Diseño de Arquitectura para un software que integra tecnologías de OCR y LLM para la captura de datos de procesos judiciales de juzgados civiles y penales de la ciudad de Pereira.

Daniela Delgado Galeano
Camilo Andrés Valencia Ramírez

Proyecto presentada(o) como requisito parcial para optar al título de:
Magister en Ingeniería de Software

Director(a):
(Ph.D.) Jaime Alberto Chavarriaga Lozano

Pontificia Universidad Javeriana Cali
Facultad de Ingeniería y Ciencias
Departamento de Electrónica y Ciencias de la Computación
Cali, Colombia
22 de febrero de 2026

Ficha Resumen
Proyecto de Trabajo de Grado

Título: Diseño de Arquitectura para un Software que integra tecnologías de OCR y LLM para la captura de datos de procesos judiciales de juzgados civiles y penales de la ciudad de Pereira.

1. Área de trabajo: Arquitectura de Software
2. Tipo de proyecto (Aplicado, Innovación, Investigación): Aplicado
3. Estudiante (S): Daniela Delgado Galeano, Camilo Andrés Valencia Ramírez
4. Correo electrónico: dani0812@javerianacali.edu.co, camilo0812@javerianacali.edu.co
5. Dirección y teléfono: Manzana 38 Casa 26 Poblado 2, Pereira. 3142048328, 3136907504
6. Director: Jaime Alberto Chavarriaga Lozano
7. Vinculación del director: Profesor de Maestría en Ingeniería De Software
8. Correo electrónico del director: jaime.chavarriaga@javerianacali.edu.co.
9. Co-Director (Si aplica): No aplica
10. Grupo o empresa que lo avala (Si aplica): No aplica
11. Otros grupos o empresas: No aplica
12. ODS que aplica al proyecto (Agenda 2030): Educación de Calidad
13. Palabras clave(al menos 5): Arquitectura de Software, Requerimientos Arquitecturalmente Significativos, Reconocimiento óptico de caracteres, Modelos de lenguaje de gran tamaño.
14. Fecha de inicio: 02/06/2025

15. Fecha de finalización: 26/01/2026

16. Resumen: En Colombia, el sistema SIERJU de la Rama Judicial genera una serie de indicadores y reportes trimestrales con información estadística de los procesos que se llevan a cabo en todos los juzgados del país. Lamentablemente, la recopilación y cargue de la información es un proceso manual, dispendioso y propenso a errores, teniendo en cuenta que involucra consultar algunos datos de otros sistemas de software, revisar los documentos físicos de cada proceso y cargar la información en el sistema estadístico a través de múltiples formularios web.

    Este trabajo presenta el diseño y evaluación de una solución de software orientada a automatizar la captura de información de los documentos judiciales mediante el uso de tecnologías de Reconocimiento Óptico de Caracteres (OCR) y Modelos de Lenguaje de Gran Tamaño (LLM). La arquitectura propuesta fue validada a través de un prototipo funcional, cuyos resultados evidencian la viabilidad técnica del enfoque, así como mejoras en la eficiencia y consistencia del proceso de extracción de información, aportando una alternativa tecnológica para apoyar la alimentación del sistema de información SIERJU.

Índice

1. Introducción **8**
1.1. Contexto 8
1.2. Problema 8
1.3. Objetivos 12

2. Marco teórico de referencia y antecedentes **13**
2.1. Rama Judicial de la República de Colombia 13
2.2. Arquitectura De Software 15
2.3. Reconocimiento Óptico de Caracteres (OCR) 18
2.4. Modelos de Lenguaje de Gran tamaño (LLM) 18

3. Desarrollo **21**
3.1. Metodología 21
3.2. Drivers arquitectónicos 22
3.3. Plan de iteraciones 29
3.4. Iteración 1: Diseño Funcional 29
3.5. Iteración 2: Flujo de Proceso LLM 31
3.6. Iteración 3: selección de los modelos LLM 33
3.7. Iteración 4: Integración con servicios en la nube 37
3.8. Definición de arquitectura 39
3.9. Prototipo del sistema 42

4. Evaluación **46**
4.1. Evaluación técnica 46
4.2. Evaluación de modelos 56
4.3. Discusión 58

5. Conclusiones **60**
5.1. Lecciones aprendidas 60
5.2. Trabajo futuros 61

Anexos **62**
A. Gestión de Datos Privados en el Proyecto 62
B. Pasos De La Metodología ADD 65
C. Repositorios del Proyecto 67

Índice de figuras

1. How work Large Language Models (LLMs) 19
2. Pasos y artefactos de ADD 22
3. Diagrama diseño funcional 31
4. Flujo de procesamiento Mistral OCR y GPT 4.1 mini 32
5. Diagrama modulo de extracción 36
6. Diagrama Integración Con Servicios De Nube 39
7. Diagrama De Contenedores C4 40
8. Diagrama De Despliegue Azure 41
9. Login Prototipo Extracta 42
10. Carga de Documentos Extracta 43
11. Dashboard Extracta 43
12. Estado de Carga de Proceso Extracta 44
13. Estado de Carga de Proceso Extracta 44
14. Matriz de resultados de datos extraídos 45
16. Diagrama De Despliegue Azure - Cambios Aplicados 54
15. Diagrama de Contenedores C4 – Cambios Aplicados 55
17. Comparación de porcentajes de aciertos y fallos por modelo 58

Resumen

En Colombia, el sistema SIERJU de la Rama Judicial toma información de los procesos que llevan a cabo todos los juzgados del país para generar indicadores y reportes trimestrales. Lamentablemente, la recopilación y cargue de esta información es un proceso manual, dispendioso y propenso a errores.

Este proyecto plantea una solución de software que utiliza tecnologías de Reconocimiento Óptico de Caracteres (OCR) y Modelos de lenguaje de gran tamaño (LLM) que facilite este proceso, automatizando la captura de información del sistema Justicia XXI y los documentos de los procesos judiciales en los despachos judiciales de Pereira, para su carga posterior en el sistema de información SIERJU.

El desarrollo del proyecto incluyó la definición de un conjunto de requerimientos arquitecturalmente significativos (ASR), el diseño de una arquitectura de software que satisfaga esos requerimientos y la evaluación de su viabilidad técnica a través de prototipos, experimentos y pruebas con esos prototipos.

Palabras Clave: Arquitectura de Software, Requerimientos Arquitecturalmente Significativos, Reconocimiento óptico de caracteres, Modelos de lenguaje de gran tamaño.

Abstract

In Colombia, the Judicial Branch's SIERJU system generates a series of indicators and quarterly reports with statistical information on the proceedings carried out in all the country's courts. Unfortunately, collecting and uploading this information is a manual, time-consuming, and error-prone process, as it involves consulting data from other software systems, reviewing the physical documents for each case, and uploading the information to the statistical system through multiple web forms.

This dissertation presents a software solution that combines Optical Character Recognition (OCR) and Large Language Models (LLM) technologies to automate the capture of information from the Justicia XXI system and documents from judicial

proceedings in the court offices of Pereira, for the subsequent upload into the SIERJU information system.

This project included activities for defining a set of architecturally significant requirements, designing a software architecture that meets those requirements, and evaluating the proposed design and its technical feasibility through prototypes and through experiments and tests with those prototypes.

Keywords: Software Architecture, Architecturally Significant Requirements, Optical Character Recognition, Large Language Model.

1. Introducción

1.1. Contexto

En Colombia, los despachos judiciales registran la información de los procesos que realizan dentro del Sistema Estadístico De La Rama Judicial (SIERJU) para generar estadísticas e indicadores de gestión y, posteriormente, realizar diagnósticos, tomar decisiones y hacer seguimiento al rendimiento[1]. Lamentablemente, en la actualidad, este proceso de registro de información se realiza de forma manual y, por consiguiente, es dispendioso y propenso a errores.

Aunque los despachos judiciales cuentan con sistemas de software que soportan sus procesos, tales como el sistema Justicia XXI, el Justicia XXI Web y Tyba[2], algunos de los datos solicitados por el sistema SIERJU suelen ser muy específicos, no se encuentran en esos sistemas y requieren de la inspección manual de los documentos de los procesos por parte de funcionarios de los juzgados. En la actualidad, los secretarios de los juzgados civiles y penales de la ciudad de Pereira recopilan manualmente estos datos, consolidan la información en archivos en Excel y cargan la información, una por una, en los formularios web del sistema SIERJU. No existe un sistema que consolide y cargue esta información de forma automática.

1.2. Problema

Algunos de los datos solicitados por el sistema SIERJU, tales como el grupo étnico del demandado, su identificación sexual, los procesos acumulados y reactivados, no se encuentran en los sistemas que manejan los juzgados (p.ej., el sistema Justicia XXI y el Justicia XXI Web– Tyba) y, por tanto, requieren de la inspección manual de los documentos de los procesos.

Además, el proceso de carga de información en el sistema SIERJU es un proceso manual que se hace en formularios web. Aunque algunos juzgados recopilan los datos en hojas de cálculo, estos archivos solo son usados para consolidar la información,

[1] `https://sistemaestadistico.ramajudicial.gov.co/Sierju-Web/`
[2] `https://procesojudicial.ramajudicial.gov.co/justicia21/`

no se pueden constatar automáticamente con los diferentes sistemas y el proceso de carga es manual. Por todo esto, este proceso se torna muy dispendioso y propenso a errores.

Consolidación y Carga de datos en SIERJU: En muchos juzgados del país, los secretarios de los juzgados recopilan estos datos buscando en el software y en los documentos, consolidan la información en archivos en Excel y cargan manualmente los datos en el SIERJU, cada trimestre, a través de una serie de formularios web diseñados para cada una de las jurisdicciones, categorías y especialidades[3]. Este proceso resulta no solo dispendioso y demorado, sino también propenso a errores.

Algunos juzgados civiles y penales de la ciudad de Pereira utilizan hojas de cálculo en Excel en donde los funcionarios registran los datos de los procesos en una hoja y, de forma automática, se consolida la información en una hoja adicional de "Estadísticas". Sin embargo, aunque este archivo ayuda a consolidar los datos, no facilita el proceso de carga de datos. En la práctica, los funcionarios de los juzgados al final usan este archivo como una fuente para cargar los datos manualmente, uno por uno, en los formularios web del sistema SIERJU.

Tecnologías para recopilar y cargar datos: En la actualidad, en el mercado existen una serie de tecnologías que facilitan procesos similares en otro tipo de empresas e instituciones. Por ejemplo, es posible utilizar sistemas de *Reconocimiento Óptico de Caracteres* (Optical Character Recognition, OCR) para obtener información de documentos impresos y *Modelos de Lenguaje de Gran Tamaño* (Large Language Models, LLM) para extraer información de documentos y archivos de texto no estructurados. Estas tecnologías podrían usarse para capturar información de los diferentes documentos de forma automática, independiente del formato o de la estructura de los documentos.

Desarrollo de un sistema para recopilar y cargar datos: El desarrollo de sistemas que integren soluciones de OCR y LLM para capturar la información de los

[3]Acuerdo No PSAA16-10476 del 1 de marzo de 2016

procesos judiciales y cargarla en SIERJU no es una tarea trivial debido a diferentes consideraciones:

- Los datos básicos de los procesos, tales como el número identificador del proceso, las fechas de recepción y el estado actual del mismo, se encuentran en los sistemas Justicia XXI y Tyba; sistemas sobre los que no se tiene acceso a la base de datos ni exponen un API para obtener la información.
- Otros datos de los procesos, tales como el grupo étnico o la identidad sexual del demandado, no se encuentran en los sistemas de software y deben obtenerse directamente de los documentos del proceso; documentos que no tienen un formato pre-establecido, donde la información no está estructurada y que hoy requieren de una inspección manual para extraer la información.
- Aunque algunos datos de los procesos son públicos y cualquier ciudadano puede acceder a ellos, por ejemplo en el sistema Tyba, es posible que algunos datos sean privados y se deban implementar controles de privacidad de la información, por ejemplo, evitar que parte o toda la información sea accesible por LLMs y sistemas externos a los de los juzgados.
- Además, mientras existen varias tecnologías de OCR y LLM en el mercado, en muchos casos el costo de operación de estas tecnologías puede ser excesivo y/o díficil de estimar. Por ejemplo, los servicios de LLM en nube se cobran por *tokens* (fragmentos de texto) y no siempre es fácil determinar cuanto puede costar procesar un lote de documentos. Por consiguiente, es importante proponer una solución eficiente, que permita reducir los tiempos de procesamiento con un costo de operación razonable.

Requerimientos la consolidación y carga de datos: A partir de las consideraciones arriba mencionados, es posible establecer una serie de preocupaciones y requerimientos sobre atributos de calidad que permitan orientar el diseño de la arquitectura:

→ **Confiabilidad:** El sistema debe capturar y extraer información de los sistemas y documentos de los procesos, de forma automatizada, con niveles aceptables de precisión.

→ **Seguridad de la información:** El sistema debe implementar controles que permitan garantizar la protección y la privacidad de la información.

→ **Rendimiento y eficiencia:** El sistema debe realizar los procesos de captura y carga de datos con tiempos de respuesta y costos menores a los que se obtienen hoy con el proceso manual.

→ **Compatibilidad:** El sistema debe poder funcionar sin inconvenientes en los entornos de trabajo de los juzgados, por ejemplo, operando con sistemas Windows, interactuando con entornos de nube privada y *on-premises*, sistemas con operación vía web y documentos en formato PDF, escaneados como imágenes y documentos Word sin estructura definida.

Formulación del problema

¿Cómo diseñar una arquitectura de software que integre tecnologías para automatizar de forma confiable y segura el cargue de información contenida en documentos judiciales no estructurados, con el fin de optimizar la generación de estadísticas en los despachos judiciales civiles y penales de Pereira, considerando las restricciones técnicas y operativas del entorno?

1.3. Objetivos

Categoría	Descripción
Objetivo General	Diseñar y evaluar la viabilidad técnica de un software que integre tecnologías de OCR y Modelos de Lenguaje de Gran Escala (LLM) para automatizar la captura y el cargue de información de los procesos judiciales en el sistema SIERJU, con el fin de generar estadísticas de los despachos judiciales de la ciudad de Pereira.
Objetivos Específicos	1) Establecer un conjunto de requerimientos arquitecturalmente significativos para un software que integre OCR y LLM, orientado a la captura de información del sistema Justicia XXI y de los documentos de los procesos judiciales, así como a su posterior carga en el sistema SIERJU. 2) Plantear un diseño de arquitectura de software que permita satisfacer el conjunto de requerimientos arquitecturalmente significativos definidos. 3) Evaluar y determinar la viabilidad técnica del diseño arquitectónico propuesto.
Resultados Esperados	1) Documento de Requerimientos Arquitectónicos Significativos, que identifique y priorice los requerimientos funcionales, no funcionales y los atributos de calidad que orientarán el diseño de la arquitectura. 2) Diseño de Arquitectura de Software que especifique una solución arquitectónica alineada con los requerimientos arquitecturalmente significativos propuestos. 3) Informe de Evaluación Técnica de la Arquitectura, resultado de prototipos, experimentos y/o pruebas no funcionales que permitan determinar la viabilidad técnica de la propuesta arquitectónica.

Tabla 1: Objetivos y resultados esperados del proyecto

2. Marco teórico de referencia y antecedentes

Considerando el problema expuesto, relacionado con el diseño de arquitectura para un software que permita capturar y cargar información usando tecnologías de OCR y LLM, se ve pertinente ahora profundizar en algunos aspectos teóricos relacionados. Las siguientes secciones describen la Rama Judicial de la República de Colombia, los sistemas de información que usan los Juzgados del país, el diseño de la arquitectura de software y las tecnologías de OCR y LLM.

