Desarrollo de una plataforma cognitiva para agilizar los procesos de gestión documental en el área de logística.

Eduardo Velasquez Velez

Trabajo de grado presentado para optar al título de Ingeniero de Sistemas

Asesores
Carlos Mauricio Duque Restrepo, Ingeniero de Sistemas
Martín Elías Quintero, Ingeniero de Sistemas

Universidad de Antioquia
Facultad de Ingeniería
Ingeniería de Sistemas
Medellín, Antioquia, Colombia
2022

Cita	Velasquez Velez [1]
Referencia Estilo IEEE (2020)	[1] E. Velasquez Velez, “Desarrollo de una plataforma cognitiva para agilizar los procesos de gestión documental en el área de logística.”, Trabajo de grado profesional, Ingeniería de Sistemas, Universidad de Antioquia, Medellín, Antioquia, Colombia, 2022.

TABLA DE CONTENIDO

LISTA DE TABLAS

LISTA DE FIGURAS

SIGLAS, ACRÓNIMOS Y ABREVIATURAS

IEEE	Institute of Electrical and Electronics Engineers
UdeA	Universidad de Antioquia
API	Application Programming Interface

RESUMEN

Uno de los grandes retos dentro del sector de logística de transporte es la denominada gestión de documentos, esta representa una carga que genera costos y retrasos para el negocio ya que con las tecnologías y arquitecturas tradicionales es una tarea tediosa que requiere un enorme esfuerzo y personal. Sin embargo en los últimos años los grandes volúmenes de datos y estrategias basadas en inteligencia artificial han adquirido un papel fundamental en el desarrollo de sistemas que permitan automatizar procesos y mejorar los tiempos de respuesta dentro de este sector.

Si bien la inteligencia artificial y el diseño de sistemas cognitivos han demostrado ser de gran ayuda para solventar esta problemática, también es cierto que existen retos intrínsecos a estos sistemas, tales como la optimización en las predicciones de los modelos y la escalabilidad masiva de los mismos.

En este trabajo se presenta el estudio de una arquitectura dirigida por eventos, esto con el fin de mejorar los tiempos de respuesta, consistencia y mantenibilidad del sistema. Dentro del alcance del proyecto se propone la búsqueda de diferentes tecnologías existentes para realizar el nuevo diseño arquitectónico, así como su implementación y evaluación de rendimiento para determinar su eficacia.

Palabras clave — **inteligencia artificial, big data, sistemas cognitivos, arquitectura dirigida por eventos.**

ABSTRACT

One of the major challenges within the transport logistics sector is the so-called document management, which represents a burden that generates costs and delays for the business, as with traditional technologies and architectures it is a tedious task that requires enormous effort and personnel. However, in recent years, big data and artificial intelligence based strategies have taken on a fundamental role in the development of systems that automate processes and improve response times within this sector.

While artificial intelligence and cognitive systems design have proven to be of great help in solving this problem, it is also true that there are inherent challenges to these systems, such as optimizing model predictions and their scalability on a massive scale.

This work presents the study of an event driven architecture, with the intention of improving the response times, consistency and maintainability of the system. Within the scope of the project, it is proposed to search for different existing technologies to carry out the new architectural design, as well as its implementation and performance evaluation to determine its effectiveness.

Keywords — **artificial intelligence, big data, cognitive systems, event driven architecture.**

I. INTRODUCCIÓN

Los grandes volúmenes de datos (big data), acompañados de estrategias basadas en la inteligencia artificial, han adquirido un papel fundamental en la toma de decisiones empresariales en múltiples industrias. Uno de ellos es el sector logístico, donde uno de los mayores retos a los que hay que enfrentarse es la denominada gestión de documentos dado el masivo volumen de estos que hay que procesar, analizar y clasificar para agilizar los procesos y racionalizar las estrategias de negocio. Procesar toda esta información supone un enorme esfuerzo, pero resulta una tarea tediosa que requiere mucho personal y es difícil de automatizar con las tecnologías y metodologías tradicionales. Aquí es donde la inteligencia artificial y el diseño de sistemas cognitivos demuestran ser de gran valor en toda la cadena de procesos. Sin embargo, desde la ingeniería de inteligencia artificial surgen retos intrínsecos de estos sistemas como la optimización en las predicciones de los modelos y la escalabilidad masiva de los mismos.

