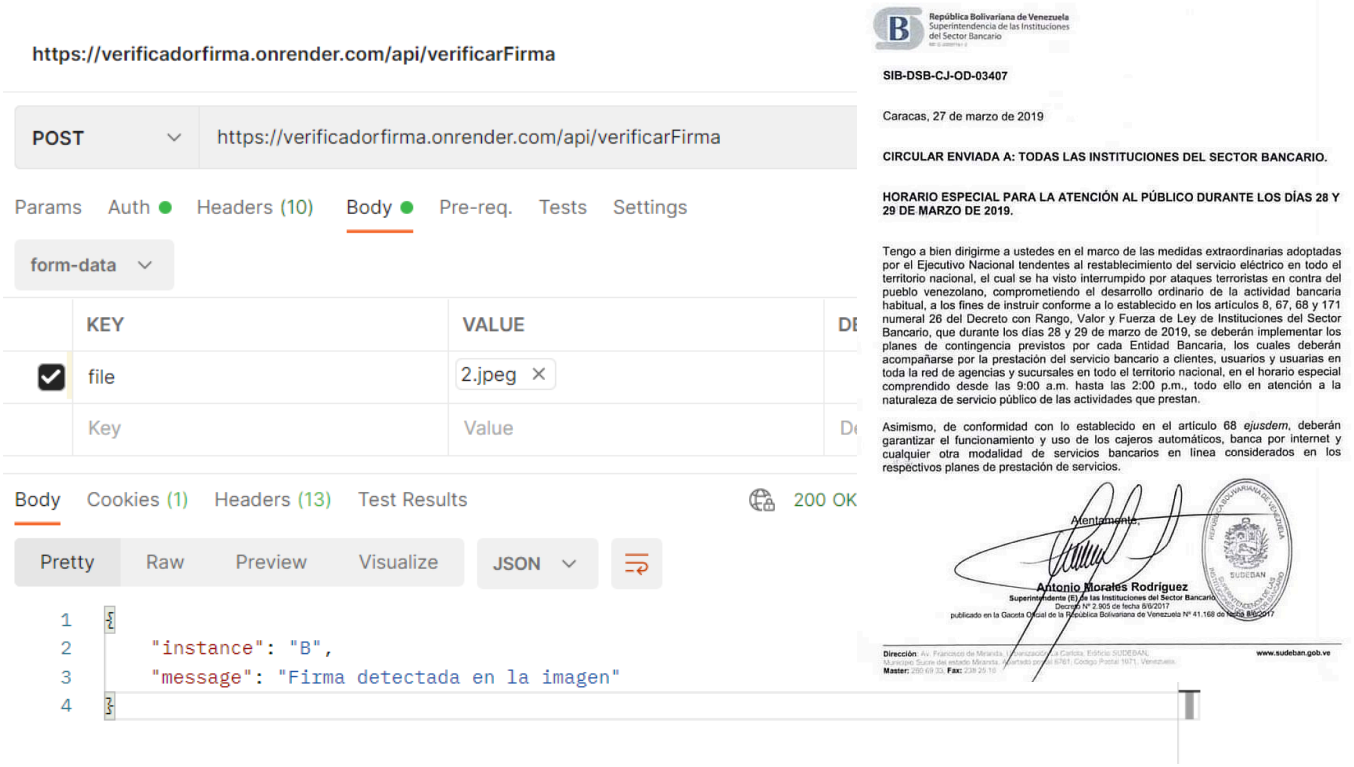
Caso #1: Imágenes con firma manuscrita

Se examinó la habilidad del prototipo para identificar firmas manuscritas en imágenes JPEG. Se emplearon múltiples imágenes de prueba que incluían una firma manuscrita sobre un archivo digital. Los pasos del procedimiento incluyeron:

1. Conseguir un token válido usando el endpoint de login.
2. Generar una solicitud POST hacia el endpoint /api/verificarFirma.
3. Añadir en la cabecera Authorization: Bearer [token].
4. Añadir el archivo de imagen al cuerpo con formato multipart/form-data usando la clave "file".

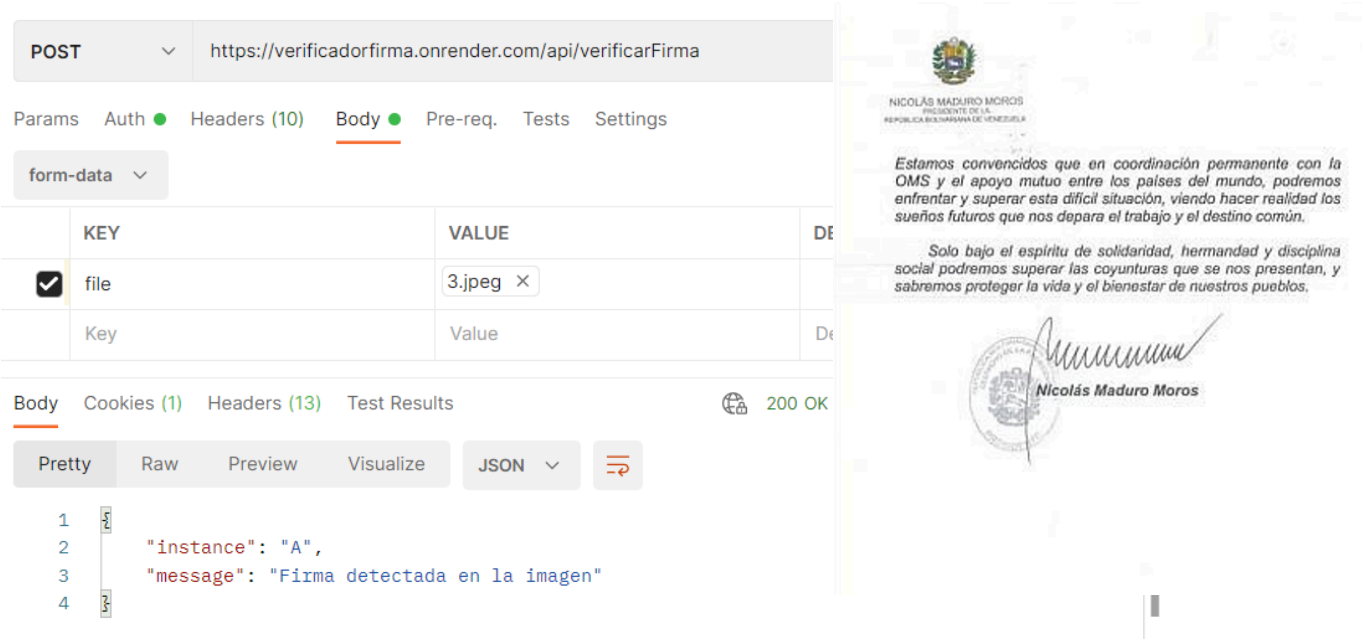
La imagen fue procesada de manera adecuada por el sistema: el token fue validado por el decorador @token_required; la función image_to_base64() convirtió la imagen en Base64; y, finalmente, la función procesar_imagen_base64() hizo una solicitud a OpenAI GPT-4.1 con el prompt. El modelo dio respuesta positiva. El endpoint retornó código HTTP 200 con el mensaje JSON {"message": "Firma detectada en la imagen", "instance": "B" }. Las Figuras 15, 16 y 17 muestran la petición en Postman con el archivo adjunto y la respuesta obtenida.

Figura 15. Verificar firma: JPG 1.



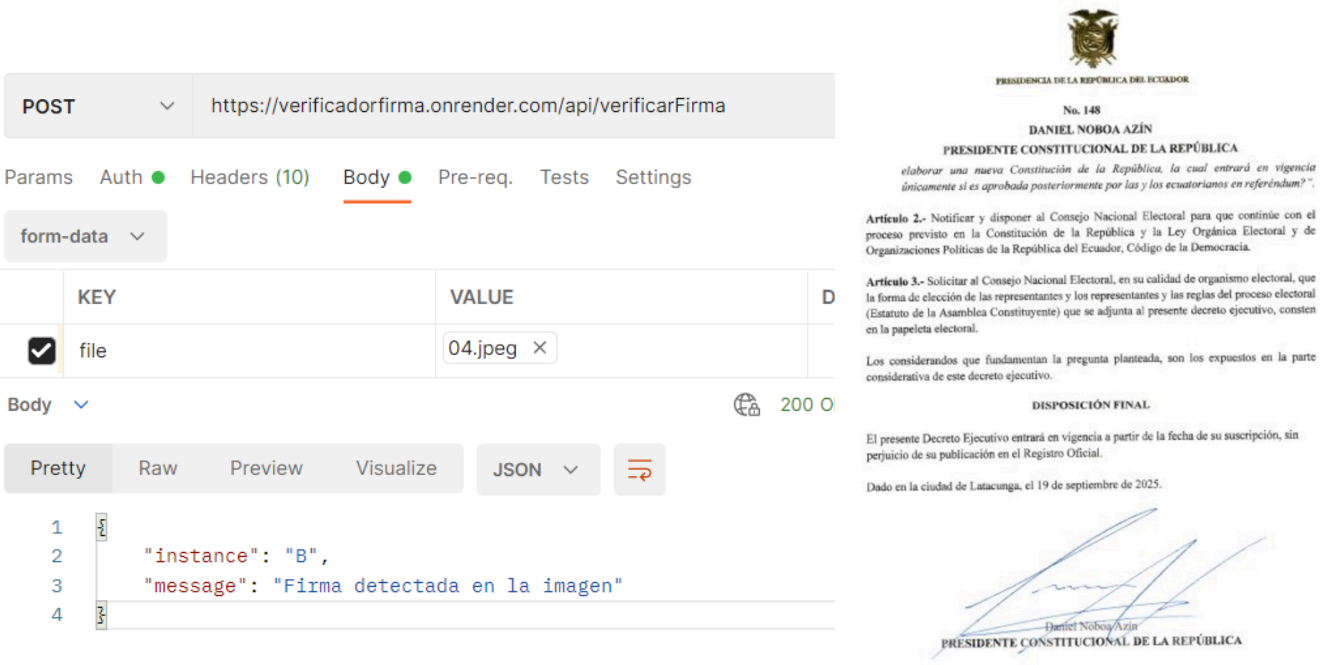
Nota: Captura postman del consumo del endpoint verificarFirma imagen JPG. Fuente: Elaboración propia.

