UNIVERSIDAD EUROPEA DE MADRID

ESCUELA DE ARQUITECTURA, INGENIERÍA Y DISEÑO

MASTER UNIVERSITARIO EN ANALISIS DE DATOS MASIVOS (BIG DATA)

TRABAJO FIN DE MÁSTER

Sistema Inteligente para el Análisis Multimodal y Evaluación Automática de Presentaciones Técnicas en Entornos Formativos

MATTEO GATTI

Dirigido por

YUDITH COROMOTO CARDINALE

CURSO 2024-2025

TÍTULO: Sistema Inteligente para el Análisis Multimodal y Evaluación Automática de Presentaciones Técnicas en Entornos Formativos

AUTOR: MATTEO GATTI

TITULACIÓN: MASTER UNIVERSITARIO EN ANALISIS DE DATOS MASIVOS (BIG DATA)

DIRECTOR/ES DEL PROYECTO: YUDITH COROMOTO CARDINALE

FECHA: Noviembre de 2025

Resumen

En muchos entornos formativos y corporativos se acumulan vídeos de presentaciones y materiales complementarios que rara vez se vuelven a utilizar por la falta de herramientas que permitan analizarlos y estructurar la información que contienen. Para resolver este problema se desarrolló una solución basada en arquitectura Big Data que automatiza el procesamiento y almacenamiento de contenidos multimodales, combinando técnicas de transcripción automática, limpieza y enriquecimiento del texto con inteligencia artificial generativa. El sistema, denominado *Webinariitor*, fue diseñado e implementado íntegramente en Google Cloud, empleando un pipeline ETL desplegado en Cloud Run y un datalake estructurado en Google Cloud Storage con las capas *raw, bronze, silver y gold*.

La herramienta permite subir vídeos y documentos, generar transcripciones claras, crear resúmenes, formular preguntas tipo *test* y habilitar un chat contextual que responde exclusivamente con el material procesado. Todos los datos se almacenan de forma trazable mediante metadatos y políticas de gobernanza básicas, cumpliendo los principios de calidad y reproducibilidad del dato propios de un entorno Big Data.

Los resultados demuestran que la arquitectura desarrollada permite transformar presentaciones dispersas en una fuente estructurada de conocimiento, reduciendo tiempos de análisis y facilitando su reutilización. El trabajo pone de manifiesto el valor de integrar flujos ETL, datalakes y modelos generativos en la nube para automatizar la gestión del conocimiento técnico.

Palabras clave: Big Data, ETL, Google Cloud Run, Google Cloud Storage, IA generativa, procesamiento multimodal.

Abstract

In many educational and corporate environments, numerous videos of presentations and related materials are stored without being reused, mainly due to the lack of tools capable of analyzing and structuring the information they contain. To address this problem, a Big Data–oriented solution was developed to automate the processing and storage of multimodal content, combining automatic transcription, text cleaning, and enrichment techniques with generative artificial intelligence. The system, called *Webinariitor*, was designed and implemented entirely on Google Cloud, using an ETL pipeline deployed on Cloud Run and a data lake structured in Google Cloud Storage following the *raw*, *bronze*, *silver*, and *gold* layers.

The tool allows users to upload videos and documents, generate clear transcriptions, create summaries, produce multiple-choice *tests*, and enable a contextual chat that responds exclusively based on the processed material. All data are stored with complete traceability through metadata and basic governance policies, adhering to data quality and reproducibility principles typical of a Big Data environment.

The results show that the developed architecture successfully transforms dispersed presentations into a structured and reusable source of knowledge, reducing analysis time and improving accessibility. The work highlights the value of integrating ETL pipelines, data lakes, and generative models in the cloud to automate the management of technical knowledge.

Keywords: Big Data, ETL, Google Cloud Run, Google Cloud Storage, generative AI, multimodal processing.

AGRADECIMIENTOS

Quiero expresar mi más sincero agradecimiento a mi supervisora Yudith Cardinale, cuyo apoyo y orientación han sido fundamentales en el desarrollo de este trabajo.
No puedo dejar de agradecer a mi familia, por su incondicional apoyo y comprensión durante este proceso, así como a mis amigos, quienes me han animado y motivado en cada etapa. Este logro es también de todos ellos.

Cita - frase célebre

"El dolor es inevitable, el sufrimiento es opcional." - Haruki Murakami

TABLA RESUMEN

	DATOS
Nombre y apellidos:	Matteo Gatti
Título del proyecto:	Sistema Inteligente para el Análisis Multimodal y Evaluación Automática de Presentaciones Técnicas en Entornos Formativos
Directores del proyecto:	Yudith Coromoto Cardinale
El proyecto se ha realizado en colaboración de una empresa o a petición de una empresa:	NO
El proyecto ha implementado un producto: (esta entrada se puede marcar junto a la siguiente)	SI
El proyecto ha consistido en el desarrollo de una investigación o innovación: (esta entrada se puede marcar junto a la anterior)	SI
Objetivo general del proyecto:	Desarrollar una solución basada en arquitectura Big Data que automatice el procesamiento, almacenamiento y análisis de presentaciones técnicas y materiales formativos mediante un pipeline ETL desplegado en la nube, integrando servicios de transcripción, inteligencia artificial generativa y un datalake estructurado en Google Cloud para transformar contenidos dispersos en conocimiento accesible y reutilizable.

Índice

Índice de Figuras

Índice de Tablas

Capítulo 1. INTRODUCCIÓN

En numerosos contextos formativos y en reuniones técnicas se graban presentaciones y se generan materiales de apoyo como diapositivas, documentos o tablas que, aun conteniendo información valiosa, acaban archivados y apenas se reutilizan porque acceder a lo sustantivo exige volver a recorrer el vídeo completo o revisar manualmente cada archivo; este trabajo surge precisamente para resolver ese cuello de botella, proponiendo una solución con enfoque de ingeniería de datos que automatiza la ingesta y el tratamiento de esos contenidos mediante un flujo ETL en la nube y un datalake estructurado, de modo que lo grabado se transforme en conocimiento limpio, trazable y fácilmente reutilizable sin depender de visionados exhaustivos ni de procesos manuales dispersos.

1.1 Contexto y justificación

Tanto en el ámbito académico como en el corporativo se crean constantemente presentaciones orales acompañadas de materiales visuales como PowerPoint Word Excel o PDF. Aunque inicialmente tienen una función clara como exponer o explicar conceptos, lo habitual es que después queden almacenadas sin ser procesadas ni evaluadas. Recuperar la información que contienen suele implicar ver vídeos enteros o revisar manualmente varios documentos. Esto requiere tiempo y en la práctica muchas veces ni siquiera se hace.

Este proyecto surge también de una situación concreta vivida en el entorno profesional del autor. En su empresa a los becarios se les pide preparar presentaciones sobre temas relacionados con el mundo del dato. Estas exposiciones se graban y quedan disponibles, pero luego no se analizan ni se sacan conclusiones. El material se pierde aunque podría ser útil para formar a futuros compañeros o consolidar aprendizajes clave.

1.2 Planteamiento del problema

El reto principal es que no existe una forma centralizada y accesible para procesar todo ese contenido. Por un lado hay herramientas que transcriben audio otras que resumen texto y algunas que generan preguntas tipo *test* pero todas funcionan de forma separada. No hay una solución que combine en un mismo flujo la transcripción optimizada del vídeo el análisis de documentos complementarios y además permita interactuar directamente con el contenido. La consecuencia es una pérdida sistemática de conocimiento en sectores académicos y empresariales que manejan grandes volúmenes de información, evidenciando la necesidad de una arquitectura *cloud* con enfoque Big Data que garantice eficiencia, trazabilidad y aprovechamiento del dato.

1.3 Objetivo general

El proyecto nace con una intención clara: desarrollar una herramienta capaz de transformar una presentación técnica grabada en una fuente estructurada de conocimiento interactivo y accesible. Para ello se diseñó una solución con enfoque Big Data que automatiza todo el flujo de procesamiento desde la ingesta de datos hasta su visualización mediante un pipeline ETL desplegado en la nube. Este flujo abarca la transcripción del audio, la limpieza y enriquecimiento del texto con modelos de inteligencia artificial generativa, la integración de documentos complementarios y la creación de contenido evaluativo como resúmenes y pruebas tipo *test*, todo dentro de una arquitectura escalable basada en un datalake en Google Cloud Storage.

La necesidad de automatizar este proceso surge tanto en el ámbito académico como en el corporativo, donde el volumen de contenido en vídeo y materiales asociados crece de manera constante y rara vez se analiza o reutiliza. El sistema desarrollado aborda este desafío proporcionando una solución integral que cubre desde la entrada del vídeo y los documentos hasta la interacción final mediante un chat contextualizado, permitiendo acceder al conocimiento de forma dinámica y sin requerir conocimientos técnicos avanzados. A lo largo del trabajo se tomaron decisiones estratégicas orientadas a la eficiencia y la sostenibilidad del proyecto, como el uso de servicios del ecosistema Google, Speech-to-Text, Gemini y Cloud Run y el desarrollo de la interfaz con Streamlit, priorizando la simplicidad de uso y la fiabilidad técnica dentro de una arquitectura *cloud* moderna.

1.4 Objetivos específicos

1. Analizar y seleccionar las herramientas más adecuadas del ecosistema gratuito de Google como Gemini, Speech-to-Text, Cloud Storage y Cloud Run que permitan construir un flujo ETL completo en la nube, asegurando su integración y compatibilidad dentro de una arquitectura Big Data.

2. Desarrollar el módulo de ingesta y extracción, encargado de recibir los archivos de entrada (vídeos y documentos), separar la pista de audio y realizar la transcripción automática mediante Google Speech-to-Text, garantizando precisión y escalabilidad en el procesamiento.

3. Implementar una fase de transformación y mejora del texto, en la que se realice el post-procesamiento de las transcripciones mediante modelos de inteligencia artificial generativa, corrigiendo errores de reconocimiento y optimizando la claridad del contenido.

4. Diseñar e integrar un generador de resúmenes automáticos que permita obtener una visión global del material procesado (vídeo y documentos complementarios) sin necesidad de revisar manualmente el contenido completo.

5. Incorporar un módulo de análisis documental que procese presentaciones, hojas de cálculo y textos en distintos formatos (PPTX, DOCX, PDF, XLSX), extrayendo información relevante y normalizándola dentro del flujo ETL.
6. Desarrollar un componente evaluativo automatizado, capaz de generar preguntas de opción múltiple basadas en el contenido consolidado, con una única respuesta válida y formato estructurado en JSON para su reutilización y trazabilidad.
7. Implementar una interfaz conversacional contextualizada, que permita al usuario realizar preguntas en lenguaje natural y obtener respuestas fundamentadas exclusivamente en la información procesada, integrando las capacidades de Gemini.
8. Diseñar una arquitectura flexible y modular, sustentada en un datalake estructurado en capas (*raw, bronze, silver, gold*) y desplegada en Cloud Run, que facilite tanto su mantenimiento como su futura ampliación a entornos productivos o académicos.

1.5 Resultados obtenidos

El desarrollo del proyecto permitió consolidar un flujo ETL completo capaz de procesar de manera automática vídeos y documentos en cuestión de minutos dentro de una arquitectura *cloud*. El sistema extrae la pista de audio, la transcribe con precisión mediante Google Speech-to-Text y la mejora posteriormente con Gemini para obtener un texto más claro y legible. De forma paralela, analiza y normaliza las diapositivas y los documentos complementarios en distintos formatos, integrando toda la información en un único bloque de contenido coherente. A partir de este material consolidado, genera resúmenes estructurados, crea automáticamente preguntas de opción múltiple y habilita un chat contextualizado que responde únicamente en base a los datos cargados.

En las pruebas realizadas se comprobó la estabilidad del pipeline y su capacidad para ejecutar todas las etapas del proceso sin intervención manual, almacenando los resultados en un datalake organizado por capas (*raw, bronze, silver y gold*), con metadatos y métricas de rendimiento asociadas a cada ejecución. Los resultados obtenidos demuestran que la integración de inteligencia artificial generativa, procesamiento multimodal y servicios *cloud* dentro de una arquitectura Big Data permite automatizar la gestión del conocimiento y dar un nuevo valor analítico a contenidos audiovisuales y documentales que, en condiciones normales, permanecerían inactivos y sin aprovechamiento real.

1.6 Estructura de la memoria

La memoria se organiza en siete capítulos que siguen un orden lógico para facilitar la lectura y comprensión del trabajo además de las secciones de referencias y apéndices.

- **Capítulo 1, Introducción**, recoge el contexto en el que surge el proyecto, los motivos que lo justifican, el problema detectado y los objetivos que guiaron su desarrollo. También se presentan los resultados obtenidos y la estructura de la memoria.

- **Capítulo 2, Marco teórico y marco tecnológico**, describe los conceptos clave que sustentan el enfoque del trabajo, abordando los principios de la arquitectura Big Data, el funcionamiento de los flujos ETL y el papel de los servicios de Google Cloud en la solución desarrollada.

- **Capítulo 3, Estado del arte**, analiza las soluciones existentes en el mercado que abordan tareas similares, comparándolas con la herramienta propuesta y destacando su aportación diferencial.

- **Capítulo 4, Metodología**, presenta las fases de trabajo que se siguieron durante el desarrollo del proyecto y el presupuesto estimado.

- **Capítulo 5, Arquitectura del sistema y del datalake**, presenta el diseño general de la solución, explicando cómo se organiza el datalake en capas *(raw, bronze, silver y gold*), la función de cada zona dentro del pipeline y la forma en que los datos y metadatos se almacenan y versionan. También se describe la orquestación de los procesos y su relación con el despliegue en Cloud Run.

- **Capítulo 6, Desarrollo e implementación técnica**, describe en profundidad cada uno de los módulos del sistema, transcripción, limpieza, resumen, *test* y chat, así como la estructura del flujo ETL, la integración con el datalake, la gestión de metadatos y las métricas de rendimiento generadas durante el procesamiento.

- **Capítulo 7, Resultados y discusión**, presenta los resultados obtenidos tras la ejecución de las pruebas en distintos escenarios, analiza los tiempos de procesamiento y la estabilidad del pipeline, y evalúa el impacto de la arquitectura *cloud* en la eficiencia, escalabilidad y trazabilidad del sistema.

- **Capítulo 8, Conclusiones y futuras líneas de trabajo**, recoge los aprendizajes derivados del proyecto, reflexiona sobre su aplicabilidad en entornos educativos y corporativos, y propone líneas de evolución centradas en la mejora del rendimiento, la incorporación de nuevos formatos de datos y la extensión del sistema hacia entornos de producción.

- **Apéndice 1: Descripción del código**, detalla los módulos de desarrollo creados específicamente para la herramienta y su papel dentro del flujo técnico.

- **Apéndice 2: Materiales utilizados**, incluye ejemplos de los vídeos y documentos empleados durante las pruebas para validar el funcionamiento del sistema.

Capítulo 2. MARCO TEÓRICO Y MARCO TECNOLÓGICO

Este capítulo reúne los fundamentos conceptuales y tecnológicos que sustentan el desarrollo de la herramienta propuesta. En primer lugar, se presentan las bases teóricas que justifican el uso de contenidos audiovisuales, la transcripción automática, los resúmenes generados por inteligencia artificial y el papel de los chatbots educativos en entornos formativos. En segundo lugar, se explican las decisiones tecnológicas que guiaron la implementación del sistema, describiendo los componentes empleados, la lógica del flujo de datos y la integración de los servicios *cloud*. Finalmente, se aborda la arquitectura Big Data sobre la que se apoya la aplicación, centrada en un pipeline ETL desplegado en la nube y un datalake estructurado según el modelo Medallion.

