Sistema de Clasificación de Imágenes PDF con IA

Solución Técnica Detallada para la Tarea 2 del GeoAI Engineer Test

Implementación con FastAPI, PyMuPDF e Integración de IA (OpenAI GPT-4o)

Desarrollado por: José Armando Son Rojas

29/05/2025

Resumen

Resumen Ejecutivo: Este documento presenta una descripción técnica exhaustiva de la solución implementada para la Tarea 2 del GeoAI Engineer Test. El objetivo principal es la construcción de un proceso automatizado para la extracción de todas las imágenes de un documento PDF y su posterior clasificación en categorías específicas: Mapa, Tabla o Imagen (Picture). Se detalla la arquitectura del sistema, el diseño de sus componentes, el flujo de procesamiento de datos y cómo se abordan los requerimientos de la tarea. Se enfatiza la claridad, modularidad y efectividad del enfoque, que se basa en un servidor API con FastAPI, la librería PyMuPDF para la extracción de imágenes, y la integración con la API de OpenAI (específicamente el modelo GPT-4o o similar) para la clasificación inteligente de las imágenes. El documento también cubre el almacenamiento de los resultados en una base de datos SQLite y la persistencia de las imágenes clasificadas en el sistema de archivos.

Índice

1. Introducción y Visión General del Sistema

El presente documento técnico describe la concepción, diseño e implementación de un sistema automatizado para la extracción y clasificación de imágenes de documentos PDF, desarrollado como respuesta a la Tarea 2 del GeoAI Engineer Test. El objetivo principal es construir un proceso robusto y eficiente capaz de procesar un archivo PDF, identificar y extraer todas las imágenes contenidas en él, y luego utilizar un modelo de inteligencia artificial para clasificar cada imagen en una de tres categorías predefinidas.

1.1. Planteamiento del Problema (Tarea 2)

La Tarea 2 del test técnico plantea el desafío de utilizar herramientas de IA para extraer y categorizar imágenes de un documento. Los puntos clave del problema, que esta solución busca abordar integralmente, son:

- **Documento Específico:** Utilizar el informe proporcionado en `https://d1vqu5ynjuwmto.cloudfront.net/assets/files/10526/q4_2023_txg_ep.pdf` como base, aunque la solución está diseñada para ser genérica.
- **Extracción de Imágenes:** Extraer todas las imágenes del archivo PDF.
- **Clasificación de Imágenes:** Clasificar cada imagen extraída en una de las siguientes categorías:
  - **Map**: Mapas, diagramas geográficos, planos.
  - **Table**: Tablas, gráficos con datos estructurados.
  - **Picture**: Fotografías, ilustraciones, imágenes generales, diagramas complejos no clasificables como mapa o tabla.
- **Herramientas de IA:** Utilizar cualquier modelo de IA o enfoque de clasificación de imágenes considerado apropiado (e.g., OpenAI, Gemini, Vision Transformers, etc.). Esta solución emplea la API de OpenAI.

1.2. Arquitectura de la Solución Propuesta

Para abordar estos requerimientos de manera efectiva, se ha optado por una arquitectura de microservicio con una API RESTful desarrollada con FastAPI. La lógica de extracción de imágenes se maneja mediante la librería PyMuPDF (fitz), mientras que la clasificación inteligente de imágenes se delega a la API de OpenAI (modelo GPT-4o). Los metadatos de los documentos y las clasificaciones de imágenes se almacenan en una base de datos SQLite, y las imágenes extraídas se guardan en el sistema de archivos local, organizadas por su categoría. El procesamiento de los PDF se realiza de forma asíncrona para no bloquear las respuestas de la API.

1.3. Objetivos Clave de la Solución para la Tarea 2

La solución se enfoca en cumplir con los puntos de la Tarea 2, prestando especial atención a los siguientes aspectos:

- **Precisión en la Extracción:** Asegurar que la mayor cantidad posible de imágenes relevantes sean extraídas del PDF, aplicando filtros para descartar elementos gráficos muy pequeños o irrelevantes.
- **Exactitud en la Clasificación:** Lograr una alta precisión en la clasificación de las imágenes utilizando un modelo de IA avanzado y un diseño de prompt efectivo.

- **Robustez y Manejo de Errores:** Diseñar un sistema capaz de manejar diferentes tipos de PDF y posibles fallos durante la extracción o clasificación (e.g., errores de API, archivos corruptos).
- **Persistencia y Acceso a Resultados:** Almacenar de forma estructurada los resultados de la extracción y clasificación para su posterior consulta y análisis.
- **Modularidad y Escalabilidad:** Organizar el código en componentes bien definidos, facilitando su mantenimiento y posible extensión futura.