Figura 16. Verificar firma: JPG 2.



Nota: Captura postman del consumo del endpoint verificarFirma imagen JPG. Fuente: Elaboración propia.

Figura 17. Verificar firma: JPG 3.



Nota: Captura postman del consumo del endpoint verificarFirma imagen JPG. Fuente: Elaboración propia.

Se comprobó la identificación de firmas en formato PNG, que es un formato con compresión sin pérdida frecuentemente empleado para documentos escaneados de alta calidad. Se utilizó una imagen en formato PNG de un documento escaneado. El prototipo trató el archivo de la misma manera que para JPEG: validación del token, apertura de la imagen, codificación a Base64 en formato PNG (manteniendo la calidad inicial), remisión a OpenAI y procesamiento de respuesta. El sistema devolvió el código HTTP 200 con el mensaje "Firma detectada en la imagen" después de que el modelo GPT-4.1 determinó correctamente que había una firma manuscrita. No se observaron diferencias significativas en precisión o tiempo de respuesta entre formatos JPEG y PNG, confirmando que el sistema maneja ambos formatos de manera equivalente según lo especificado en los requisitos del prototipo.

Figura 18. Verificar firma: PNG 1.



Nota: Captura postman del consumo del endpoint verificarFirma imagen PNG. Fuente: Elaboración propia.

Figura 19. Verificar firma: PNG 2.



Nota: Captura postman del consumo del endpoint verificarFirma imagen PNG. Fuente: Elaboración propia.

Caso #2: Imágenes sin firma

Se validó que el prototipo no generara falsos positivos al procesar imágenes que no contienen firmas manuscritas. Se utilizaron entonces varias imagenes de documentos escaneados que no tuvieran la presencia de alguna firma manuscrita. El prototipo procesó dichas imágenes correctamente: las codificó en Base64, las envió a OpenAI GPT-4.1 con el mismo prompt utilizado en las pruebas anteriores, y el modelo analizó la imagen determinando correctamente la ausencia de firmas. El endpoint retornó código HTTP 200 con el mensaje JSON {"message": "No se detectó firma en la imagen"}. Esta prueba, que se llevó a cabo con 20 imágenes distintas de varios tipos de archivos, como certificados, facturas, documentos de texto y formularios administrativos, donde el prototipo no identifico firmas manuscritas en ninguno de las imagenes. Esto demuestra que el modelo GPT-4.1 puede distinguir entre las firmas manuscritas auténticas, los elementos gráficos del documento y el texto impreso con una alta tasa de precisión.

Figura 20. Verificar firma: imagen 1 sin firma.



Nota: Captura postman del consumo del endpoint verificarFirma imagen sin firma. Fuente: Elaboración propia.

Figura 21. Verificar firma: imagen 2 sin firma.

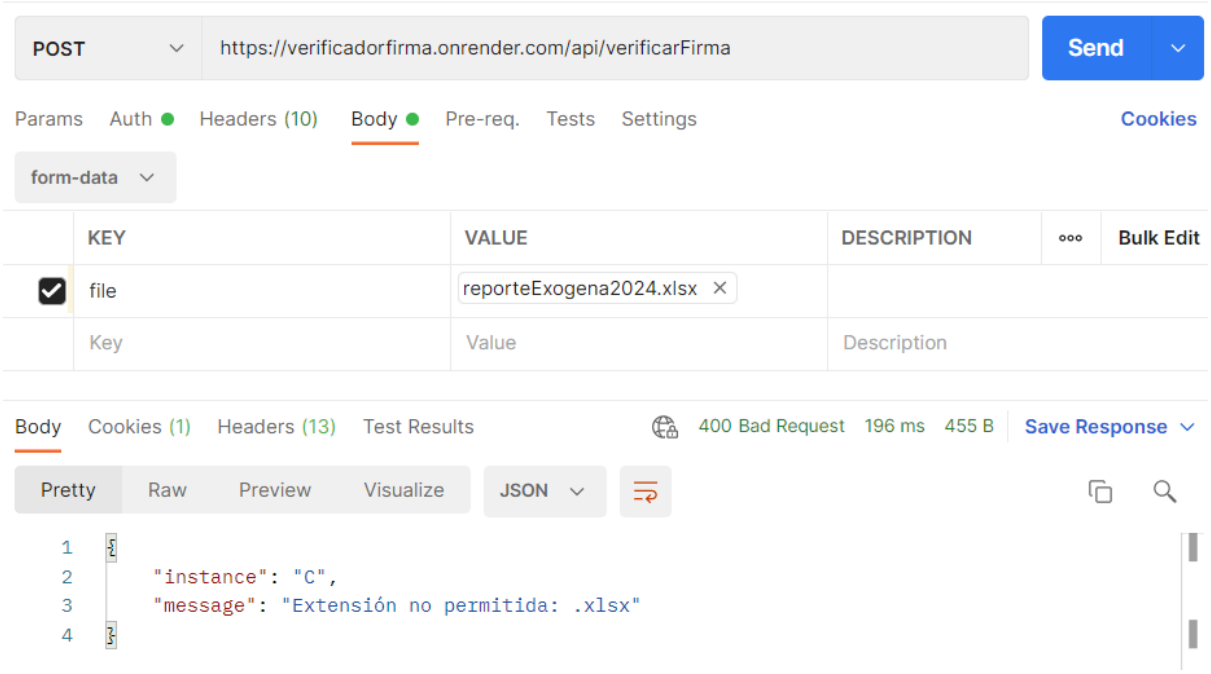


Nota: Captura postman del consumo del endpoint verificarFirma imagen sin firma. Fuente: Elaboración propia.

Caso #3: Formatos de archivos diferentes a los permitidos

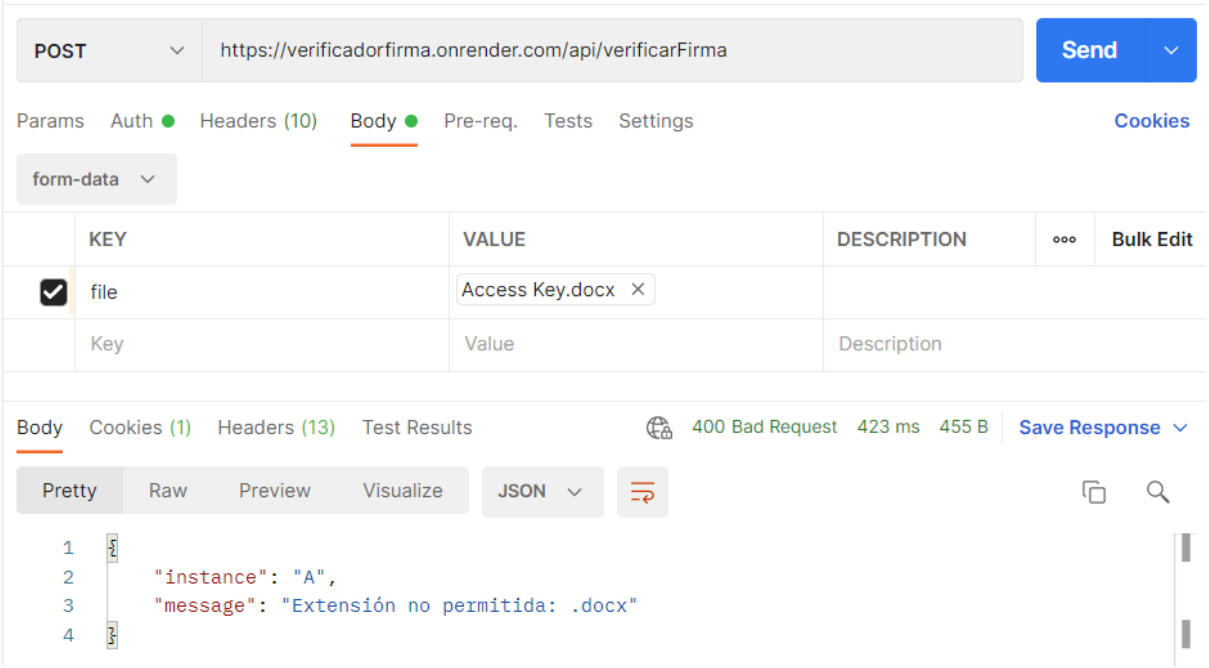
Se validó el procesamiento de diferentes formatos de archivo para verificar que el prototipo acepta únicamente los formatos especificados en el diseño (JPEG, PNG, PDF) y rechaza apropiadamente formatos no soportados. Las siguientes figuras, muestran archivos con extensiones como: .txt, .xlsx y .docx, los cuales fueron enviados al endpoint de verificación por la prueba. El prototipo validó que el archivo no tenía la extensión adecuada y por ello devolvió una solicitud con un código HTTP 400 (Bad Request) y el mensaje JSON {“message”: "Extensión no autorizada: Nombre extension". Esta conducta es la que se prevé y evidencia que el prototipo está implementando de manera adecuada la validación de formatos de entrada, lo cual protege frente a intentos de procesar tipos de archivo no compatibles, los cuales podrían generar fallos o vulnerabilidades en términos de seguridad.