2.1 Marco teórico

El uso de contenidos en vídeo en educación ha crecido de forma notable en los últimos años y cada vez se utiliza más como soporte central en la formación tanto en la universidad como en el mundo empresarial. La literatura coincide en que cuando están bien diseñados estos vídeos ayudan a entender y retener mejor los conceptos. Noetel et al. (2021) demostraron que cuando un vídeo está orientado a objetivos concretos mejora el nivel de atención y comprensión por parte del estudiante. Mayer (2009) también destacó el poder del aprendizaje multimedia al combinar narración oral con elementos visuales y Guo et al. (2014) señalaron que la duración o el estilo del vídeo influyen directamente en la implicación del alumno.

Estos estudios refuerzan el valor del enfoque multimodal en el aprendizaje que consiste en mezclar distintos estímulos visuales auditivos y textuales para generar una experiencia más rica. La combinación de vídeos con materiales escritos y algún tipo de interacción como *test* o preguntas ayuda a consolidar el aprendizaje. En este sentido los *chatbots* educativos también se han convertido en una herramienta útil para guiar al estudiante resolver dudas o reforzar ideas clave. Winkler y Söllner (2018) defienden que estos sistemas fomentan la autonomía del estudiante y su capacidad para aprender por su cuenta.

A esto se suma el potencial de la inteligencia artificial generativa que permite transformar contenidos de forma automática. Modelos como Gemini son capaces de generar resúmenes crear preguntas de opción múltiple o responder preguntas sobre el contenido sin necesidad de intervención humana directa. Mittal et al. (2024) explican cómo estas tecnologías están cambiando la manera de enseñar y aprender mientras que Denny et al. (2024) advierten de la importancia de diseñar sistemas que mantengan el control sobre las fuentes para evitar errores o respuestas sin base.

Todo esto encaja con la necesidad que motivó este proyecto dar un uso real y práctico a las presentaciones técnicas grabadas que muchas veces se almacenan y se olvidan sin ningún tipo de análisis ni reutilización.

2.2 Marco tecnológico

Desde el inicio del proyecto se decidió trabajar dentro del ecosistema de Google Cloud, al ofrecer un conjunto de herramientas bien integradas, con autenticación unificada, gestión sencilla de credenciales y la posibilidad de operar bajo un enfoque *serverless*, sin necesidad de desplegar infraestructuras complejas. Esta elección garantizó la escalabilidad, la trazabilidad y la sostenibilidad del sistema, pilares esenciales en una arquitectura de tipo Big Data.

El sistema integra distintos componentes dentro de un flujo ETL (*Extract, Transform, Load*) completamente automatizado. En la fase de extracción, se utiliza la librería MoviePy junto con pydub para separar la pista de audio del vídeo original, la cual se envía a la API Google Speech-to-Text para su transcripción automática. Este servicio convierte el audio en texto, reconociendo pausas, signos de puntuación y entonaciones naturales, lo que proporciona una base sólida para el procesamiento posterior.

En la etapa de transformación, la transcripción es limpiada y mejorada mediante el modelo Gemini 2.5 Flash, que corrige errores de reconocimiento, reorganiza frases y optimiza la legibilidad del texto. De forma paralela, se extrae información de los documentos complementarios subidos por el usuario presentaciones PowerPoint, textos Word, hojas Excel y archivos PDF utilizando librerías específicas como python-pptx, python-docx, openpyxl y PyPDF2. Todo el contenido procesado se consolida en un único bloque textual, que sirve como fuente para las siguientes etapas del pipeline: la generación de resúmenes automáticos, la creación de *test* de comprensión y la interacción mediante chat contextualizado.

La capa de carga y visualización se implementa en una aplicación web desarrollada con Streamlit, seleccionada por su sencillez y capacidad para construir interfaces limpias y funcionales directamente en Python. La aplicación gestiona las distintas fases del flujo de trabajo, subida, procesamiento y visualización, además de almacenar los resultados y las métricas en un entorno *cloud* totalmente integrado. Aunque se evaluaron alternativas como Flask, Dash o React, Streamlit ofreció la mejor relación entre velocidad de desarrollo, mantenibilidad y experiencia de usuario.

2.3 Arquitectura cloud y datalake

La arquitectura técnica del proyecto se sustenta en un pipeline ETL desplegado en Google Cloud Run y respaldado por un datalake estructurado en Google Cloud Storage (GCS), siguiendo el modelo Medallion ampliamente utilizado en entornos Big Data. Este diseño permite organizar los datos en diferentes capas de calidad y transformación, asegurando la trazabilidad y la evolución del contenido a lo largo del proceso.

El datalake se estructura en cuatro zonas principales:

- **Raw**: almacenamiento inmutable de los archivos originales (vídeos, documentos y materiales de apoyo).
- **Bronze**: datos derivados mínimos, como los audios extraídos en formato MP3.
- **Silver**: textos limpios y estructurados, incluyendo transcripciones optimizadas y contenido normalizado de las distintas fuentes documentales.
- **Gold (curated)**: artefactos finales generados por el sistema, como resúmenes, *test*, métricas y el texto combinado.

Cada artefacto almacenado en el datalake incluye un archivo de metadatos en formato JSON, con información sobre su identificador, hash, fecha de creación, tipo de contenido y origen, lo que permite una trazabilidad completa del flujo de datos.

El despliegue en Google Cloud Run ofrece un entorno *serverless* que garantiza la ejecución eficiente de la aplicación sin necesidad de mantener servidores activos. Además, el uso de Secret Manager para la gestión de credenciales y Cloud Build para la contenerización y despliegue automatizado asegura una arquitectura reproducible, segura y fácilmente escalable.

La integración de estas tecnologías permite que la herramienta funcione como un sistema coherente de procesamiento de datos multimodales en la nube, combinando la automatización propia del Big Data con las capacidades analíticas de la inteligencia artificial generativa. Este enfoque no solo optimiza la gestión del conocimiento, sino que sienta las bases para futuras ampliaciones orientadas a la analítica avanzada, la evaluación automatizada y la minería de contenido educativo.

Capítulo 3. ESTADO DEL ARTE

Este capítulo presenta una revisión de los principales enfoques y herramientas actualmente disponibles que abordan tareas similares a las desarrolladas en este proyecto. Se analizan soluciones comerciales y académicas centradas en la transcripción de audio, la generación de resúmenes, la evaluación automática del conocimiento y la interacción mediante chatbots. El objetivo es situar la herramienta propuesta dentro del panorama tecnológico actual, identificando sus limitaciones, su grado de integración y el valor diferencial que aporta el sistema desarrollado a partir de una arquitectura *cloud* con enfoque Big Data.

3.1 Soluciones actuales en el mercado

En los últimos años han surgido múltiples herramientas enfocadas en mejorar la gestión del contenido audiovisual en entornos educativos y profesionales. Plataformas como Fireflies.ai y Otter.ai están diseñadas principalmente para transcribir reuniones, generar notas automáticas y organizar conversaciones. Funcionan bien cuando se trata de extraer el texto hablado, pero no permiten añadir documentos complementarios ni realizar análisis más profundos del contenido. Tampoco ofrecen funciones de evaluación como *tests* o interacción conversacional con el material.

Por otro lado, Eightify ha centrado sus esfuerzos en resumir vídeos de YouTube y podcasts. Su enfoque está pensado para ahorrar tiempo en el consumo de contenidos largos, aunque carece de funciones que permitan trabajar activamente con ese resumen, como la generación de preguntas o una interfaz para hacer consultas sobre el vídeo. Tampoco ofrece integración con otros tipos de documentos, lo que limita su utilidad en contextos más técnicos o formativos.

Una propuesta distinta es la de EaseMate ChatPPT, que permite chatear con documentos en formato PowerPoint, Word o PDF. Aunque esta funcionalidad resulta interesante, no permite trabajar con vídeo ni con contenido sonoro. Además, su sistema no unifica las fuentes, por lo que no hay un análisis conjunto de todo el material cargado. Esto obliga al usuario a consultar por separado cada archivo, sin una visión consolidada del contenido global.

Otras herramientas como Notta o Sonix también ofrecen buenos resultados en tareas de transcripción, pero suelen trabajar con modelos cerrados y orientados al mercado anglosajón. Esto limita la personalización y dificulta su uso en proyectos que requieren adaptación a idiomas o contextos diferentes.

Además, muchas de estas plataformas operan bajo licencias comerciales y entornos no configurables, lo que dificulta su integración en proyectos personalizados o en arquitecturas orientadas a Big Data.

En conjunto, la mayoría de las soluciones existentes comparten un patrón común: abordan tareas aisladas de procesamiento, transcripción, resumen o consulta, pero carecen de un pipeline integrado, una arquitectura *cloud* reproducible y una estructura de almacenamiento

escalable y trazable que permita gestionar el dato como un activo reutilizable a lo largo del tiempo.

3.2 Aportación diferencial del sistema propuesto

Frente a las soluciones descritas, la herramienta desarrollada en este trabajo presenta un enfoque integral que combina la transcripción, el análisis, la evaluación y la interacción en un mismo flujo de procesamiento. La aplicación no se limita a una tarea específica, sino que implementa un pipeline ETL completo en la nube, donde cada fase, extracción, transformación y carga, se ejecuta de manera automatizada y trazable.

El sistema permite procesar vídeos junto con documentos complementarios en distintos formatos, integrando todos los datos en un único bloque de información coherente. Este bloque se convierte en la base para generar una transcripción mejorada, un resumen estructurado, un *test* de comprensión interactivo y un chat contextualizado, todo ello accesible desde una interfaz ligera y clara desarrollada con Streamlit.

A diferencia de las soluciones comerciales, esta herramienta se apoya en una arquitectura *cloud-native* y escalable, desplegada en Google Cloud Run y respaldada por un datalake estructurado en capas (*raw, bronze, silver, gold*) dentro de Google Cloud Storage, lo que permite conservar y versionar tanto los datos como los metadatos de cada ejecución. Esta organización aporta un nivel de trazabilidad y control del ciclo de vida del dato que no está presente en las aplicaciones tradicionales.

Otro elemento distintivo es el uso de modelos abiertos y configurables, concretamente Gemini 2.5 Flash, que ofrece flexibilidad en el diseño de prompts, control sobre los tiempos de respuesta y personalización del tipo de salida generada. En lugar de actuar como una “caja negra”, el sistema desarrollado otorga al usuario control sobre los procesos de generación, transformación y almacenamiento del contenido.

Además, el enfoque educativo y analítico del proyecto lo diferencia de las herramientas orientadas a productividad personal o gestión de reuniones. La posibilidad de transformar una presentación técnica en una fuente estructurada de conocimiento con capacidades de autoevaluación e interacción convierte esta solución en una propuesta innovadora dentro del campo del aprendizaje asistido por IA.

Por último, la decisión de construir una interfaz sencilla, sin dependencias de licencias comerciales ni infraestructuras complejas, permite que el sistema sea fácilmente adaptable a contextos reales de formación o investigación. Su arquitectura modular y su despliegue basado en servicios gratuitos o de bajo coste dentro del ecosistema Google garantizan una implementación sostenible, escalable y alineada con los principios de la ingeniería de datos moderna.

En definitiva, la aportación diferencial del sistema propuesto radica en su capacidad para unificar procesamiento multimodal, gobernanza del dato y generación automática de

conocimiento dentro de una arquitectura *cloud* robusta, posicionándose como una herramienta avanzada para la gestión, el análisis y la reutilización de presentaciones técnicas en entornos educativos y corporativos.

Capítulo 4. METODOLOGÍA

Este capítulo describe las fases de desarrollo del proyecto, desde la planificación inicial hasta las pruebas finales, explicando cómo se estructuró el trabajo para validar cada componente antes de su integración en una arquitectura única. Se detalla la metodología seguida, de carácter iterativo y modular, lo que permitió desarrollar cada módulo de forma independiente, comprobar su funcionamiento y, finalmente, consolidar el sistema dentro de un flujo ETL desplegado en la nube. Asimismo, se incluye una estimación general de los recursos técnicos y humanos empleados, junto con las decisiones estratégicas adoptadas en materia de diseño, escalabilidad y gobernanza del dato.

4.1 Fases de trabajo

El enfoque metodológico adoptado fue iterativo y modular, orientado a la validación progresiva de cada parte del sistema antes de su integración en el conjunto. Desde el inicio se concibió una arquitectura con capacidad de crecimiento, flexible y adaptada a los principios de la ingeniería de datos moderna, de manera que nuevos módulos o fuentes de información pudieran incorporarse sin necesidad de rehacer el sistema existente.

Fase 1: Exploración y selección de tecnologías

En la primera etapa se realizó un análisis del ecosistema de Google Cloud, identificando los servicios gratuitos o de bajo coste más adecuados para el desarrollo de un flujo de procesamiento automatizado. Se probaron distintas combinaciones de herramientas como Gemini, Google Speech-to-Text, Cloud Storage, Colab y Drive, evaluando su integración y escalabilidad. Tras varias pruebas, se seleccionaron Speech-to-Text para la transcripción automática del audio, Gemini 2.5 Flash para las tareas de limpieza, resumen y generación de preguntas tipo *test*, y Google Cloud Storage como base para el datalake. Esta combinación garantizaba interoperabilidad, coherencia técnica y una gestión sencilla de credenciales bajo una única cuenta de servicio.

Fase 2: Diseño del flujo ETL y definición del datalake

En la segunda fase se definió el flujo de datos completo ETL (extracción, transformación y carga) y se diseñó la estructura del datalake siguiendo el modelo *Medallion Architecture* con capas *raw, bronze, silver y gold*. Se estableció la forma en que los datos serían almacenados, organizados y versionados, incorporando un esquema de metadatos (.meta.json) para cada artefacto generado.

Durante esta etapa también se definieron los formatos de entrada admitidos: vídeo MP4, presentaciones PowerPoint (PPTX), documentos Word (DOCX), hojas Excel (XLSX) y archivos PDF. La estructura resultante debía ser lo suficientemente flexible como para incorporar nuevos tipos de archivo en futuras versiones, manteniendo la coherencia de los procesos.

Fase 3: Desarrollo modular del sistema

Una vez definido el flujo, se procedió al desarrollo independiente de cada módulo. El trabajo comenzó con la implementación de la transcripción automática, integrando la extracción del audio mediante MoviePy y pydub y su envío al bucket temporal para el procesamiento con Google Speech-to-Text.

Posteriormente se añadió la fase de limpieza y reorganización del texto, gestionada por Gemini, que permitía obtener una transcripción coherente y legible a partir del texto original.

Después se implementaron los módulos de resumen automático y generación de *test*, ambos basados en la API de Gemini 2.5 Flash, y finalmente se incorporó el módulo de análisis documental, encargado de extraer el texto de los archivos complementarios y fusionarlo con el material transcrito. Cada componente se validó individualmente, verificando su rendimiento y estabilidad antes de integrarlo en el pipeline general.

Fase 4: Integración en una aplicación unificada

En la cuarta fase se integraron todos los módulos en una única interfaz desarrollada con Streamlit, elegida por su facilidad de uso y su capacidad para construir aplicaciones web interactivas con Python. La aplicación permitió centralizar el control del flujo, desde la carga de archivos hasta la visualización de los resultados, e incorporar elementos de diseño que facilitaron una navegación fluida entre las distintas secciones (transcripción, resumen, *test*, chat y contenido combinado).