Al ser el sector logístico una industria muy demandada, las respuestas deben darse en el menor tiempo posible permitiendo extender los modelos de inteligencia artificial como componentes dentro de una arquitectura de software que permita, según los diseños arquitectónicos, el correcto uso del sistema.

Existen algunas soluciones prácticas que pueden resolver directamente los retos anteriores, como el uso de equilibradores de carga o el aumento constante de la capacidad de cálculo de los modelos, pero estas soluciones lineales acaban siendo económicamente inviables.

Este trabajo estudia la aplicación de una arquitectura dirigida por eventos para un sistema de gestión de documentos cognitivos. Se observa cómo un diseño arquitectónico pensado para este problema es más eficiente que las versiones tradicionales. En este sentido, se ha evidenciado que el impacto en el diseño es tan importante que mejora no sólo los tiempos de respuesta, sino también la consistencia y la mantenibilidad del sistema.

II. OBJETIVOS

A. Objetivo general

Implementar una arquitectura dirigida por eventos con el uso de procesos asíncronos en el Servicio cognitivo que permite agilizar los procesos de gestión documental.

B. Objetivos específicos

- Investigar cómo funciona Celery en la implementación de colas de tareas.
- Identificar diferentes formas de implementar una arquitectura dirigida por eventos.
- Seleccionar el método de implementación que permita la mayor modularización y escalado del Servicio.
- Diseñar la arquitectura del Servicio.
- Evaluar que la arquitectura diseñada cumpla el objetivo de mejorar la Servicio en la velocidad del procesado de documentos y en el escalado de los módulos críticos.
- Revisar el Servicio cognitivo para su refactorización.
- Implementar la arquitectura en el Servicio cognitivo.
- Evaluar el desempeño del módulo según su desempeño anterior.

III. MARCO TEÓRICO

La inteligencia artificial es un campo de las ciencias de la computación con un gran foco de interés en la actualidad, que consiste en la generación de máquinas inteligentes capaces de operar con la menor intervención humana. En el desarrollo de la inteligencia artificial se encuentran distintas técnicas y campos de aplicación tales como: el *Machine Learning*, la lógica difusa y el procesamiento del lenguaje natural.[1]

Una de las más mencionadas es el *Machine Learning*, definida como la rama de la inteligencia artificial que otorga a las máquinas la capacidad de aprender sin ser expresamente programadas. Esta rama se centra especialmente en la creación de programas capaces de generar automáticamente modelos de clasificación, pronóstico y predicción como otros ejemplos posibles.[2]

Una vez establecido el modelo cognitivo, se envuelve en una capa de software que sirve de interfaz del sistema. Así, el uso del modelo se vuelve transparente para los usuarios, debido a que el usuario solo debe saber de parámetros entrada y la respuesta del sistema, por esa razón se hizo común el uso de APIs para establecer esta comunicación, ya que permite ocultar complejidad a los desarrolladores, extender las aplicaciones, organizar el código y reusar componentes [3].

Las APIs pueden ser desarrolladas de distintas maneras, si no se requiere guardar la sesión de las peticiones, se puede usar el paradigma REST. REST (*Representational State Transfer*) es un conjunto de principios, propiedades y restricciones orientados hacia los recursos; su uso fue potenciado desde la publicación del trabajo de doctorado de Roy Thomas Fielding [4]. Para la comunicación REST usa el protocolo HTTP (Hypertext Transfer Protocol), que es un protocolo a nivel de aplicación para hipermedia distribuida colaborativa en sistemas de información [5].