Figura 22. Verificar firma: archivo con formato no válido: xlsx.



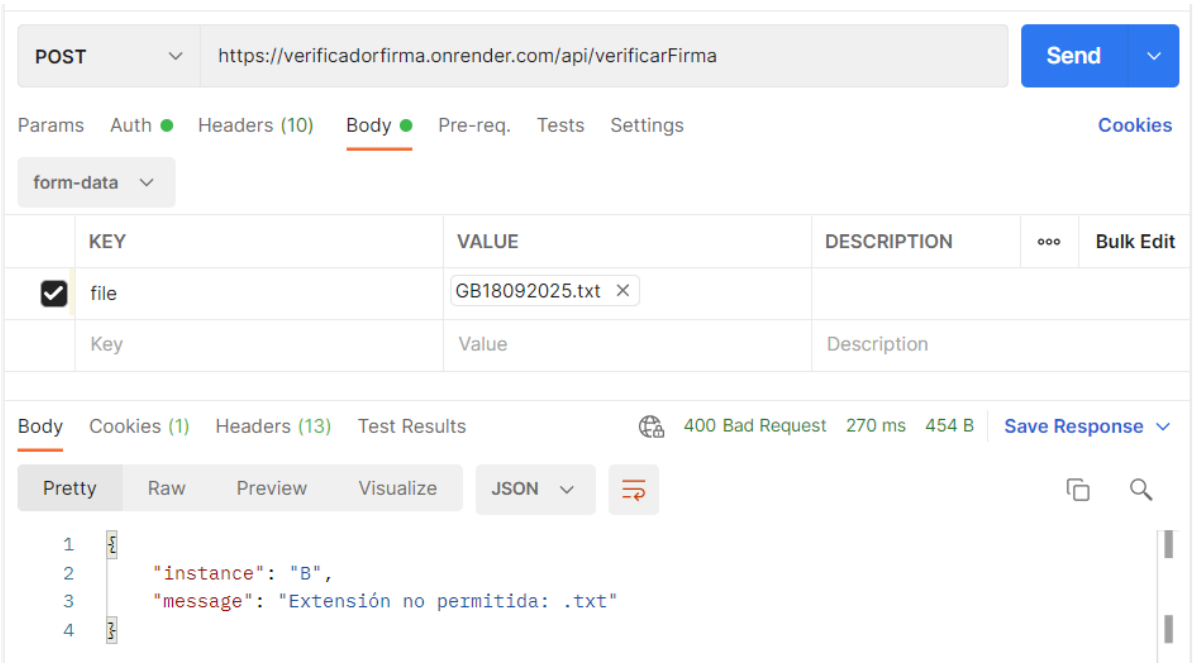
Nota: Captura postman del consumo del endpoint verificarFirma formato no válido. Fuente: Elaboración propia.

Figura 23. Verificar firma: archivo con formato no válido: docx.



Nota: Captura postman del consumo del endpoint verificarFirma formato no válido. Fuente: Elaboración propia.

Figura 24. Verificar firma: archivo con formato no válido: docx.



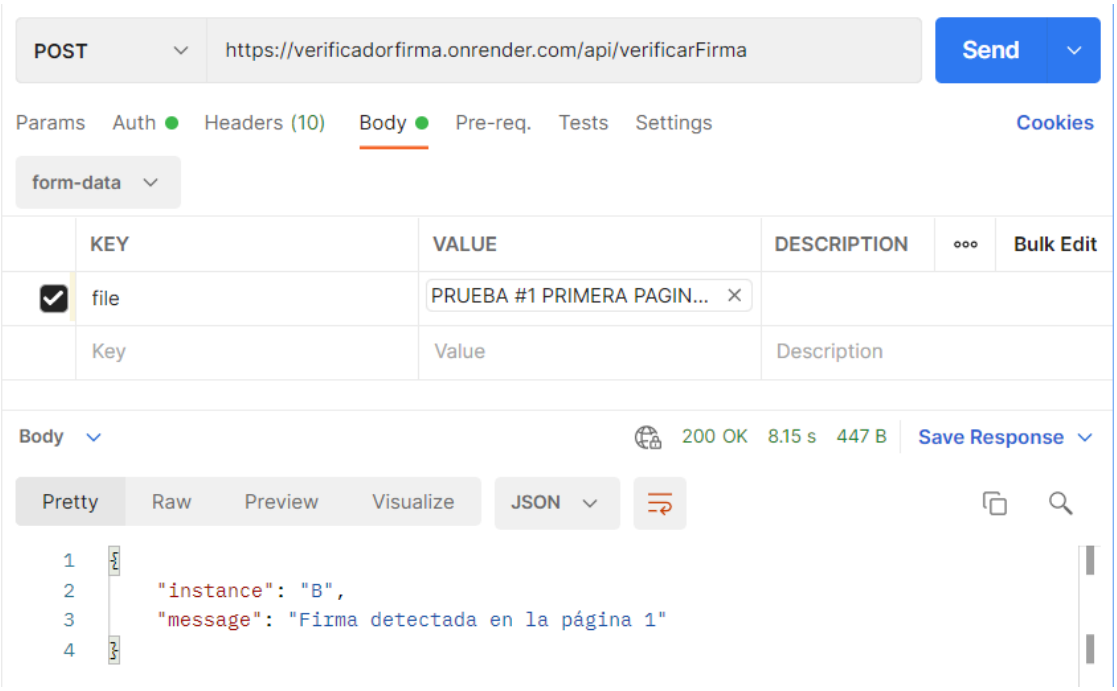
Nota: Captura postman del consumo del endpoint verificarFirma formato no válido. Fuente: Elaboración propia.

5.2.3 PRUEBA DE VERIFICACIÓN CON DOCUMENTOS PDF

Caso #1: PDF con firma en primera página

Se examinó el manejo de archivos PDF con firma manuscrita situada en la primera página. El prototipo analizó el archivo, validó el token JWT, posteriormente, leyó el archivo PDF desde la memoria y, finalmente, transformó las páginas en imagenes. El prototipo itero cada imagen resultante la cual fue codificada en Base64 y luego enviada a OpenAI GPT-4.1. El modelo identificó adecuadamente la firma manuscrita en la primera página y el sistema devolvió un código HTTP 200 junto con el mensaje JSON {"message": "Firma detectada en la página 1". El sistema no procesó la página 2 debido a que se implementó la detección temprana, lo cual optimizó el uso de tokens de OpenAI. La solicitud en Postman y la respuesta que se recibió están ilustradas en la Figura 25.

Figura 25. Verificar firma: PDF con firma en primera página.

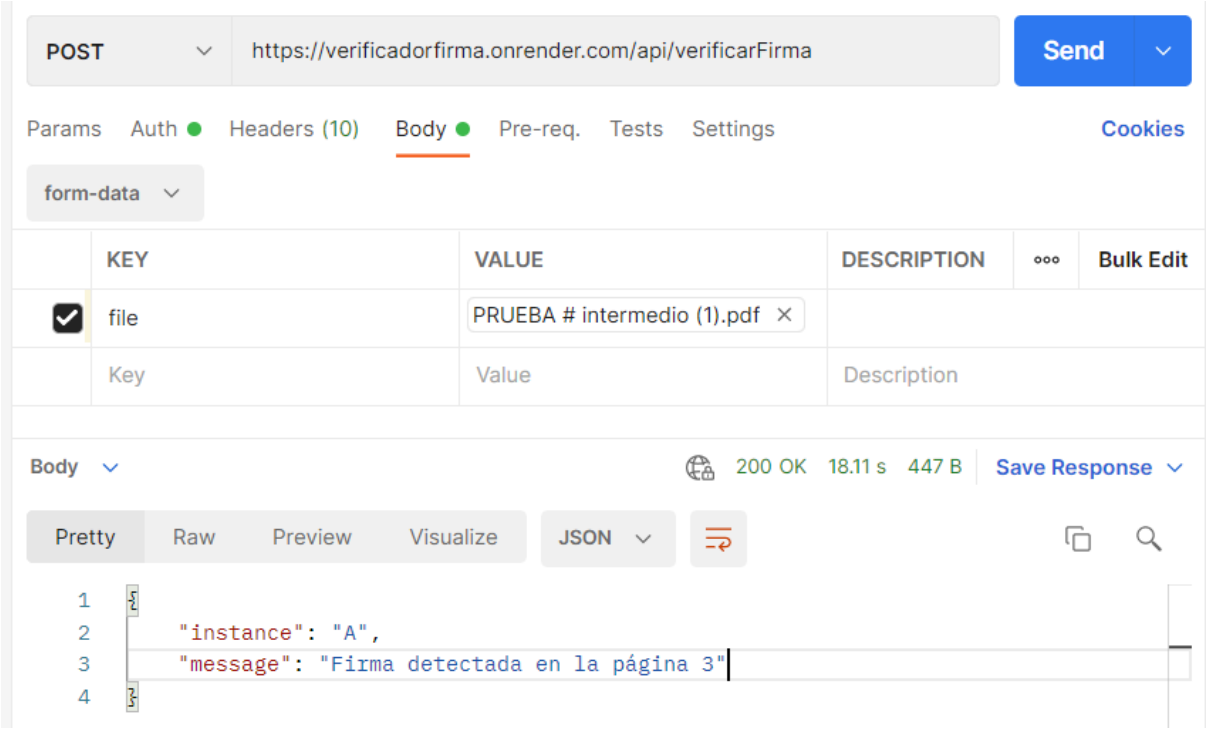


Nota: Captura postman del consumo del endpoint verificarFirma: PDF con firma en primera página. Fuente: Elaboración propia.

