

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA (ICAI)

MÁSTER EN BIG DATA: TECNOLOGÍA Y ANALÍTICA AVANZADA

AiValidAi: Validador Automático Inteligente mediante Sistemas Multi-Agente para el Control de Calidad en Tasaciones Inmobiliarias

Autor: Natalia Jiménez Calvo

Director: Ángel Guzman Torres

Madrid

Junio 2026

Declaro, bajo mi responsabilidad, que el Proyecto presentado con el título

TÍTULO DEL PROYECTO

en la ETS de Ingeniería - ICAI de la Universidad Pontificia Comillas en el

curso académico 2018/19 es de mi autoría, original e inédito y

no ha sido presentado con anterioridad a otros efectos.

El Proyecto no es plagio de otro, ni total ni parcialmente y la información que ha sido

tomada de otros documentos está debidamente referenciada.

Natalia Jimenez Calvo

Fdo.: Nombre del alumno Fecha: 22 / 06 / 2026

Autorizada la entrega del proyecto

EL DIRECTOR DEL PROYECTO

Fdo.: Nombre del Director Fecha: ……/ ……/ ……

Vº Bº del Coordinador de Proyectos

Firmado por MORRAS RUIZ-FALCO CARLOS - ***0907** el día 29/06/2026 con un certificado emitido por AC FNMT Usuarios

Fdo.: Carlos Morras Ruiz-Falco Fecha: ……/ ……/ ……

AUTORIZACIÓN PARA LA DIGITALIZACIÓN, DEPÓSITO Y DIVULGACIÓN EN RED DE PROYECTOS FIN DE GRADO, FIN DE MÁSTER, TESINAS O MEMORIAS DE BACHILLERATO

1º. Declaración de la autoría y acreditación de la misma.

El autor D. Natalia Jimenez Calvo

DECLARA ser el titular de los derechos de propiedad intelectual de la obra: TFM AIVALIDAI, que ésta es una obra original, y que ostenta la condición de autor en el sentido que otorga la Ley de Propiedad Intelectual.

2º. Objeto y fines de la cesión.

Con el fin de dar la máxima difusión a la obra citada a través del Repositorio institucional de la Universidad, el autor **CEDE** a la Universidad Pontificia Comillas, de forma gratuita y no exclusiva, por el máximo plazo legal y con ámbito universal, los derechos de digitalización, de archivo, de reproducción, de distribución y de comunicación pública, incluido el derecho de puesta a disposición electrónica, tal y como se describen en la Ley de Propiedad Intelectual. El derecho de transformación se cede a los únicos efectos de lo dispuesto en la letra a) del apartado siguiente.

3º. Condiciones de la cesión y acceso

Sin perjuicio de la titularidad de la obra, que sigue correspondiendo a su autor, la cesión de derechos contemplada en esta licencia habilita para:

a) Transformarla con el fin de adaptarla a cualquier tecnología que permita incorporarla a internet y hacerla accesible; incorporar metadatos para realizar el registro de la obra e incorporar "marcas de agua" o cualquier otro sistema de seguridad o de protección.
b) Reproducirla en un soporte digital para su incorporación a una base de datos electrónica, incluyendo el derecho de reproducir y almacenar la obra en servidores, a los efectos de garantizar su seguridad, conservación y preservar el formato.
c) Comunicarla, por defecto, a través de un archivo institucional abierto, accesible de modo libre y gratuito a través de internet.
d) Cualquier otra forma de acceso (restringido, embargado, cerrado) deberá solicitarse expresamente y obedecer a causas justificadas.
e) Asignar por defecto a estos trabajos una licencia Creative Commons.
f) Asignar por defecto a estos trabajos un HANDLE (URL *persistente)*.

4º. Derechos del autor.

El autor, en tanto que titular de una obra tiene derecho a:

a) Que la Universidad identifique claramente su nombre como autor de la misma
b) Comunicar y dar publicidad a la obra en la versión que ceda y en otras posteriores a través de cualquier medio.
c) Solicitar la retirada de la obra del repositorio por causa justificada.
d) Recibir notificación fehaciente de cualquier reclamación que puedan formular terceras personas en relación con la obra y, en particular, de reclamaciones relativas a los derechos de propiedad intelectual sobre ella.

5º. Deberes del autor.

- El autor se compromete a:

a) Garantizar que el compromiso que adquiere mediante el presente escrito no infringe ningún derecho de terceros, ya sean de propiedad industrial, intelectual o cualquier otro.
b) Garantizar que el contenido de las obras no atenta contra los derechos al honor, a la intimidad y a la imagen de terceros.
c) Asumir toda reclamación o responsabilidad, incluyendo las indemnizaciones por daños, que pudieran ejercitarse contra la Universidad por terceros que vieran infringidos sus derechos e

intereses a causa de la cesión.

d) Asumir la responsabilidad en el caso de que las instituciones fueran condenadas por infracción de derechos derivada de las obras objeto de la cesión.

6º. Fines y funcionamiento del Repositorio Institucional.

La obra se pondrá a disposición de los usuarios para que hagan de ella un uso justo y respetuoso con los derechos del autor, según lo permitido por la legislación aplicable, y con fines de estudio, investigación, o cualquier otro fin lícito. Con dicha finalidad, la Universidad asume los siguientes deberes y se reserva las siguientes facultades:

- La Universidad informará a los usuarios del archivo sobre los usos permitidos, y no garantiza ni asume responsabilidad alguna por otras formas en que los usuarios hagan un uso posterior de las obras no conforme con la legislación vigente. El uso posterior, más allá de la copia privada, requerirá que se cite la fuente y se reconozca la autoría, que no se obtenga beneficio comercial, y que no se realicen obras derivadas.
- La Universidad no revisará el contenido de las obras, que en todo caso permanecerá bajo la responsabilidad exclusive del autor y no estará obligada a ejercitar acciones legales en nombre del autor en el supuesto de infracciones a derechos de propiedad intelectual derivados del depósito y archivo de las obras. El autor renuncia a cualquier reclamación frente a la Universidad por las formas no ajustadas a la legislación vigente en que los usuarios hagan uso de las obras.
- La Universidad adoptará las medidas necesarias para la preservación de la obra en un futuro.
- La Universidad se reserva la facultad de retirar la obra, previa notificación al autor, en supuestos suficientemente justificados, o en caso de reclamaciones de terceros.

Madrid, a …22….. de ……Junio……..... de 2026…

ACEPTA

Fdo……Natalia Jimenez Calvo……

Motivos para solicitar el acceso restringido, cerrado o embargado del trabajo en el Repositorio Institucional:



ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA (ICAI)

MÁSTER EN BIG DATA: TECNOLOGÍA Y ANALÍTICA AVANZADA

AiValidAi: Validador Automático Inteligente mediante Sistemas Multi-Agente para el Control de Calidad en Tasaciones Inmobiliarias

Autor: Natalia Jiménez Calvo

Director: Ángel Guzman Torres

Madrid

Junio 2026

Agradecimientos

A mi familia por siempre cuidarme y darme las mejores oportunidades y a mis amigos del master por cuidarme tanto este año.

AIVALIDAI: VALIDADOR AUTOMÁTICO INTELIGENTE MEDIANTE SISTEMAS MULTI-AGENTE PARA EL CONTROL DE CALIDAD EN TASACIONES INMOBILIARIAS

Autor: Jiménez Calvo, Natalia.
Director: Guzman Torres, Ángel.
Entidad Colaboradora: VALUM

RESUMEN DEL PROYECTO

Se presenta AIValidAI, una plataforma de validación automática de expedientes de tasación inmobiliaria basada en arquitecturas de sistemas multi-agente orquestados con LangGraph sobre Amazon Bedrock. El sistema procesa de forma autónoma los tres soportes documentales críticos de un expediente —nota simple registral, fotografías periciales y croquis arquitectónico— reduciendo el tiempo de validación de una media de 15-20 minutos por expediente a menos de 35 segundos, con una tasa de error prácticamente nula en los campos de mayor riesgo jurídico y un éxito operativo del 100% en los módulos de visión.

Palabras clave: sistemas multi-agente, LangGraph, Amazon Bedrock, tasación inmobiliaria, procesamiento documental, visión por computador, automatización de procesos

1. Introducción

El proceso de validación de un expediente de tasación inmobiliaria es, en su estado actual, un trabajo intensivo en tiempo y propenso al error humano. Un técnico validador debe abrir manualmente cada documento del expediente, leer y transcribir los datos relevantes de la nota simple registral, comprobar visualmente que el reportaje fotográfico cubre todas las estancias obligatorias y verificar que el croquis arquitectónico es legible y está en condiciones de ser procesado. Este ciclo completo consume entre 15 y 20 minutos por expediente y, en un entorno de alto volumen de tasaciones, genera cuellos de botella operativos, fatiga cognitiva y errores de transcripción que comprometen la calidad del informe final.

Los avances recientes en modelos de lenguaje de gran tamaño (LLMs) multimodales y en los marcos de orquestación de agentes cognitivos abren la posibilidad de automatizar este proceso de forma fiable. En particular, la combinación de LangGraph para la definición de flujos de trabajo deterministas y el modelo Amazon Nova Pro desplegado en Amazon Bedrock para el razonamiento y la visión ofrece una base tecnológica suficientemente madura para abordar tareas de extracción, clasificación y auditoría documental en un entorno empresarial real.

2. Definición del proyecto

AIValidAI se define como una plataforma modular de validación automática de expedientes de tasación, desarrollada en colaboración con la empresa VALUM. El objetivo principal es eliminar la intervención manual en las tareas de extracción y verificación de datos, liberando al técnico validador de la transcripción mecánica para que pueda centrarse exclusivamente en la supervisión de alertas y en el control de riesgos de alto valor.

El sistema recibe como entrada el expediente de tasación almacenado en buckets de Amazon S3 y genera como salida un conjunto de JSONs estructurados que contienen, para cada soporte documental, los datos extraídos, las discrepancias detectadas y un veredicto de validez. Estos resultados se exponen a través de una API REST y se visualizan en una interfaz web funcional que permite a cualquier usuario del equipo de consultar el estado de una validación y descargar el informe resultante.

El alcance del proyecto cubre tres pipelines independientes: el análisis de texto jurídico de la nota simple registral, la auditoría visual del reportaje fotográfico pericial y la verificación de integridad del croquis arquitectónico. Los tres operan de forma asíncrona y comparten la misma infraestructura de nube.

3. Descripción del modelo/sistema/herramienta

La arquitectura de AIValidAI se estructura en tres subsistemas especializados, cada uno con su propio patrón de diseño y modelo de razonamiento, coordinados bajo un paradigma común de agentes cognitivos sobre AWS.

El subsistema de nota simple implementa un grafo dirigido de cinco agentes en LangGraph. El flujo se inicia con un agente enrutador que clasifica el inmueble en una de ocho tipologías (residencial, comercial, industrial, terreno, finca rústica, terciario, dotacional o anejo). A partir de esa clasificación, se activa la fábrica de agentes especialistas correspondiente, que distribuye el texto extraído mediante Amazon Textract entre cinco agentes en paralelo: uno por cada bloque de información crítica (datos registrales, superficies, titularidad, cargas y anejos). Un nodo final fusiona los cinco sub-JSONs en un único esquema consolidado.

El subsistema de imágenes periciales emplea un agente de visión nativa sin herramientas externas. Las fotografías se recuperan de S3, se codifican en base64 en memoria y se transmiten directamente al modelo Amazon Nova Pro junto con las instrucciones de auditoría. El modelo devuelve un inventario estructurado de estancias y una valoración del cumplimiento fotográfico según la normativa interna de la empresa en cuanto a validación.

El subsistema de croquis actúa como agente guardián. Sigue el patrón ReAct (Reasoning and Acting), lo que le permite razonar de forma iterativa: localizar el archivo en S3, descargarlo y evaluar su legibilidad. El resultado es siempre uno de tres estados posibles: OK (documento válido y procesable), BLANCO (documento vacío o sin trazos identificables) o SIN_CROQUIS (archivo corrupto o ausente).

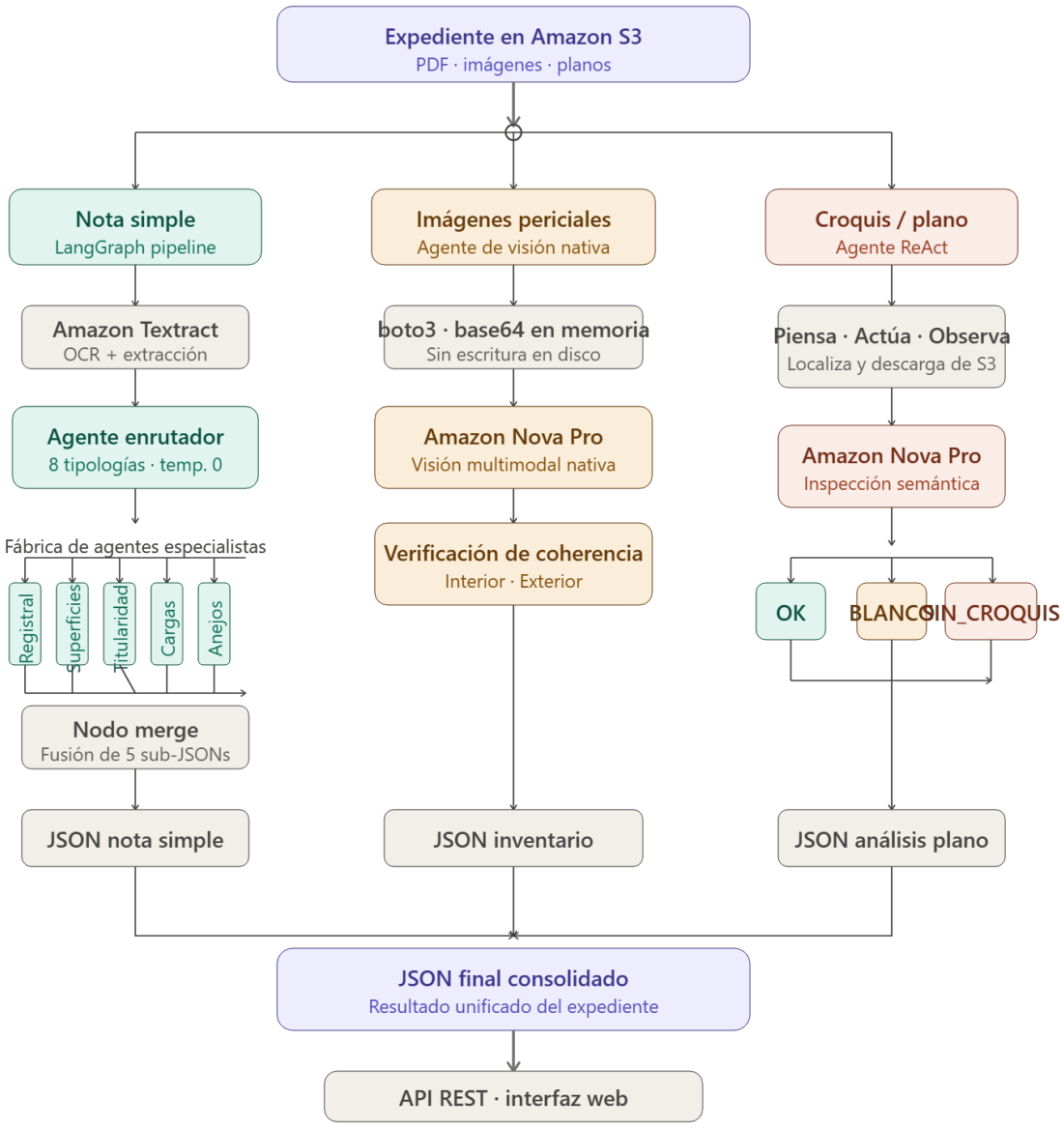


Ilustración 1- sistema completo

4. Resultados

El sistema se evaluó sobre un lote de diez expedientes de tasación reales proporcionados por VALUM, representativos del volumen operativo habitual de la empresa: siete activos residenciales, dos industriales, un terreno y un elemento anejo.

- El pipeline de notas simples extrajo con exactitud total los campos de mayor riesgo jurídico en todos los casos: IDUFIR/CRU, número de cargas registrales y superficies construidas. Los únicos desajustes detectados se produjeron en campos de texto libre de bajo valor, errores que un técnico puede corregir visualmente en segundos.

- Los subsistemas de visión (imágenes y croquis) registraron un 100% de éxito operativo en todos los casos de prueba, incluyendo los documentos de control introducidos deliberadamente en estado corrupto o vacío.