Además del modelo cognitivo, se crearán múltiples procesos asíncronos para distribuir de manera eficiente el trabajo entre ellos se usará una cola de tareas (*Task queues*) donde cada proceso asíncrono será un nodo. La entrada de una cola de tareas es una unidad de trabajo denominada tarea. Los nodos se dedicarán a monitorear constantemente las colas de tareas para realizar nuevos trabajos. Para implementar la cola de tareas se usará la librería Celery que es un sistema distribuido simple, flexible y confiable para procesar grandes cantidades de mensajes y proporciona las herramientas necesarias para mantener dicho sistema [6].

Para el uso del sistema de colas es necesario otras dos herramientas The RabbitMQ y Redis, la primera siendo un *Message Broker* que permite al servicio comunicarse entre sí e intercambiar información. El *Message Broker* desempeña esta función traduciendo mensajes entre protocolos de mensajería formal. Esto permite que los servicios interdependientes "hablen" entre sí directamente, incluso si fueron escritos en diferentes idiomas o implementados en diferentes plataformas [7].

La segunda siendo un almacén de estructura de datos en memoria de código abierto, que se utiliza como base de datos, caché y agente de mensajes. Redis proporciona estructuras de datos como cadenas, hashes, listas, conjuntos, conjuntos ordenados con consultas de rango, mapas de bits, hiperloglogs, índices geoespaciales y flujos [8]. En este proyecto se usará como almacenamiento de los estados de las tareas, así se podrá tener un historial de las tareas realizadas y sus respectivos estados.

La sinergia entre las tecnologías es un requisito indispensable a la hora de escoger el stack tecnológico debido a que estas serán la base del sistema y es importante que no tengan ninguna incompatibilidad o mal funcionamiento. Celery al ser compatible con múltiples tecnologías es perfecto para ser la base sobre el cual se construye el stack tecnológico, teniendo una amplia gama de motores de base de datos donde escoger para el stack se escogió Redis, debido a que es un motor ligero y rápido capaz de guardar múltiples estructuras de datos. Debido a la naturaleza del sistema, es fundamental asegurar la entrega de todas las tareas generadas, por esto se escogió The RabbitMQ ya que asegura la entrega de la totalidad de las tareas a Celery y otorga una implementación sencilla con todo el stack del proyecto.

IV. METODOLOGÍA

En el proyecto se implementó un modelo de casada dividido en 3 fases como se muestra en la Fig. 1.