Caso #2: PDF con firma en página intermedia

Se realizó la prueba con un pdf de 5 páginas para evaluar cómo se comportaba el prototipo con archivos más grandes, el pdf contenía una firma manuscrita en la página 3. El prototipo entonces procesó correctamente las primeras 2 páginas sin detectar alguna firma manuscrita, identificó asi la firma en la página 3, y detuvo el procesamiento sin analizar las páginas 4 y 5. El endpoint devolvió el código HTTP 200 con el mensaje {"message": "Firma detectada en la página 3"} en cerca de 18 segundos. Este examen demuestra que el tiempo requerido para procesar se incrementa de manera lineal con la cantidad de páginas que es necesario analizar antes de hallar una firma. Para documentos largos en los que la firma está en las últimas páginas, los tiempos de respuesta pueden resultar significativos.

Figura 26. Verificar firma: PDF con firma en página intermedia.

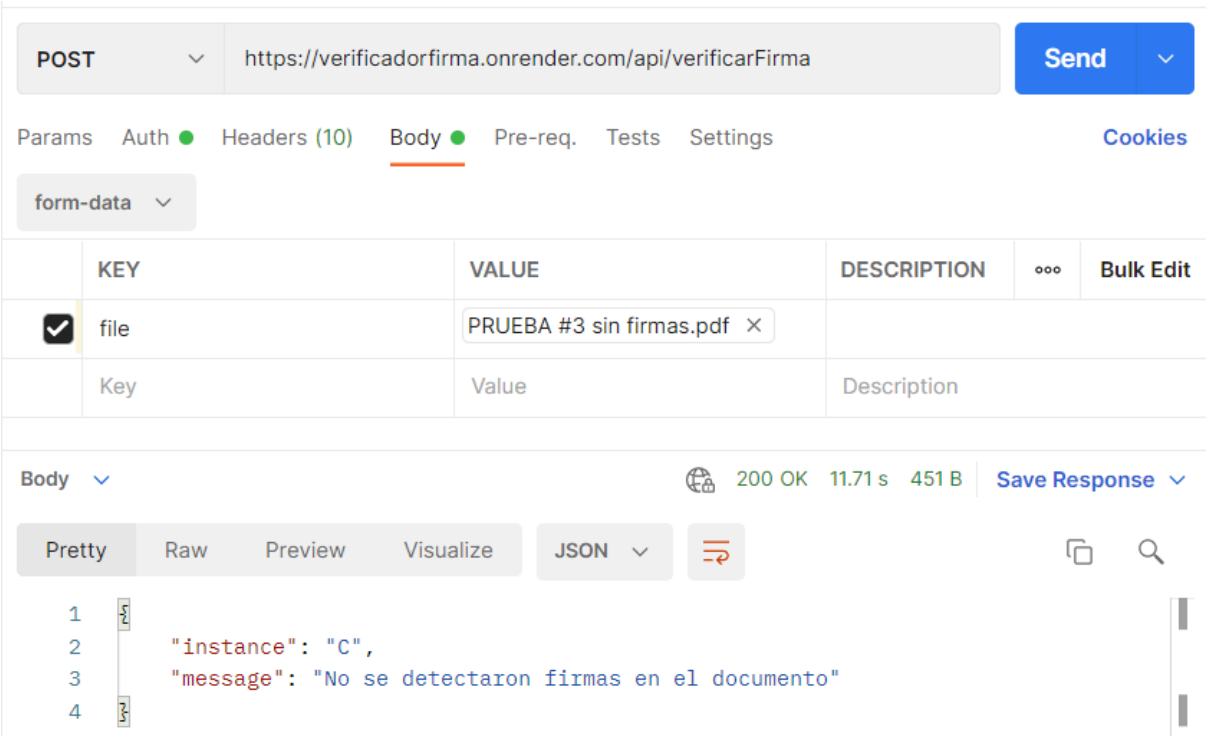


Nota: Captura postman del consumo del endpoint verificarFirma: PDF con firma en página intermedia. Fuente: Elaboración propia.

Caso #4: PDF sin firmas

Se verificó el comportamiento del prototipo cuando un documento PDF no contiene ninguna firma manuscrita en ninguna de sus páginas. Se utilizó un documento de 3 páginas conteniendo únicamente texto impreso sin firmas manuscritas. El prototipo convirtió cada página en imagen y procesó secuencialmente las 3 páginas; codificación Base64, y consulta a OpenAI GPT-4.1. El prototipo respondió que no habia presencia de alguna firma manuscrita en las 3 ocasiones. Al completar el procesamiento de todas las páginas sin detectar firma en ninguna, el prototipo retornó ({"message": "No se detectaron firmas en el documento"}), 200. Este comportamiento es el esperado ya que se debe agotar todas las páginas antes de concluir definitivamente que el documento no contiene firmas.

Figura 27. Verificar firma: PDF sin firma.



Nota: Captura postman del consumo del endpoint verificarFirma: PDF sin firma. Fuente: Elaboración propia.

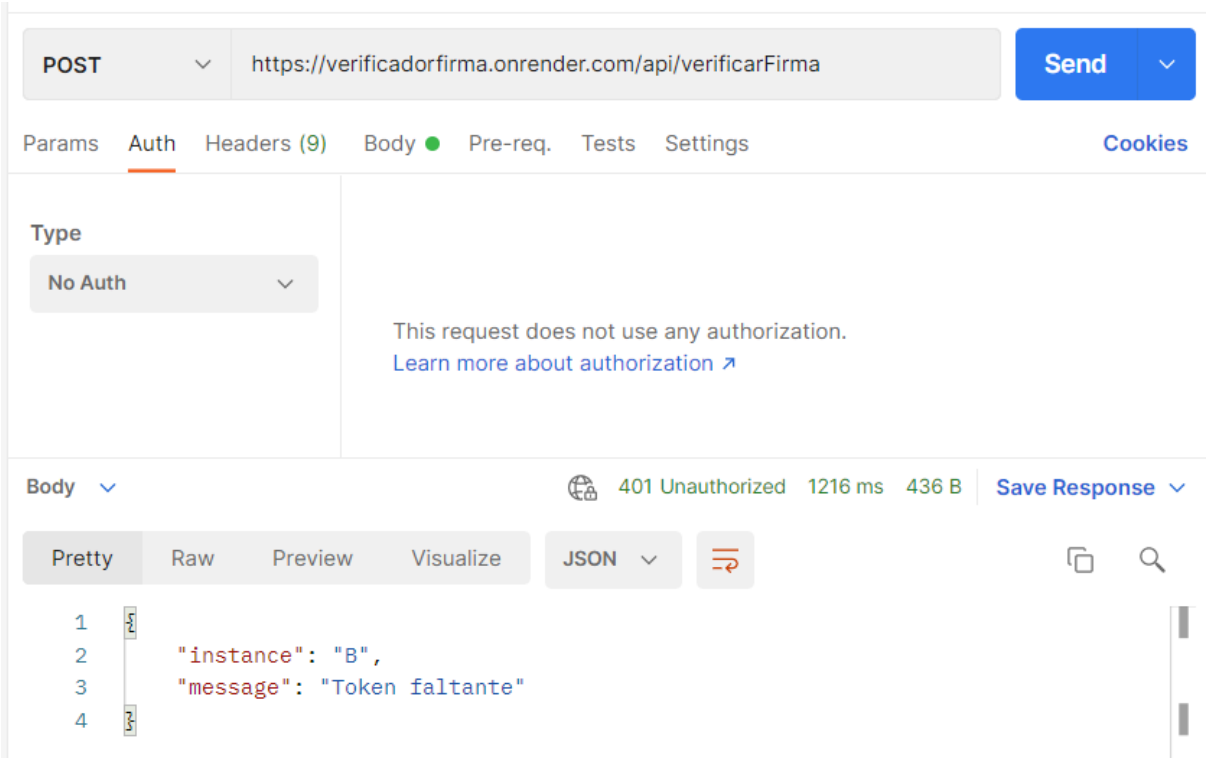
5.3 PRUEBAS DE SEGURIDAD

Las pruebas de seguridad verifican que los mecanismos de autorización y autenticación, implementados a través de JSON Web Tokens (JWT) protegen los recursos del prototipo, evitando accesos no autorizados y asegurando que solo las personas con tokens válidos tengan la posibilidad de hacer uso del endpoint para verificar firmas.

5.3.1 ACCESO SIN TOKEN

Se verificó el comportamiento del prototipo cuando un usuario intenta entrar en el endpoint protegido /api/verificarFirma sin ofrecer un token de autenticación. Se creó una solicitud POST al endpoint, anexando un archivo válido en el cuerpo y omitiendo el encabezado Authorization. El decorador @token_required interceptó la solicitud antes de llevar a cabo la lógica del endpoint, comprobó que no había un encabezado de autorización y devolvió al instante el mensaje JSON {"message": con el código HTTP 401 (Unauthorized). "Token faltante". El archivo no fue procesado ni se comunicó con OpenAI porque la función del endpoint nunca se activó. Este comportamiento se ajusta al principio de "fail secure" (fallar de manera segura), que establece que cuando no hay credenciales válidas, el sistema niega el acceso sin procesar información. La solicitud y la respuesta de rechazo en Postman están representadas en la Figura 28.

Figura 28. Verificar firma: sin token.