2. Arquitectura del Sistema Detallada

La arquitectura del sistema se fundamenta en principios de diseño modular y separación de responsabilidades, utilizando tecnologías modernas para el desarrollo de API y el procesamiento de datos.

2.1. Modelo de Arquitectura y Flujo de Procesamiento

El sistema sigue un flujo de procesamiento bien definido. A continuación, se presenta una visualización conceptual del flujo, seguida de la descripción detallada de los componentes y el stack tecnológico.

Flujo Conceptual del Procesamiento

1. **Carga PDF:** Usuario envía PDF vía API FastAPI.
2. **Guardado y Cola:** PDF almacenado; tarea creada en DB (‘queued‘).
3. **Procesamiento Asíncrono:** Tarea en segundo plano iniciada.
   - Estado a ‘processing‘.
   - **Extracción (PyMuPDF):** Imágenes extraídas, filtradas y guardadas temporalmente (con datos base64).
   - ‘total_images_extracted‘ actualizado.
4. **Clasificación IA:** Para cada imagen:
   - **OpenAI GPT-4o:** Envío de imagen base64 y prompt.
   - Recepción de categoría (Map, Table, Picture).
5. **Almacenamiento:**
   - Clasificación guardada en DB (‘image_classifications‘).
   - Imagen movida a carpeta de categoría final.
   - ‘total_images_classified‘ actualizado.
6. **Finalización:** Estado a ‘completed‘ o ‘failed‘. PDF temporal eliminado.
7. **Consulta:** Usuario consulta estado/resultados vía API.

Este flujo asegura que la API permanezca responsiva mientras el procesamiento, que puede ser intensivo, ocurre en segundo plano.

2.2. Componentes Principales y sus Responsabilidades

- **Servidor API (FastAPI):** Expone endpoints RESTful, valida entradas, orquesta el flujo, gestiona la DB y sirve imágenes.
- **Extractor de Imágenes PDF (PyMuPDF):** Usa PyMuPDF para extraer, filtrar y preparar imágenes de PDFs.
- **Clasificador de Imágenes con IA (OpenAI):** Interactúa con la API de OpenAI (GPT-4o) para clasificar imágenes.
- **Base de Datos (SQLite):** Almacena metadatos de documentos y clasificaciones.
- **Almacenamiento de Archivos (Sistema de archivos local):** Gestiona PDFs temporales e imágenes clasificadas organizadas por categoría.

2.3. Stack Tecnológico Empleado

Categoría	Tecnología/Librería
Lenguaje	Python 3.8+
Framework API	FastAPI
Servidor ASGI	Uvicorn
Procesamiento PDF	PyMuPDF (fitz)
IA (Clasificación)	OpenAI API (GPT-4o)
Base de Datos	SQLite
Validación/Esquemas	Pydantic
Tareas Asíncronas	‘asyncio‘, FastAPI ‘BackgroundTasks‘

Cuadro 1: Resumen del Stack Tecnológico Utilizado.

2.4. Endpoints de la API Expuestos

La API REST, implementada con FastAPI, proporciona los siguientes puntos de interacción principales:

2.4.1. POST /upload-pdf/

Inicia el proceso de carga y procesamiento asíncrono de un archivo PDF. Devuelve un ‘document_id‘ para seguimiento.

2.4.2. GET /status/{document_id}

Obtiene el estado detallado del procesamiento para un documento, incluyendo las imágenes clasificadas y sus URLs.

2.4.3. GET /documents/

Lista de forma paginada todos los documentos procesados o encolados.

2.4.4. GET /health

Verifica el estado de salud de la API y la conexión a la base de datos.

2.4.5. GET /classified_images/{path:path}

Sirve estáticamente las imágenes clasificadas.

3. Solución Detallada de los Puntos de la Tarea 2

3.1. Punto 1: Acceso al Documento PDF

Requerimiento: Üsing this report https://d1vqu5ynjuwmto.cloudfront.net/assets/files/10526/q4_2023_txg_ep.pdf"**Solución:** El sistema acepta cualquier PDF vía el endpoint ‘/upload-pdf/‘. El PDF especificado se ha usado para pruebas y validación.