Durante esta etapa se implementó también el registro automático de métricas de rendimiento, midiendo los tiempos de ejecución de cada fase (extracción de audio, transcripción, limpieza, resumen, *test* y tiempo total) y almacenándolos en el datalake en formato JSON y CSV dentro de la carpeta gold/metrics.

Fase 5: Despliegue en la nube y orquestación

Una vez validado el sistema localmente, se procedió a su despliegue en Google Cloud Run bajo un modelo *serverless*, utilizando Docker y Cloud Build para la contenerización del proyecto. Las credenciales se gestionaron mediante Secret Manager, garantizando la seguridad del entorno y la automatización del despliegue. Esta fase permitió comprobar el comportamiento del sistema en un entorno *cloud* real, asegurando su escalabilidad, la correcta interacción entre servicios (Speech-to-Text, Gemini, Cloud Storage) y la persistencia de datos en el datalake.

Fase 6: Validación funcional y pruebas

La última fase correspondió a la validación del sistema completo. Se realizaron pruebas con presentaciones reales en vídeo, acompañadas de documentos complementarios en distintos formatos, para evaluar el comportamiento del pipeline. Los resultados demostraron que el flujo se ejecuta de forma estable, con una duración media de entre dos y tres minutos por proceso, sin errores en la transcripción ni en la generación de artefactos.

Las métricas recogidas evidenciaron la eficiencia del sistema y confirmaron que la arquitectura *cloud* implementada soporta la automatización de procesos complejos de análisis y evaluación

de contenido en un tiempo reducido. El detalle de los materiales empleados durante esta etapa se presenta en el Apéndice 2.

4.2 Presupuesto

Aunque se utilizaron principalmente herramientas gratuitas, se estima el coste aproximado que tendría este proyecto en un entorno real en la Tabla 1, incluyendo horas de trabajo, equipamiento y licencias si fueran necesarias.

Tabla 1. Presupuesto

Tipo de coste	**Valor**	**Comentarios**
Horas de trabajo en el proyecto	500 h	Incluye tiempo dedicado a la planificación, diseño, desarrollo, pruebas, documentación, escritura de la memoria y preparación de entregas.
Equipo técnico utilizado	1.500 €	Portátil MSI P65 Creator 9SE con GPU dedicada, adecuado para tareas de desarrollo intensivo, análisis de audio/vídeo y ejecución local de herramientas de IA.
Servicios en la nube	0 € (dentro del plan gratuito)	Uso de APIs de Google Cloud (Speech-to-Text, Gemini, Cloud Storage, Cloud Run) dentro del límite gratuito.
Software utilizado	0 €	Todo el software empleado es de uso libre o gratuito: Python, Streamlit, Google Cloud APIs (dentro del límite gratuito), Visual Studio Code, MoviePy, etc.
Estudios e informes	0 €	No se ha requerido la compra de informes externos ni acceso a revistas de pago. Las fuentes utilizadas han sido de libre acceso académico o técnico.
Materiales empleados	0 €	No se utilizaron sensores, hardware adicional ni otro tipo de material físico. Todo el desarrollo se realizó en entorno digital.

El coste total estimado para replicar el proyecto en un entorno académico o formativo se mantiene, por tanto, dentro de los márgenes mínimos, confirmando la viabilidad económica del sistema incluso en escenarios con recursos limitados.

4.3 Conclusión metodológica

La metodología aplicada permitió avanzar desde una fase exploratoria hacia un sistema funcional y escalable, basado en principios de modularidad, integración y automatización. Cada componente fue desarrollado, probado y ensamblado dentro de un flujo coherente que refleja las etapas clásicas de un pipeline ETL moderno.

El uso de servicios *cloud* y la estructuración del datalake facilitaron no solo la ejecución técnica del proyecto, sino también su gobernanza y reproducibilidad, consolidando la solución como un ejemplo práctico de cómo aplicar metodologías de ingeniería de datos al análisis multimodal y a la gestión automatizada del conocimiento.

Capítulo 5. ARQUITECTURA DEL SISTEMA Y DEL DATALAKE

Este capítulo describe la arquitectura general del sistema, detalla cómo se estructura el flujo de procesamiento y explica la organización del datalake sobre el que se apoya toda la solución. Se presentan los componentes técnicos que intervienen en cada etapa, la relación entre las distintas capas del pipeline ETL y la forma en que los datos y metadatos son almacenados, versionados y gestionados dentro del entorno *cloud*. Finalmente, se expone cómo la aplicación se despliega en Google Cloud Run bajo un modelo *serverless*, garantizando escalabilidad, eficiencia y trazabilidad en cada ejecución.

5.1 Diseño general del sistema

La aplicación se concibe como un pipeline ETL completo (*Extract, Transform, Load*) que automatiza la gestión del contenido audiovisual y documental de una presentación técnica. Todo el flujo se desarrolla y ejecuta sobre la nube de Google, combinando distintos servicios que trabajan de forma coordinada dentro de un entorno controlado y reproducible.

En la fase de extracción, el sistema recibe como entrada un vídeo y documentos complementarios en distintos formatos. El vídeo se procesa localmente para extraer la pista de audio mediante las librerías MoviePy y pydub, y este archivo se envía al bucket temporal de Google Cloud Storage para su transcripción automática con Google Speech-to-Text, un servicio optimizado para el reconocimiento de voz natural en distintos contextos.

La fase de transformación comienza una vez obtenida la transcripción bruta. El texto es limpiado, reorganizado y enriquecido mediante Gemini 2.5 Flash, que corrige errores de interpretación, ajusta la puntuación y mejora la legibilidad general. Paralelamente, los documentos asociados como PowerPoint, Word, Excel y PDF se procesan con librerías específicas que extraen su contenido textual y lo preparan para integrarse con la transcripción. Todos estos datos convergen en un bloque de texto unificado que constituye la base de los artefactos generados posteriormente: resúmenes automáticos, *test* de comprensión y chat contextualizado.

Por último, la fase de carga guarda toda la información resultante en el datalake, estructurado en distintas capas que representan niveles de calidad y transformación del dato. Esta organización permite gestionar grandes volúmenes de información de manera ordenada, asegurar la trazabilidad de cada ejecución y facilitar la reutilización de los resultados en futuros análisis.

5.2 Arquitectura del datalake

El datalake, implementado sobre Google Cloud Storage (GCS), constituye el núcleo de persistencia del sistema. Siguiendo el modelo *Medallion Architecture*, se estructura en cuatro niveles, *raw*, *bronze*, *silver* y *gold*, que reflejan las fases del pipeline y los distintos grados de refinamiento de los datos.

- **Raw:** Es la zona de aterrizaje donde se almacenan los archivos originales tal y como fueron cargados por el usuario. Incluye vídeos, presentaciones y documentos complementarios en su formato nativo, sin ningún tipo de transformación. Su función es preservar la versión inicial del material y servir como fuente inmutable para auditorías o reprocesamientos futuros.
- **Bronze:** Contiene los datos derivados mínimos generados durante el proceso de extracción, como las pistas de audio separadas del vídeo original. Esta capa representa el primer nivel de transformación del flujo, donde el contenido ya se encuentra preparado para las tareas de análisis automático.
- **Silver:** Almacena los textos limpios y normalizados, tanto las transcripciones mejoradas como la información textual extraída de los documentos complementarios. Aquí los datos ya han pasado por una validación de coherencia y formato, lo que permite su integración en etapas analíticas superiores.
- **Gold (Curated):** Reúne los artefactos finales listos para el consumo o la evaluación. Incluye los resúmenes automáticos, los *tests* generados, el texto combinado y los ficheros de métricas y manifiestos. Esta capa es el resultado consolidado del pipeline, donde los datos adquieren su máximo valor analítico y pedagógico.

Cada archivo almacenado en el datalake genera un fichero adicional con extensión *.meta.json* que contiene su metainformación estructurada, como el identificador único, el *hash* de verificación, la fecha de creación, el tipo de artefacto, las etiquetas de clasificación y su relación con otros elementos del flujo. Esta estrategia de metadatos garantiza la trazabilidad de los procesos y facilita la gobernanza de los datos dentro del sistema.

Además, la estructura jerárquica por fecha (año/mes/día) permite organizar las ejecuciones y mantener un registro ordenado y versionado de todas las operaciones realizadas, reforzando la reproducibilidad del entorno.

En la Ilustración 1 se representa de manera esquemática la arquitectura del datalake desarrollada, basada en el modelo *Medallion Architecture*. Cada capa cumple una función específica dentro del flujo ETL, asegurando la trazabilidad y la progresiva mejora de la calidad del dato.

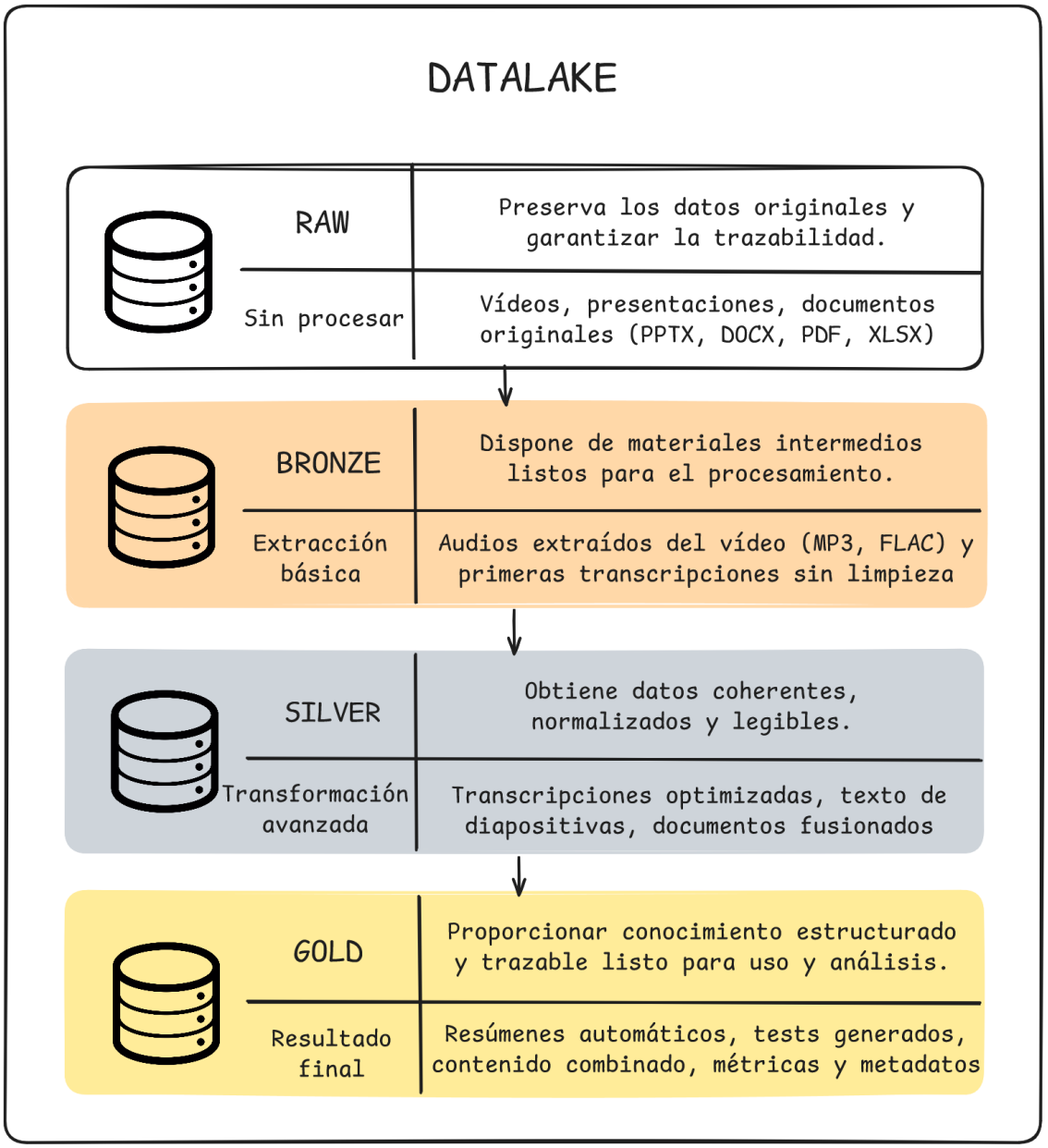

Ilustración 1 Arquitectura Medallon del Datalake

5.3 Orquestación y gestión del pipeline

El flujo ETL está completamente automatizado y orquestado desde la aplicación desarrollada en Streamlit, que actúa como punto de control y supervisión de todas las etapas. Desde la interfaz, el usuario carga los archivos, inicia el procesamiento y visualiza los resultados sin necesidad de intervención técnica. Detrás de esa simplicidad se encuentra un sistema que coordina las operaciones en tiempo real y gestiona los servicios *cloud* de forma transparente.

Cada vez que se ejecuta un análisis, la aplicación mide y registra las métricas de rendimiento de cada fase, extracción de audio, transcripción, limpieza, resumen, *test* y tiempo total de ejecución, y las guarda en el datalake dentro de la carpeta gold/metrics, tanto en formato JSON como CSV. Este componente de observabilidad permite identificar tiempos medios, optimizar recursos y evaluar el comportamiento del pipeline bajo diferentes cargas de trabajo.

Los procesos de subida al datalake utilizan la API nativa de Google Cloud Storage y una serie de funciones auxiliares que garantizan la creación automática de rutas, la nomenclatura homogénea de archivos y la generación de metadatos legibles. Esta automatización reduce los errores humanos y mantiene una estructura coherente en todas las ejecuciones.

5.4 Despliegue en la nube

El sistema se despliega sobre Google Cloud Run, bajo un modelo *serverless*, lo que significa que la infraestructura se adapta dinámicamente a la demanda sin requerir servidores dedicados ni configuraciones manuales. Todo el código fuente se empaqueta en un contenedor Docker, gestionado mediante Google Cloud Build, que automatiza el proceso de construcción y publicación de la imagen.

Las credenciales y claves necesarias para acceder a los distintos servicios *cloud* se gestionan mediante Google Secret Manager, garantizando la seguridad del entorno y evitando exponer información sensible en el repositorio. La comunicación entre los servicios Cloud Run, Cloud Storage, Speech-to-Text y Gemini se realiza dentro del ecosistema Google, lo que simplifica la autenticación y elimina dependencias externas.

Este enfoque de despliegue proporciona escalabilidad automática, alto rendimiento y una arquitectura limpia y mantenible, en línea con los principios modernos de la ingeniería de datos. Además, permite que el sistema pueda ejecutarse de forma controlada, manteniendo la trazabilidad completa del proceso y asegurando la persistencia de los datos en cada ejecución.

5.5 Síntesis de la arquitectura

En conjunto, la arquitectura del sistema puede describirse como una integración coherente de procesamiento multimodal, inteligencia artificial generativa y gestión del dato en la nube. El pipeline ETL no solo permite transformar presentaciones en conocimiento estructurado, sino que además proporciona una infraestructura capaz de garantizar la trazabilidad, la escalabilidad y la calidad del dato.

La combinación de Google Cloud Storage, Speech-to-Text, Gemini y Cloud Run conforma un ecosistema homogéneo y eficiente que hace posible automatizar un proceso que antes dependía de tareas manuales dispersas. De esta manera, el proyecto trasciende el ámbito de la aplicación educativa y se consolida como una solución de arquitectura Big Data aplicada al análisis y reutilización del conocimiento en entornos formativos y empresariales.