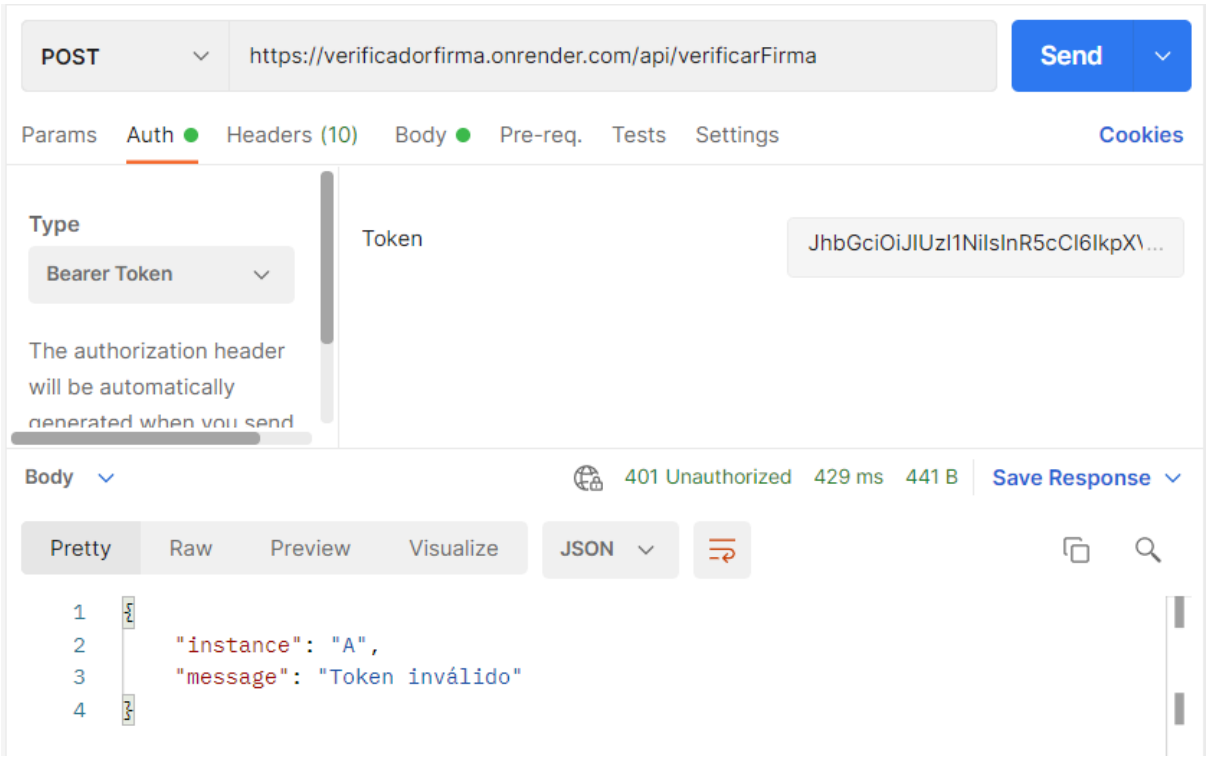


Nota: Captura postman del consumo del endpoint verificarFirma sin envío de token. Fuente: Elaboración propia.

5.3.2 TOKEN INVÁLIDO O MODIFICADO

Se evaluó la capacidad del prototipo para detectar y rechazar tokens JWT que han sido modificados o son inválidos. Se genero un token valido y se modificó entonces manualmente uno de sus caracteres (simulando un intento de manipulación o un token corrupto), y se envió una petición al endpoint protegido incluyendo este token modificado en el header Authorization: Bearer [token_modificado]. La verificación de firma falló debido a que la firma del token no corresponde con el contenido modificado. La biblioteca PyJWT lanzó una excepción jwt.InvalidTokenError, resultando en el retorno de código HTTP 401 con el mensaje {"message": "Token inválido"}. Esta prueba verifica que el prototipo lleva a cabo de manera adecuada la comprobación criptográfica de la integridad de los tokens, lo cual evita que se acepten tokens que han sido cambiados.

Figura 29. Verificar firma: token invalido.

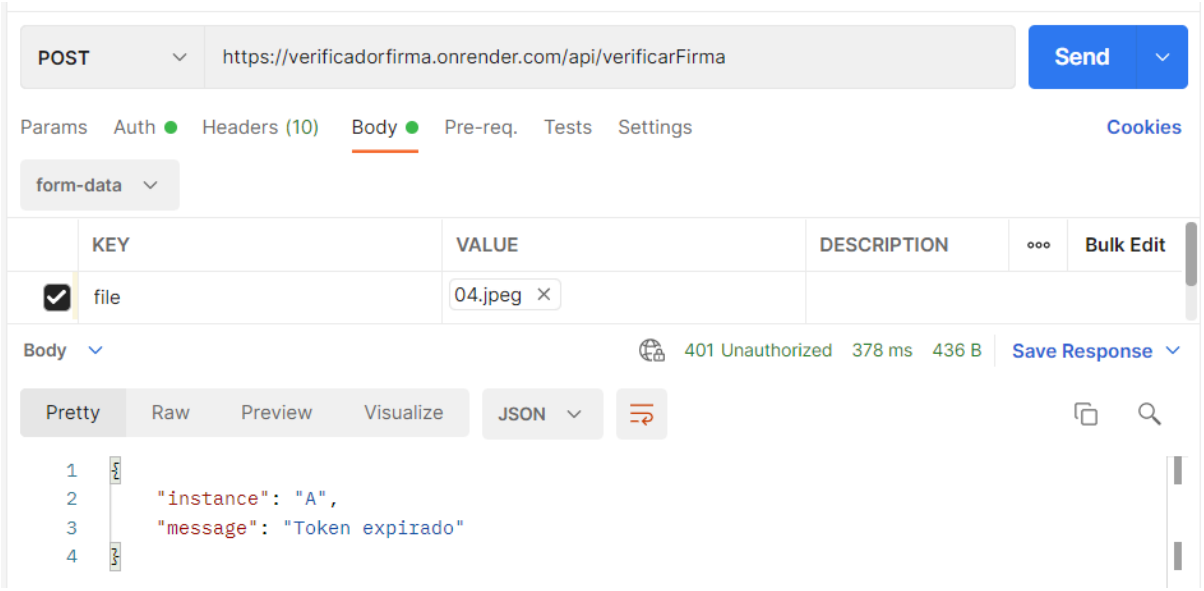


Nota: Captura postman del consumo del endpoint verificarFirma con token invalido. Fuente: Elaboración propia.

5.3.3 TOKEN EXPIRADO

Para comprobar la caducidad de los tokens, se generó primero un token valido. Este token fue guardado y se dejó pasar 21 minutos, más que el tiempo de expiración establecido (20 minutos), antes de intentar acceder al endpoint protegido /api/verificarFirma usando el token en el encabezado de autorización. el servicio examinó el token que recibió. Lo decodificó con PyJWT y determinó acertadamente que el campo "exp" del payload contenía un timestamp anterior al presente, lo que significa que el token había caducado. El sistema denegó correctamente la solicitud al devolver el código HTTP 401 (No autorizado) y el mensaje JSON {"message": "Token caducado"}.

Figura 30. Verificar firma: token expirado.



Nota: Captura postman del consumo del endpoint verificarFirma con token expirado. Fuente: Elaboración propia.

5.4 PRUEBAS DE BALANCEO DE CARGA

Las pruebas de balanceo de carga validan que el sistema cumple con el objetivo de "desarrollar un prototipo para el reconocimiento automático de firmas manuscritas en documentos digitales, integrándose en una arquitectura que implemente balanceo de carga para su posterior despliegue". La arquitectura implementada consiste de un contenedor Docker que corre al mismo tiempo tres instancias del servicio Flask (denominadas A, B y C) en los puertos 5000, 5001 y 5002 respectivamente. Un servidor Nginx que opera como proxy inverso y balanceador de carga; su función es repartir las solicitudes entrantes entre las tres instancias a través del algoritmo round-robin.

5.4.1 DISTRIBUCIÓN DE PETICIONES ENTRE INSTANCIAS

Se hizo la prueba para validar que el balanceador Nginx reparte las solicitudes de manera justa entre las tres instancias de Flask. Cada instancia fue entonces configurada con una variable de ambiente INSTANCE_ID única (A, B o C), la cual se

incorpora en todas las respuestas JSON a través del campo "instance", lo que posibilita la identificación de qué instancia gestionó cada solicitud. Se mandaron 15 solicitudes seguidas al servicio que opera en Render a través del endpoint /api/verificarFirma, cada una con un token JWT válido y archivos de imagen que contenían firmas manuscritas. El campo "instance" del servicio, demostró un patrón cíclico al examinar las respuestas JSON, la instancia A manejó la primera solicitud, la B la segunda, la C la tercera y después volvió a procesar la cuarta por A, asi sucesivamente. Esta secuencia confirma entonces que Nginx implementa correctamente el algoritmo round-robin, donde cada instancia recibe exactamente el mismo número de peticiones.

Tabla 5. Resultado de distribución entre instancias

Petición #	Instancia (instance)	Mensaje
1	A	Firma detectada en la imagen
2	B	Firma detectada en la imagen
3	C	Firma detectada en la imagen
4	A	Firma detectada en la imagen
5	B	Firma detectada en la imagen
6	C	Firma detectada en la imagen
7	A	Firma detectada en la imagen
8	B	Firma detectada en la

		imagen
9	C	Firma detectada en la imagen
10	A	Firma detectada en la imagen
11	B	Firma detectada en la imagen
12	C	Firma detectada en la imagen
13	A	Firma detectada en la imagen
14	B	Firma detectada en la imagen
15	C	Firma detectada en la imagen

Nota: La tabla presenta el resultado de la distribución entre instancias realizadas por el prototipo. Fuente: Elaboración Propia.