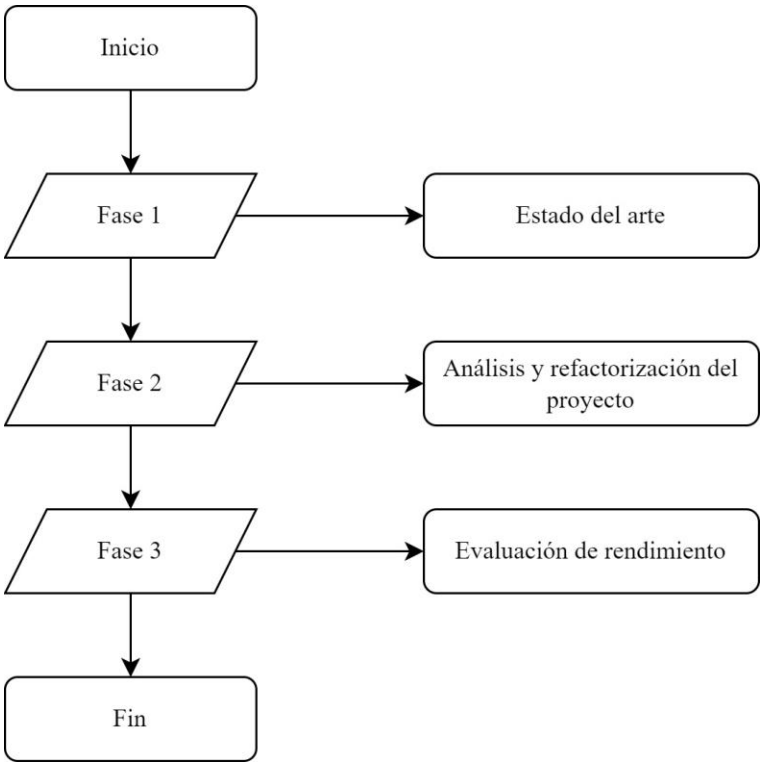

Fig. 1. Fases de la metodología

Fase 1: Estado del arte

Para cualquier proyecto es importante comenzar con una profunda recolección de información, de manera que primero se establece el estado del arte, donde se reunió toda la información y se determinó las tecnologías para la realización del proyecto. En esta recolección de información encontramos varias tecnologías que podría resolver los requerimientos del proyecto, Al ser un proyecto complejo y haber comenzado ya su desarrollo, optamos por tecnologías sencillas de implementar pero robustas.

Las tecnologías escogidas para este proyecto fueron: Celery, Redis y The RabbitMQ para la arquitectura dirigida por eventos, FastAPI y Python para la creación/adaptación de la API.

Fase 2: Análisis y refactorización del proyecto

Antes de idear la arquitectura con las nuevas tecnologías se analizó el proyecto, a partir del análisis se llegó a la conclusión que lo óptimo de implementar en este proyecto es un sistema de colas de tareas. Donde cada trabajador asíncrono se encargará de un paso del proceso. de esta manera se puede crear réplicas de los trabajadores asíncronos que lo requieran sin tener que escalar toda la aplicación en conjunto.

Luego de crear la arquitectura se realizó un plan de migración del proyecto a la nueva arquitectura, el cual consistió en separar el proyecto en diferentes trabajadores asíncronos y al mismo tiempo se añadieron funcionalidades necesarias para el funcionamiento en la nueva arquitectura y optimizar el flujo del proceso.

Fase 3: Evaluación de rendimiento

Luego de tener una versión estable del proyecto con la nueva arquitectura se le realizó múltiples pruebas de flujo completo para probar la estabilidad y rendimiento de este. Posteriormente se compararon los resultados con el proyecto anterior para comprobar que la aplicación de la arquitectura generó resultados positivos.

Las 3 fases de la metodología se realizaron a lo largo de 20 semanas de desarrollo, en la TABLA I se muestran la duración en semanas de las actividades realizadas en este proyecto.

TABLA I
CRONOGRAMA DE ACTIVIDADES

Actividad / Semana	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Fase 1																				
Recolección de información	x																			
Selección de tecnologías		x																		
Fase 2																				
Verificar arquitectura actual del sistema			x																	
Analizar el módulo cognitivo				x																
Creación de arquitectura orientada a eventos					x	x	x													
Separa en módulos más pequeños la API cognitiva siguiendo la arquitectura creada.								x	x	x	x	x								
Creación de nuevas funcionalidades													x	x						
Refactorización del código															x	x	x	x		
Fase 3																				
Evaluar el rendimiento del sistema resultante																			x	x

V RESULTADOS

Al final de este proyecto, obtuvimos una aplicación que implementa la arquitectura dirigida por eventos creada, además segmenta el procesamiento de los documentos en 4 pasos llamados: Pre Procesamiento, Procesamiento de tablas, Reconocimiento de entidades y Post procesamiento. El flujo del sistema es el siguiente:

