Assmaa Ouladali

SISTEMA AUTOMATIZADO DE GENERACIÓN DE ACTAS PARA REUNIONES DE MICROSOFT TEAMS MEDIANTE PROCESAMIENTO MULTIMEDIA E INTELIGENCIA ARTIFICIAL

TRABAJO DE FIN DE GRADO

Dirigido por Edgar Batista De Frutos

Grado en Ingeniería Informática

Tarragona

2025

Resumen.

Este trabajo aborda el desafío de la documentación manual de reuniones virtuales, un proceso intensivo en tiempo y propenso a errores. Se presenta el diseño e implementación de un sistema integral y automatizado para la generación de actas de reuniones de Microsoft Teams. La justificación se basa en la necesidad de eficiencia y seguridad, ya que, a pesar de existir soluciones comerciales, la compañía requiere control absoluto sobre datos confidenciales.

La metodología se basa en una arquitectura de microservicios y grafos dirigidos (streams) dentro de la plataforma DOCs de la empresa. El flujo automatizado incluye: descarga de grabaciones desde SharePoint, procesamiento de audio y vídeo (con FFmpeg y AWS), transcripción y diarización (Amazon Transcribe), extracción de nombres de participantes mediante visión por computador y LLMs, generación y corrección del acta con un modelo de lenguaje (Anthropic Claude vía Bedrock), y finalmente, el envío del acta en PDF por correo electrónico (Outlook) y la integración con el asistente inteligente AIDA para consultas interactivas. Una parte clave de la metodología fue un análisis comparativo exhaustivo de FFmpeg y AWS para tareas de procesamiento multimedia, evaluando rendimiento, coste y uso de recursos.

Los resultados demuestran un sistema funcional que genera actas estructuradas y formales de manera autónoma. El análisis técnico reveló que FFmpeg demostró un rendimiento de procesamiento superior para la mayoría de operaciones de audio/vídeo en comparación con los servicios en la nube de AWS. Además, la precisión de la transcripción se evaluó rigurosamente utilizando la Tasa de Error de Palabra (WER), y se midió la precisión de la diarización, ambos mostrando resultados satisfactorios para la generación automatizada de actas. El proyecto no solo automatiza una tarea tediosa, sino que también establece las bases para futuras mejoras, como la implementación de memoria a largo plazo para el asistente AIDA.

Resum.

Aquest treball aborda el repte de la documentació manual de reunions virtuals, un procés intensiu en temps i propens a errors. Es presenta el disseny i implementació d'un sistema integral i automatitzat per a la generació d'actes de reunions de Microsoft Teams. La justificació es basa en la necessitat d'eficiència i seguretat, ja que, tot i existir solucions comercials, la companyia requereix control absolut sobre dades confidencials.

La metodologia es basa en una arquitectura de microserveis i grafs dirigits (streams) dins de la plataforma DOCs de l'empresa. El flux automatitzat inclou: descàrrega de gravacions des de SharePoint, processament d'àudio i vídeo (amb FFmpeg i AWS), transcripció i diarització (Amazon Transcribe), extracció de noms de participants mitjançant visió per computador i LLMs, generació i correcció de l'acta amb un model de llenguatge (Anthropic Claude vía Bedrock), i finalment, l'emissió de l'acta en PDF per correu electrònic (Outlook) i la integració amb l'assistent intel·ligent AIDA per a consultes interactives. Una part clau de la metodologia va ser una anàlisi comparativa

exhaustiva de FFmpeg i AWS per a tasques de processament multimèdia, avaluant rendiment, cost i ús de recursos.

Els resultats demostren un sistema funcional que genera actes estructurades i formals de manera autònoma. L'anàlisi tècnica va revelar que FFmpeg va demostrar un rendiment de processament superior per a la majoria d'operacions d'àudio/vídeo en comparació amb els serveis de núvol d'AWS. A més, la precisió de la transcripció es va avaluar rigorosament utilitzant la Taxa d'Error de Paraula (WER), i es va mesurar la precisió de la diarització, tots dos mostrant resultats satisfactoris per a la generació automatitzada d'actes. El projecte no només automatitza una tasca tediosa, sinó que també estableix les bases per a futures millores, com l'implementació de memòria a llarg termini per a l'assistent AIDA.

Abstract.

This Bachelor's Thesis addresses the challenge of manual documentation of virtual meetings, a time-consuming and error-prone process. It presents the design and implementation of a comprehensive and automated system for generating minutes of Microsoft Teams meetings. The justification is based on the need for efficiency and security, as, despite existing commercial solutions, the company requires absolute control over confidential data.

The methodology is based on a microservices architecture and directed graphs (streams) within the company's DOCs platform. The automated workflow includes: downloading recordings from SharePoint, audio and video processing (with FFmpeg and AWS), transcription and diarization (Amazon Transcribe), extraction of participant names using computer vision and LLMs, generation and correction of the minutes using a language model (Anthropic Claude via Bedrock), and finally, sending the minutes in PDF format by email (Outlook) and integration with the AIDA intelligent assistant for interactive queries. A key part of the methodology was a comprehensive comparative analysis of FFmpeg and AWS for media processing tasks, evaluating performance, cost, and resource usage.

The results demonstrate a functional system that generates structured and formal minutes autonomously. The technical analysis revealed that FFmpeg demonstrated superior processing performance for most audio/video operations compared to AWS cloud services. Furthermore, the transcription accuracy was rigorously evaluated using Word Error Rate (WER), and diarization precision was measured, both showing satisfactory results for automated minute generation. The project not only automates a tedious task but also lays the groundwork for future improvements, such as the implementation of long-term memory for the AIDA assistant.

Índice

Índice de tablas

Índice de figuras

1 Introducción

La digitalización de los entornos laborales, acelerada por el auge del teletrabajo, ha convertido las reuniones virtuales en un pilar fundamental para la cooperación y la toma de decisiones dentro de las empresas. Plataformas como Microsoft Teams se han consolidado como herramientas esenciales para la comunicación y coordinación de equipos distribuidos, generando un volumen sin precedentes de contenido audiovisual y discusiones que deben ser documentadas.

Las actas de reunión siguen siendo un recurso clave para garantizar la trazabilidad, la toma de decisiones y el seguimiento de acuerdos. Sin embargo, su elaboración manual es una tarea repetitiva, propensa a errores y con un elevado consumo de tiempo que podría destinarse a actividades de mayor valor añadido. Este contexto evidencia una necesidad clara de automatizar el proceso de documentación, pero siempre garantizando la máxima confidencialidad y seguridad, aspectos críticos cuando se trata con información corporativa sensible.

En este marco, el presente Trabajo de Fin de Grado tiene como objetivo el diseño y desarrollo de un sistema integral que permita la generación automática, segura y precisa de actas de reuniones de Microsoft Teams, contribuyendo a optimizar procesos empresariales y a garantizar la fiabilidad de la información registrada.

Es importante destacar que, aunque existen soluciones comerciales que ofrecen servicios similares, la empresa ha optado por desarrollar su propio software. Esta decisión responde tanto a criterios económicos como, especialmente, a la necesidad de mantener el máximo control y protección sobre los datos confidenciales, reforzando la seguridad y privacidad de la información corporativa.

1.1 Descripción general del proyecto

Mi proyecto es una generación automática de actas de reuniones de Microsoft Teams. En las empresas cada vez hay un mayor porcentaje de personas que teletrabajan, lo que aumenta considerablemente el número de reuniones virtuales, muchas de ellas celebradas en Teams. El proyecto se basa en generar automáticamente actas de reuniones a partir de las grabaciones, que deberán descargarse desde SharePoint, por lo que será necesario integrar el sistema con este software.

Posteriormente, se realizará un procesamiento de audio y vídeo: reducción de ruido, normalización y preparación del material mediante AWS MediaConvert, Rekognition, SQS, SNS, S3 y FFmpeg. La transcripción y diarización[1] se llevará a cabo con Amazon Transcribe, y se explorarán técnicas para extraer el nombre de los hablantes (visibles en la interfaz de Teams), así como información relevante del contenido compartido en pantalla.

El acta será generada mediante un LLM[2] y se evaluará su precisión, validez y veracidad antes de ser entregada al usuario en formato PDF a través de Outlook. Además, el sistema se integrará con el asistente empresarial AIDA[3], en el cual se creará una sesión contextualizada

[1] Proceso de separar un audio por hablantes, identificando qué parte del diálogo corresponde a cada persona.

[2] Modelo de lenguaje, del inglés Large Language Model

[3] Artificial Intelligence Digital Assistant

con la información de la reunión, lo que permitirá realizar consultas interactivas sobre su contenido y ampliará las posibilidades más allá del acta estática.

1.2 Necesidades

Este proyecto surge de la necesidad de combatir la ineficiencia del proceso manual de creación de actas y de dar respuesta a exigencias empresariales específicas en materia de seguridad y control. Tal y como se expuso en la introducción, aunque existen soluciones comerciales que abordan parcialmente este problema, la compañía ha optado por desarrollar su propia herramienta debido a la confidencialidad de los datos, al tratarse de información altamente sensible.

Las necesidades concretas que motivan este proyecto son las siguientes:

- **Eficiencia y ahorro de tiempo:** Reducir la carga administrativa derivada de la elaboración manual de actas y permitir que los profesionales dediquen su tiempo a tareas de mayor valor añadido.
- **Automatización integral:** Eliminar la intervención manual en todo el flujo, desde la descarga de la grabación en SharePoint hasta la generación y envío del acta final en formato PDF mediante Outlook.
- **Extracción de contexto enriquecido:** Ir más allá de la simple transcripción de la reunión, incluyendo la identificación de hablantes, la detección de momentos en los que se comparte pantalla y la extracción de información significativa de dicho contenido.
- **Precisión y verificación:** Asegurar que el acta generada por el LLM refleje de forma fiel y coherente el desarrollo de la reunión, garantizando su validez, precisión y veracidad.
- **Integración con el ecosistema Microsoft 365:** Conectarse de forma fluida con las herramientas ya utilizadas en la empresa (Teams, SharePoint y Outlook), garantizando una experiencia de usuario continua y sin fricciones.
- **Confidencialidad y seguridad:** Garantizar la protección de los datos sensibles mediante un software desarrollado y gestionado por la propia empresa, lo que permite mantener un control absoluto sobre la información y evita riesgos asociados al uso de soluciones de terceros.

En conjunto, estas necesidades reflejan la voluntad de la empresa de disponer de una solución propia que combine eficiencia, automatización, seguridad e integración, aportando valor añadido a los procesos de documentación de reuniones virtuales.

1.3 Previsión de uso

El sistema tendrá una aplicación práctica en múltiples ámbitos empresariales:

- **Empresas y corporaciones:** Documentación automática de reuniones de trabajo, consejos de administración y encuentros con clientes.
- **Departamentos de recursos humanos:** Seguimiento de reuniones internas, entrevistas y procesos de selección.

- **Equipos de proyectos distribuidos:** Generación de actas para garantizar la coordinación y el seguimiento de acuerdos entre miembros en diferentes ubicaciones.
- **Asistencia interactiva:** Gracias a la integración con AIDA, los trabajadores podrán consultar información de manera dinámica, más allá del acta estática.

1.4 Objetivos del TFG

El presente Trabajo de Fin de Grado tiene como finalidad desarrollar una solución integral que automatice la generación de actas de reuniones de Microsoft Teams y permita la interacción inteligente con la información extraída. Los objetivos se pueden clasificar en técnicos/funcionales y formativos/aprendizaje:

Objetivos técnicos y funcionales:

- **Procesamiento de grabaciones:** Desarrollar un sistema capaz de gestionar grabaciones de Microsoft Teams, incluyendo la descarga desde SharePoint y el posterior procesamiento de audio y vídeo.
- **Transcripción precisa:** Obtener transcripciones fiables mediante Amazon Transcribe y técnicas adicionales de limpieza y preprocesamiento de señal.
- **Identificación de participantes y contenido compartido:** Implementar un módulo de diarización de hablantes y detección de contenido compartido, así como extracción de información relevante de dicho contenido.
- **Generación automática de actas:** Producir actas precisas, coherentes y verificables utilizando un modelo de lenguaje, en formato formal PDF, y distribuirlas automáticamente mediante Outlook.
- **Integración con asistentes inteligentes:** Conectar la solución con el asistente empresarial AIDA para permitir interacciones avanzadas con la información de la reunión.

Objetivos formativos y de aprendizaje:

- **Integración de sistemas y APIs:** Diseñar y orquestar un flujo de trabajo que combine múltiples servicios, como Microsoft Graph API, AWS S3, Transcribe, MediaConvert, Rekognition, SNS/SQS y la API del servicio de envío de correos (SMTP).
- **Procesamiento multimedia:** Aplicar técnicas avanzadas de audio y vídeo usando herramientas industriales como FFmpeg y servicios cloud (MediaConvert, Rekognition), comprendiendo los desafíos de calidad y sincronización.
- **Desarrollo en la nube:** Adquirir experiencia práctica con arquitecturas serverless y servicios gestionados de IA en AWS, incluyendo aspectos de escalabilidad, seguridad y costes.
- **Ingeniería de prompts y evaluación de LLMs:** Experimentar con técnicas de prompt engineering para generar actas estructuradas y diseñar métricas objetivas que aseguren su validez y veracidad.

- **Resolución de problemas complejos de IA:** Combinar visión por computador y procesamiento de lenguaje natural para extraer información contextual, nombres de participantes y momentos clave de la reunión.

En conjunto, estos objetivos buscan automatizar y optimizar la generación de actas, al tiempo que consolidan competencias técnicas avanzadas y habilidades prácticas en un proyecto con alto valor añadido para la empresa y proyección profesional para el estudiante.

1.5 Planificación

A continuación, se presenta la planificación del proyecto mediante un diagrama de Gantt, que incluye las distintas etapas de análisis, diseño, desarrollo, integración y pruebas finales. De manera complementaria, se incorpora la tarea recurrente de elaboración de documentación durante el proceso, así como una tarea final destinada a la redacción de la memoria del proyecto, en la que se sistematiza la información recopilada y se lleva a cabo un análisis de los resultados obtenidos.

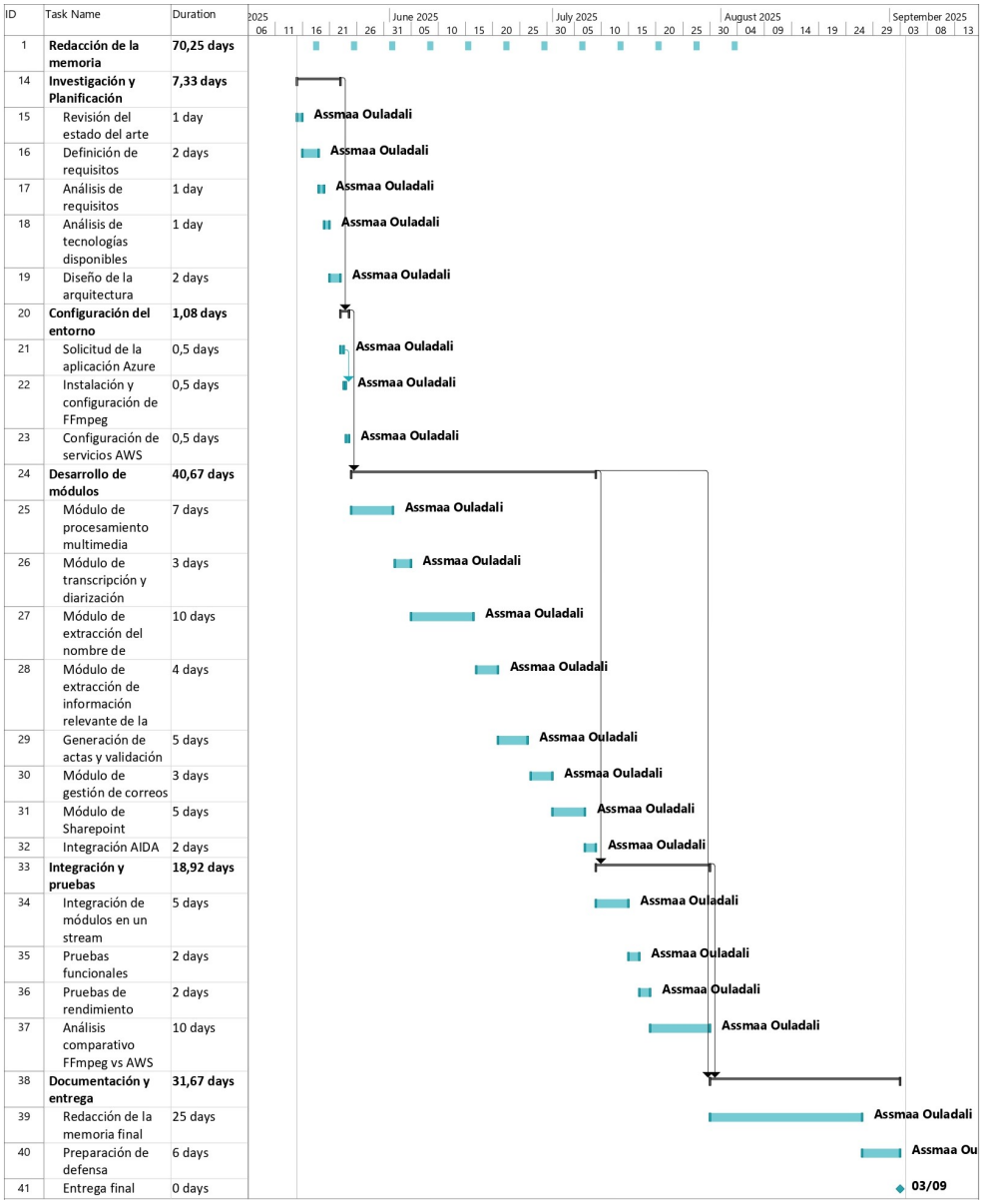


Figura 1. Diagrama de Gantt: Planificación del proyecto

2 Estado del arte

2.1 Procesamiento de Lenguaje Natural

El **Procesamiento de Lenguaje Natural** (NLP, por sus siglas en inglés) es una rama de la Inteligencia Artificial dedicada a la interacción entre computadoras y lenguaje humano. Su propósito es lograr que las máquinas comprendan, interpreten y generen lenguaje de forma natural y útil. Entre sus aplicaciones más comunes se encuentran los asistentes virtuales, traductores automáticos, análisis de sentimiento, chatbots, así como el reconocimiento y la síntesis de voz.

2.1.1 BERT (2018), RoBERTa (2019) y BART (2019)

El NLP ha avanzado de forma revolucionaria gracias a arquitecturas basadas en transformers y técnicas de preentrenamiento. Modelos como BERT[4] [1] y RoBERTa[5] [2] marcaron un nuevo estándar en tareas lingüísticas como comprensión lectora e inferencia del lenguaje.

BERT es un modelo diseñado para preentrenar representaciones bidireccionales profundas a partir de texto sin etiquetar, considerando simultáneamente el contexto izquierdo y derecho en todas sus capas. RoBERTa surge como una optimización de BERT, tras comprobarse que este estaba subentrenado. Al aplicar mejoras en su entrenamiento y aumentar el volumen de datos, RoBERTa logró igualar e incluso superar a modelos posteriores.

En síntesis, BERT fue pionero en el procesamiento bidireccional del lenguaje, mientras que RoBERTa alcanzó un rendimiento superior gracias a técnicas de entrenamiento optimizadas y más datos, aunque a costa de un mayor consumo computacional y solo un incremento marginal en el número de parámetros.

BART[6] [3] representa una evolución que combina un codificador bidireccional con un decodificador autorregresivo. Se emplea en generación de lenguaje natural, traducción y comprensión, unificando y mejorando enfoques previos como BERT y GPT[7] mediante un método de denoising autoencoding, en el cual el modelo aprende a reconstruir texto original a partir de entradas corrompidas con ruido.

BART integra lo mejor de BERT (bidireccionalidad) y GPT (autorregresividad), alcanzando el estado del arte en tareas de summarización y mostrando alta competitividad en comprensión. Además, ofrece gran flexibilidad para adaptarse a distintos escenarios incluso con datos limitados. No obstante, presenta desventajas como tendencia a generar información no respaldada por la fuente (alucinaciones), requiere un 10% más de parámetros que BERT de igual tamaño y demanda elevados costos computacionales en su preentrenamiento.

[4] Bidirectional Encoder Representations from Transformers

[5] Robustly Optimized BERT Pretraining Approach

[6] Bidirectional and Auto-Regressive Transformers

[7] Generative Pre-trained Transformer

2.1.2 Ventajas e inconvenientes

La principal ventaja del NLP es que permite la interacción sin necesidad de comandos específicos o interfaces complejas, lo que lo hace especialmente útil en plataformas de educación a distancia. También posibilita estudios de reputación y marca, facilita la recuperación de información en grandes repositorios y mejora la comprensión humano-máquina. Además, amplía el acceso a sistemas informáticos para personas mayores o con discapacidad.

Sus principales desventajas son la limitada capacidad de las máquinas para entender sobreentendidos, lo que requiere razonamiento avanzado, y la necesidad de gran precisión en el procesamiento de documentos, pues los errores de interpretación pueden derivar en consecuencias negativas en la toma de decisiones [4].

2.2 Transcripción automática

2.2.1 Transformadores en sistemas de diálogo hablado

Los transformadores se han consolidado como una arquitectura eficaz en el procesamiento del habla gracias a su capacidad para manejar datos secuenciales mediante autoatención, lo que ha revolucionado tareas como el reconocimiento automático y la síntesis de voz.

Estudios recientes muestran que están reemplazando a las **redes neuronales recurrentes (RNN)** en ASR[8] y TTS[9], logrando resultados prometedores. En particular, el preentrenamiento con técnicas de aprendizaje auto-supervisado, como la herramienta wav2vec, acelera la convergencia y mejora el rendimiento.

Además, investigaciones [5] indican que los modelos híbridos que combinan transformadores con técnicas de modelado acústico convencionales mejoran las tasas de error de palabras y reducen la complejidad computacional, constituyendo un enfoque prometedor para el futuro.

2.2.2 Técnicas avanzadas de aprendizaje profundo en ASR

A continuación, se presentan las técnicas avanzadas de aprendizaje profunda para mejorar los resultados de la transcripción automática que fueron presentadas en un estudio del 2024 [6].

2.2.2.1 Desafíos del ASR Tradicional

A pesar de los avances, los sistemas tradicionales de reconocimiento automático de voz presentan limitaciones:

- Necesitan grandes volúmenes de datos de entrenamiento.
- Manejan información confidencial, lo que genera problemas de privacidad.
- Requieren altos recursos computacionales y de almacenamiento.

[8] Automatic Speech Rekognition

[9] Text-to-Speech

- Suelen asumir que los datos de entrenamiento y prueba pertenecen al mismo dominio.

Estas limitaciones han impulsado el desarrollo de técnicas avanzadas de aprendizaje profundo.

2.2.2.2 Enfoques Innovadores

- **Aprendizaje por Transferencia (DTL):** Permite entrenar modelos de alto rendimiento con pocos datos específicos, reutilizando conocimiento de dominios relacionados.
- **Aprendizaje Federado (FL):** Entrena modelos de forma distribuida en dispositivos locales, compartiendo solo parámetros y preservando la privacidad.
- **Aprendizaje por Refuerzo (DRL):** Mejora la adaptación del ASR a entornos cambiantes mediante recompensas, aumentando la robustez y reduciendo costes a largo plazo.

El estudio destaca que estas técnicas son más efectivas cuando se combinan con transformers, lo que incrementa rendimiento, eficiencia y precisión en ASR.

2.2.2.3 Desafíos

Los principales desafíos en este campo han sido identificados en la literatura [7] e incluyen:

- Diseñar arquitecturas eficientes que equilibren precisión y complejidad (ej. Conformer).
- Garantizar buen desempeño en entornos reales con ruido, disfluencias y habla espontánea.
- Optimizar recursos y latencia para un uso práctico en dispositivos limitados, sin perder exactitud en tiempo real.

2.3 Evaluación de Sistemas de Documentación y Resúmenes Automáticos

Se analizan metodologías para evaluar la calidad de documentación y resúmenes generados automáticamente, considerando aplicaciones de IA en entornos clínicos [8].

2.3.1 Inteligencia Artificial en Documentación Clínica

Panorama general y metodología:

La IA está transformando la documentación clínica mediante tecnologías como NLP, ASR y ML[10], optimizando la gestión de registros y reduciendo la carga administrativa. Se realizó una revisión sistemática siguiendo PRISMA[11], con búsquedas en Ovid, PubMed y BMJ, seleccionando 36 estudios recientes (desde 2019) para un análisis detallado.

[10] Aprendizaje automático, del inglés machine learning

[11] PRISMA (Preferred Reporting Items for Systematic Reviews and Meta-Analyses) es una guía para transparentar y estandarizar la elaboración y reporte de revisiones sistemáticas y metaanálisis

Resultados y tendencias principales:

- Reducción de carga administrativa: La IA disminuye el tiempo dedicado a la documentación.
- Mejora de legibilidad: Los LLMs generan resúmenes médicos más comprensibles.
- Integración con sistemas existentes: Se busca compatibilidad con EHR[12].
- Precisión: Persisten desafíos como errores o "alucinaciones" que requieren supervisión humana.

Desafíos y consideraciones éticas:

- Gestión de errores y alucinaciones.
- Incertidumbre legal sobre responsabilidad.
- Obstáculos de integración con EHR.
- Preocupaciones éticas sobre privacidad, consentimiento y uso de datos de pacientes.

Paralelismos con generación de actas y resúmenes ejecutivos:

La documentación clínica con IA comparte tecnologías y beneficios con la generación automática de actas y resúmenes: ASR para transcribir, NLP para estructurar y LLM para redactar documentos. Beneficios: ahorro de tiempo, estandarización y documentación completa. Desafíos: errores, precisión contextual y necesidad de verificación. Diferencia clave: en salud los errores afectan pacientes, en empresas son riesgos operativos o económicos. La convergencia tecnológica permite que avances en un área beneficien a la otra.

2.4 Soluciones comerciales

Las reuniones virtuales se han convertido en un recurso clave para la colaboración moderna, aunque capturar y organizar su contenido sigue siendo un reto. Diversas soluciones comerciales basadas en inteligencia artificial optimizan la productividad mediante la automatización de actas, transcripciones y resúmenes de reuniones [9] [10] [11] [12] [13] [14].

2.4.1 Otter.ai

Otter.ai transcribe conversaciones en tiempo real con alta precisión y cuenta con más de 15 millones de usuarios. Ha sido reconocida por The Wall Street Journal en junio de 2023.

Características clave: AI Meeting Agent para notas y resúmenes automáticos, transcripción en vivo compatible con Zoom, Google Meet y Teams, resúmenes de 30-60 minutos en 30 segundos, OtterPilot para unirse automáticamente a reuniones y Otter AI Chat para interactuar con el contenido.

Ventajas: Ahorro de tiempo, transcripción en tiempo real, automatización de reuniones, compatibilidad multiplataforma, colaboración mejorada, integraciones con herramientas empresariales y generación de resúmenes accionables.

[12] Electronic Health Record

Inconvenientes: Curva de aprendizaje, funcionalidad limitada sin conexión, errores con conversaciones superpuestas, soporte de idiomas restringido y opciones de personalización limitadas.

2.4.2 Fireflies.ai

Fireflies.ai es un asistente virtual que graba, transcribe y analiza reuniones automáticamente, integrándose con calendarios y plataformas de videoconferencia.

Características clave: Transcripción multi-idioma, motor de IA para análisis de contenido, plataforma colaborativa, búsqueda contextual, integraciones con CRMs, análisis conversacional y seguridad de datos.