2.1. Rama Judicial de la República de Colombia

En Colombia, la Rama Judicial es un componente central del Estado, comprende las instituciones y órganos de justicia encargadas de aplicar las normas para proteger los derechos de los ciudadanos, garantizar el cumplimiento de las obligaciones y resolver conflictos, promoviendo así una convivencia pacífica en el país. (Ministerio-Justicia, 2025)

En la Rama Judicial se consideran tres jurisdicciones, la ordinaria, la constitucional y la contencioso-administrativa. En particular, la jurisdicción ordinaria se encuentra a cargo de El Consejo Superior de la Judicatura (CSJ) y comprende todos juzgados civiles y penales del país.

2.1.1. Juzgados Civiles y Penales

El Consejo Superior de la Judicatura (CSJ) es el órgano de gobierno y administración de la jurisdicción ordinaria. Todos los juzgados del país cuentan con un número, denominación y características definidas por la Sala Administrativa del CSJ y, además, cumplen las funciones específicas establecidas por la ley procesal del municipio o circuito al cual pertenece. (Función-Pública, 2025)

En Colombia, los Juzgados pueden ser de dos tipos:

- **Juzgados Civiles**, encargados de asuntos y conflictos entre particulares, tales como contratos, propiedad, familia y demás relaciones civiles; y

- **Juzgados Penales**, que se ocupan de casos de delitos y de la aplicación de penas.

2.1.2. Sistema Estadístico de la Rama Judicial (SIERJU)

Para gestionar la jurisdicción ordinaria, el CSJ cuenta con el **Sistema Estadístico de la Rama Judicial (SIERJU)**, un sistema que permite generar estadísticas y hacer seguimiento a las labores de cada uno de los Juzgados.

El SIERJU recopila la información de los diferentes Juzgados del país. Cada trimestre, todos los Juzgados Civiles y Penales deben reportar información sobre los procesos judiciales que ingresan y egresan de sus despachos. A partir de esta información, el sistema genera una serie de indicadores de gestión que son usados por el CSJ para apoyar la toma de decisiones, hacer seguimiento al rendimiento de las corporaciones judiciales y evaluar del factor de eficiencia de todos los funcionarios de carrera[4].

Lamentablemente, aunque los despachos judiciales cuentan con una serie de sistemas de software que soportan sus procesos, estos sistemas no se integran de forma automática con el SIERJU y las tareas de carga de datos deben hacerse manualmente.

2.1.3. Otros sistemas de información de los Juzgados

Entre los sistemas de software que tienen los despachos judiciales para soportan sus procesos, cabe mencionar tres:

- **Justicia XXI**, una plataforma de gestión judicial y documental de la Rama Judicial, que funciona en tecnología cliente-servidor sobre Windows y que inició su operación en 2002. Esta plataforma incluye funcionalidades tales como reparto, registro de procesos, actuaciones, sentencias y firma electrónica. (Rama-Judicial, 2025)

- **Justicia XXI WEB**, una nueva versión de la plataforma Justicia XXI, que funciona con tecnologías web, que es desarrollada por personal al interior de la

[4]Ley 270 de 1996, Ley Estatutaria De Administración De Justicia, artículo 1

rama judicial y que inició su transición en 2015 (Rama-Judicial, 2025), y

- **Tribunal Virtual de Bolsillo de la Abogacía (TYBA)**, un software que permite a los ciudadanos acceder a la información judicial y, por ejemplo, consultar el estado de los diferentes procesos. Este sistema, que busca brindar transparencia a la ciudadanía sobre los procesos legales, ofrece información de acceso público y está sujeta a las normas de protección de datos personales. (El-Tiempo, 2023)

Como estos sistemas no están integrados entre sí, los funcionarios encargados de cargar la información en el SIERJU deben, por un lado, obtener algunos datos de varios sistemas de software y, por otro lado, obtener datos adicionales directamente de los documentos físicos de cada proceso. Al final, este proceso de recopilación y cargue de información termina siendo una tarea manual, dispendiosa y propensa a errores. El propósito de este proyecto es plantear el diseño de arquitectura para un software que facilite esta labor.

2.2. Arquitectura De Software

La **Arquitectura de Software** hace referencia al conjunto de estructuras necesarias para razonar y tomar decisiones de diseño sobre un sistema de software. Estas estructuras comprenden los elementos que hacen parte del software, las relaciones entre estos elementos y las propiedades de ambos. (Bass et al., 2021)

Diseñar apropiadamente la arquitectura de software es un aspecto clave para lograr el éxito del mismo. Por ejemplo, la capacidad que tiene cualquier software para cumplir con requerimientos no funcionales sobre atributos de calidad (*-illities*) como escalabilidad, rendimiento o seguridad dependen en gran medida de su arquitectura. Si se utiliza una arquitectura no apropiada, el software no cumplirá con ese tipo de requisitos. (Kazman and Cervantes, 2014)(Bass et al., 2021)

Richards and Ford (2025) plantean que, para diseñar y documentar la arquitectura de un software, es necesario considerar cuatro dimensiones:

- **El estilo de arquitectura**, el enfoque general sobre el tipo de elementos y patrones que se usan para el diseño del software;
- **las características de arquitectura**, el conjunto de preocupaciones, requerimientos funcionales y sobre la calidad que debe soportar el software;
- **los componentes**, el diseño de los elementos que componen el software y las relaciones entre ellos, y
- **las decisiones de arquitectura**, las diferentes consideraciones que justifican el diseño propuesto.

De forma similar, el **Attribute-Driven Design (ADD)** plantea que el diseño de la arquitectura de software debe basarse en requerimientos arquitecturalmente significativos (ASR), es decir, en un subconjunto de los requerimientos que orientan el proceso de diseño. En este método, estos orientadores (*drivers*) de arquitectura son usados para seleccionar que estilos, patrones y tácticas de arquitectura se pueden usar, para definir los elementos que hacen parte del diseño del software y para documentar el diseño y las decisiones de arquitectura. (Wojcik et al., 2006)(Wood, 2007)

Estas metodologías para el diseño de la arquitectura de un software tienen en común dos cosas: por un lado, proponen partir de los requerimientos arquitecturalmente significativos y, por otro lado, plantean que se deben seleccionar patrones y tácticas de diseño de catálogos de patrones y marcos de trabajo de arquitectura conocidos.

2.2.1. Requerimientos Arquitecturalmente Significativos (ASR)

Los Requerimientos Arquitecturalmente Significativos son requisitos que tienen un profundo efecto en la arquitectura de software, es decir, son aquellos requerimientos que, de no estar presentes, podrían cambiar dramáticamente la arquitectura del software. (Bass et al., 2021)(Wojcik et al., 2006)

Por ejemplo, algunos requerimientos sobre la seguridad de la aplicación, como un tipo específico de cifrado o la protección sobre algún tipo de ataque, pueden tener

un impacto profundo en la arquitectura. Igualmente, alguna restricción sobre si se debe usar una tecnología antigua o algún tipo de hardware específico también podría implicar la posibilidad de usar o no algunos patrones de diseño. Para un arquitecto, es importante determinar cuáles son estos requerimientos antes de iniciar el proceso de diseño.

En general, los ASR pueden clasificiarse en tres tipos:

- **Requierimientos funcionales significativos**, funcionalidades y casos de uso que hacen parte del núcleo del software y/o que requiren especial atención del arquitecto;
- **Requerimientos sobre atributos de calidad**, por ejemplo, requerimientos sobre el rendimiento, la escalabilidad o la seguridad del software, que pueden especificarse usando escenarios de calidad; y
- **Restricciones**, condiciones que debe cumplir el diseño del software o del proyecto que limitan, por ejemplo, el tipo de tecnologías a utilizar o la duración y el presupuesto del proyecto.

Para el caso concreto del software para captura y carga de información de los procesos judiciales, los autores plantean diseñar una arquitectura que combine tecnologías como el Reconocimiento Óptico de Caracteres (OCR) y los Modelo de Lenguaje de Gran Tamaño (LLM).

2.3. Reconocimiento Óptico de Caracteres (OCR)

El **Reconocimiento Óptico de Caracteres (OCR)** es una técnica de computación que permite transformar imágenes que contienen texto, tales como imágenes y archivos en formato PDF, en archivos de texto que las computadoras pueden interpretar y manipular.

Por ejemplo, cuando un usuario escanea un documento, el documento se guarda como una imagen en donde todos el contenido se representa como pixeles y, por tanto, no es posible editar o buscar texto en su contenido. Al utilizar las tecnologías de OCR, esa imagen se puede convertir de forma automática en un texto digital editable y procesable. (AWS, 2024a)

2.4. Modelos de Lenguaje de Gran tamaño (LLM)

Los **Modelos de Lenguaje de Gran Tamaño (LLM)** son una tecnología de inteligencia artificial que permite procesar y generar texto en lenguaje natural. En términos generales, estos modelos son entrenados con grandes cantidades de texto para que puedan "entender" y generar lenguaje humano de forma fluida, realizando tareas como responder preguntas, resumir textos y generar contenido. (AWS, 2024c)

Normalmente los arquitectos y desarrolladores de software toman modelos LLM que han sido entrenados previamente, Modelos de Base o Fundacionales (AWS, 2024d) como GPT[5], Llama[6] o Granite[7], y utilizan estos modelos como procesadores de lenguaje natural, usan estos modelos para interpretar texto, extraer información relevante y generar respuestas que parecen humanas.

En estos casos, los desarrolladores usan los modelos como un componente de software al que se le puede dar un contexto (*context*) (Mayer, 2023), por ejemplo, un documento de texto, y hacer preguntas (*prompts*) (Xataka, 2023) sobre ese contexto.

[5] `https://openai.com/es-ES/chatgpt/overview/`
[6] `https://www.llama.com/`
[7] `https://www.ibm.com/es-es/granite`

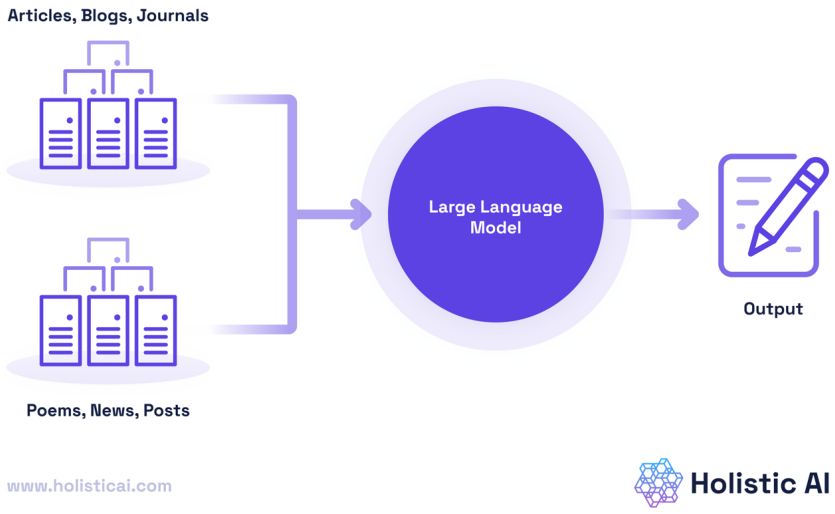

Figura 1: How work Large Language Models (LLMs)

2.4.1. Tipos de Aplicaciones basadas en LLM

Las aplicaciones que utilizan LLM se pueden clasificar de acuerdo con su funcionalidad y su arquitectura. Entre estas clasificaciones podemos mencionar:

- **Los Sistemas de Generación Pura (Pure Generation o PG)** (PureStorage, 2025) o sistemas basados en prompts, sistemas que obtienen solicitudes del usuario (prompts) y usa un modelo LLM para generar respuestas, usando solamente el conocimiento que existe en el modelo, sin usar fuentes adicionales.
- **Los Sistemas de Generación Aumentada por Recuperación (Retrieval-Augmented Generation o RAG)** (AWS, 2024b), en donde el sistema obtiene solicitudes del usuario, recupera información relacionada con la solicitud en bases de datos optimizadas para esta tarea y genera la respuesta usando algún modelo de lenguaje.
- **Los Sistemas de Preguntas-Respuestas sobre Documentos (Document Question-Answering Systems** (Hugging-Face, 2024), en donde el

sistema combina tecnologías de procesamiento de archivos y reconocimiento óptico de caracteres para extraer texto de archivos PDF y Modelos LLM para procesar y responder preguntas sobre el contenido de los documentos.

- **Los Agentes de IA**, en donde el sistema combina modelos de inteligencia artificial para obtener solicitudes del usuario, planear tareas que se requieren hacer para completar la solicitud, ejecutar las tareas planificadas, revisar los resultados y generar una respuesta. En algunos casos, los agentes utilizan ciclos de retroalimentación y, en caso de que los resultados no cumplan con ciertos criterios, se hagan modificaciones al plan y se ejecuten nuevamente las tareas hasta que se cumplan los criterios o se superen los límites de tiempo o de la cantidad de ciclos de ejecución.

Actualmente en el mercado existen herramientas, basadas en inteligencia artificial, que permiten interactuar de forma dinámica con documentos. Por ejemplo,

- **AskYourPDF (2025)**[8], una herramienta que permite cargar archivos PDF y hacer preguntas sobre su contenido, como si se estuviera conversando directamente con el documento.

- **Docling (2024)**[9], una herramienta que ofrece una experiencia similar, pero que permite trabajar no solo con archivos PDFs, sino también con tipos de archivos como PDF, DOCX, PPTX, HTML.

En este contexto herramientas como LangChain[10] (IBM, 2023) y n8n [11] juegan un papel fundamental, ya que estas proporcionan una plataforma en donde los desarrolladores pueden definir flujos de trabajo que conectan modelos LLM con fuentes de información externas, tales como bases de datos y APIs, y pueden construir aplicaciones que ejecutan estos flujos de trabajo como parte de su funcionamiento.

[8] `https://askyourpdf.com/`
[9] `https://github.com/docling-project/docling`
[10] `https://github.com/langchain-ai/langchain`
[11] `https://n8n.io/`

3. Desarrollo

3.1. Metodología

El desarrollo de este proyecto sigió un plan de trabajo basado en la metodología **Attribute Driven Design (ADD)**, un método de diseño de arquitecturas de software propuesto por el Software Engineering Institute (Wojcik et al., 2006) centrado en satisfacer requerimientos significativos a nivel funcional, a nivel no funcional y requerimientos de atributos de calidad. La figura 2 muestra los pasos de la metodología ADD. [12]

El proyecto inició definiendo el objetivo de la aplicación y un conjunto de requerimientos arquitecturalmente significativos. Luego, el proyecto desarrolló una serie de iteraciones en donde se establece un objetivo para cada iteración, se planten diseños y se hacen pruebas de concepto y experimentos para revisar que si el diseño satisface los requerimientos arquitecturalmente significativos. El proyecto realizó tres (3) iteraciones para cumplir con todos los requerimientos del proyecto.