Capítulo 6. DESARROLLO

Este capítulo describe con detalle todo el proceso de desarrollo de la herramienta, siguiendo las fases metodológicas establecidas y reflejando las decisiones técnicas que marcaron su evolución. Se explican las elecciones de diseño, las herramientas empleadas, las dificultades encontradas y las soluciones aplicadas en cada paso del flujo de trabajo. La estructura modular adoptada permitió validar cada componente de manera independiente y garantizar su correcta integración dentro de una arquitectura coherente, escalable y orientada a la nube.

6.1 Fase de exploración de herramientas

Durante las primeras etapas del proyecto se realizó una exploración exhaustiva de las herramientas disponibles en el ecosistema de Google Cloud, con el objetivo de identificar aquellas que ofrecieran el mejor equilibrio entre funcionalidad, facilidad de integración y coste cero dentro de los planes gratuitos. El propósito era construir una solución completa que pudiera ejecutarse sin infraestructuras complejas ni dependencias externas, manteniendo la sostenibilidad de un entorno académico.

Se evaluaron diferentes servicios y librerías de procesamiento de datos, priorizando las que permitieran trabajar con transcripción de audio, generación de texto, análisis de documentos y almacenamiento persistente en la nube. Tras varias pruebas, se seleccionaron tres componentes principales que terminaron constituyendo el núcleo técnico del sistema.

El primero fue Google Cloud Speech-to-Text, utilizado para convertir de forma automática los archivos de audio extraídos de los vídeos en texto, con soporte para el idioma español y puntuación automática. El segundo fue Gemini 2.5 Flash, accedido directamente a través de la API oficial google-generativeai. Este modelo se encargó de limpiar y reorganizar las transcripciones, generar resúmenes, crear preguntas tipo *test* y proporcionar respuestas contextuales al chat. Por último, se empleó Google Cloud Storage (GCS) como base de almacenamiento centralizada, actuando como datalake del sistema.

Para la interfaz, se eligió Streamlit, una herramienta que demostró ser ideal por su integración directa con Python, su capacidad para crear aplicaciones web interactivas con poco código y su facilidad de despliegue en entornos locales o *cloud*. Aunque se valoraron opciones como Flask o Dash, Streamlit ofreció una mejor relación entre simplicidad, velocidad y presentación visual, aspectos especialmente relevantes en el contexto de un TFM.

Esta primera fase permitió confirmar la viabilidad técnica del proyecto y establecer una arquitectura limpia y coherente que sirviera de base para las siguientes etapas de desarrollo.

6.2 Fase de diseño y arquitectura

Desde el principio se diseñó una arquitectura modular, con una separación clara entre la lógica de *backend* y la interfaz. La herramienta se estructuró en diferentes carpetas:

utils/ contiene los archivos con las funciones clave: transcripción (transcripcion.py), resumen (resumen.py), *test* (test.py), chat (chat.py), configuración de claves (config.py) y subida a la nube (gcs.py).

data/uploads/ y data/resultados/ son carpetas para guardar temporalmente los archivos cargados y los resultados procesados.

En la raíz del proyecto se colocaron dos claves de acceso en formato .json (google_key.json y gemini_key.json) necesarias para autenticar las peticiones a las APIs de Google. Estas claves se integraron mediante rutas absolutas para asegurar que fueran reconocidas correctamente en cualquier entorno de ejecución.

El flujo técnico general se construyó siguiendo el paradigma ETL (*Extract, Transform, Load*). En la fase de extracción, el sistema recibe los archivos cargados, separa el audio del vídeo mediante MoviePy y pydub, y lo guarda temporalmente antes de subirlo al bucket de GCS. Posteriormente, Speech-to-Text transcribe el audio y devuelve un texto bruto que pasa por una fase de limpieza con Gemini, encargado de corregir errores, mejorar la estructura y generar un texto legible.

De forma paralela, los documentos complementarios (PowerPoint, Word, Excel y PDF) se procesan con librerías específicas (python-pptx, python-docx, openpyxl, PyPDF2), extrayendo el contenido textual de cada fuente. Todo el material se fusiona después en un único bloque de texto unificado, que sirve de base para generar el resumen, el *test* de comprensión y el chat interactivo. Para una descripción detallada de los archivos fuente implementados en la aplicación, puede consultarse el Apéndice 1.

Esta arquitectura modular, además de facilitar la depuración y la escalabilidad, permitió incorporar metadatos y métricas de rendimiento a cada fase, alineando el sistema con los principios de trazabilidad y observabilidad propios de la ingeniería de datos.

En la Ilustración 2 se representa de forma esquemática la arquitectura general de la aplicación, indicando para cada etapa las tecnologías utilizadas y los archivos que intervienen en el proceso. Este diagrama resume el recorrido que sigue el contenido desde que el usuario sube los archivos hasta que accede a los resultados generados.

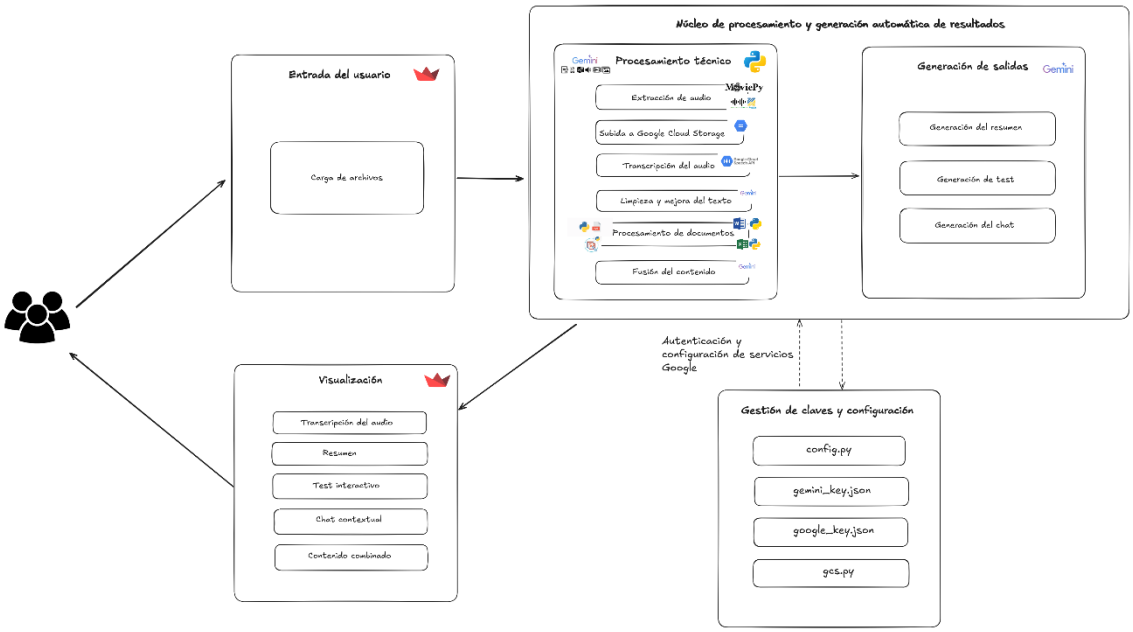

Ilustración 2. Arquitectura general del sistema y flujo técnico de funcionamiento.

6.3 Fase de desarrollo por módulos

Como se ha ilustrado previamente en la figura de arquitectura, el sistema se divide en distintos módulos que a continuación se detallan.

6.3.1 Transcripción y mejora del texto

En el módulo de transcripción (transcripcion.py), se programaron tres funciones: extraer_audio para separar el audio del vídeo y convertirlo a MP3, transcribir_audio_desde_gcs para llamar a la API de Google, y limpiar_transcripcion, que envía el texto transcrito a Gemini para mejorarlo. Esta limpieza no solo corrige errores de reconocimiento, sino que reorganiza las frases para hacer el texto más fluido y fácil de leer.

Uno de los primeros retos fue que la API no accedía correctamente al archivo subido en el *bucket*. Para solucionarlo fue necesario configurar el *bucket* manualmente, activar el acceso uniforme, habilitar las APIs necesarias y asignar el rol Storage Object Viewer al servicio que realiza la transcripción.

6.3.2 Procesamiento de documentos

El sistema admite formatos .pptx, .docx, .xlsx y .pdf, que se procesan mediante distintas librerías de Python. En PowerPoint se recorre cada *slide* para obtener el texto con python-pptx. En Word se accede a cada párrafo utilizando python-docx. En Excel se usa pandas y openpyxl para convertir todas las celdas a texto. Y para los PDF se utiliza PyPDF2, extrayendo el contenido de cada página.

Todo el contenido extraído se concatena junto con la transcripción limpia, creando un único bloque de texto que representa todo el conocimiento del material subido por el usuario.

6.3.3 Generación de resumen

En resumen.py, se utiliza Gemini para generar un resumen del bloque completo. El *prompt* está cuidadosamente diseñado para que el texto generado sea conciso pero completo. Se probaron varias versiones hasta dar con una instrucción que no perdiera información relevante ni generara frases genéricas. El modelo utilizado fue gemini-2.5-flash, que ofrece buena velocidad y coherencia en tareas de síntesis.

6.3.4 Creación del *test*

El archivo test.py genera de forma automática diez preguntas tipo *test*. Cada una tiene cuatro opciones de respuesta, claramente diferenciadas, con una única opción correcta. El *prompt* incluye ejemplos de formato para que Gemini devuelva siempre un JSON válido, con las claves pregunta, opciones (una lista de cuatro elementos) y respuesta_correcta. Inicialmente se producían errores de formato, respuestas incompletas o repetidas, pero afinando el *prompt* y validando la respuesta se solucionaron.

6.3.5 Chat contextual

El módulo de chat (chat.py) permite hacer preguntas en lenguaje natural sobre el contenido procesado. Se utiliza Gemini con un *prompt* que incluye el texto completo y una instrucción explícita de que solo se debe responder en base al material cargado. De este modo se evitan respuestas inventadas o fuera de contexto. No se usaron *embeddings* ni memoria para simplificar la implementación, pero el sistema puede ampliarse en el futuro con técnicas más avanzadas de *retrieval*.

6.3.6 Interfaz en Streamlit

La aplicación principal está en app.py. Allí se definen todos los componentes de la interfaz: subida de archivos, botón de procesamiento, indicadores de progreso y las distintas secciones de resultados (transcripción, resumen, *test*, chat y contenido combinado).

El estilo visual se personalizó mediante CSS embebido para adaptar los colores y el diseño al logotipo institucional. También se incluyó una barra de navegación sencilla con botones para cambiar de sección sin recargar la aplicación. Se gestionó el estado de la sesión con st.session_state, permitiendo que los resultados se mantuvieran visibles tras el procesamiento.

6.4 Fase de integración y flujo en la nube

Una vez validados los módulos principales, se integró la aplicación dentro del entorno Google Cloud, donde cada proceso se ejecuta de manera automática y coordinada. El flujo completo, desde la carga inicial de los archivos hasta la obtención de los resultados finales, quedó orquestado en la nube mediante Google Cloud Storage y Cloud Run.

El bucket del datalake se estructuró siguiendo las capas del modelo *Medallion Architecture*:

- raw/ para los archivos originales;
- bronze/ para los derivados inmediatos como el audio MP3;
- silver/ para los textos limpios y normalizados;
- gold/ para los artefactos finales, incluyendo los resúmenes, *tests*, métricas y el contenido combinado.

Cada artefacto se acompaña de su correspondiente archivo de metadatos (*.meta.json*), con información sobre su origen, tipo, fecha, hash y etiquetas de clasificación, garantizando la trazabilidad y el control de versiones.

El despliegue se realizó en Google Cloud Run mediante contenedores Docker generados con Cloud Build. Las credenciales y claves se gestionaron de forma segura mediante Secret Manager, lo que evitó la exposición de información sensible y permitió mantener un flujo de autenticación uniforme entre servicios.

Este modelo *serverless* proporcionó escalabilidad automática y una gestión simplificada del sistema, eliminando la necesidad de configurar servidores o balanceadores de carga manualmente.

6.5 Síntesis del desarrollo

En conjunto, el proceso de desarrollo combinó planificación técnica, iteración constante y validación práctica. Cada módulo fue concebido, probado e integrado dentro de un pipeline que sigue la lógica natural del tratamiento de datos en entornos Big Data: extraer, transformar, almacenar y analizar. La arquitectura final logró un equilibrio entre simplicidad operativa y solidez técnica, permitiendo transformar materiales heterogéneos como vídeos, presentaciones y documentos en conocimiento estructurado, navegable y reutilizable.

El resultado es una herramienta moderna, construida sobre una infraestructura *cloud* reproducible, que aplica principios de ingeniería de datos e inteligencia artificial generativa a un contexto formativo real, demostrando que es posible automatizar todo el ciclo de gestión de contenidos sin recurrir a infraestructuras complejas ni dependencias de pago.

6.6 Fase de pruebas y validación

Para validar el correcto funcionamiento del sistema se llevaron a cabo numerosas pruebas con vídeos reales de entre diez y quince minutos de duración, acompañados de presentaciones en formato PowerPoint y documentos complementarios en Word, Excel y PDF. El objetivo era comprobar el comportamiento del flujo completo en distintos escenarios y asegurar su estabilidad incluso ante datos incompletos, formatos heterogéneos o archivos con errores ortográficos y símbolos especiales.

Los resultados demostraron que el pipeline funcionaba de forma continua y sin interrupciones en todas las ejecuciones. El tiempo total de procesamiento, incluyendo la transcripción, la limpieza, el resumen y la generación de preguntas, osciló entre dos y tres minutos, lo que confirma la eficiencia del sistema. Se verificó que las transcripciones generadas por la aplicación eran más claras y coherentes que las devueltas directamente por la API de Google, gracias a la fase de mejora gestionada por Gemini, que reestructuraba las frases y corregía las inconsistencias propias del reconocimiento de voz.

Asimismo, se comprobó que los resúmenes sintetizaban correctamente las ideas principales, que las preguntas del *test* estaban bien formuladas y mantenían una única respuesta válida, y que el chat ofrecía respuestas precisas y contextualizadas sin desviarse del contenido procesado. Se revisaron los archivos generados en formato JSON para los *tests* y los resúmenes, verificando la validez de su estructura, la coherencia de las claves y la integridad de los metadatos asociados.

También se realizaron pruebas de tolerancia, ejecutando el sistema con documentos ausentes o incompletos, para confirmar que los módulos continuaban operando de manera estable y que la aplicación informaba correctamente de la ausencia de datos sin interrumpir el flujo general. Estos ensayos demostraron la robustez del diseño y la capacidad del sistema para adaptarse a distintos contextos de uso manteniendo la coherencia de los resultados.

6.7 Despliegue técnico

El despliegue de la herramienta se concibió con dos modalidades de ejecución: una local, destinada a las pruebas funcionales y a la validación del flujo, y otra en la nube, orientada a garantizar la escalabilidad y la trazabilidad del sistema en entornos reales. A continuación, se detalla la configuración utilizada en el entorno local, la organización interna del proyecto y las consideraciones técnicas necesarias para su correcta ejecución.

6.7.1 Entorno local de ejecución

La herramienta se ejecuta de forma local en un entorno Windows. No se ha utilizado Docker ni servicios en la nube para el despliegue de la interfaz, ya que Streamlit permite ejecutar la aplicación directamente mediante el comando *streamlit run app.py*. Este enfoque simplifica las pruebas y el uso individual sin necesidad de infraestructura adicional.

6.7.2 Organización del proyecto

El proyecto está organizado en una estructura de carpetas clara y modular. En la raíz se encuentran los archivos principales app.py, google_key.json y gemini_key.json. Estos dos últimos contienen las credenciales de acceso a los servicios de Google Cloud necesarios para utilizar las APIs de transcripción y de generación con Gemini.