Ventajas: Elimina tareas manuales, captura completa de información, mejora la comunicación post-reunión, compatibilidad multiplataforma, acceso histórico eficiente, centralización de datos, optimización de reuniones y seguridad avanzada.

Inconvenientes: Dependencia de internet, dificultades con acentos, ausencia de función offline, restricciones por plan y curva de aprendizaje.

2.4.3 Fathom

Fathom combina grabación, transcripción y análisis para generar resúmenes con destacados automáticos de información relevante.

Características clave: Asistente de IA para reuniones, análisis inteligente de puntos clave, integración con CRMs y herramientas como Zapier, y compartición personalizable de clips.

Ventajas: Ahorro de tiempo significativo, mejora en la participación, acceso inmediato a resultados, seguridad garantizada, compatibilidad multiplataforma.

Inconvenientes: Dependencia tecnológica, preocupaciones de privacidad, compatibilidad limitada con plataformas, sin funcionalidad offline, curva de aprendizaje, y limitaciones en colaboración en tiempo real.

2.4.4 Análisis comparativo

2.4.4.1 Factores generales

Característica	**Otter.ai**	**Fireflies.ai**	**Fathom**
Precio	- Básico: Gratis - Pro: $8.33/usuario x mes - Business: $20/usuario x mes - Enterprise: Personalizado	- Básico: Gratis - Pro: $10/usuario x mes - Business: $19/usuario x mes -Enterprise: $39/ usuario x mes	- Básico: Gratis - Pro: $15/usuario x mes - Team Edition: $19/usuario x mes - Team Edition Pro: $29/ usuario x mes
Puntuación (rate)	4.3/5 (G2)	4.8/5 (G2)	5/5 (G2)

Integración	Google, Microsoft office, Amazon S3, Android app, iOS app, HubSpot, Jira, Asana, Dialpad, Dropbox, Egnyte, JustCall, Notion, Outreach, RingCentral, Salesforce, Slack, Snowflake, Zapier, Zoom y Clickup	Google, Microsoft office, Dropbox, HubSpot, Jira, Salesforce, Zoom, Slack, BambooHR, Greenhouse, Lever, Notion, Zapier, Any.Do, Linear, Webex, Skype, Clickup, Asana, GoToMeeting, Monday, LifeSize…	Google, Microsoft office, HubSpot, Jira, Amie, Asana, Brave, Calendly, ChatGPT, Claude AI, ClickUp, Close, CoPilot AI, monday Work, Management, Motion, Notion, Otter.ai, Pipedrive, Salesforce, Skype, Slack, Smartsheet, Text Reader AI, Trello, Zapier…

Tabla 1. Análisis comparativo de tecnologías de transcripción. Factores generales

Si se prioriza el presupuesto, Otter.ai es la opción más económica. Fathom sobresale en experiencia del usuario con su calificación perfecta, mientras que Fireflies.ai y Fathom ofrecen mayor versatilidad de integración. La elección final dependerá de las necesidades del usuario o empresa, equilibrando precio, calidad y compatibilidad con su ecosistema de herramientas.

2.4.4.2 Factores de transcripción de reuniones

Factor	Otter.ai	FireFiles.ai	Fathom
Speech-to-Text	88%	89%	91%
Transcription Accuracy	80%	86%	90%
Speaker Identification	75%	85%	91%
AI Text Summarization	86%	89.5%	96%
Real-Time Transcription	85%	85%	Funcionalidad no ofrecida

Tabla 2. Análisis comparativo de tecnologías de transcripción. Factores de transcripción de reuniones

La comparativa indica que Fathom es ideal para quienes priorizan precisión y calidad sin necesidad de procesamiento en tiempo real, Fireflies.ai ofrece un buen equilibrio entre precisión y funcionalidad en tiempo real, y Otter.ai resulta adecuado si el presupuesto o ciertas integraciones son clave. La decisión dependerá de las necesidades específicas de cada usuario o empresa.

2.4.4.3 Factores de calidad y experiencia

Categoría	Otter.ai	FireFiles.ai	Fathom
Precisión y Fiabilidad	4.7/5	4.5/5	4.8/5

Facilidad de Uso	4.5/5	4.2/5	4.7/5
Funcionalidad y Características	4.8/5	4.6/5	4.6/5
Rendimiento y Velocidad	4.6/5	4.3/5	4.5/5
Personalización y Flexibilidad	4.4/5	4.1/5	4.2/5
Privacidad y Seguridad de Datos	4.3/5	4.8/5	4.9/5
Soporte y Recursos	4.2/5	4.0/5	4.3/5
Puntuación Global	4.5/5	4.4/5	4.6/5

Tabla 3. Análisis comparativo de tecnologías de transcripción. Factores de calidad y experiencia

La comparativa indica que Fathom es ideal para quienes priorizan precisión, seguridad y facilidad de uso; Otter.ai destaca por su amplia funcionalidad; y Fireflies.ai sobresale en seguridad de datos. La elección final, igual que en las comparativas anteriores, dependerá de las prioridades y necesidades específicas de cada usuario u organización.

2.5 Análisis de tecnologías disponibles

2.5.1 Transcripción automática de audio y video: Google, Microsoft y Amazon

La transcripción automática convierte audio o video en texto, facilitando accesibilidad, análisis de contenido y elaboración de actas de reuniones mediante IA y redes neuronales profundas. Permite guardar discusiones, generar subtítulos y traducir conversaciones multilingües [15][16], y sirve como base para herramientas de IA generativa que resuman puntos clave y asignen tareas.

2.5.1.1 Google Cloud Speech-to-Text

De acuerdo con la literatura [15], Google ofrece un servicio de reconocimiento de voz compatible con más de 125 idiomas, modelos para subtitulado, transcripción síncrona, asíncrona o streaming, con etiquetado de oradores, baja latencia y personalización. Implementa cifrado y cumplimiento regulatorio.

Ventajas: Alta precisión, soporte multilingüe, transcripción en tiempo real y por lotes, integración con Google Cloud.

Inconvenientes: Requiere internet, costo elevado en grandes volúmenes y procesamiento en servidores de Google.

2.5.1.2 Microsoft Copilot para Teams

Según diversas fuentes [17][18][19][20][21][22], este asistente de IA integrado en Teams transcribe reuniones en vivo en múltiples idiomas, genera resúmenes y tareas automáticamente, y respeta políticas organizacionales.

Ventajas: Integración nativa, IA avanzada, correcciones en tiempo real, integración con Microsoft 365.

Inconvenientes: Costo adicional por licencias, limitado a Teams, depende de la transcripción previa.

2.5.1.3 Amazon Transcribe de AWS

Tal como señalan varios estudios [16][23], Amazon Transcribe es un servicio ASR de AWS para desarrolladores, compatible con más de 100 idiomas, puntuación, diarización, análisis de sentimientos, personalización de vocabulario e integración con otros servicios AWS.

Ventajas: Escalabilidad, integración AWS, precisión robusta, personalización.

Inconvenientes: Requiere conocimientos de AWS, estructura de precios compleja, integración limitada fuera de AWS.

2.5.1.4 Caso de uso: Resúmenes y actas de reuniones

Google, Amazon y Microsoft permiten transcribir reuniones en Teams, Meet o Zoom, con enfoques distintos. Microsoft Copilot automatiza transcripción y generación de resúmenes, acciones y seguimiento sin necesidad de herramientas externas [19].

2.5.2 Modelos LLM

Se comparan GPT-4/3.5, Claude, Llama 2, Gemini/PaLM y Titan según precisión, ventana de contexto y velocidad de respuesta [24][25] [26][27][28][29][30].

2.5.2.1 Precisión

GPT-4 lidera benchmarks MMLU (86.4%), seguido de Claude 3.5, Llama 2 y Gemini 1.5. GPT-4 genera resúmenes detallados; Claude sintetiza de manera más concisa. Titan Premier ofrece calidad comparable en entornos empresariales.

2.5.2.2 Ventana de contexto

- GPT-4.1: hasta 1M tokens (experimental)
- GPT-3.5 Turbo: Aproximadamente 16K
- Claude 3.5: 200K Llama 2: 4K
- Gemini 1.5 Pro: 128K (1M privado)
- Titan Premier: 32K

2.5.2.3 Velocidad de respuesta

GPT-4.1 mini y nano reducen latencia respecto a GPT-4; Claude 3.5 Sonnet es más veloz que Claude 3 Opus. Llama 2 responde rápido con optimizaciones locales; Titan Premier está optimizado para entornos productivos. En resumen, variantes mini/nano o turbo de GPT-4 y Claude 3.5 ofrecen mejor velocidad, mientras Llama 2 destaca en implementaciones locales.

2.6 Mejores prácticas

La investigación integró documentación técnica de Google Cloud Speech-to-Text [31], AWS Transcribe/Bedrock, soluciones comerciales (NVIDIA Riva [32], Otter.ai, Fireflies.ai), literatura científica de ICASSP y arXiv, y casos de implementación empresarial (Adam.ai,

meeting.ai [33][34]), asegurando soluciones robustas con respaldo académico y viabilidad comercial.

2.6.1 Criterios de selección

Se priorizaron tecnologías por precisión (bajo WER/DER), adopción comercial, viabilidad de implementación y respaldo académico. Destacan: Transcribe-to-Diarize de Microsoft Research [35], NVIDIA Riva "nivel humano" y AWS Transcribe con 97% de precisión [34]. Myrtle.ai logra latencias de casi 0.15s con FPGA[13] frente a 0.6-1.1s de servicios convencionales [36].

2.6.1.1 Procesamiento de audio

- **Reducción de ruido:** RNNoise y filtros espectrales mejoran la señal antes del ASR [37].
- **Detección de voz (VAD/SAD):** Silero VAD y WebRTC VAD agilizan el pipeline [38][39].
- **Normalización:** Ajuste de volumen y muestreo para compatibilidad ASR.
- **Segmentación de hablantes:** Kaldi y pyannote-audio etiquetan "quién habla cuándo" [35][37].
- **Estrategias multimicrófonos:** Agrupación por dispositivo evita confusión de canales [40].
- **Segmentación temporal:** Fragmentos (aproximadamente de 20 minutos) aceleran diarización con mínima pérdida [40].

2.6.1.2 Estrategias de prompting para LLMs

- **Instrucciones claras:** Prompts guían al modelo [41].
- **System prompts:** Definir rol/formato mejora la salida [42].
- **Few-shot:** Ejemplos de transcripción-resumen entrenan al modelo [43].
- **SOPs[14] y plantillas:** Alinean LLM con flujos humanos de notas [43].
- **Corrección de transcripción:** Instrucciones para "adivinar palabras mal transcritas" mejoran la calidad [42].

2.6.2 Patrones exitosos en sistemas similares

- **Sistemas comerciales integrados:** Plataformas combinan audio/video, ASR, análisis LLM y resúmenes (Otter.ai, Fireflies.ai).
- **Pipelines DIY:** Whisper y LLM para resumen; pyannote-audio antes de Whisper mejora diarización [42][44].

---

[13] Field-Programmable Gate Array, dispositivo de hardware reconfigurable usado para acelerar procesamiento digital y tareas de inteligencia artificial

[14] Standard Operating Procedures, procedimientos operativos estándar que establecen pasos consistentes para realizar tareas específicas

- **LLM especializado:** RAG[15] o memorias de reuniones (Grain.ai, Google Workspace) con embeddings conversacionales.
- **Herramientas especializadas:** APIs (AssemblyAI, Soniox, Rev AI) y sistemas FPGA (CAIMAN) con baja latencia y WER competitivo [36].

2.6.3 Arquitecturas de Referencia: Transcripción y Resumen Automático

adam.ai (NVIDIA Riva)

- Pipeline de microservicios en Google Cloud (Dataflow, Pub/Sub, Kubernetes) con ASR de baja latencia y LLM para resumen [33].
- Precisión alta y baja latencia, extracción de actas casi en tiempo real [33].

meeting.ai - EKS y Bedrock (Claude 3) [34]

- Backend en AWS EKS, RDS, ElastiCache y Kafka.
- Bedrock (Claude 3) con aproximadamente un 97% de precisión en transcripción y 55% más satisfacción en resúmenes.

AWS Chime Meeting Summarizer

- Chime SDK captura audio y lo envía a Transcribe [45].
- Step Functions coordina Transcribe y Bedrock en flujo automatizado [46].
- Soporte multiplataforma (Zoom, Teams, Webex) con transcripción near real-time.
- Incluye redacción de información personal (PII) [46].

2.7 Exploración de casos de uso específicos

Este apartado identifica casos de uso particulares y sus requisitos técnicos, funcionales y normativos, para evaluar cómo adaptar la solución a necesidades específicas. Aunque la solución es genérica, podrá aplicarse a distintos escenarios empresariales según el usuario.

2.7.1 Casos de uso identificados

Reuniones técnicas

- Discusión de detalles técnicos (software, arquitectura, bugs).
- Requisitos: Integración con repositorios (GitHub, Azure DevOps), detección automática de términos técnicos, generación de actas con secciones estructuradas (decisiones técnicas, tareas, plazos).

Sesiones de brainstorming

- Reuniones creativas con alta densidad de ideas y colaboración.
- Requisitos: Clasificación de ideas por tema mediante NLP, generación de mapas mentales visuales, priorización automática de ideas por menciones o votos.

[15] Retrieval-Augmented Generation

Seguimiento de proyectos

- Reuniones de actualización de estado (metas, riesgos, recursos).
- Requisitos: Extracción de métricas clave (KPIs[16]), integración con herramientas de gestión (Jira, Asana), detección de riesgos mediante análisis de sentimiento.

Reuniones con clientes

- Reuniones comerciales o de soporte.
- Requisitos: Enmascaramiento de datos sensibles, generación de resúmenes internos y para cliente, cumplimiento de GDPR[17] y LGPD[18].

Reuniones de formación

- Sesiones educativas o workshops.
- Requisitos: Extracción de FAQs, identificación de "momentos clave" en video, integración con LMS[19] como Moodle o Canvas para subir materiales.

2.7.2 Consideraciones finales de los casos de uso

Cada escenario presenta necesidades particulares. Adaptar la solución a cada caso requeriría desarrollo personalizado. La estrategia actual es implementar una solución genérica que siente las bases del sistema. Posteriormente, se podrá escalar y adaptar la plataforma a necesidades específicas, como integración con repositorios para reuniones técnicas o análisis de sentimiento para seguimiento de proyectos. La plataforma permite convertir las conversaciones en información útil y accionable, sin importar el área (desarrollo, marketing, ventas).

[16] Key Performance Indicator

[17] General Data Protection Regulation

[18] Ley General de Protección de Datos

[19] Learning Management System

3 Introducción a DOCs

DOCs es un proyecto de automatización en el que he trabajado desde mis prácticas académicas y en el que sigo trabajando actualmente en la empresa Applus IDIADA, donde también integré mi Trabajo de Fin de Grado.

La herramienta es relativamente joven, y el hecho de haberme integrado al equipo en una etapa temprana me ha permitido entender y participar en gran parte de su desarrollo. En el momento de mi incorporación, la herramienta aún estaba en fase de desarrollo y no había sido desplegada en producción. Esto me ha dado la oportunidad de seguir su evolución desde la etapa de desarrollo hasta su recién despliegue en los entornos de preproducción y producción.

Dado este contexto, resulta necesario presentar brevemente el proyecto DOCs antes de profundizar en los requisitos, análisis, diseño e implementación del TFG, ya que su estructura y arquitectura condicionan directamente muchas de las decisiones adoptadas durante su desarrollo. Conocer el funcionamiento general del sistema permite entender mejor el enfoque seguido y las soluciones técnicas implementadas en este trabajo.

3.1 ¿Qué es DOCs?

DOCs es una *herramienta de automatización de procesos de datos basada en inteligencia artificial* que permite diseñar procesos para extraer el máximo valor de los datos disponibles. Esta plataforma se caracteriza por su capacidad de ingesta de datos desde múltiples orígenes, permitiendo una conexión directa con diversos sistemas existentes.

DOCs está diseñado para ser utilizado por cualquier organización que necesite automatizar procesos basados en datos.

A continuación, se muestra la arquitectura del sistema:

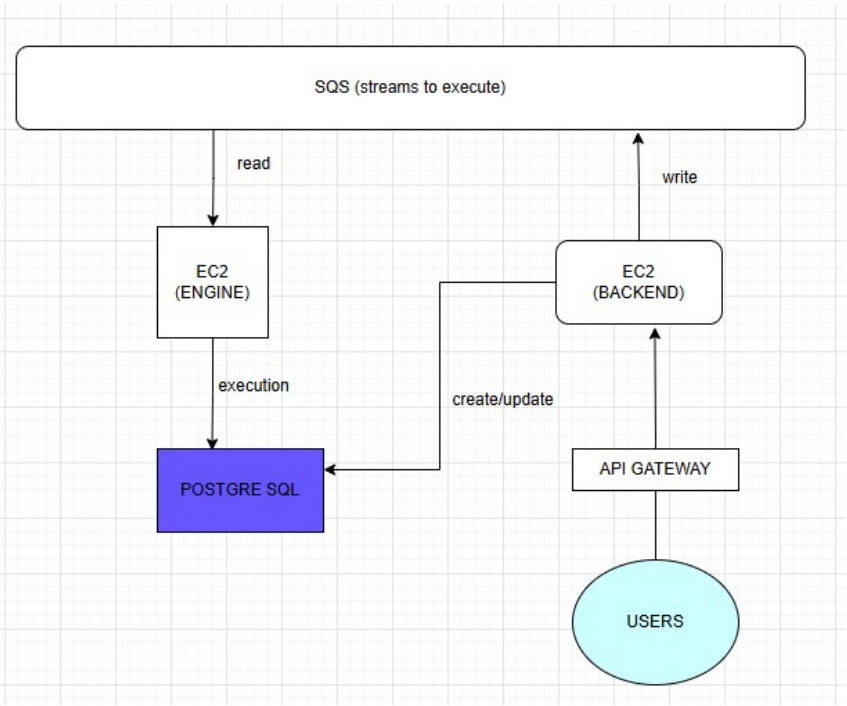


Figura 2. Arquitectura DOCs

DOCs presenta una arquitectura distribuida y escalable compuesta por los siguientes componentes principales:

- **Usuarios:** Interactúan con el sistema a través del API Gateway
- **API Gateway:** Recibe las peticiones de los usuarios y las redirige al backend
- **EC2 (BACKEND):** Gestiona las peticiones API, actualiza la base de datos PostgreSQL y envía tareas a la cola SQS
- **SQS (Streams to Execute):** Cola de mensajes que almacena las tareas pendientes de ejecución
- **EC2 (ENGINE):** Lee las tareas desde SQS, ejecuta las operaciones requeridas y actualiza la base de datos
- **PostgreSQL:** Base de datos central que almacena toda la información del sistema

Flujo de Datos: El flujo de información sigue el patrón: Usuarios → API Gateway → EC2 (BACKEND) → SQS → EC2 (ENGINE) → PostgreSQL, lo que permite un procesamiento asíncrono y una clara separación de responsabilidades para mejorar la escalabilidad y mantenibilidad del sistema.

Para que un usuario pueda solicitar la ejecución de un stream, este debe haber sido previamente creado. La creación de un stream consiste, fundamentalmente, en almacenarlo en la base de datos junto con un identificador único.

Cuando el usuario solicita la ejecución de un stream a través de la API Gateway, la petición es procesada por una instancia EC2 (backend), que se encarga de encolar el identificador del stream (asignado en el momento de su creación) en el sistema SQS. Una vez que el turno de ese stream llega en la cola, el sistema accede a la base de datos para recuperar el stream correspondiente al identificador encolado y, finalmente, lo ejecuta en la instancia EC2 (engine) encargada del procesamiento.

3.2 Streams de DOCs

Después de la breve introducción de la herramienta en sí, es necesario entender qué son los *"streams"* en DOCs.

Un **stream** en DOCs es esencialmente un *grafo dirigido* que define un proceso de automatización completo. En este contexto, **cada nodo del grafo representa una tarea específica** que debe ejecutarse, mientras que **las conexiones entre nodos definen el orden de ejecución y el flujo de datos** a través del proceso.

La característica fundamental de los streams es que la salida de un nodo se convierte en la entrada de otros nodos, creando así una cadena de procesamiento donde los datos fluyen de manera estructurada y secuencial a través de las diferentes etapas del proceso automatizado.

Los streams se definen mediante **estructuras JSON** que especifican todos los componentes del grafo de procesamiento, que está compuesto por **nodos (Tareas)**, donde cada nodo representa una unidad de trabajo específica y se identifica mediante un ID único, nombre descriptivo, información de la tarea, configuración de entrada que define los datos que recibe el nodo, ya sea de otro nodo o no, y conexiones condicionales para construir saltos y bucles en el grafo.

Las conexiones condicionales permiten evaluar condiciones específicas sobre los datos procesados y según el resultado, la ejecución puede dirigirse a diferentes nodos lo que permite crear flujos de procesamiento adaptativos y dinámicos.

Los streams suportan ejecuciones paralelas, lo que permite que diferentes ramas del grafo se ejecuten en paralelo si entre ellas no existe una dependencia, y también permite tener subgrafos anidados, es decir, en un grafo se puede hacer referencia a otro.

Trabajar con este modelo de streams permite tener flexibilidad, reutilización, escalabilidad, mantenibilidad, extensibilidad y trazabilidad.

Toda esta explicación se entenderá mejor cuando se explique el diseño del stream (grafo) elaborado para el TFG.

4 Requisitos

En esta sección se explican los diferentes requisitos funcionales, con las correspondientes reglas de negocio y casos de uso y su especificación textual, y los requisitos no funcionales.

4.1 Requisitos funcionales

4.1.1 Reglas de negocio

01. El usuario debe ser capaz de establecer la conexión a SharePoint.
02. El usuario debe ser capaz de indicar la localización de la grabación de reunión en SharePoint.
03. El sistema debe ser capaz de descargar la grabación de SharePoint mediante la información proporcionada por el usuario.
04. El sistema debe notificar al usuario en caso de no poder establecer la conexión a SharePoint, falta de permisos o la no localización del archivo indicado.
05. El sistema y usuario deben ser capaces de separar el audio del video.
06. El sistema y usuario deben ser capaces de reducir el ruido del audio extraído de la grabación de reunión.
07. El sistema y usuario deben ser capaces de normalizar el audio extraído de la grabación de reunión.
08. El sistema y usuario deben ser capaces de extraer un frame determinado en formato de imagen a partir del video de reunión.
09. El sistema y usuario deben ser capaces de detectar los segmentos de cambios de escena a partir de un video.
10. El usuario debe ser capaz de configurar a su gusto los parámetros de separar el audio del video, reducción de ruido y normalización de un audio, extracción de frames y detección de segmentos de cambios de escena en un video.
11. El sistema y usuario deben ser capaces de realizar la transcripción y diarización de un audio mediante Amazon Transcribe.
12. El usuario debe ser capaz de indicar un vocabulario personalizado que debe ser usado en la transcripción de audio mediante Amazon transcribe.
13. El sistema debe ser capaz de extraer el nombre de los “speakers” identificados mediante diarización, a través de la interfaz de la grabación de la reunión de Microsoft Teams.
14. El sistema debe ser capaz de corregir la transcripción de reuniones mediante LLMs.
15. El sistema debe enviar la transcripción a un modelo LLM en Amazon Bedrock (Claude Sonnet u otro) para generar el contenido del acta de reunión.
16. El sistema y usuario deben ser capaz de usar un LLM de verificación para aumentar la confiabilidad del acta generada.
17. El sistema debe ser capaz de generar un documento PDF bien estructurado y formal con el acta de reunión.

18. El sistema debe crear una nueva sesión en el asistente inteligente empresarial AIDA y cargar su contexto con la transcripción de la reunión.
19. El sistema debe enviar por correo electrónico (a través de Outlook) el acta de reunión en formato PDF y mostrar un botón que redirija al asistente AIDA, con la nueva sesión con el contexto de la reunión, para que el usuario pueda realizar las preguntas necesarias de manera cómoda e interactiva.
20. El sistema debe notificar al usuario en caso de errores en cualquier punto del pipeline.
21. El sistema debe registrar logs de las operaciones clave para auditoría y diagnóstico.

4.1.2 Diagrama de casos de uso

A continuación, se muestra el diagrama de casos de uso del sistema a elaborar.

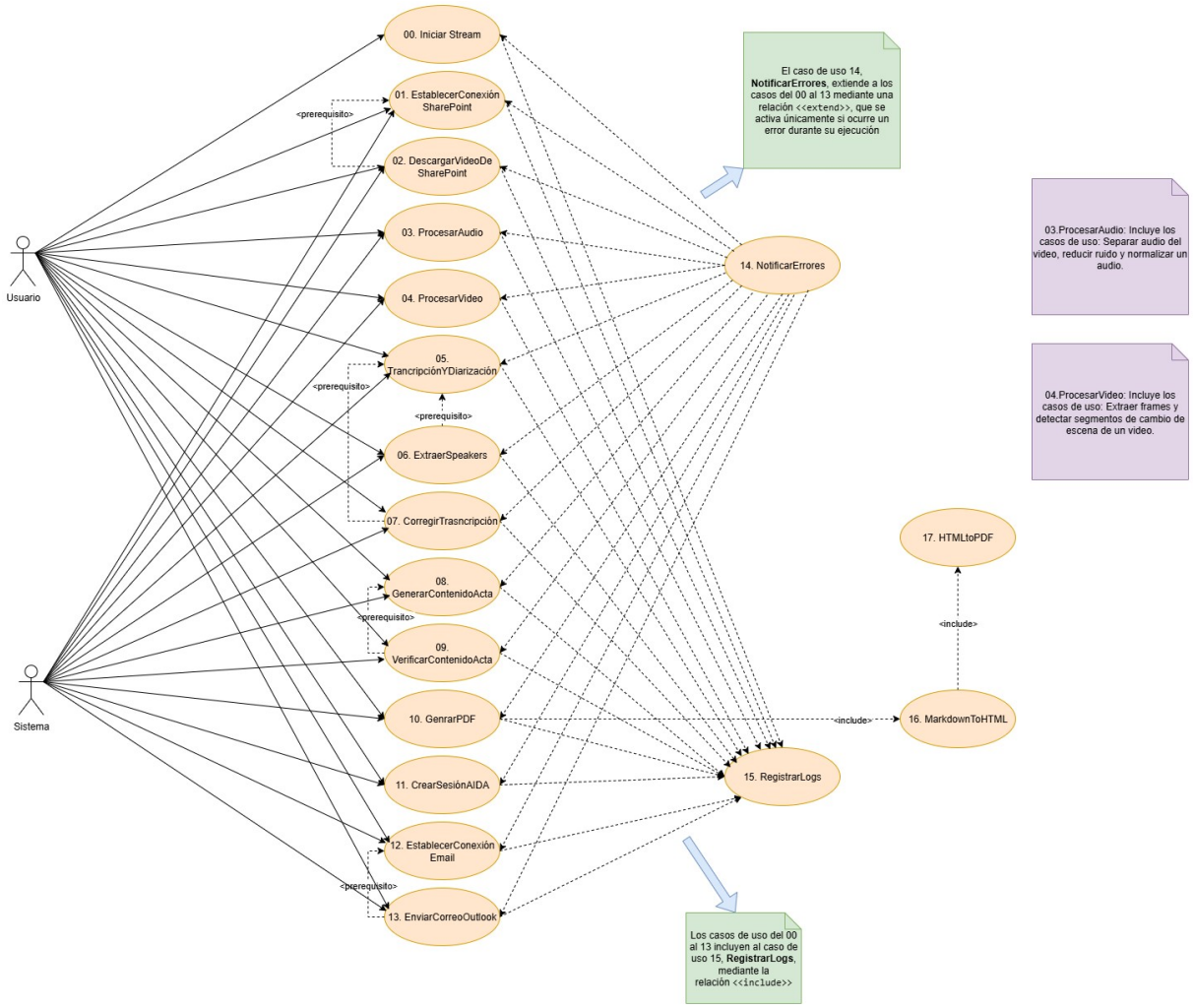


Figura 3. Diagrama de casos de uso

Se puede observar que hay dos actores: el sistema y el usuario.