[12] El anexo B detalla los pasos de la metodología ADD

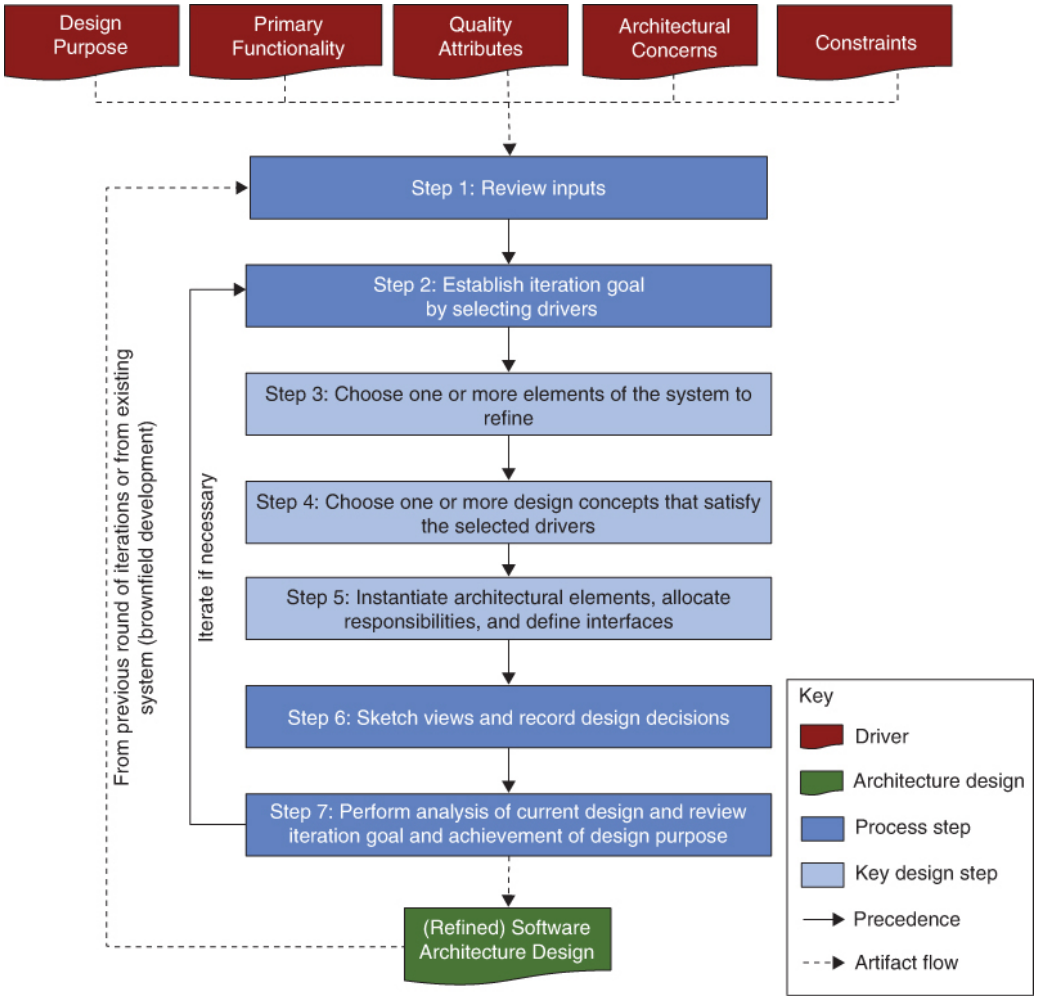

Figura 2: Pasos y artefactos de ADD
(Kazman and Cervantes, 2014)

3.2. Drivers arquitectónicos

A continuacion se describe los Requerimientos Arquitecturalmente Significativos para el proyecto.

3.2.1. Objetivo

El objetivo de la aplicación es construir un software que permita a los funcionarios de los juzgados civiles y penales de la ciudad de Pereira analizar las diferentes demandas que se reciben, extraer la información y cargar los datos en el sistema estadístico sierju.

3.2.2. Requerimientos funcionales principales

Entre las principales funcionalidades se pueden mencionar:

CU	**Nombre**	**Descripción**
CU1	Carga de Demandas	Permite a los funcionarios de los juzgados cargar las demandas en archivos PDF, escaneados del original o enviados por correo en ese formato.
CU2	Análisis de Demandas	Permite a los funcionarios de los juzgados extraer la información de manera automática.
CU3	Revisión por datos extraídos	Permite a los funcionarios revisar que los datos extraídos coincidan con la demanda original.
CU4	Reporte de datos extraídos	Permite obtener información sobre el conjunto de datos extraídos.
CU5	Cargar la información en el sistema sierju	Permite, una vez revisada la información, cargarla en el sistema sierju.

3.2.3. Requerimientos no funcionales

Entre los requerimientos no funcionales para la aplicación estan:

RC	Nombre	Descripción
RC1	Usabilidad	El sistema debe ser fácil de aprender y de utilizar por abogados sin experiencia en IA
RC2	Confiabilidad	El sistema debe extraer datos de forma confiable, en donde la mayoría de las veces extraiga los datos de forma apropiada
RC3	Seguridad	El sistema no debe permitir que usuarios no autorizados tengan acceso a la información de las demandas

3.2.4. Preocupaciones de los Stakeholders

Considerando que los *stakeholders* del proyecto son

- **Los funcionarios de los juzgados** que desean extraer información de las demandas
- **Los desarrolladores** que desean construir el software
- **Los administradores del sistema** que deben instalar, actualizar y sacar copias de respaldo

Las principales preocupaciones de los stakeholders son:

Stakeholder	Preocupación
Funcionarios de los juzgados	Que el sistema sea fácil de aprender y de utilizar por abogados sin experiencia en IA
Desarrolladores	Que el sistema sea fácil de desarrollar y de mantener (detectar y corregir errores sin introducir nuevos)

Continúa en la siguiente página...

Stakeholder	Preocupación
Administradores del sistema	Que el sistema sea fácil de administrar, en particular, detectar errores y corregir en producción y generar y restaurar copias de respaldo

3.2.5. Restricciones

Restricción	Descripción
Lenguaje de programación	Debe desarrollarse en Python
Ambiente	Debe funcionar en un entorno Web
Navegadores	Debe funcionar en Google Chrome 100 o superior

3.2.6. Escenarios de Calidad

Una vez definidos los atributos de calidad que orientan el diseño arquitectónico del sistema, es necesario establecer escenarios de calidad que permitan evaluar de forma objetiva el cumplimiento de dichos atributos. Estos escenarios permiten traducir los requerimientos arquitecturalmente significativos en situaciones concretas que describen cómo debe responder el sistema frente a distintos estímulos durante su operación.

De acuerdo con Attribute Driven Design (ADD) (Bass et al., 2021), cada escenario de calidad se define mediante seis elementos:

- **Fuente del estímulo**, usuarios o sistemas externos que generan los eventos en donde se puede medir el cumplimiento del requerimiento del atributo calidad.
- **Estímulo**, evento o condición específica que ocurre y causa una respuesta donde podemos medir el atributo de calidad.
- **Artefacto**, sistema o parte del sistema que recibe el estímulo.
- **Ambiente**, las condiciones bajo las cuáles ocurre el estímulo, p.ej., en operación normal, al momento de una sobrecarga o en algún modo de operación

específico.

- **Respuesta**, la acción que realiza el sistema ante el estímulo.
- **Medición de la respuesta**, una métrica cuantificable para evaluar el éxito de la respuesta.

De estos elementos, la *Respuesta* y la *Medición de la respuesta* resultan fundamentales, ya que permiten establecer métricas cuantificables para validar si la arquitectura satisface los requerimientos definidos en torno a los atributos de calidad.

En el contexto de esta tesis, orientada al diseño de una arquitectura de software que integra tecnologías OCR y LLM los escenarios de calidad permiten evaluar aspectos clave como la confiabilidad de la extracción, la seguridad de la información y la usabilidad de la aplicación.

Las siguientes tablas presentan escenarios de calidad que permiten validar el comportamiento de la arquitectura propuesta frente a los requerimientos de calidad del sistema.

Escenario de Calidad QS-01 - Usabilidad	
Fuente	Funcionario del juzgado que usa el software por primera vez.
Estímulo	Desea analizar un documento.
Artefacto	Sistema “Extracta”.
Entorno	En el ambiente actual desplegado en Azure.
Respuesta	El sistema permite que el usuario aprenda a cargar y analizar el documento sin ayuda de un usuario adicional.
Medida de Respuesta	El usuario debe completar el proceso en menos de 15 minutos.

Tabla 3: Escenario de calidad orientado a usabilidad del sistema Extracta

Escenario de Calidad QS-02 - Confiabilidad	
Fuente	Usuarios del sistema.
Estímulo	Analizan un conjunto de aproximadamente 100 documentos.
Artefacto	Sistema "Extracta".
Entorno	Ambiente de pruebas configurado con datos controlados y documentos previamente validados manualmente.
Respuesta	El sistema genera un análisis consistente con la información contenida en los documentos, minimizando errores de extracción.
Medida de Respuesta	El análisis debe ser correcto en al menos el 80 % de los documentos procesados.

Tabla 4: Escenario de calidad orientado a la confiabilidad en ambiente de pruebas

Escenario de Calidad QS-03 - Seguridad	
Fuente	Usuarios del sistema.
Estímulo	Al ingresar al sistema.
Artefacto	Sistema "Extracta".
Entorno	En el ambiente actual desplegado en Azure.
Respuesta	El sistema permite que el usuario acceda únicamente a los documentos para los cuales posee autorización.
Medida de Respuesta	El control de acceso debe cumplirse en el 100 % de los casos.

Tabla 5: Escenario de calidad orientado a la seguridad del sistema Extracta

Escenario de Calidad QS-04 - Seguridad	
Fuente	Usuarios del sistema.
Estímulo	Intentan acceder a un documento para el cual no tienen autorización (por ejemplo, mediante un URL directo).
Artefacto	Sistema “Extracta”.
Entorno	En el ambiente actual desplegado en Azure.
Respuesta	El sistema bloquea el acceso, impidiendo que el usuario visualice o descargue el documento.
Medida de Respuesta	La restricción de acceso debe cumplirse en el 100 % de los casos.

Tabla 6: Escenario de calidad orientado al control de acceso en el sistema Extracta

Escenario de Calidad QS-05 - Confiabilidad	
Fuente	Equipo de pruebas del proyecto.
Estímulo	Se ejecuta el procesamiento masivo de un lote controlado de 100 documentos.
Artefacto	Sistema “Extracta”.
Entorno	Ambiente de pruebas configurado con datos controlados y documentos previamente validados manualmente.
Respuesta	El sistema procesa los documentos sin interrupciones, errores críticos ni pérdida de información durante la ejecución del lote.
Medida de Respuesta	El sistema debe completar el procesamiento del lote con una tasa de fallos inferior al 5 % y sin caídas del servicio.

Tabla 7: Escenario de calidad orientado a la confiabilidad en ambiente de pruebas

3.3. Plan de iteraciones

Para el desarrollo del proyecto, se definieron cuatro (4) iteraciones,

#	**Iteración**	**Objetivo**
1	Diseño funcional	Determinar una arquitectura funcional que defina los módulos del sistema
2	Flujo de proceso LLM	Determinar el flujo de proceso LLM para el análisis de demandas
3	Modelos LLM	Determinar los modelos LLM a utilizar para el análisis de demandas
4	Integración con servicios en la nube	Definir la integración del sistema con servicios de nube que soporten el almacenamiento, el procesamiento asíncrono y el uso de modelos de inteligencia artificial

3.4. Iteración 1: Diseño Funcional

Objetivo Determinar una arquitectura funcional que defina los módulos principales del sistema.

Descripción de la Iteración En la primera iteración se determinó una descomposición del software en módulos, planteando para las responsabilidades para cada uno.

Descomposición de módulos El software se compone de los siguientes módulos:

#	**Módulo**	**Responsabilidades**
M1	Carga de demandas	Permitir a los funcionarios de los juzgados cargar las demandas en archivos PDF, ya sean escaneados del original o enviados por correo en ese formato.
M2	Extracción de información	Permitir a los funcionarios de los juzgados extraer la información de las demandas de manera automática.
M3	Revisión de resultados	Permitir a los funcionarios revisar que los datos extraídos coincidan con la demanda original.
M4	Carga de datos en el sistema	Permitir, una vez revisada la información, cargarla en el sistema sierju.
M5	Gestión de usuarios	No permitir que usuarios no autorizados tengan acceso a la información de las demandas.

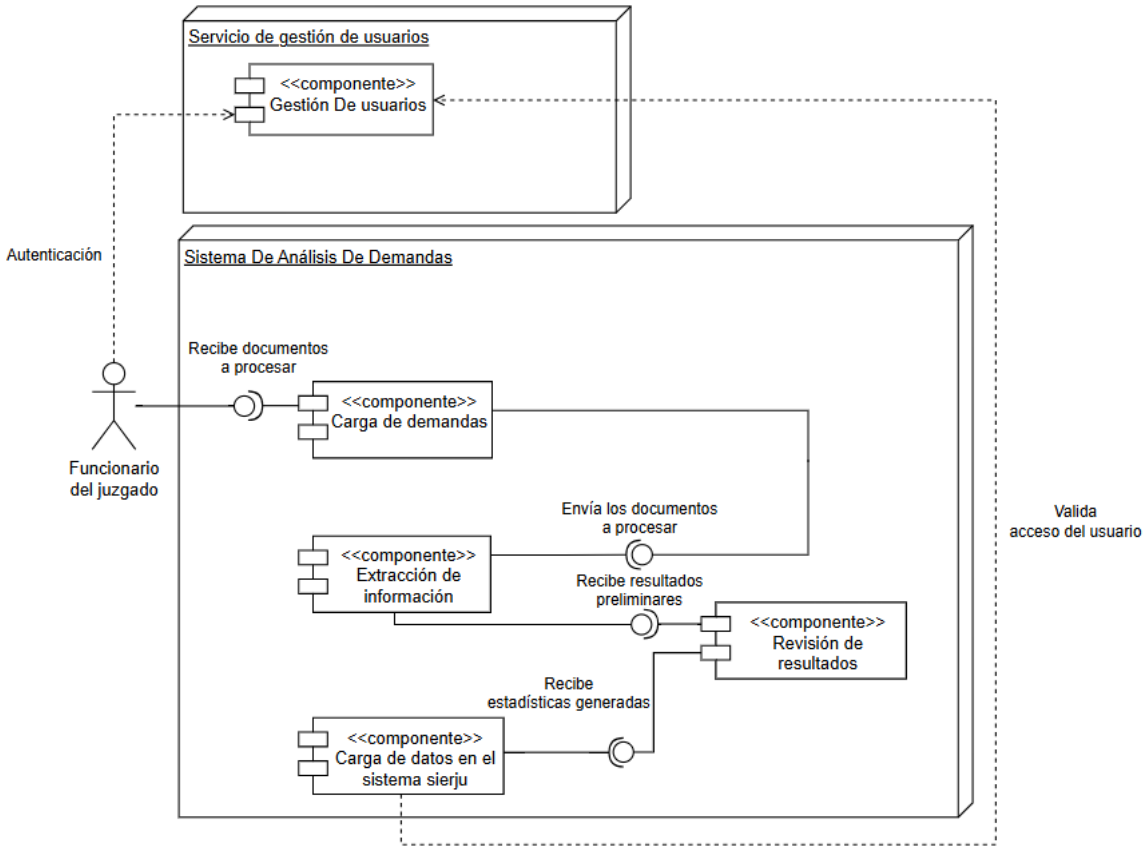

Figura 3: Diagrama diseño funcional

Diagrama de Módulos

3.5. Iteración 2: Flujo de Proceso LLM

Objetivo Determinar los componentes que hacen parte del flujo de procesamiento de los archivos de demanda haciendo uso de modelos LLM.

Descripción de la iteración En esta iteración se diseño el flujo de procesamiento, incluyento procesamiento OCR y el uso de modelos LLM usando n8n.

Diseño del flujo de procesamiento A continuación se muestra el flujo de procesamiento.

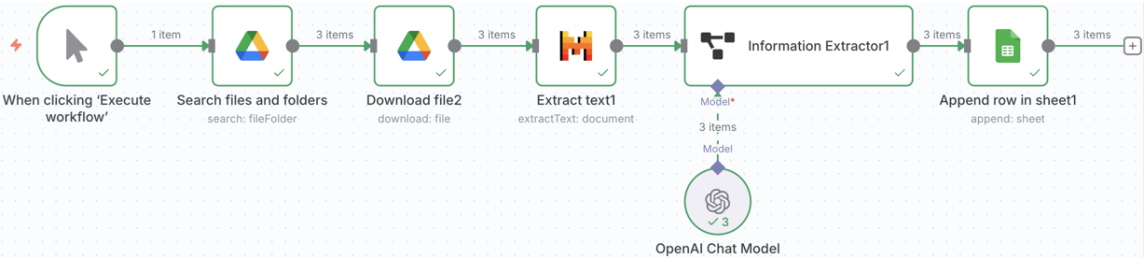

Figura 4: Flujo de procesamiento Mistral OCR y GPT 4.1 mini

El flujo incluye los siguientes elementos,

#	Componente	Responsabilidades
1	**Carga de archivo** (Search files, Download file)	Localiza y descarga los documentos judiciales desde la carpeta *n8n-Automatizaciones* en Google Drive.
2	**Ejecución del OCR** (Extract text)	Utiliza la API de Mistral (modelo mistral-ocr-latest) para convertir el contenido de los documentos a texto plano en formato Markdown.
3	**Extracción de información** (Information Extractor, OpenAI Chat Model)	Recibe el texto plano y, mediante prompts enviados al modelo GPT-4.1 Mini, extrae entidades legales específicas (como el demandado) y las estructura en JSON.
4	**Almacenamiento de datos** (Append row in sheet)	Toma los datos extraídos y los añade como una nueva fila en una hoja de cálculo de Google Sheets.