La carpeta data contiene dos subcarpetas:

uploads: donde se almacenan los archivos subidos por el usuario (vídeos, PowerPoint, etc.).

resultados: donde se guardan los resultados generados, como los archivos de audio temporales (audio_temp.flac, audio_temp.mp3).

La carpeta ffmpeg incluye los ejecutables ffmpeg.exe, ffplay.exe y ffprobe.exe en el subdirectorio bin, necesarios para la extracción del audio del vídeo original. No se realiza una instalación global de FFmpeg, sino que se accede directamente a los ejecutables locales desde el script de transcripción.

En la carpeta utils se encuentran los diferentes módulos funcionales del sistema:

- chat.py: gestiona el flujo de preguntas y respuestas del *chatbot* contextualizado.
- config.py: contiene funciones auxiliares para cargar claves y configurar el entorno.
- gcs.py: gestiona la subida de archivos al *bucket* de Google Cloud Storage.
- resumen.py: genera el resumen automático a partir del contenido transcrito y los documentos.
- test.py: construye el *test* de comprensión con cuatro respuestas por pregunta y evalúa las respuestas del usuario.
- transcripcion.py: ejecuta la conversión del vídeo a audio y la transcripción mediante la API de Speech-to-Text.

Todos estos módulos son importados en el archivo app.py, que funciona como punto de entrada de la aplicación. Desde aquí se gestiona la interfaz de usuario en Streamlit, la navegación por secciones, el procesamiento de los archivos y la visualización de los resultados.

6.7.3 Consideraciones de ejecución

Para que la aplicación funcione correctamente, es necesario tener conexión a *internet* y contar con una cuenta de Google Cloud con facturación activa. Las claves JSON deben estar ubicadas en la raíz del proyecto y los permisos del *bucket* configurados para permitir el acceso a los servicios de Speech-to-Text y Gemini. La ejecución se realiza en local, pero el procesamiento del audio y la generación del contenido dependen de servicios externos.

6.7.4 Entorno cloud de ejecución

Además de su ejecución local, la herramienta fue desplegada y validada en un entorno *cloud* completamente gestionado mediante Google Cloud Run, lo que permitió comprobar su comportamiento en condiciones reales de escalabilidad y garantizar la trazabilidad completa del pipeline. Esta modalidad de ejecución en la nube aseguró que todas las fases del flujo, desde la carga de archivos hasta la generación de los resultados finales, se desarrollaran de forma automatizada y bajo una misma infraestructura orquestada.

Para su despliegue, el proyecto se empaquetó en un contenedor Docker, configurado con un archivo Dockerfile que define la instalación de dependencias, las librerías de Python necesarias y la ejecución de la aplicación en Streamlit dentro del puerto asignado por el entorno. La imagen generada se subió posteriormente a Google Container Registry y fue desplegada con Google Cloud Build, lo que permitió automatizar el proceso y mantener un historial de versiones.

El servicio se configuró para ejecutarse en la región europea (europe-southwest1), con la opción allow-unauthenticated activada para permitir el acceso público al entorno de demostración. La gestión de credenciales se realizó a través de Google Secret Manager, donde se almacenaron las claves google_key.json y gemini_key.json, necesarias para acceder de forma segura a las APIs de Speech-to-Text y Gemini, evitando incluirlas directamente en el contenedor.

La interacción entre servicios Cloud Run, Cloud Storage y las APIs de Google se mantuvo dentro del mismo ecosistema, lo que simplificó la autenticación y redujo la latencia de comunicación. Durante las pruebas se comprobó que el sistema mantenía una respuesta fluida incluso con cargas simultáneas y que los resultados eran almacenados correctamente en el bucket del datalake, respetando la estructura de carpetas definida para las capas *raw, bronze, silver y gold*.

El entorno *cloud* demostró ofrecer un equilibrio óptimo entre rendimiento y mantenimiento. La naturaleza *serverless* de Cloud Run permitió escalar automáticamente los recursos según la demanda, garantizando que la aplicación permaneciera operativa sin necesidad de administración manual de servidores ni supervisión continua. Además, la trazabilidad del pipeline quedó reforzada gracias al registro automático de métricas y a la persistencia de logs en el panel de operaciones de Google Cloud.

En conjunto, este despliegue en la nube consolidó el proyecto como una solución plenamente funcional dentro del ecosistema Google, combinando una interfaz sencilla de uso con una infraestructura robusta, segura y capaz de gestionar datos multimedia de manera eficiente bajo los principios de la arquitectura Big Data.

En la Ilustración 3 se muestra la arquitectura final del sistema desplegado en la nube mediante Google Cloud Run. El diagrama ilustra la relación entre la aplicación ejecutada en Streamlit, los servicios utilizados, Gemini y Speech-to-Text para el procesamiento del contenido y Google Cloud Storage como datalake, así como la conexión con Secret Manager, encargado de almacenar de forma segura las credenciales necesarias para el acceso a las APIs.

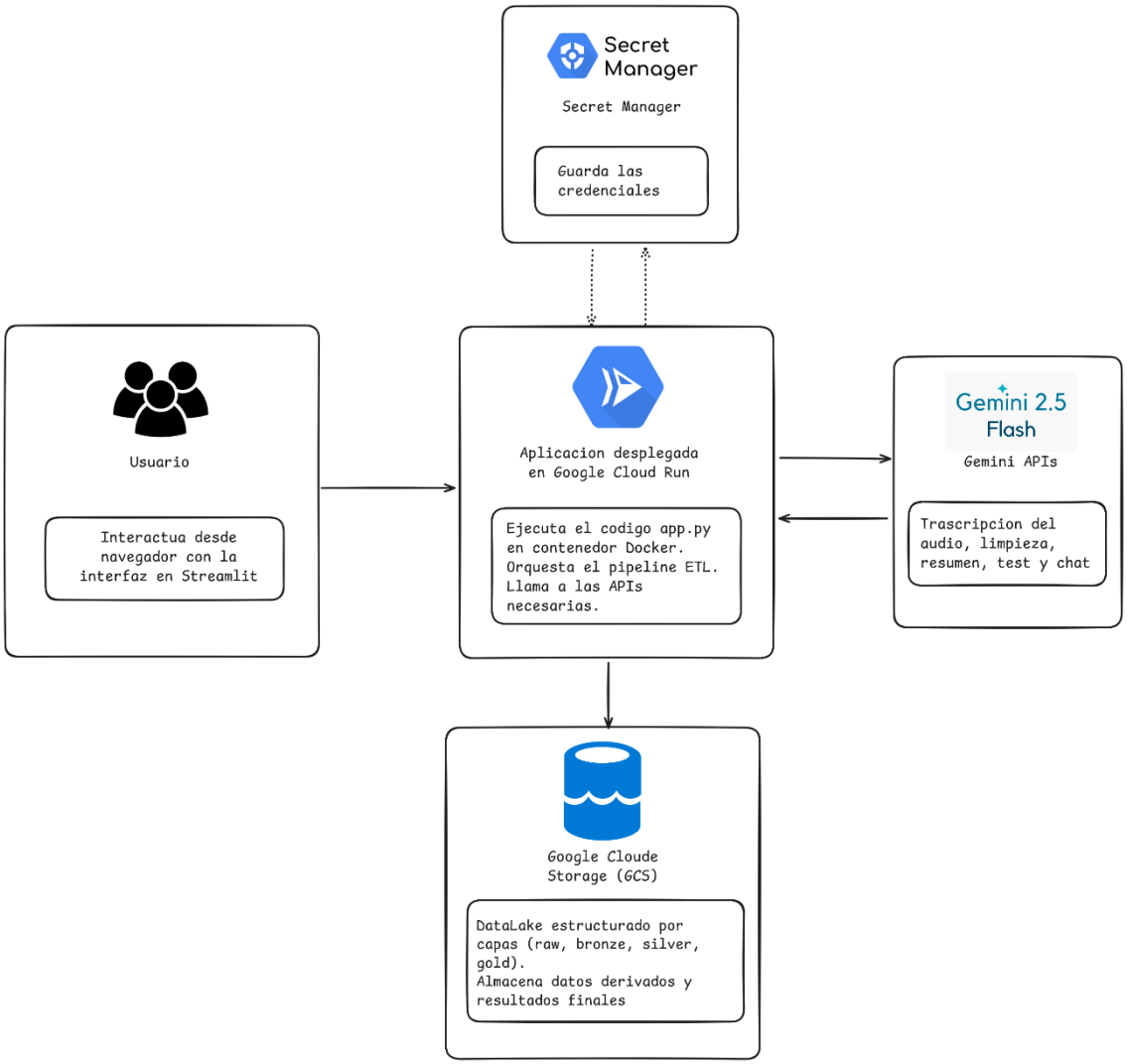

Ilustración 3. Arquitectura Cloud

6.8 Guia de uso de la herramienta

Este apartado explica de forma práctica cómo funciona la aplicación desarrollada. Se describe el flujo completo desde la carga de archivos hasta la visualización de los resultados. Las capturas de pantalla ilustran cada paso para que cualquier usuario pueda entender fácilmente su funcionamiento.

6.8.1 Vista general de la interfaz

Al iniciar la aplicación se muestra una pantalla principal sencilla e intuitiva. En la parte superior aparece el encabezado con el nombre de la herramienta y el logotipo institucional. Debajo se encuentran los botones para subir los archivos y procesarlos tal como se aprecia en la Ilustración 4.

Ilustración 4. Vista general de la interfaz

6.9 Carga de archivos

El usuario debe subir al menos un archivo de vídeo en formato MP4 y una presentación en PowerPoint. También puede añadir de forma opcional documentos Word, hojas de cálculo Excel y archivos PDF que complementen el contenido. Una vez cargados, basta con pulsar el botón **Procesar** para que la herramienta inicie automáticamente el flujo de análisis como se muestra en la Ilustración 5.

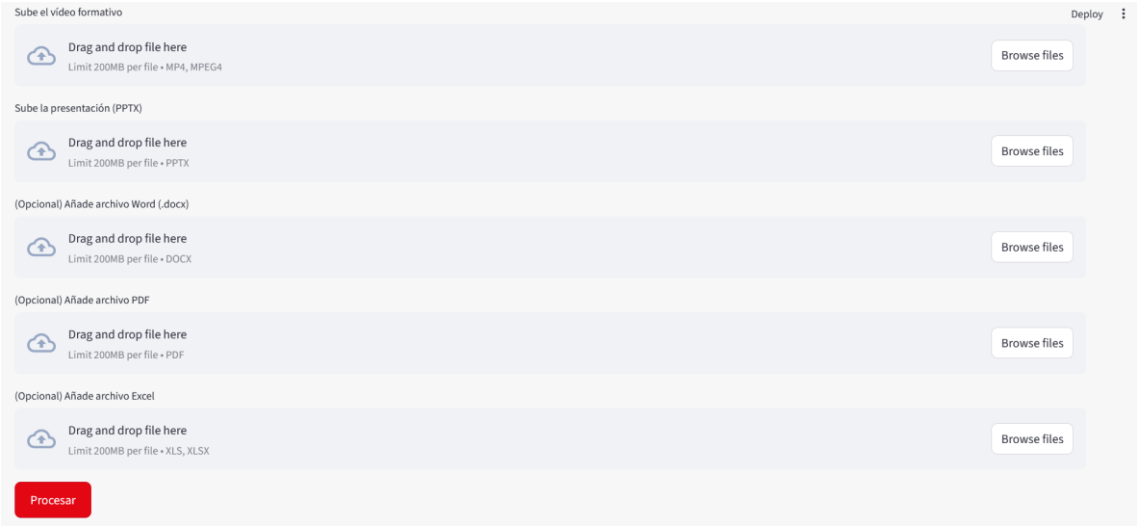

Ilustración 5. Carga de archivos

6.10 Procesamiento del contenido

Cuando se pulsa Procesar, la aplicación extrae el audio del vídeo, lo transcribe y mejora la legibilidad del texto. Al mismo tiempo analiza las diapositivas y documentos complementarios. Todo el contenido se fusiona en un único bloque de texto que servirá como base para generar el resumen, las preguntas tipo *test* y alimentar el chat.

Durante este proceso aparece un indicador de progreso como se aprecia en la Ilustración 6 y una barra de carga para que el usuario sepa que la aplicación está trabajando.

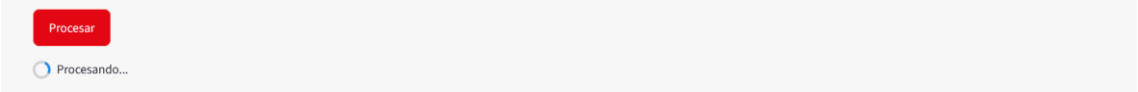

Ilustración 6. Procesamiento del contenido

6.11Selección de secciones

Una vez procesado el material se presentan diferentes secciones que permiten acceder a los resultados generados tal como se muestra en la Ilustración 7. Estas secciones incluyen la transcripción del vídeo, el resumen, el *test* de evaluación, el chat con el contenido y el bloque con toda la información combinada.

Ilustración 7. Selección de secciones

6.12 Visualización de la transcripción

Una vez finalizado el análisis se puede acceder a la sección Transcripción donde se muestra el texto procesado y optimizado a partir del audio del vídeo como se aprecia en la Ilustración 8. Esta transcripción es mucho más clara que la generada inicialmente por la API ya que ha pasado por una capa de corrección con Gemini.

Ilustración 8. Visualización de la transcripción

6.13 Consulta del resumen

En la sección Resumen se presenta un texto condensado con las ideas principales del material como se muestra en la Ilustración 9. El resumen combina la información del vídeo con las diapositivas y documentos ofreciendo una visión global sin necesidad de revisar todo el contenido original.

Ilustración 9. Consulta del resumen

6.14Evaluación de la calidad del resumen generado

Además de la generación automática de resúmenes mediante Gemini, la aplicación incorpora una funcionalidad que permite evaluar la calidad del texto sintetizado de forma cualitativa. Esta función añade un componente de validación manual dentro del flujo de análisis, permitiendo al usuario contrastar el resultado generado por la IA con su propia interpretación del contenido.

En la pestaña Resumen, tras mostrar el texto sintetizado, el usuario puede introducir su propio resumen humano de referencia mediante un campo de texto y asignar puntuaciones entre 1 y 5 a distintos criterios de evaluación: coherencia, cobertura del contenido, precisión y redundancia. Adicionalmente, se incluye un espacio para comentarios abiertos que permite registrar observaciones o sugerencias sobre la calidad general del resumen.

Cuando se pulsa el botón Guardar evaluación, la aplicación crea un archivo en formato JSON, donde se almacenan de forma estructurada las puntuaciones, el texto de referencia, los comentarios, la fecha de la evaluación y los metadatos asociados. Este archivo se guarda automáticamente en el datalake dentro de la carpeta gold/evaluation, siguiendo el mismo formato y política de versionado utilizados para el resto de artefactos.

Con esta incorporación, el sistema añade una capa de análisis cualitativo que permite comparar la producción automática con juicios humanos y sentar las bases para futuras métricas objetivas

de calidad. Esta estrategia amplía el alcance del proyecto hacia un enfoque más evaluativo, integrando el criterio humano como parte del proceso analítico.

En la Ilustración 10 se muestra la interfaz de la aplicación en la sección dedicada al resumen humano de referencia. En este espacio, el usuario puede introducir un texto elaborado manualmente que sirve como punto de comparación con el resumen generado automáticamente por Gemini. Esta entrada constituye la base de la evaluación cualitativa y permite registrar diferencias en estructura, estilo o nivel de cobertura del contenido.