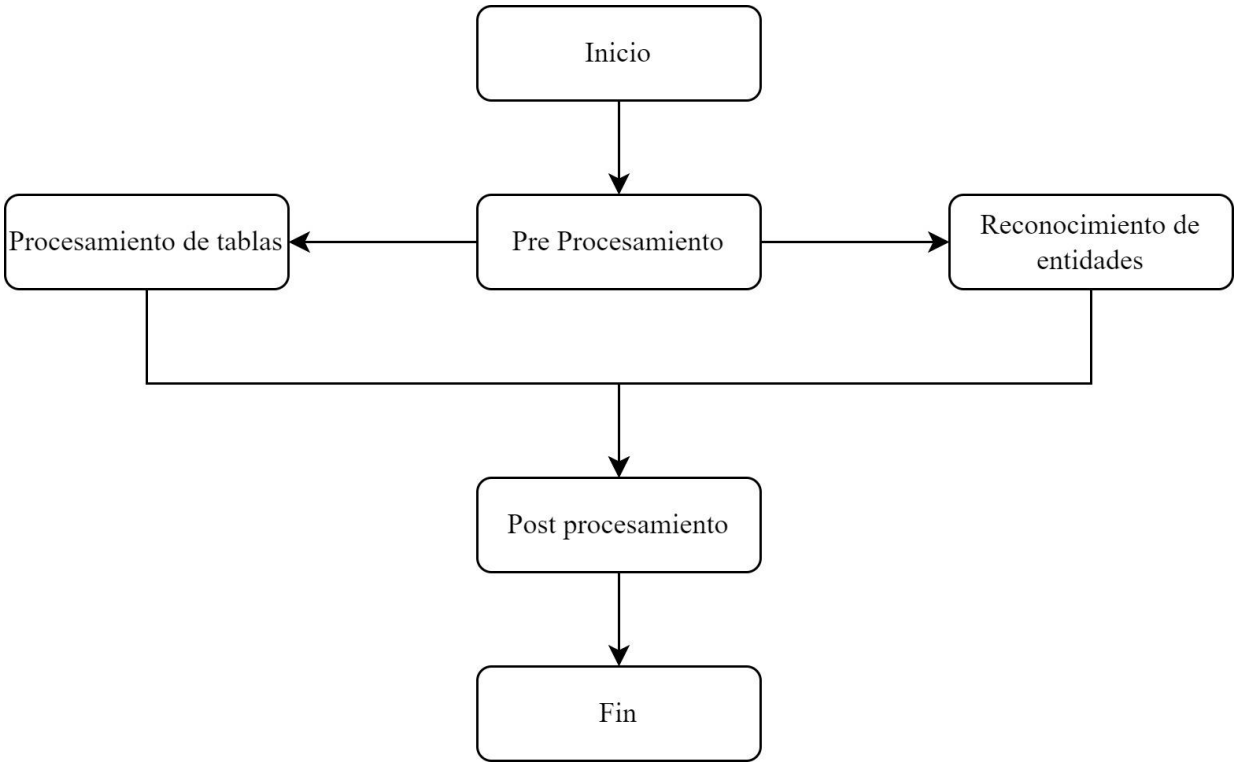

Fig. 2. Flujo del sistema

En la Fig. 2, podemos ver como los pasos de Procesamiento de tablas y Reconocimiento de entidades se ejecutan de manera simultánea mientras que el paso de Post procesamiento se ejecuta solo cuando dos ya mencionado hayan terminado.

Como se puede observar en la TABLA II, la mejora que ofrece la arquitectura dirigida por eventos cuando se procesa un solo documento no es muy importante, esto se debe a que para el funcionamiento de toda la arquitectura se realizan muchas operaciones adicionales como inserciones y lectura de base de datos.

Para documentos con mayor cantidad de párrafos y tablas. La mejora es más pronunciada debido a que los pasos de Procesamiento de tablas y Reconocimiento de entidades son los que más

carga reciben. En la arquitectura anterior se realizaba todo el proceso de manera secuencial mientras que en la nueva arquitectura estos pasos se realizan de forma paralela reduciendo el tiempo de procesamiento del documento.

Por otro lado, en la TABLA III se puede apreciar que la mejora en el tiempo de procesamiento es muy significativa, esto se debe a que la nueva arquitectura procesa los documentos por segmentos, es decir, cuando se termina de procesar alguno de los pasos, inmediatamente obtiene el siguiente documento en la cola para ser procesado y así sucesiva hasta que no haya más documentos en la cola. Esto sucede en todos los pasos del proceso logrando que la aplicación nunca tenga tiempos de inactividad. En cambio la arquitectura anterior solo comenzaba un nuevo documento cuando completaba todo proceso, dando como resultado tiempo en el cual la aplicación tenía inactividad en alguno de sus pasos.