Evaluación Para revisar que el flujo permite cumplir con los requerimientos,

- Se implementó el modelo en n8n
- Se ejecutaron pruebas con 50 documentos judiciales simulados para extraer la información

3.6. Iteración 3: selección de los modelos LLM

Objetivo Determinar los modelos LLM a utilizar para el procesamiento de archivos PDF (usando OCR) y la extracción de información (para proveer el contenido del documento como contexto al modelo LLM).

Descripción de la iteración En esta iteración se hicieron una serie de experimentos para determinar el modelo más apropiado de acuerdo con los criterios de evaluación y requerimientos del proyecto.

Alternativas para los modelos de OCR En esta iteración se evaluaron los siguientes modelos de OCR,

#	**Alternativa**	**Descripción**
OCR1	Mistral OCR	Es capaz de comprender estructuras complejas como tablas y diagramas con alta precisión. Fue seleccionado por su plan de precios competitivo, que permite procesar mil páginas por dólar.
OCR2	Gemini-2.0-flash	Combina comprensión multimodal para extraer texto con alta precisión e interpretar el contexto visual, incluso en documentos de baja calidad. Su diseño es eficiente y de bajo costo operativo.

Alternativas para los modelos LLM se evaluaron también los siguientes modelos LLM,

#	Alternativa	Descripción
LLM1	Gpt 4.1 mini	Seleccionado por su bajo costo, una gran ventana de contexto de 1 millón de tokens y por conservar la capacidad de razonamiento y comprensión avanzada de los modelos GPT-4.
LLM2	Mistral large	Ofrece una alta comprensión gramatical y de contextos culturales. Posee una ventana de contexto de 32K tokens para recuperar información de documentos grandes.
LLM3	Gpt 5.0 mini	Orientado para entornos donde el rendimiento, el precio por token y las respuestas rápidas son las principales restricciones, a la vez que sigue proporcionando sólidas capacidades de propósito general.

Criterios de evaluación Para evaluar los diferentes modelos se consideraron los siguientes criterios:

#	Criterio	Descripción
1	Precisión	Se busco que la capacidad del sistema para capturar y extraer información de los documentos tenga niveles aceptables de exactitud y confiabilidad de por lo menos un 80 %. La evaluación se enfoca en la precisión del reconocimiento del texto y en que los datos extraídos coincidan con los originales.
2	Costo	Se busco una solución con un costo de operación razonable, comparando los costos y tiempos de las tecnologías OCR y LLM frente al proceso manual actual.

Diseño del experimento Para determinar los modelos a utilizar, se plantearon las siguientes actividades,

- Se estableció un conjunto de datos simulados de 50 archivos judiciales en PDF, para hacer la evaluación.
- Se extrajeron los datos simulados a mano de los archivos PDF y se generó un archivo CSV con la información extraída.
- Se realizaron una serie de iteraciones
    1. Se implementó un flujo de procesamiento en n8n para procesar cada uno de los archivos y comparar el resultado
    2. Se modificó el flujo de procesamiento para ajustar los prompts para la extracción de información
    3. Se modificó el flujo de procesamiento para usar un LLM como juez y evaluar los resultados.

Evaluación La evaluación de los modelos de lenguaje se realizó mediante un enfoque de **modelo de lenguaje como juez (*LLM as Judge*)**, siguiendo un procedimiento estructurado.

En primer lugar, se construyó un dataset de validación, previamente diseñado, verificado y aprobado por los stakeholders, el cual contiene documentos judiciales con información esperada claramente definida para cada campo a extraer.

Posteriormente, cada uno de los modelos se ejecutó sobre el mismo conjunto de datos de validación, generando salidas de extracción de información de manera independiente. Estas salidas fueron comparadas con los valores esperados del dataset por el modelo juez, el cual evaluó el grado de correspondencia entre la información extraída y la información de referencia.

El modelo juez asignó una calificación a cada resultado de extracción con base en criterios previamente establecidos, tales como exactitud, completitud y consistencia semántica. A partir de estas calificaciones, se calcularon los niveles de acierto de cada modelo, permitiendo obtener métricas comparables de desempeño.

Finalmente, los resultados de la evaluación fueron consolidados y analizados para identificar diferencias en términos de precisión y efectividad entre los modelos evaluados, los cuales se presentan de manera resumida en una tabla de resultados.

Diagrama componente de extracción A continuación se muestra un diagrama con los componentes del módulod de extracción,

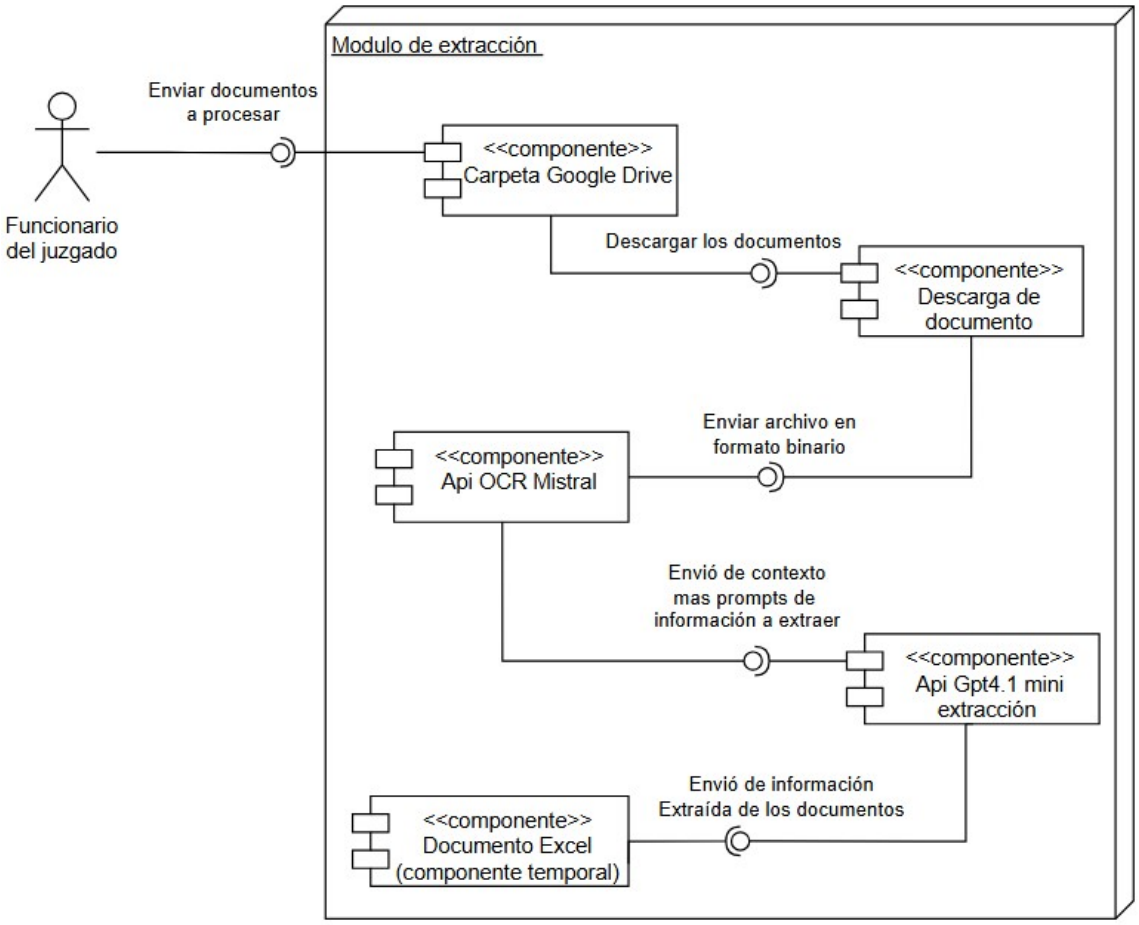

Figura 5: Diagrama modulo de extracción

3.7. Iteración 4: Integración con servicios en la nube

Objetivo Definir la integración del sistema con servicios de nube que soporten el almacenamiento, el procesamiento asíncrono y el uso de modelos de inteligencia artificial.

Descripción de la Iteración En esta iteración se establecieron los servicios de la plataforma Azure que se requieren para poner en funcionamiento los módulos del sistema. El objetivo es definir un esquema de operación que permita garantizar escalabilidad, desacoplamiento y tolerancia a fallos mediante el uso de servicios gestionados.

Integraciones principales Los principales servicios de Azure que utiliza el sistema son:

- El Backend Operativo almacena los documentos en el servicio de Blob Storage.
- Las solicitudes de procesamiento son enviadas al Queue Storage Service, permitiendo un procesamiento desacoplado.
- El Backend IA consume los mensajes de la cola y utiliza los Azure Foundry Models para la extracción de información.
- Los resultados del procesamiento y el estado de cada transacción se almacenan en Azure Cosmos DB, permitiendo el seguimiento y la auditoría del sistema.

Relación de Servicios en nube Se plantea el uso de los siguientes servicios en nube Azure:

#	**Módulo**	**Responsabilidades**
M1	Azure Cosmos DB	Almacena documentos, metadatos, resultados de extracción, estados de procesamiento y trazabilidad de las cargas.
M2	Azure Queue Storage	Almacena y distribuye mensajes de procesamiento para ejecutar de forma asíncrona la extracción de información de los documentos cargados.
M3	App Service Backend Operativo	Expone APIs para recibir documentos, orquestar procesos de extracción, gestionar estados, errores y persistir resultados.
M4	App Service Backend IA	Procesa documentos judiciales ejecutando tecnologías OCR y LLM para extraer información estructurada y validar resultados automáticamente.
M5	Azure Foundry Models Service	Provee agentes y modelos de inteligencia artificial para extraer información automáticamente de documentos judiciales.
M6	Azure Gateway	Gestiona, asegura y monitorea las APIs y servicios del sistema, controlando acceso, tráfico y versionado.
M7	App Service Frontend	Ofrece una interfaz para cargar documentos judiciales y monitorear el proceso de extracción.
M8	Blob Storage Service	Almacena los documentos judiciales cargados en el frontend, sirviendo como repositorio central para procesamiento.

Tabla 8: Módulos del sistema y sus responsabilidades

Diagrama componentes de Azure

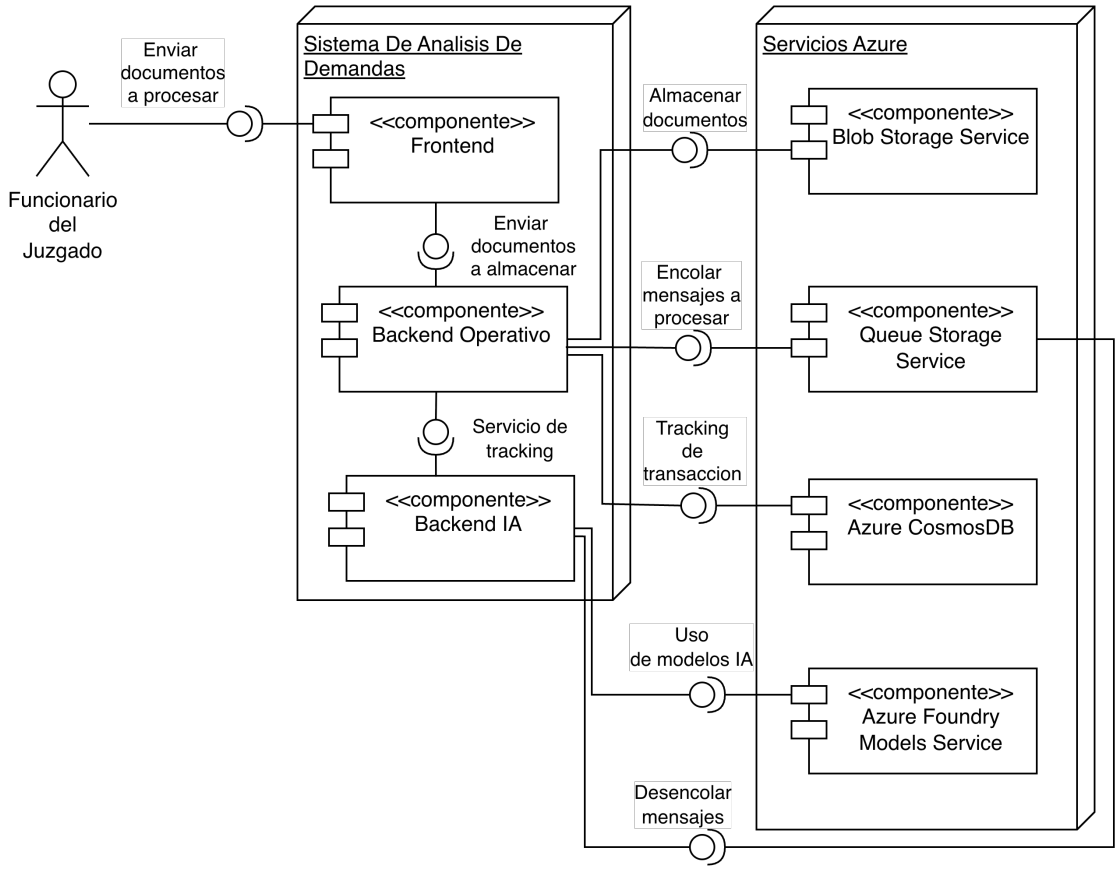

Figura 6: Diagrama Integración Con Servicios De Nube

3.8. Definición de arquitectura

Con el propósito de describir la estructura lógica y tecnológica del sistema propuesto, se presenta el Diagrama de Contenedores del modelo C4, el cual permite visualizar los principales contenedores de software que conforman la aplicación, así como sus responsabilidades, tecnologías utilizadas y relaciones de comunicación.