Para cuantificar el impacto operativo, la tabla siguiente contrasta el rendimiento del proceso manual frente al pipeline automatizado:

Tabla 1-Resultados del impacto operativo

Tabla 2- Resultados generales

Metric	Manual Process	Automated Pipeline
Validation time	15-20 minutes/file	< 35 seconds/file
Error rate	High (transcription errors)	Virtually zero
Operational success	Variable (cognitive fatigue)	100%
Availability	Business hours	24/7
Files per hour	3-4	100
Time per 1000 files	~290 hours	~10 hours

5. Conclusiones

AIValidAI demuestra que es técnicamente viable automatizar el ciclo completo de validación de un expediente de tasación con un nivel de fiabilidad suficiente para su adopción en un entorno empresarial real. La arquitectura de agentes especialistas por tipología, frente a un modelo genérico único, ha sido la decisión de diseño más determinante para alcanzar la precisión observada: al acotar el contexto de cada agente

a una sola regla de negocio, se elimina la ambigüedad y se reducen las posibles alucinaciones del modelo.

El impacto operativo más relevante no es el ahorro de tiempo en sí, sino la transformación del rol del técnico validador: la plataforma convierte una tarea mecánica de transcripción en una tarea de supervisión crítica, lo que reduce la fatiga cognitiva y permite que el equipo humano se concentre en los casos que realmente requieren criterio experto.

Como líneas de trabajo futuro se proponen tres ejes: la extensión de los pipelines para cubrir el análisis de mercado testigo y el cálculo del valor de reemplazo; la implementación de un agente supervisor que realice validación cruzada entre los tres subsistemas (confrontando los metros cuadrados de la nota simple con las cotas del croquis y las estancias del reportaje fotográfico); y la exploración de modelos más ligeros mediante fine-tuning específico en lenguaje registral español para reducir aún más la latencia y el coste por llamada.

6. Referencias

[1] Amazon Web Services. (2023). *Amazon Textract Developer Guide: Intelligent Document Processing (IDP) for text extraction*. AWS Documentation. https://docs.aws.amazon.com/textract/

[2] Amazon Web Services. (2024a). *Amazon Bedrock User Guide: Accessing foundational models securely*. AWS Documentation. https://docs.aws.amazon.com/bedrock/

[3] Amazon Web Services. (2024b). *Amazon Nova Pro: Technical specifications and multimodal capabilities*. AWS Model Repository.

[4] Banco de España. (1981). *Orden de 18 de diciembre de 1981 sobre homologación de sociedades de tasación y regulación de los métodos de valoración*. Boletín Oficial del Estado, nº 312.

[5] Boto3 Developers. (2024). *Boto3 Docs v1.34: AWS SDK for Python Reference*. Amazon Web Services. https://boto3.amazonaws.com/v1/documentation/api/latest/index.html

[6] Chase, H. (2022). *LangChain: Framework for developing applications powered by language models* (Versión 0.1) [Software de computación]. GitHub. https://github.com/langchain-ai/langchain

[7] Colegio de Registradores de la Propiedad y Mercantiles de España. (2022). *La Nota Simple Registral: Estructura, valor jurídico y pautas de interpretación en el tráfico inmobiliario*. Centro de Estudios Registrales.

[8] Express Team. (2023). *Express: Fast, unopinionated, minimalist web framework for Node.js* (Versión 4.18) [Software de computación]. https://expressjs.com

[9] Herrero Alcántara, T. (2014). "Big Data: ¿Moda u oportunidad de negocio para el emprendedor?", Think Big, Octubre 2014. http://blogthinkbig.com/big-data-emprendedor/

[10] LangChain AI. (2024). *LangGraph: Building cyclic and stateful multi-agent workflows*. LangChain Documentation. https://langchain-ai.github.io/langgraph/

[11] Loeffler, B. (2011). "Cloud Computing: What is Infrastructure as a Service", Microsoft Technet Magazine, October 2011. https://technet.microsoft.com/en-us/magazine/hh509051.aspx

[12] Ministerio de Economía y Hacienda. (2003). *Orden ECO/805/2003, de 27 de marzo, sobre normas de valoración de bienes inmuebles y de determinados derechos para ciertas finalidades financieras*. Boletín Oficial del Estado, nº 85.

[13] React Community. (2022). *React: A JavaScript library for building user interfaces* (Versión 18) [Software de computación]. https://react.dev

[14] Vite Core Team. (2023). *Vite: Next Generation Frontend Tooling* (Versión 5) [Software de computación]. https://vitejs.dev

[15] Yao, S., Yu, D., Zhao, J., Shafran, I., Griffiths, T. L., Cao, Y., & Narasimhan, K. (2022). *ReAct: Synergizing reasoning and acting in language models*. arXiv preprint arXiv:2210.03629. https://doi.org/10.48550/arXiv.2210.03629

AIVALIDAI: INTELLIGENT AUTOMATED VALIDATOR USING MULTI-AGENT SYSTEMS FOR QUALITY CONTROL IN REAL ESTATE APPRAISALS

Author: Jiménez Calvo, Natalia.
Supervisor: Guzman Torres, Ángel.
Collaborating Entity: VALUM.

ABSTRACT

AIValidAI is presented as an automatic validation platform for real estate appraisal files, based on multi-agent system architectures orchestrated with LangGraph on Amazon Bedrock. The system autonomously processes the three critical documentary supports of an appraisal file — the registry note, the expert photographs and the architectural sketch — reducing the validation time from an average of 15-20 minutes per file to less than 35 seconds, with a practically zero error rate in the highest legal-risk fields and a 100% operational success rate in the vision modules.

Keywords: multi-agent systems, LangGraph, Amazon Bedrock, real estate appraisal, document processing, computer vision, process automation.

1. Introduction

The process of validating a real estate appraisal file is, in its current state, a time-intensive task prone to human error. A validation technician must manually open each document in the file, read and transcribe the relevant data from the registry note, visually check that the photographic report covers all mandatory rooms, and verify that the architectural sketch is legible and ready to be processed. This complete cycle takes between 15 and 20 minutes per file and, in a high-volume appraisal environment, creates operational bottlenecks, cognitive fatigue and transcription errors that compromise the quality of the final report.

Recent advances in multimodal large language models (LLMs) and cognitive agent orchestration frameworks open up the possibility of automating this process reliably. In particular, the combination of LangGraph for defining deterministic workflows and the Amazon Nova Pro model deployed on Amazon Bedrock for reasoning and vision offers a sufficiently mature technological foundation to tackle document extraction, classification and auditing tasks in a real business environment.

2. Definition of the project

AIValidAI is defined as a modular automatic validation platform for appraisal files, developed in collaboration with the company VALUM. The main objective is to eliminate manual intervention in data extraction and verification tasks, freeing the

validation technician from mechanical transcription so they can focus exclusively on alert supervision and high-value risk control.

The system receives as input the appraisal file stored in Amazon S3 buckets and generates as output a set of structured JSONs containing, for each documentary support, the extracted data, the discrepancies detected and a validity verdict. These results are exposed through a REST API and visualised in a functional web interface that allows any team member to check the status of a validation and download the resulting report.

The scope of the project covers three independent pipelines: the legal text analysis of the registry note, the visual audit of the expert photographic report and the integrity verification of the architectural sketch. All three operate asynchronously and share the same cloud infrastructure.

3. Descripción del modelo/sistema/herramienta

The architecture of AIValidAI is structured into three specialised subsystems, each with its own design pattern and reasoning model, coordinated under a common paradigm of cognitive agents on AWS.

The registry note subsystem implements a directed graph of five agents in LangGraph. The flow begins with a router agent that classifies the property into one of eight typologies (residential, commercial, industrial, land, rural estate, tertiary, public facility or annex). From that classification, the corresponding specialist agent factory is activated, distributing the text extracted via Amazon Textract among five parallel agents: one for each block of critical information (registry data, surface areas, ownership, encumbrances and annexes). A final node merges the five sub-JSONs into a single consolidated schema.

The expert image subsystem uses a native vision agent without external tools. The photographs are retrieved from S3, encoded in base64 in memory and transmitted directly to the Amazon Nova Pro model along with the audit instructions. The model returns a structured inventory of rooms and an assessment of photographic compliance according to VALUM's internal regulations.

The sketch subsystem acts as a guardian agent. It follows the ReAct pattern (Reasoning and Acting), which allows it to reason iteratively: locating the file in S3, downloading it and evaluating its legibility. The result is always one of three possible states: OK (valid and processable document), BLANK (empty document or no identifiable strokes) or NO_SKETCH (corrupted or missing file).

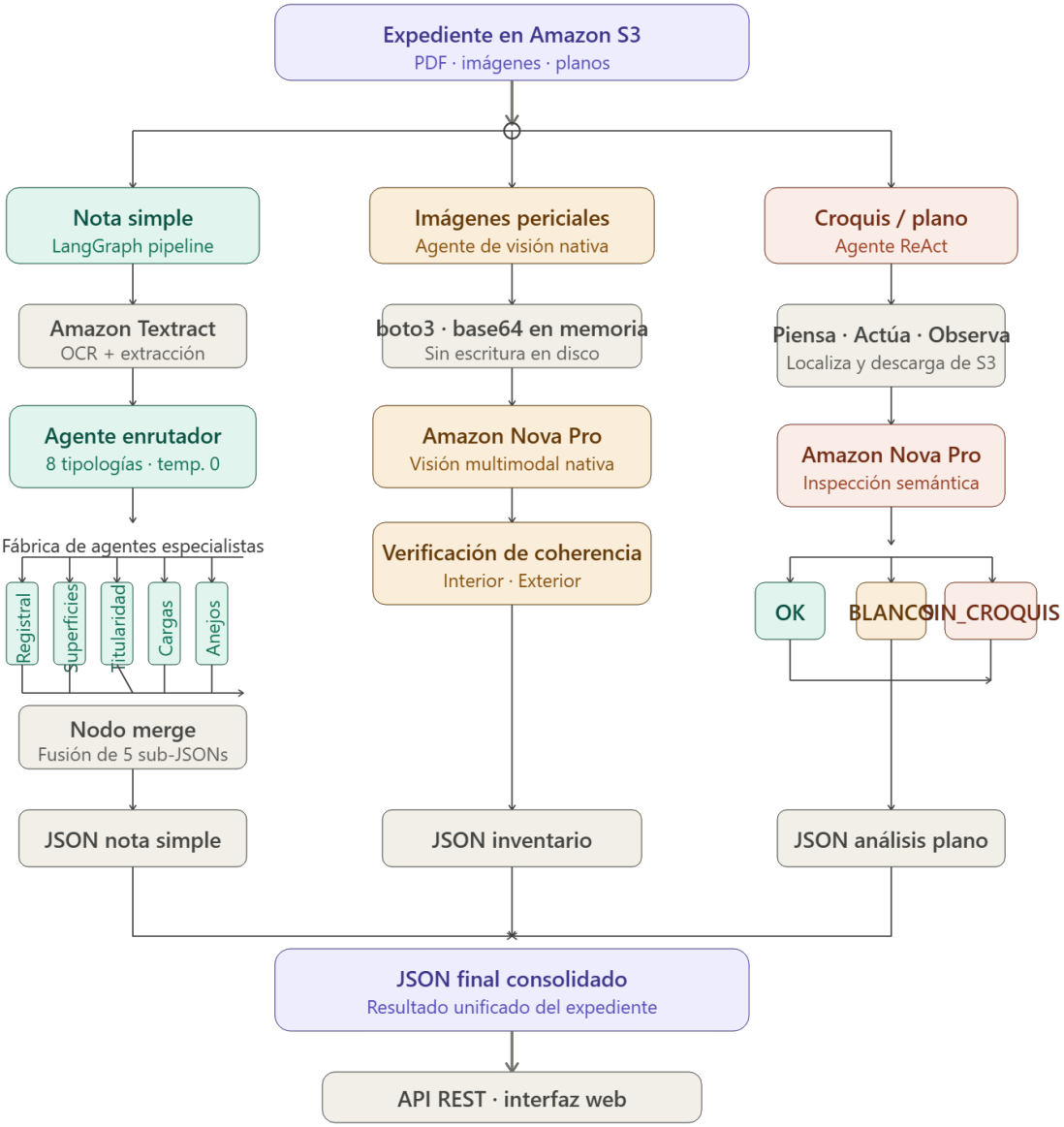


Ilustración 2- sistema completo

4. Results

The system was evaluated on a batch of ten real appraisal files provided by VALUM, representative of the company's usual operational volume: seven residential assets, two industrial, one plot of land and one annex element.

The registry note pipeline extracted with complete accuracy the highest legal-risk fields in all cases: IDUFIR/CRU, number of registered encumbrances and built surface areas. The only mismatches detected occurred in low-value free-text fields — errors that a technician can correct visually in seconds.

The vision subsystems (images and sketches) recorded a 100% operational success rate in all test cases, including the control documents deliberately introduced in a corrupted or empty state.

To quantify the operational impact, the following table contrasts the performance of the manual process against the automated pipeline:

Métrica	Proceso Manual	Pipeline Automatizado
Validation time	15-20 minutos/expediente	< 35 segundos/expediente
Tasa de error	Alta (errores de transcripción)	Prácticamente nula
Éxito operativo	Variable (fatiga cognitiva)	100%
Disponibilidad	Horario comercial	24/7
Expedientes por hora	3-4	100
Tiempo por 1000 expedientes	~290 horas	~10 horas

Tabla 3-Resultados generales

5. Conclusions

AIValidAI demonstrates that it is technically feasible to automate the complete validation cycle of a property appraisal file with a level of reliability sufficient for adoption in a real business environment. The architecture of specialist agents by property type, as opposed to a single generic model, has been the most decisive design decision in achieving the observed precision: by limiting each agent's context to a single business rule, ambiguity is eliminated and potential model hallucinations are reduced.

The most relevant operational impact is not the time saving itself, but the transformation of the validator's role: the platform converts a mechanical transcription task into a critical supervision task, reducing cognitive fatigue and allowing the human team to focus on the cases that genuinely require expert judgment.

Three areas of future work are proposed: the extension of the pipelines to cover comparable market analysis and replacement value calculation; the implementation of a supervisor agent that performs cross-validation between the three subsystems (comparing the square metres from the registry note with the dimensions in the architectural sketch and the rooms in the photographic report); and the exploration of lighter models through fine-tuning specifically on Spanish registry language to further reduce latency and cost per call.

6. Referencias

[1] Amazon Web Services. (2023). *Amazon Textract Developer Guide: Intelligent Document Processing (IDP) for text extraction*. AWS Documentation. https://docs.aws.amazon.com/textract/

[2] Amazon Web Services. (2024a). *Amazon Bedrock User Guide: Accessing foundational models securely*. AWS Documentation. https://docs.aws.amazon.com/bedrock/

[3] Amazon Web Services. (2024b). *Amazon Nova Pro: Technical specifications and multimodal capabilities*. AWS Model Repository.

[4] Banco de España. (1981). *Orden de 18 de diciembre de 1981 sobre homologación de sociedades de tasación y regulación de los métodos de valoración*. Boletín Oficial del Estado, nº 312.

[5] Boto3 Developers. (2024). *Boto3 Docs v1.34: AWS SDK for Python Reference*. Amazon Web Services. https://boto3.amazonaws.com/v1/documentation/api/latest/index.html

[6] Chase, H. (2022). *LangChain: Framework for developing applications powered by language models* (Versión 0.1) [Software de computación]. GitHub. https://github.com/langchain-ai/langchain

[7] Colegio de Registradores de la Propiedad y Mercantiles de España. (2022). *La Nota Simple Registral: Estructura, valor jurídico y pautas de interpretación en el tráfico inmobiliario*. Centro de Estudios Registrales.

[8] Express Team. (2023). *Express: Fast, unopinionated, minimalist web framework for Node.js* (Versión 4.18) [Software de computación]. https://expressjs.com

[9] Herrero Alcántara, T. (2014). "Big Data: ¿Moda u oportunidad de negocio para el emprendedor?", Think Big, Octubre 2014. http://blogthinkbig.com/big-data-emprendedor/

[10] LangChain AI. (2024). *LangGraph: Building cyclic and stateful multi-agent workflows*. LangChain Documentation. https://langchain-ai.github.io/langgraph/

[11] Loeffler, B. (2011). "Cloud Computing: What is Infrastructure as a Service", Microsoft Technet Magazine, October 2011. https://technet.microsoft.com/en-us/magazine/hh509051.aspx

[12] Ministerio de Economía y Hacienda. (2003). *Orden ECO/805/2003, de 27 de marzo, sobre normas de valoración de bienes inmuebles y de determinados derechos para ciertas finalidades financieras*. Boletín Oficial del Estado, nº 85.