TABLA II
PROCESAMIENTO DOCUMENTO ÚNICO

Documento	Tamaño del Documento (KB)	Tiempo arquitectura anterior	Tiempo arquitectura dirigida por eventos	Mejora
Documento #1	542	0:06:49	0:05:42	0:01:07
Documento #2	466	0:03:56	0:03:53	0:00:03
Documento #3	276	0:02:34	0:02:27	0:00:07
Documento #4	1052	0:15:20	0:13:07	0:02:13

TABLA III
PROCESAMIENTO DE DOCUMENTOS SIMULTÁNEOS

Cantidad de Documentos simultáneos	Tiempo arquitectura anterior	Tiempo arquitectura dirigida por eventos	Mejora
3	0:18:46	0:12:15	0:06:31
5	0:26:23	0:14:02	0:12:21
7	1:09:31	0:28:25	0:41:06
10	0:51:07	0:30:31	0:20:36

VI. CONCLUSIONES

- Se logró una mejora sustancial en los tiempos de respuesta del sistema al procesar simultáneamente diferentes documentos, esto fue posible gracias a que se implementó una arquitectura orientada a eventos que utiliza un sistema de colas de tareas para orquestar el trabajo del sistema, de tal manera que las tareas se ejecutan al instante que el trabajador esté disponible. Añadido a esto, por la segmentación del flujo del sistema la mantenibilidad de este mejoró, permitiendo observar de manera precisa el flujo de información, como esta se comporta en el transcurso del proceso y de esta manera conocer dónde y porqué ocurre las fallas en el sistema.
- La implementación de este tipo de arquitectura suele ser significativamente más compleja que una arquitectura tradicional debido al sistema de comunicación entre procesos, la cola de tareas, base de datos y coste tanto computacional como económico para el funcionamiento de toda la arquitectura, añadido a esto, las arquitecturas que manejan procesos asíncronos y/o simultáneos presentan el problema llamado condición de carrera que se genera en el instante que dos procesos intentan usar el mismo recurso al mismo tiempo.
- Un análisis apresurado ocasiona que la planeación y diseño de sistemas no sea el óptimo para el problema, es idóneo un análisis profundo al problema para observar la mayor cantidad de detalles, de esta manera crea el mejor plan de desarrollo para el proyecto. Así mismo se evita una reestructuración del proyecto que puede tomar una gran cantidad de tiempo y recursos.

REFERENCIAS

[1] APD, “Métodos y técnicas de inteligencia artificial: ¿cuáles son y para qué se usan?,” 07 01 2021. [En línea]. Disponible en: https://www.apd.es/tecnicas-de-la-inteligencia-artificial-cuales-son-y-para-que-se-utilizan.

[2] BBVA NOTICIAS, “'Machine learning': ¿qué es y cómo funciona?,” 08 11 2019. [En línea]. Disponible en: https://www.bbva.com/es/machine-learning-que-es-y-como-funciona.

[3] J. Freeman, “What is an API? Application programming interfaces explained,” 08 08 2019. [En línea]. Disponible en: https://www.infoworld.com/article/3269878/what-is-an-api-application-programming-interfaces-explain.

[4] R. T. Fielding, Architectural styles and the design of network-based software architectures [tesis de doctorado]. Univ. California, Irvine, CA, USA, 2000.

[5] W3.org., “Hypertext Transfer Protocol -- HTTP/1.1.,” 1999 [En línea]. Disponible en: https://www.w3.org/Protocols/HTTP/1.1/rfc2616.pdf

[6] Docs.celeryproject.org. “Introduction to Celery — Celery documentation.,” 2021. [En línea]. Disponible en: https://docs.celeryproject.org/en/master/getting-started/introduction.html

[7] Ibm.com, “¿Qué son los Message Brokers?,” 2021. [En línea]. Disponible en: https://www.ibm.com/co-es/cloud/learn/message-brokers.

[8] Redis, “Redis.,” 2021. [En línea]. Disponible en: https://redis.io/