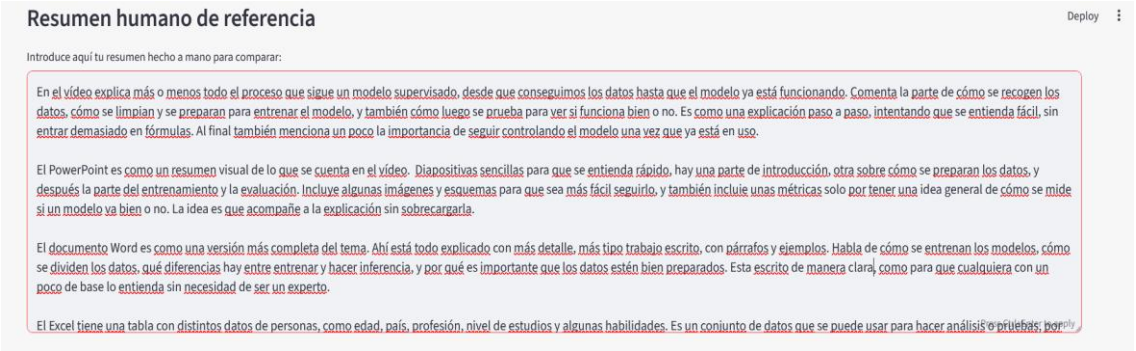

Ilustración 10. Interfaz para introducir el resumen humano de referencia.

La Ilustración 11 muestra la interfaz de evaluación, donde el usuario puede puntuar el resumen automático en distintos criterios de calidad. Las puntuaciones se realizan mediante barras deslizantes, complementadas con un campo de comentarios que permite aportar observaciones adicionales sobre la claridad, la fidelidad o la utilidad del texto generado.

Ilustración 11. Evaluación de calidad del resumen generado mediante criterios de coherencia, cobertura, precisión y redundancia.

6.15 Evaluación con preguntas tipo *test*

La sección *Test* permite evaluar la comprensión del contenido. Se generan automáticamente diez preguntas de opción múltiple con su respuesta correcta. El usuario puede contestar y validar sus respuestas viendo al final cuántos aciertos obtuvo como se observa en las Ilustraciones 10 y 11.

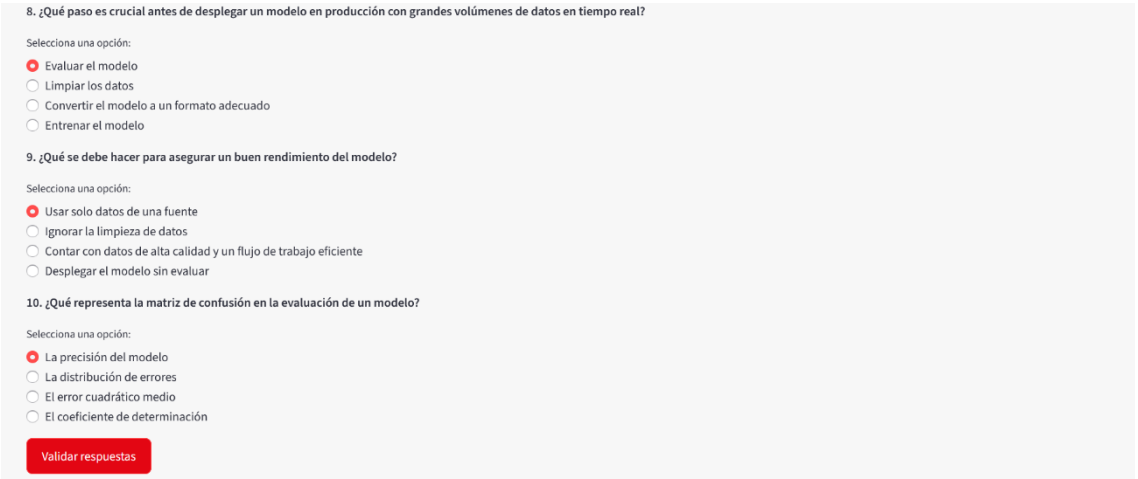

Ilustración 12. Preguntas tipo test

Ilustración 13. Evaluacion preguntas tipo test

6.16 Interacción mediante chat contextualizado

Por último en la sección Chat el usuario puede escribir cualquier pregunta relacionada con el material procesado. El sistema responde exclusivamente con la información analizada evitando respuestas genéricas o irrelevantes como se aprecia en la Ilustración 12. Esta función es útil para buscar conceptos específicos sin tener que leer toda la transcripción o el resumen.

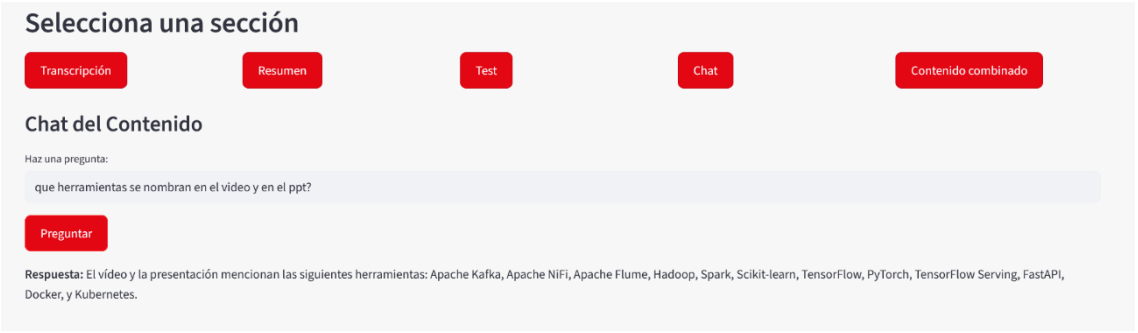

Ilustración 14. Interacción mediante chat contextualizado

6.17 Contenido combinado

Esta sección muestra todo el contenido unificado en un solo bloque transcripción mejorada contenido de la presentación y adjuntos como se muestra en la Ilustración 13. Sirve como referencia completa para repasar o volver a consultar los conceptos tratados.

Ilustración 15. Contenido combinado

Capítulo 7. RESULTADOS Y DISCUSION

En este capítulo se presentan los resultados alcanzados tras el desarrollo e implementación de la herramienta, así como las principales observaciones derivadas del proceso técnico y de las pruebas funcionales. Se analiza cómo el sistema logra procesar automáticamente vídeos y documentos, generando transcripciones claras, resúmenes sintéticos, preguntas tipo *test* y un chat contextualizado que permite interactuar con el contenido de manera directa. Además, se comentan los desafíos encontrados durante la implementación, como la configuración del entorno *cloud*, los límites de uso de las APIs de Google o los ajustes necesarios en la gestión de los modelos generativos. Finalmente, se discute el valor añadido de la solución frente a otras alternativas existentes y se reflexiona sobre su potencial en contextos educativos y profesionales.

7.1 Resultados

El desarrollo de la herramienta permitió unificar en un único flujo automatizado todas las tareas relacionadas con el análisis de presentaciones técnicas, integrando en una misma arquitectura la extracción, el procesamiento y la generación de conocimiento. Cuando el usuario sube un vídeo, el sistema extrae de manera local la pista de audio mediante las librerías MoviePy y pydub, y ese archivo temporal se envía automáticamente a un bucket en Google Cloud Storage, donde se activa el servicio de Speech-to-Text para generar una primera transcripción del contenido hablado.

En algunos casos, las transcripciones iniciales resultaban poco precisas debido a ruidos de fondo, solapamientos de voz o pronunciaciones rápidas. Para resolverlo, se añadió una segunda capa de procesamiento con Gemini 2.5 Flash, accedida directamente a través de su API oficial, encargada de limpiar el texto, corregir errores de reconocimiento, reorganizar las frases y mejorar la legibilidad general. Este paso supuso una mejora sustancial en la calidad del resultado, obteniéndose textos mucho más coherentes, claros y fieles al contenido original del vídeo.

Paralelamente, el sistema procesa los documentos complementarios asociados a cada presentación. De los archivos PowerPoint se extrae el texto de cada diapositiva; de los documentos Word, el contenido de todos los párrafos; de las hojas Excel, los datos de las celdas concatenadas; y de los archivos PDF, las páginas completas procesadas con PyPDF2. Toda esta información se integra en un bloque textual unificado, que constituye la base del resto de funcionalidades.

A partir de este texto consolidado, el sistema genera un resumen automático que combina la información procedente del vídeo con la de los documentos escritos, ofreciendo una visión global y estructurada del contenido. Con el mismo material se construyen de forma automática diez preguntas tipo *test*, diseñadas para evaluar la comprensión del tema tratado, validando las respuestas del usuario y mostrando un resumen de aciertos y errores. Finalmente, se habilita un

chat interactivo, que permite formular preguntas en lenguaje natural y obtener respuestas precisas basadas exclusivamente en el contenido procesado, sin recurrir a información externa.

Durante las pruebas realizadas con vídeos de entre diez y quince minutos de duración y presentaciones de tamaño medio, el flujo completo, desde la subida de los archivos hasta la generación de los resultados finales, se completó en un intervalo de entre dos y tres minutos, lo que demuestra la eficiencia del pipeline en términos de rendimiento y estabilidad. Los resúmenes generados consiguieron condensar los conceptos principales con claridad, los *test* ofrecieron preguntas coherentes y equilibradas, y el chat mostró respuestas precisas, siempre centradas en el material cargado por el usuario.

En paralelo, se registraron las métricas de rendimiento de cada ejecución, guardadas en el datalake dentro de la capa gold/metrics, en formato JSON y CSV. Estos registros permiten analizar la duración de cada fase del proceso (extracción, transcripción, limpieza, resumen, *test* y tiempo total), lo que constituye una primera aproximación a la observabilidad del sistema.

En la Ilustración 14 se muestra un ejemplo de las métricas de rendimiento obtenidas durante la ejecución de una prueba real del sistema. En esta vista, la aplicación presenta el tiempo total y el tiempo parcial de cada una de las fases del pipeline, extracción del audio, transcripción, limpieza, resumen y generación del *test*, así como la confirmación de que los resultados se almacenan correctamente en el datalake.

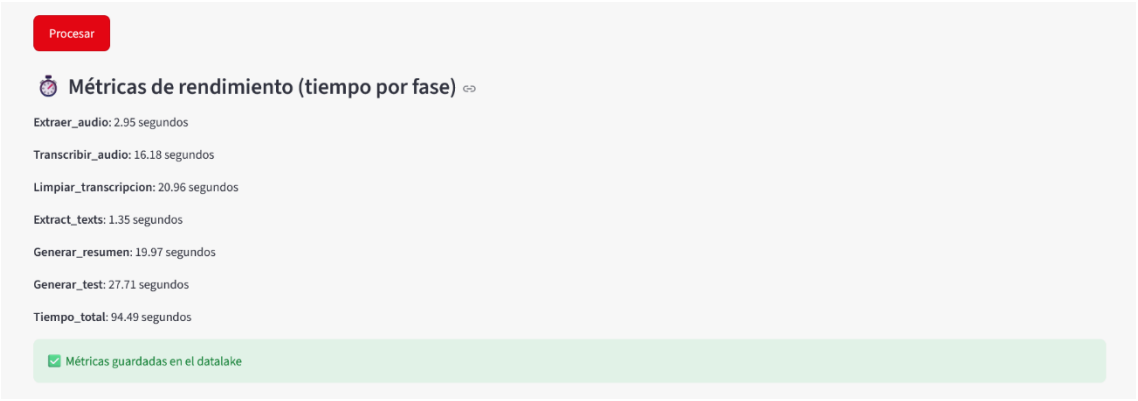

Ilustración 16. Ejemplo de métricas de rendimiento generadas por la aplicación (tiempo por fase)

Este registro automático permite analizar la eficiencia del flujo ETL y sirve como base para futuras optimizaciones en el rendimiento y la escalabilidad del sistema.

En conjunto, los resultados obtenidos confirman que la herramienta cumple su propósito de automatizar la transformación de contenido audiovisual y documental en conocimiento estructurado, aplicando de forma efectiva los principios de la ingeniería de datos en un entorno *cloud* accesible, escalable y con costes mínimos.

7.1.1 Métricas de rendimiento y comparación histórica

Junto al registro básico de tiempos de ejecución, el sistema implementa una funcionalidad de análisis comparativo que permite observar el comportamiento del pipeline frente a la media de ejecuciones anteriores. Cada vez que se completa un procesamiento, los tiempos parciales de las distintas fases, extracción del audio, transcripción, limpieza, resumen, generación del test y tiempo total, se almacenan como registros independientes en el datalake, dentro de la carpeta gold/metrics, tanto en formato JSON como CSV.

Como se puede apreciar en la ilustración 17, en la interfaz de Streamlit, las métricas de la ejecución actual se presentan junto con un gráfico de comparación que muestra, mediante barras horizontales, el tiempo actual (en color rojo) frente a la media histórica (en color gris). Para generar esta visualización, la aplicación se conecta al datalake, recupera todas las ejecuciones previas, calcula la media por fase con la librería pandas y genera el gráfico con matplotlib.

Este mecanismo permite evaluar de manera visual la eficiencia de cada proceso y detectar posibles desviaciones o mejoras entre ejecuciones sucesivas. A medida que se acumulan nuevas pruebas, el sistema enriquece automáticamente su base de comparación, dotando al entorno de un componente de observabilidad propio de las arquitecturas Big Data.

La posibilidad de visualizar la evolución temporal del rendimiento aporta una dimensión analítica adicional al proyecto, permitiendo no solo procesar información, sino también medir, auditar y mejorar el comportamiento del flujo ETL a lo largo del tiempo. Este enfoque refuerza la madurez técnica del sistema, alineándolo con los principios de monitorización y control de calidad del dato característicos de los entornos *cloud* profesionales.

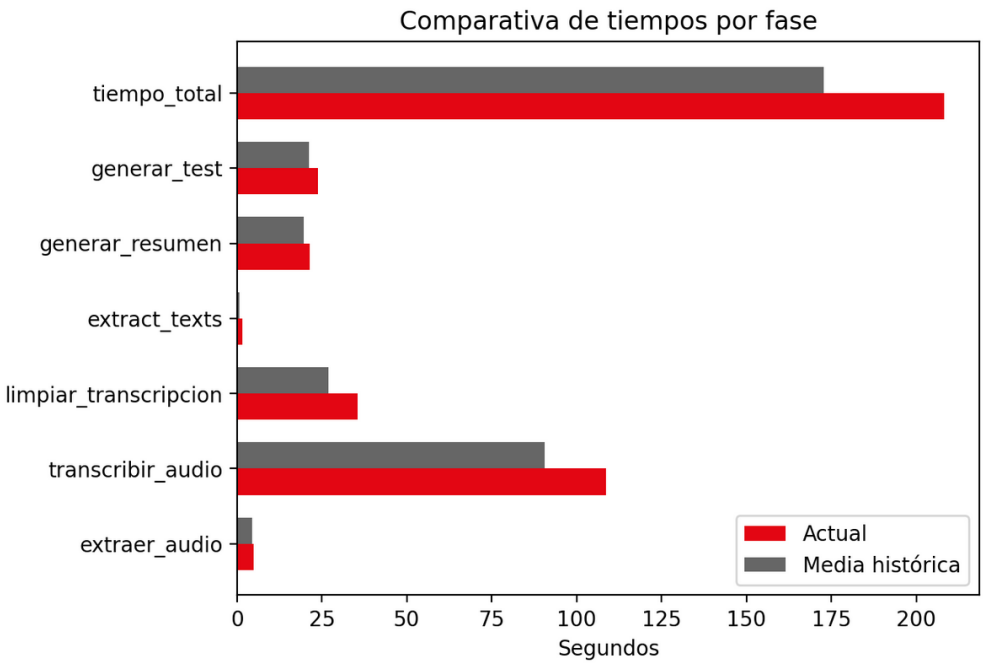

Ilustración 17. Comparativa de metricas. Tiempo por fase.