Ambos pueden realizar las mismas acciones, con excepción del caso de uso *00.IniciarStream*, que solo puede ser ejecutado por el usuario.

Esto se debe a que las funcionalidades desarrolladas permiten al usuario utilizarlas de forma independiente, es decir, puede realizar tareas como el procesamiento de audio sin necesidad de que estén integradas en el flujo completo de generación del acta de reunión.

Sin embargo, el usuario también puede ejecutar el caso de uso *00. IniciarStream*, que inicia un flujo automático para generar el acta de reunión sin requerir intervención en las fases posteriores. En este caso, el sistema se encarga de ejecutar internamente todos los pasos necesarios del procesamiento.

Por esta razón, todos los casos de uso pueden ser ejecutados por el sistema si forman parte de un flujo automático (como el de generación de actas), o bien pueden ser invocados directamente por el usuario de forma puntual si así lo requiere.

Cuando se habla de flujo automático, se hace referencia a los streams de DOCs que se han introducido en el apartado *3.2 Streams de DOCs*.

4.1.3 Especificación textual de los casos de uso

Caso de uso 00. IniciarStream

Resumen de la funcionalidad: Iniciar la ejecución de un flojo automático.

Actor: Usuario

Postcondición: El sistema ejecuta el stream especificado por el usuario y entrega la respuesta.

Proceso normal principal:

1. Encolar la petición de la ejecución del stream.
2. Ejecutar el stream.
3. Se ejecuta el caso de uso *15. RegistrarLogs*.
4. Entregar la respuesta del stream al usuario.

Alternativas de proceso y excepciones:

5a. Si se produce algún error durante el proceso, el sistema ejecuta el caso de uso *14. NotificarError*.

Caso de uso 01. EstablecerConexiónSharePoint

Resumen de la funcionalidad: Función encargada de establecer la conexión a la aplicación de Azure AD que da acceso a SharePoint.

Parámetros de salida: Objeto de la conexión a SharePoint.

Actor: Usuario y sistema (si el caso de uso forma parte de un flujo automático).

Postcondición: Conexión a SharePoint establecida, con el token adecuado.

Proceso normal principal:

1. Se configura una aplicación cliente confidencial con msal para autenticarse en Azure Active Directory y obtener tokens de acceso a APIs de Microsoft.
2. Se solicita el token de acceso con la aplicación configurada anteriormente.
3. Se obtiene el token.

4. Se ejecuta el caso de uso *15. RegistrarLogs*.

Alternativas de proceso y excepciones:

5a. Si se produce algún error durante el proceso, el sistema ejecuta el caso de uso *14. NotificarError*.

Caso de uso 02. DescargarVideoDeSharePoint

Resumen de la funcionalidad: Descarga de un archivo video des de SharePoint.

Parámetros de entrada: Objeto de conexión a SharePoint.

Parámetros de salida: Archivo de video descargado.

Actor: Usuario y sistema (si el caso de uso forma parte de un flujo automático).

Precondición: Haber establecido la conexión a SharePoint.

Proceso normal principal:

1. Solicitar metadatos del archivo para realizar validaciones.
2. Una vez validado, se solicitan los bytes del archivo.
3. Se ejecuta el caso de uso *15. RegistrarLogs*.
4. Se devuelve el archivo descargado.

Alternativas de proceso y excepciones:

5a. Si se produce algún error durante el proceso, el sistema ejecuta el caso de uso *14. NotificarError*.

Caso de uso 03. ProcesarAudio

Resumen de la funcionalidad: Procesamiento de archivos audio.

Parámetros de entrada: Archivo multimedia a procesar.

Parámetros de salida: Archivo multimedia procesado.

Actor: Usuario y sistema (si el caso de uso forma parte de un flujo automático).

Proceso normal principal:

1. Almacenamiento temporal del archivo multimedia para su procesamiento.
2. Realizar el procesamiento solicitado.
3. Se ejecuta el caso de uso *15. RegistrarLogs*.
4. Devolver archivo procesado.

Alternativas de proceso y excepciones:

5a. Si se produce algún error durante el proceso, el sistema ejecuta el caso de uso *14. NotificarError*.

Caso de uso 04. ProcesarVideo

Resumen de la funcionalidad: Procesamiento de video.

Parámetros de entrada: Archivo multimedia a procesar.

Parámetros de salida: Archivo multimedia procesado.

Actor: Usuario y sistema (si el caso de uso forma parte de un flujo automático).

Proceso normal principal:

1. Almacenamiento temporal del archivo multimedia para su procesamiento.
2. Realizar el procesamiento solicitado.
3. Se ejecuta el caso de uso *15. RegistrarLogs*.
4. Devolver archivo procesado.

Alternativas de proceso y excepciones:

5a. Si se produce algún error durante el proceso, el sistema ejecuta el caso de uso *14. NotificarError*.

Caso de uso 05. TrancripciónYDiarización

Resumen de la funcionalidad: Transcripción y diarización de un audio.

Parámetros de entrada: Audio a transcribir. En el caso de mi flujo el audio a transcribir es uno recibido desde el caso de uso que se encarga del procesamiento audio.

Parámetros de salida: Texto transcrito junto a las marcas que indican la diarización.

Actor: Usuario y sistema (si el caso de uso forma parte de un flujo automático).

Proceso normal principal:

1. Almacenamiento temporal del audio a transcribir.
2. Configuración del servicio de transcripción y diarización.
3. Ejecutar transcripción y diarización.
4. Recuperar y devolver resultados, además de ejecutar el caso de uso *15. RegistrarLogs*.

Alternativas de proceso y excepciones:

5a. Si se produce algún error durante el proceso, el sistema ejecuta el caso de uso *14. NotificarError*.

Caso de uso 06. ExtraerNombreSpeakers

Resumen de la funcionalidad: Extraer nombre de los hablantes indicados por la diarización de la interfaz visual de Microsoft Teams.

Parámetros de entrada: Transcripción y diarización y video de la grabación de reunión.

Parámetros de salida: Nombre de hablantes.

Actor: Usuario y sistema (si el caso de uso forma parte de un flujo automático).

Precondición: Transcripción y diarización realizadas.

Proceso normal principal:

1. Preparar los datos de hablantes.
2. Buscar el nombre del hablante en los segmentos donde participa dentro de la interfaz de Microsoft Teams, repitiendo el proceso en bucle hasta localizarlo.
3. Se ejecuta el caso de uso *15. RegistrarLogs*.

Alternativas de proceso y excepciones:

4a. Si se produce algún error durante el proceso, el sistema ejecuta el caso de uso *14. NotificarError*.

Caso de uso 07. CorregirTrasncripción

Resumen de la funcionalidad: Corrección gramatical, ortográfica, fonética y de puntuación en la transcripción.

Parámetros de entrada: Trascripción de un audio.

Parámetros de salida: La misma transcripción de la entrada corregida.

Actor: Usuario y sistema (si el caso de uso forma parte de un flujo automático).

Precondición: Transcripción realizada.

Postcondición: Transcripción corregida.

Proceso normal principal:

1. Configuración del LLM a usar indicando que se busca hacer una corrección gramatical, ortográfica, fonética y de puntuación.
2. Corregir de manera automática la trascripción pasada al LLM.
3. Se ejecuta el caso de uso *15. RegistrarLogs*.
4. Devolver la transcripción corregida.

Alternativas de proceso y excepciones:

5a. Si se produce algún error durante el proceso, el sistema ejecuta el caso de uso *14. NotificarError*.

Caso de uso 08. GenerarContenidoActa

Resumen de la funcionalidad: Generación automática del contenido del acta.

Parámetros de entrada: Texto e información de base para la generación del acta. En el caso de mi proyecto, se utiliza la transcripción junto con la diarización, los nombres de los participantes y la información relevante extraída de la compartición de pantalla.

Parámetros de salida: Contenido del acta.

Actor: Usuario y sistema (si el caso de uso forma parte de un flujo automático).

Proceso normal principal:

1. Configuración del LLM.

2. Procesamiento y elaboración del acta mediante el LLM a partir de la información de entrada.
3. Se ejecuta el caso de uso *15. RegistrarLogs*.
4. Devolución del contenido del acta.

Alternativas de proceso y excepciones:

5a. Si se produce algún error durante el proceso, el sistema ejecuta el caso de uso *14. NotificarError*.

Caso de uso 09. VerificarContenidoActa

Resumen de la funcionalidad: Verificación de la fiabilidad y exactitud del contenido del acta generado por el LLM.

Parámetros de entrada: Contenido del acta e información de referencia proporcionada al LLM para la generación del acta.

Parámetros de salida: Valor de confianza entre 0 y 1 y un análisis.

Actor: Usuario y sistema (si el caso de uso forma parte de un flujo automático).

Precondición: Contenido del acta generado.

Proceso normal principal:

1. Procesamiento y comparación de los datos de entrada para su validación.
2. Devolución del valor de confianza y análisis correspondiente.
3. Se ejecuta el caso de uso *15. RegistrarLogs*.

Alternativas de proceso y excepciones:

4a. Si se produce algún error durante el proceso, el sistema ejecuta el caso de uso *14. NotificarError*.

Caso de uso 10. GenrarPDF

Resumen de la funcionalidad: Generar un archivo PDF a partir del contenido en Markdown.

Parámetros de entrada: Contenido del PDF.

Parámetros de salida: PDF generado.

Actor: Usuario y sistema (si el caso de uso forma parte de un flujo automático).

Proceso normal principal:

1. Se ejecuta el caso de uso *16. MarkdownToHTML*.
2. Se ejecuta el caso de uso *15. HTMLToPDF*.
3. Agregar el pie de página si el usuario ha indicado uno.
4. Se ejecuta el caso de uso *15. RegistrarLogs*.

Alternativas de proceso y excepciones:

5a. Si se produce algún error durante el proceso, el sistema ejecuta el caso de uso *14. NotificarError*.

Caso de uso 11. CrearSesiónAIDA

Resumen de la funcionalidad: Crear sesión en el asistente inteligente de AIDA.

Actor: Usuario y sistema (si el caso de uso forma parte de un flujo automático).

Proceso normal principal:

1. Obtención del token de verificación del usuario.
2. Validar con llamadas a la API del middle de AIDA que el usuario tiene permiso de realizar la acción.
3. Llamada a crear sesión de AIDA, con el título correspondiente.
4. Si el usuario ha indicado un fichero de contexto, se enlaza a la sesión.
5. Se ejecuta el caso de uso *15. RegistrarLogs*.

Alternativas de proceso y excepciones:

6a. Si se produce algún error durante el proceso, el sistema ejecuta el caso de uso *14. NotificarError*.

Caso de uso 12. EstablecerConexiónEmail

Resumen de la funcionalidad: Configurar la conexión al servicio del correo.

Parámetros de salida: Objeto de conexión al email.

Actor: Usuario y sistema (si el caso de uso forma parte de un flujo automático).

Proceso normal principal:

1. Realizar configuración del servicio de correo (IMAP o SMTP) y ajustar el puerto adecuado.
2. Se ejecuta el caso de uso *15. RegistrarLogs*.
3. Devolver el objeto de conexión al email.

Alternativas de proceso y excepciones:

4a. Si se produce algún error durante el proceso, el sistema ejecuta el caso de uso *14. NotificarError*.

Caso de uso 13. EnviarCorreoOutlook

Resumen de la funcionalidad: Envío de un correo a un usuario mediante Outlook.

Parámetros de entrada: Objeto de conexión al email.

Actor: Usuario y sistema (si el caso de uso forma parte de un flujo automático).

Precondición: Haber establecido la conexión al correo electrónico.

Postcondición: Correo enviado.

Proceso normal principal:

1. Adjuntar los archivos al correo, si existen.
2. Establecer una conexión segura con el servicio SMTP para envío de correos.
3. Enviar el correo utilizando el servicio SMTP.
4. Cerrar la conexión con el servicio.
5. Se ejecuta el caso de uso *15. RegistrarLogs*.

Alternativas de proceso y excepciones:

6a. Si se produce algún error durante el proceso, el sistema ejecuta el caso de uso *14. NotificarError*.

Caso de uso 14. NotificarErrores

Resumen de la funcionalidad: Informe de manera inmediata sobre cualquier fallo ocurrido durante la ejecución de los procesos.

Parámetros de entrada: Error ocurrido.

Actor: El del caso de uso principal.

Precondición: Se ha producido un error durante la ejecución de alguna operación, lo que provoca la invocación de este caso de uso.

Postcondición: Ejecución detenida.

Proceso normal principal:

1. Se genera un registro del error en los logs internos del sistema.
2. Se construye la notificación de error, incluyendo los datos capturados.
3. El sistema envía la notificación del error.

Caso de uso 15. RegistrarLogs

Resumen de la funcionalidad: Guardar los registros de la operación.

Actor: El del caso de uso principal.

Precondición: Se ha ejecutado alguna operación que invoca el caso de uso.

Postcondición: Los logs registrados en el sistema.

Proceso normal principal:

1. Añadir los registros a un diccionario.
2. Guardado de los registros en el sistema.

Caso de uso 16. MarkdownToHTML

Resumen de la funcionalidad: Convertir el contenido Markdown a HTML.

Actor: El del caso de uso principal.

Proceso normal principal:

1. Se define una plantilla CSS con el estilo de HTML deseado.
2. Convertir el contenido en formato Markdown a HTML utilizando las librerías correspondientes, prestando especial atención al procesamiento de elementos visuales, como diagramas y aplicando la plantilla CSS para el formato final.

Caso de uso 17. HTMLToPDF

Resumen de la funcionalidad: Convertir un archivo HTML a PDF.

Actor: El del caso de uso principal.

Proceso normal principal:

1. Convertir el archivo HTML a PDF mediante la librería adecuada.

4.2 Requisitos no funcionales

Requisitos de seguridad:

- Toda la comunicación entre servicios debe estar cifrada (HTTPS, TLS).
- Las credenciales del usuario no deben ser almacenadas; se usará autenticación basada en tokens si es posible.
- Los videos y transcripciones deben eliminarse automáticamente después de ser procesados y enviados.

Requisitos de escalabilidad:

- El sistema debe escalar horizontalmente para manejar múltiples solicitudes simultáneas. (Esto se logra gracias al uso de EC2 y S3)
- La arquitectura debe permitir integración futura con otras fuentes de video (Teams, OneDrive, etc.).

Requisitos de rendimiento:

- El tiempo total de procesamiento (descarga + transcripción + resumen + envío) no debe exceder los 5 minutos para videos de hasta 30 minutos.
- La latencia del resumen (del LLM) debe ser menor a 1 minuto por cada 30 minutos de transcripción.

Mantenibilidad y monitoreo:

- El sistema debe permitir un monitoreo de métricas clave.
- El código debe estar documentado y tener pruebas unitarias para al menos el 80% de cobertura.

5 Análisis de los requisitos funcionales

Debido a que el proyecto ha sido construido bajo una arquitectura basada en grafos, donde cada nodo representa una función específica que actúa como unidad de procesamiento dentro de un flujo automático, no es posible representar todos los casos de uso en un único diagrama de clases y a veces se complica su representación en un diagrama de secuencias.

Este enfoque implica que muchas funcionalidades se implementan como funciones independientes, conectadas dinámicamente según el flujo del grafo, y no responden a una estructura tradicional orientada a objetos ni a una secuencia lineal de eventos. Por ello, se ha optado por mostrar únicamente los diagramas correspondientes a casos de uso representativos o relevantes, entendiendo que cada nodo puede funcionar de forma autónoma y estar involucrado en múltiples flujos distintos.

Cada funcionalidad que ofrece el sistema ha sido desarrollada como un módulo independiente, lo que permite representar su estructura mediante un diagrama de clases correspondiente. Sin embargo, existen otras funcionalidades cuya lógica no reside en un único módulo, sino que se construyen mediante la composición o anidación de nodos dentro del grafo que define el flujo de ejecución.

En el caso concreto de este TFG, se han implementado de forma modular el procesamiento multimedia, el envío de correos electrónicos y la conexión y descarga de videos desde SharePoint, lo que permite representarlos mediante sus respectivos diagramas de clases y de secuencias.

El resto de las funcionalidades, al estar distribuidas en el flujo dinámico del grafo, se documentan en el apartado de *diseño*, donde se detalla el funcionamiento del stream, es decir, la arquitectura basada en grafos que permite la ejecución automática del sistema.

5.1 Diagrama de clases

5.1.1 Módulo de procesamiento multimedia

En esta sección se presenta y explica el diagrama de clases correspondiente al módulo de procesamiento de audio y vídeo.

Este módulo integra dos tecnologías principales, AWS y FFmpeg, para ofrecer un conjunto de estrategias y utilidades especializadas que permiten el procesamiento multimedia de manera flexible y eficiente. A continuación, se describen las clases y componentes clave que conforman este sistema:

- **MediaProcessorFactory** es una clase responsable de devolver implementaciones concretas de estrategias en función del proveedor seleccionado. Los proveedores disponibles están definidos en el enumerado **ProcessorType**, que actualmente incluye dos opciones: FFmpeg y AWS.
- **Interfaces (Estrategias):** El sistema define cinco interfaces, cada una correspondiente a una estrategia diferente: AudioSeparationStrategy, NoiseReductionStrategy, AudioNormalizationStrategy, FrameExtractionStrategy y SceneDetectionStrategy. Cada estrategia cuenta con una implementación utilizando FFmpeg y otra con AWS, excepto la reducción de ruido, que solo dispone de implementación en FFmpeg, ya que AWS no proporciona esta funcionalidad.

- **Modelo de Datos:** La clase FileData que encapsula metadatos y datos binarios de un archivo que permite la entrada/salida estandarizada para todas las estrategias. Por otra parte, la clase AudioConfig unifica la configuración base del audio utilizada por las estrategias de procesamiento de audio.
- **Utilidades Técnicas:** Las clases FileUtils, FFmpegUtils, AWSUtils, TimeUtils y ValidationUtils encapsulan funciones comunes y técnicas, como ejecutar comandos FFmpeg, gestionar servicios AWS, validar parámetros, manejar archivos temporales y normalizar tiempos. Esto evita duplicación, mejora la organización y facilita el mantenimiento del código.

  Estas clases están diseñadas para ofrecer funcionalidades a través de métodos estáticos, lo que permite utilizarlas directamente sin necesidad de instanciar objetos.

A continuación, se muestra el diagrama de clases de dicho módulo, en el cual se han empleado las dos tecnologías distintas para llevar a cabo las operaciones:

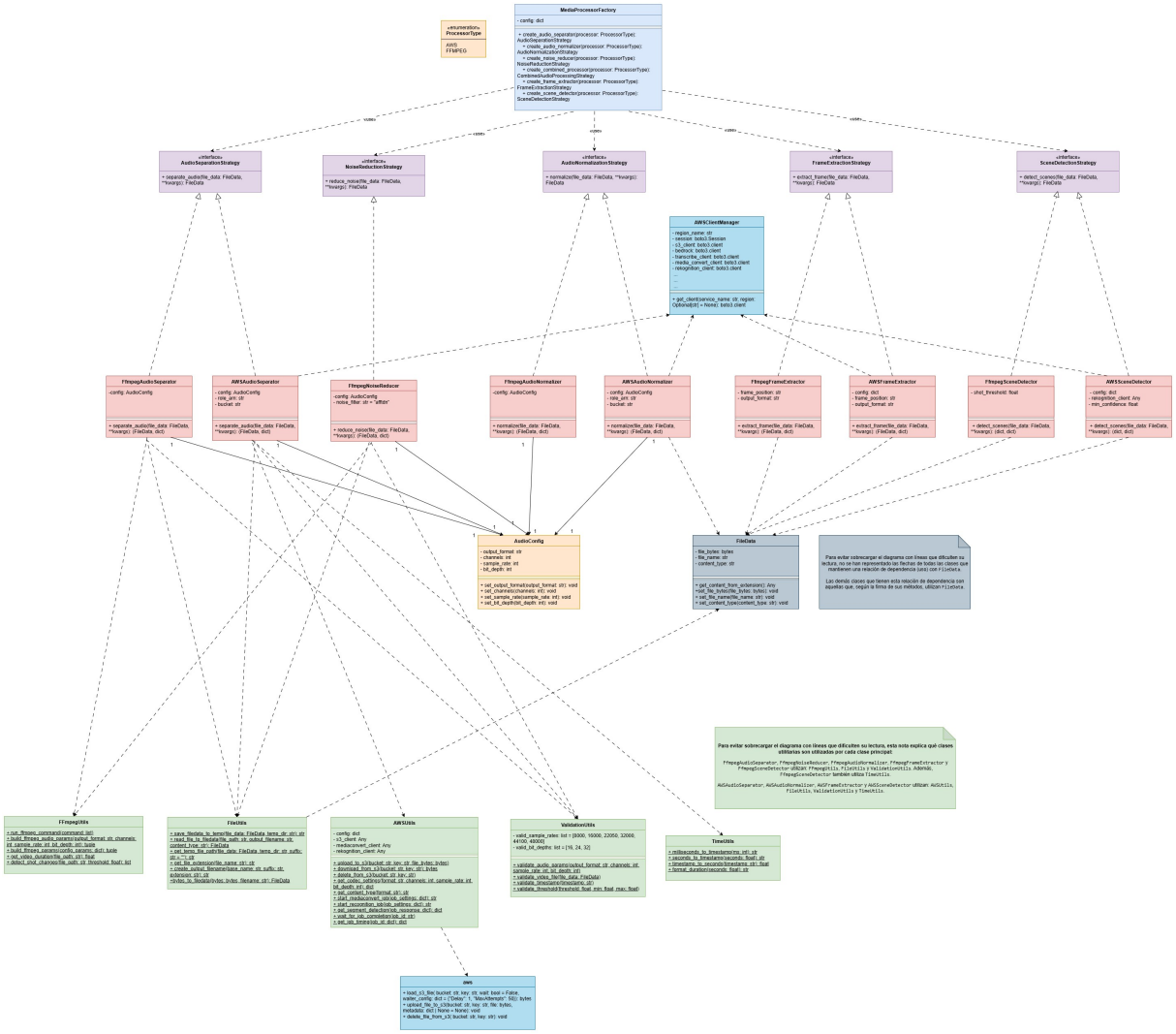


Figura 4. Diagrama de clases del módulo de procesamiento multimedia

Debido a la arquitectura de DOCs basada en grafos, la mayoría de las relaciones son dependencias de uso porque las clases interactúan principalmente invocando funcionalidades específicas de otras clases, sin mantener referencias constantes ni estados compartidos. Esta

dinámica favorece la flexibilidad, escalabilidad y testabilidad, al evitar asociaciones permanentes y permitir que los componentes funcionen de manera autónoma y reemplazable. Así, las dependencias de uso reflejan la naturaleza transitoria y orientada a la ejecución puntual típica de un grafo, en lugar de relaciones estructurales fijas.

Por ejemplo, la clase de datos FileData se pasa únicamente como parámetro en las funciones y se devuelve tras su procesamiento o creación, pero nunca se almacena como un atributo dentro de la clase que la utiliza.

Este diseño se fundamenta en sólidos principios de ingeniería de software para afrontar el reto de ejecutar operaciones multimedia con distintos proveedores (como AWS, FFmpeg y futuros como Azure). Se apoya en patrones como Strategy y Abstract Factory, junto con la aplicación de principios clave para garantizar flexibilidad, escalabilidad y un mantenimiento eficiente.

Las ventajas del diseño y la disposición de las clases que se observan en el diagrama son las siguientes:

Patrón Strategy que aporta flexibilidad y desacoplamiento:

- Cada tipo de operación (normalización, separación, detección de escenas...) se define como una interfaz, con distintas implementaciones concretas (por ejemplo, AWSSceneDetector, FfmpegSceneDetector).
- **Intercambio dinámico**: Cambio en tiempo real entre AWS/FFmpeg mediante interfaces comunes (AudioSeparationStrategy, SceneDetectionStrategy…), lo que permite cambiar de backend sin alterar el cliente.
- **Favorece la extensibilidad:** Añadir nuevos proveedores solo requiere implementar la interfaz sin modificar el código base.
- Aplica el **Principio Abierto/Cerrado** (OCP) debido a que la estructura es abierta a la extensión, pero cerrada a la modificación. Es decir, se pueden añadir nuevas funcionalidades sin alterar el código existente.

Abstract Factory que aporta creación unificada:

- MediaProcessorFactory centraliza la creación de estrategias según el proveedor elegido (ProcessorType).
- Facilita el mantenimiento al separar creación y uso de objetos.
- Es fácilmente ampliable a nuevos entornos (Azure, GCP...).
- Inversión de dependencias: Clientes interactúan solo con interfaces.

Utilidades especializadas, reutilización y SRP[20] que aportan cohesión operativa:

- Clases como FFmpegUtils, AWSUtils, FileUtils, TimeUtils y ValidationUtils encapsulan tareas transversales: Ejecución de comandos, validación de parámetros, manejo de archivos temporales, etc.
- Cada clase tiene una única responsabilidad (SRP), lo que facilita su testeo y reutilización.

[20] Single Responsibility Principle

Modelo de datos unificado que aporta simplicidad y contrato estable:

- La clase FileData encapsula los datos de entrada y salida y es usada por todas las estrategias.
- Este contrato común simplifica interfaces, reduce errores y mejora interoperabilidad.
- Esta ventaja también se aplica a la clase AudioConfig.

Desacoplamiento técnico mediante interfaces y dependencias flexibles:

- Las estrategias están desacopladas de implementaciones concretas gracias al uso de interfaces y dependencias utilitarias externas que se invocan de forma directa, sin acoplamiento fuerte ni herencia.
- Mantenimiento localizado: Los cambios en una estrategia o proveedor no afectan al resto del sistema.
- Facilidad de pruebas: Al estar desacopladas, las estrategias se pueden testear usando mocks o stubs.
- Aislamiento de cambios: Actualización SDK AWS solo afecta AWSUtils.

Validación centralizada: Prevención de costos debido a que rechaza operaciones inválidas antes de ejecución.

Escalabilidad y extensibilidad natural: Añadir un nuevo proveedor implica implementar las interfaces (AudioNormalizer, SceneDetector, etc.) y añadir soporte en la fábrica, sin tocar el código ya existente.

Paralelización y aislamiento:

- Las estrategias son independientes, lo que permite su ejecución concurrente en flujos paralelos (ideal para pipelines multimedia).
- Este enfoque es idóneo para sistemas construidos como grafos como el caso de DOCs, donde cada nodo ejecuta una función puntual y autónoma.

La siguiente tabla presenta de manera concisa y organizada las principales ventajas del diseño de las clases descritas.

Categoría	Ventaja técnica
Flexibilidad	Sustituir proveedor sin tocar lógica del cliente (Strategy + Interfaces)
Cohesión	Utilidades con una sola responsabilidad (SRP)
Desacoplamiento	Interfaces + Dependencias flexibles en lugar de herencia
Escalabilidad y extensibilidad (OCP)	Soporte fácil para nuevos proveedores o estrategias
Testabilidad	Clases pequeñas, desacopladas y mockeables
Robustez	Validaciones previas evitan ejecuciones fallidas o inconsistentes
Interoperabilidad	FileData como contrato común entre todas las estrategias

Tabla 4. Ventajas diseño de clases módulo de procesamiento multimedia

Este diseño demuestra una arquitectura **moderna, desacoplada, flexible y escalable**, alineada con los principios y buenas prácticas de diseño. Es especialmente adecuada para sistemas modulares y multi-proveedor, como entornos de procesamiento multimedia que combinan tecnologías cloud y locales. Gracias a su estructura clara y su extensibilidad, es una base sólida para futuros crecimientos, integraciones o adaptaciones.