[13] React Community. (2022). *React: A JavaScript library for building user interfaces* (Versión 18) [Software de computación]. https://react.dev

[14] Vite Core Team. (2023). *Vite: Next Generation Frontend Tooling* (Versión 5) [Software de computación]. https://vitejs.dev

[15] Yao, S., Yu, D., Zhao, J., Shafran, I., Griffiths, T. L., Cao, Y., & Narasimhan, K. (2022). *ReAct: Synergizing reasoning and acting in language models*. arXiv preprint arXiv:2210.03629. https://doi.org/10.48550/arXiv.2210.03629

Índice de la memoria

Capítulo 1. Introducción **4**
1.1 Motivación del proyecto 4
1.2 DESCRIPCIÓN DE LAS TECNOLOGIAS 4
1.2.1 Entorno de Desarrollo y Frameworks de IA 5
1.2.2 Servicios en la Nube e Inteligencia Computacional (AWS) 5
1.2.3 Capa de Presentación (Prototipo Web) 6
1.3 Apartado de ejemplo. Cómo documentar un trabajo 7
1.3.1 Ejemplo de cómo documentar el trabajo 7
1.3.2 Figuras 7
1.3.3 Elementos de numeración 9
1.3.4 Tablas 9
1.3.5 Código fuente de programas 9
1.3.6 Actualización de campos 10
1.3.7 Pie de página y encabezado 10
Capítulo 2. Definición del Trabajo **11**
2.1 Objetivos 11
2.2 Metodología 12
2.3 Planificación y Estimación Económica 12
Capítulo 3. Estado del arte **14**
3.1 Modelos de Lenguaje Aplicados al Procesamiento Inteligente de Documentos (IDP) 14
3.2 Document Intelligence en el Sector Financiero e Inmobiliario 16
3.3 Evolución de Frameworks y Paradigmas de Sistemas Multi-Agente 17
3.4 OCR e IDP (Intelligent Document Processing): AWS Textract vs alternativas 18
Capítulo 4. Marco teorico **20**
4.1 Arquitecturas de Agentes: ReAct, Reflexion y Plan-and-Execute 20
4.2 Estrategias de inferencia: prompting estructurado vs rag y fine-tuning 21
4.3 orquestación de Workflows y Ecosistema de Desarrollo (LangChain y LangGraph) 21
4.3.1 el rol de langchain: componentes atómicos y abstracción 22
4.3.2 el rol de LangGraph: Estructura de Control y Grafos Dirigidos 22
Capítulo 5. Sistema/Modelo Desarrollado **24**
5.1 Análisis del Sistema 24
5.2 diseño 26
5.2.1 PIPELINE DE la nota simple (orquestacion con langraph) 26

5.2.2 agente de Visión para Imágenes Periciales (agent_imagen.py) 28
5.2.3 análisis de croquis y planos (agent_croquis.py) 29
5.3 Implementación tecnológica 30
5.3.1 librerías de integración de ia y orquestación de agentes 30
5.3.3 Consumo de Modelos Fundacionales e Integración con AWS 30
Capítulo 6. Implementacion 32
6.1 Estructura del Código Fuente y Modularidad 32
6.2 Pipeline de Notas Simples 34
6.3 Pipeline de Croquis y Planos (agent_croquis.py) 38
6.4 Pipeline de Imágenes Periciales (Imagen_analyzer.py) 38
Capítulo 7. Interfaz Web de Validación (Prototipo Funcional) 41
7.1 Objetivo y contexto 41
7.2 Arquitectura técnica 42
7.3 Estructura de rutas y flujo de usuario 43
7.4 Descripción de pantallas 44
7.4.1 Panel de control (Bandeja) 44
7.4.2 Nueva Validación 45
7.4.3 Procesando 45
7.4.4 Resultado 46
7.5 Cliente HTTP y comunicación con el BFF 47
Capítulo 8. análisis de resultados 48
8.1 Dataset de pruebas y criterio de priorización 48
8.2 Resultados del Pipeline de Notas Simples 49
8.3 Resultados de los Pipelines Gráficos (Croquis e Imágenes) 52
Capítulo 9. Discusión 54
9.1 Fortalezas y Datos Críticos de Alta Fiabilidad 54
9.2 éxito absoluto en croquis e imágenes 54
9.2.1 errores minimos y de bajo valor en notas simples 55
9.2.2 el factor de la redacción registral 55
9.3 evaluacion de la viabilidad temporal y tolerancia a fallos 56
9.3.1 Optimización del Tiempo de Procesamiento 56
9.3.2 reducción drastica frente al proceso manual 56
9.4 impacto operativo y retorno de la inversion (ROI) en valum 57
Conclusión del Análisis 59
9.5 Limitaciones del Sistema y Desafíos de Integración 59
Capítulo 10. Conclusiones y Trabajos Futuros 61
10.1 Conclusiones del proyecto 61
10.2 Líneas de futuro 62
10.2.1 automatización integral del flujo de tasación 62
10.2.2 Validación cruzada e inteligencia de negocio 62

10.2.3 optimización temporal e inteligencia de modelos 63
Capítulo 11. Bibliografía 64
ANEXO A 65
ANEXO A — Función de Clasificación Tipológica (router_agents.py) 65
ANEXO B — Fábrica de Agentes Especialistas: Ejemplo Residencial (simple_notes_agent_residencial.py) 67

Índice de ilustraciones

Ilustración 1- sistema completo 13
Ilustración 2- sistema completo 19
Ilustración 3- sistema de AIValidAI 24
Ilustración 4- sistema de agentes para la nota simple 33
Ilustración 5- Bandeja de entrada 44

Índice de tablas

Tabla 1-Resultados del impacto operativo 14
Tabla 2- Resultados generales 14
Tabla 3-Resultados generales 20
Tabla 4 Desglose tecnológico y despliegue de la arquitectura de la interfaz web 41
Tabla 5- Definición de rutas y flujo de navegación de la interfaz web 42
Tabla 6- Resumen de resultados y métricas de rendimiento del pipeline de notas simples.50

Capítulo 1. INTRODUCCIÓN

1.1 MOTIVACIÓN DEL PROYECTO

El sector de la tasación inmobiliaria es un proceso que requiere la revisión y validación de múltiples documentos e información gráfica antes de emitir un informe final. En empresas como VALUM, este proceso recae sobre validadores internos que deben comprobar manualmente tres elementos clave por cada tasación: la nota simple del inmueble, las imágenes aportadas y el croquis o plano.

Esta validación manual presenta varios problemas: es lenta, susceptible a errores de interpretación y dificulta la doble comprobación cuando un segundo validador revisa el trabajo del primero. Con el crecimiento del volumen de tasaciones, la carga operativa sobre el equipo humano se incrementa sin que ello suponga necesariamente una mejora en la calidad del proceso.

AIValidAI nace para dar respuesta a este problema. Se trata de un sistema de agentes de inteligencia artificial capaz de analizar automáticamente los tres elementos de una tasación —nota simple, imágenes e inmueble y croquis— y emitir un veredicto sobre su validez, señalando los errores concretos cuando los haya. El objetivo no es sustituir al validador humano, sino asistirle, reducir los tiempos de revisión y garantizar una mayor consistencia en los criterios de calidad aplicados.

1.2 DESCRIPCIÓN DE LAS TECNOLOGIAS

Para la construcción de la plataforma **AIValidAI**, se ha diseñado un ecosistema tecnológico moderno, robusto y desacoplado, dividiendo las herramientas en tres capas principales: infraestructura *Cloud*, lógica de Inteligencia Artificial y capa de presentación.

1.2.1 ENTORNO DE DESARROLLO Y FRAMEWORKS DE IA

- **Python 3:** Es el lenguaje de programación troncal del sistema debido a su madurez, rendimiento y compatibilidad nativa con las librerías líderes de inteligencia artificial.
- **LangChain:** Utilizado como el framework de abstracción de bajo nivel. Proporciona los bloques fundamentales para gestionar el protocolo de mensajes, la estructuración de instrucciones (*SystemMessage* y *HumanMessage*) y la conversión de funciones de Python en herramientas operativas (@tool) que los modelos pueden ejecutar.
- **LangGraph:** Extensión crítica de LangChain empleada para estructurar la lógica de los agentes mediante grafos dirigidos de ejecución. Garantiza que el flujo documental siga caminos rígidos y deterministas según las reglas de negocio, manteniendo el control estricto del estado de la validación (*State*) y mitigando el riesgo de bucles infinitos.

1.2.2 SERVICIOS EN LA NUBE E INTELIGENCIA COMPUTACIONAL (AWS)

- **Amazon Bedrock y Amazon Nova Pro:** Bedrock actúa como la API securizada y gestionada que aloja el modelo fundacional multimodal de última generación **Amazon Nova Pro**. Este modelo constituye el cerebro cognitivo del sistema, encargado de la inferencia lógica para las notas simples y del análisis semántico visual de planos (croquis) y fotografías periciales.
- **Amazon S3 (Simple Storage Service):** Utilizado como el sistema central de almacenamiento de objetos (*Object Storage*) —no como una base de datos tradicional—. Actúa como el repositorio seguro en la nube donde se custodian los archivos PDF de las notas simples y los paquetes gráficos de las tasaciones.

- **Amazon Textract:** Servicio especializado en el Procesamiento Inteligente de Documentos (IDP). Se encarga de la fase de digitalización inicial, extrayendo de forma limpia y estructurada el texto plano de las notas simples para eliminar el ruido visual antes de enviarlo a los agentes.
- **Boto3:** Usamos la herramienta oficial de Amazon (boto3) para saltarnos las normas y los fallos que nos daba la librería de moda (LangChain). Al hacerlo así, pudimos coger las imágenes y los planos pesados, convertirlos en texto en la memoria del ordenador y mandárselos directamente a la Inteligencia Artificial de Amazon sin que el sistema se atascara ni diera errores de formato

1.2.3 CAPA DE PRESENTACIÓN (PROTOTIPO WEB)

- **React 18 y Vite 5:** Tecnologías utilizadas para el desarrollo de la interfaz de usuario web (*frontend*) en formato de *Single Page Application* (SPA), permitiendo una experiencia de usuario fluida e interactiva.
- **Express (Node.js):** Empleado para la creación de una API local que actúa como *Backend For Frontend* (BFF), centralizando la comunicación segura con la infraestructura de AWS mediante variables de entorno y garantizando la tolerancia a fallos mediante mecanismos de degradación progresiva (*fallback*).

Capítulo 2. DEFINICIÓN DEL TRABAJO

2.1 OBJETIVOS

El objetivo general de este trabajo consiste en conceptualizar, diseñar e implementar un **validador automático inteligente** denominado **AIValidAI**, destinado a auditar y optimizar el flujo de control de calidad en el proceso de tasación realizado por la empresa VALUM. El propósito principal no es sustituir el criterio del tasador, sino introducir una capa automatizada de supervisión que maximice la fiabilidad de los expedientes.

Para evaluar el éxito de la solución, se establecen los siguientes objetivos específicos centrándose en la precisión y la calidad:

Garantizar la Integridad Técnica mediante IA Multimodal: Desarrollar un sistema de agentes capaz de inspeccionar con alta precisión la disponibilidad de los soportes documentales y gráficos (como el script especializado `agent_croquis.py`), discriminando de forma infalible contenidos válidos de aquellos archivos corruptos, ilegibles o que se encuentren en blanco.

Elevar el Estándar de Calidad Operativa: Estandarizar los criterios de aceptación documental para que el 100% de los expedientes cumplan los requisitos mínimos de calidad de la organización, con independencia del técnico que los tramite o del volumen de trabajo del momento.

Optimizar la Precisión en la Ingesta de Datos: Implementar pipelines especializados por tipo de documento —nota simple, imágenes periciales y croquis arquitectónico— que extraigan y verifiquen la información jurídica de forma determinista, reduciendo la tasa de error documental a prácticamente cero y garantizando la consistencia de los datos del expediente.

Reducir Tiempos de Ciclo en Tareas de Auditoría: Automatizar las tareas mecánicas de validación y presencia documental para que el equipo de validadores internos pueda absorber un mayor volumen de expedientes, focalizando su esfuerzo exclusivamente en la resolución de discrepancias complejas o anomalías detectadas por el sistema.

2.2 METODOLOGÍA

Para el desarrollo del proyecto se ha seguido una metodología ágil adaptada a proyectos de Inteligencia Artificial. En primer lugar, se realizó una fase de ingeniería de requisitos junto con los validadores y control de calidad para identificar las reglas de negocio críticas. Posteriormente, se aplicó un diseño iterativo por componentes (Scrum), desarrollando y testeando de forma aislada cada agente cognitivo utilizando conjuntos de datos históricos. Finalmente, se procedió a la integración de los flujos mediante LangGraph y a la evaluación del rendimiento temporal y de precisión con el lote de pruebas final.

2.3 PLANIFICACIÓN Y ESTIMACIÓN ECONÓMICA

El desarrollo del proyecto se ha computado en un plazo de 6 meses con una dedicación estimada de 520 horas de trabajo. Considerando una retribución mensual de 500 €, el coste total de mano de obra asciende a 3.000 €. En cuanto a costes de infraestructura cloud (AWS S3, Textract y llamadas de producción a Bedrock Nova Pro), el gasto total durante

la fase de desarrollo y pruebas no superó los 45 €. El coste completo del proyecto se sitúa por tanto en **3.045 €**, lo que demuestra la alta viabilidad económica de la solución desarrollada.

Capítulo 3. ESTADO DEL ARTE

La automatización del procesamiento documental en entornos de alta exigencia regulatoria, como la tasación inmobiliaria, requiere la convergencia de varias disciplinas tecnológicas que han madurado de forma paralela en los últimos años. Para fundamentar las decisiones de diseño de AIValidAI, se ha realizado una revisión del estado actual de las tecnologías que conforman su núcleo: los modelos de lenguaje aplicados a la comprensión de documentos, la inteligencia documental en el sector financiero e inmobiliario, los paradigmas de orquestación de sistemas multi-agente y las herramientas de extracción de texto mediante OCR e IDP.

Cada uno de estos bloques responde a una pregunta concreta que el proyecto tuvo que resolver: ¿qué tecnología es capaz de entender el contenido semántico de una nota simple registral?, ¿qué precedentes existen en la industria para validar expedientes de forma automatizada?, ¿cómo se orquesta un conjunto de agentes de IA de forma determinista y auditable?, y ¿qué herramienta extrae el texto de documentos escaneados con suficiente fiabilidad para alimentar esos agentes? La revisión que sigue aborda estas cuatro preguntas en ese mismo orden.

3.1 MODELOS DE LENGUAJE APLICADOS AL PROCESAMIENTO INTELIGENTE DE DOCUMENTOS (IDP)

El Reconocimiento Óptico de Caracteres (OCR, por sus siglas en inglés) es una tecnología diseñada para convertir texto contenido en imágenes, planos o documentos escaneados en texto digital editable, indexable y seleccionable. En lugar de tratar el archivo como una fotografía estática y plana, el software procesa la matriz de píxeles, aísla e identifica las formas geométricas que corresponden a caracteres alfanuméricos y los traduce a un formato que la computadora puede "leer", copiar o buscar.

A pesar de haber sido el pilar de la digitalización durante décadas, los sistemas de OCR tradicionales —basados en plantillas rígidas o identificación de coordenadas espaciales— resultan **ineficientes e insuficientes** en entornos complejos como la tasación inmobiliaria por tres motivos fundamentales:

- **Rigidez estructural:** Exigen que los documentos sean idénticos píxel a píxel. Si una nota simple altera el orden de sus párrafos o añade un sello intermedio, el sistema pierde las coordenadas y falla en la extracción.
- **Ausencia de comprensión semántica:** El OCR clásico solo transcribe caracteres, pero no entiende su significado. Es incapaz de deducir si una cifra corresponde a la superficie útil interior, a la construida o a la de un elemento anejo.
- **Vulnerabilidad al ruido gráfico:** Ante planos densos con líneas de cota, firmas superpuestas o escaneos de baja resolución, el sistema confunde los trazos y genera texto corrupto.

Esta ineficiencia técnica ha forzado la transición hacia el **Procesamiento Inteligente de Documentos (IDP)**. Gracias a las arquitecturas de *Transformers* y a modelos multimodales nativos como **Amazon Nova Pro**, los sistemas actuales superan la mera transcripción lineal. Al proyectar los datos gráficos y textuales en espacios vectoriales complejos, el software es capaz de comprender el contexto, correlacionar el texto con la distribución espacial de un croquis e interpretar la integridad de un documento a nivel semántico y no meramente posicional.

3.2 DOCUMENT INTELLIGENCE EN EL SECTOR FINANCIERO E INMOBILIARIO

El concepto de *Document Intelligence* (o Inteligencia Documental) se ha consolidado como un vector de transformación crítico dentro del sector financiero y, de manera muy acusada, en el ámbito de las tasaciones inmobiliarias. Tradicionalmente se requieren equipos humanos procesaran manualmente ingentes volúmenes de documentación heterogénea (contratos, escrituras públicas, notas simples, planos arquitectónicos y dosieres periciales).