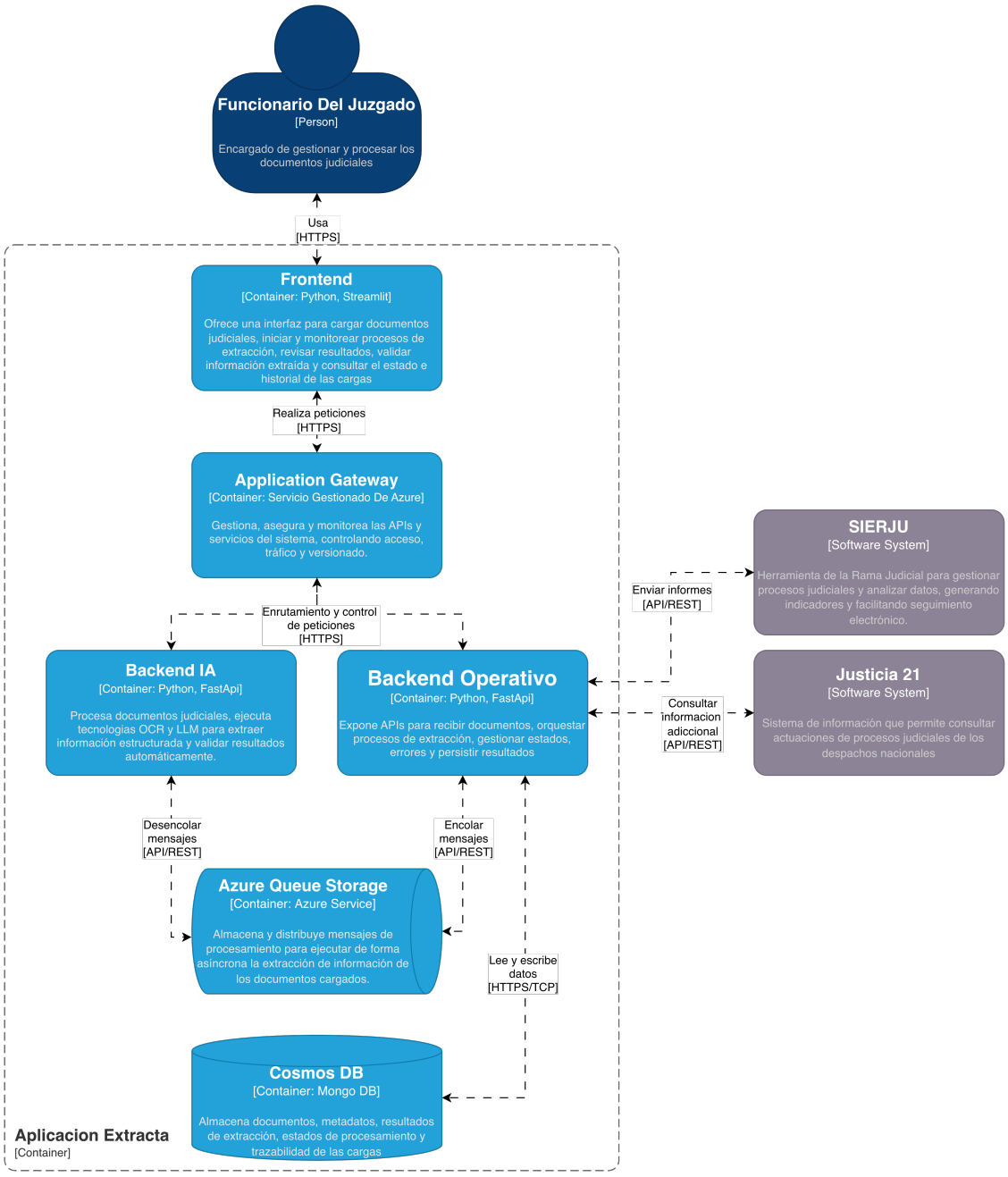

Figura 7: Diagrama De Contenedores C4

NOTA: La conexión que existe entre el componente interno del backend operativo y los sistemas externos Justicia 21 y SIERJU es hipotética ya que los autores no cuentan con los contratos de conexión y autorización formal de las entidades correspondientes para hacer integraciones con estos sistemas.

Con el fin de complementar la vista lógica presentada en el Diagrama de Contenedores C4, se presenta el Diagrama de Despliegue, el cual describe cómo se implementan y distribuyen los contenedores del sistema en la infraestructura de Azure.

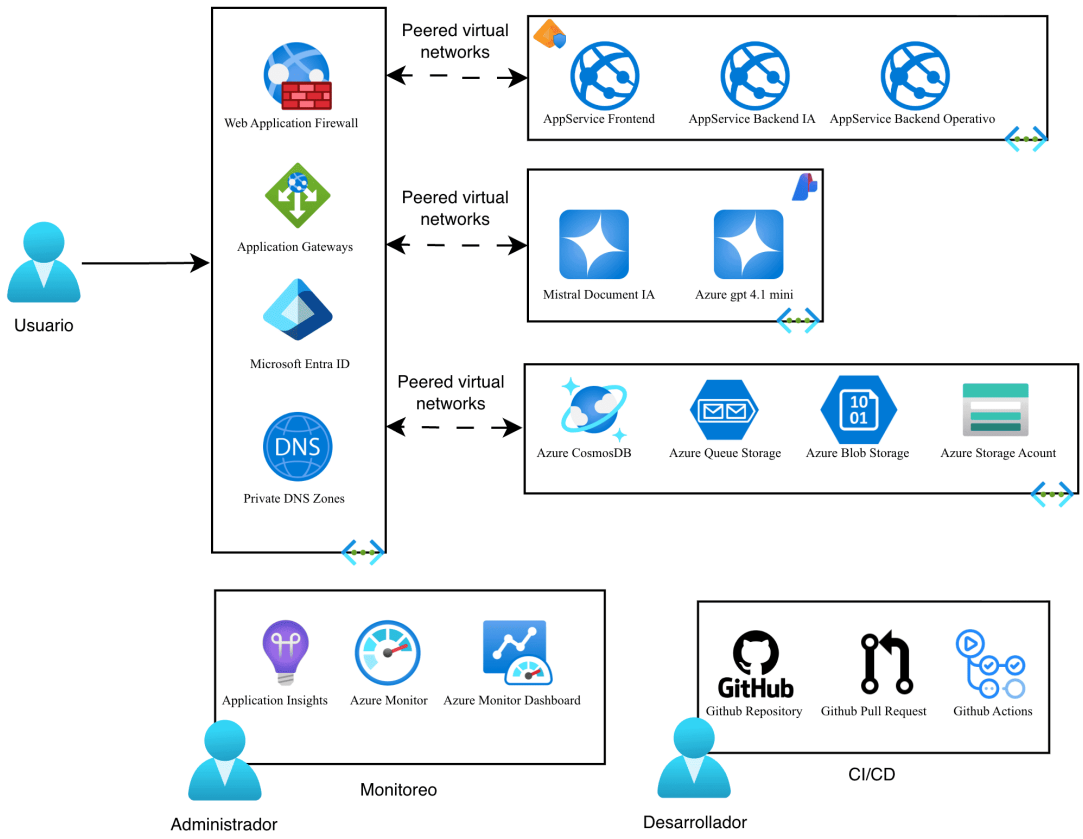

Figura 8: Diagrama De Despliegue Azure

3.9. Prototipo del sistema

El prototipo se realizó con el propósito de validar la viabilidad técnica de la arquitectura propuesta y comprobar la correcta interacción entre los módulos definidos en el diagrama C4. El prototipo se construyó siguiendo un enfoque incremental, implementando los principales contenedores del sistema y sus flujos de comunicación.

En primer lugar, se desarrolló el Frontend, usando Streamlit y Python. Este frontent permite al funcionario del juzgado cargar documentos judiciales, iniciar los procesos de extracción y monitorear el estado y los resultados de las cargas. Este componente actúa como punto de entrada al sistema y se comunica de forma segura mediante HTTPS con el backend.

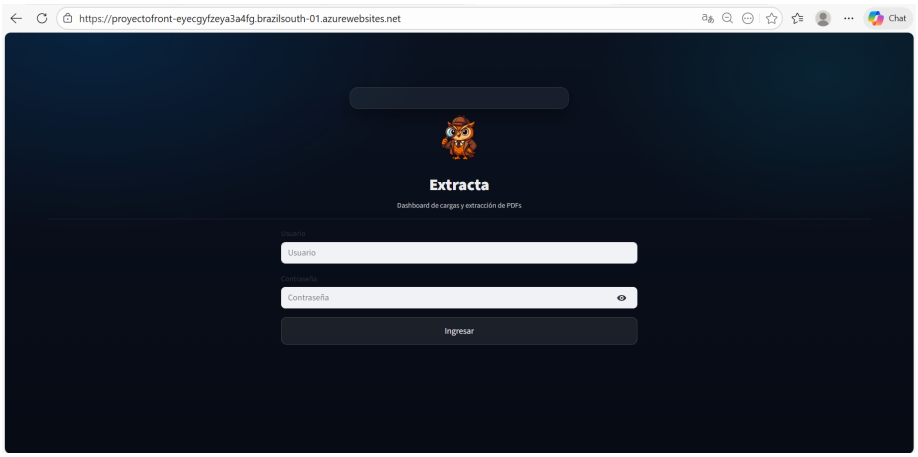

Figura 9: Login Prototipo Extracta

Posteriormente, se configuró Azure API Gateway, encargado de centralizar el acceso a las APIs del sistema. Este componente del prototipo permite gestionar y asegurar las peticiones provenientes del frontend, aplicando control de acceso, enrutamiento, versionado y monitoreo de las solicitudes hacia los servicios backend.

A nivel de procesamiento, el prototipo incorpora dos servicios backend independientes. El Backend Operativo, implementado como un servicio FastAPI, se encarga de recibir los documentos, orquestar el flujo de procesamiento, gestionar estados y errores, y persistir los resultados obtenidos. Por su parte, el Backend IA es respon-

sable de procesar los documentos judiciales mediante la ejecución de tecnologías de OCR y modelos de lenguaje de gran escala (LLM), realizando la extracción automática de información estructurada y la validación de los resultados.

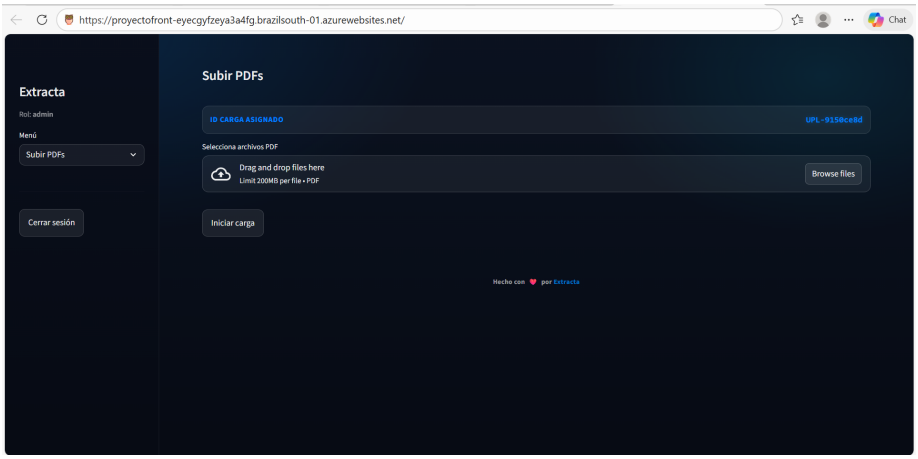

Figura 10: Carga de Documentos Extracta

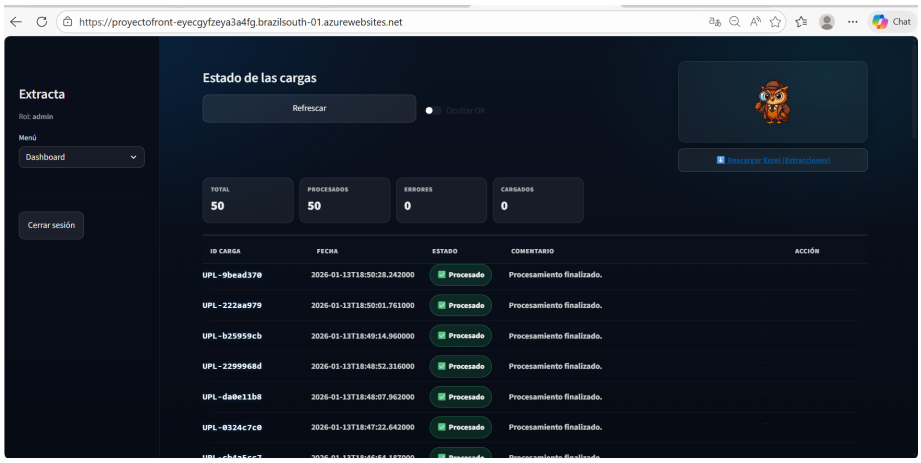

Figura 11: Dashboard Extracta

Para desacoplar la carga de documentos del procesamiento intensivo, se integró Azure Queue Storage, el cual permite encolar y desencolar mensajes de procesamiento de manera asíncrona. El Backend Operativo envía las solicitudes de procesamiento

a la cola, mientras que el Backend IA consume estos mensajes, ejecuta el análisis correspondiente y reporta los resultados.

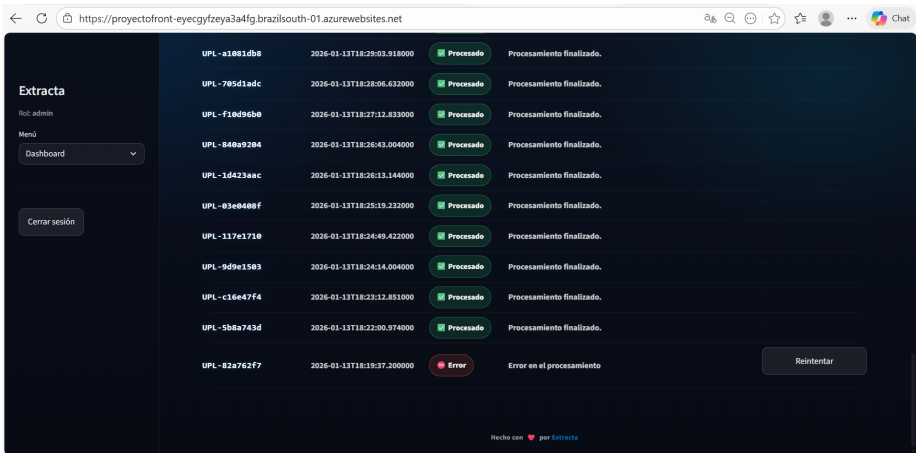

Figura 12: Estado de Carga de Proceso Extracta

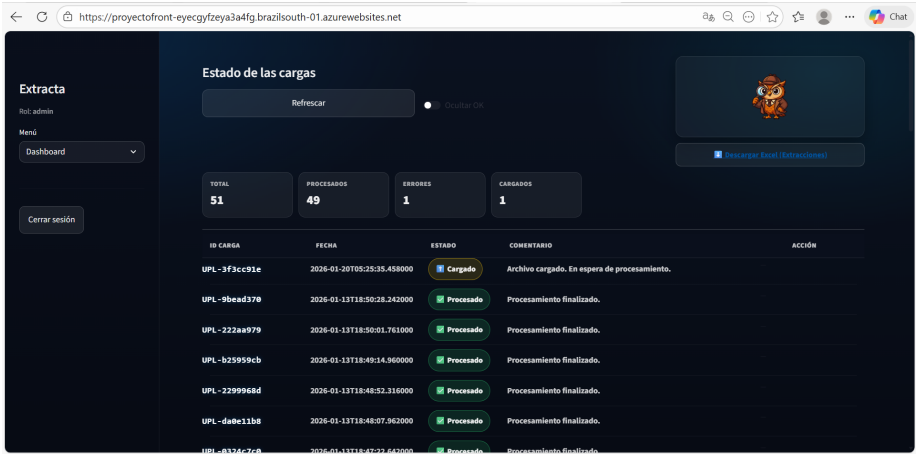

Figura 13: Estado de Carga de Proceso Extracta

Finalmente, el prototipo utiliza Azure Cosmos DB como sistema de persistencia, donde se almacenan los documentos, metadatos, resultados de extracción, estados de procesamiento y la trazabilidad completa de cada carga. Esta base de datos permite consultar el historial y el estado de los procesos desde el frontend.

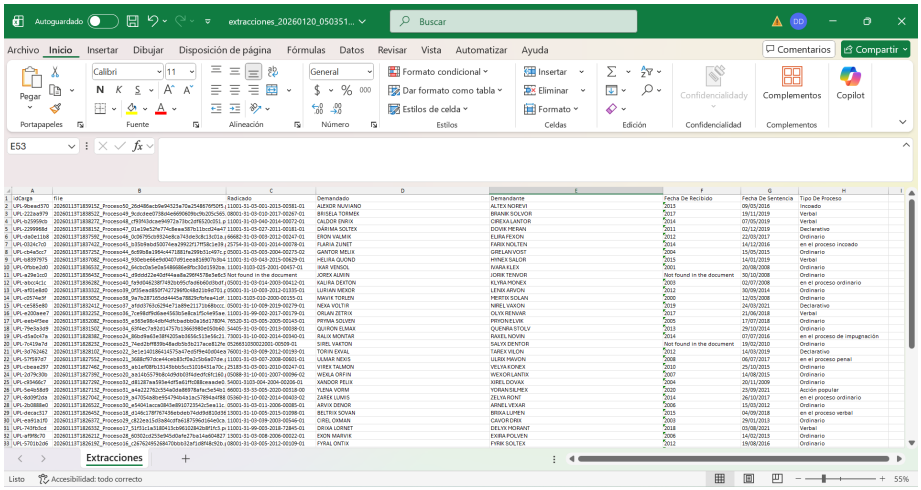

Figura 14: Matriz de resultados de datos extraídos

Adicionalmente, el prototipo contempla la integración con sistemas externos, como Justicia XXI para la consulta de información adicional de los procesos judiciales y SIERJU para el envío de informes y resultados mediante servicios API/REST, validando así la interoperabilidad del sistema con plataformas institucionales existentes.