3.2. Punto 2: Extracción de Todas las Imágenes

Requerimiento: .ExtExtract all images from the file..."**Solución:** La lógica de extracción (usando PyMuPDF) se enfoca en imágenes relevantes, aplicando filtros de tamaño para descartar elementos gráficos menores.

3.3. Punto 3: Clasificación de Cada Imagen

Requerimiento: Çlassify each image as either: Map, Table, Picture"**Solución:** Se utiliza la API de OpenAI (GPT-4o).

3.3.1. Enfoque de Clasificación y Herramienta de IA

Se emplea el modelo multimodal GPT-4o de OpenAI.

3.3.2. Categorías de Clasificación

'Map', 'Table', y 'Picture'. Internamente se manejan 'ErrorAPI' y 'Original' (para respuestas no estándar de la IA).

3.3.3. Implementación de la Clasificación con IA

Se envía la imagen (codificada en base64) y un prompt específico a OpenAI. El prompt es:

> *.Analyze this image carefully. Considering its content and structure, classify it as exactly one of the following categories: 'Map' (if it primarily shows geographical areas, routes, or locations), 'Table' (if it primarily consists of data organized in rows and columns), or 'Picture' (for any other type of illustration, photograph, diagram not fitting Map/Table, or complex graphics). Respond with only the single chosen classification word (Map, Table, or Picture). Do not add any other text or punctuation."*

La respuesta se normaliza. Se utiliza 'asyncio.Semaphore' para controlar la concurrencia.

4. Flujo del Proceso de Extracción y Clasificación

El proceso sigue un flujo automatizado y orquestado.

4.1. Diagrama de Secuencia Conceptual

Flujo de Secuencia Principal del Procesamiento

Actor/Comp.	Interacción	Destino	Acción/Resultado
Usuario/API	POST /upload →	FastAPI App	Inicia tarea fondo.
FastAPI App	← HTTP 202	Usuario/API	Msg: "PDF recibido".
(Inicio Tarea Background: 'process_document_async')			
FastAPI App	'extract()' →	'PDFExtractor'	Solicita extracción.
'PDFExtractor'	Usa PyMuPDF	(Interno)	Extrae/filtra imgs.
'PDFExtractor'	← Imgs Lista	FastAPI App	Imgs (b64)+meta.
FastAPI App	Bucle por cada imagen:		
	'classify()' →	'ImgClassifier'	Solicita clasific.
'ImgClassifier'	API Call →	OpenAI GPT-4o	Envía img+prompt.
OpenAI GPT-4o	← Resp.	'ImgClassifier'	Categoría (e.g., "Map").
'ImgClassifier'	← Cat.	FastAPI App	Res. clasificación.
FastAPI App	DB Write →	SQLite	Guarda clasific.
FastAPI App	Move File →	File Sys.	Img a carpeta cat.
(Fin Tarea Background)			
FastAPI App	(Upd. estado)	SQLite	Doc. 'completed'/'failed'.

Figura 1: Diagrama de secuencia conceptual del flujo de extracción y clasificación.

5. Diseño del Sistema (Diagramas C4 Simplificados)

5.1. Nivel 1: Diagrama de Contexto del Sistema

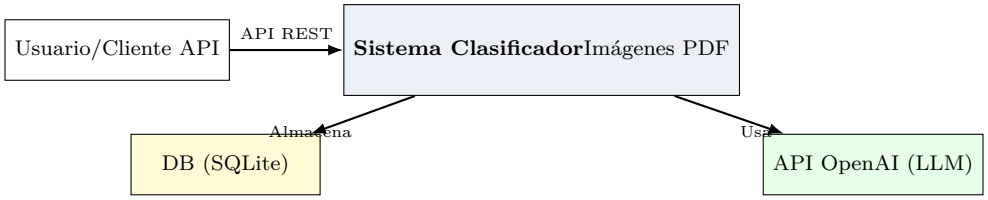

Figura 2: Diagrama de Contexto C4 simplificado del sistema y sus interacciones externas principales.