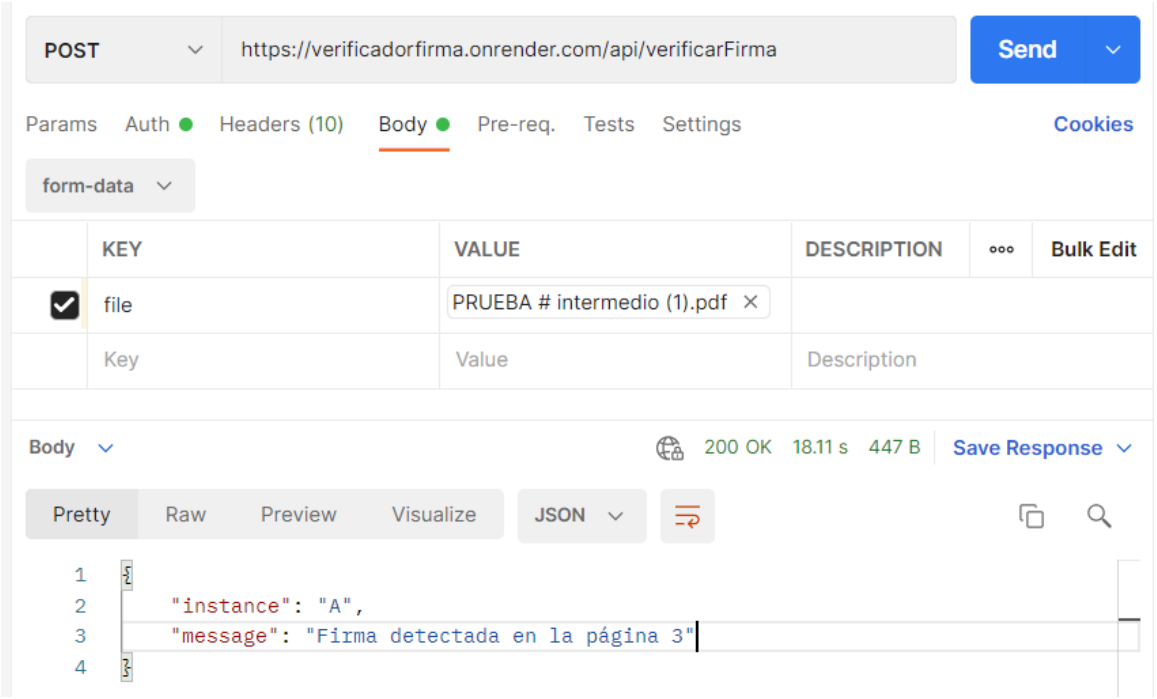
Distribución de carga:

- Instancia A: 5 peticiones (33.3%)
- Instancia B: 5 peticiones (33.3%)
- Instancia C: 5 peticiones (33.3%)

Las figuras 31, 32 y 33 presentan imágenes de Postman en las que se puede apreciar cómo el campo "instance" alterna entre las instancias A, B y C en sucesivas respuestas. La prueba realizada demuestra entonces que el balanceo de carga está

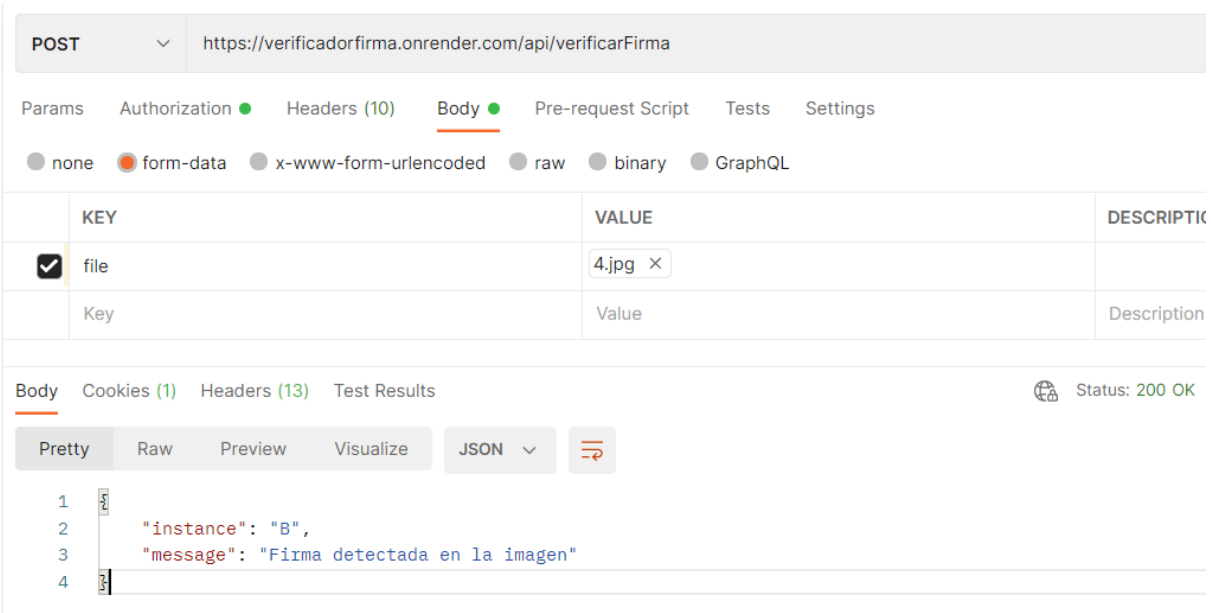
adecuadamente según lo propuesto, cumpliendo con el objetivo de distribuir equitativamente la carga entre múltiples instancias del servicio.

Figura 31. Verificar firma: instancia A.



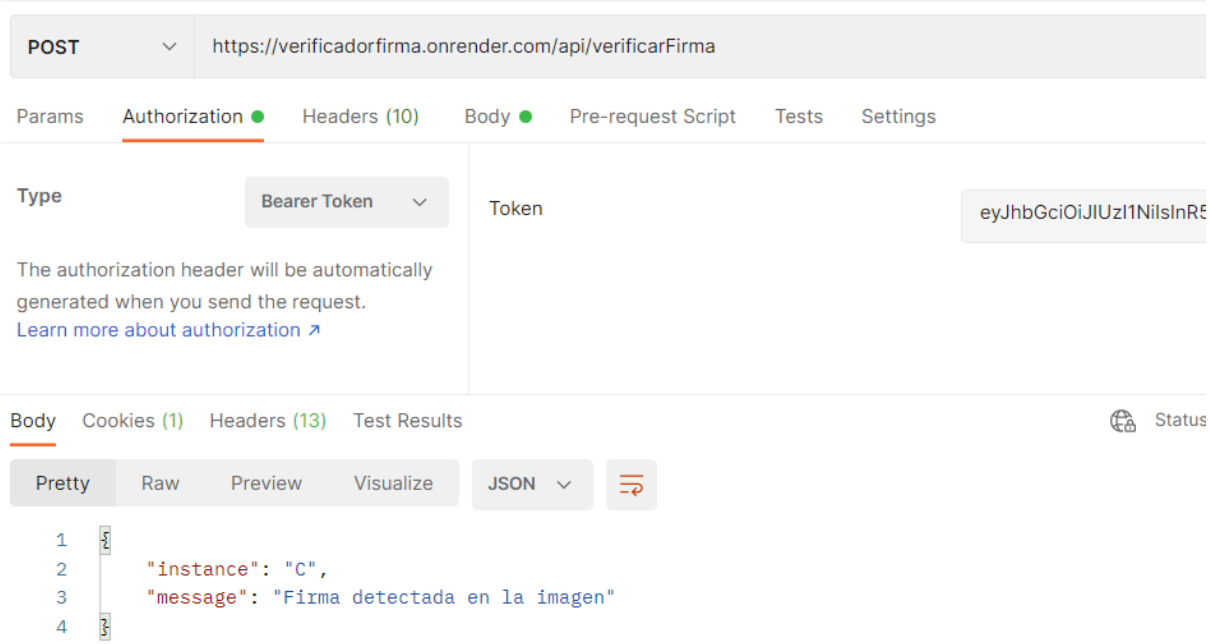
Nota: Captura postman del consumo del endpoint verificarFirma en la instancia A. Fuente: Elaboración propia.

Figura 32. Verificar firma: instancia B.



Nota: Captura postman del consumo del endpoint verificarFirma en la instancia B. Fuente: Elaboración propia.

Figura 33. Verificar firma: instancia C.



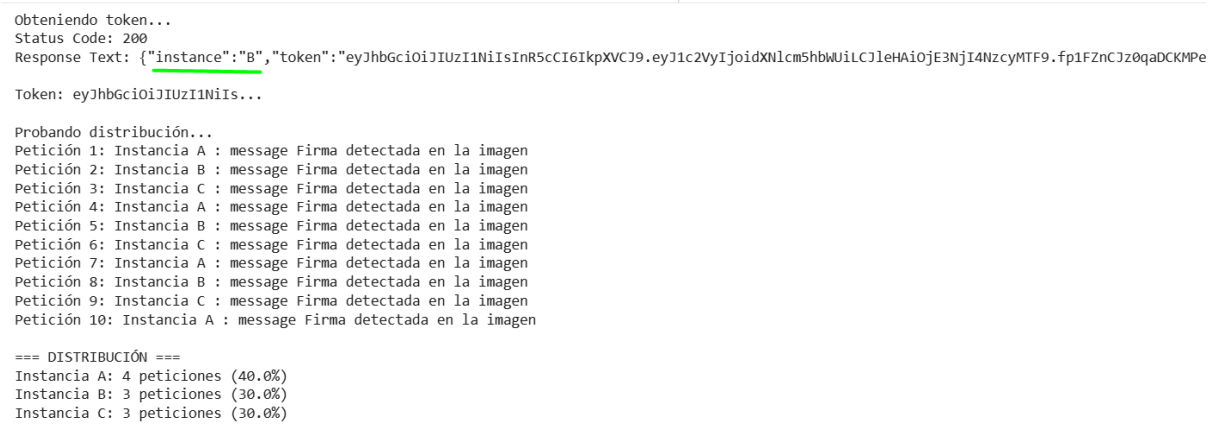
Nota: Captura postman del consumo del endpoint verificarFirma en la instancia C. Fuente: Elaboración propia.