4. Evaluación

4.1. Evaluación técnica

En esta sección se muestran los resultados de la evaluación a nivel técnico. La evaluación técnica de la arquitectura se dividió en evaluación funcional y evaluación no funcional.

4.1.1. Evaluación funcional

El propósito de la evaluación funcional era verificar que el diseño de la arquitectura tuviera la estructura necesaria (componentes y comunicaciones) para satisfacer, a alto nivel, los requerimientos funcionales principales. Cada requerimiento funcional se evaluó para verificar su funcionamiento dentro del diseño de la arquitectura. Para esto, se definió el siguiente contrato de evaluación a nivel funcional:

- Requerimiento funcional que se evalúa
- Determinar si el requerimiento funcional se debe satisfacer en la arquitectura.
- Diagrama de secuencia de alto nivel con los componentes que resuelven el requerimiento funcional.

Evaluación Funcional - Requerimiento funcional - Carga de Demandas	
Requerimiento funcional	**¿Se debe cumplir en la arquitectura?**
Carga de Demandas	Sí.

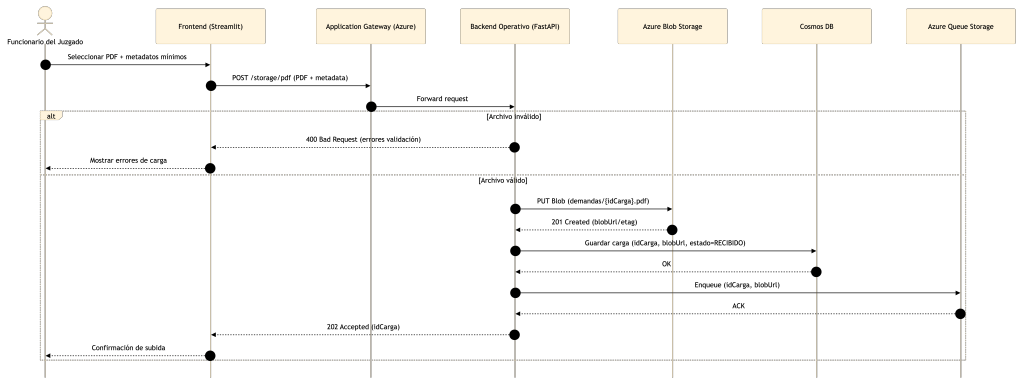

Tabla 9: Evaluación funcional del requerimiento Carga De Demandas

Evaluación Funcional - Requerimiento funcional - Análisis de demandas	
Requerimiento funcional	**¿Se debe cumplir en la arquitectura?**
Análisis de demandas	Sí

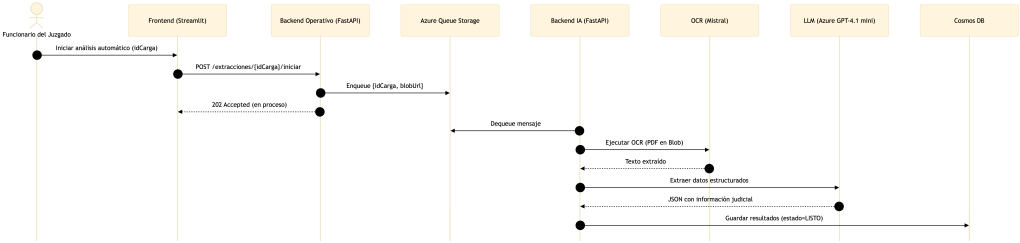

Tabla 10: Evaluación funcional del requerimiento Análisis de demandas

Evaluación Funcional - Requerimiento funcional - Revisión por datos extraídos	
Requerimiento funcional	**¿Se debe cumplir en la arquitectura?**
Revisión por datos extraídos	Sí

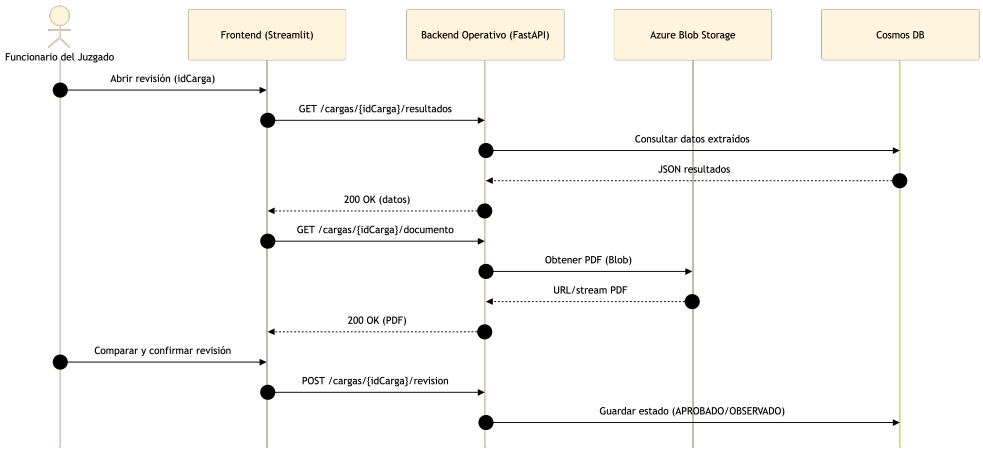

Tabla 11: Evaluación funcional del requerimiento Revisión por datos extraídos

Evaluación Funcional - Requerimiento funcional - Reporte de datos extraídos	
Requerimiento funcional	**¿Se debe cumplir en la arquitectura?**
Reporte de datos extraídos	Sí

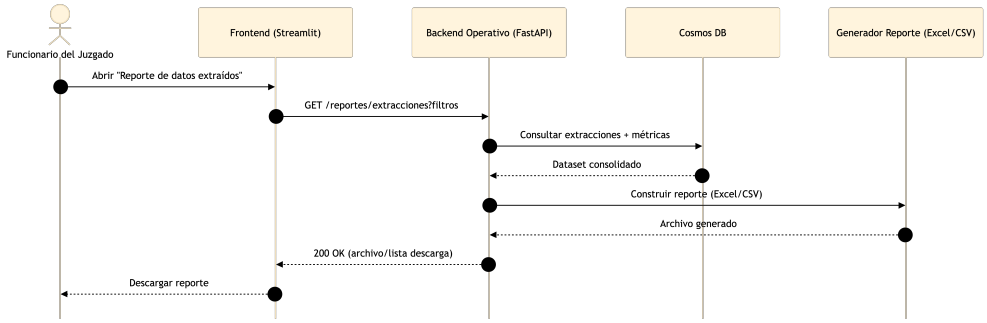

Tabla 12: Evaluación funcional del requerimiento Reporte de datos extraídos

Evaluación Funcional - Requerimiento funcional - Carga la información en sistema sierju	
Requerimiento funcional	**¿Se debe cumplir en la arquitectura?**
Carga la información en sistema sierju	Sí

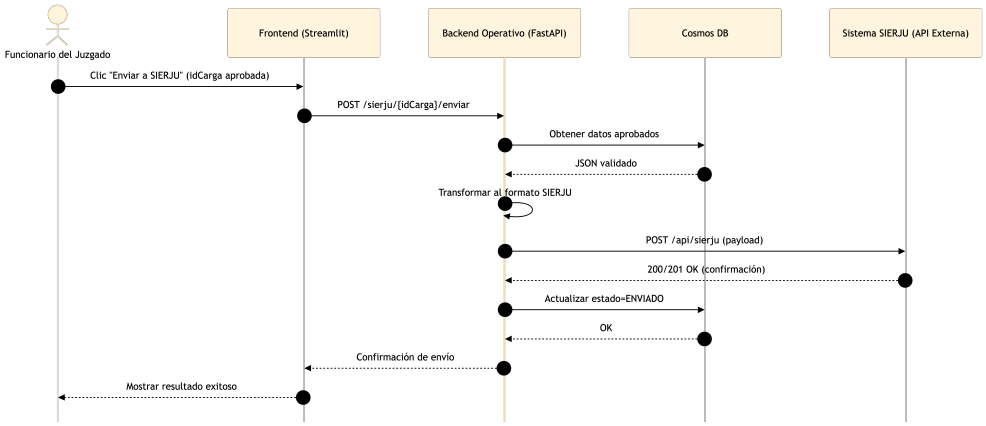

Tabla 13: Evaluación funcional del requerimiento Carga la información en sistema sierju

4.1.2. Evaluación no funcional

La evaluación no funcional de la arquitectura tiene como objetivo determinar en qué medida el diseño propuesto satisface los escenarios de calidad definidos. Esta evaluación se fundamenta en el modelo arquitectónico seleccionado y en un conjunto de criterios que permiten analizar el grado de cumplimiento de cada escenario dentro del diseño.

Clasificación de los escenarios de calidad

Los escenarios se clasifican según su nivel de cumplimiento en la arquitectura:

- **Escenario Directo:** Se satisface completamente mediante el diseño arquitectónico actual, sin requerir modificaciones adicionales.
- **Escenario Indirecto:** No se satisface inicialmente con la arquitectura propuesta. Requiere identificar, documentar e incorporar cambios estructurales o

tecnológicos para lograr su cumplimiento. Posteriormente, se debe evaluar si es necesaria una nueva validación del escenario.

- **Escenario Parcial:** No depende en un 100 % del diseño arquitectónico. Sin embargo, la arquitectura aporta mecanismos que contribuyen significativamente a su satisfacción, por lo que debe documentarse el alcance de dicha contribución.

Contrato de evaluación

Con el fin de garantizar un análisis sistemático y trazable, se definió un contrato de evaluación compuesto por los siguientes elementos:

1. **Identificación del escenario de calidad evaluado.**
2. **Clasificación del escenario** (directo, indirecto o parcial), incluyendo la justificación correspondiente.
3. **Definición de ajustes arquitectónicos**, cuando aplique, y descripción de los cambios necesarios para satisfacer el escenario.
4. **Resumen consolidado de resultados**, indicando la clasificación final y los cambios realizados en el diseño.
5. **Actualización de los modelos arquitectónicos** (Contexto y Despliegue), en caso de que se hayan incorporado modificaciones.

En las siguientes tablas se presentan los resultados detallados de la evaluación individual de cada escenario de calidad, evidenciando el nivel de cumplimiento alcanzado por la arquitectura propuesta.

Evaluación Escenario Calidad QS-01 - Usabilidad	
Escenario de Calidad	**Tipo de Escenario**
QS-01 – Usabilidad	Directo
¿Por qué es un escenario directo?	
La integración con Application Insights permite medir tiempos de interacción y monitorear el uso del sistema, soportando la validación del tiempo máximo definido para completar el proceso.	

Tabla 14: Evaluación del escenario de calidad QS-01

Evaluación Escenario Calidad QS-02 - Confiabilidad	
Escenario de Calidad	**Tipo de Escenario**
QS-02 – Confiabilidad	Indirecto
¿Por qué es un escenario indirecto?	
La arquitectura contempla el procesamiento automático, pero no define controles explícitos para asegurar el umbral de exactitud del (80 %): validaciones automáticas, reglas de consistencia, evaluación de calidad (métricas), ni un mecanismo formal de *human-in-the-loop* cuando la confianza de extracción es baja.	
Refinamiento requerido en la arquitectura	
Agregar métricas de calidad (por documento y por campo), umbrales de confianza y estados (p. ej. `REQUIERE_REVISION`); incorporar reglas de validación (consistencia de radicado, fechas, NIT/CC, etc).	

Tabla 15: Evaluación del escenario de calidad QS-02

Evaluación Escenario Calidad QS-03 - Seguridad (Autorización)	
Escenario de Calidad	**Tipo de Escenario**
QS-03 – Seguridad (Autorización)	Directo
¿Por qué es un escenario directo?	
El diseño contempla control de acceso centralizado a través del Gateway/Entra ID y la gestión de peticiones hacia los servicios. Adicionalmente, el Backend Operativo actúa como punto único para consultar resultados y entregar acceso al documento, permitiendo aplicar validaciones de autorización antes de retornar información.	

Tabla 16: Evaluación del escenario de calidad QS-03

Evaluación Escenario Calidad QS-04 - Seguridad (Bloqueo URL Directo)	
Escenario de Calidad	**Tipo de Escenario**
QS-04 – Seguridad (Bloqueo URL directo)	Directo
¿Por qué es un escenario directo?	
Dado que el acceso a los documentos se gestiona a través del Backend Operativo y el Gateway, la arquitectura permite bloquear accesos no autorizados incluso si un usuario intenta ingresar por URL directa. El Backend puede denegar la solicitud cuando no exista autorización y, adicionalmente, el Blob Storage puede mantenerse sin acceso público, evitando descargas directas sin validación previa.	

Tabla 17: Evaluación del escenario de calidad QS-04

Evaluación Escenario Calidad QS-05 - Confiabilidad (Ambiente de Pruebas)	
Escenario de Calidad	**Tipo de Escenario**
QS-05 – Confiabilidad (Ambiente de pruebas)	Directo
¿Por qué es un escenario directo?	
La arquitectura fue desplegada en un ambiente de pruebas controlado donde se ejecutó el procesamiento masivo de un lote de 100 documentos. Durante la ejecución se monitoreó el comportamiento del sistema mediante los servicios de observabilidad en Azure, verificando que no se presentaran caídas del servicio, pérdida de información ni errores críticos que interrumpieran el flujo completo de procesamiento.	
Resultado de la evaluación	
El sistema completó el procesamiento del lote dentro de los parámetros definidos, manteniendo una tasa de fallos inferior al 5 % y sin interrupciones del servicio, evidenciando estabilidad operativa en ambiente de pruebas.	

Tabla 18: Evaluación del escenario de calidad QS-05

Resumen Evaluación No Funcional		
Escenario de Calidad	**Tipo**	**Cambios realizados**
QS-01 – Usabilidad	Directo	No aplica.
QS-02 – Confiabilidad	Indirecto	Agregar métricas de calidad (por documento/campo), umbrales de confianza y estado, incluir reglas de consistencia.
QS-03 – Seguridad	Directo	No aplica.
QS-04 – Seguridad	Directo	No aplica.
QS-05 – Confiabilidad	Directo	No aplica.

Tabla 19: Resumen evaluación no funcional

Los cambios que se deben aplicar en los modelos de Contexto y Despliegue son los siguientes:

- Agregar un servicio de Azure que le permita al sistema tener una retroalimentación activa por parte de los funcionarios del juzgado.
- Vincular la información de calificación de los usuarios con los modelos de IA para mejorar su desempeño.

Estos cambios son para satisfacer el escenario indirecto QS 02 y se deben reflejar en los diagramas de estos modelos para finalizar la evaluación no funcional del diseño de la arquitectura de software.