5.2. Nivel 2: Diagrama de Contenedores

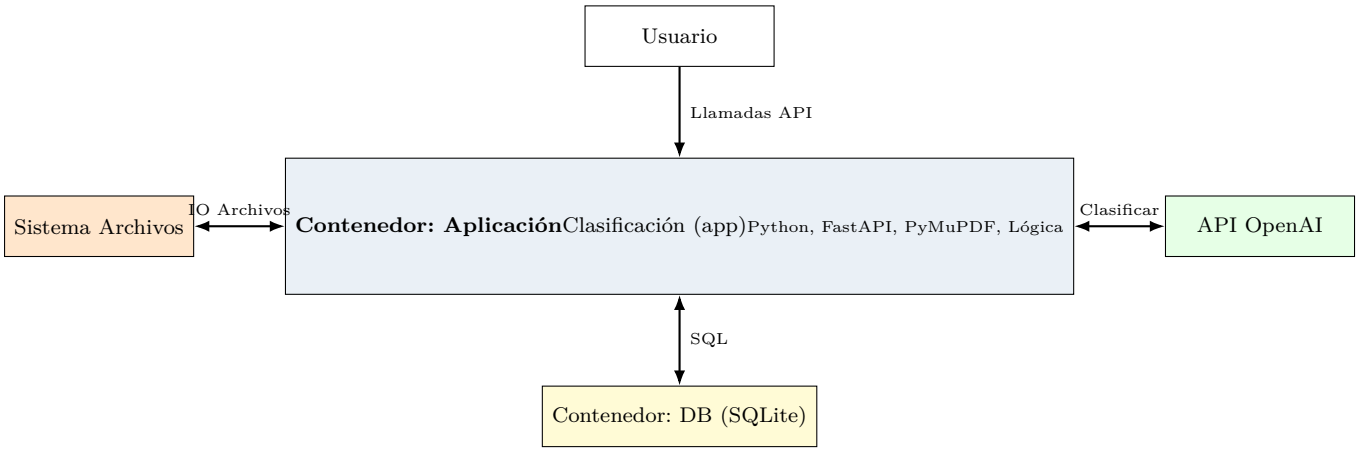

Figura 3: Diagrama de Contenedores C4 simplificado mostrando los principales componentes lógicos y sus interacciones.

6. Características y Etapas del Desarrollo

6.1. Características Principales Implementadas

- Carga Asíncrona de PDFs (‘BackgroundTasks‘ FastAPI).
- Extracción Eficiente de Imágenes (PyMuPDF, filtro por tamaño).
- Clasificación Inteligente con IA (OpenAI GPT-4o).
- Persistencia de Datos (SQLite).
- Almacenamiento Organizado de Imágenes (carpetas por categoría).
- API RESTful (carga, estado, listado) con documentación automática.

- Manejo Básico de Errores.
- Configuración por Variables de Entorno.

6.2. Etapas del Desarrollo (Simplificadas)

1. **Análisis y Diseño:** Comprensión Tarea 2, arquitectura.
2. **Entorno:** Python, dependencias, variables (‘OPENAI_API_KEY‘).
3. **Módulo Extracción:** Con PyMuPDF.
4. **Módulo Clasificación:** Con OpenAI API.
5. **Lógica de Negocio:** Orquestación asíncrona.
6. **API REST:** Endpoints FastAPI.
7. **Base de Datos:** Lógica SQLite.
8. **Pruebas y Refinamiento.**
9. **Documentación.**

7. Cumplimiento de Criterios de Evaluación (Tarea 2)

- **Uso de Herramientas IA:** OpenAI GPT-4o.
- **Extracción de Imágenes:** PyMuPDF.
- **Categorización:** ‘Map‘, ‘Table‘, ‘Picture‘.

8. Puntos Clave para el Éxito (Tarea 2)

8.1. Robustez de la Extracción

PyMuPDF es eficiente. El filtrado por tamaño mejora relevancia.

8.2. Precisión de la Clasificación

Depende del modelo (GPT-4o) y prompt. ‘detail: "low« en API OpenAI es un compromiso.

»8.3. Manejo de Errores/Limitaciones API

»Gestión de ‘RateLimitError‘. Semáforo para controlar concurrencia.

»8.4. Escalabilidad y Rendimiento

»Procesamiento asíncrono. SQLite para escala moderada. API IA es cuello de botella.

»9. Conclusión y Direcciones Futuras

»La solución cumple requisitos de Tarea 2. Es una base sólida.

»9.1. Posibles Mejoras Futuras

- »▪ Integración OCR.
- »▪ Modelos locales/open source.
- »▪ Mejorar puntajes de confianza.
- »▪ Mecanismo de feedback/reentrenamiento.
- »▪ Escalabilidad con colas de mensajes.
- »▪ Soporte otros formatos.
- »▪ Interfaz de Usuario (Frontend).
- »▪ Contenerización (Docker).
- »▪ Autenticación/Autorización.
- »▪ Procesamiento por lotes.

»El sistema actual es una implementación efectiva para el desafío.