5.1.2 Módulo de gestión de correos electrónicos

En esta sección se presenta y explica el diagrama de clases correspondiente al módulo de gestión de correos electrónicos. Este diagrama es mucho más sencillo debido a que este módulo no requiere poseer muchas funcionalidades.

A continuación, se describen las clases y componentes clave que conforman este sistema:

- **FileData:** La misma clase explicada en el punto anterior.
- **EmailConnector:** Gestiona la conexión y comunicación con servidores IMAP y SMTP para enviar y recibir correos.

  Las ventajas de esta clase son que encapsula toda la lógica relacionada con la conexión y el manejo de protocolos de correo, facilita la reutilización y centralización del envío de emails, y permite la configuración dinámica de los servidores IMAP y SMTP según el dominio.
- **EmailFunctions:** Funciones utilitarias que usan EmailConnector para enviar correos, integrándose en flujos o procesos externos. Su ventaja es la clara separación entre la lógica de conexión (EmailConnector) y la lógica de uso o flujo (EmailFunctions), lo que facilita la integración y reutilización en distintos contextos sin modificar EmailConnector.

A continuación, se muestra el diagrama de clases de dicho módulo:

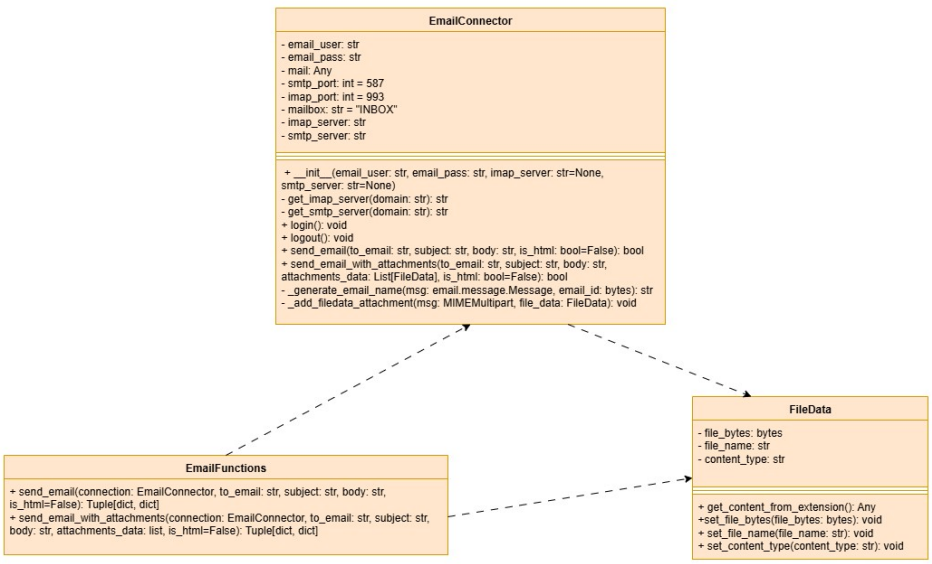


Figura 5. Diagrama de clases del módulo de gestión de correos electrónicos

En este diseño se utiliza el patrón **Facade**, donde EmailConnector actúa como una fachada que oculta la complejidad de los protocolos SMTP e IMAP. EmailConnector simplifica el manejo de estos protocolos al ofrecer una interfaz unificada para enviar correos,

y en el futuro puede extenderse para incluir funciones de recepción y lectura de correos. Esto permite que otras partes del sistema interactúen con el correo sin necesidad de conocer los detalles internos de su implementación.

Además, el hecho de que los cambios en los protocolos de email solo afecten a EmailConnector favorece la mantenibilidad y refleja un buen nivel de desacoplamiento.

5.1.3 Módulo de SharePoint

En esta sección se presenta y explica el diagrama de clases correspondiente al módulo de SharePoint.

Este diagrama de clases presenta la misma estructura que el del módulo anterior, utilizando el mismo patrón de diseño y, por lo tanto, conservando las ventajas explicadas en el módulo de gestión de correos electrónicos.

A continuación, se describen las clases y componentes clave que conforman este sistema:

- **FileData:** La misma clase explicada en los puntos anteriores.
- **SharePointConnector:** Clase cliente especializada que gestiona la conexión y autenticación con Graph API. Su función principal es descargar archivos de SharePoint de manera segura. La clase incorpora los parámetros department y project para contextualizar y ayudar a identificar la ubicación específica de los archivos dentro de la estructura de SharePoint. Maneja automáticamente la renovación de tokens de acceso y proporciona un robusto manejo de errores para distintos códigos de estado HTTP.
- **SharePointFunctions:** Clase de funciones de servicio u operaciones. Utiliza una instancia de SharePointConnector para realizar operaciones específicas de negocio (en este caso, la ingesta de videos). Se encarga de la lógica de alto nivel, incluyendo el registro de logs (logging), la captura de métricas operacionales y el manejo estructurado de excepciones, sirviendo como puente entre la lógica de aplicación y el conector de bajo nivel.

A continuación, se muestra el diagrama de clases de dicho módulo:

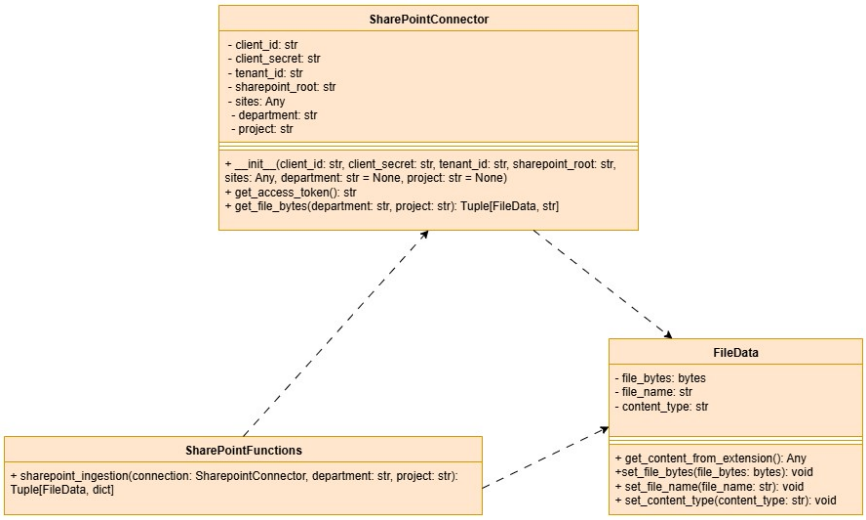


Figura 6. Diagrama de clases del módulo de SharePoint

5.2 Diagramas de secuencias

Para facilitar la representación y comprensión de los siguientes diagramas, no se muestran todos los parámetros con exactitud, ya que el objetivo principal es ilustrar la lógica del funcionamiento, las llamadas y la interacción entre las distintas entidades y módulos.

Cabe señalar que muchos de los parámetros de configuración provienen de la petición inicial realizada por el usuario al API Gateway de DOCs, la cual es posteriormente procesada por el motor de procesamiento (EC2 Engine).

En los casos de procesamiento multimedia, ya sea a partir de un video o de un archivo de audio, como con el caso de la lista de archivos a adjuntar en el envío de correo electrónico, se asume que los archivos ya han sido cargados y están disponibles en formato FileData. Estos archivos pueden obtenerse fácilmente mediante un nodo del stream responsable de la ingesta de documentos desde diversas fuentes.

Por este motivo, esta fase inicial no se representa en los diagramas, ya que no forma parte del enfoque principal de los mismos.

Es importante mencionar que, tal como se puede observar en los diagramas, al finalizar cada proceso se generan métricas que incluyen los registros (logs) correspondientes, los cuales se almacenan en el sistema para su posterior consulta. En caso de que ocurra algún error durante la ejecución, el sistema detiene inmediatamente el proceso, registra el error y actualiza el estado de la operación a FAILED en la base de datos, incluyendo una descripción del motivo del fallo.

Aunque no se representa en los diagramas el lanzamiento explícito de errores en cada posible punto de fallo, para evitar sobrecargar visualmente la representación, el sistema sí realiza un monitoreo continuo de errores en todas las fases del procesamiento. Únicamente se ha incluido en los diagramas el caso en que se lanza un error al detectar un fallo en el análisis inicial de la petición, para mostrar un caso.

De este modo, se garantiza una trazabilidad completa y una notificación precisa ante cualquier incidencia.

5.2.1 Módulo de procesamiento multimedia

5.2.1.1 Diagramas de procesamiento multimedia con FFmpeg

A continuación, se puede ver el diagrama de secuencias correspondiente a una petición para separar el audio del video mediante FFmpeg:

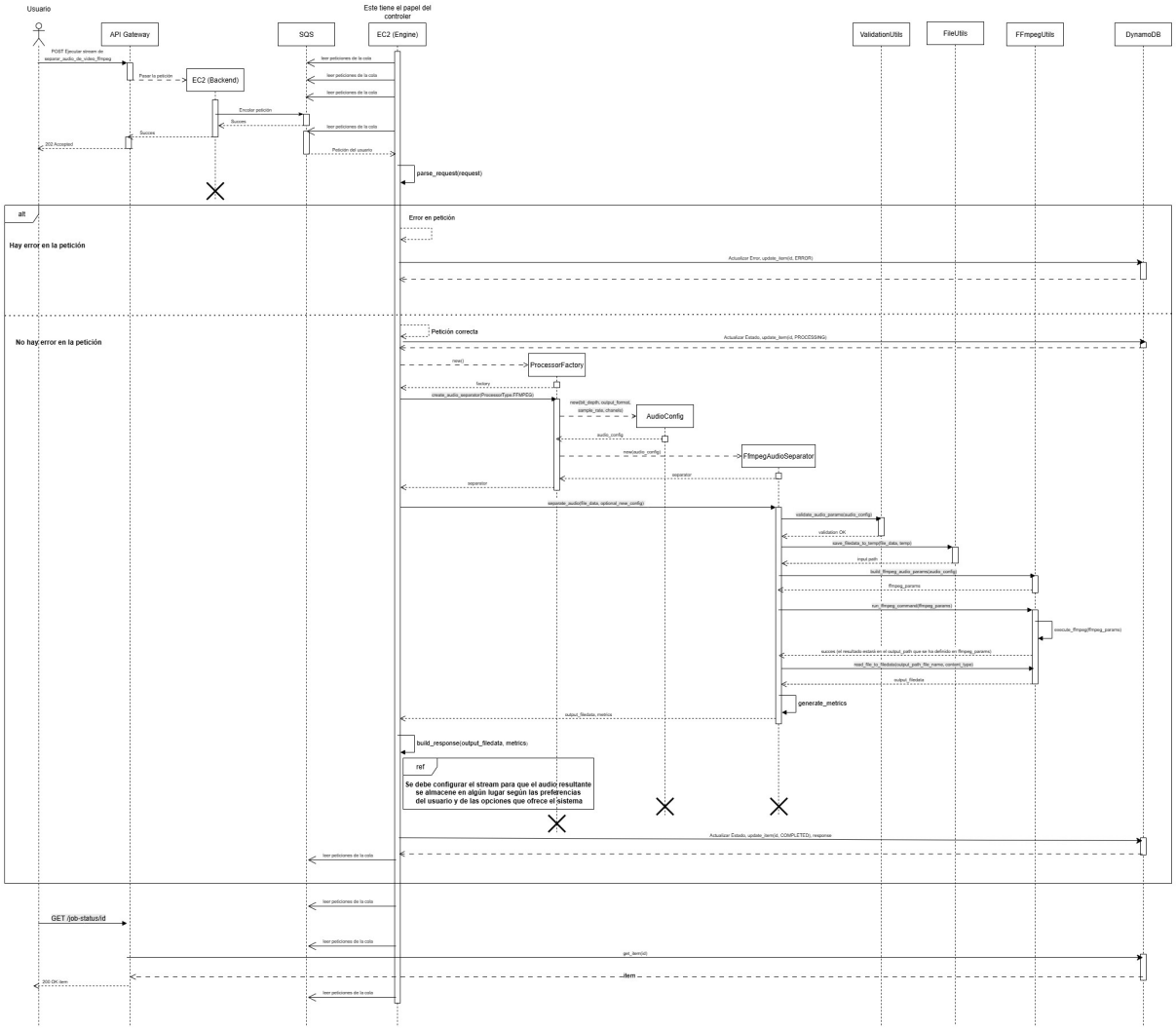


Figura 7. Diagrama de secuencias procesamiento audio FFmpeg

El resto de los diagramas correspondientes a la normalización y reducción de ruido de un audio con FFmpeg presentan el mismo comportamiento al del diagrama de secuencias de la operación de separación de audio y video, con la excepción de posibles llamadas de verificación distintas para comprobar los parámetros específicos de cada proceso. Por este motivo, no se incluyen dichos diagramas, ya que resultarían repetitivos.

A continuación, se muestra un diagrama de la extracción de un fotograma (frame) de un video. Se puede observar que el comportamiento es muy similar al del procesamiento de audio. Sin embargo, se decidió incluir un ejemplo relacionado con video para ilustrar esta similitud. Las principales diferencias radican en ciertas llamadas a funciones de verificación o a clases de utilidades específicas para el manejo de video.

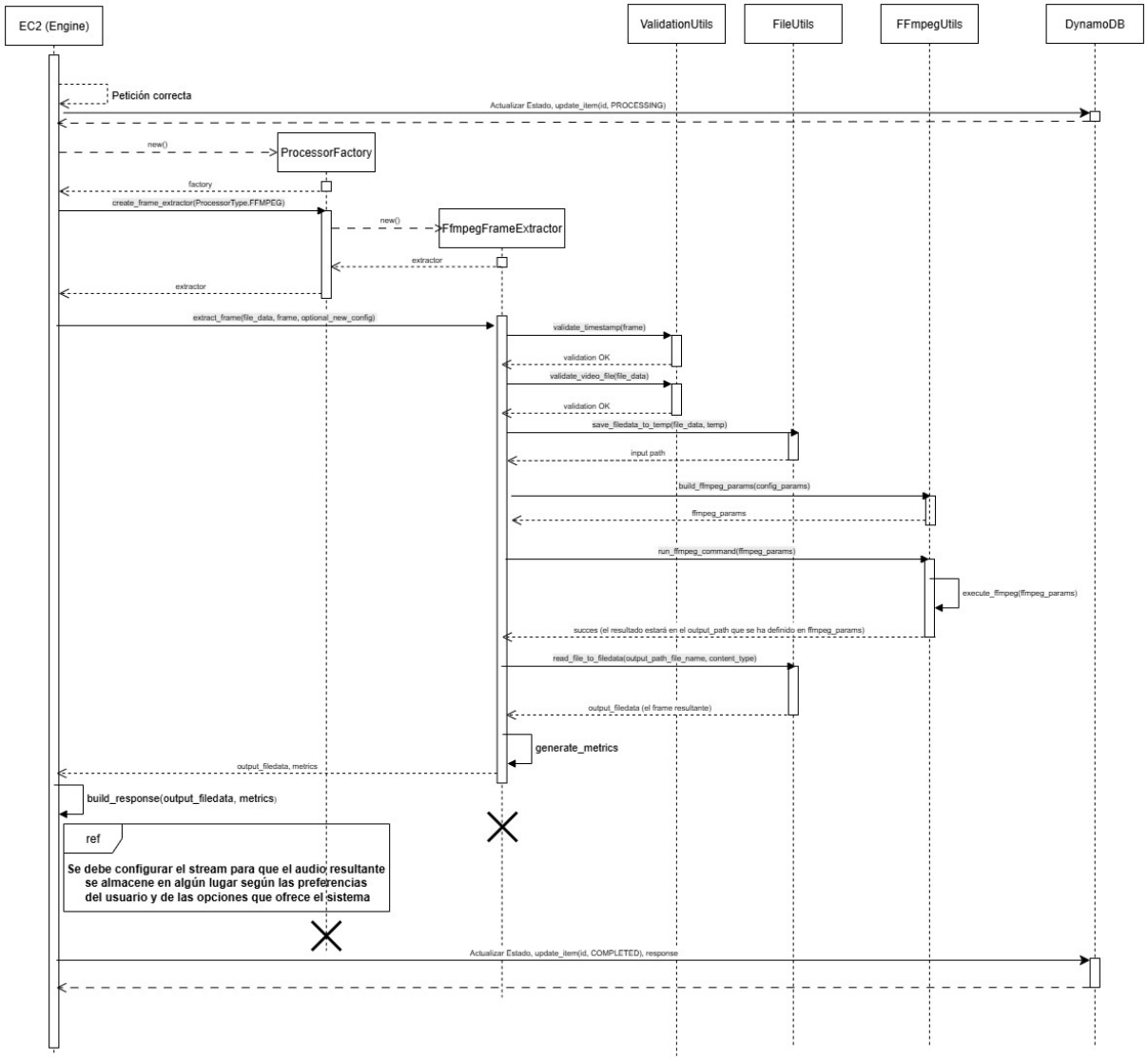


Figura 8. Diagrama de secuencias procesamiento video FFmpeg

Se puede observar en el diagrama que la lógica del sistema referente a la interacción entre API Gateway, el backend en EC2, SQS y el motor de procesamiento (EC2 Engine) no se representa en su totalidad, ya que es común al comportamiento mostrado en la operación de separación de audio y video. Por ello, únicamente se muestra el procesamiento específico de la petición una vez esta es recibida por el motor.

Con el fin de evitar repeticiones innecesarias, no se incluye el diagrama correspondiente a la detección de cambios de escena, ya que también sigue una lógica muy parecida, con la diferencia de que se utiliza la clase TimeUtils, que no se emplea en los otros casos.

5.2.1.2 Diagrama de procesamiento multimedia con AWS

A continuación, se presentan los diagramas correspondientes a las operaciones de procesamiento multimedia utilizando servicios de AWS. Con el objetivo de evitar repeticiones innecesarias, se ha optado por mostrar únicamente dos diagramas de secuencias: uno de procesamiento de audio y otro de procesamiento de video.

Para el caso del audio, se ha seleccionado la operación de normalización, ya que difiere de la operación mostrada previamente con FFmpeg. En el caso del video, se ha escogido la

detección de cambios de escena, también distinta a la operación presentada anteriormente con FFmpeg.

A continuación, se muestra el diagrama de secuencias correspondiente a la normalización de un audio mediante AWS.

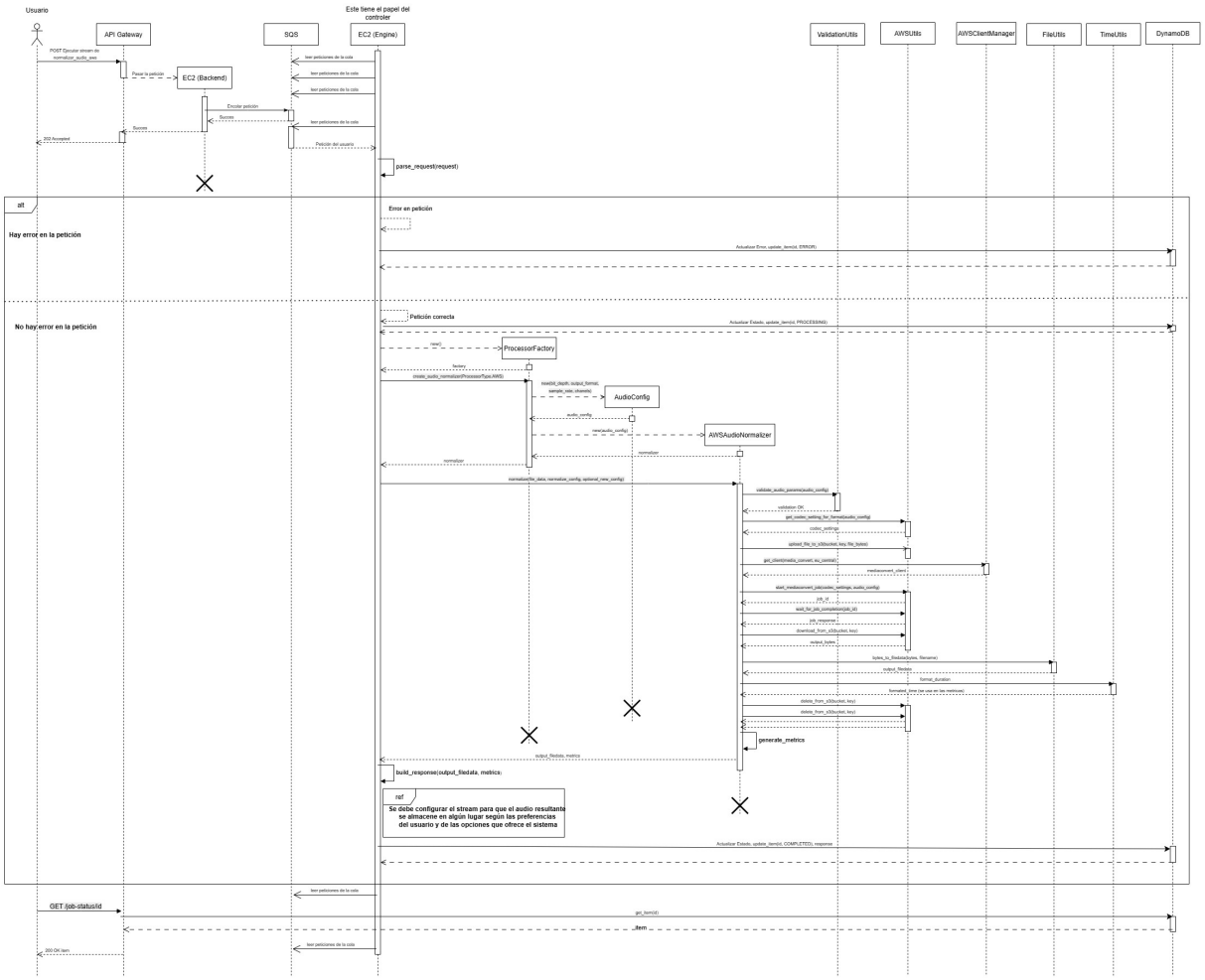


Figura 9. Diagrama de secuencias procesamiento audio AWS

Considero relevante mostrar el funcionamiento interno que se produce al crear un trabajo con AWS MediaConvert y esperar su finalización mediante un sistema de mensajes y colas.

A continuación, se presenta un diagrama de secuencias que representa esta lógica, con el fin de facilitar la comprensión del proceso descrito.

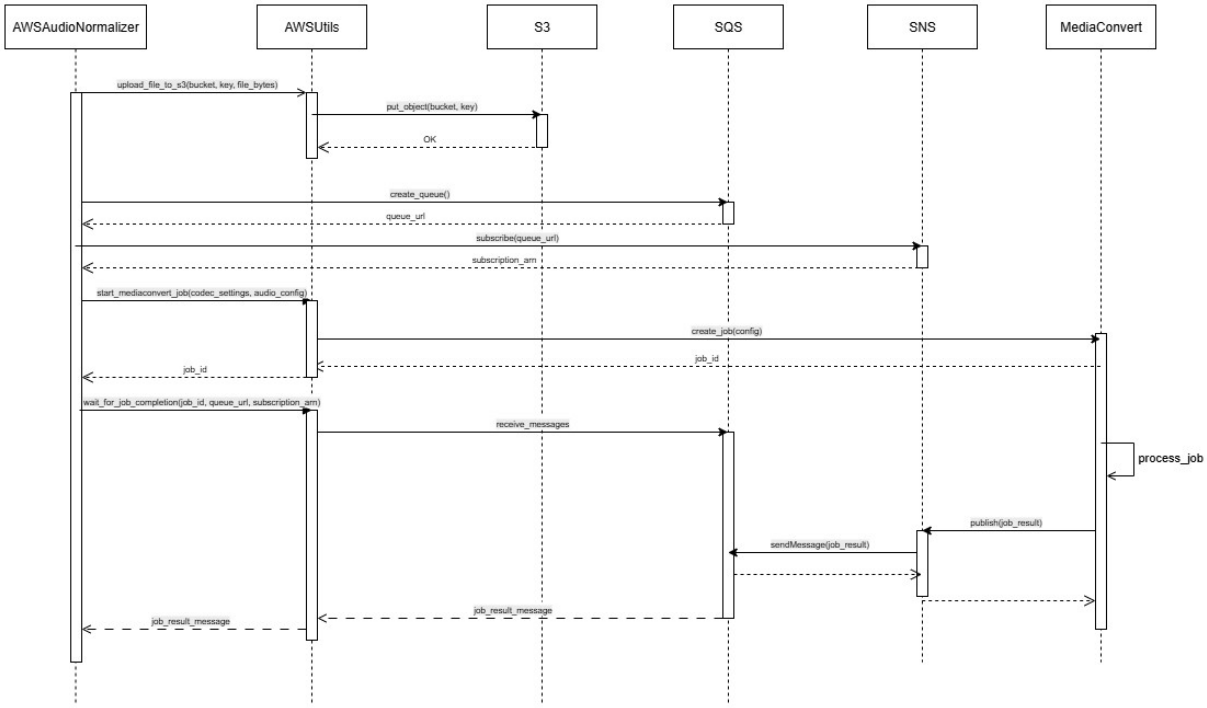


Figura 10. Diagrama de secuencias de la espera de la finalización de un trabajo de AWS con SQS y SNS

A continuación, se muestra el diagrama de detección de cambios de escena con AWS.