4.1.4. Nuevo diagrama de despliegue

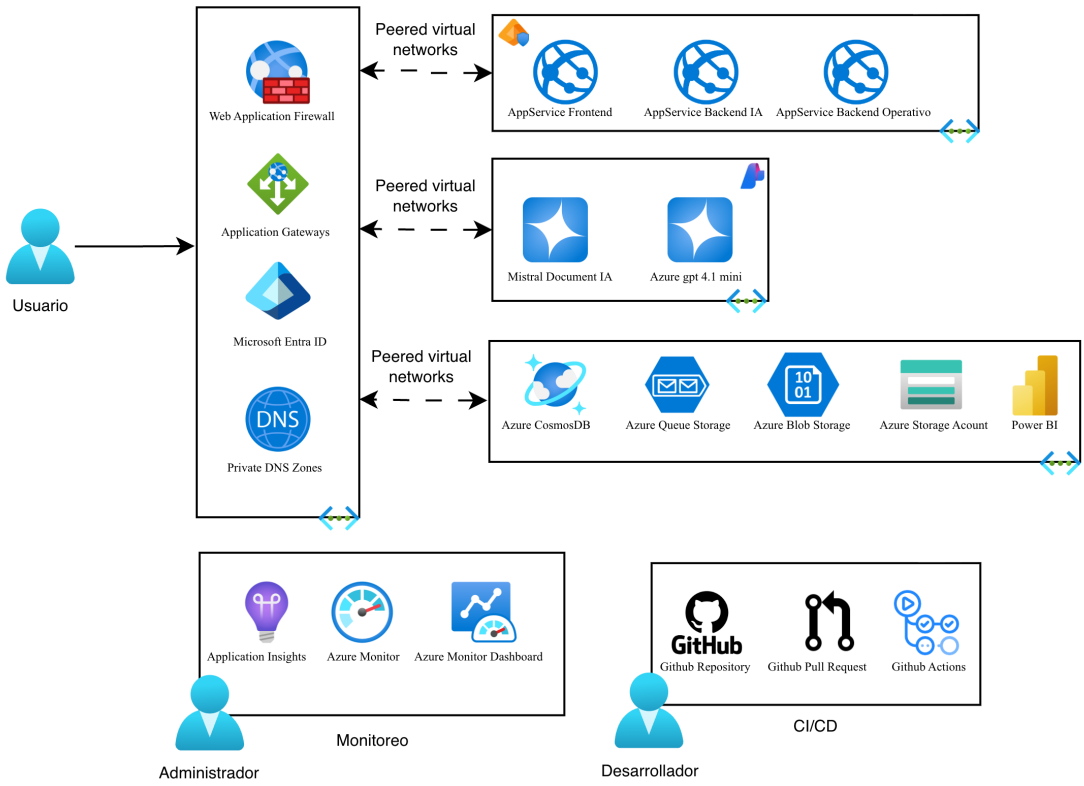

Figura 16: Diagrama De Despliegue Azure - Cambios Aplicados

4.1.3. Nuevo diagrama de contenedores C4

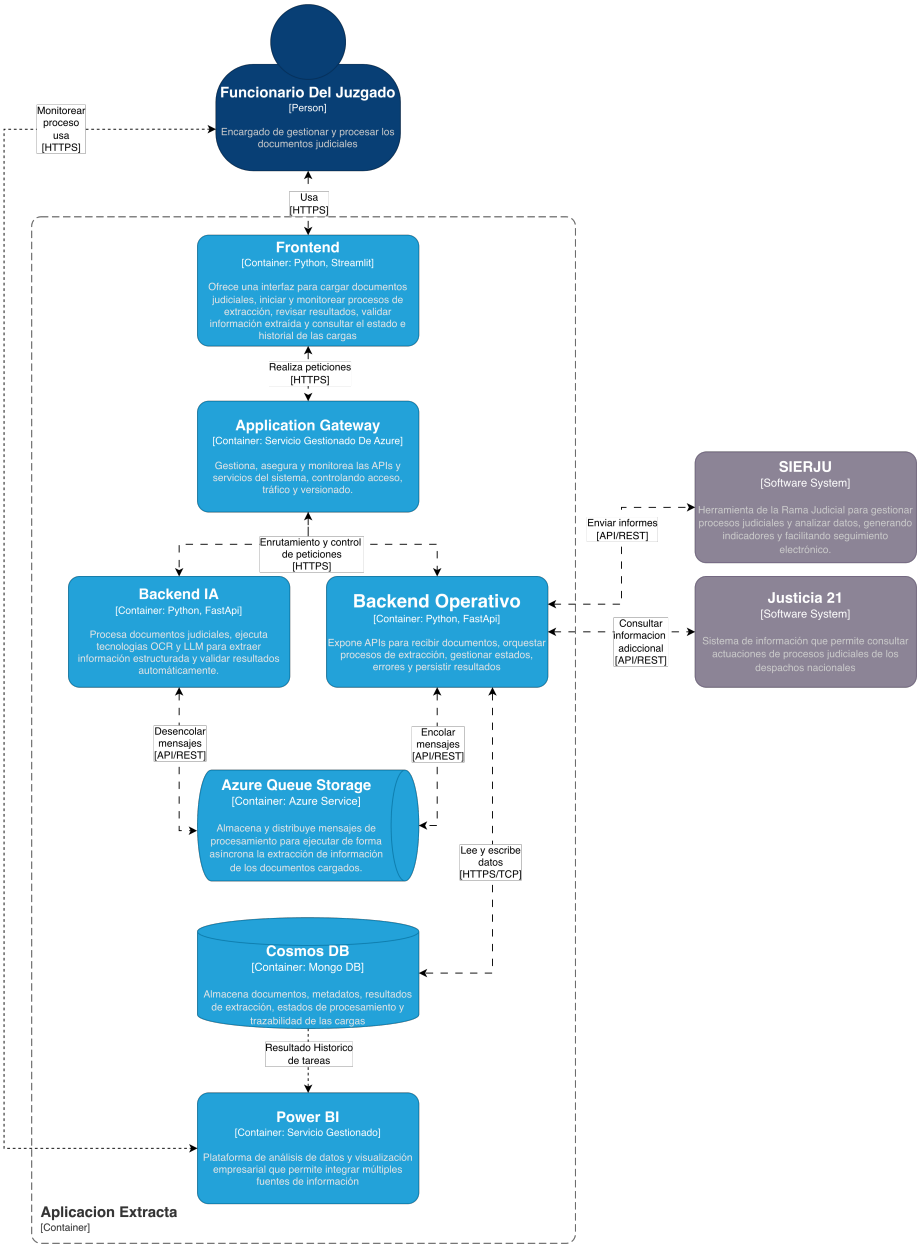

Figura 15: Diagrama de Contenedores C4 – Cambios Aplicados

4.1.5. Conclusiones evaluación a nivel no funcional

Después de realizar la evaluación al diseño de la arquitectura de software a nivel no funcional, se concluye que el diseño es capaz de satisfacer los escenarios de calidad definidos. Los cambios después de la evaluación fueron solo para agregar nuevos elementos:

- Se incorpora Power BI como componente de analítica para consolidar métricas de calidad del proceso de extracción, monitorear el rendimiento del procesamiento documental y analizar la retroalimentación de los funcionarios del juzgado. Esto permite generar indicadores de confiabilidad, identificar patrones de error y soportar la mejora continua de los modelos de IA en el marco del escenario QS-02.

No hubo cambios en la estructura de componentes y relaciones definidas desde el inicio, lo que refleja que el diseño arquitectónico tuvo en cuenta los escenarios de calidad definidos.

4.2. Evaluación de modelos

Para determinar los modelos a utilizar en el proyecto, se realizaron una serie de pruebas utilizando un **modelo de lenguaje como juez (LLM as Judge)**. En nuestro caso, se evaluó el desempeño de dos soluciones OCR y tres modelos LLM, y el juez basó sus evaluaciones en un **dataset de validación**, determinando los niveles de acierto de cada modelo y generando una sere de tablas comparativos. A continuación se muestran los resultados de la evaluación, evidenciando las diferencias en precisión y efectividad de cada uno de los modelos evaluados.

Rendimiento

En el proyecto se evaluaron dos soluciones OCR (Mistral OCR y Google Flash) y tres modelos LLM (GPT 4.1 mini, Mistral Large y GPT 5.0 mini). La tabla 20

muestra los resultados, mostrando la cantidad de aciertos y de fallos procesando 50 archivos.

#	OCR	LLM	Archivos Procesados	Aciertos	Fallos	Porcentaje Aciertos	Porcentaje Fallos
1	Mistral OCR	Gpt 4.1 mini	50	255	44	85 %	15 %
2	Google Flash	Gpt 4.1 mini	50	238	61	80 %	20 %
3	Mistral OCR	Mistral large	50	215	79	73 %	27 %
4	Mistral OCR	Gpt 5.0 mini	50	215	79	79 %	21 %

Tabla 20: Resultados de evaluación de combinaciones OCR y LLM

De acuerdo a nuestra evaluación, la combinación de **Mistral OCR** con **GPT-4.1 mini** presenta el mayor porcentaje de aciertos (un 85 % de aciertos). Se evidencia también que, aunque se use la misma solución OCR, las variaciones en el modelo LLM influyen directamente en el desempeño global del sistema. Además, curiosamente, para el caso de los modelos OpenAI, el modelo más reciente no parece ofrecer un mejor rendimiento qe los modelos anteriores (aquí, GPT4 4.1 mini ofrece mejores resultados que GPT 5.0 mini)

Costos

Nuestra evaluación también nos permitió comparar los costos del funcionamiento. La Tabla 21 muestra un comparativo de los costos considerando los precios durante el segundo semestre de 2025.

Modelo	Función	Tokens Entrada	Tokens Salida	Páginas	Costos de Entrada	Costos de Salida	Costo en Páginas
Mistral	OCR	-	-	15.687	-	-	$12.86 USD
Mistral Large	LLM	732.767 tokens	40.577 tokens	-	$1.47 USD	$0.24 USD	-
Google Flash	OCR	-	-	15.687	-	-	$10.11 USD
Gpt 4.1 mini	LLM	6.59M tokens	806.607 tokens	-	$0.40 USD	$0.55 USD	-
Gpt 5.0 mini	LLM	6.59M tokens	806.607 tokens	-	$0.25 USD	$0.52 USD	-

Tabla 21: Costos de procesamiento por modelo

De acuerdo a nuestra evaluación, el modelo LLM que tiene un menor costo es el GPT-5.0 mini, mientras que la opción de OCR más económica es Google Flash, con respecto al número de páginas procesadas.

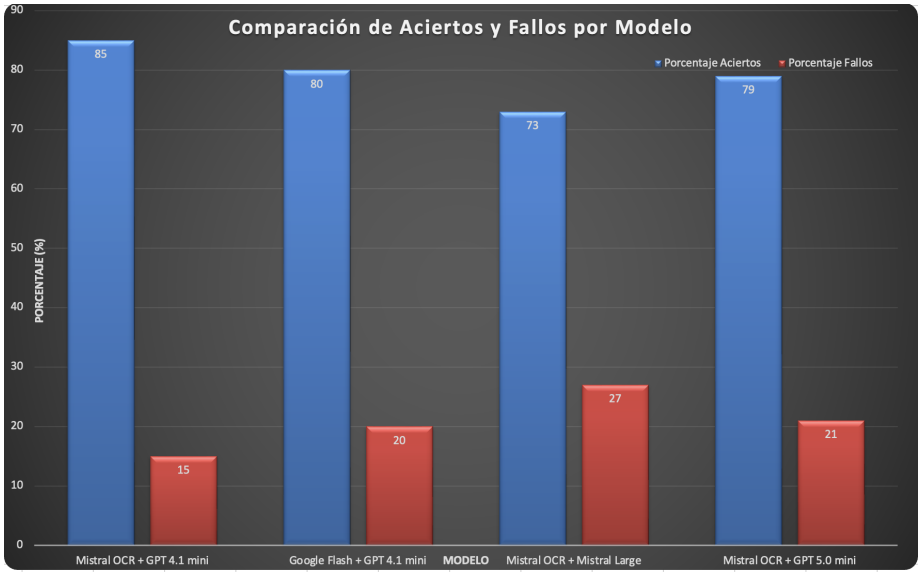

Figura 17: Comparación de porcentajes de aciertos y fallos por modelo

La Figura 17 permite contrastar el desempeño de las distintas combinaciones OCR y LLM evaluadas, evidenciando que la combinación Mistral OCR + GPT 4.1 mini alcanza el mayor porcentaje de aciertos (85 %), mientras que Mistral OCR + Mistral Large presenta el menor nivel de precisión (73 %).

4.3. Discusión

La evaluación no funcional realizada en esta tesis se centró principalmente en el escenario de calidad **QS-05 (Confiabilidad en ambiente de pruebas)**, el cual validó la estabilidad operativa del sistema al procesar un lote controlado de documentos, verificando que no hubiera caídas del servicio y que la tasa de fallos fuera inferior al 5 %. En este sentido, la evaluación evidenció que la arquitectura propuesta soporta el procesamiento masivo en condiciones controladas, cumpliendo con el objetivo de confiabilidad definido para el entorno de pruebas. De manera complementaria, se llevó a cabo una evaluación de modelos bajo el enfoque de *LLM as Judge*, con el fin de comparar el desempeño de diferentes combinaciones OCR y LLM en términos de

precisión y costo, aportando evidencia para la selección tecnológica del sistema.

La evaluación realizada mediante el enfoque de *LLM as Judge* permitió comparar de forma homogénea el desempeño de las distintas combinaciones de tecnologías OCR y modelos de lenguaje, considerando criterios de precisión y costo definidos en el proyecto.

Los resultados muestran que la combinación **Mistral OCR + GPT 4.1 mini** obtiene el mayor nivel de precisión, con un **85 % de aciertos** y el menor porcentaje de fallos (**15 %**). Esta combinación cumple de manera más consistente con el criterio de precisión establecido para el sistema y con el requerimiento de confiabilidad en la extracción de información.

El modelo **GPT 5.0 mini** alcanza un **79 % de aciertos**, superando a **Mistral Large** (**73 %**) y situándose cercano al umbral de precisión definido. En términos de costos, GPT 5.0 mini presenta valores inferiores frente a Mistral Large, lo que lo convierte en una alternativa viable cuando se prioriza el equilibrio entre precisión y costo de operación.

En el caso de **Google Flash OCR** combinado con GPT 4.1 mini, se obtiene un **80 % de aciertos**, inferior a Mistral OCR con el mismo modelo de lenguaje, lo que evidencia que el componente OCR influye en los resultados finales del proceso de extracción.

En síntesis, la confiabilidad del sistema se valida desde dos dimensiones complementarias: por un lado, el escenario QS-05 demuestra la estabilidad operativa del procesamiento masivo en ambiente de pruebas, asegurando continuidad del servicio y control de fallos; por otro, la evaluación mediante *LLM as Judge* permite medir la exactitud de la extracción, evidenciando el desempeño real de las combinaciones OCR y LLM. De esta manera, la confiabilidad no se limita únicamente a la disponibilidad del sistema, sino que integra también la precisión del resultado generado, consolidando una visión integral del atributo de calidad dentro del diseño arquitectónico propuesto.

5. Conclusiones

El proyecto diseñó y evaluó una arquitectura de software orientada a la automatización de la captura y procesamiento de información de procesos judiciales, mediante la integración de tecnologías de Reconocimiento Óptico de Caracteres (OCR) y Modelos de Lenguaje de Gran Escala (LLM). Los resultados obtenidos permiten concluir que la propuesta es técnicamente viable y adecuada para apoyar la generación de información estructurada requerida por el sistema SIERJU.

La aplicación de la metodología Attribute Driven Design (ADD) facilitó la identificación y priorización de los requerimientos arquitecturalmente significativos, permitiendo definir una arquitectura modular, desacoplada y alineada con atributos de calidad como usabilidad, confiabilidad y Seguridad. La arquitectura basada en servicios en la nube demostró ser apropiada para soportar flujos de procesamiento asíncronos y la integración de componentes de inteligencia artificial.

El desarrollo del prototipo validó la correcta interacción entre los contenedores definidos en el modelo C4 y evidenció que la incorporación de LLM para la extracción de información representa una alternativa efectiva frente a los procesos manuales actuales. Asimismo, la evaluación comparativa de modelos, utilizando un enfoque de LLM as Judge, permitió identificar diferencias relevantes en términos de precisión y efectividad, resaltando la importancia de una adecuada selección del modelo para este tipo de aplicaciones.