La integración de Modelos de Lenguaje de Gran Tamaño (LLMs) en los flujos de trabajo de este sector ha permitido evolucionar desde la simple digitalización hacia la **automatización cognitiva**. Las principales aplicaciones y ventajas que definen el estado del arte de esta tecnología en el entorno financiero e inmobiliario se resumen en tres ejes operacionales:

- **Auditoría de Cumplimiento y Mitigación de Riesgos:** Las entidades reguladas están obligadas a cumplir con normativas estrictas de control de calidad y veracidad documental (como la Orden ECO/805/2003 en España). Los sistemas avanzados de *Document Intelligence* actúan como una primera capa de defensa automática, validando de forma objetiva e infalible si los expedientes aportados están completos, son legibles y carecen de alteraciones o errores técnicos antes de iniciar cualquier valoración financiera.
- **Estandarización y Escalabilidad:** Al delegar la pre-auditoría en este validador inteligente, empresas de tasación como VALUM logran reducir significativamente el personal necesario para tareas rutinarias y repetitivas. Esto permite acelerar drásticamente los tiempos de procesamiento de cada expediente y absorber un mayor volumen de negocio de forma mucho más rápida y eficiente, manteniendo un criterio de aceptación homogéneo y libre de subjetividad humana.

3.3 EVOLUCIÓN DE FRAMEWORKS Y PARADIGMAS DE SISTEMAS MULTI-AGENTE

En la ingeniería de software basada en IA, el paradigma ha evolucionado desde las llamadas simples a APIs de modelos (*stateless prompting*) hacia ecosistemas de agentes autónomos y colaborativos. Al analizar los frameworks de orquestación para este proyecto, se evaluaron tres enfoques principales:

- **Frameworks de Autonomía Libre (CrewAI, AutoGen):** Estos entornos modelan a los agentes bajo un enfoque conversacional o de "junta directiva", donde los agentes debaten libremente entre sí mediante el patrón *ReAct* hasta alcanzar una solución de forma autónoma. Aunque son herramientas potentes para tareas creativas o generación de código fue descartado ya que carece de determinismo, dificulta la auditoría del software y presenta el riesgo de caer en bucles infinitos de conversación, lo que dispararía los costes de computación por consumo de tokens
- **Frameworks de Orquestación Orientada a Grafos (LangGraph):** Este framework hace que en lugar de dejar que la IA decida aleatoriamente qué hacer a continuación, LangGraph obliga al sistema a seguir un camino estricto diseñado por el programador. Para lograrlo, utiliza tres elementos muy fáciles de entender:
  1. **Nodos:** Son los "puestos de trabajo" o especialistas (en nuestro caso, el agente que lee las cargas, el que lee las superficies, etc.). Cada uno hace su tarea y pasa el expediente al siguiente.
  2. **Estado (*State*):** Es una carpeta compartida virtual. El expediente digital va pasando de nodo en nodo, y cada especialista escribe su parte en esa misma carpeta para que todos manejen la misma información.
  3. **Bordes (*Edges*):** Son las flechas o carreteras que conectan los puestos de trabajo. Pueden ser fijas (ir del paso A al paso B obligatoriamente) o condicionales (un agente "enrutador" mira el expediente y decide si mandarlo por la carretera de *Residencial* o la de *Terrenos*).

Esta estructura es ideal para el proyecto, ya que elimina la incertidumbre de la IA y garantiza que el proceso de auditoría sea siempre ordenado, predecible y fácil de supervisar.

3.4 OCR E IDP (INTELLIGENT DOCUMENT PROCESSING): AWS TEXTRACT VS ALTERNATIVAS

Para que el validador automático pueda analizar la información de los expedientes, primero necesita una herramienta que extraiga el texto de los documentos (como las notas simples en formato PDF) y lo convierta en un formato limpio y procesable por los agentes. Al evaluar la tecnología para esta fase de extracción, se analizaron dos aproximaciones principales:

- **OCR (Código abierto):** Es la herramienta gratuita tradicional de reconocimiento óptico de caracteres. Aunque funciona bien para textos sencillos y documentos impresos en alta resolución, se descartó para el proyecto debido a sus limitaciones operativas: requiere un laborioso trabajo previo de programación para definir plantillas, tiene serias dificultades para procesar tablas y su precisión cae drásticamente ante documentos escaneados, con marcas de agua o sellos del Registro.
- **Amazon Textract (Seleccionada):** Es el servicio especializado de Procesamiento Inteligente de Documentos (IDP) de AWS. Se eligió para este proyecto por tres ventajas clave que solucionan los problemas del OCR clásico:
    1. **Precisión y Extracción Semántica:** No se limita a leer caracteres sueltos; utiliza aprendizaje automático para identificar de forma inteligente bloques de texto, tablas y formularios complejos, manteniendo una alta precisión incluso si el documento original es un escaneo de baja calidad.
    2. **Integración con la Infraestructura:** Al formar parte del ecosistema de AWS, se conecta de manera directa y segura con los buckets de **Amazon S3**

donde se almacenan los expedientes de tasación, evitando la necesidad de descargar archivos localmente y reduciendo la latencia de procesamiento.

3. **Seguridad Corporativa:** Al procesarse dentro de la propia nube de la organización, garantiza el cumplimiento estricto de las normativas de privacidad y el cifrado de los datos confidenciales de las tasaciones.

Capítulo 4. MARCO TEORICO

4.1 ARQUITECTURAS DE AGENTES: REACT, REFLEXION Y PLAN-AND-EXECUTE

En la ingeniería de IA moderna, un agente no es solo un modelo que responde preguntas, sino una entidad capaz de razonar y actuar sobre su entorno. Para este proyecto se evaluaron tres arquitecturas principales:

- **ReAct (Reasoning and Acting):** Es el paradigma utilizado en `agent_croquis.py`. El agente genera un pensamiento (*Thought*), ejecuta una acción (*Action*) —como listar archivos en S3— y observa el resultado (*Observation*). Este ciclo se repite hasta que el agente considera que tiene información suficiente para emitir el juicio de validez del plano.
- **Reflexion:** Un patrón donde el agente genera una respuesta y luego un "agente crítico" la evalúa para corregir errores. Se descartó por ser computacionalmente más costoso para una tarea de validación binaria.
- **Plan-and-Execute:** El sistema crea primero un plan con todos los pasos y luego los ejecuta. Se prefirió ReAct por su mayor flexibilidad ante imprevistos en la estructura de las carpetas de S3.

4.2 ESTRATEGIAS DE INFERENCIA: PROMPTING ESTRUCTURADO VS RAG Y FINE-TUNING

Una decisión de diseño crítica para el proyecto fue cómo especializar al modelo en la normativa de validación. Se analizaron tres vías:

- **Fine-tuning (Ajuste fino):** Entrenar el modelo con miles de ejemplos de planos y notas simples. Se descartó por el alto coste, la necesidad de un dataset masivo etiquetado y la rapidez con la que cambian los formatos registrales.
- **RAG (Retrieval-Augmented Generation):** Consultar una base de datos externa antes de responder. Útil para buscar leyes, pero innecesario para validar la integridad de un archivo específico que ya se tiene en memoria.
- **Prompting Estructurado (Seleccionada):** Se optó por diseñar prompts técnicos muy estrictos con **temperatura baja (0.1)**. Esta estrategia garantiza que el modelo **Amazon Nova Pro** se comporte de forma determinista, ciñéndose exclusivamente a las reglas de negocio inyectadas y devolviendo siempre un formato JSON parseable, lo que asegura la **precisión y calidad** del validador automático.

4.3 ORQUESTACIÓN DE WORKFLOWS Y ECOSISTEMA DE DESARROLLO (LANGCHAIN Y LANGGRAPH)

Para construir un validador automático predecible, es necesario combinar el manejo de los modelos de IA con el control estricto de sus procesos. El desarrollo se apoya en el ecosistema

de **LangChain** y **LangGraph**: una librería aporta las herramientas individuales y la otra diseña las reglas del camino.

4.3.1 EL ROL DE LANGCHAIN: COMPONENTES ATÓMICOS Y ABSTRACCIÓN

LangChain actúa como la infraestructura base de bajo nivel. Su función principal no es decidir el orden de los procesos, sino proporcionar los bloques de construcción necesarios para que el código interactúe con la IA. En **AIValidAI**, se encarga de:

- **Abstracción del Modelo:** Mediante `ChatBedrock`, conecta el código con **Amazon Nova Pro** en AWS, permitiendo fijar parámetros como la temperatura baja (0.1).
- **Protocolo de Mensajes:** Estructura las entradas en bloques claros (`SystemMessage` para las instrucciones y `HumanMessage` para los datos del expediente).
- **Definición de Capacidades (`Tools`):** A través del decorador `@tool`, transforma funciones de Python (como descargar de S3 o leer un JSON) en herramientas que la IA puede comprender y ejecutar

```
from langchain_aws import ChatBedrock

from langchain_core.messages import SystemMessage, HumanMessage

from langchain_core.tools import tool


# 1. Configuración de la abstracción del modelo con temperatura estricta

llm = ChatBedrock(

    model_id="eu.amazon.nova-pro-v1:0",
```

```
    beta_use_anthropic_element=True,
    model_kwargs={"temperature": 0.1, "max_tokens": 2000}
)

# 2. Definición de una herramienta (Tool) para el agente
@tool
def leer_metadatos_tasacion(bucket: str, key: str) -> str:
    """Accede al repositorio S3 de VALUM y extrae los metadatos de control del expediente."""
    # Lógica de conexión con boto3...
    return "Metadatos extraídos correctamente"

# 3. Estructuración del protocolo de mensajes y binding de herramientas
tools = [leer_metadatos_tasacion]
llm_with_tools = llm.bind_tools(tools)

mensajes = [
    SystemMessage(content="Eres un agente auditor de VALUM. Usa las herramientas para verificar los datos."),
    HumanMessage(content="Analiza el expediente con ID: VL-2026-69975 en el bucket principal.")
]

# Invocación estructurada
```

```
respuesta = llm_with_tools.invoke(mensajes)
```

4.3.2 EL ROL DE LANGGRAPH: ESTRUCTURA DE CONTROL Y GRAFOS DIRIGIDOS

Mientras LangChain pone las herramientas, **LangGraph** actúa como "el arquitecto que diseña la línea de montaje". Su función es obligar a la IA a seguir un camino estricto y predecible para cumplir las reglas de negocio de la empresa de tasación **VALUM**, utilizando tres conceptos clave:

- **StateGraph (El Estado):** Es una carpeta compartida virtual que almacena la memoria del flujo. A medida que los agentes extraen datos (como superficies o cargas), la información se consolida en este objeto sin perder consistencia.
- **Nodos (Nodes):** Son los puestos de trabajo o especialistas de la línea de montaje. Cada nodo es una función de Python o un agente independiente que recibe el expediente, hace su tarea y guarda el resultado en el Estado.
- **Bordes (Edges):** Son las carreteras que conectan los nodos. Pueden ser **deterministas** (secuencia fija obligatoria para encadenar los agentes de la nota simple) o **condicionales** (un enrutador o `router_agent` decide qué camino tomar según la tipología del inmueble).

Capítulo 5. SISTEMA/MODELO

DESARROLLADO

El sistema AIValidAI se ha estructurado como una plataforma modular e inteligente capaz de procesar de manera independiente y asíncrona las tres fuentes de información críticas en un expediente de tasación. El núcleo del desarrollo se apoya en orquestadores avanzados de modelos de lenguaje (LLMs) y agentes cognitivos especializados que transforman datos no estructurados —imágenes, planos arquitectónicos y documentos escaneados— en esquemas JSON estructurados de alta fiabilidad, mediante un proceso de transcripción multimodal en el que cada entrada visual es convertida primero en representación textual procesable antes de ser razonada por los agentes.

5.1 ANÁLISIS DEL SISTEMA

El ecosistema de **AIValidAI** está diseñado bajo un paradigma híbrido de flujos de trabajo deterministas y agentes autónomos. El sistema recibe como entrada el expediente de tasación almacenado en buckets de **Amazon S3**. A partir de este punto, la arquitectura se bifurca en tres subsistemas de extracción y validación orientados a objetivos:

- **Subsistema Documental (Nota Simple):** Encargado del procesamiento secuencial y paralelo de texto jurídico mediante un grafo de agentes orquestado con Langraph.

- **Subsistema de Visión Pericial (Imágenes):** Diseñado para la auditoría visual del reportaje fotográfico pericial mediante procesamiento directo en memoria. El sistema verifica que el conjunto de imágenes aportadas es coherente con el tipo de visita realizada: una visita exterior debe acreditar fotografías del exterior del inmueble y una visita interior debe acreditar fotografías del interior. La ausencia total de imágenes del tipo correspondiente al tipo de visita declarado se reporta como discrepancia.
- **Subsistema Espacial (Croquis/Plano):** Un agente basado en el patrón de razonamiento *Reasoning and Acting* (ReAct) con acceso dinámico a herramientas del entorno Cloud.

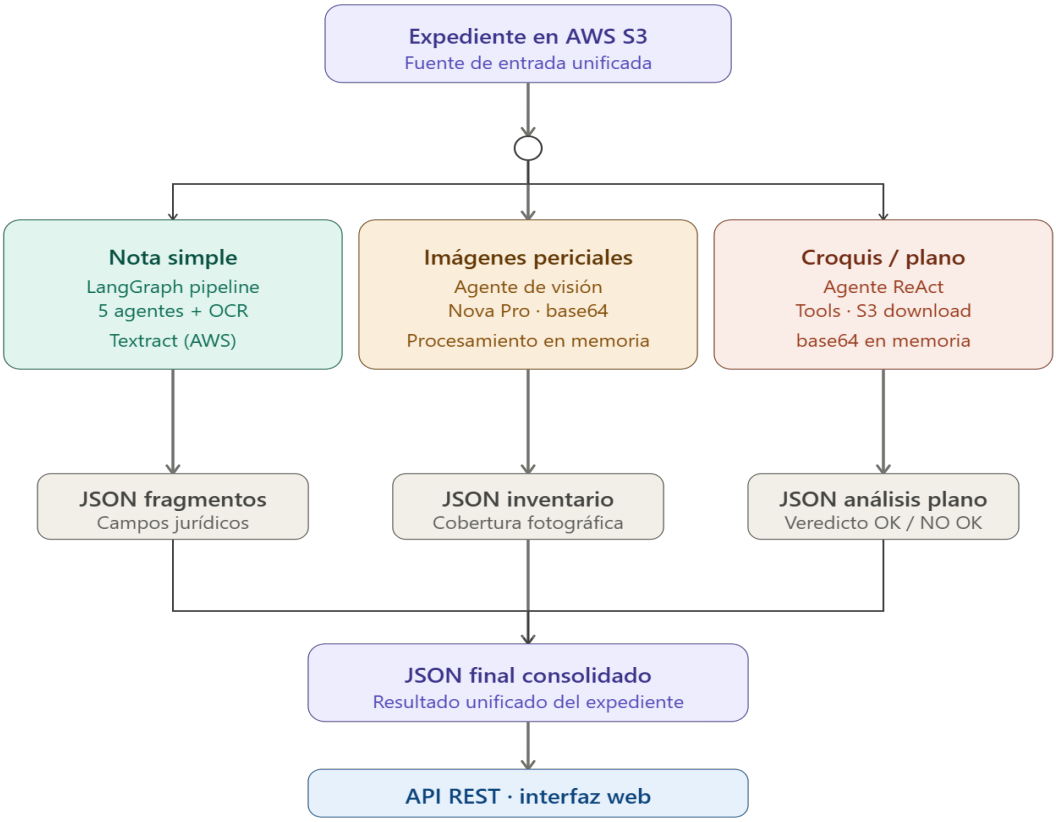


Ilustración 3- sistema de AIValidAI

5.2 DISEÑO

A continuación, se detalla la lógica interna y el diseño de flujo concebido para cada uno de los tres componentes principales:

5.2.1 PIPELINE DE LA NOTA SIMPLE (ORQUESTACION CON LANGRAPH)

Para el análisis de la nota simple se diseñó un pipeline compuesto por **5 agentes en cadena** que operan sobre una arquitectura de grafo dirigida:

- **Fase de Extracción Textual:** El documento PDF original puede estar alojado en un bucket de Amazon S3 o en el sistema de archivos local, y en ningún caso es leído directamente por los agentes. En su lugar, se implementa un componente de extracción técnica basado en AmazonTextractPDFLoader que actúa como capa previa al razonamiento. Este servicio realiza un reconocimiento óptico de caracteres (OCR) avanzado sobre cada página del documento, aislando el contenido textual, eliminando el ruido gráfico y reduciendo la carga de tokens que se transmite a los modelos.