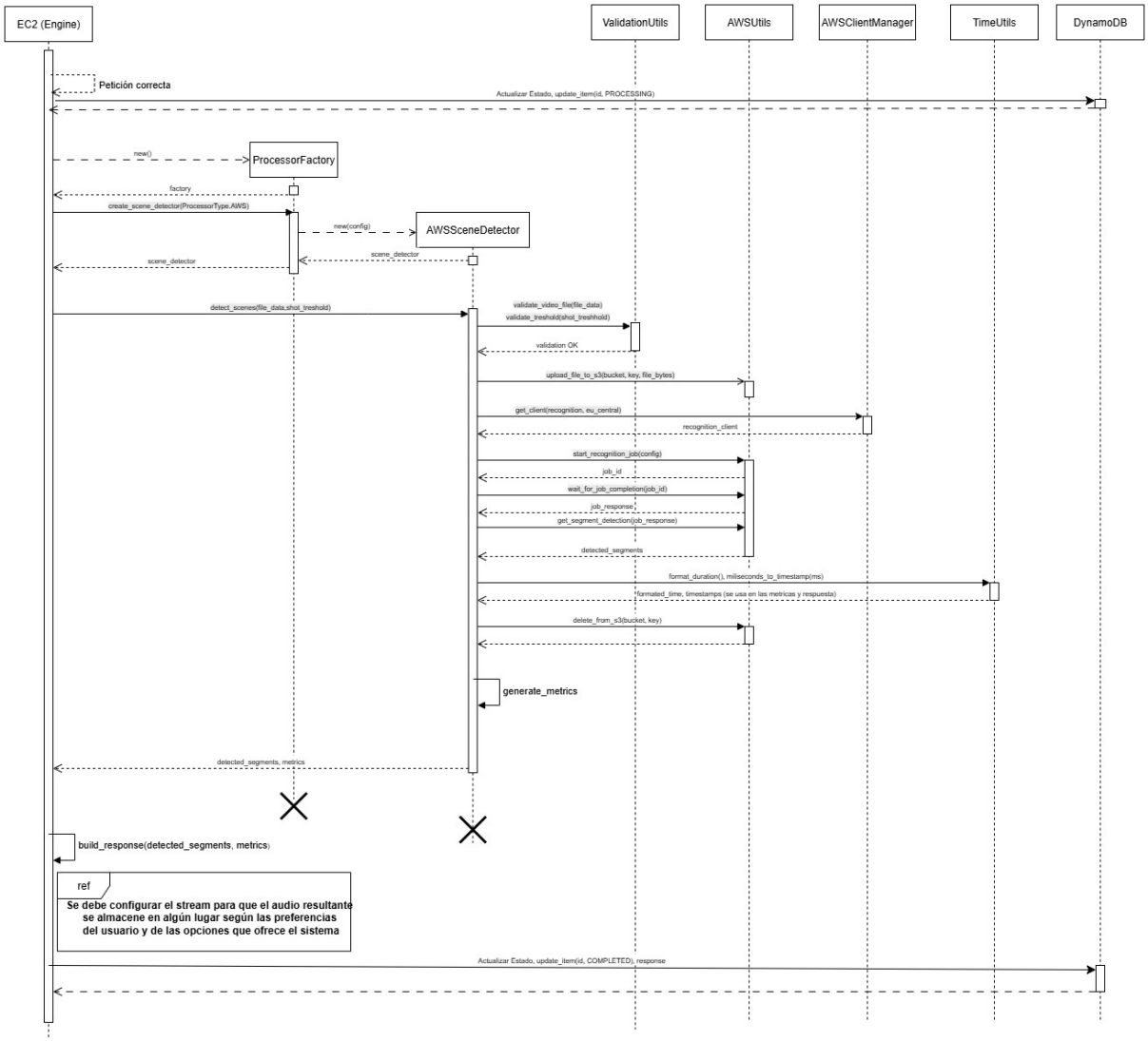


Figura 11. Diagrama de secuencias procesamiento video con AWS

Igual que en el caso de FFmpeg, se puede observar en el diagrama que la lógica del sistema referente a la interacción entre API Gateway, el backend en EC2, SQS y el motor de procesamiento (EC2 Engine) no se representa en su totalidad, ya que es común al comportamiento mostrado en la operación de normalización. Por ello, únicamente se muestra el procesamiento específico de la petición una vez esta es recibida por el motor.

En cuanto a la espera de la finalización del trabajo de AWS Rekognition, se ha utilizado el mismo método empleado con MediaConvert, basado en el uso de colas SQS y notificaciones SNS. Este mecanismo ya fue representado anteriormente mediante un diagrama de secuencias.

5.2.2 Módulo de gestión de correos electrónicos

A continuación, se muestra el diagrama de secuencias correspondiente al establecimiento de conexión con el servicio de correo electrónico.

Es importante tener en cuenta que, en esta fase, únicamente se preparan los parámetros necesarios para el envío y la lectura de correos. La conexión real con el servidor SMTP se establece en el momento en que se desea enviar o recibir un mensaje.

Este comportamiento puede observarse en el siguiente diagrama de secuencias.

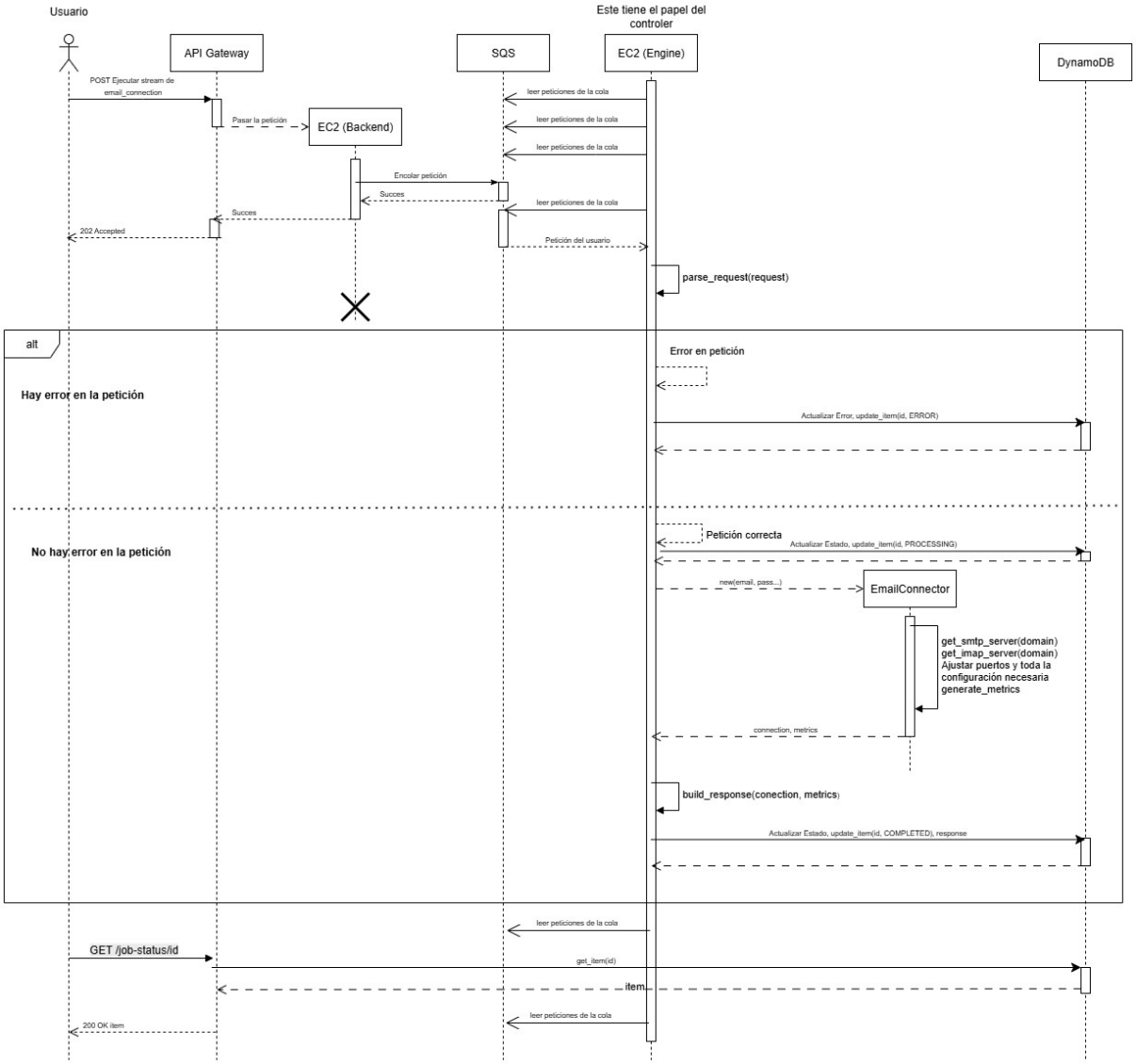


Figura 12. Diagrama de secuencias del establecimiento de conexión con el servicio de correo electrónico

El diagrama a continuación representa el proceso de envío de un correo electrónico con archivos adjuntos.

Para facilitar la visualización y mejorar la claridad del diagrama, se asume que la conexión al servidor de correo ya ha sido establecida (como se muestra en el diagrama anterior). Asimismo, tal y como se explicó al principio de todo, se supone que los archivos a adjuntar ya han sido cargados y están disponibles en una lista en formato FileData.

Además, se ha omitido la representación del flujo completo de peticiones entre API Gateway, el backend en EC2 y el motor de procesamiento (EC2 Engine), con el fin de evitar repeticiones y mantener el diagrama más limpio. Por ello, en este caso, el diagrama se centra exclusivamente en el envío del correo electrónico con archivos adjuntos mediante Outlook.

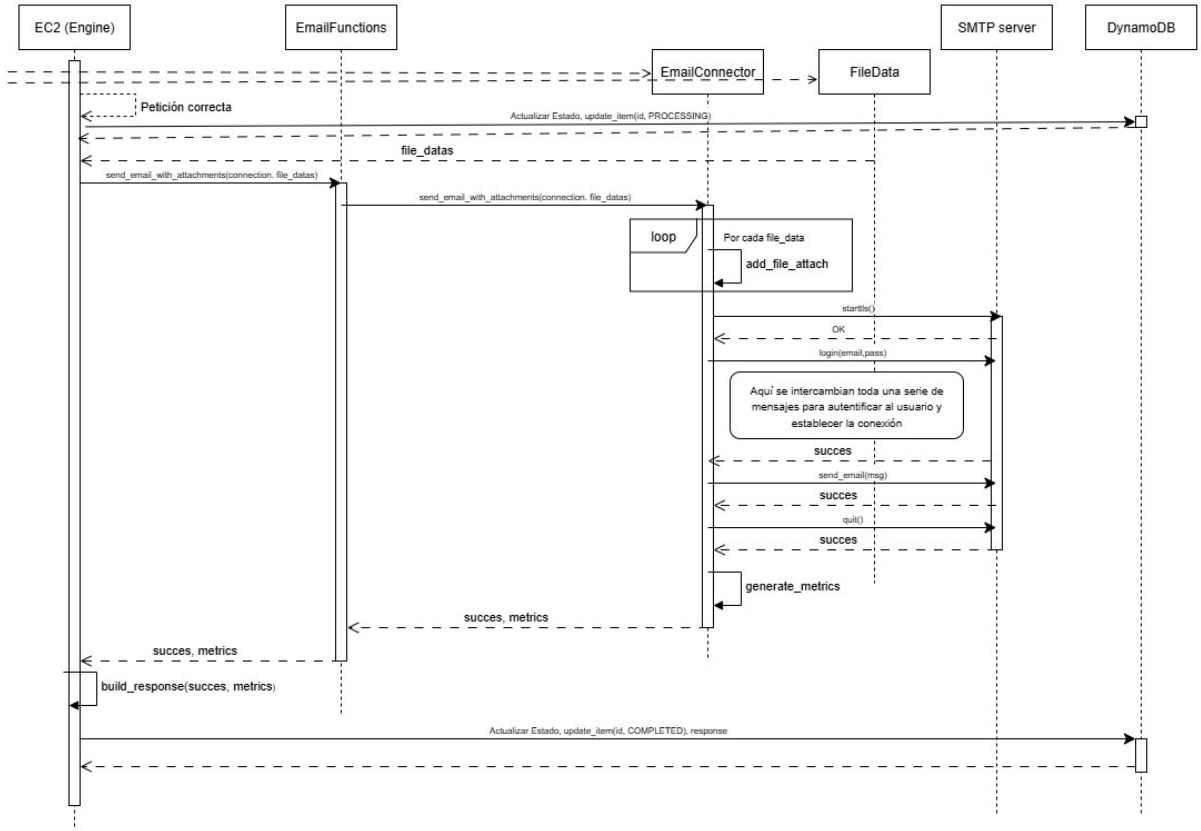


Figura 13. Diagrama de secuencias del envío de un correo (Outlook) con archivos adjuntos

5.2.3 Módulo de SharePoint

En esta sección se presentan los diagramas de secuencias correspondientes al establecimiento de conexión y a la descarga de un archivo desde SharePoint. En la representación se ha omitido el flujo completo de peticiones entre API Gateway, el backend en EC2 y el motor de procesamiento (EC2 Engine), con el fin de evitar repeticiones, ya que fue mostrado en los diagramas anteriores, y mantener el diagrama más limpio.

A continuación, se muestra el diagrama de secuencias correspondiente al establecimiento de conexión con SharePoint.

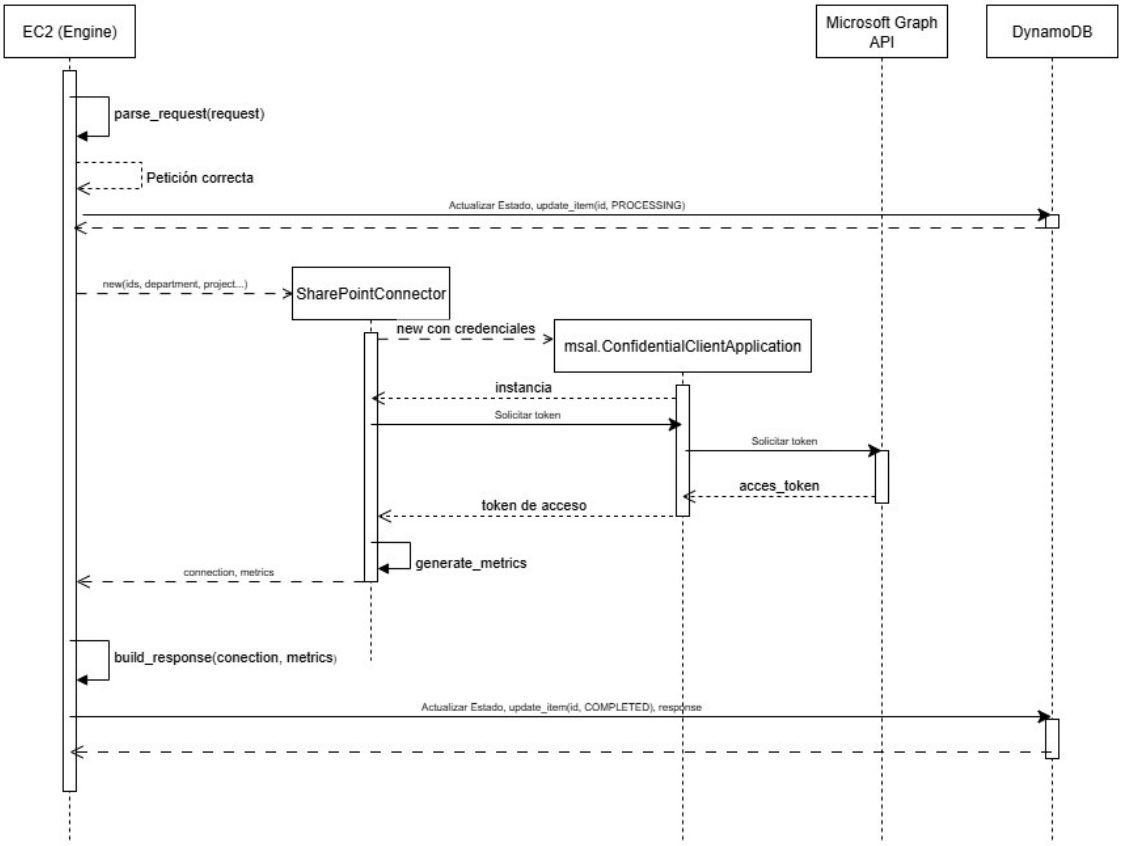


Figura 14. Diagrama de secuencias del establecimiento de conexión con SharePoint

El siguiente diagrama de secuencias muestra el proceso de descarga de un archivo desde SharePoint, suponiendo que la conexión ya ha sido establecida previamente.

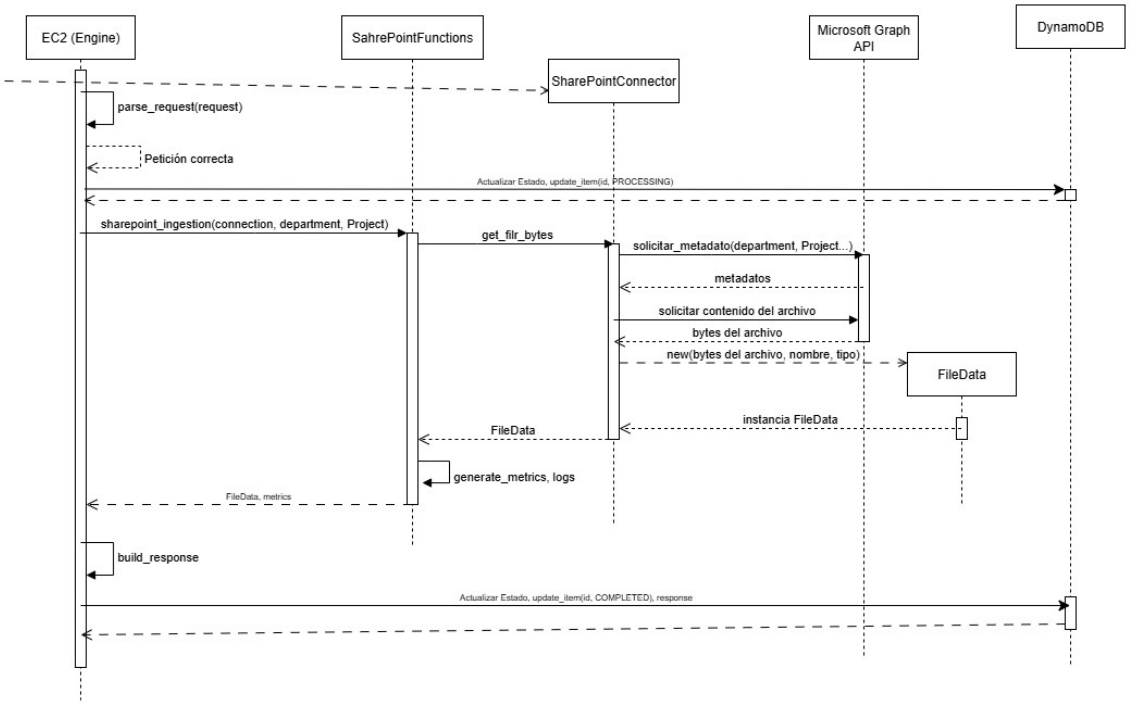


Figura 15. Diagrama de secuencias de la descarga de un archivo desde SharePoint

5.2.4 Crear sesión en AIDA

Un flujo particularmente interesante de representar es la interacción y las llamadas involucradas en la creación de una sesión de AIDA con el contexto de la reunión. Este proceso de comunicación se ilustra en el siguiente diagrama de secuencias.

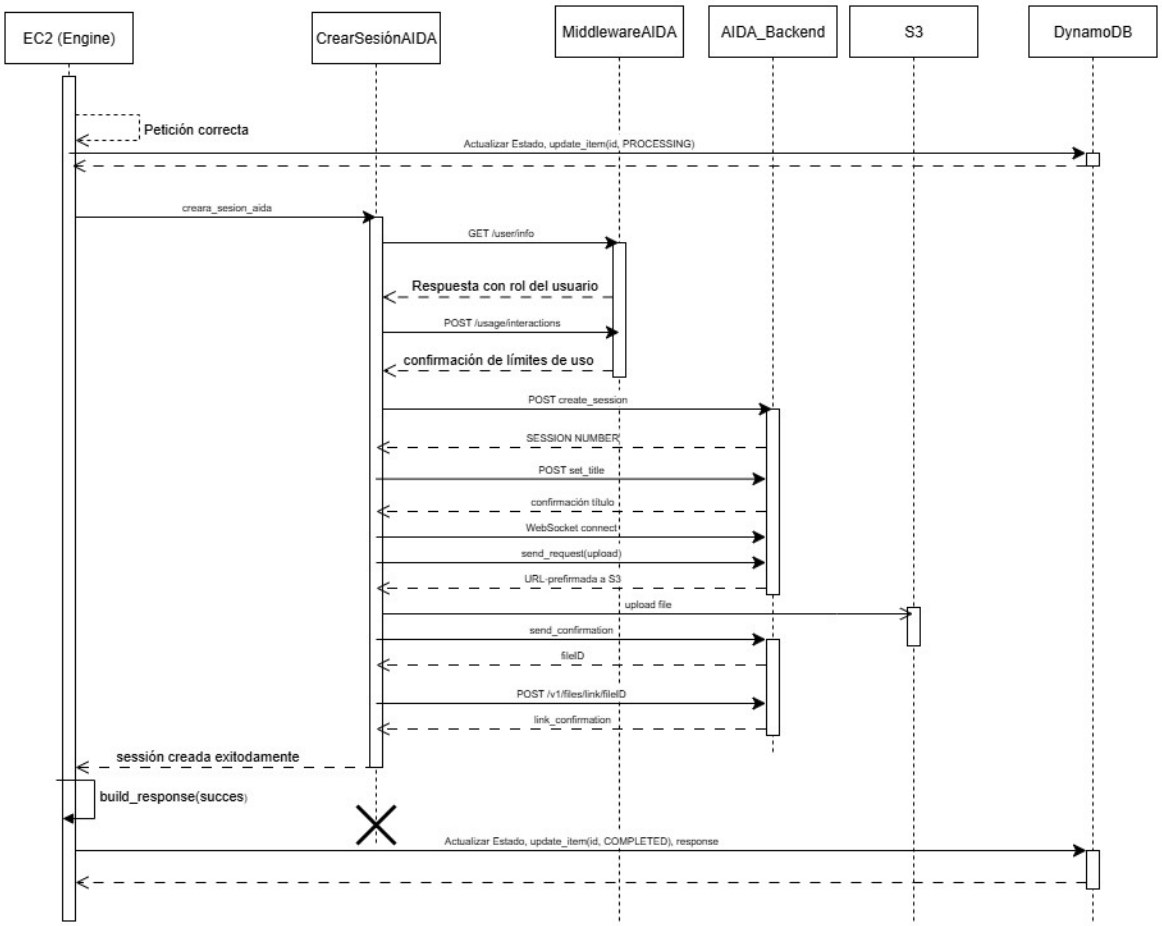


Figura 16. Diagrama de secuencias de la creación de una sesión en AIDA, asociando un archivo como contexto.

6 Diseño

En esta sección se explica el diseño de la arquitectura, así como el diseño del grafo dirigido (stream) de DOCs, que representa el flujo de procesamiento automatizado.

6.1 Arquitectura global

En esta sección se presenta la propuesta inicial para el diseño arquitectónico global del sistema, que abarca tanto la integración con la infraestructura de Microsoft como los diversos procesos de procesamiento necesarios para la obtención del acta de reunión automatizada.

6.1.1 Integración con Microsoft

En lugar de autenticar individualmente a cada usuario, se utiliza Microsoft Graph API para acceder a un directorio centralizado en SharePoint donde se almacenan automáticamente todas las grabaciones de reuniones de la empresa. Estas grabaciones están organizadas siguiendo un esquema predefinido (por departamento y proyecto), lo que permite acceder fácilmente a cada archivo mediante su nombre.

Los permisos de acceso a cada grabación se determinan a partir de los metadatos del archivo, los cuales incluyen información como el nombre y el correo electrónico del usuario que la organizó. De este modo, es posible verificar si un usuario tiene autorización para acceder a una grabación concreta sin necesidad de que se autentique manualmente. La aplicación puede descargar el video correspondiente y, si procede, enviar por correo electrónico los resultados del procesamiento automatizado.

6.1.2 Procesamiento audio y video

6.1.2.1 Desafíos técnicos a resolver

Separación por hablantes (diarización):

- Implementación de técnicas de detección de hablantes mediante análisis de audio.
- Extracción de nombres de participantes mediante el video.
- Evaluación comparativa de rendimiento y coste entre métodos basados solo en audio o en audio y video.

Extracción de elementos visuales relevantes:

- Análisis de la viabilidad de extraer información de presentaciones y diagramas mostrados durante la reunión.
- Evaluación del impacto en eficiencia y velocidad de procesamiento al incorporar esta funcionalidad.

Optimización del procesamiento de audio:

- **Eliminación del silencio:** Evaluación de las siguientes alternativas:
  - Análisis de la ganancia en eficiencia al eliminar silencios.
  - Viabilidad de realizar transcripción con audio limpio mientras se mantiene el audio original para el tratamiento del video.

- Desarrollo de un sistema de mapeo temporal entre audio original y audio procesado (por cada segundo X del audio limpio indicar a qué segundo Y pertenece del audio original).

6.1.2.2 Objetivos de optimización

Determinar la combinación óptima de técnicas que maximice la precisión manteniendo un nivel aceptable de eficiencia computacional, evaluando específicamente las ventajas de procesamiento conjunto de audio y video frente al procesamiento exclusivo de audio.

6.1.3 Uso de LLMs

- **Mejora de la calidad de transcripción:** Evaluación del impacto de utilizar LLMs para corregir la transcripción base y la implementación de técnicas para mejorar puntuación, capitalización y corrección de errores fonéticos o gramaticales.
- **Análisis de confiabilidad:** Desarrollo de métodos para evaluar la confiabilidad del resultado final y la implementación de técnicas automatizadas para detección y corrección de posibles errores en caso de ser necesario.

6.1.4 Diseño de la arquitectura global

A continuación, se presenta un diseño simplificado de la arquitectura propuesta. Esta arquitectura está organizada en tres módulos principales que interactúan entre sí:

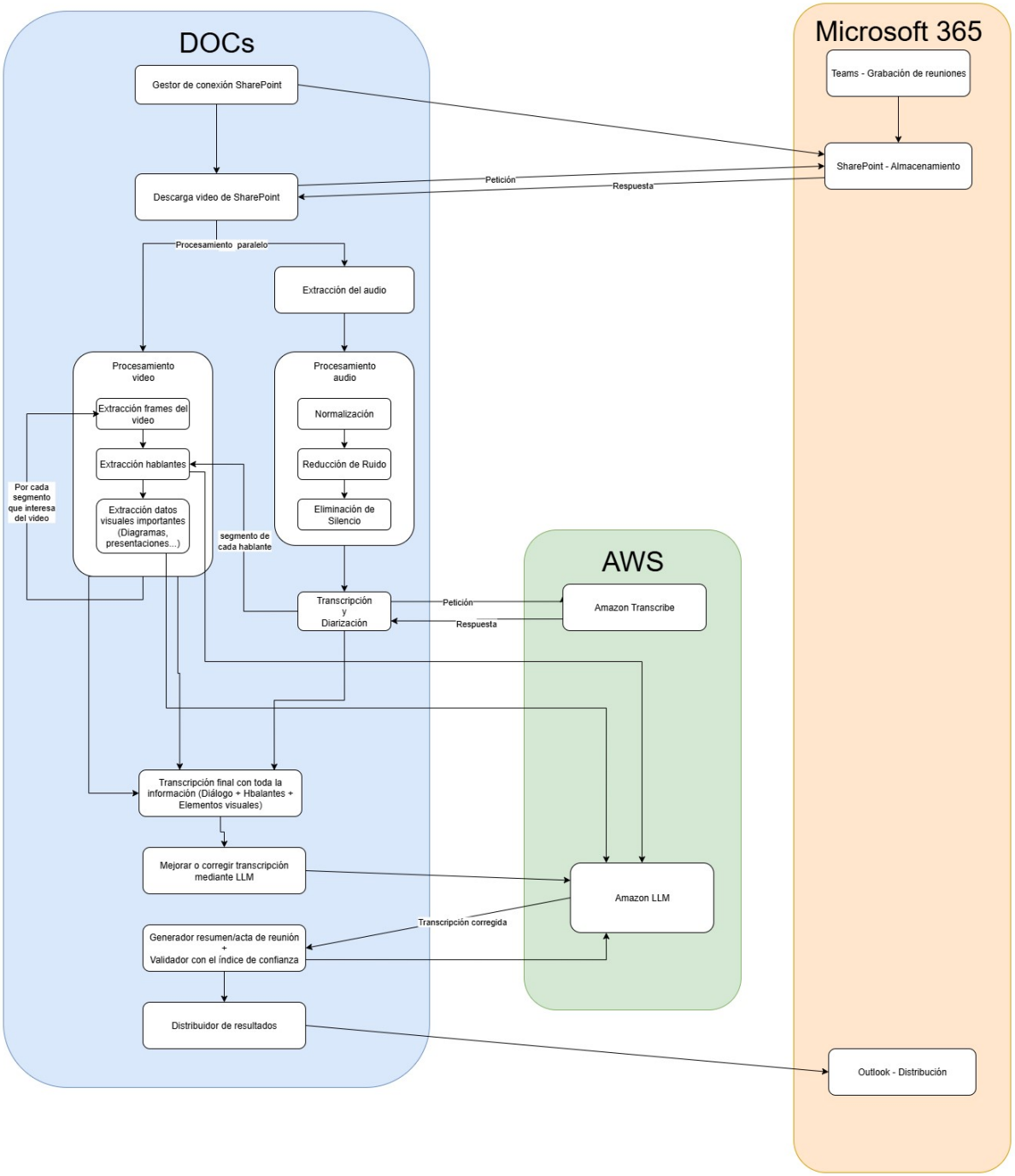


Figura 17. Diseño de la arquitectura

En el diagrama se muestran los 3 módulos separados: Módulo de Microsoft 365, AWS y DOCs.