5.1. Lecciones aprendidas

Una de las principales lecciones aprendidas fue la relevancia de identificar tempranamente los requerimientos arquitecturalmente significativos, ya que estos orientan de manera decisiva las decisiones de diseño. Se evidenció que la variabilidad y falta de estructura de los documentos judiciales constituye un reto significativo para estos sistemas, lo que exige que el sistema cuente con un alto nivel de pruebas para poder afrontar cualquier inconveniente que se pueda presentar en las transacciones.

Adicionalmente, se constató que los modelos de lenguaje requieren un diseño cuidadoso de los mecanismos de evaluación y validación, siendo el enfoque de LLM as

Judge una alternativa útil para la comparación sistemática de resultados. Finalmente, el uso de servicios en la nube complementa el diseño propuesto con el fin de añadir la infraestructura necesaria para que la aplicación pueda funcionar en entornos de producción reales y pueda solucionar los problemas operativos que se tienen en la actualidad en los despachos judiciales de la ciudad de Pereira.

5.2. Trabajo futuros

Como trabajos futuros se propone la ampliación del prototipo hacia un entorno de producción, incorporando mecanismos avanzados de seguridad y cumplimiento normativo. Adicionalmente, se sugiere evaluar el sistema con un conjunto más amplio y diverso de documentos judiciales reales, con el fin de analizar su desempeño frente a distintos formatos y calidades de digitalización.

Igualmente, como parte de la ampliación del prototipo, se plantea la necesidad de realizar diseños y evaluaciones adicionales que permitan abordar todos los escenarios de calidad planteados en el proyecto. En particular, desarrollar el software definitivo considerando no solo la confiabilidad de los modelos sino también considerar los escenario de calidad asociados a usabilidad, confiabilidad y seguridad.[13]

Finalmente, también se plantea como trabajo adicional realizar un análisis detallado de costos y desempeño que permita evaluar la sostenibilidad de la solución en escenarios de uso a gran escala.

[13]Ver los Escenarios de Calidad planteados en la sección 3.2.6

Anexos

A. Gestión de Datos Privados en el Proyecto

Para evitar infringir cualquier norma relacionada con la protección de datos personales y con la gestión y privacidad de la información, el proyecto realizará los experimentos y validaciones utilizando documentos y datos simulados, sin usar documentos reales.

A.1. Aspectos Legales

En Colombia, la Ley 1581 de 2012 reconoce y protege la privacidad de la información personal. Esta ley establece derechos sobre los titulares y obligaciones sobre las entidades que la procesan. Con base en esta ley, las entidades publicas o privadas deben garantizar a las personas el derecho a conocer, actualizar, rectificar y suprimir sus datos personales, así como a oponerse a uso.

Esta ley presta especial atención a los **datos personales**, i.e., toda aquella información asociada a una persona que permite su identificación, tal como su documento de identidad, lugar de nacimiento, estado civil, edad y lugar de residencia; así como información sensible que no debe ser de público conocimiento, tal como su estado de salud, sus características físicas, ideología política o vida sexual, entre otros aspectos. [14]

Cuando un ciudadano realiza algún trámite o proceso legal, el ciudadano acepta el uso de sus datos personales en el marco del trámite en cuestión. Si los datos van a ser usados para alguna actividad o propósito diferente, la persona debe ser informada y debe aceptar su manejo.

Es importante aclarar que los reportes y estadísticas del SIERJU no incluyen datos personales individuales, sino cantidades y sumatorias sobre la totalidad de los procesos. Por lo cual, estos reportes no infringen la ley de protección de datos.

[14] `https://www.minambiente.gov.co/politica-de-proteccion-de-datos-personales/`

A.2. Consultas sobre los Procesos Judiciales

El sistema Justicia XXI Web y el módulo de consultas TYBA permiten la consulta de los procesos que se realizan en los despachos judiciales a nivel nacional, en el marco de la Ley de Transparencia y del Derecho de Acceso a la Información Pública Nacional.

Actualmente, el sistema de consultas de procesos de la rama judicial ofrece tres opciones para la búsqueda de información:

- **Consultar por nombre o razón social**, donde se pueden ver los procesos judiciales relacionados con una persona natural o jurídica. Para hacer la consulta se tienen, como datos obligatorios, el Tipo de Persona y su Nombre y Apellido o Razón Social. Como datos opcionales se puede ingresar información de Departamento, Ciudad, Entidad, Especialidad y Despacho.

- **Consultar por número de radicación**, donde se puede buscar un proceso por el Número de Radicación asignado una vez se recibe el proceso por parte del juzgado. Este número se compone de 23 dígitos. (de Colombia, 2025)

- **Consultar por juez que está llevando el caso**, donde se pueden ver procesos asignados a un juez y se requiere ingresar el nombre del juez.

A.3. Información requerida para los formularios del SIERJU

Para generar los informes del SIERJU, los funcionarios de la rama judicial cargan los datos de cada uno de los procesos, de acuerdo con el manual instructivo para diligenciar los formularios del SIERJU en los juzgados civiles municipales(Rama Judicial de Colombia, 2019)(Tribunal Superior de Cúcuta, 2019).

Los datos que se registran en las plantillas de Excel y en los formularios de carga de información del SIERJU, incluyen:

- **Información del proceso**, tal como fecha de radicación, número de radiación, tipo de proceso, cuantía, estado (vigente o no), etapa procesal, sentencia, fecha de sentencia, motivo de salida y fecha de salida.

- **Información del demandante**, como identificación (CC o NIT), nombre, género, edad y grupo étnico; y
- **Información del demandando**, incluyendo como identificación (CC o NIT), nombre, género, edad y grupo étnico.

Como puede verse, la información ingresada en los formularios incluye datos personales. Sin embargo, esta información puede ingresarse en el SIERJU ya que este hace parte de los procesos de ejecución y seguimiento de los procesos judiciales.

A.4. Documentos y Datos de Prueba para el proyecto

Dado que el objetivo del proyecto busca utilizar tecnologías de Modelos de Lenguaje de Gran Tamaño (LLM) y Reconocimiento Óptico de Caracteres (OCR) para extraer estos datos de imágenes de documentos y archivos de texto, los autores han planteado el uso de documentos simulados que no involucran ninguna persona ni proceso real que se esté trabajando o se haya trabajado en los despachos judiciales, ya que no se cuenta con el acceso y la autorización del manejo de esta información.

Para evitar cualquier posible infracción a las normas de datos personales y privacidad de la información, el proyecto realizará los experimentos y validaciones utilizando documentos y datos simulados, sin usar documentos reales.

NOTA: Los autores realizando este proyecto no tienen relación directa ni hacen parte de la rama judicial o sus diferentes entes de control.

B. Pasos De La Metodología ADD

ADD es un proceso iterativo en donde cada iteración comprende una serie de pasos (Wojcik et al., 2006)(Kazman and Cervantes, 2014):

1. **Revisar las entradas para la iteración**: antes de comenzar cada iteración, los arquitectos debe asegurarse de que los ASR (los *drivers* de arquitectura) estén disponibles y sean correctos. En concreto, deben esta disponibles: el propósito de la iteración de diseño, los requisitos ASR funcionales, los escenarios de atributos de calidad y las restricciones para diseñar el sistema.

2. **Seleccionar el objetivo de la iteración** a través de la selección de drivers: cada iteración se enfoca en lograr un objetivo en particular. Este objetivo implica actualizar el diseño para satisfacer un subconjunto de los ASR. Un objetivo de iteración podría ser, por ejemplo, crear un diseño que permita lograr un escenario (un requerimiento) de confiabilidad en torno a los datos extraídos de archivos PDF y Word no estructurados.

3. **Escoger los elementos del sistema para refinar**: identificar y seleccionar uno o más elementos arquitectónicos del sistema que requieran refinamiento, mediante su descomposición, combinación o mejora, con el fin de satisfacer los *drivers* de diseño y avanzar en la definición detallada de la arquitectura. Por ejemplo, se podrían seleccionar algunos componentes responsables de la captura de datos, para revisar su diseño y ajustarlo para que cumplan el escenario de confiabilidad mencionado arriba.

4. **Elegir uno o más conceptos de diseño que satisfagan los drivers**: seleccionar tecnologías, estilos, patrones y tácticas de arquitectura que pueden usarse para lograr el objetivo de la iteración y luego tomar decisiones sobre cuáles de estas alternativas podría utilizarse para cumplir con los *drivers* seleccionados. Por ejemplo, se podrían considerar una o más librerías disponibles en Internet, tales como Docling, para mejorar la confiabilidad de la captura de datos.

5. **Crear instancias de elementos arquitectónicos, asignar responsabilidades y definir interfaces**: concretar las ideas y decisiones sobre la arquitectura mediante la creación de elementos de arquitectura, la definición de sus funciones, la asignación de responsabilidades y la definición de sus interfaces. Por ejemplo, se podría incluir un nuevo elemento o reemplazar alguno por un nuevo componente de software que use la librería seleccionada.

6. **Diagramar las vistas y registrar las decisiones de diseño**: documentar los cambios en el diseño, los nuevos elementos y las decisiones de arquitectura que se han realizado en los pasos anteriores. En particular, se registrarán las decisiones importantes tomadas en la iteración, así como las razones (la justificación) que motivaron estas decisiones. Esta información es importante para facilitar el análisis y la comprensión posteriores del proceso de diseño de y las decisiones que se se tomaron.

7. **Realizar un análisis del diseño actual y revisar el logro del objetivo de la iteración y del propósito del diseño**: Revisar, posiblemente con el apoyo de un grupo de arquitectos, si el diseño resultante de la iteración permite lograr el propósito establecido para el diseño. Esto significa considerar, en este punto, si se realizaron las suficientes iteraciones de diseño o si se necesitan iteraciones adicionales.

8. **Iterar si es necesario**: idealmente, el diseño se obtiene a partir de varias iteraciones, especialmente para los *drivers* más importantes. Sin embargo, por limitaciones de tiempo o recursos, esto no siempre es posible. Se priorizarán los *drivers* críticos y se evaluara si el diseño es "lo suficientemente bueno". En los métodos de desarrollo iterativo, cada iteración debe enfocarse primero en los *drivers* de mayor prioridad y después en los de menor prioridad o los nuevos que surjan.

C. Repositorios del Proyecto

Con el fin de garantizar la trazabilidad, reproducibilidad y transparencia del desarrollo realizado, el proyecto cuenta con repositorios de código fuente alojados en la plataforma GitHub. Estos repositorios contienen la implementación de los distintos componentes del sistema, de acuerdo con la arquitectura propuesta y las iteraciones definidas en la metodología ADD.

C.1. Repositorio Backend Operativo

Este repositorio contiene la implementación del backend operativo del sistema, encargado de la recepción de documentos, la orquestación del flujo de procesamiento, la gestión de estados y errores, y la persistencia de los resultados.

- **Repositorio:** Proyecto_Back
- **URL:** `https://github.com/valenciacamilo12/Proyecto_Back`
- **Descripción:** Servicio backend desarrollado en Python (FastAPI), responsable de exponer las APIs para la carga de documentos judiciales, la gestión de solicitudes de procesamiento y la comunicación con los servicios de almacenamiento y mensajería.

C.2. Repositorio Backend de Inteligencia Artificial

Este repositorio agrupa los componentes relacionados con el procesamiento mediante tecnologías de OCR y Modelos de Lenguaje de Gran Tamaño (LLM), así como la lógica de extracción de información y validación de resultados.

- **Repositorio:** Proyecto_BackIA
- **URL:** `https://github.com/valenciacamilo12/Proyecto_BackIA`

- **Descripción:** Backend especializado en el procesamiento de documentos judiciales, encargado de ejecutar los modelos OCR y LLM, estructurar la información extraída y aplicar los mecanismos de evaluación mediante el enfoque de *LLM as Judge*.

C.3. Repositorio Frontend del Sistema

Este repositorio contiene la implementación del frontend del sistema, el cual permite a los funcionarios interactuar con la aplicación para cargar documentos, iniciar procesos de extracción y visualizar los resultados.

- **Repositorio:** Proyecto_front
- **URL:** `https://github.com/daniidega/Proyecto_front`
- **Descripción:** Interfaz web desarrollada en Python utilizando Streamlit, que actúa como punto de entrada al sistema y permite el monitoreo del estado y los resultados de las cargas de documentos.

Referencias

AskYourPDF (2025). AskYourPDF: The Best PDF AI Chat App. https://askyourpdf.com/es.

AWS (2024a). ¿Qué es el OCR? https://aws.amazon.com/es/what-is/ocr/.

AWS (2024b). ¿Qué es RAG? https://aws.amazon.com/es/what-is/retrieval-augmented-generation/.

AWS (2024c). ¿Qué es un LLM? https://aws.amazon.com/es/what-is/large-language-model/.

AWS (2024d). ¿Qué son los modelos básicos en la IA generativa? https://aws.amazon.com/what-is/foundation-models/.

Bass, L., Clements, P., and Kazman, R. (2021). *Software Architecture in Practice, 4th Edition*. Addison-Wesley Professional.

de Colombia, R. J. (2025). Manual de radicación de procesos. https://consultaprocesos.ramajudicial.gov.co/manual/numRadicacion.html.

Docling (2024). Github - docling-project/docling. https://github.com/docling-project/docling.

El-Tiempo (2023). ¿Qué es TYBA y cómo consultar procesos judiciales en la plataforma? https://www.eltiempo.com/justicia/servicios/que-es-tyba-y-como-consultar-procesos-judiciales-en-la-plataforma-773129.

Función-Pública (2025). Manual de la estructura del estado: Rama judicial. Accedido: 26 de abril de 2025. https://www.funcionpublica.gov.co/eva/gestornormativo/manual-estado/pdf/26_Rama_Judicial.pdf.

Hugging-Face (2024). What is Document Question Answering? https://huggingface.co/tasks/document-question-answering.

IBM (2023). ¿Qué es LangChain? https://www.ibm.com/es-es/think/topics/langchain.

Kazman, R. and Cervantes, H. (2014). *Designing Software Architectures: A Practical Approach, 2nd Edition*. O'Reilly.

Mayer, A. (2023). Understanding Context in AI. https://www.linkedin.com/pulse/understanding-context-ai-andrew-mayer.

Ministerio-Justicia (2025). Introducción a la rama judicial. https://www.minjusticia.gov.co/transparencia/Paginas/SEJ-Rama-Judicial-Introduccion.aspx.

Pure-Storage (2025). ¿Qué es la IA generativa? https://www.purestorage.com/es/knowledge/what-is-generative-ai.html.

Rama-Judicial (2025). Informe de transformación digital de la rama judicial. Accedido: 26 de abril de 2025. https://www.ramajudicial.gov.co/documents/10228/35666503/INFORME+TRANSFORMACIN+DIGITAL+RAMA+JUDICIAL..PDF/53701101-e30c-466b-841a-98faf9fce8e9.

Rama Judicial de Colombia (2019). Manual instructivo civil municipal. Accedido el 26 de abril de 2025.

Richards, M. and Ford, N. (2025). *Fundamentals Of Software Architecture, 2nd Edition*. O'Reilly.

Tribunal Superior de Cúcuta (2019). Formularios definitivos del sierju. https://tribunalsuperiordecucuta.gov.co/2019/06/12/formularios-definitivos-del-sierju/. Accedido el 26 de abril de 2025.

Wojcik, R., Bachmann, F., Bass, L., Clements, P., Merson, P., Nord, R., and Wood, W. (2006). *Attribute-Driven Design (ADD), Version 2.0*. Software Engineering Institute.

Wood, W. G. (2007). *A Practical Example of Applying Attribute-Driven Design (ADD), Version 2.0*. Software Engineering Institute.

Xataka (2023). ¿Qué es un prompt y por qué son tan importantes para usar la inteligencia artificial? https://www.xataka.com/basics/que-prompt-que-importantes-para-usar-inteligencia-artificial.