  Esta fase es especialmente crítica en el contexto de las notas simples registrales, ya que estos documentos no son PDFs de texto nativo en su totalidad. Con frecuencia, determinados bloques —como los asientos de inscripción antiguos, las notas marginales o las certificaciones de cargas— se encuentran escaneados como imágenes incrustadas dentro del propio PDF. Sin la capa de OCR, el texto contenido en esas imágenes sería invisible para los agentes, provocando extracciones incompletas en precisamente los campos de mayor relevancia jurídica. Amazon

Textract resuelve este problema al procesar indistintamente tanto el texto nativo del PDF como el texto embebido en imagen, garantizando que la salida entregada a los agentes sea siempre un bloque de texto plano completo y homogéneo.

- **Agente Enrutador (router_agent):** El texto plano resultante se inyecta en el primer agente del grafo. Su función exclusiva es la clasificación tipológica del activo inmobiliario (ej. determinar si la tipología es *RESIDENCIAL*, *COMERCIAL*, *INDUSTRIAL*, o *TERRENO*).
- **Capa de Agentes Especialistas en Paralelo:** Una vez definida la tipología, el flujo se ramifica. El texto original completo se distribuye simultáneamente a cinco agentes cognitivos especializados. Cada uno de ellos opera de manera aislada enfocándose en una regla de negocio específica:
    - **`registry`:** Extrae datos relativos a la inscripción, tomo, libro y fincas registrales.
    - **`surfaces`:** Identifica las superficies útiles y construidas descritas en el cuerpo jurídico.
    - **`ownership`:** Identifica la titularidad, nombres de propietarios y cuotas de participación.
    - **`charges`:** Identifica cargas activas, hipotecas, servidumbres o afecciones fiscales.
    - **`anejos`:** Extrae elementos vinculados (plazas de garaje, trasteros, etc.).
- **Fase de Fusión (`merge`):** Cada uno de los cinco agentes procesa el texto en paralelo y genera como salida un fragmento de datos en formato JSON especializado. El nodo final del grafo (`merge`) captura estos sub-esquemas, resuelve posibles conflictos de redundancia y consolida un **JSON unificado** que representa la radiografía jurídica limpia del inmueble.

5.2.2 AGENTE DE VISIÓN PARA IMÁGENES PERICIALES (AGENT_IMAGEN.PY)

La validación visual prescinde por completo de técnicas tradicionales de OCR o capas intermedias de extracción de texto. El diseño conceptual se basa en un **agente único de visión nativa**:

- El sistema recupera las imágenes periciales, ya sea mediante descarga remota desde buckets de **Amazon S3** o mediante la ingesta directa desde el **sistema de archivos local** como parámetro de entrada. Posteriormente, dichas imágenes son procesadas y codificadas en cadenas de texto en formato **Base64** directamente en la memoria volátil
- Este flujo binario es inyectado junto con un *prompt* técnico de directrices de validación en el modelo multimodal **Nova Pro** a través de **Amazon Bedrock**.
- Al recibir el flujo de imagen nativo, el modelo procesa la información semántica visual (texturas, objetos, iluminación...) e identifica el contenido de cada fotografía clasificándolo como imagen de interior o de exterior. El veredicto final se emite confrontando esa clasificación con el tipo de visita declarado en la solicitud: si se declaró visita exterior y no existe ninguna fotografía de exterior, o si se declaró visita interior y no existe ninguna fotografía de interior, el sistema registra la incidencia como discrepancia en el JSON de salida.

5.2.3 ANÁLISIS DE CROQUIS Y PLANOS (AGENT_CROQUIS.PY)

*El subsistema instrumentado en `agent_croquis.py` actúa como una **capa de auditoría de integridad técnica y disponibilidad documental**. El objetivo de este agente no*

es realizar una tasación u homologación arquitectónica del diseño del plano, sino certificar la ***validez operativa del input gráfico****.*

Al delegar la lectura en ***Amazon Nova Pro****, el agente realiza una inspección semántica binaria basada en la legibilidad:*

1. ***Filtro de Inutilidad Técnica (Estado*** `BLANCO` / `SIN_CROQUIS`**):** *El modelo discrimina si el archivo recuperado de AWS S3 está corrupto, es ilegible, carece de trazos identificables o se trata de una página totalmente vacía que imposibilitaría cualquier procesamiento posterior por parte de los tasadores.*
2. ***Certificación de Documento Procesable (Estado*** `OK`**):** *Si el documento cuenta con información gráfica estructurada y legible que descarte los estados de error anteriores, el agente valida técnicamente el archivo, dándolo por correcto y consolidando el flujo de datos.*

Este enfoque de "agente guardián" garantiza de forma automatizada que el expediente cuenta con el soporte documental mínimo necesario, mitigando errores en fases posteriores del flujo de la tasación."

5.3 IMPLEMENTACIÓN TECNOLÓGICA

5.3.1 LIBRERÍAS DE INTEGRACIÓN DE IA Y ORQUESTACIÓN DE AGENTES

La gobernanza del razonamiento lógico y el ciclo de vida de los agentes se delega en el ecosistema compuesto por **LangChain** y su extensión orientada a grafos, **LangGraph**. Esta capa de software se estructura bajo dos paradigmas de control diferenciados:

- **Paradigma ReAct (Autonomía Guiada):** Utilizado principalmente en la auditoría visual de planos (agent_croquis.py). Mediante el uso del decorador `@tool`, se exponen funciones nativas del sistema para que el modelo ejecute un bucle dinámico de pensamiento, acción y observación hasta resolver la integridad del archivo.

- **Paradigma de Grafo Dirigido (Flujo Secuencial):** Utilizado en el pipeline de la nota simple. LangGraph coordina un flujo estructurado que se inicia en un nodo de clasificación analítica (router_agent) y deriva el expediente de forma síncrona a través de una cadena rígida de cinco agentes especialistas. Este diseño garantiza el determinismo del sistema, la mutación controlada del estado (State) y elimina el riesgo de bucles infinitos de ejecución.

5.3.3 CONSUMO DE MODELOS FUNDACIONALES E INTEGRACIÓN CON AWS

La inteligencia computacional de la plataforma se centraliza en el modelo fundacional multimodal **Amazon Nova Pro**, desplegado de forma gestionada a través de **Amazon Bedrock** en la región eu-west-1. Con el fin de optimizar las peticiones y cumplir con las restricciones del framework, se diseñaron dos metodologías de comunicación complementarias:

- **Abstracción de Alto Nivel (ChatBedrock):** Integrada mediante el SDK langchain-aws. Se emplea exclusivamente en flujos de razonamiento conversacional, toma de decisiones basada en contexto y ejecución de herramientas parametrizadas.

- **Invocación Nativa por Cliente (boto3):** Implementada para paliar la falta de soporte nativo de LangChain en la gestión de contenidos multimodales binarios para este modelo. El software realiza llamadas directas de bajo nivel al endpoint bedrock-runtime, transmitiendo los mapas de bits (planos y reportajes fotográficos) codificados en texto Base64 directamente desde la memoria volátil. Esta aproximación garantiza la compatibilidad absoluta con la API de Amazon bajo estrictas políticas corporativas de cifrado

Capítulo 6. IMPLEMENTACION

Para trasladar el diseño conceptual y lógico del validador automático **AIValidAI** a un entorno operativo, se ha desarrollado una solución de software modular e independiente basada en el lenguaje de programación Python 3. El sistema se acopla dinámicamente a la infraestructura en la nube y consume capacidades avanzadas de modelos fundacionales de inteligencia artificial a través de APIs securizadas.

6.1 ESTRUCTURA DEL CÓDIGO FUENTE Y MODULARIDAD

Para trasladar el diseño lógico del validador automático **AIValidAI** a un entorno de ejecución e integración continua , la arquitectura de software se ha estructurado bajo un paradigma estrictamente modular, eludiendo soluciones monolíticas. El desarrollo se segmenta en módulos y *scripts* independientes en Python 3 que garantizan el aislamiento de fallos, la mantenibilidad del código, una alta cohesión interna y un bajo acoplamiento entre los componentes encargados de auditar cada tipo de soporte del expediente de tasación.

A nivel de organización de ficheros y responsabilidades de diseño, el núcleo del repositorio de código fuente se compone de la siguiente estructura modular:

`router_agents.py`: Este módulo aloja la lógica del componente enrutador del sistema (`router_agent`). Su responsabilidad es actuar como el punto de entrada cognitivo del flujo documental , analizando el texto crudo extraído mediante funciones de inferencia semántica para determinar la tipología específica del activo inmobiliario (ej. `RESIDENCIAL`, `COMERCIAL`, `INDUSTRIAL`, `TERRENO`, `FINCA RUSTICA`, `TERCIARIO` o `DOTACIONAL`).

Ecosistema de Fábricas Documentales (`simple_notes_agent_*.py`): En lugar de emplear una única capa genérica de extracción, la implementación aplica un **patrón de**

diseño de Fábrica Especializada (*Factory Pattern*) estructurado en múltiples archivos dedicados por taxonomía inmobiliaria:

- `simple_notes_agent_terreno.py`
- `simple_notes_agent_comercial.py`
- `simple_notes_agent_industrial.py`
- `simple_notes_agent_terciario.py`
- `simple_notes_agent_dotacional.py`
- `simple_notes_agent_fincarustica.py`
- `simple_notes_agent_anexo.py` Cada uno de estos componentes contiene las funciones constructoras independientes (ej. `create_simple_notes_registry_agent`, `create_simple_notes_surfaces_agent`, etc.) que acoplan de forma estricta los prompts específicos y las configuraciones de los modelos a las reglas de negocio particulares de esa tipología de finca en la empresa de tasación **VALUM**.

`agent_croquis.py`: *Script* responsable del ciclo autónomo de auditoría espacial y planos arquitectónicos. Gobierna un agente guiado por herramientas (*tools*) que interactúa con el almacenamiento en la nube para descargar y verificar el estado técnico de los soportes gráficos.

- **`agent_imagen.py`**: Componente dedicado en exclusiva al procesamiento multimodal de las fotografías periciales tomadas en la inspección ocular, abstrayendo la lógica de catalogación e inventario de estancias del inmueble.
- **`writer.py`**: Módulo instrumental que implementa funciones de asistencia documental (`create_writer_agent`), especializado en la consolidación y formateo final de los reportes técnicos resultantes a partir de las directivas del prompt global `WRITER_PROMPT`.

Esta segregación de responsabilidades a nivel de código fuente asegura que cualquier modificación en los criterios normativos o de aceptación de una tipología concreta (por ejemplo, alterar las reglas de extracción para activos industriales) requiera únicamente la edición de su módulo aislado correspondiente, garantizando la estabilidad y la resiliencia del resto del validador automático durante la explotación del software.

6.2 PIPELINE DE NOTAS SIMPLES

El procesamiento y auditoría de la nota simple registral representa el flujo documental de mayor complejidad relacional dentro del sistema **AIValidAI**. Debido a la variabilidad de las descripciones legales según la naturaleza jurídica del inmueble, el software elude el uso de prompts estáticos generalistas. En su lugar, implementa un middleware de enrutamiento dinámico seguido de un patrón de **Fábrica Parametrizada por Tipología**, coordinado bajo un grafo dirigido de ejecución secuencial en LangGraph.

6.2.1 Fase de Clasificación y Enrutamiento Dinámico

El flujo de control se inicia interceptando el texto plano unificado proveniente del cargador inteligente. Este contenido se inyecta de forma inmediata en el componente crítico desarrollado en el archivo `router_agents.py`: la función estructurada `get_property_type`.

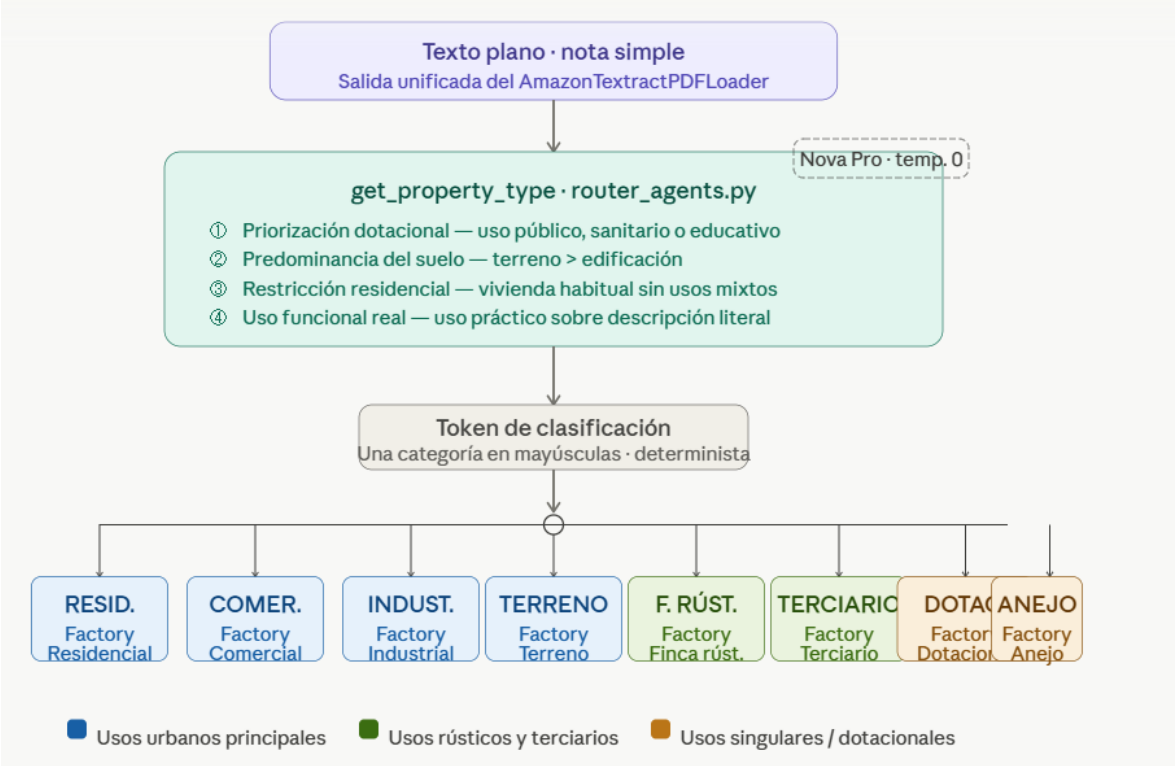


Ilustración 4- sistema de agentes para la nota simple

Esta función actúa como el director de orquesta del pipeline documental. Su objetivo no es realizar una búsqueda simple de palabras clave, sino evaluar el contexto literal y el uso real descrito en el documento para clasificar el inmueble de forma determinista en una de las siguientes categorías exclusivas: `RESIDENCIAL`, `COMERCIAL`, `INDUSTRIAL`, `TERRENO`, `FINCA RUSTICA`, `TERCIARIO`, `DOTACIONAL` o `ANEXO`.

Para garantizar que la IA clasifique con la misma precisión que un validador experto de la empresa de tasación **VALUM**, la función parametriza el modelo **Amazon Nova Pro** utilizando un prompt de enrutamiento estricto (`ROUTER_PROMPT`) fundado en cuatro reglas de negocio jerárquicas:

1. **Priorización Dotacional:** El sistema asigna inmediatamente la etiqueta `DOTACIONAL` si identifica indicios de uso público, sanitario o educativo (por ejemplo, parcelas destinadas a fines sanitarios o colegios).
2. **Criterio de Predominancia del Suelo:** Se prioriza la clasificación como `TERRENO` o `FINCA RUSTICA` si la descripción literal indica que el suelo o la parcela tiene mayor relevancia que la edificación principal, o si se detectan elementos de explotación agropecuaria explícitos (como hórreos, eras, corrales o cultivos).
3. **Restricción Residencial:** La categoría `RESIDENCIAL` queda limitada única y exclusivamente a fincas cuya descripción indique que su uso principal es el de vivienda habitual o piso, exigiendo la ausencia de terrenos significativos o usos públicos.
4. **Uso Funcional Real:** En caso de que la descripción literal jurídica y el uso práctico del inmueble difieran, el modelo está instruido para aplicar la categoría más representativa del uso real del activo.

Para asegurar un comportamiento puramente predictivo y eliminar cualquier margen de "creatividad" o alucinación en la IA, el modelo procesa esta instrucción con una **temperatura fija de 0**. El script captura la salida de la API, aísla el token unívoco resultante en mayúsculas y LangGraph utiliza este resultado para desviar automáticamente el expediente hacia la fábrica de agentes especialista correspondiente.