El primero es Microsoft 365, donde se registra la grabación de Teams y de donde es descargada, además de utilizarse para enviar por Outlook los resultados finales.

El segundo es AWS, donde se gestionan los servicios de Amazon, la transcripción automática y la llamada a los LLMs. Además del uso de AWS en los casos que el procesamiento de video o audio, como la normalización, se realiza en AWS MediaConvert como se ha mostrado en el apartado de análisis.

Finalmente, el tercer módulo es el de DOCs, que es el proyecto de automatización en donde se ejecuta el stream, definido en el siguiente apartado, con el flujo de trabajo. Aquí es donde se gestiona la conexión con Microsoft 365, se descarga el video, se realiza en paralelo el procesamiento de audio y de video, la transcripción automática (llamada al servicio de Amazon Transcribe), se genera el acta de reunión, se mide la confiabilidad de los resultados y, finalmente, se distribuyen estos mediante Outlook.

6.1.5 Problemas, alternativas y soluciones

6.1.5.1 Integración con Microsoft

El principal desafío encontrado durante la fase de investigación es la falta de permisos para crear una aplicación en Azure AD, necesaria para gestionar la conexión con Microsoft según el diseño propuesto inicial en dónde se requería de la autentificación directa de los usuarios para poder tener acceso a descargar las grabaciones.

Alternativas:

- **Compartición directa desde el usuario:** El usuario podría compartir un enlace del video de la reunión en SharePoint con los permisos adecuados para su descarga. Sin embargo, esto presenta limitaciones:
  - Los permisos disponibles podrían están restringidos por políticas de IT corporativas.
  - Solo el organizador de la reunión (propietario del archivo) puede establecer permisos para todo el grupo IDIADA.
- **Uso de OneDrive como intermediario:** El usuario podría subir el video a OneDrive y compartir el enlace, permitiendo el acceso mediante peticiones HTTP utilizando herramientas como wget, curl o la biblioteca requests.
- **Carga directa:** Implementación de un sistema donde el usuario sube el video directamente para su procesamiento, evitando la integración con Microsoft Graph API.

Solución propuesta y final:

Para mitigar estos problemas, se ha propuesto al equipo de IT centralizar el almacenamiento de las grabaciones de Teams en un único directorio con una estructura organizada. Se requiere además una aplicación registrada en Azure AD que proporcione acceso programático a este directorio y permita la descarga de los archivos.

6.1.5.2 Análisis y extracción de información visual de un video

Problema identificado:

El procesamiento de video para la extracción de información de elementos visuales relevantes (presentaciones, diagramas, etc.) presenta desafíos significativos si se requiere entrenar modelos específicos para el reconocimiento de estos elementos.

Alternativas:

- **Entrenamiento de modelos personalizados:** Desarrollar y entrenar modelos específicos para reconocimiento de elementos visuales en reuniones de Teams. Esta alternativa es problemática debido a que requeriría muchos recursos y tiempo de procesamiento.

- **LLMs con capacidad de procesamiento visual (video):** Utilizar modelos avanzados como Amazon Nova, que combinan procesamiento de lenguaje y visión en video.

Solución propuesta:

La aproximación recomendada es utilizar los LLMs de Amazon que pueden procesar tanto texto como imágenes y videos.

Esta solución aprovecha la infraestructura en la nube de Amazon para el procesamiento intensivo, mitigando los problemas de recursos computacionales necesarios, además de ser una solución escalable y mantenible a largo plazo.

Solución final:

Tras probar el modelo LLM Nova de AWS, que permite la entrada de vídeo, se ha observado que ofrece una calidad deficiente y presenta limitaciones con respecto al tamaño del archivo de entrada. En particular, cuando el vídeo es demasiado grande, no puede procesarse en su totalidad.

Además, estos modelos no son capaces de leer texto, lo que impide extraer información relevante del contenido mostrado en la pantalla compartida durante la reunión.

Como solución a este inconveniente, en el presente proyecto se ha optado por aplicar un enfoque basado en la detección de cambios de escena. Estos cambios suelen producirse al comenzar o finalizar una compartición de pantalla, o bien cuando se producen transiciones bruscas en el contenido visualizado. A partir de estos puntos de cambio, se identifican los segmentos correspondientes a pantalla compartida, de los cuales se extraen fotogramas clave. Estos fotogramas se analizan posteriormente mediante un modelo LLM con capacidades de visión, con el objetivo de extraer información relevante directamente de las imágenes.

Es importante señalar que, a diferencia del modelo con entrada directa de vídeo, el modelo utilizado para el análisis de fotogramas es mucho más óptimo, está mejor entrenado y, además, posee la capacidad de leer texto dentro de las imágenes, lo cual resulta fundamental para interpretar correctamente el contenido compartido.

En los anexos se detalla la implementación del proceso de extracción de información a partir de vídeo mediante los modelos Nova de AWS.

6.1.5.3 Eliminación del silencio

En el diseño inicial se propuso la eliminación de silencios, ya que, tal y como se expone en el estado del arte, esta técnica puede reducir el tiempo de procesamiento de la transcripción en determinados casos. Sin embargo, al aplicarla durante la implementación del proyecto, se observó que, si bien puede ofrecer mejoras en tiempo de procesamiento, impacta negativamente en la diarización. Esto se debe a que los silencios son útiles para detectar cambios de hablante y variaciones en la voz.

Además, la mejora de rendimiento solo es significativa cuando el audio contiene largos períodos de silencio; de lo contrario, no es notable.

Por esta razón, la eliminación de silencios no se ha incluido en la solución final. Aunque en el diagrama de la arquitectura global aún se representa para indicar su posición dentro del flujo de procesamiento audio, concretamente al final del mismo, y ha sido retirada de la implementación, del stream mostrado en el apartado siguiente y de los diagramas del apartado de análisis para reflejar el estado final del proyecto.

6.1.6 Diseño técnico de conectores

Conector SharePoint: Se usa Microsoft Graph API, que es una API REST específica de Microsoft que permite acceder de forma unificada a datos y servicios de Microsoft 365, como SharePoint, OneDrive, Outlook o Teams.

Conector Amazon Transcribe:

- Uso del SDK (boto3) de AWS para Python.
- Ofrece transcripción y diarización de audios largos y cortos.
- Permite la creación de vocabulario personalizado para mejorar la precisión en términos específicos.

Conector AWS Bedrock:

- Uso del SDK (boto3) de AWS para Python.
- Integración con modelos LLM alojados en AWS Bedrock.
- Procesamiento de prompts y análisis semántico de texto o imágenes.

Conector AWS MediaConvert y Rekognition:

- Uso del SDK (boto3) de AWS para Python.
- **AWS MediaConvert:** Conversión y procesamiento de archivos de vídeo y audio.
- **AWS Rekognition:** Análisis de vídeo para detección de cambios de escena.

6.2 Stream de DOCs

A continuación, se muestra el diseño del stream como un grafo dirigido que representa la ejecución automatizada del proceso.

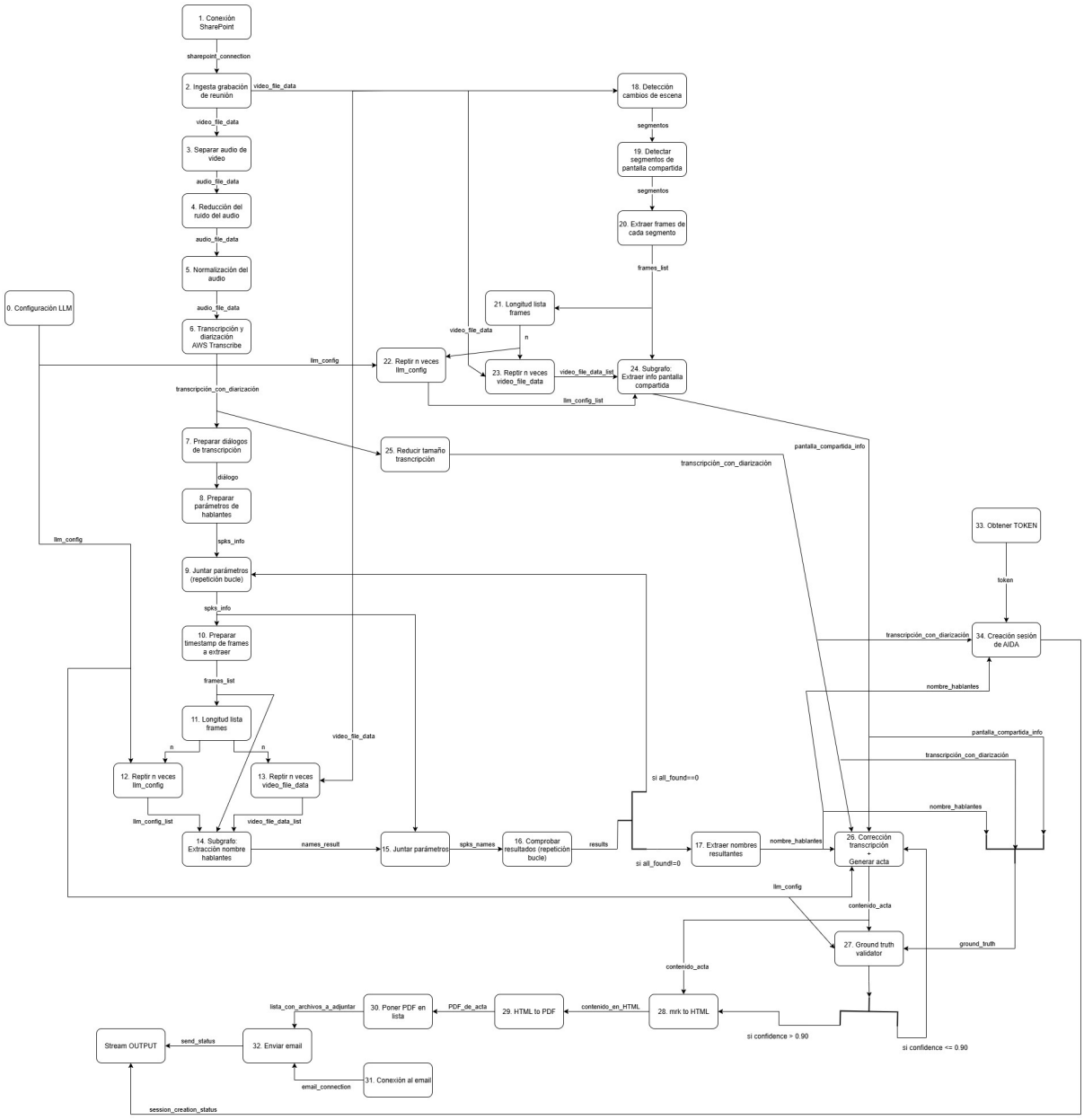


Figura 18. Diseño del stream de DOCs

Cada nodo del stream representa una tarea con una funcionalidad específica. A continuación, se presenta un resumen del objetivo y funcionalidad general del stream. Más adelante, en la sección de implementación, se detallan en profundidad la funcionalidad de cada nodo y los aspectos técnicos correspondientes.

Cabe destacar, que las dependencias entre los nodos marcan el orden de ejecución.

El flujo comienza con la descarga de la grabación de reunión y se inicializan configuraciones del LLM. A continuación, se ejecutan dos ramas en paralelo.

Una de las ramas detecta los segmentos con cambios de escena en el video y, dentro de ellos, identifica los que presentan pantalla compartida para extraer fotogramas clave, que luego se envían a un LLM con capacidades de visión con el fin de obtener información relevante.

A continuación, se muestra el subgrafo encargado de la extracción de información de la pantalla compartida en donde se muestra la extracción del fotograma y la llamada al LLM.

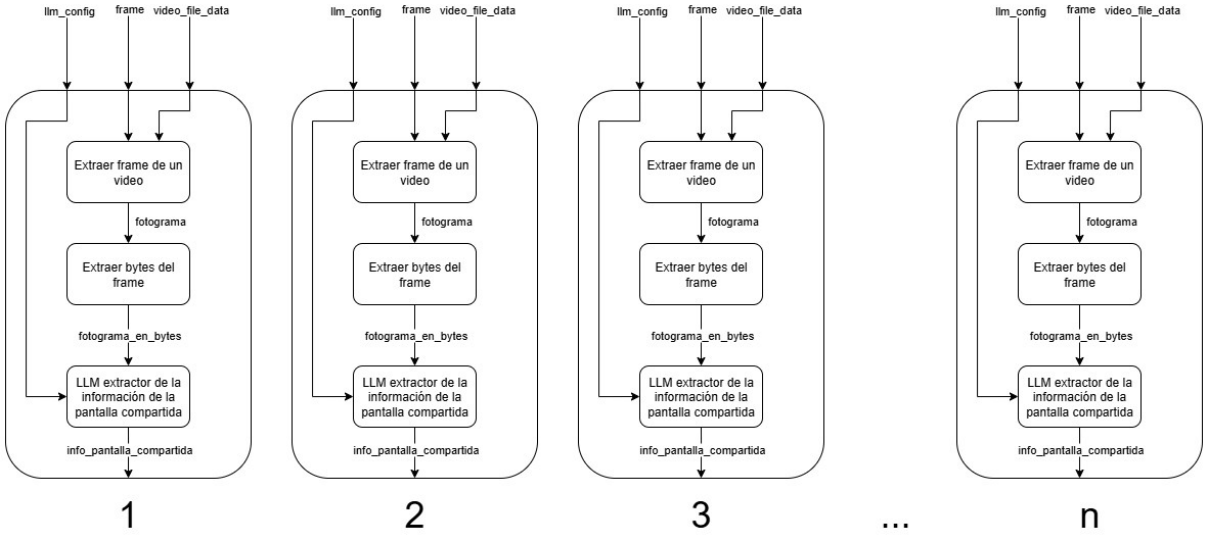


Figura 19. Diseño del subgrafo de extracción de información de la pantalla compartida

El subgrafo lanza tantas ramas paralelas como segmentos de pantalla compartida, y cada una evalúa en paralelo el contenido de su fotograma correspondiente.

La otra rama empieza con la reducción de ruido y la normalización del audio, lo que permite obtener una señal más clara para la transcripción y diarización posterior.

El nodo *6. Transcripción y diarización*, como su nombre indica, se encarga de realizar tanto la transcripción del audio como la diarización, es decir, la segmentación del diálogo por hablantes. Esto permite identificar qué dice cada persona, generando un resultado del cual se puede extraer el contenido hablado por cada locutor. La diarización asigna identificadores genéricos a los hablantes, como spk_0, spk_1, spk_2, etc. Por esta razón, a continuación, comienza el flujo encargado de extraer los nombres reales de los hablantes a partir del video de la reunión de Teams, en el cual estos sí son visibles.

Los nodos *7. Preparar diálogos de transcripción* y *8. Preparar parámetros de hablantes* se encargan de construir la estructura de datos con la información necesaria sobre los segmentos correspondientes a cada hablante, a partir de la transcripción. Esto permite hacer un seguimiento del proceso de extracción de nombres de hablantes, ya que es posible que no se identifiquen en la primera iteración y se requieran varias pasadas. Por ello, la información debe estar bien estructurada para asegurar un seguimiento preciso.

El nodo *9. Juntar parámetros* actúa como enlace entre el inicio del flujo y la anidación del bucle. En la primera iteración, este nodo devuelve la información que recibe del nodo 8, mientras que, si la iteración se repite, devuelve la entrada proveniente del nodo *16. Comprobar resultados*.

A continuación, se extrae un fotograma del segmento correspondiente a cada hablante con el objetivo de identificar su nombre mediante un LLM con capacidades de visión. Para ello, se utiliza el mismo tipo de subgrafo que en la extracción de información de la pantalla compartida que lanza tantas ramas paralelas de extracción de un fotograma de un video y procesamiento como hablantes haya. En caso de que la iteración del bucle se repita porque no se logró identificar a todos los hablantes, solo se procesan aquellos cuyo nombre aún no ha sido extraído.

A continuación, se muestra el subgrafo de extracción del nombre de los hablantes. Puede observarse que ambos subgrafos comparten la misma estructura, diferenciándose únicamente en la tarea específica que debe realizar cada LLM.

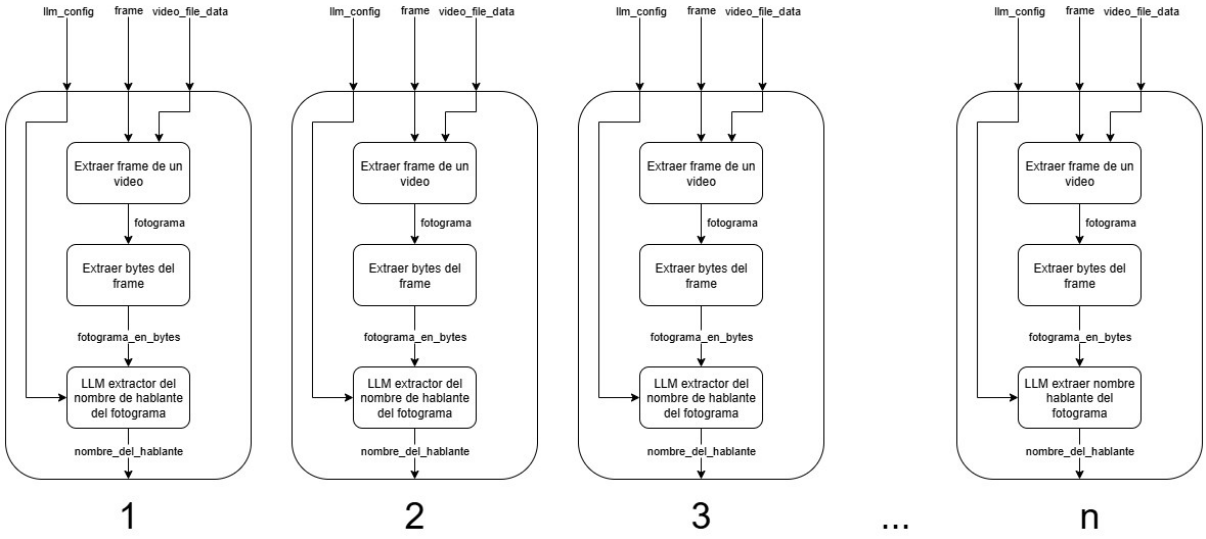


Figura 20. Diseño del subgrafo de extracción del nombre de hablantes

Una vez que se han extraído todos los nombres de los hablantes, se pasa al siguiente paso: la corrección de la transcripción mediante un LLM. Esto se debe a que, en muchos casos, las transcripciones generadas automáticamente presentan errores de puntuación, gramática o confusión entre palabras fonéticamente similares, que un LLM puede identificar y corregir con mayor precisión.

Además, en esta fase se genera el contenido del acta de reunión en formato Markdown, utilizando la transcripción corregida, los nombres identificados de los hablantes y la información extraída de la pantalla compartida durante la reunión.

Acto seguido, el contenido del acta generado es evaluado por un validador, que también es un LLM, el cual determina un nivel de confianza final sobre la precisión del resultado. Este validador utiliza como base la transcripción, la cual incorpora una asignación del valor de confianza a cada palabra según la certeza del reconocimiento, los nombres identificados de los hablantes y la información extraída de los elementos visuales. A partir de estos datos, proporciona una evaluación y calcula una puntuación que refleja qué tan fiel es el contenido del acta respecto a las fuentes originales. Por ejemplo, valida que toda la información descrita en el acta realmente esté presente y se corresponda con lo observado en la reunión.

En esta etapa se aplica una condición que puede derivar en un bucle: si la confianza evaluada es menor o igual al 90 %, el contenido del acta se genera nuevamente. En esta nueva iteración, es más probable obtener un resultado más preciso, ya que se incorpora como entrada al generador del contenido del acta tanto el nivel de confianza anterior (en forma de valor entre 0 y 1) como una evaluación detallada de los aciertos y errores detectados en la versión anterior, lo que permite al LLM corregirlos de manera más efectiva.

Una vez completada esta etapa y alcanzada una confianza superior al 90 %, el contenido del acta en formato Markdown se convierte a HTML para aplicar el formato deseado, y luego se transforma en un archivo PDF que se adjunta en el correo enviado al usuario final.

Paralelamente a todo este flujo de trabajo, se crea una sesión en el asistente inteligente de la empresa, denominado AIDA.

En dicha sesión se carga como contexto la transcripción de la reunión junto con los nombres de los hablantes. Además, en el correo de la reunión se incluye automáticamente un enlace al asistente, donde los usuarios pueden acceder a la sesión con el contexto ya preparado, lo que les permite realizar preguntas sobre la reunión de forma cómoda e interactiva.

Finalmente, los datos procesados se envían al nodo de salida del flujo, donde son almacenados o distribuidos según los requisitos de la aplicación. En conjunto, el diagrama representa una arquitectura modular y escalable para el procesamiento de grabaciones de reuniones, con capacidades avanzadas de análisis y automatización.

7 Implementación

En este apartado se describe la implementación de las distintas funcionalidades del proyecto, así como las tecnologías y bibliotecas utilizadas para llevarlas a cabo. Se detallan los componentes clave, su propósito dentro del flujo general y las herramientas empleadas en cada caso.

7.1 Funciones generales

Se presentan aquí funciones generales que sirven de apoyo a múltiples componentes del sistema, y cuya aplicación se explicará en secciones posteriores.

7.1.1 Clase FileData

Con el fin de homogeneizar el tratamiento de todo tipo de ficheros (imágenes, documentos de Word, archivos PDF, vídeo, audio, JSON, texto, Excel, etc.), en el proyecto todos ellos se gestionan mediante objetos de la clase FileData.

Esta clase contiene tres atributos fundamentales que encapsulan toda la información necesaria para un manejo unificado de archivos:

- Los datos binarios completos del archivo, para su manipulación en memoria.
- El nombre original del archivo, incluyendo su extensión.
- El tipo MIME del archivo (por ejemplo, application/pdf, text/plain).

7.1.2 Creación del cliente S3

Para realizar las operaciones de S3 hay definir el cliente que realizará las peticiones para subir, descargar y borrar los ficheros en la nube.

A continuación, se muestra cómo se define el cliente:

```
client = boto3.client(
        service_name="s3",
        aws_access_key_id="clave_acceso",
        aws_secret_access_key="clave_secreta",
        region_name="eu-central-1",
    )
```

Código 1. Creación del cliente S3.

Es necesario especificar las credenciales privadas y la región del servicio. En este caso, el bucket de S3 alquilado se encuentra en Europa Central (Frankfurt).

Se utiliza la librería *boto3*, la herramienta oficial de AWS para Python, para la autenticación y conexión con el servicio S3, junto con la librería *os*, que permite acceder a las variables de entorno donde se almacenan de forma segura las credenciales de AWS (clave de acceso y clave secreta).

7.1.3 Descargar un fichero desde S3 (load_s3_file)

La función *load_s3_file* está diseñada para descargar archivos desde Amazon S3 usando Python.

La función recibe cuatro parámetros: Bucket, key, wait y waiter_config.

- Bucket: Nombre del bucket S3. En términos de S3, es similar a un repositorio o contenedor donde se almacenan los archivos.
- Key: Ruta del archivo en el bucket.
- Wait: Parámetro opcional, un booleano que permite especificar si se debe esperar a que el archivo exista antes de descargarlo.
- waiter_config: Configuración de reintentos para la espera, con valores por defecto de 1 segundo de espera y 50 intentos máximos.

Para autenticarte en S3, se crea un objeto de tipo *resource* (recurso), el cual permite identificarse y realizar operaciones sobre el servicio correspondiente, en este caso, S3. Para ello, se utilizan las mismas librerías que en la creación del cliente S3 descrita en el apartado anterior: *boto3* y *os*.

En *boto3*, existen dos formas principales de interactuar con los servicios de AWS: *client* y *resource*. El cliente (*client*) proporciona una interfaz de bajo nivel, con llamadas directas a la API, mientras que el recurso (*resource*) ofrece una abstracción de más alto nivel, más sencilla de usar para ciertas operaciones comunes. Dependiendo del caso de uso, puede ser conveniente elegir una u otra.

A continuación, se muestra cómo se define el *resource*:

```
s3 = boto3.resource(

        service_name="s3",

        aws_access_key_id="clave_acceso",

        aws_secret_access_key="clave_secreta",

        region_name="eu-central-1",

    )
```

Código 2. Creación del *resource* para S3.

En este caso, se utilizan los mismos parámetros que los indicados al crear el cliente en el apartado anterior.

Se instancia el cliente, como se ha indicado en el apartado anterior, para realizar las peticiones a S3, y se emplea también un objeto *resource* para ejecutar la operación de descarga (GET) del archivo desde S3.

El cliente de S3 se utiliza específicamente para obtener un waiter, un mecanismo de *boto3* que permite verificar repetidamente si un recurso está disponible antes de continuar.

Si el parámetro wait está activado (True), se inicia el siguiente flujo de validación:

```
Si wait == True:

    Obtener waiter desde el cliente

    Repetir hasta que:

        - El archivo esté disponible en S3
```

```
        - O se alcance el tiempo máximo definido en la configuración

    Si el archivo no aparece a tiempo:

        Lanzar excepción
```

Código 3. Pseudocódigo waiter de la descarga de archivos de S3.

Una vez verificado, el archivo se descarga como un objeto binario (bytes) y se devuelve su contenido en formato binario.

La función contempla el manejo de errores en los siguientes casos:

1. Si ocurre un error al crear el cliente o el recurso, se captura la excepción y se muestra un mensaje claro al usuario indicando el tipo de fallo.
2. Si el parámetro wait está activado y se superan los intentos definidos en waiter_config, se lanza una excepción automáticamente.
3. Si se produce un error al recuperar el archivo, la excepción se captura y se notifica al usuario con un mensaje informativo sobre el problema.

7.1.4 Cargar un fichero a S3 (upload_file_to_s3)

La función *upload_file_to_s3* está diseñada para subir archivos a un bucket de Amazon S3.

La función recibe cuatro parámetros: Bucket, key, file y metadata.

- Bucket: Nombre del bucket S3.
- Key: Ruta del archivo en el bucket.
- File: Contenido del archivo a subir en formato binario (bytes).
- Metadata (opcional): Diccionario con metadatos personalizados para adjuntar al archivo.

Se crea el cliente como se ha mostrado en el apartado *7.1.2 Creación del cliente S3*, mediante las librerías *boto3* y *os*.

Para subir un archivo, se construye un diccionario de configuración para la petición (request) que incluye el contenido del archivo, el nombre del bucket y la ruta de destino (key), además de agregar los metadatos en caso de que se hayan indicado por parámetro.

Después de tener la configuración preparada, se ejecuta la operación **put_object** del cliente de S3 para realizar la subida del archivo.

Durante todo el proceso se controla la posible aparición de excepciones:

- Si ocurre algún error al crear el cliente, se captura la excepción y se notifica al usuario con un mensaje claro sobre el tipo de fallo.
- Si ocurre algún error durante la subida del archivo, se captura la excepción y se informa al usuario con un mensaje detallado sobre el problema.