5.4.2 PRUEBA DE PERSISTENCIA DE SESIÓN (STATELESS JWT)

Se validó si la arquitectura implementada es stateless, es decir, sin estado, lo cual es una parte fundamental para que el balanceo de carga funcione correctamente sin dependencias de sesión. Se obtuvo un token JWT mediante autenticación en el endpoint /api/login, que según el campo "instance" de la respuesta fue procesado por la instancia B, por consiguiente se utilizó ese mismo token para realizar 10 peticiones consecutivas al endpoint protegido /api/verificarFirma. Nginx repartió estas diez solicitudes de manera cíclica entre las tres instancias: A proceso 4 solicitudes, B proceso 3 y C proceso 4. Lo importante de esta prueba es que las tres instancias A, B y C validaron con éxito el token JWT creado originalmente por la

instancia B sin necesidad de que las instancias se comunicaran entre sí o accedieran a una base de datos centralizada de sesiones.

Figura 34. Verificar firma: instancia C.



Nota: Captura de log de la prueba de persistencia de sesión. Fuente: Elaboración propia.

CONCLUSIONES

Se logró desarrollar un prototipo para el reconocimiento automático de firmas manuscritas en documentos digitales aplicable a Eprocess Company S.A.S., cumpliendo con los cuatro objetivos específicos planteados. Se llevó a cabo un estudio técnico comparativo de tres tecnologías: YOLO, Mistral Vision y GPT-4.1 Mini. Se eligió a GPT-4.1 Mini como la alternativa más conveniente por su mayor exactitud en benchmarks especializados, su integración sencilla mediante API REST y la eliminación de las exigencias de infraestructura local especializada. El prototipo que se ha implementado analiza adecuadamente documentos en formatos PNG, JPEG y PDF mediante un algoritmo de detección temprana, que optimiza el uso de tokens al paralizar el análisis cuando una firma es identificada.

La arquitectura distribuida implementada con NGINX como balanceador de carga, Gunicorn como servidor de aplicaciones y tres instancias independientes de Flask desplegadas en Render demostró funcionar correctamente, distribuyendo las peticiones de manera equitativa mediante el algoritmo round-robin. El diseño stateless basado en tokens JWT posibilita que cualquier instancia valide tokens creados por otras, sin necesidad de intercambiar estado; esto asegura la escalabilidad horizontal sin tener que cambiar la lógica de aplicación. Las pruebas de seguridad demostraron que el sistema niega correctamente accesos sin token, tokens caducados e inválidos, lo cual se ajusta a las prácticas de autenticación para servicios web RESTful.

RECOMENDACIONES

Con base en los resultados obtenidos durante el desarrollo e implementación del prototipo para el reconocimiento de firmas manuscritas en documentos digitales, se plantean las siguientes recomendaciones:

Se aconseja crear un sistema de respaldo o contingencia para reducir el riesgo que conlleva depender del servicio externo de OpenAI. Esto podría incluir la incorporación de un segundo proveedor de servicios de inteligencia artificial como opción (por ejemplo, Google Cloud Vision API o Mistral Vision) que pueda encenderse automáticamente si el servicio principal se interrumpe por mucho tiempo, o la puesta en marcha de una cola de procesamiento para guardar temporalmente documentos pendientes durante interrupciones cortas.

Se recomienda la creación de un módulo de preprocesamiento para optimizar los costos operativos y mejorar el rendimiento del sistema. Este módulo examinaría las propiedades fundamentales del documento (tamaño, cantidad de páginas, distribución del contenido) con el fin de prever dónde es más probable que se encuentren las firmas y modificar dinámicamente el orden en que se procesan las páginas en PDFs, empezando por las páginas finales o primeras basándose en patrones históricos.

Para facilitar la adopción por personal administrativo, puede diseñarse una interfaz intuitiva que permita cargar documentos, visualizar resultados y gestionar reportes sin necesidad de interactuar directamente con el endpoint.

BIBLIOGRAFÍA

PROGRESSSOFT CORPORATION. Is manual signature verification obsolete?. [en línea]. Ammán: ProgressSoft Corporation, 2023. [citado el 12 de agosto de 2025]. Disponible en: https://www.progressoft.com/blogs/is-manual-signature-verification-obsolete

CORDERO-PÉREZ, M. I.; SALAS-DUARTE, P. E. Inteligencia Artificial en la Gestión de Redes Telemáticas: Avances, Tendencias y Aplicaciones Actuales. En: Revista Vínculos. [en línea]. Bogotá. Vol. 21, No. 1 (2024). [citado el 12 de agosto de 2025]. Disponible en: https://revistas.udistrital.edu.co/index.php/vinculos/article/view/22565

BERNARDIS, E.; BERNARDIS, H.; BERÓN, M.; MONTEJANO, G. A. Seguridad en servicios web. [en línea]. La Plata: Universidad Nacional de La Plata, SEDICI, 2017. [citado el 12 de agosto de 2025]. Disponible en: http://sedici.unlp.edu.ar/handle/10915/62726

BBVA LABS; BBVA. JSON Web tokens (JWT): claves para usarlos de manera segura. En: BBVA NOTICIAS. [en línea]. Madrid, 22 de marzo de 2023. [citado el 12 de agosto de 2025]. Disponible en: https://www.bbva.com/es/innovacion/json-web-tokens-jwt-claves-para-usarlos-de-manera-segura/

CUÉLLAR, MARÍA ISABEL. Prototipo para el Reconocimiento de Firmas Manuscritas en Documentos Digitales. [Entrevista]. En: MANCERA, LINA. Proyecto de Grado: Prototipo para el Reconocimiento de Firmas Manuscritas en Documentos Digitales aplicable a Eprocess Company S.A.S. Bogota, 2025, Anexo 1.

DELOITTE. Digital Document Management Trends 2022. [en línea]. [Ciudad de publicación no especificada]: Deloitte, 2022. [citado el 13 de agosto de 2025]. Disponible en: https://www2.deloitte.com/

HAFEMANN, L. G.; SABOURIN, R.; OLIVEIRA, L. S. Learning Features for Offline Handwritten Signature Verification using Deep Convolutional Neural Networks. [Preprint].

En: arXiv preprint. [en línea]. [Localidad no especificada]: arXiv, 2017. [citado el 13 de agosto de 2025]. Disponible en: https://arxiv.org/abs/1705.05787

MANUEL, F. R. E. La firma como un método biométrico de identificación. [en línea]. Ciudad de México: Instituto Politécnico Nacional (IPN), 2008. [citado el 14 agosto de 2025]. Disponible en: https://www.repositoriodigital.ipn.mx/handle/123456789/8502

AMERICAN BAR ASSOCIATION. Understanding Electronic Signatures and Their Legal Validity. [en línea]. Chicago (Sede): American Bar Association, 2023. [citado el 14 de agosto de 2025]. Disponible en: https://www.americanbar.org/groups/business_law/publications/blt/2023/01/electronic-signatures-legal/

BRADSKI, G. The OpenCV Library. En: Dr. Dobb's Journal of Software Tools. [Localidad no especificada]: Dr. Dobb's Journal, 2000. [citado el 16 de agosto de 2025].

ATENOS. Diagrama de cómo funciona Scrum. [Imagen]. [Localidad no especificada]: Atenos, s.f. [citado el 16 de agosto de 2025]. Disponible en: https://atenos.com/agile/como-funciona-scrum/

JOCHER, G.; CHAURASIA, A.; STOKEN, A.; BOROVEC, J.; ULTRALYTICS. YOLOv5 by Ultralytics. [Código fuente en línea]. [Localidad no especificada]: GitHub, 2023. [citado el 20 de agosto de 2025]. Disponible en: https://github.com/ultralytics/yolov5

REDMON, J.; DIVVALA, S.; GIRSHICK, R.; FARHADI, A. You Only Look Once: Unified, Real-Time Object Detection. En: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). [en línea]. [Localidad no especificada], 2016; p. 779–788. [citado el 20 de agosto de 2025]. Disponible en: https://doi.org/10.1109/CVPR.2016.91

BOCHKOVSKIY, A.; WANG, C.-Y.; LIAO, H.-Y. M. YOLOv4: Optimal Speed and Accuracy of Object Detection. [Preprint]. En: arXiv preprint. [en línea]. [Localidad no especificada]: arXiv, 2020. [citado el 20 de agosto de 2025]. Disponible en: https://arxiv.org/abs/2004.10934

ULTRALYTICS. YOLOv11 Models: Deployment on Edge Devices. [Documentación en línea]. [Localidad no especificada]: Ultralytics, 2025. [citado el 23 de agosto de 2025]. Disponible en: https://docs.ultralytics.com/models/yolo11/#can-yolo11-be-deployed-on-edge-devices

DESAI, M.; MEWADA, H.; PIRES, I. M.; ROY, S. End-to-end YOLO Implementation. En: Proceedings of Computer Science. [en línea]. [Localidad no especificada], 2024; p. [Páginas del artículo]. [citado el 2 de septiembre de 2025]. Disponible en: https://doi.org/10.1016/j.procs.2024.11.096