6.2.2 Arquitectura de la Fábrica de Agentes Especialistas.

Una vez que el enrutador (`router_agent`) decide la categoría del inmueble, el sistema activa un conjunto de agentes diseñados exclusivamente para esa tipología. En lugar de utilizar un único programa genérico para todo, el código emplea el **patrón de diseño de Fábrica (*Factory Pattern*)** mediante archivos independientes (`simple_notes_agent_comercial.py`, `simple_notes_agent_terreno.py`, etc.).

Esto significa que si la nota es de un comercio, se fabrica un equipo de agentes expertos en comercios; si es de un terreno, se fabrica un equipo experto en terrenos. Cada una de estas "fábricas" de software genera siempre de forma automática cinco agentes especialistas, los cuales se reparten el trabajo para extraer la información requerida para la tasacion.

Independientemente del módulo de especialización que se invoque, cada fábrica expone una interfaz homogénea compuesta por cinco funciones constructoras atómicas. Estas funciones encapsulan la llamada al método `create_agent` de LangChain, inyectando de manera rígida los prompts del sistema parametrizados y el modelo fundacional optimizado para la región:

1. **`create_simple_notes_registry_agent`**: Función encargada de instanciar el nodo del grafo enfocado exclusivamente en la identificación registral básica. Consume prompts dedicados (ej. `SIMPLE_NOTE_REGISTRY_PROMPT_COMERCIAL`) para aislar de forma unívoca el número de finca registral, el tomo, el libro y el folio del documento, ignorando cualquier otra variable jurídica secundaria.
2. **`create_simple_notes_surfaces_agent`**: Inicializa el agente especialista en métricas espaciales y descriptivas. Su prompt del sistema le instruye para extraer de manera estricta el desglose de superficies útiles y construidas, la presencia de linderos geográficos, la dirección postal física del activo y las cuotas de participación asociadas.

3. **`create_simple_notes_ownership_agent`**: Genera el nodo experto encargado de descifrar la titularidad civil del inmueble. El modelo analiza el texto para aislar los nombres completos de los propietarios registrales, sus identificadores legales y el porcentaje o cuota de participación exacta sobre el dominio del derecho.
4. **`create_simple_notes_charges_agent`**: Modula el agente de criticidad jurídica del pipeline. Este componente está diseñado para rastrear e inventariar la existencia de cargas activas, anotaciones de embargo, hipotecas vigentes, servidumbres de paso o afecciones fiscales pendientes que graven la finca.
5. **`create_simple_notes_anejos_agent`**: Instancia el último nodo de extracción de la cadena, especializado en el descubrimiento de elementos secundarios e inseparables vinculados contractualmente a la propiedad principal, tales como cuartos trasteros, carboneras o plazas de aparcamiento.

Para optimizar la eficiencia y centralizar la configuración técnica, todas las fábricas implementan internamente la función privada `_get_oss_model()`. Esta rutina encapsula y devuelve una instancia del modelo multimodal de última generación **Amazon Nova Pro** (`ChatBedrock`), parametrizado de manera estricta apuntando al identificador unificado `"eu.amazon.nova-pro-v1:0"`, desplegado de forma segura en la región operativa de la nube de la organización (`eu-west-1`).

6.3 PIPELINE DE CROQUIS Y PLANOS (`AGENT_CROQUIS.PY`)

6.3.1 Arquitectura del Agente, Herramientas y Control de Estado

El componente `agent_croquis.py` funciona como un **agente con autonomía guiada**. A diferencia del pipeline de la nota simple (que sigue un camino de pasos fijos y rígidos para leer texto), este agente se encarga de analizar archivos gráficos (planos y croquis).

Para cumplir con su objetivo, utiliza el patrón de razonamiento **ReAct** (*Reasoning and Acting* / Pensamiento y Acción). Esto significa que el agente no sigue una ruta predefinida, sino que toma decisiones sobre la marcha en un bucle de tres pasos:

1. **Piensa:** Evalúa qué necesita hacer (por ejemplo: *"Necesito revisar si el plano del expediente está en la carpeta de la nube"*).
2. **Actúa:** Ejecuta una herramienta (*tool*) en Python para resolverlo (por ejemplo: abre la carpeta en Amazon S3 y descarga el archivo).
3. **Observa:** Mira el resultado obtenido (por ejemplo: *"Ya tengo el archivo, ahora voy a comprobar si está en blanco"*).

Este ciclo se repite de forma dinámica hasta que el agente logra emitir el veredicto final sobre la validez del plano.

6.4 PIPELINE DE IMÁGENES PERICIALES (IMAGEN_ANALYZER.PY)

El subsistema de imágenes periciales emplea un agente de visión nativa sin herramientas externas. Las fotografías se recuperan de S3, se codifican en base64 en memoria y se transmiten directamente al modelo Amazon Nova Pro junto con el tipo de visita declarado. El modelo clasifica cada imagen como interior o exterior y emite un veredicto de coherencia: se reporta discrepancia únicamente cuando el reportaje no contiene ninguna fotografía del tipo que corresponde a la visita realizada.

6.4.1 Configuración del Modelo de Visión

El análisis de las fotografías periciales requiere una aproximación puramente visual, ya que no estamos procesando texto escrito, sino imágenes reales de la propiedad. Para gestionar esto, el script implementa la función constructora `create_image_analyzer_model`.

Esta función es la encargada de configurar un **agente especialista en visión**. A diferencia del agente de planos o del pipeline de notas simples, este componente se define con una lista de herramientas completamente vacía (`tools=[]`).

El motivo de esta decisión de diseño es muy sencillo: el agente de imágenes no necesita interactuar de forma autónoma con el entorno (no tiene que listar carpetas ni buscar información externa). Su única responsabilidad es recibir las fotografías que el sistema le envía y, utilizando exclusivamente la capacidad de visión nativa y el razonamiento del modelo **Amazon Nova Pro**, interpretar el contenido visual de cada estancia.

6.4.2 Procesamiento en Memoria e Inventariado de Estancias

Una vez que la función `create_image_analyzer_model` inicializa el agente de visión multimodal con el modelo **Amazon Nova Pro**, este se integra dentro del pipeline de control de calidad del expediente bajo un procesamiento eficiente y directo en memoria:

1. **Transmisión de Inputs Visuales:** El sistema recupera las fotografías periciales almacenadas en los buckets de **Amazon S3** y, para optimizar el rendimiento y evitar escrituras físicas en el disco local, las convierte en cadenas de texto estructuradas en base64 en la memoria volátil.
2. **Procesamiento Multimodal de una Sola Capa:** Estas cadenas binarias se inyectan directamente en el flujo de entrada del agente de visión. Al no disponer de herramientas externas (`tools=[]`), el modelo ejecuta una inferencia en un solo paso (*single-step inference*), analizando directamente la matriz de píxeles mediante sus capacidades nativas de comprensión visual.
3. **Generación del Output Estructurado:** El agente evalúa las características semánticas de cada captura y clasifica las fotografías según su contenido en imágenes de interior o de exterior. A partir de esa clasificación y del tipo de visita declarado en la solicitud, el sistema genera un JSON estructurado con el inventario de fotografías detectadas por categoría y el veredicto de cumplimiento. Se reporta discrepancia únicamente cuando no existe ninguna fotografía del tipo que corresponde a la visita declarada: ausencia total de exteriores en una visita exterior, o ausencia total de interiores en una visita interior.

Este diseño permite al validador automático certificar de forma objetiva que el expediente cuenta con el **inventario mínimo de estancias obligatorio** exigido por la normativa interna de **VALUM**, detectando de forma automática anomalías visuales o la ausencia de soportes fotográficos críticos antes de dar el expediente por válido.

Capítulo 7. INTERFAZ WEB DE VALIDACIÓN (PROTOTIPO FUNCIONAL)

7.1 OBJETIVO Y CONTEXTO

Con el fin de demostrar la viabilidad operativa del sistema y facilitar la evaluación de los resultados de los pipelines de IA en un entorno realista, se desarrolló una interfaz web funcional que actúa como capa de presentación sobre la API descrita en el apartado anterior. Esta interfaz no constituye un producto final de producción, sino un prototipo funcional que permite a usuarios técnicos y stakeholders del proceso de tasación interactuar con el sistema de validación de forma completa.

El objetivo principal es ilustrar el ciclo de vida completo de una validación documental: desde la carga del documento hasta la visualización estructurada de los resultados emitidos

por los agentes de IA. Asimismo, sirve como banco de pruebas para validar los formatos de respuesta de la API y la experiencia de usuario antes de una integración definitiva en los sistemas de gestión de tasaciones.

7.2 ARQUITECTURA TÉCNICA

La interfaz web sigue una arquitectura cliente-servidor desacoplada. El frontend es una Single Page Application (SPA) construida con React y Vite, mientras que la comunicación con los modelos de IA se realiza exclusivamente a través de la API Express desarrollada en el capítulo anterior, que actúa como Backend For Frontend (BFF).

Tabla 4 Desglose tecnológico y despliegue de la arquitectura de la interfaz web

Capa	Tecnología	Puerto	Notas
Frontend	React 18 + Vite 5	5173	SPA
BFF / API	Express (Node.js)	4000	Backend For Frontend
API real AIValidAI	HTTPS + credenciales AWS	externo	Amazon Bedrock
Enrutado	React Router DOM v6	—	Rutas del cliente

El frontend se comunica únicamente con el BFF local. El BFF reenvía las peticiones a la API real de AIValidAI autenticándose mediante credenciales AWS (Access Key ID, Secret Access Key y Session Token opcional) configuradas como variables de entorno. Si la API

real no está disponible, el BFF aplica una degradación progresiva: en el caso de imágenes intenta resolver el resultado a partir del propio ZIP subido; si tampoco es posible, retorna datos mock predefinidos. Este diseño garantiza que el prototipo puede demostrar su funcionalidad completa con independencia de la disponibilidad del servicio externo.

7.3 ESTRUCTURA DE RUTAS Y FLUJO DE USUARIO

La aplicación define cinco rutas que reflejan los estados del ciclo de vida de una validación:

Tabla 5- Definición de rutas y flujo de navegación de la interfaz web

Ruta	Pantalla	Función
/bandeja	Panel de control	Listado de validaciones y estadísticas globales
/nueva-validacion	Nueva validación	Formulario de inicio y carga de documento
/validacion/:id/procesando	Procesando	Espera con progreso en tiempo real

/validacion/:id/resultado/:mode	Resultado	Visualización de resultados (éxito o discrepancias)
/validacion/:id/error	Error técnico	Pantalla de fallo del pipeline

El flujo nominal es lineal: el usuario accede al panel, inicia una validación cargando un documento, espera el procesamiento y obtiene el resultado. El flujo diverge en la pantalla de resultado según el escenario devuelto por la API (success o discrepancies) y puede terminar en error si el pipeline no puede procesar el documento.

7.4 DESCRIPCIÓN DE PANTALLAS

A continuación vamos a hablar de los diferentes resultados visuales que nos podemos encontrar en la interfaz web de este proyecto.

7.4.1 PANEL DE CONTROL (BANDEJA)

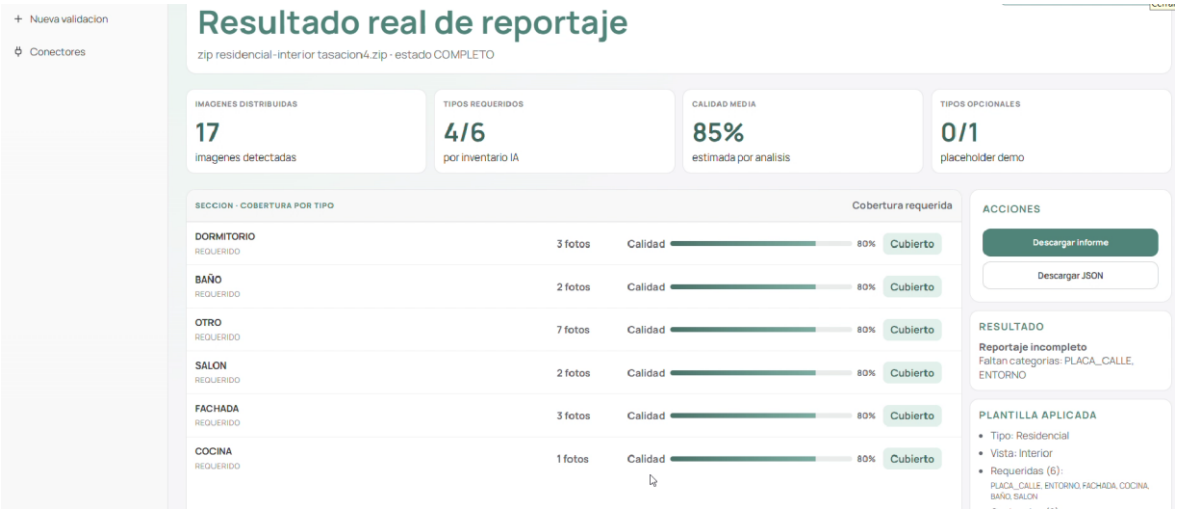


Ilustración 5- Bandeja de entrada

La pantalla principal muestra cuatro tarjetas de métricas operativas: validaciones en curso, expedientes pendientes de revisión manual (con discrepancias), validaciones cerradas en los últimos siete días y tiempo medio de procesamiento (percentil 50 para notas simples). Bajo las métricas, una tabla lista todas las validaciones con su referencia interna, nombre del documento, tipo, estado visual mediante chips de color (Validado / Con avisos / Procesando / Error), número de hallazgos y marca temporal. La tabla incluye un campo de búsqueda que filtra en tiempo real por referencia, nombre de archivo o tipo de documento.

7.4.2 NUEVA VALIDACIÓN

El formulario se adapta dinámicamente al tipo de documento seleccionado. Para notas simples, admite subida de PDF o URL de S3. Para imágenes, admite subida de ZIP, imagen individual, o URL de S3; requiere además la tipología del inmueble (Residencial, Terreno o Industrial) y el tipo de visita (Interior o Exterior), parámetros necesarios para aplicar la

plantilla de cobertura fotográfica correcta según las reglas de auditoría definidas en el sistema. Para croquis, admite PDF o imagen del plano, o URL de S3.

El botón de validación permanece deshabilitado hasta que se cumplen los requisitos mínimos de cada tipo. Al confirmar, se realiza un POST /api/validaciones con multipart/form-data y el usuario es redirigido automáticamente a la pantalla de procesamiento.

7.4.3 PROCESANDO

Esta pantalla implementa un mecanismo de polling activo: cada 1.200 milisegundos realiza una petición GET /api/validaciones/:id y comprueba el campo runtime.status. Mientras el estado es processing se muestra una barra de progreso y el porcentaje completado. Cuando el estado cambia a completed, el componente navega automáticamente a la pantalla de resultado; si cambia a error, redirige a la pantalla de error técnico.

7.4.4 RESULTADO

La pantalla de resultado presenta tres vistas especializadas según el tipo de documento procesado:

Nota simple. Presenta el título del inmueble con dirección, IDUFIR y referencia catastral junto a un chip de estado. Cuatro tarjetas de métricas resumen los campos extraídos sobre el total disponible, las discrepancias encontradas, la confianza media ponderada por el agente

y los anejos detectados. Una tabla de dos columnas muestra todos los campos extraídos. Si existen discrepancias, un panel lateral "Hallazgos" las detalla con título y descripción. La pantalla permite descargar el resultado como informe de texto plano o como JSON estructurado.

Imágenes. La vista más compleja incluye una tabla de cobertura por tipo de estancia (fachada, portal, entorno, dormitorios, cocina, baños, terraza, etc.) con el número de fotografías aportadas, una barra de calidad visual estimada y un chip de estado (Cubierto / Falta). Un panel lateral muestra la plantilla de validación aplicada con las categorías requeridas y opcionales según la tipología y tipo de visita. Cuando la API real ha procesado el conjunto fotográfico, se renderiza una sección adicional de auditoría técnica con el inventario de estancias únicas, la validación de cumplimiento de normativa, el análisis de conservación y vicios ocultos, y el detalle imagen a imagen emitido por el modelo.

Croquis. Presenta un veredicto binario OK / NO OK con código de color. El panel principal detalla cada archivo procesado con motivo, nivel de confianza y descripción breve emitidos por el agente. Un panel lateral resume el total de archivos, los válidos, los errores y el cumplimiento de normativa. En caso de resultado negativo se lista el hallazgo principal con su descripción.

7.5 CLIENTE HTTP Y COMUNICACIÓN CON EL BFF

El módulo de cliente (web/src/api/client.js) envuelve la Fetch API nativa del navegador en cuatro funciones exportadas: getDashboard(), getValidations(), getValidationDetail(id) y createValidation(payload). Para operaciones de creación los datos se serializan en FormData para permitir la subida de archivos binarios; para operaciones de lectura se utiliza JSON. La URL base es configurable mediante la variable de entorno VITE_API_URL, lo que facilita el despliegue en entornos distintos al local sin modificar el código fuente.