7.1.5 Borrar un fichero de S3 (delete_file_from_s3)

La función *delete_file_from_s3* está diseñada para eliminar archivos específicos de un bucket en Amazon S3.

La función recibe dos parámetros: Bucket y key.

- Bucket: Nombre del bucket S3.
- Key: Ruta del archivo en el bucket.

Se crea el cliente como se ha mostrado en el apartado *7.1.2 Creación del cliente S3*, mediante las librerías *boto3* y *os*.

Se ejecuta el método **delete_object** del cliente de S3, pasando como argumentos el nombre del bucket y la key correspondiente. Es importante tener en cuenta que la eliminación es inmediata y permanente, ya que S3 no dispone de una 'papelera de reciclaje' por defecto. Por ello, es fundamental actuar con precaución al ejecutar esta operación, ya que los archivos eliminados no podrán recuperarse.

7.2 Procesamiento multimedia

Aquí se describe la implementación de las distintas funciones utilizadas para el procesamiento de audio y video, así como las tecnologías empleadas en el proceso: servicios de AWS y la herramienta FFmpeg.

Cabe destacar que las operaciones que se apoyan en servicios de AWS se basan en una arquitectura *serverless*.

7.2.1 MediaConvert de AWS

Uno de los muchos servicios que ofrece AWS es MediaConvert, una herramienta bastante potente que permite hacer procesamiento de diversos archivos multimedia y que en este caso nos interesa para hacer el procesamiento de audio y video.

7.2.1.1 Configuración de permisos

Para poder utilizar esta herramienta, es necesario configurar los permisos adecuadamente. En el proyecto DOCs no se había utilizado anteriormente, por lo que no existía ninguna configuración de permisos que permitiera su funcionamiento.

Permisos del usuario AWS:

El usuario de AWS que se utilice debe contar con tres permisos fundamentales: iam:PassRole, mediaconvert:CreateJob y mediaconvert:GetJob.

- **iam:PassRole:** Este permiso permite al usuario delegar un rol de IAM[21] a un servicio de AWS. En este caso, es necesario para que AWS MediaConvert pueda asumir un rol previamente definido con los permisos necesarios (por ejemplo, acceso a S3) y así ejecutar correctamente el trabajo.
- **mediaconvert:CreateJob:** Este permiso permite al usuario crear nuevos trabajos en AWS MediaConvert.
- **mediaconvert:GetJob:** Este permiso permite al usuario consultar el estado y los detalles de los trabajos creados en MediaConvert.

[21] Identity and Access Management

¿Cómo se asignan estos permisos?

Para ello, se debe utilizar la consola de AWS, ya sea a través de la interfaz web o mediante la línea de comandos. En este caso, se explica el procedimiento utilizando la consola web. Una vez iniciada la sesión, hay que buscar IAM, un servicio fundamental de AWS que gestiona la seguridad y los permisos dentro de tu cuenta.

Dentro del servicio IAM, se debe acceder al apartado de *Administración del acceso*, y luego a la sección de *Personas*.

Allí se mostrará un listado con todos los usuarios disponibles; hay que localizar el usuario que se utiliza para el trabajo que se desea configurar y acceder a su perfil. Una vez dentro, se debe entrar a la sección denominada *nombreusuario_iam_user_policy* para modificar sus permisos.

En esa sección se muestra un archivo JSON que contiene la lista de permisos asignados al usuario. Desde allí se pueden añadir o eliminar permisos manualmente. El formato del JSON es el siguiente:

```
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Action": [
            "ssm:List*",
            "ssm:Get*",
            "ssm:Describe*",
            "sqs:SendMessage",
            "sqs:ReceiveMessage",
            "s3:PutObject",
            ...
            "iam:PassRole",    #Se añaden estos 3 permisos
            "mediaconvert:CreateJob",
            "mediaconvert:GetJob"
         ],
         "Effect": "Allow",
         "Resource": "*"}
```

Código 4. JSON con los permisos del usuario usado en la ejecución del stream de DOCS.

A continuación, se muestra una captura de la interfaz web donde puede verse el mismo JSON dentro de la sección de permisos del usuario:

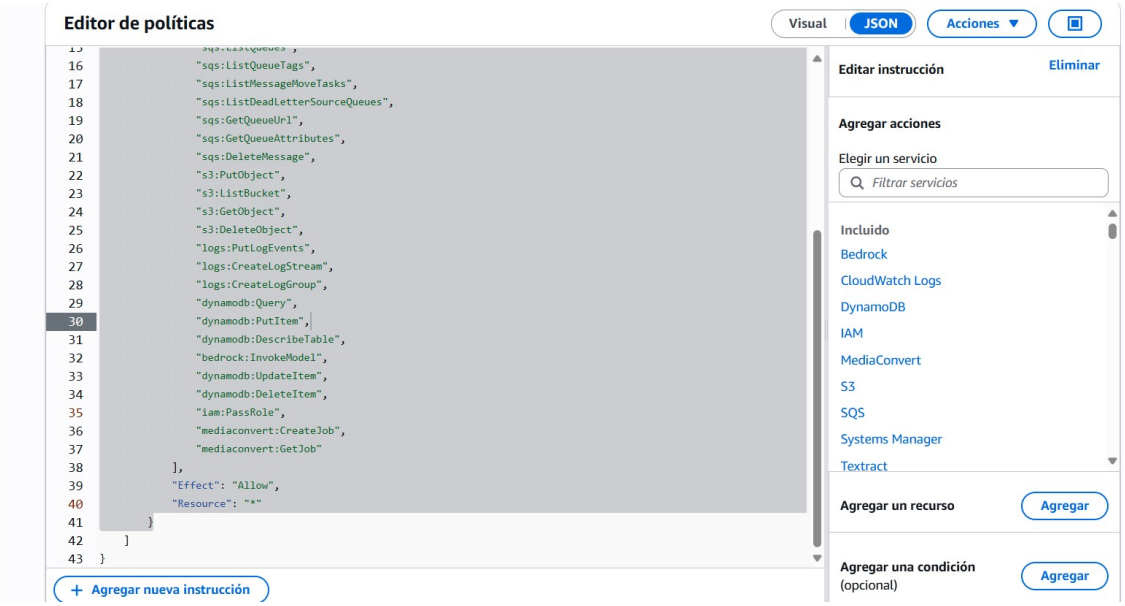


Figura 21. Permisos del usuario de AWS

Se debe tener en cuenta que, para modificar los permisos de los usuarios en una cuenta de AWS, es necesario contar con acceso de administrador.

Las operaciones de procesamiento multimedia se realizan sobre archivos almacenados en S3, y los resultados también se guardan allí. Por tanto, el usuario debe tener permisos de lectura y escritura en S3, específicamente *s3:GetObject* y *s3:PutObject*.

7.2.1.2 Creación del rol ARN

Es necesario asignar lo que se denomina un rol ARN[22] al trabajo en MediaConvert.

Un rol ARN identifica de forma única un rol de IAM que contiene un conjunto de permisos. Este rol permite a MediaConvert acceder de forma segura a otros servicios de AWS en nombre del usuario. Por ejemplo, puede otorgar acceso a buckets de S3 para leer archivos de entrada y escribir archivos de salida. Sin este rol, MediaConvert no podrá realizar ninguna operación sobre los recursos necesarios.

Para asignarlo, se debe acceder a la consola de AWS a través de la interfaz web y buscar el servicio *AWS Elemental MediaConvert*.

Una vez dentro, la consola permite comenzar a crear un trabajo; sin embargo, como en este caso lo que se busca es generar el rol ARN, hay que dirigirse al apartado denominado *Integración con AWS*, ubicado en la sección de *Configuración del trabajo*.

En esta sección, se solicitan algunos parámetros de configuración necesarios para generar automáticamente el rol ARN con los permisos requeridos.

[22] Amazon Resource Name

La configuración utilizada es la siguiente:

- Control del rol de servicio: Crear un nuevo rol de servicio con permisos completos.
- Nombre del nuevo rol: Se puede asignar cualquier nombre deseado; en este caso, se utilizó MediaConvert_DOCs_Role.
- Origen de etiquetas de facturación: Seleccionar “Trabajo”, ya que en este caso se van a crear trabajos (*create_job*) y se desea que la facturación esté asociada a cada uno de ellos.

A continuación, se muestra una captura de la interfaz web con la configuración establecida, justo antes de confirmar la creación del rol ARN:

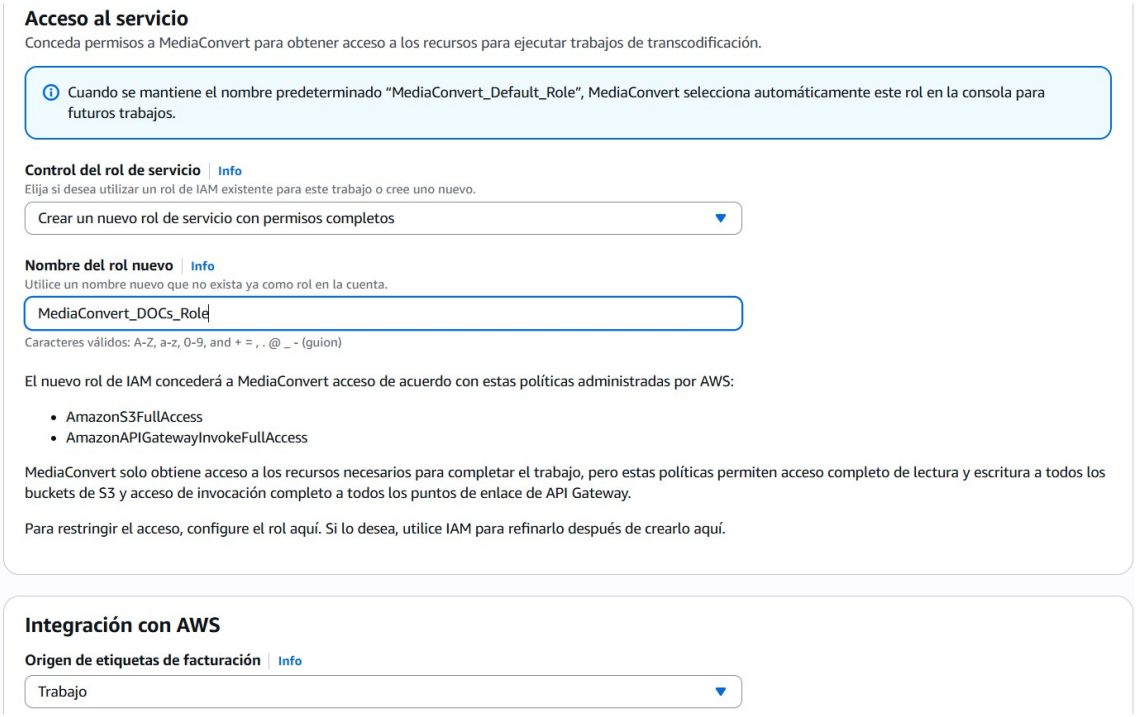


Figura 22. Configuración del rol ARN de AWS MediaConvert

Una vez se hace clic en 'Crear', se genera y muestra el rol ARN junto con los permisos asignados.

Ejemplo de ARN creado: arn:aws:iam::778564179299:role/service-role/MediaConvert_DOCs_Role

7.2.1.3 Creación del cliente

Para realizar llamadas HTTP REST a la API de AWS MediaConvert a través del SDK de AWS para Python (*Boto3*), es necesario crear un cliente.

Por motivos de seguridad y eficiencia, se evita crear el cliente múltiples veces, ya que este se utiliza en distintos puntos del proyecto y podría ser invocado repetidamente. Por ello, se define en un archivo unificado, donde se establecen diversos parámetros de configuración.

Este archivo se ejecuta una única vez al iniciar la máquina, lo que permite optimizar recursos y garantiza una gestión eficiente del cliente.

Para definir el cliente de MediaConvert, primero es necesario establecer la sesión de *AWS Boto3* de la siguiente manera:

```
session = boto3.Session(

    region_name="eu-central-1",

    aws_access_key_id="clave_acceso",

    aws_secret_access_key="clave_secreta",

)
```

Código 5. Establecimiento de la sesión con AWS Boto3.

Una vez establecida la sesión con las claves de acceso y definida la región (que puede ser la del ejemplo u otra diferente), se procede a crear el cliente de MediaConvert de la siguiente manera:

```
media_convert_client = session.client("mediaconvert", region_name="eu-central-1")
```

Código 6. Creación del cliente de AWS MediaConvert.

La región puede ajustarse según las necesidades del proyecto. En este caso, se ha seleccionado Europa Central por razones de eficiencia, ya que al ser la zona más cercana se reduce la latencia en comparación con otras regiones.

7.2.2 Rekognition de AWS

Amazon Rekognition es un servicio de AWS para el análisis de imágenes y videos basado en machine learning.

En este proyecto, se utiliza para detectar los fragmentos de la grabación de la reunión en los que se comparte pantalla. Para identificar estos fragmentos, se analizan los cambios de escena presentes en el video.

Al igual que en el caso de MediaConvert, las operaciones que utilizan este servicio se basan en una arquitectura serverless.

7.2.2.1 Configuración de permisos

Para poder utilizar esta herramienta, es necesario configurar los permisos adecuadamente.

En el proyecto DOCs, al igual que con MediaConvert, este servicio nunca se había usado, por lo que no existían permisos configurados que permitieran su funcionamiento.

El usuario de AWS que se utiliza debe contar con dos permisos fundamentales: **rekognition:StartSegmentDetection** y **rekognition:GetSegmentDetection**.

El nombre de cada permiso indica la funcionalidad a la que otorga acceso: El primero permite iniciar la detección de segmentos de cambio de escena, y el segundo permite obtener los resultados de dicha detección.

Para asignar estos permisos, se deben seguir los mismos pasos que se detallaron para el servicio MediaConvert. En este caso, con AWS Rekognition, no es necesario crear un rol ARN.

7.2.2.1.1 Creación del cliente

Para crear el cliente que permite realizar peticiones a la API de AWS Rekognition, se utiliza la misma sesión mostrada previamente en el caso de MediaConvert. La creación del cliente se realiza de forma similar, con la única diferencia de que, en este caso, en lugar de usar el nombre de servicio *"mediaconvert"*, se debe usar *"rekognition"*.

7.2.3 SQS y SNS de AWS

En la implementación inicial del proyecto, la detección de la finalización de los trabajos en MediaConvert se realizaba mediante polling activo, es decir, haciendo consultas periódicas para verificar el estado del trabajo mediante la API de MediaConvert o Rekognition. Sin embargo, este método presentaba varias desventajas significativas:

- **Ineficiencia en recursos:** Cada consulta de estado consume ciclos de CPU y ancho de banda.
- **Latencia en la detección:** Existe un retraso entre la finalización real del trabajo y el momento en que se detecta, dependiendo del intervalo de consulta.
- **Costo operativo:** Las llamadas repetidas a la API de MediaConvert y Rekognition son muy costosas.
- **Escalabilidad limitada:** No resulta óptimo para cargas de trabajo altas o procesos paralelos.

Por lo tanto, se puede observar perfectamente que este método es poco preciso, desperdicia tiempo de procesamiento valioso y puede afectar negativamente la experiencia del usuario.

Para mejorar esta situación, se optó por un enfoque que combina eventos con polling optimizado, utilizando Amazon SNS para enviar notificaciones cuando cambian los estados de los trabajos y Amazon SQS para recibirlas de forma fiable.

Si bien esta solución no elimina por completo el polling, ya que la función sigue consultando la cola SQS para leer los mensajes, reduce significativamente la frecuencia de esas consultas. De esta manera, se disminuye el consumo de recursos y los costos, y se consigue un sistema más desacoplado y escalable. Además, el coste de usar los servicios SQS + SNS es mucho más económico frente a realizar múltiples llamadas de consulta del estado del trabajo en MediaConvert o Rekognition. Aunque la detección no es estrictamente en tiempo real, la latencia es mucho menor y el proceso resulta más eficiente y económico.

Para poder usar estos servicios, es necesario configurar ciertos permisos, al igual que con los demás servicios de AWS explicados anteriormente. Cabe destacar que SQS y SNS ya se habían utilizado previamente en el proyecto DOCs, por lo que los permisos ya estaban configurados. No obstante, aquí se listan los permisos que se requieren para la funcionalidad desarrollada:

- Para SQS: **sqs:CreateQueue, sqs:DeleteQueue, sqs:GetQueueAttributes, sqs:SetQueueAttributes, sqs:ReceiveMessage y sqs:DeleteMessage**.
- Para SNS: **sns:CreateTopic, sns:Subscribe, sns:Unsubscribe y sns:GetTopicAttributes**.

La configuración de estos permisos se realiza de la misma manera que en los casos explicados anteriormente y no es necesaria la creación de un rol ARN.

En resumen, este enfoque basado en eventos con polling optimizado ofrece un equilibrio eficiente entre costos, latencia y escalabilidad, mejorando considerablemente la experiencia de usuario frente al polling activo tradicional.

Aunque se consideraron alternativas más avanzadas como Amazon EventBridge o Step Functions para gestionar la finalización de trabajos, se decidió optar por la combinación de SNS + SQS debido a su simplicidad operativa, menor complejidad arquitectural y mejor relación costo-beneficio. Un sistema completamente event-driven con Lambda u otros mecanismos reactivos implicaría costos adicionales y una complejidad innecesaria para la magnitud y frecuencia de los procesos que se manejan. La solución con SNS y SQS ofrece un equilibrio óptimo al brindar las ventajas de un enfoque basado en eventos, pero sin la sobrecarga de configuración y costos que requieren las alternativas más sofisticadas, alineándose así con el principio de simplicidad tecnológica y garantizando eficiencia y facilidad de mantenimiento para el caso específico de notificación de estados en trabajos multimedia.

7.2.3.1.1 Creación del cliente

Para crear el cliente que permite realizar peticiones a la API de SQS o SNS, se utiliza la misma sesión mostrada previamente en el caso de MediaConvert. La creación del cliente se realiza de forma similar, con la única diferencia de que, en este caso, en lugar de usar el nombre de servicio *"mediaconvert"*, se debe usar *"sqs"* y *"sns"* respectivamente.

7.2.4 Funciones

En esta sección se describe la implementación de las distintas funciones de procesamiento de audio y video, utilizando dos tecnologías diferentes: AWS y FFmpeg.

Las funciones que se presentan a continuación utilizan los mismos parámetros de entrada tanto en su implementación con AWS como con FFmpeg. Esto permite mantener una interfaz común y facilita tanto la configuración como la comparación de ambas tecnologías, que se realiza al final del proyecto.

Las funciones implementadas incluyen: Separar el audio del video, normalizar un audio, reducir el ruido de un audio, extraer un frame de un video y detectar cambios de escena en un video, lo cual permite identificar los momentos de compartición de pantalla.

7.2.4.1 Parámetros de entrada de las funciones

Separar el audio del video:

- Video como objeto FileData del cual se debe extraer el audio.
- El formato de salida del audio (flac, wav o mp3).
- Cantidad de canales: 1 (mono) o 2 (estéreo).
- Frecuencia de muestreo (sample rate).
- Profundidad de bits (bit Depth).

Normalización de un audio: Esta función tiene los mismos parámetros que la anterior de separar el audio del video, con la diferencia de que el archivo de entrada como objeto FileData es un audio en lugar de ser un video.

Hay 3 parámetros adicionales relacionados con la normalización:

- I value (Target Loudness): Nivel de sonoridad objetivo en LUFS. Controla el volumen promedio del audio.
- Tp value (True Peak Limit): Límite máximo de picos en dBTP (Decibels True Peak). Previene distorsión por clipping, que es distorsión por recorte de la onda de sonido. (típico: -1 a -3 dBTP).
- LRA (Loudness Range): Rango dinámico en LU. Mide la diferencia entre partes silenciosas y fuertes (típico de 8-20 LU).

Reducción del ruido de un audio: Igual que en el caso de la normalización de audio, esta función tiene los mismos parámetros que la anterior de separar el audio del video, con la diferencia de que el archivo de entrada como objeto FileData es un audio en lugar de ser un video.

Hay 3 parámetros adicionales relacionados con la reducción de ruido:

- Frecuencia de Corte (Highpass frequency): Frecuencia mínima (Hz) que se conserva. Elimina zumbidos graves.
- Intensidad (noise reduction): Cantidad de reducción (dB). Nivel de atenuación del ruido.
- Umbral de Ruido (noise_floor): Nivel donde comienza la reducción. Sonidos bajo este umbral se consideran ruido.

Extracción de un fotograma de un video:

- Video como objeto FileData del cual se debe extraer el fotograma.
- Fotograma especificado mediante timestamp en formato HH:MM:SS.sss que se desea extraer del video.
- Formato de salida del fotograma, por ejemplo .jpg.

Detección de segmentos de cambios de escena:

- Video como objeto FileData en el cual detectar los cambios de escena.
- Umbral de escena (shot threshold) es un valor entre 0 y 1 que determina la sensibilidad para detectar cambios de escena en un video; valores más bajos detectan más cortes (más sensible), mientras que valores más altos detectan solo cambios más evidentes.

7.2.4.2 AWS

Como paso inicial, se detallan a continuación las librerías empleadas en la implementación de estas funciones desarrolladas en Python.

- **boto3**: Cliente oficial de AWS para Python, que permite interactuar con MediaConvert, Rekognition, S3, SNS y SQS.
- **uuid**: Para generar identificadores únicos y evitar colisiones en nombres de archivos temporales.
- **json**: Para manejar estructuras de datos.
- **os**: Para obtener variables de entorno (como el bucket de S3 o la región).

- **time**: Para gestionar el tiempo de espera al recibir notificaciones de finalización del trabajo.

7.2.4.2.1 Estructura común de las funciones

Las funciones que permiten realizar tareas de procesamiento audiovisual con AWS se han diseñado siguiendo un patrón modular y reutilizable, que permite configurar y lanzar trabajos dinámicamente según los parámetros recibidos. A nivel general, todas las funciones se estructuran en base a los siguientes bloques:

```
try:

    # 1. Proceso de validación de parámetros

    # 2. Gestión de almacenamiento temporal en S3

    # 3. Creación de recursos temporales SQS y SNS

    # 4. Configuración y emisión de trabajos

    # 5. Espera de la finalización del trabajo

    # 6. Descarga y retorno de resultados

finally:

    # 7. Limpieza de recursos temporales
```

Código 7. Estructura común procesamiento multimedia con AWS.

1. Proceso de validación de parámetros

Antes de iniciar cualquier proceso, se valida que los parámetros recibidos (formato, sample rate, canales, bit depth, etc.) sean compatibles con los valores permitidos por AWS MediaConvert o Rekognition. Esta validación anticipada previene errores de ejecución más adelante y garantiza que los parámetros de entrada cumplen con los requisitos técnicos de cada operación.

Por ejemplo, la configuración de canales solo puede ser 1 o 2, y la profundidad de bits debe ser 16, 24 o 32.

Además de validar los parámetros relacionados con el contenido multimedia, también se verifica que el archivo proporcionado como parámetro sea una instancia de la clase FileData.

2. Gestión de almacenamiento temporal en S3

Para que AWS MediaConvert y Rekognition puedan acceder a los archivos, estos deben estar previamente almacenados en Amazon S3. Por ello:

- Entre los datos de entorno que se establecen al principio de todo al levantar la máquina, se extrae el nombre del bucket que se usa en el proyecto para almacenar los archivos en S3.
- Se genera una clave única (*key*), que indica la ruta al archivo dentro del bucket y el nombre del fichero, combinando un prefijo específico por función (ej: temp_audio_normalization/), un UUID y el nombre del fichero original. Por lo tanto, el archivo en S3 estará en: bucketDelProyecto/temp_audio_normalization/UUID + NombreFichero.

- Se extraen los bytes del archivo y se suben a dicha clave y bucket mediante una función reutilizable (*upload_file_to_s3*, explicada en el apartado *7.1.4 Cargar un fichero a S3*).
- Una vez completado el trabajo, los archivos temporales se eliminan automáticamente con la función *delete_file_from_s3* explicada en la misma sección que *upload_file_to_s3*.

3. Creación de recursos temporales SQS y SNS

- En primer lugar, se crea una cola SQS temporal mediante una petición al cliente de SQS con una llamada a *create_queue*, cuyo nombre incluye un identificador único UUID para evitar colisiones y garantizar su unicidad.
- Una vez creada la cola, se obtiene su URL y ARN, mediante la llamada al método *get_queue_attributes*, necesarios para la suscripción posterior.
- Se crea, mediante la llamada al método *create_topic*, un tema SNS con un nombre predefinido, como "*MediaConvertEvents*" o "*RekognitionEvents*". Este tema actuará como el canal que publicará los eventos cuando un trabajo de MediaConvert o Rekognition finalice.
- Se suscribe la cola SQS al tema SNS, mediante la llamada el método *subscribe*, lo que permite que los mensajes publicados en el tema se entreguen automáticamente a la cola. Esta suscripción utiliza el protocolo 'sqs' y como endpoint se especifica el ARN de la cola creada anteriormente.

  A continuación, se muestra el código ejemplo de la suscripción y la obtención del ARN de suscripción:

```
subscription = sns.subscribe(

        TopicArn=topic_arn,

        Protocol='sqs',

        Endpoint=queue_arn

    )

    subscription_arn = subscription['SubscriptionArn']
```

  Código 8. Suscripción de la cola SQS al tema SNS.

- Finalmente, se establece una política de acceso en la cola SQS, la cual permite explícitamente que el servicio SNS pueda enviarle mensajes. Esta política se define en formato JSON y se configura mediante *sqs.set_queue_attributes()*. Sin esta política, la cola no recibiría los mensajes del tema SNS, ya que AWS requiere una autorización explícita entre servicios.

4. Configuración y emisión de trabajos

En AWS hay que configurar lo que se llaman trabajos (Jobs), que se emiten y son ejecutados en la nube. Al finalizar el trabajo, se puede recuperar la respuesta o resultado.

Cada función genera dinámicamente una configuración (job_settings) en formato JSON, en la cual se:

- Define la entrada desde S3.
- Especifica la configuración de salida según los parámetros recibidos.
- Se asigna una ruta de salida también en S3.
- Se añade una sección específica para configurar la notificación de eventos de finalización mediante SNS.
- Se indica el rol ARN con permisos para ejecutar un trabajo de MediaConvert.

A continuación, se presenta un ejemplo de configuración del códec de audio según el formato de salida y los parámetros de entrada.

```
if format == ".mp3":
            return {
                "Codec": "MP3",
                "Mp3Settings": {
                    "Bitrate": 320000,
                    "Channels": channels,
                    "RateControlMode": "CBR"
                }}
elif format == ".wav":
            return {
                "Codec": "WAV",
                "WavSettings": {
                    "BitDepth": 24 if bit_depth > 24 else bit_depth,
                    "Channels": channels,
                    "SampleRate": sample_rate,
                    "Format": "RIFF"
                }}
        elif format == ".flac":
            return {
                "Codec": "FLAC",
                "FlacSettings": {
                    "BitDepth": 24 if bit_depth > 24 else bit_depth,
```

```
                    "Channels": channels,
                    "SampleRate": sample_rate
                }}
        else: # devolver error
```

Código 9. Configuración del códec del audio de salida según el formato de salida y los parámetros de entrada.

En el código se muestra cómo se define la configuración del códec que luego se integra al resto de la configuración que se utiliza al crear el trabajo en AWS.

Existen algunas excepciones, como en el caso del formato MP3, cuyo bitrate se fija en 32000, ya que la herramienta no permite establecer otro valor. Por otro lado, los formatos FLAC y WAV solo admiten profundidades de bits de 16 o 24. Por esta razón, si el usuario especifica una profundidad superior, se ajusta automáticamente a 24.