FREE SOFTWARE FOUNDATION. GNU General Public License version 2. [Documento legal en línea]. Boston, Massachusetts: Free Software Foundation, 2007. [citado el 22 de agosto de 2025]. Disponible en: https://www.gnu.org/licenses/old-licenses/gpl-2.0.html

FREE SOFTWARE FOUNDATION. GNU General Public License version 3. [Documento legal en línea]. Boston, Massachusetts: Free Software Foundation, 2007. [citado el 22 de agosto de 2025]. Disponible en: https://www.gnu.org/licenses/gpl-3.0.html

ULTRALYTICS. Ultralytics Open-Source License. [Documento legal en línea]. [Localidad no especificada]: GitHub, 2023. [citado el 22 de agosto de 2025]. Disponible en: https://github.com/ultralytics/ultralytics/blob/main/LICENSE

MISTRAL AI. Announcing Pixtral 12B. [Anuncio en línea]. [Localidad no especificada]: Mistral AI, 17 de septiembre de 2024. [citado el 26 de agosto de 2025]. Disponible en: https://mistral.ai/news/pixtral-12b

LLM STATS. Pixtral-12B: Pricing, Context Window, Benchmarks, and More. [Página web en línea]. [Localidad no especificada]: LLM Stats, 2024. [citado el 26 de agosto de 2025]. Disponible en: https://llm-stats.com/models/pixtral-12b-2409

PERENS, B. Understanding Open Source Licenses: Apache 2.0 and Beyond. En: Open Source Initiative. [Documento en línea]. [Localidad no especificada]: Open Source Initiative, 2023. [citado el 26 de agosto de 2025]. Disponible en: https://opensource.org/licenses/Apache-2.0

ULTRALYTICS. Computer Vision Tasks supported by Ultralytics YOLO11. [en línea]. [Localidad no especificada]: Ultralytics, 30 de marzo de 2025. [citado el 26 de agosto de 2025]. Disponible en: https://docs.ultralytics.com/es/tasks/

DATACAMP. Cómo funcionan los transformadores: Una exploración detallada. [en línea]. [Localidad no especificada]: DataCamp, 11 de septiembre de 2024. [citado el 27 de agosto de 2025]. Disponible en: https://www.datacamp.com/es/tutorial/how-transformers-work

WAGH, A. What's new in GPT-4: Architecture and Capabilities. En: Medium. [Artículo en línea]. [Localidad no especificada]: Medium, 2 de julio de 2023. [citado el 27 de agosto de 2025]. Disponible en: https://medium.com/@amol-wagh/whats-new-in-gpt-4-an-overview-of-the-gpt-4-architecture-and-capabilities-of-next-generation-ai-900c445d5ffe

HUGHES, John; ATKINS, Lawrence. GPT-4: How does it work? En: Speechmatics Blog. [Artículo en línea]. [Localidad no especificada]: Speechmatics, 14 de marzo de 2023. [citado el 27 de agosto de 2025]. Disponible en: https://www.speechmatics.com/company/articles-and-news/gpt-4-how-does-it-work

OPENAI. Pricing. [Página web en línea]. [Localidad no especificada]: OpenAI, 2024. [citado el 27 de agosto de 2025]. Disponible en: https://openai.com/pricing

OPENAI. Introducing GPT-4.1 in the API. [Documento en línea]. [Localidad no especificada]: OpenAI, 14 de abril de 2025. [citado el 27 de agosto de 2025]. Disponible en: https://openai.com/index/gpt-4-1/

OPENAI. Terms of use. [Documento legal en línea]. [Localidad no especificada]: OpenAI, 2024. [citado el 28 de agosto de 2025]. Disponible en: https://openai.com/terms

OPENAI. Vision: Image inputs. En: OpenAI Platform Documentation. [Documentación en línea]. [Localidad no especificada]: OpenAI, 2024. [citado el 1 de septiembre de 2025]. Disponible en: https://platform.openai.com/docs/guides/vision

JOSEFSSON, S. The Base16, Base32, and Base64 Data Encodings. RFC 4648 [en línea]. Internet Engineering Task Force (IETF), octubre 2006. [Consultado: 16 septiembre 2025]. Disponible en: https://www.rfc-editor.org/rfc/rfc4648

PYTHON SOFTWARE FOUNDATION. base64 — Base16, Base32, Base64, Base85 Data Encodings. En: Python Documentation [en línea]. 2024. [Consultado: 16 septiembre 2025]. Disponible en: https://docs.python.org/3/library/base64.html

MASINTER, L. The "data" URL scheme. RFC 2397 [en línea]. Internet Engineering Task Force (IETF), agosto 1998. [Consultado: 16 septiembre 2025]. Disponible en: https://www.rfc-editor.org/rfc/rfc2397

JONES, M. JSON Web Signature (JWS). RFC 7515 [en línea]. Internet Engineering Task Force (IETF), mayo 2015. [Consultado: 16 septiembre 2025]. Disponible en: https://www.rfc-editor.org/rfc/rfc7515

IETF (Internet Engineering Task Force). RFC 7519: JSON Web Token (JWT). [Norma en línea]. [Localidad no especificada]: IETF, 2015. [citado el 30 de septiembre de 2025]. Disponible en: https://www.rfc-editor.org/rfc/rfc7519

VAN ROSSUM, G.; DRAKE, F. The Python Language Reference Manual. [Manual técnico]. [Localidad no especificada]: Python Software Foundation, 2023. [citado el 30 de septiembre de 2025].

PYTHON SOFTWARE FOUNDATION. Flask Documentation. [Documentación en línea]. [Localidad no especificada]: Python Software Foundation, 2024. [citado el 1 de octubre de 2025]. Disponible en: https://flask.palletsprojects.com

ARTIFEX SOFTWARE. PyMuPDF Documentation. [Documentación en línea]. [Localidad no especificada]: PyMuPDF, 2024. [citado el 1 de octubre de 2025]. Disponible en: https://pymupdf.readthedocs.io/

ANEXOS

ACTA DE ENTREVISTA

Proyecto: *Prototipo para el Reconocimiento de Firmas Manuscritas en Documentos Digitales*

Fecha: 22 de mayo de 2025

Hora: 10:30 a.m.

Modalidad: Virtual (Google Meet)

Duración: 40 min

Entrevistador:
Lina Maria Mancera Peña - Estudiante de Ingeniería Telemática, Universidad Distrital Francisco José de Caldas.

Entrevistado:
María Isabel Cuellar Izquierdo– Coordinadora de Proyectos de Software, Eprocess Company S.A.S.

Objetivo de la entrevista:

Recolectar información sobre las necesidades y problemas actuales relacionados con la gestión y validación de firmas manuscritas en documentos digitales en la organización, para justificar la implementación del Prototipo para el Reconocimiento de Firmas Manuscritas en Documentos Digitales.

Preguntas realizadas y respuestas resumidas:

1. **¿Cuál es el proceso actual para verificar que un documento digital tenga firma?**
   Maria Cuellar: *Actualmente, el proceso es manual, lo que implica revisar cada documento visualmente por parte del personal administrativo para confirmar la presencia de la firma, lo cual consume mucho tiempo y es propenso a errores.*

2. **¿Qué tipo de documentos deben contener firma manuscrita?**
   Maria Cuellar: Contratos, autorizaciones de pago, certificaciones laborales, y solicitudes internas.

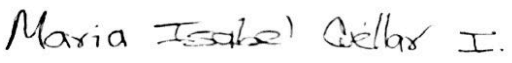
3. **¿Cuántos documentos con firma se procesan mensualmente?**
   Maria Cuellar: Aproximadamente entre 5000 documentos al mes, incluso más en ciertos periodos internos de gestión.

4. **¿Cuáles son las principales dificultades que enfrentan con este proceso manual?**
   María Cuellar: El proceso es lento, especialmente cuando hay grandes volúmenes de documentos. Además, la posibilidad de errores en la revisión puede generar problemas retrasos en trámites importantes. También los costos operativos son elevados porque requiere mucho personal dedicado a esta tarea y reprocesos.

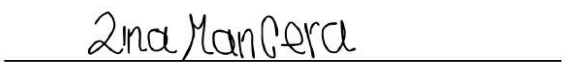
5. **¿Considera útil automatizar la detección de firmas?**
   Sí, sería de gran ayuda, contar con un prototipo *que verifique automáticamente si un documento contiene o no una firma manuscrita antes de pasar al siguiente paso del flujo nos permite reducir errores, también facilitará el manejo de grandes volúmenes de documentos y optimizar los recursos. Además que centra bases para futuras mejoras e implementaciones sobre el mismo prototipo, esto nos abre una puerta para automatizar un proceso muy tedioso para el equipo administrativo.*

Conclusión:

Durante la entrevista se identificó que Eprocess S.A.S. tiene interés en desarrollar un prototipo de reconocimiento automático de firmas manuscritas, en respuesta a una necesidad real detectada. Esta validación sustenta y justifica el enfoque del proyecto propuesto, orientado al desarrollo de una solución tecnológica que permita, a través de un servicio web, recibir un documento digital y retornar si contiene o no una firma manuscrita.



Firma del entrevistado:
Maria Isabel Cuellar Izquierdo
Coordinadora de Proyectos de Software
Eprocess Company S.A.S.
maria.cuellar@eprocess.com.co



Firma del entrevistador:
Lina Maria Mancera Peña
Estudiante de Ingenieria Telematica
Universidad Distrital Francisco José de Caldas.
Linamancera99@gmail.com