7.2 Discusión

Para que la aplicación funcionara de forma estable y reproducible no bastó con desarrollar la lógica interna de cada módulo; fue necesario comprender y configurar correctamente todo el ecosistema de Google Cloud en el que se apoya la solución. Desde las primeras pruebas se evidenció que el éxito del sistema dependía tanto del código como de la infraestructura que lo sostenía.

En un principio se trabajó con una cuenta gratuita, pero los límites de uso de Gemini eran demasiado restrictivos, ya que apenas permitían unas pocas solicitudes antes de suspender temporalmente el servicio. Esta limitación hacía inviable un proceso iterativo de validación. Por ese motivo se creó una cuenta completa de Google Cloud con facturación habilitada, lo que permitió desbloquear los límites y trabajar sin interrupciones, manteniendo siempre el control sobre las credenciales, las métricas de uso y los costes generados.

Dentro de la consola de Google Cloud se activaron tres servicios fundamentales: Speech-to-Text, responsable de la transcripción del audio; Cloud Storage, empleado para la gestión de archivos y el almacenamiento persistente; y la API de Gemini, utilizada directamente a través de la librería oficial google-generativeai, para la optimización de transcripciones, la generación de resúmenes y la creación de preguntas tipo *test*. Cada uno de estos servicios requirió permisos específicos: por ejemplo, Speech-to-Text necesitó acceso de lectura al bucket del datalake mediante el rol roles/storage.objectViewer, mientras que la aplicación ejecutaba las operaciones de escritura con una cuenta de servicio dotada del rol roles/storage.objectAdmin.

Durante la configuración inicial surgieron varios problemas técnicos que obligaron a profundizar en la gestión de permisos y dependencias. En las primeras ejecuciones aparecieron errores como AxiosError: 500, provocados por la creación del bucket en una región incorrecta o por la ausencia de políticas de acceso uniforme. Estos inconvenientes se solucionaron recreando manualmente el bucket desde la consola de Google Cloud, estableciendo la región adecuada, activando el acceso uniforme y habilitando las APIs necesarias.

Uno de los puntos más relevantes fue la implementación del datalake en Google Cloud Storage, concebido siguiendo el modelo *Medallion Architecture*. Se estructuró en cuatro zonas: *raw*, para los archivos originales; *bronze*, para los derivados mínimos como los audios extraídos; *silver*, para los textos limpios y normalizados; y *gold*, para los artefactos finales como los resúmenes, *tests*, métricas y contenido combinado. Cada archivo generado en el proceso se acompaña de un fichero .meta.json con metadatos estructurados (identificador, tipo de artefacto, fecha, hash, etiquetas y relaciones con otros objetos), lo que garantiza la trazabilidad y versionado de los datos. Este esquema resultó clave para dotar al sistema de orden, coherencia y capacidad de auditoría, ya que permite rastrear el origen y evolución de cada elemento dentro del flujo ETL.

La creación de este datalake no fue un mero ejercicio de almacenamiento, sino una decisión estratégica: permitió gestionar el dato como un activo, separando las fases del procesamiento y conservando cada etapa de transformación. Gracias a esta organización, cualquier ejecución puede reproducirse o verificarse sin pérdida de información. Esta filosofía, muy propia de los entornos Big Data, dotó al proyecto de una base sólida, escalable y extensible a futuros

desarrollos más complejos, como la integración de catálogos de datos o la visualización analítica de logs.

Otro componente decisivo fue el despliegue en la nube mediante Google Cloud Run, que permitió trasladar la aplicación a un entorno *serverless* totalmente gestionado. La herramienta se empaquetó en un contenedor Docker, configurado con todas las dependencias necesarias y desplegado mediante Cloud Build, lo que garantizó la reproducibilidad y facilitó las actualizaciones. Las claves de acceso se gestionaron con Secret Manager, evitando incluir información sensible en el código y fortaleciendo la seguridad del sistema.

El entorno Cloud Run demostró ser especialmente eficaz para este tipo de aplicación: ofreció escalabilidad automática, tiempos de arranque reducidos y una comunicación directa con los servicios de Google Cloud, sin necesidad de configurar servidores ni mantener instancias activas. Durante las pruebas, el sistema mostró una latencia mínima en la interacción con las APIs y un consumo eficiente de recursos. Además, los logs centralizados y las métricas de rendimiento almacenadas en la capa gold/metrics del datalake permitieron monitorizar cada ejecución de forma precisa, añadiendo un nivel de observabilidad que no estaba previsto en las primeras versiones del proyecto.

Incluso con Gemini fue necesario realizar varios ajustes hasta lograr un comportamiento estable. En los primeros intentos se utilizó la versión pro, pero su latencia era mayor y su coste más elevado de lo necesario. El cambio a Gemini 2.5 Flash mejoró el rendimiento general, redujo los tiempos de respuesta y permitió mantener la calidad en las tareas de corrección y generación de resúmenes. También se afinaron los prompts para obtener resultados más naturales: cuando las instrucciones eran demasiado generales, el modelo tendía a devolver textos repetitivos, por lo que se ajustaron para que priorizara la claridad, la coherencia y el respeto al sentido original.

Respecto a la interfaz, la elección de Streamlit fue especialmente acertada. Aunque era una herramienta nueva en este trabajo, su sencillez de uso y su capacidad para integrar distintas secciones, transcripción, resumen, *test* y chat, en una misma aplicación la convirtieron en una solución eficiente y visualmente clara. Frente a alternativas como Dash o Flask, Streamlit requería menos configuración y ofrecía una curva de aprendizaje más rápida, lo que permitió centrar los esfuerzos en el flujo de datos y no en la capa visual.

Durante la integración del sistema también se valoró el uso de otras APIs de inteligencia artificial como OpenAI o Azure Cognitive Services, pero se decidió mantener todo el flujo dentro del ecosistema Google para evitar problemas de autenticación y simplificar el manejo de credenciales. El hecho de que Speech-to-Text, Gemini y Cloud Storage compartan el mismo sistema de permisos y autenticación simplificó enormemente la orquestación del pipeline.

Otro aspecto fundamental fue la gestión de dependencias necesarias para procesar los diferentes tipos de archivo. Librerías como python-pptx, python-docx, PyPDF2 u openpyxl resultaron imprescindibles para leer correctamente los formatos de presentación, texto, PDF y hoja de cálculo. Sin ellas, parte de los módulos no hubiera funcionado o habrían devuelto resultados incompletos. Este proceso evidenció que la fiabilidad del sistema dependía tanto del código como del entorno en el que se ejecutaba.

En definitiva, esta fase dejó claro que un proyecto de este tipo no consiste solo en conectar APIs, sino en diseñar un ecosistema completo donde cada servicio, librería y proceso esté configurado y coordinado para funcionar como un todo. Configurar Google Cloud, habilitar las APIs, asignar roles y permisos en el bucket, definir la estructura del datalake, contenerizar la aplicación para Cloud Run, optimizar los prompts de Gemini y gestionar las dependencias requirió tiempo y pruebas, pero permitió construir una solución robusta, escalable y totalmente operativa en la nube.

Esta integración de Big Data, IA generativa y arquitectura *cloud* se alinea con la visión de una inteligencia artificial aplicada a la educación como herramienta de personalización y mejora del aprendizaje (Luckin et al., 2016), reforzando la relevancia de emplear sistemas de análisis multimodal y chatbots en entornos formativos. Hasta el momento de redacción de este trabajo no se identificaron soluciones que integren en un único flujo la transcripción de vídeo, la lectura de documentos, la generación de resúmenes, la creación de *test* interactivos y un chat basado exclusivamente en el contenido procesado, lo que confirma el carácter innovador y diferencial de esta propuesta.

7.3 Valor innovador

La herramienta desarrollada aporta un enfoque claramente diferenciador frente a las soluciones actualmente disponibles en el mercado. No se limita a transcribir o resumir vídeos de manera aislada, sino que integra en un único flujo automatizado el análisis conjunto de múltiples formatos, combinando el audio de las presentaciones con las diapositivas, documentos y otros materiales de apoyo. Esta integración multimodal permite generar resultados más ricos, coherentes y útiles que los obtenidos por aplicaciones centradas en una sola tarea, ya que el conocimiento se construye a partir de la relación entre todas las fuentes y no de su procesamiento independiente.

El uso coordinado de Speech-to-Text y Gemini, dentro de una arquitectura de datos orquestada en la nube, añade un nivel de optimización que mejora de forma significativa la calidad de las transcripciones y la claridad de los textos finales. Gemini no solo limpia y reorganiza el contenido, sino que también genera resúmenes comprensibles, preguntas de evaluación y respuestas contextualizadas que se limitan exclusivamente al material procesado, evitando las respuestas genéricas propias de otros asistentes basados en modelos abiertos. De esta manera, el sistema logra un equilibrio entre autonomía y control, garantizando precisión en el contenido y coherencia con las fuentes originales.

Otro aspecto innovador reside en la arquitectura técnica que sustenta la herramienta. La implementación de un pipeline ETL desplegado en Google Cloud Run y respaldado por un datalake estructurado en Google Cloud Storage proporciona una infraestructura moderna y escalable, capaz de gestionar datos multimodales con trazabilidad completa. La separación por capas (*raw, bronze, silver, gold*) permite conservar cada estado del dato y analizar su evolución, siguiendo los principios de gobernanza propios de los entornos Big Data. A su vez, el modelo

serverless de Cloud Run garantiza una ejecución eficiente sin necesidad de servidores dedicados, reduciendo costes y mantenimiento.

A nivel de experiencia de usuario, la elección de Streamlit permitió construir una interfaz sencilla, intuitiva y visualmente clara, que convierte un sistema complejo en una herramienta accesible para cualquier perfil académico o profesional. La posibilidad de desplegar la aplicación con pocos recursos, manteniendo al mismo tiempo su capacidad de expansión, refuerza el valor práctico del proyecto como solución adaptable a distintos escenarios.

En comparación con las aplicaciones comerciales revisadas, el valor distintivo de esta propuesta radica precisamente en su enfoque integral. La herramienta no solo automatiza procesos aislados, sino que conecta todas las fases del análisis, ingesta, transformación, almacenamiento y generación de conocimiento, en un flujo coherente, accesible y sostenible. Está orientada tanto a entornos educativos como corporativos, donde la reutilización del conocimiento y la sistematización de la información representan un desafío constante.

En suma, la innovación de este trabajo reside en haber logrado una convergencia práctica entre inteligencia artificial generativa, ingeniería de datos y computación en la nube, transformando un conjunto de herramientas dispersas en una solución unificada que automatiza la gestión del conocimiento técnico de forma estructurada, trazable y reutilizable.

Capítulo 8. CONCLUSIONES Y FUTURAS LINEAS DE TRABAJO

Este capítulo recoge las conclusiones derivadas del desarrollo de la herramienta y plantea las posibles líneas de mejora para futuras versiones. A lo largo del trabajo se ha demostrado cómo la combinación de inteligencia artificial generativa, análisis multimodal y servicios en la nube puede automatizar tareas tradicionalmente complejas mediante un sistema estable, accesible y reproducible. El proceso de desarrollo permitió entender de manera práctica el funcionamiento de las APIs de Google, las dependencias entre servicios y los principios de diseño de arquitecturas en la nube. Surgieron obstáculos técnicos que obligaron a investigar, probar y ajustar, y cada uno de ellos contribuyó a mejorar la comprensión del ecosistema y a consolidar la robustez de la solución. Al mismo tiempo, se identificaron oportunidades de mejora relacionadas con la calidad de las transcripciones, la ampliación de formatos o la integración en plataformas educativas, que servirán como base para continuar evolucionando la herramienta.

8.1 Conclusiones del trabajo

El proyecto demostró que es posible automatizar el análisis y la evaluación de presentaciones técnicas combinando IA generativa, procesamiento multimodal y arquitectura *cloud* dentro de un flujo de trabajo ETL completo.

La elección del ecosistema Google Cloud resultó fundamental: simplificó la gestión de credenciales, permitió integrar servicios como Speech-to-Text, Gemini y Cloud Storage, y proporcionó la infraestructura necesaria para desplegar la aplicación de forma escalable en Cloud Run bajo un modelo *serverless*.

El uso de Streamlit facilitó la creación de una interfaz web funcional, intuitiva y ligera, que convirtió una arquitectura compleja en una herramienta de uso accesible y directo.

El desarrollo del datalake estructurado en capas (*raw, bronze, silver, gold*) fue una de las aportaciones más relevantes, ya que permitió gestionar los datos con trazabilidad completa y conservar cada etapa de transformación. Este diseño no solo garantizó el orden y la reproducibilidad del proceso, sino que también dotó al sistema de una base sólida para futuras ampliaciones y análisis.

El trabajo no estuvo exento de dificultades: los límites de uso con cuentas gratuitas, los errores de permisos en el bucket, los ajustes de los modelos de Gemini y las dependencias adicionales fueron desafíos que exigieron tiempo, paciencia y comprensión del ecosistema. Sin embargo, cada obstáculo se convirtió en una oportunidad para aprender a configurar entornos *cloud* reales, optimizar flujos de datos y ajustar modelos de IA generativa a las necesidades del proyecto.

El resultado final es una herramienta estable, coherente y útil, orientada a docentes, estudiantes y profesionales que necesitan acceder de manera dinámica al contenido de vídeos y

documentos. Permite resumir, evaluar y consultar información técnica sin tener que revisar manualmente cada recurso, lo que reduce el tiempo de análisis y favorece la reutilización del conocimiento.

8.2 Conclusiones personales

La realización de este trabajo ha sido una experiencia muy completa que me permitió unir la parte técnica del desarrollo con la comprensión práctica de cómo funcionan las arquitecturas *cloud* y los servicios que las sustentan. Más allá de la programación, fue necesario aprender a gestionar credenciales, permisos, configuraciones de red y límites de uso, comprendiendo que el desarrollo de una aplicación en la nube implica mucho más que escribir código: requiere orquestar correctamente los distintos componentes para que todo funcione de forma integrada.

Trabajar con el ecosistema de Google fue un aprendizaje valioso. Configurar Speech-to-Text, Gemini y Cloud Storage me permitió entender cómo se conectan los servicios dentro de un entorno de datos centralizado y cómo se despliegan aplicaciones reales mediante Cloud Run y Docker, asegurando una ejecución escalable sin servidores dedicados. También descubrí el potencial de Streamlit como herramienta para construir interfaces claras y funcionales, una opción ideal para convertir proyectos técnicos en aplicaciones presentables y comprensibles para cualquier usuario.

Uno de los aprendizajes más importantes fue trabajar con modelos de IA generativa. No se trató solo de utilizarlos, sino de entender cómo interactúan con las instrucciones y cómo pequeñas variaciones en el prompt pueden alterar por completo los resultados. Ajustar las peticiones, definir límites, estructurar respuestas y seleccionar la versión del modelo adecuada (en este caso, Gemini 2.5 Flash) fueron tareas que exigieron experimentación y criterio.

Los problemas surgidos en el camino desde errores de permisos en los buckets hasta dependencias que no se instalaban correctamente fueron también parte esencial del proceso formativo. Me enseñaron a investigar, comparar soluciones y validar hipótesis con paciencia, entendiendo que en los proyectos reales de inteligencia artificial y servicios *cloud* no todo funciona a la primera, y que cada error es una oportunidad para aprender y afinar la solución.