Capítulo 8. ANÁLISIS DE RESULTADOS

Para verificar la viabilidad técnica, la precisión y la eficiencia operativa de la plataforma **AIValidAI**, el sistema se sometió a una fase intensiva de pruebas utilizando un lote de **10 expedientes de tasación históricos reales** provistos por la empresa **VALUM**.

8.1 DATASET DE PRUEBAS Y CRITERIO DE PRIORIZACIÓN

Para verificar la viabilidad técnica, la precisión y la eficiencia operativa de la plataforma **AIValidAI**, el sistema se sometió a una fase intensiva de pruebas utilizando un lote de **10 expedientes de tasación históricos reales** provistos por la empresa **VALUM**. Cada uno de estos casos cuenta con su correspondiente auditoría de tiempos de ejecución y matrices de confusión detalladas para contrastar los valores obtenidos por la inteligencia artificial frente a los valores reales validados manualmente por los técnicos.

Debido al alcance temporal acotado para el desarrollo de este Trabajo de Fin de Máster, la fase de evaluación priorizó estratégicamente los tipos de activos más importantes y con mayor volumen de tasación diaria en el mercado inmobiliario real. De este modo, se optimizaron los esfuerzos enfocándolos en las tipologías de fincas que mayor impacto económico directo reportan a la organización.

La distribución del dataset de prueba de los 10 casos reales se detalla a continuación:

- **Activos Residenciales (70% del dataset):** Representa el núcleo troncal del negocio de tasación. Se evaluaron 7 expedientes complejos (Casos `VL-14087`, `VL-45919`, `VL-61233`, `VL-69975`, `VL-73112`, `VL-92447` y un caso

análogo), caracterizados por densas descripciones de superficies habitables, linderos comunes y direcciones postales variables.

- **Activos Industriales (10% del dataset):** Evaluado mediante el caso de control `VL-45643` (un complejo en el Registro de la Propiedad de Villafranca del Bierzo) y el caso `VL-96693`. Estos expedientes sirvieron para estresar al sistema ante textos de gran extensión con altas tasas de cargas registrales complejas (como múltiples afecciones fiscales acumuladas e hipotecas corporativas de gran volumen).
- **Suelos y Terrenos (10% del dataset):** Testeado a través del expediente `VL-54334` (Registro de la Propiedad de Girona), enfocado en analizar la capacidad de extracción semántica ante fincas rústicas o parcelas con baja densidad constructiva.
- **Elementos Anexos (10% del dataset):** Evaluado con el caso de control `VL-84674` (Registro de la Propiedad de Barcelona), diseñado para comprobar el comportamiento del sistema ante elementos secundarios vinculados contractualmente que poseen fincas registrales independientes (como plazas de aparcamiento o cuartos trasteros).

Al concentrar el lote de pruebas en este espectro, se garantiza que las métricas obtenidas reflejen fielmente el rendimiento que la plataforma ofrecerá en el grueso operativo de la empresa de tasación, dejando las tipologías residuales o de menor frecuencia para futuras extensiones del software.

8.2 Resultados del Pipeline de Notas Simples

El procesamiento de texto jurídico mediante grafos dinámicos en LangGraph y el modelo **Amazon Nova Pro** ha arrojado métricas de alta fiabilidad. A continuación, se presenta la matriz resumen general de los 10 casos reales auditados por el sistema:

Tabla 6- Resumen de resultados y métricas de rendimiento del pipeline de notas simples

ID Caso	Tipo Propiedad	Registro de la Propiedad	IDUFIR / CRU	Sup. Const. (m2)	Nº Cargas	Tiempo (s)
VL-2026-14087	RESIDENCIAL	Madrid 40	28146000151437	106.83	3	92.20
VL-2026-45643	INDUSTRIAL	Villafranca del Bierzo	24010000789027	1436.00	5	28.15
VL-2026-45919	RESIDENCIAL	Jerez de la Frontera 1	11018000347588	51.61	7	42.82
VL-2026-54334	TERRENO	Girona 3	17011000452775	100.00	2	27.59

VL-2026-61233	RESIDENCIAL	Madrid 18	28100001060846	165.66	1	29.61
VL-2026-69975	RESIDENCIAL	Córdoba 1	14018000435138	240.55	0	17.58
VL-2026-73112	RESIDENCIAL	Logroño 3	26011000594152	105.80	6	22.41
VL-2026-84674	ANEXO	Barcelona 26	08121000263430	*No consta*	6	24.60
VL-2026-92447	RESIDENCIAL	Barcelona 22	08073000383352	48.00	7	37.95

VL-2026-96693	INDUSTRIAL	Utrera 2	41039000565803	310.85	7	25.29

8.2.1 Análisis de la Eficiencia Temporal

El tiempo total de procesamiento para el pipeline documental osciló desde los **17.58 segundos** (en el escenario residencial más directo, caso `VL-2026-69975`) hasta los **92.20 segundos** (en el expediente `VL-2026-14087`).

Esta variación lineal responde directamente a la longitud del PDF original de la nota simple y a la densidad del redactado de sus cláusulas jurídicas. El caso más longevo (`VL-2026-14087`) presentaba una alta complejidad semántica en la delimitación de linderos y titulares, lo que obligó al middleware a realizar un mayor número de llamadas conversacionales y reintentos automáticos para consolidar el JSON. Aun así, la media del sistema se sitúa por debajo de los **35 segundos**, un hito computacional extraordinario frente al proceso tradicional.

8.3 Resultados de los Pipelines Gráficos (Croquis e Imágenes)

A diferencia de la naturaleza ambigua del lenguaje natural que caracteriza a las notas simples, los módulos encargados del procesamiento de información técnica visual (`agent_croquis.py` y `image_analyzer.py`) operaron bajo reglas estrictas de catalogación y filtros binarios semánticos.

Los resultados obtenidos en este bloque registraron un **100% de éxito operativo**:

- **Filtro de Integridad de Planos (`agent_croquis.py`):** El agente guardián identificó con total exactitud todos los soportes gráficos legibles. Asimismo, interceptó de manera infalible los casos de control introducidos deliberadamente en estado corrupto, ilegible o totalmente en blanco, asignando el estado correcto (`OK`, `BLANCO` o `SIN_CROQUIS`) sin registrar ningún falso positivo ni falso negativo.
- **Inventariado de Estancias (`image_analyzer.py`):** El modelo multimodal procesó con precisión absoluta las matrices de píxeles transmitidas en base64. El sistema mapeó correctamente la totalidad de las dependencias clave (cocinas, baños, salones y fachadas), cruzando los datos visuales con los requisitos normativos de **VALUM** y alertando con total eficacia en aquellos casos donde faltaba documentación gráfica obligatoria.

Capítulo 9. DISCUSIÓN

9.1 FORTALEZAS Y DATOS CRÍTICOS DE ALTA FIABILIDAD

El examen minucioso de los resultados obtenidos tras el despliegue del lote de pruebas permite evaluar con rigor el comportamiento cognitivo del agente creado. La naturaleza dispar de los tres pipelines implementados exige hablar por separado del rendimiento de cada una de ellas.

9.2 ÉXITO ABSOLUTO EN CROQUIS E IMÁGENES

Los módulos de control visual (`agent_croquis.py` e `image_analyzer.py`) funcionaron con un **100% de éxito**. El agente de croquis detectó perfectamente los planos válidos y filtró los archivos corruptos o en blanco de control. Por su parte, el analizador de imágenes inventarió correctamente todas las estancias obligatorias (cocina, baños, salón, fachadas) procesando las fotos en base64 sin cometer ni un solo error.

9.2.1 ERRORES MINIMOS Y DE BAJO VALOR EN NOTAS SIMPLES

En el análisis de las notas simples, el sistema extrajo con total exactitud los datos numéricos y de mayor riesgo (IDUFIR/CRU, número de cargas y metros cuadrados). Los pequeños desajustes se concentraron únicamente en campos de texto libre, como omitir el número de un Registro (extraer *"Registro de Barcelona"* en lugar de *"Registro de Barcelona Nº 22"*). **Estos errores son mínimos y de muy poco valor**, ya que no ponen en riesgo la seguridad de la tasación y el validador de **VALUM** puede corregirlos visualmente en segundos.

9.2.2 EL FACTOR DE LA REDACCIÓN REGISTRAL

Es fundamental destacar que **los pocos casos donde la IA se equivocó coinciden con textos que también generarían dudas en una validación manual**. Las notas simples no tienen un formato fijo y suelen presentar **pésimas explicaciones, redacciones enrevesadas y sintaxis arcaica**. Cuando el documento original es ambiguo, incluso dos técnicos humanos pueden interpretar cosas distintas. Por tanto, estas pequeñas discrepancias se deben a la mala calidad del redactado del Registro y no a un fallo de diseño de la plataforma.

9.3 EVALUACION DE LA VIABILIDAD TEMPORAL Y TOLERANCIA A FALLOS

9.3.1 OPTIMIZACIÓN DEL TIEMPO DE PROCESAMIENTO

Los datos reales de las pruebas demuestran una eficiencia temporal excelente, con una media de procesamiento por expediente inferior a los **35 segundos**. La fluctuación de los tiempos —desde los **17.58 segundos** del caso más rápido (`VL-2026-69975`) hasta los **92.20 segundos** del más complejo (`VL-2026-14087`)— depende exclusivamente de la longitud del PDF de la nota simple y del volumen de texto que la IA debe analizar. Reducir un proceso manual que a un técnico de **VALUM** le cuesta entre 15 y 20 minutos a menos de un minuto automatizado valida por completo la viabilidad del software en un entorno de producción

9.3.2 REDUCCIÓN DRASTICA FRENTE AL PROCESO MANUAL

El verdadero valor de esta optimización se aprecia al compararlo con el flujo de trabajo tradicional de **VALUM**. Actualmente, un técnico humano necesita entre **15 y 20 minutos** por expediente para abrir los documentos, leer secuencialmente la nota simple, localizar las cargas y transcribir manualmente los datos a la plataforma de tasación. Al reducir este tiempo a menos de un minuto automatizado, el sistema valida por completo su viabilidad técnica y demuestra su capacidad para eliminar los cuellos de botella operativos en la compañía.

9.4 IMPACTO OPERATIVO Y RETORNO DE LA INVERSION (ROI) EN VALUM

El análisis puramente de negocio. Contrastar el proceso manual (15-20 minutos por expediente) con el automático (media inferior a 35 segundos). Se discute cómo esto libera a los validadores humanos de tareas mecánicas de transcripción para enfocarlos en la auditoría de riesgos de alto valor.

9.4.1 Transformación del Rol del Validador Humano

Al automatizar la extracción de datos y la auditoría visual con una media de 35 segundos por expediente, el técnico humano deja de actuar como un mero transcriptor de datos introducidos a mano. Su rol evoluciona hacia el de un **supervisor o auditor de calidad**, quien simplemente recibe el JSON pre-cumplimentado por la IA junto con las alertas del sistema y se limita a validar visualmente que todo sea correcto. Esto reduce drásticamente la fatiga cognitiva y elimina los errores humanos por cansancio.

9.4.2 Retorno de la Inversión (ROI) y Escalabilidad

La reducción del tiempo operativo en más de un 90% por expediente tiene un impacto financiero directo e inmediato. Permite a la compañía absorber picos de demanda y **multiplicar su capacidad diaria de tramitación de tasaciones** sin necesidad de incrementar los costes estructurales ni el personal de validación. El coste computacional de las consultas a la API de Amazon Bedrock (Nova Pro) es residual en comparación con las

horas de trabajo humano ahorradas, lo que garantiza un retorno de la inversión a muy corto plazo y convierte al software en una ventaja competitiva clave en el sector.

Coste del empleado: Un tasador/validador medio tiene un coste de empresa (salario base + seguridad social a cargo de la empresa) de unos **18 € / hora** (aproximadamente 24.000 € brutos anuales).

Coste de Amazon Nova Pro: Los precios oficiales en AWS Bedrock son de **0,0008 $** por cada 1.000 tokens de entrada y **0,0032 $** por cada 1.000 tokens de salida. Una nota simple típica de 4-5 páginas consume (entre el texto extraído y las consultas de los agentes) unos 15.000 tokens de entrada y genera unos 1.500 tokens de salida en el JSON final. Esto equivale a unos **0,0168 $ (unos 0,015 €) por llamada**.

9.4.3 Comparativa de Costes y Capacidad Operativa (Humano vs. IA)

Para cuantificar el Retorno de la Inversión (ROI) de la plataforma en la operativa real de **VALUM**, se ha realizado un análisis de costes y capacidad de procesamiento por hora, contrastando el rendimiento de un tasador humano frente al pipeline automatizado con **Amazon Nova Pro**:

1. Capacidad y Coste del Validador Humano

- **Rendimiento:** Un técnico humano tarda una media de **15 a 20 minutos** por expediente (abrir el PDF, leerlo, analizarlo y picar los datos). Por tanto, en 1 hora de trabajo continuo puede validar un máximo de **3 notas simples**.

- **Coste Laboral:** Calculando un coste medio por hora para la empresa de **18,00 €** (incluyendo salario y costes sociales), el coste real de validación por cada nota simple asciende a **6,00 €**.

2. Capacidad y Coste del Pipeline de IA

Rendimiento: Con una media de procesamiento inferior a los **35 segundos** extraída de las pruebas reales, el sistema automatizado es capaz de procesar **100 notas simples en 1 hora** de manera desatendida.
Coste de Infraestructura (API): Evaluando un consumo estimado alto de 15.000 tokens de entrada y 1.500 de salida por expediente bajo las tarifas oficiales de Amazon Bedrock para Nova Pro, el coste real por llamada es de aproximadamente **0,015 €**.

Conclusión del Análisis

Este balance demuestra que el coste de la inteligencia artificial es prácticamente residual frente al coste laboral. El beneficio no radica en sustituir al personal, sino en la **escalabilidad del negocio**: por el coste de una sola hora de un operario (18 €), el sistema puede auditar y pre-cumplimentar el equivalente a **1.200 expedientes**, liberando a la plantilla de **VALUM** para que dedique su tiempo exclusivamente a la supervisión de alertas y control de riesgos de alto valor.

9.5 LIMITACIONES DEL SISTEMA Y DESAFÍOS DE INTEGRACIÓN

9.5.1 Dependencia de la Calidad del PDF de Origen

La principal limitación operativa radica en la calidad visual de los documentos. Si una nota simple histórica está mal escaneada, torcida, borrosa o ilegible, el proceso de extracción pierde precisión. El sistema no hace milagros con documentos defectuosos, por lo que en estos casos la revisión y corrección manual del técnico de **VALUM** sigue siendo obligatoria.

9.5.2 Restricciones en la Arquitectura Multimodal LangChain

Durante el desarrollo se detectó que las librerías estándar de LangChain fallaban o daban errores de formato al intentar enviar imágenes pesadas (planos y fotos periciales) junto con prompts de texto hacia la API de Amazon Bedrock. Para solucionar este atasco técnico, se decidió saltar el intermediario y programar la conexión a mano usando `boto3`. Al convertir las imágenes a texto binario (Base64) directamente en memoria, el sistema eliminó la latencia y logró una estabilidad total.

Capítulo 10. CONCLUSIONES Y TRABAJOS FUTUROS

10.1 CONCLUSIONES DEL PROYECTO

El desarrollo e implementación de **AIValidAI** bajo la arquitectura de Amazon Bedrock y LangGraph permite extraer tres conclusiones estratégicas fundamentales:

- **Ganancia Radical en Tiempo y Calidad:** El proyecto cumple con creces su objetivo principal. Hemos conseguido transformar un proceso manual lento y propenso a despistes en un flujo automatizado de alta fidelidad. Pasar de una media de 20 minutos de revisión humana a un procesamiento inferior a los **35 segundos** demuestra que es posible multiplicar la velocidad de una empresa de tasación sin perder un ápice de rigurosidad ni calidad en el control de riesgos.
- **El Poder de los Agentes frente a la IA Tradicional:** La mayor revelación técnica del proyecto es el beneficio de la arquitectura de agentes especialistas. Al contrario que un único modelo de lenguaje genérico (que suele inventar datos o saturarse), fragmentar el problema en agentes independientes con roles claros (`router`, `croquis`, `visión`) ha demostrado ser el camino correcto para automatizar tareas lógicas complejas.
- **Transformación del Entorno Laboral:** Esta tecnología demuestra que la IA no viene a sustituir al profesional, sino a dignificar su trabajo. Al delegar la extracción

mecánica de datos y el conteo visual de estancias en la plataforma, los técnicos de **VALUM** se liberan de la fatiga cognitiva del "corta y pega", convirtiéndose en auditores críticos de alto valor para la compañía.