Esta experiencia me deja la satisfacción de haber construido una herramienta funcional, útil y replicable, pero sobre todo la sensación de haber interiorizado una forma de trabajar basada en la integración, la precisión y la mejora continua, principios fundamentales en la ingeniería de datos moderna.

8.3 Futuras líneas de trabajo

El desarrollo de esta herramienta deja abiertas diversas oportunidades de mejora y ampliación que permitirían aumentar su utilidad y alcance tanto en entornos educativos como corporativos. Durante las pruebas realizadas se identificaron limitaciones propias del prototipo, pero también

surgieron nuevas ideas que marcan un camino claro hacia una evolución más completa del sistema.

Una de las principales líneas de mejora se centra en la calidad del audio. Aunque Google Speech-to-Text ofrece resultados satisfactorios, los vídeos con ruido de fondo o con voces poco nítidas continúan generando transcripciones imprecisas. Incorporar un módulo previo de limpieza de audio, basado en técnicas de reducción de ruido o modelos especializados en separación de voz, permitiría obtener transcripciones más precisas y, en consecuencia, generar resúmenes y evaluaciones de mayor calidad.

Otra ampliación importante sería la integración de un sistema OCR (Reconocimiento Óptico de Caracteres) para procesar PDF escaneados o documentos con imágenes embebidas, que actualmente no pueden analizarse de manera efectiva. Con esta mejora, la herramienta ampliaría su capacidad para procesar materiales educativos o técnicos más heterogéneos, fusionando texto e imagen dentro del flujo multimodal de análisis.

También se plantea la posibilidad de incorporar nuevos formatos, como imágenes, infografías o grabaciones de audio independientes, con el objetivo de enriquecer el contexto de las presentaciones y potenciar el análisis multimodal. De este modo, la herramienta podría evolucionar hacia un sistema capaz de procesar no solo vídeos y documentos tradicionales, sino también material visual complementario que aporte una visión más completa de los contenidos.

En cuanto a la interfaz, una línea prometedora sería desarrollar una versión avanzada con despliegue profesional, integrada directamente en plataformas de e-learning como Moodle o Google Classroom. Esto permitiría que los docentes subieran sus clases, obtuvieran resúmenes automáticos, *tests* y un chat contextual, todo ello accesible desde el mismo entorno educativo. Con esta integración, el sistema dejaría de ser un prototipo independiente para convertirse en una herramienta útil de apoyo docente y evaluación continua.

Otra dirección de desarrollo sería la optimización del motor de generación de preguntas, incorporando algoritmos que clasifiquen las preguntas por nivel de dificultad, incluyan cuestiones abiertas o de desarrollo y permitan personalizar los cuestionarios en función del usuario. Este avance dotaría al sistema de una dimensión evaluativa más completa, orientada tanto a la autoevaluación como a la evaluación formativa.

Desde el punto de vista técnico, el sistema ya se encuentra completamente contenedorizado en Docker y desplegado en la nube mediante Google Cloud Run, con un proceso de construcción y actualización gestionado por Cloud Build. Esta configuración permite que el entorno sea fácilmente reproducible y escalable, facilitando su distribución o integración en distintos contextos sin necesidad de configuraciones manuales.

Como líneas de evolución, podría incorporarse un sistema de monitorización avanzada del pipeline ETL, con paneles de control de métricas y alertas integradas a través de Google Cloud Monitoring, lo que reforzaría la observabilidad y el seguimiento del rendimiento en tiempo real.

Asimismo, se plantea la posibilidad de ampliar las capacidades del datalake, añadiendo políticas de versionado automático, almacenamiento jerárquico por proyecto o usuario y mecanismos de

auditoría que permitan estudiar la evolución histórica de los datos procesados. Estas mejoras contribuirían a fortalecer la arquitectura Big Data ya existente, orientándola no solo al procesamiento inmediato de resultados, sino también a la gestión y gobernanza completa del conocimiento generado.

Finalmente, una línea de investigación particularmente interesante se relaciona con la evolución del sistema de evaluación de resúmenes. En la versión actual, el proceso es manual y permite registrar valoraciones humanas sobre la coherencia, la precisión o la cobertura del contenido. Sin embargo, esta base podría ampliarse hacia un enfoque semi-automatizado, integrando métricas objetivas utilizadas en el ámbito del procesamiento del lenguaje natural, como BLEU, ROUGE o METEOR, que comparan automáticamente el resumen generado con una referencia humana.

La incorporación de este tipo de indicadores permitiría cuantificar la calidad textual de los resúmenes con mayor rigor, establecer comparaciones sistemáticas entre modelos y enriquecer el conjunto de métricas almacenadas en el datalake. De este modo, el sistema avanzaría hacia un marco de evaluación híbrido, en el que la percepción humana y el análisis automático se complementan para ofrecer una visión más completa del rendimiento del modelo generativo.

En conjunto, todas estas líneas de trabajo permitirían que la herramienta evolucionara desde un prototipo funcional hacia una solución integral, capaz de automatizar no solo la transcripción y el análisis de presentaciones, sino también la gestión, evaluación y monitorización del conocimiento. El objetivo final sería consolidar el sistema como una plataforma Big Data de análisis multimodal e inteligencia educativa, donde los datos de diferentes fuentes converjan en un entorno *cloud* escalable, trazable y diseñado para transformar información dispersa en conocimiento estructurado y reutilizable.

REFERENCIAS

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. Springer.

Burns, B., Grant, B., Oppenheimer, D., Brewer, E., & Wilkes, J. (2016). Borg, Omega, and Kubernetes. *Communications of the ACM, 59*(5), 50–57.

Chollet, F. (2021). *Deep Learning with Python* (2nd ed.). Manning.

Databricks. (2021). *The Medallion Architecture*. Databricks Documentation. https://docs.databricks.com/en/lakehouse/medallion.html

Denny, P., Gulwani, S., Heffernan, N. T., Käser, T., Moore, S., Rafferty, A. N., & Singla, A. (2024). Generative AI for education (GAIED): Advances, opportunities, and challenges. *arXiv preprint arXiv:2402.01580*. https://doi.org/10.48550/arXiv.2402.01580

EaseMate AI. (s.f.). *Chat with PPT and PDF*. https://www.easemate.ai

Eightify. (s.f.). *AI Summaries for YouTube Videos*. https://eightify.app

Fireflies.ai. (s.f.). *AI Notetaker for Meetings*. https://fireflies.ai

Google. (2024). *Gemini API overview*. https://ai.google.dev/gemini-api/docs

Google Cloud. (2024). *Speech-to-Text documentation*. https://cloud.google.com/speech-to-text/docs

Google Cloud. (2024). *Cloud Monitoring overview.* https://cloud.google.com/monitoring

Guo, P. J., Kim, J., & Rubin, R. (2014). How video production affects student engagement: An empirical study of MOOC videos. *Proceedings of the First ACM Conference on Learning at Scale*, 41–50. https://doi.org/10.1145/2556325.2566239

Jurafsky, D., & Martin, J. H. (2023). *Speech and Language Processing* (3rd ed.). Pearson.

Kohler, S., & Dietrich, T. C. (2021). Potentials and limitations of educational videos on YouTube for science communication. *Frontiers in communication*, *6*, 581302

Lin, C.-Y. (2004). ROUGE: A package for automatic evaluation of summaries. *Text Summarization Branches Out: Proceedings of the ACL Workshop*, 74–81.

Luckin, R., Holmes, W., Griffiths, M., & Forcier, L. B. (2016). *Intelligence unleashed: An argument for AI in education*. Pearson Education.

Mayer, R. E. (2009). *Multimedia learning* (2nd ed.). Cambridge University Press.

Mittal, U., Sai, S., & Chamola, V. (2024). A comprehensive review on generative AI for education. *IEEE Access, 12*, 16572–16590. https://doi.org/10.1109/ACCESS.2024.3367715

Moreno, R., & Mayer, R. E. (2007). Interactive multimodal learning environments. *Educational Psychology Review, 19*(3), 309–326. https://doi.org/10.1007/s10648-007-9047-2

Noetel, M., Griffith, S., Delaney, O., Sanders, T., Parker, P., del Pozo Cruz, B., & Lonsdale, C. (2021). Video improves learning in higher education: A systematic review. *Review of Educational Research, 91*(2), 204–236. https://doi.org/10.3102/0034654321990713

Notta AI. (s.f.). *Transcribe, Summarize, and Translate*. https://notta.ai

Otter.ai. (s.f.). *Transcribe and Summarize Meetings*. https://otter.ai

Papineni, K., Roukos, S., Ward, T., & Zhu, W.-J. (2002). BLEU: A method for automatic evaluation of machine translation. *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, 311–318.

Rajadell, M., & Garriga-Garzón, F. (2017). Educational videos: After the why, the how. *Intangible Capital*, *13*(5), 902-922.

Sonix AI. (s.f.). *Audio Transcription and Subtitling*. https://sonix.ai

Streamlit Inc. (s.f.). *Streamlit: The fastest way to build data apps*. https://streamlit.io

Winkler, R., & Söllner, M. (2018). Unleashing the potential of chatbots in education: A state-of-the-art analysis. *Academy of Management Proceedings, 2018*(1), 15903. https://doi.org/10.5465/AMBPP.2018.15903abstract

Young, T., Hazarika, D., Poria, S., & Cambria, E. (2018). Recent trends in deep learning based natural language processing. *IEEE Computational Intelligence Magazine, 13*(3), 55–75. https://doi.org/10.1109/MCI.2018.2840738

Apéndice 1: Descripción del código

Este apéndice describe los archivos de código desarrollados específicamente para implementar la herramienta presentada en este trabajo. Cada uno de los scripts .py cumple una función específica dentro del flujo de análisis, y en conjunto conforman la arquitectura completa de la aplicación.

- app.py
  Archivo principal desde el cual se ejecuta la aplicación en Streamlit. Gestiona la interfaz web y orquesta la ejecución de todos los módulos funcionales: transcripción del audio, mejora del texto, análisis de documentos, resumen, generación del *test* y funcionamiento del *chatbot*. Define la estructura visual, la navegación por secciones y los controles de interacción con el usuario.
  En las versiones más recientes, el archivo app.py incorpora dos funcionalidades adicionales orientadas a la evaluación y observabilidad del sistema.
  La primera corresponde a la evaluación cualitativa del resumen generado, donde el usuario puede introducir un resumen humano de referencia y valorar el resultado de la IA mediante sliders de puntuación y comentarios abiertos.
  Los datos recopilados se almacenan de forma estructurada en la capa gold/evaluation del datalake.
  La segunda funcionalidad implementa un sistema de registro de métricas de rendimiento, que mide el tiempo de ejecución de cada fase del pipeline (extracción, transcripción, limpieza, resumen, test y total), guarda los resultados en la carpeta gold/metrics y los compara con la media histórica de ejecuciones anteriores, mostrando un gráfico de barras generado con matplotlib.
  Estas incorporaciones refuerzan el carácter analítico del sistema y consolidan su enfoque en la trazabilidad y evaluación del flujo completo.

- transcripcion.py
  Contiene las funciones para extraer el audio del vídeo con MoviePy y convertirlo a formato FLAC y MP3 usando pydub. También gestiona la subida del audio al *bucket* de Google Cloud Storage y ejecuta la transcripción automática utilizando la API de Speech-to-Text. Además, incluye una función para optimizar y limpiar el texto transcrito mediante Gemini 2.5 Flash.

- resumen.py
  Este módulo se encarga de generar un resumen automático a partir del bloque de texto unificado (transcripción + documentos). Utiliza Gemini para sintetizar el contenido, conservando los conceptos clave y estructurándolo de forma clara y accesible para el usuario.

- test.py
  Implementa la generación automática de preguntas de tipo *test*. A partir del contenido global, construye diez preguntas de opción múltiple, cada una con cuatro posibles respuestas y una única opción correcta. Permite validar las respuestas del usuario desde la interfaz e informa al final del número de aciertos.

- chat.py
  Define el funcionamiento del *chatbot* contextual. Este componente permite hacer preguntas sobre el contenido procesado (vídeo y documentos), ofreciendo respuestas generadas por Gemini únicamente en base a la información cargada, sin recurrir a fuentes externas. Garantiza así coherencia y precisión.

- gcs.py
  Incluye las funciones necesarias para subir archivos a Google Cloud Storage, comprobar que la subida sea correcta y gestionar la ruta de acceso. Es clave para garantizar que el audio pueda ser procesado por la API de transcripción.

- config.py
  Módulo auxiliar que gestiona la carga y validación de las claves JSON necesarias para autenticar el acceso a los servicios de Google Cloud. Lee las credenciales desde google_key.json (Speech-to-Text) y gemini_key.json (Gemini) y las pone a disposición del resto de la aplicación.

En conjunto, estos módulos permiten ejecutar de forma secuencial y automatizada todas las tareas necesarias: desde la carga del vídeo hasta la generación de los contenidos derivados, incluyendo la interacción conversacional.

Se proporciona a continuación el enlace a la carpeta de Google Drive que contiene los códigos empleados en la implementación del sistema, con el fin de facilitar su consulta y revisión. https://drive.google.com/drive/folders/1LJU6CsoPFOwmFXfBjjuUY9mZBHgtTpWw?usp=sharing
Este proyecto requiere credenciales para Google Speech-to-Text y Gemini.
Por motivos de seguridad, los archivos JSON no se incluyen en el repositorio público.
El usuario debe generar sus propias credenciales en Google Cloud y guardarlas como:

- google_key.json

- gemini_key.json

en la carpeta raíz del proyecto.

La arquitectura del sistema sigue una estructura de datalake en Google Cloud Storage organizada en capas (raw, bronze, silver y gold), lo que garantiza la trazabilidad de todos los artefactos generados.

Apéndice 2: Materiales utilizados para pruebas

Durante el desarrollo se emplearon diversos archivos de prueba para validar cada componente del sistema. A continuación, se describen los materiales empleados y su función dentro del proceso:

- Vídeos de presentación
  Archivos MP4 con una duración media de entre 10 y 15 minutos, que sirvieron para evaluar el funcionamiento del módulo de extracción y transcripción de audio, así como la calidad del texto final procesado.

- Presentaciones PowerPoint (.pptx)
  Documentos utilizados como material complementario a los vídeos. Se procesaron diapositiva por diapositiva utilizando la librería python-pptx para extraer el contenido textual y combinarlo con la transcripción.

- Documentos Word (.docx)
  Archivos que contenían explicaciones adicionales o fragmentos de teoría. Se analizaron mediante python-docx y se integraron en el bloque global de información.

- Hojas de cálculo Excel (.xlsx)
  Materiales con tablas y esquemas relevantes para complementar la exposición oral. Se procesaron usando la librería openpyxl y se incorporaron al análisis.

- Archivos PDF
  Fueron tratados con PyPDF2 para extraer texto de presentaciones en formato cerrado. Su contenido también se integró al bloque principal sobre el que se generaron los resúmenes, *test* y respuestas del chat.

Todos los materiales fueron utilizados exclusivamente con fines de validación y respetando las condiciones de uso de las APIs de Google. Sirvieron para comprobar el comportamiento real del sistema en diferentes escenarios y formatos, garantizando su flexibilidad y robustez.

Asimismo, se emplearon estos mismos materiales para validar la correcta generación de métricas y evaluaciones, confirmando la estabilidad del flujo completo desde la carga hasta la obtención de los resultados finales.

[PÁGINA INTENCIONADAMENTE EN BLANCO]