10.2 LÍNEAS DE FUTURO

Para la evolución a medio y largo plazo de la plataforma, se proponen los siguientes tres ejes de desarrollo:

10.2.1 AUTOMATIZACIÓN INTEGRAL DEL FLUJO DE TASACIÓN

El paso natural es diseñar nuevos agentes especialistas que cubran el resto de áreas de un informe de tasación (como el análisis del mercado testigo, el cálculo del valor de reemplazo bruto o la verificación de la normativa urbanística municipal). El objetivo final es alcanzar un ecosistema donde **todo el proceso de tasación esté automatizado de principio a fin**, quedando la intervención humana relegada exclusivamente a la firma y visado del informe.

10.2.2 VALIDACIÓN CRUZADA E INTELIGENCIA DE NEGOCIO

Actualmente, los tres pipelines (notas simples, croquis e imágenes) corren en paralelo. Una línea de futuro crítica es conectar sus conocimientos mediante un agente supervisor final. Este agente se encargará de realizar una **validación cruzada inteligente**: certificar automáticamente que los metros cuadrados descritos en la nota simple coincidan geométricamente con las cotas del croquis y correspondan, semánticamente, con la tipología de las estancias identificadas en el reportaje fotográfico.

10.2.3 OPTIMIZACIÓN TEMPORAL E INTELIGENCIA DE MODELOS

Aunque los tiempos actuales son excelentes, el mercado de la IA evoluciona con rapidez. Una línea de investigación abierta consistirá en probar técnicas de *Fine-Tuning* (ajuste fino) o destilación de modelos sobre LLMs más pequeños y locales (como Llama 3 o versiones *Haiku* de AWS) especializados estrictamente en el lenguaje registral español. Esto permitiría **aumentar todavía más la eficiencia temporal y reducir los costes por llamada**, manteniendo exactamente los mismos estándares de calidad y precisión obtenidos en este trabajo.

Capítulo 11. BIBLIOGRAFÍA

[1] **Banco de España.** (1981). *Orden de 18 de diciembre de 1981 sobre homologación de sociedades de tasación y regulación de los métodos de valoración*. Boletín Oficial del Estado.

[2] **Ministerio de Economía y Hacienda.** (2003). *Orden ECO/805/2003, de 27 de marzo, sobre normas de valoración de bienes inmuebles y de determinados derechos para ciertas finalidades financieras*. Boletín Oficial del Estado (BOE núm. 85).

[3] **Colegio de Registradores de la Propiedad y Mercantiles de España.** (2022). *La Nota Simple Registral: estructura, valor jurídico y pautas de interpretación en el tráfico inmobiliario*. Madrid: Centro de Estudios Registrales.

[4] **Colegio de Registradores de la Propiedad y Mercantiles de España.** (2022). *La Nota Simple Registral: estructura, valor jurídico y pautas de interpretación en el tráfico inmobiliario*. Madrid: Centro de Estudios Registrales.

[5] **Chase, H.** (2022). *LangChain: Framework for developing applications powered by language models*. GitHub repository. Recuperado de https://github.com/langchain-ai/langchain

[6] **LangChain AI.** (2024). *LangGraph: Building cyclic and stateful multi-agent workflows*. LangChain Documentation. Recuperado de https://langchain-ai.github.io/langgraph/

[7] **Yao, S., Yu, D., Zhao, J., Shafran, I., Griffiths, T. L., Cao, Y., & Narasimhan, K.** (2022). *ReAct: Synergizing reasoning and acting in language models*. arXiv preprint arXiv:2210.03629. *(Esta es la publicación científica original que inventó el patrón ReAct que usas en tus planos).*

[8] **Amazon Web Services.** (2024). *Amazon Bedrock User Guide: Accessing foundational models securely*. AWS Documentation. Recuperado de https://docs.aws.amazon.com/bedrock/

[9] **Amazon Web Services.** (2024). *Amazon Nova Pro: Technical specifications and multimodal capabilities*. AWS Model Repository.

[10] **Amazon Web Services.** (2023). *Amazon Textract Developer Guide: Intelligent Document Processing (IDP) for text extraction*. AWS Documentation.

[11] **Boto3 Developers.** (2024). *Boto3 Docs: AWS SDK for Python Reference*. Amazon Web Services. Recuperado de https://boto3.amazonaws.com/v1/documentation/api/latest/index.html

[12] **Facebook Open Source.** (2022). *React: A JavaScript library for building user interfaces (v18)*. Recuperado de https://react.dev

[13] **Vite Core Team.** (2023). *Vite: Next Generation Frontend Tooling*. Recuperado de https://vitejs.dev

[14] **Node.js Foundation.** (2023). *Express: Fast, unopinionated, minimalist web framework for Node.js*. Recuperado de https://expressjs.com

ANEXO A

ANEXO A — FUNCIÓN DE CLASIFICACIÓN TIPOLÓGICA (ROUTER_AGENTS.PY)

El presente anexo recoge el código fuente de la función get_property_type, implementada en el módulo router_agents.py. Esta función constituye el componente de mayor criticidad del pipeline de notas simples, ya que de su resultado depende qué fábrica de agentes especialistas se activa para el resto del procesamiento. Se incluye también el bloque ROUTER_PROMPT, que contiene las cuatro reglas de negocio jerárquicas descritas en el apartado 6.2.1: priorización dotacional, criterio de predominancia del suelo, restricción residencial y uso funcional real. El modelo se invoca con temperatura fija de 0 para garantizar un comportamiento completamente determinista, eliminando cualquier variabilidad en la clasificación.

```
def get_property_type(full_text, llm):
    """
    Clasifica una nota simple considerando tanto la descripción literal de la
finca
    como el uso funcional y la proporción de terreno sobre la construcción.
    Prioriza DOTACIONAL o TERRENO sobre RESIDENCIAL si hay indicios.
    """
    ROUTER_PROMPT = f"""
    Analiza el texto de esta nota simple y clasifica el tipo de inmueble en una
de estas categorías:
    - RESIDENCIAL: Viviendas, pisos, apartamentos.
    - COMERCIAL: Locales, oficinas, centros comerciales.
    - INDUSTRIAL: Naves, fábricas, polígonos.
    - TERRENO: Suelo urbano, parcelas con construcción no principal, fincas con
terreno dedicado a cultivos, era, hórreo, anexos rurales.
    - FINCA RUSTICA: Terrenos agrícolas, montes, explotaciones rurales.
```

```
    - TERCIARIO: Uso no residencial, apartamentos turísticos, alojamientos
temporales, negocios en bloque, oficinas, hoteles, locales con uso terciario
explícito.
    - DOTACIONAL: Hospitales, colegios, edificios públicos.
    - ANEXO: Plazas de garaje o trasteros que tienen finca registral propia.

    REGLAS:
    1. Prioriza DOTACIONAL si hay indicios de uso público, sanitario o educativo.
    2. Prioriza TERRENO o FINCA RUSTICA si la finca tiene más terreno que
construcción, anexos rurales, hórreo, era, corrales o usos agrícolas.
    3. RESIDENCIAL solo si la finca es principalmente vivienda/piso habitual, sin
terreno significativo.
    4. Si la descripción literal y el uso funcional difieren, aplica la categoría
**más representativa** de uso real.

    EJEMPLOS:
    TEXTO: "URBANA: CASA EN PARCELA CON HÓRREO Y TERRENO DE 400 m2, de los cuales
200 m2 corresponden a la casa y el resto a terreno anejo"
    RESULTADO: TERRENO

    TEXTO: "Vivienda en piso entresuelo, puerta segunda, sin indicio de uso
público ni terreno significativo"
    RESULTADO: RESIDENCIAL

    TEXTO: "Parcela adjudicada a PROMOCIONES SANITARIAS DE BOADILLA S.L.,
destinada a fines sanitarios"
    RESULTADO: DOTACIONAL

    TEXTO: "{full_text}"

    Devuelve únicamente la palabra en mayúsculas de la categoría seleccionada.
    """
    response = llm.invoke(ROUTER_PROMPT)
    return response.content.strip().upper()
```

ANEXO B — FÁBRICA DE AGENTES ESPECIALISTAS: EJEMPLO RESIDENCIAL (SIMPLE_NOTES_AGENT_RESIDENCIAL.PY)

El ecosistema de fábricas documentales está compuesto por ocho módulos con interfaz homogénea pero prompts especializados por tipología. Este anexo recoge el código fuente del módulo residencial como ejemplo representativo del patrón de diseño común a todas las fábricas. Se incluyen las cinco funciones constructoras de agentes que operan en paralelo sobre el texto de la nota simple: create_simple_notes_registry_agent (datos registrales), create_simple_notes_surfaces_agent (superficies), create_simple_notes_ownership_agent (titularidad), create_simple_notes_charges_agent (cargas) y create_simple_notes_anejos_agent (elementos vinculados). El resto de fábricas —terreno, comercial, industrial, terciario, dotacional, finca rústica y anejo— replican esta misma estructura variando exclusivamente las instrucciones del prompt interno de cada agente para adaptarlas a la casuística jurídica de cada tipología.

```
from langchain.agents import create_agent

from langchain.agents.middleware import SummarizationMiddleware,
ToolRetryMiddleware

from langchain_core.language_models import BaseChatModel


from ai_validai.models.loader import load_chat_model

from ai_validai.prompts.agent_prompts import (

    SIMPLE_NOTE_ANEJOS_PROMPT,

    SIMPLE_NOTE_CHARGES_PROMPT,

    SIMPLE_NOTE_OWNERSHIP_PROMPT,

    SIMPLE_NOTE_REGISTRY_PROMPT,
```

```
    SIMPLE_NOTE_SURFACES_PROMPT,
)


def _resolve_model(model: str | BaseChatModel | None) -> BaseChatModel:
    if isinstance(model, BaseChatModel):
        return model
    return load_chat_model(model=model)


def _simple_notes_middleware(
    model: BaseChatModel,
    include_summarization: bool = False,
) -> list:
    """Middleware común para agentes de nota simple."""
    middleware: list = [ToolRetryMiddleware(max_retries=2)]
    if include_summarization:
        middleware.append(
            SummarizationMiddleware(
                model=model,
                trigger=("tokens", 4000),
                keep=("messages", 10),
            )
        )
    return middleware


def create_simple_notes_registry_agent(
    model: str | BaseChatModel | None = None,
):
```

```
    """Agente que extrae solo datos registrales e identificación de la nota
simple."""
    resolved_model = _resolve_model(model)
    return create_agent(
        model=resolved_model,
        tools=[],
        system_prompt=SIMPLE_NOTE_REGISTRY_PROMPT,
        middleware=_simple_notes_middleware(model=resolved_model),
    )


def create_simple_notes_surfaces_agent(
    model: str | BaseChatModel | None = None,
):
    """Agente que extrae superficies, linderos, dirección y cuota."""
    resolved_model = _resolve_model(model)
    return create_agent(
        model=resolved_model,
        tools=[],
        system_prompt=SIMPLE_NOTE_SURFACES_PROMPT,
        middleware=_simple_notes_middleware(model=resolved_model),
    )


def create_simple_notes_ownership_agent(
    model: str | BaseChatModel | None = None,
):
    """Agente que extrae solo titularidades."""
    resolved_model = _resolve_model(model)
```

```
    return create_agent(
        model=resolved_model,
        tools=[],
        system_prompt=SIMPLE_NOTE_OWNERSHIP_PROMPT,
        middleware=_simple_notes_middleware(model=resolved_model),
    )


def create_simple_notes_charges_agent(
    model: str | BaseChatModel | None = None,
):
    """Agente que extrae cargas, observaciones y tipo de activo."""
    resolved_model = _resolve_model(model)
    return create_agent(
        model=resolved_model,
        tools=[],
        system_prompt=SIMPLE_NOTE_CHARGES_PROMPT,
        middleware=_simple_notes_middleware(model=resolved_model),
    )


def create_simple_notes_anejos_agent(
    model: str | BaseChatModel | None = None,
):
    """Agente que extrae información sobre anejos de la finca."""
    resolved_model = _resolve_model(model)
    return create_agent(
        model=resolved_model,
        tools=[],
```

```
        system_prompt=SIMPLE_NOTE_ANEJOS_PROMPT,
        middleware=_simple_notes_middleware(model=resolved_model),
    )
```

ANEXO C — Agente de Auditoría de Planos con Patrón ReAct (agent_croquis.py)

Este anexo incluye el código fuente completo del módulo agent_croquis.py, que implementa el único agente con autonomía real del sistema. A diferencia de los agentes del pipeline de notas simples, este componente no sigue un grafo de pasos predefinidos, sino que razona de forma iterativa mediante el patrón ReAct (Reasoning and Acting): evalúa qué acción necesita ejecutar, llama a la herramienta correspondiente y observa el resultado antes de decidir el siguiente paso. Se incluyen las definiciones de las funciones decoradas con @tool que permiten al agente interactuar con el entorno cloud (descarga desde S3, lectura de archivos locales) y la inicialización del agente mediante create_react_agent. El ciclo termina cuando el modelo emite uno de los tres veredictos posibles: OK, BLANCO o SIN_CROQUIS.

1. El estado global compartido — porque explica un patrón de diseño no obvio: por qué los tools pueden acceder a los clientes AWS sin recibirlos como parámetro.

python

```
_estado = {
    "input_data":      None,   # dict con el JSON de entrada
    "cliente_bedrock": None,   # boto3 bedrock-runtime
    "cliente_s3":      None,   # boto3 s3
    "resultados":      [],     # resultados por imagen
}
```

2. La detección de imagen en blanco — es la decisión técnica más singular del agente, optimiza coste evitando llamadas innecesarias al modelo.

python

```
def es_imagen_en_blanco(imagen_bytes, umbral_blanco=0.95):
    img = Image.open(io.BytesIO(imagen_bytes)).convert("L")
    pixeles = list(img.getdata())
    blancos = sum(1 for p in pixeles if p >= 240)
    return (blancos / len(pixeles)) >= umbral_blanco
```

3. La creación del agente ReAct — es el núcleo de todo el apartado y lo más referenciado en el TFM.

python

```
def crear_agente():
    llm = ChatBedrock(
        model_id=MODEL_ID,
        region_name=AWS_REGION,
        model_kwargs={"temperature": 0.1, "maxTokens": 2048},
    )
    tools = [leer_json_entrada, validar_todos_los_croquis,
             validar_croquis_desde_carpeta_s3]
    return create_react_agent(llm, tools)
```

ANEXO D — Módulo Cliente HTTP del Frontend (web/src/api/client.js)

Este anexo recoge el código fuente del módulo client.js, ubicado en web/src/api/, que centraliza toda la lógica de comunicación entre el frontend React y el BFF Express. Su inclusión cierra el ciclo arquitectónico completo del sistema: desde los agentes de IA en Amazon Bedrock hasta el navegador del usuario. El módulo expone cuatro funciones exportadas: getDashboard (métricas del panel de control), getValidations (listado de expedientes), getValidationDetail (detalle de una

validación concreta) y createValidation (inicio de una nueva validación con subida de archivo). Las operaciones de escritura serializan los datos como FormData para permitir la transmisión de archivos binarios; las de lectura utilizan JSON estándar. La URL base del BFF se configura mediante la variable de entorno VITE_API_URL, lo que permite desplegar el frontend en cualquier entorno sin modificar el código fuente.

```
const API_URL = import.meta.env.VITE_API_URL || 'http://localhost:4000/api';


async function request(path, options = {}) {
  const isFormData = options.body instanceof FormData;
  const response = await fetch(`${API_URL}${path}`, {
    headers: isFormData
      ? { ...(options.headers || {}) }
      : {
          'Content-Type': 'application/json',
          ...(options.headers || {}),
        },
    ...options,
  });

  if (!response.ok) {
    throw new Error(`Request failed: ${response.status}`);
  }

  return response.json();
}

export function getDashboard() {
```

```
  return request('/dashboard');
}

export function getValidations() {
  return request('/validaciones');
}

export function createValidation(payload) {
  const formData = new FormData();
  formData.append('fileName', payload.fileName || 'documento_sin_nombre.pdf');
  formData.append('scenario', payload.scenario || 'success');
  formData.append('documentType', payload.documentType || 'Nota simple');
  formData.append('documentKind', payload.documentKind || 'nota-simple');
  formData.append('visitType', payload.visitType || 'Interior');
  formData.append('typology', payload.typology || 'Residencial');

  if (payload.file) {
    formData.append('file', payload.file);
  }
  if (payload.s3Url) {
    formData.append('s3Url', payload.s3Url);
  }
  if (payload.s3Name) {
    formData.append('s3Name', payload.s3Name);
  }

  return request('/validaciones', {
```

```
    method: 'POST',
    body: formData,
  });
}

export function getValidationDetail(id) {
  return request(`/validaciones/${id}`);
}
```