Luego de establecer la configuración en JSON se debe iniciar el trabajo.

- **MediaConvert:** Se debe crear el trabajo con una llamada HTTP REST a la API de AWS MediaConvert a través del SDK de AWS para Python (*Boto3*). Anteriormente se ha mostrado como se crea el cliente de MediaConvert y aquí se usa para hacer la llamada a *create_job*.

  ```
  job_response = mediaconvert.create_job(
                  Role=role_arn,
                  Settings=job_settings,
                  AccelerationSettings={"Mode": "PREFERRED"}
              )
  ```

  Código 10. Creación del trabajo de MediaConvert.

  En esta llamada para crear el trabajo hay que indicar el rol ARN, que anteriormente se ha explicado cómo se crea, el JSON con la configuración del trabajo y en mi caso, añadí la configuración de que preferiblemente si es posible que se active el modo acelerado para obtener resultados rápidos.

  Para ver con más detalle cómo se configuran y manejan los trabajos (Jobs) de MediaConvert se pueden consultar las siguientes fuentes oficiales [47] [48].

- **Rekognition:** Para iniciar la detección de segmentos de cambio de escena en un video almacenado en S3, se debe hacer una llamada HTTP REST a la API de AWS Rekognition a través del SDK de AWS para Python (*Boto3*). Previamente se ha mostrado cómo se crea el cliente de Rekognition, y en este punto se utiliza dicho cliente para invocar el método *start_segment_detection*.

  ```
  response = rekognition.start_segment_detection(job_settings)
  ```

  Código 11. Inicio del trabajo de Rekognition.

  En esta llamada simplemente hay que indicar el JSON con la configuración del trabajo de detección de los segmentos de los cambios de escena.

Para ver con más detalle cómo se configuran y manejan los trabajos (Jobs) de Rekognition se pueden consultar las siguientes fuentes oficiales [49] [50].

5. **Espera de la finalización del trabajo**

Se implementa un sistema basado en SNS y SQS:

- Al emitir el trabajo, como se ha especificado en el punto 4, se especifica que se notifique en el tópico de la SNS al finalizar.
- El proceso escucha pasivamente en la cola haciendo llamada al método *receive_message*, esperando el mensaje que indique la finalización exitosa (o fallo) del trabajo.
- Esto permite una espera eficiente, desacoplada y escalable del estado de cada trabajo.

6. **Descarga y retorno de resultados**

Cuando el trabajo se completa correctamente, se descarga el archivo resultante desde S3, se crea un objeto FileData que contiene el archivo en bytes, se le asigna un nombre y se especifica su tipo MIME. Este objeto se devuelve junto con la duración del trabajo y otras métricas relevantes.

7. **Limpieza de recursos temporales**

Para no dejar residuos:

- Se elimina la cola SQS y la suscripción SNS.

```
sns.unsubscribe(SubscriptionArn=subscription_arn)

sqs.delete_queue(QueueUrl=queue_url)
```

Código 12. Eliminación de la cola SQS y suscripción SNS.

- Se eliminan los archivos temporales de entrada y salida en S3 debido a que el resultado se devuelve por parámetro. Para eliminar los archivos en S3 se usa la función general *delete_file_from_s3* presentada anteriormente.

7.2.4.2.2 Particularidades de cada función

Aunque la estructura general es común, cada función tiene sus particularidades, tanto a nivel de configuración del trabajo como en los objetivos que persigue.

Separación del audio del video:

- Objetivo: Extraer la pista de audio de un archivo de video y almacenarla como archivo de audio independiente.
- MediaConvert se configura para deshabilitar la salida de video y codificar solo el audio en el formato y configuración especificadas.

Normalización de un audio:

- Objetivo: Ajustar automáticamente el volumen del audio a un nivel estándar.
- Se activa la configuración AudioNormalizationSettings dentro del trabajo de MediaConvert:

```
{  "Algorithm": "ITU_BS_1770_3",
```

```
"AlgorithmControl": "CORRECT_AUDIO",
…}
```

Código 13. Porción de la configuración de normalización en MediaConvert.

Extracción de una imagen (frame o fotograma) de un video:

- Objetivo: Obtener un fotograma del video en un instante determinado.
- Se configura MediaConvert para extraer un frame y la salida en formato imagen (.jpg, .png).
- Esta función es clave para el análisis estático del video.

Detectar segmentos de cambios de escena:

- Objetivo: Identificar automáticamente los momentos en los que cambia de escena un video (útil para detectar cortes o transiciones en una grabación).
- Se utiliza AWS Rekognition en lugar de MediaConvert.
- Se ejecuta la función *start_segment_detection* indicando que el tipo de segmento es SHOT.
- Esta función puede utilizarse como base para detectar momentos de compartición de pantalla.

7.2.4.2.3 Conclusión

La implementación de estas funciones sigue una arquitectura serverless bien estructurada, reutilizando patrones comunes como:

- Subida temporal a S3,
- Configuración dinámica de trabajos,
- Espera de la finalización de los trabajos,
- Limpieza automática de recursos.

Este diseño garantiza eficiencia, modularidad y bajo acoplamiento, lo que facilita el mantenimiento y escalado del sistema. Gracias a la abstracción mediante funciones auxiliares reutilizables, es posible extender estas capacidades a nuevos casos de uso con un mínimo esfuerzo.

7.2.4.3 FFmpeg

Como paso inicial, se detallan a continuación las librerías empleadas en la implementación de estas funciones desarrolladas en Python.

- **tempfile**: Para crear directorios y archivos temporales donde se procesan los datos antes de devolverlos.
- **subprocess:** Para ejecutar comandos externos (en este caso, ffmpeg y ffprobe).
- **os**: Para manejar rutas y operaciones con el sistema de archivos.
- **re**: Para hacer búsquedas con expresiones regulares (por ejemplo, extraer los segundos reales del timestamp en la detección de cambios de escena).

Dependencias externas:

- **FFmpeg** y **FFprobe**: No son librerías de Python, pero son ejecutables externos que deben estar instalados en el sistema para que el código funcione.

7.2.4.3.1 Estructura común de las funciones

La implementación de las distintas funciones de procesamiento de audio y video comparte una arquitectura y un flujo de trabajo muy similar. Este diseño modular facilita la reutilización de código y permite que los cambios en la lógica base se apliquen automáticamente a todas las operaciones, además de permitir configurar y ejecutar comandos de procesamiento multimedia dinámicamente según los parámetros recibidos.

A nivel general, todas las funciones se estructuran en base a los siguientes pasos:

```
try:

    # 1. Proceso de validación y configuración de parámetros

    # 2. Gestión de almacenamiento temporal

    # 3. Construcción dinámica del comando FFmpeg

    # 4. Sistema de ejecución síncrona

    # 5. Recuperación, encapsulación y retorno de resultados

finally:

    # 6. Limpieza de recursos temporales
```

Código 14. Estructura común procesamiento multimedia con FFmpeg.

1. Proceso de validación y configuración de parámetros

Antes de ejecutar cualquier operación, cada función valida rigurosamente los parámetros recibidos.

- Se verifica que los formatos solicitados estén soportados.
- Se controla que valores como canales, frecuencia de muestreo o profundidades de bits estén en rangos válidos.
- Se valida que el fichero que se pasa por parámetro es una instancia de la case FileData.
- En parámetros especiales (como frame o shot_threshold) se usan funciones de validación específicas para evitar ejecuciones con datos inválidos.

2. Gestión de almacenamiento temporal

Todas las funciones utilizan el sistema de archivos local como buffer temporal (directorios temporales) para el procesamiento.

- El archivo de entrada se guarda físicamente en disco para que FFmpeg pueda procesarlo.
- Los nombres de archivo se generan de forma controlada, asegurando que son únicos, para evitar colisiones.

- Los resultados procesados se leen desde el archivo temporal y se encapsulan en un objeto FileData.

3. Construcción dinámica del comando FFmpeg

El núcleo de la implementación reside en la construcción dinámica del comando FFmpeg. A este comando se le pueden añadir diversas directivas según la configuración indicada por parámetro, lo que facilita la realización de pruebas posteriores y permite que, en el futuro, pueda adaptarse mejor a las necesidades del momento.

El sistema genera una estructura de comando compleja que incluye:

- Definición de inputs (-i): Especifica la ubicación del archivo fuente en el sistema temporal y configura los parámetros de entrada.
- Configuración de outputs: Define los parámetros de salida, incluyendo formato, calidad y ubicación de destino.
- Mapeo de codecs: Utiliza un sistema dinámico de configuración de codecs basado en el formato de salida deseado.
- Se añaden flags específicos según la operación (por ejemplo, *-vn* para ignorar video o *-vframes* 1 para extraer una sola imagen).

En este punto, se aplica la configuración establecida según los parámetros descritos en el apartado **1. Proceso de validación y configuración de parámetros**, añadiendo las opciones del comando FFmpeg correspondientes a dicha configuración.

Un ejemplo de comando FFmpeg es el siguiente:

```
ffmpeg -i mi_video.mp4 -ac 1 -ar 16000 -vn -sample_fmt s16p -c:a flac
audio_extraido.flac
```

Código 15. Ejemplo de comando FFmpeg.

Para ver con más detalle cómo se configura y las diferentes opciones de FFmpeg, se puede consultar la fuente oficial con la documentación completa utilizada para este trabajo [51].

Es **muy importante mantener un orden correcto en los parámetros del comando FFmpeg**. De lo contrario, aunque el comando se ejecute sin mostrar errores, podría no realizar la tarea esperada.

Esto se ha comprobado personalmente: En algunas pruebas, los resultados obtenidos no tenían sentido, lo que me llevó a revisar la implementación. Fue entonces cuando descubrí que, en ciertos casos, no estaba respetando el orden correcto de los parámetros y que, aunque el comando no mostrara advertencias, en realidad no funcionaba como era esperado.

Figura 23. Orden de los parámetros del comando FFmpeg

4. Sistema de ejecución síncrona

- El método *run_ffmpeg_command* centraliza la ejecución, capturando tanto salida estándar como errores. Se ejecuta inmediatamente el comando mediante la invocación del subproceso.

- Si FFmpeg devuelve un código distinto de cero, se lanza una excepción con el detalle del error (stderr).
- El procesamiento es bloqueante, asegurando que el resultado final esté completo antes de devolverlo.

5. **Recuperación, encapsulación y retorno de resultados**
   - Tras la ejecución, el archivo resultante se lee desde el sistema de archivos temporal.
   - El contenido binario se encapsula en un objeto FileData, acompañado de su nombre y del *content_type* correspondiente, y se devuelve como resultado.
6. **Limpieza de recursos temporales**

Al finalizar todo el proceso descrito anteriormente, el directorio temporal se limpia automáticamente, eliminando los archivos generados durante el proceso. Este paso garantiza que no queden residuos innecesarios, optimiza el uso del espacio en disco y contribuye a mantener el sistema organizado y en buen estado de funcionamiento.

7.2.4.3.2 Particularidades de cada función

En las siguientes funciones, al explicar sus particularidades, se omite mencionar el objetivo, que difiere en cada una, ya que este ha sido descrito previamente en el apartado de particularidades de cada función dentro de la implementación en AWS.

Separación del audio del video: Añade parámetros como *-vn* al comando para eliminar la pista de video y *-sample_fmt* si aplica.

Normalización y reducción de ruido de un audio:

- Se añade al comando el algoritmo de normalización o de reducción de ruido, según corresponda, que se utilizará en el proceso.
- Se incorpora al comando final la configuración de los parámetros de normalización o reducción de ruido, según corresponda, indicados en los parámetros de entrada de la función.

Extracción de una imagen (frame o fotograma) de un video:

- Valida el formato de imagen y el timestamp (frame) usando una función central de validación.
- Añade *-ss* para posicionarse en el tiempo exacto (timestamp) y *-vframes 1* para limitar la captura a un único frame.

Ejemplo de comando de extracción de un frame:

```
ffmpeg -ss 00:00:10.750 -i video.mp4 -vframes 1 frame.jpg
```

Código 16. Ejemplo de comando ffmpeg de extracción de un fotograma de un video.

Detectar segmentos de cambios de escena:

- Combina dos pasos:
  1. Obtiene la duración del video usando *ffprobe*.
  2. Ejecuta FFmpeg con un filtro *select='gt(scene,threshold)'* para detectar cambios significativos en la imagen.

- Procesa las marcas de tiempo encontradas y construye una lista de segmentos con inicio, fin y duración.

7.2.4.3.3 Conclusión

Igual que en el caso de la implementación con AWS, la implementación de estas funciones está bien estructurada, reutilizando patrones comunes como:

- Crear los directorios temporales,
- Construcción dinámica del comando FFmpeg,
- Ejecución síncrona,
- Limpieza automática de recursos.

Este diseño garantiza eficiencia, modularidad y bajo acoplamiento, lo que facilita el mantenimiento y escalado del sistema. Gracias a la abstracción mediante funciones auxiliares reutilizables, es posible extender estas capacidades a nuevos casos de uso con un mínimo esfuerzo.

7.3 Transcripción y diarización

En esta sección se describe la implementación de la función de transcripción y diarización. Para llevar a cabo esta tarea, se ha utilizado el servicio ***Amazon Transcribe*** de AWS.

7.3.1 Amazon Transcribe

Amazon Transcribe es un servicio de AWS que permite transcribir audios, especificar el idioma de la transcripción y separar el texto en diálogos identificando a los diferentes hablantes (diarización).

A continuación, se detallan diversos aspectos sobre este servicio.

7.3.1.1 Configuración de permisos

Para utilizar este servicio de AWS, es necesario configurar dos permisos principales: transcribe:StartTranscriptionJob y transcribe:GetTranscriptionJob.

- **transcribe:StartTranscriptionJob:** Permite al usuario iniciar trabajos de transcripción.
- **transcribe:GetTranscriptionJob:** Permite al usuario consultar el estado y los resultados de un trabajo de transcripción.

Para asignar estos permisos, se deben seguir los mismos pasos que se detallaron para el servicio *MediaConvert*. En este caso, con Amazon Transcribe, no es necesario crear un rol ARN.

7.3.1.2 Creación del cliente

Para crear el cliente que permite realizar peticiones a la API de Amazon Transcribe, se utiliza la misma sesión mostrada previamente en el caso de AWS MediaConvert, Rekognition, SNS y SQS.

La creación del cliente se realiza de forma similar, con la única diferencia de que, en este caso, se debe usar el nombre de servicio *"transcribe"*. A continuación, se muestra la llamada:

```
    transcribe_client = session.client("transcribe", region_name="eu-
central-1")
```

Código 17. Creación del cliente de Amazon Transcribe.

Como se mencionó anteriormente, la región puede ajustarse según las necesidades del proyecto. En este caso, se ha seleccionado Europa Central por razones de eficiencia, ya que, al ser la zona geográficamente más cercana, se reduce la latencia en comparación con otras regiones.

7.3.1.3 Vocabulario personalizado

La función que he implementado permite al usuario proporcionar, como parámetro, una lista o un archivo txt, como objeto FileData, con vocabulario personalizado que puede incluir términos técnicos, nombres propios, acrónimos, entre otros, para que sean considerados durante el proceso de transcripción.

Esta funcionalidad es especialmente útil porque existen numerosas palabras técnicas y nombres que pueden sonar similares o que, siendo idénticos fonéticamente, se escriben de manera diferente según el idioma. Diversos factores pueden causar errores en la transcripción de vocabulario especializado, por lo que ofrecer la posibilidad de especificar un vocabulario personalizado resulta la mejor solución para abordar esta problemática.

Como es habitual en el ecosistema AWS, es necesario configurar permisos adicionales para gestionar este vocabulario personalizado.

Permisos del usuario AWS:

Para poder utilizar este servicio, es necesario configurar cuatro permisos: transcribe:CreateVocabulary, transcribe:DeleteVocabulary, transcribe:GetVocabulary y transcribe:ListVocabularies.

Como indican sus nombres, estos permisos permiten crear, eliminar, obtener y listar los vocabularios disponibles en Amazon Transcribe.

La asignación de estos permisos se realiza siguiendo los mismos pasos detallados previamente para el servicio *MediaConvert*.

Caracteres especiales:

Durante las pruebas realizadas para verificar el correcto funcionamiento de la función, al especificar el vocabulario personalizado surgió un problema específico: Existen ciertos caracteres especiales que AWS no soporta, y que varían según el idioma con el que se esté trabajando.

Dado que existen miles de idiomas a nivel mundial, resulta inviable implementar un sistema de normalización o análisis sintáctico que valide si el vocabulario contiene caracteres no aceptados.

Para resolver este problema, se implementó una solución basada en la captura de errores. Cuando AWS rechaza el vocabulario debido a caracteres especiales no aceptados, el sistema detecta este error y lanza un aviso al usuario, indicando que debe sustituir o eliminar dichos caracteres. Además, se proporciona la fuente oficial de AWS donde se detallan los

caracteres aceptados por idioma, facilitando así que el usuario pueda consultar fácilmente qué debe modificar en su vocabulario.

La fuente oficial con esta información, que se proporciona al usuario, es la siguiente [52].

7.3.2 Implementación de la función

Librerías usadas

- **boto3**: SDK oficial de AWS para Python, utilizado para interactuar con Amazon Transcribe, S3, SNS y SQS.
- **time**: Para manejo de tiempos y pausas.
- **uuid**: Para generar identificadores únicos.
- **os**: Para acceso a variables de entorno.
- **json**: Para manipulación de datos JSON.
- **botocore.exceptions.ClientError**: Para manejo de errores en llamadas AWS.

A continuación, se describen los pasos que realiza la función durante su implementación para llevar a cabo la transcripción y diarización:

Parámetros de la función

- file_data: El audio a transcribir, proporcionado como un objeto FileData.
- language_code: Código del idioma para la transcripción.
- media_format: Formato del audio de entrada para la transcripción y diarización.
- vocabulary: Parámetro opcional que puede ser una lista o un fichero (FileData) en formato texto con un vocabulario personalizado.

Configuración y subida de archivos a S3

Tal y como se explicó en la implementación de las funciones de procesamiento multimedia, para que AWS pueda procesar un archivo, este debe estar alojado en S3.

Por lo tanto, es necesario subir el audio que se desea transcribir a S3 y, una vez finalizado el proceso, eliminarlo para no mantener archivos innecesarios en la nube.

El nombre del archivo se genera utilizando un UUID, siguiendo el patrón: bucketDelProyecto/temp_transcription/UUID + NombreFichero.

Después de obtener este nombre único, se invoca la función de subida a S3. Para más detalles sobre esta función, consultar el apartado *7.1.4 Cargar un fichero a S3*.

Creación de vocabulario personalizado

Para mantener la función principal limpia y mantenible, la creación del vocabulario personalizado se implementa en una función auxiliar.

Primero se verifica si el parámetro es una lista o un fichero. Si es un fichero, se valida que sea un archivo de texto; de lo contrario, se lanza un error con el mensaje correspondiente.

En caso de ser un fichero de texto válido, se decodifica su contenido en UTF-8 y se convierte en una lista de términos usando los saltos de línea como separadores.

Tras validar el tipo, formato y que el vocabulario no esté vacío, se crea el vocabulario mediante una llamada a la API de AWS Amazon Transcribe a través del SDK de Python (*boto3*), como se muestra a continuación:

```
transcribe_client.create_vocabulary(

                VocabularyName=active_vocabulary_name,

                LanguageCode=language_code,

                Phrases=final_vocabulary

            )
```

Código 18. Creación del vocabulario personalizado de Amazon Transcribe.

Para evitar conflictos de nombres, se genera un nombre único con el formato custom_vocab + UUID, usando el mismo UUID asignado al archivo temporal en S3.

El código del lenguaje es un campo obligatorio que el usuario debe indicar por parámetro de entrada de la función, debido a que es el usado en la transcripción y Phrases es una lista con el vocabulario personalizado proporcionado por el usuario.

Gestión de vocabulario personalizado

La creación del vocabulario no es inmediata, por lo que se espera hasta que esté listo para proceder con la transcripción.

Desde la función encargada de crear el vocabulario, se invoca otra función auxiliar que monitoriza el estado del vocabulario:

- Si el estado es FAILED, se lanza un error.
- Si está PENDING, se continúa esperando.
- Si es READY, se termina la espera y continúa la ejecución.

Configuración del trabajo de Amazon Transcribe

Se destacan los siguientes parámetros de configuración:

ShowSpeakerLabels: True para habilitar diarización y mostrar etiquetas como spk_0, spk_1, etc.

ChannelIdentification: True para mejorar la diarización en audios multicanal, ya que el reconocimiento de canales puede contribuir significativamente a este proceso. No obstante, cabe destacar que las grabaciones de Teams son monocanal, hecho que se ha comprobado revisando las propiedades de dichas grabaciones, como se muestra a continuación:

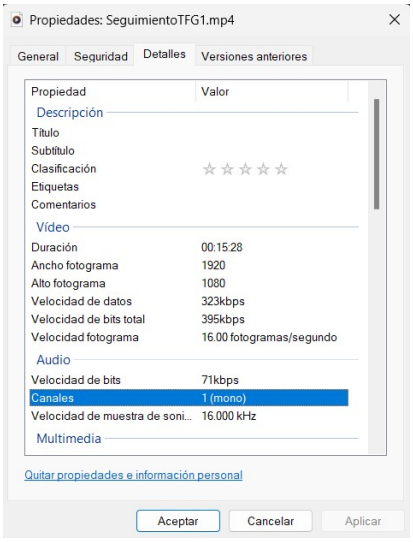


Figura 24. Propiedades de una grabación de Teams, en donde se muestra que el audio es monocanal

ShowAlternatives: True para mostrar alternativas de transcripción por palabra.

MaxAlternatives: Máximo 3 alternativas por palabra.

VocabularyName: Nombre del vocabulario personalizado, en caso de haberse creado; de lo contrario, no se incluye

Ejecución del trabajo de transcripción

Se inicia el trabajo con una llamada a la API de AWS Transcribe:

```
response = transcribe_client.start_transcription_job(

        TranscriptionJobName=job_name_id,

        Media={"MediaFileUri": s3_url},

        MediaFormat=media_format,

        OutputBucketName=Bucket,

        LanguageCode=language_code,

        Settings=settings,

    )
```

Código 19. Inicio del trabajo de transcripción en Amazon Transcribe.

Se pasa el nombre único del trabajo, la URL del audio en S3, el formato, el bucket para la salida, el idioma y la configuración.

Sistema de monitoreo

En este caso, se emplea el mismo método para esperar la finalización del trabajo de transcripción que el utilizado en el procesamiento multimedia, basado en SQS y SNS. Por ello, a continuación, se ofrece un breve resumen para evitar repetir innecesariamente la explicación completa.

- Al inicio, se deben crear la cola SQS, el tema SNS, la suscripción y todos los elementos necesarios, de la misma forma que en el caso del procesamiento multimedia.
- Al emitir el trabajo de Amazon Transcribe, en su configuración, se especifica que se notifique en el tópico de la SNS al finalizar.
- Este tema SNS está suscrito a una cola SQS, desde donde se lee los mensajes para detectar el estado.
- La función procesa los mensajes recibidos, gestionando estados COMPLETED o FAILED.
- Si el estado es FAILED, se lanza una excepción con la razón.
- Al completarse, se continúa con la descarga del resultado.
- Al final, se eliminan la cola, el tema SNS y la suscripción asociada, con el fin de liberar recursos y evitar costos innecesarios.
- Esto permite una espera eficiente, desacoplada y escalable del estado de cada trabajo.

Procesamiento de resultados

El resultado del trabajo, obtenido mediante una llamada al método *get_transcription_job* de la API, incluye metadatos sobre la duración del audio y las marcas de tiempo. Esta información también se recopila e incorpora a las métricas que el método devuelve.

La transcripción se descarga desde S3, en un archivo JSON con el nombre job_name_id.json. Este JSON se devuelve como objeto FileData para mantener la uniformidad.

Limpieza

Para evitar acumular archivos en S3, se eliminan tanto el archivo de audio original como el JSON de la transcripción.

Si se creó un vocabulario personalizado, también se elimina mediante una llamada a la API:

```
    transcribe_client.delete_vocabulary(VocabularyName=active_vocabulary_n
ame)
```

Código 20. Eliminación del vocabulario personalizado creado en AmazonTranscribe.

7.4 Reducir tamaño de transcripción

En la sección *Stream de DOCs* se presenta el stream (grafo dirigido) con diversos nodos, entre los cuales se encuentra el nodo *25. Reducir tamaño transcripción*.

En este apartado se describe la implementación de la función responsable de reducir el tamaño de la transcripción obtenida a través del servicio Amazon Transcribe, específicamente en su modalidad con diarización. El propósito principal de esta etapa es optimizar el almacenamiento y la manipulación de la información, eliminando datos no esenciales y conservando únicamente los fragmentos relevantes para el análisis posterior.

Librerías utilizadas

Para esta función no se hace uso directo de librerías externas, ya que su implementación se apoya exclusivamente en funcionalidades nativas de Python para la manipulación de estructuras de datos tipo diccionario.

Objetivo del nodo

El formato original devuelto por Amazon Transcribe incluye múltiples elementos: El texto transcrito, información de puntuación y confianza, metadatos del trabajo, información sobre los canales, así como estructuras auxiliares necesarias para casos de uso avanzados. Sin embargo, para este proyecto, solo resultan esenciales:

- transcripts: Texto transcrito de forma íntegra.
- audio_segments: Información segmentada del audio, incluyendo la diarización con la asignación de cada segmento a un hablante y sus marcas temporales.
- Confianza: Se añade un campo denominado confianza en la sección audio_segments, calculado como la media de la confianza por palabra que proporciona Amazon Transcribe. De esta manera, se obtiene un único valor de confianza promedio por segmento, en lugar de disponer de un valor individual para cada palabra.
- Palabra más confiable: Se ha verificado que, tanto en el texto de transcripts como en los segmentos de audio_segments, entre todas las posibles alternativas detectadas para una misma palabra, se selecciona aquella con el mayor valor de confianza.

La función tiene como objetivo filtrar y simplificar la transcripción original, conservando únicamente estas secciones relevantes. Esto reduce significativamente el tamaño del objeto en memoria y agiliza operaciones posteriores.

Implementación

El funcionamiento de la función comienza accediendo a la sección principal del resultado devuelto por Amazon Transcribe, almacenada en la clave "results". Esta sección agrupa todos los datos relevantes generados por el servicio, por lo que el primer paso consiste en extraerla para poder manipular sus elementos internos.

A continuación, se construye un nuevo diccionario reducido que contiene únicamente las claves necesarias para el proyecto: transcripts, que almacena el texto completo de la transcripción, y audio_segments, que recoge la información segmentada del audio junto con la asignación de hablantes y marcas temporales.

Además, dentro de la sección audio_segments se incorpora un nuevo campo denominado confianza, cuyo valor se calcula a partir de las puntuaciones de confianza por palabra proporcionadas por Amazon Transcribe. El servicio asigna a cada palabra reconocida un valor numérico que representa el grado de certeza de la predicción, comprendido entre 0 y 1.

Para obtener la confianza a nivel de segmento, se realiza una media aritmética de los valores de confianza de todas las palabras incluidas en dicho segmento. Este cálculo permite sintetizar la fiabilidad de la transcripción en un único valor representativo por fragmento de audio, evitando así trabajar con listas extensas de puntuaciones individuales.

Esta optimización no solo reduce la complejidad de la información, sino que también facilita la interpretación de los resultados y permite aplicar criterios de filtrado o validación basados en un umbral de confianza global por segmento, en lugar de evaluar palabra por palabra.

Una vez creada la estructura reducida con la inclusión del campo confianza, la función devuelve este nuevo diccionario, listo para ser utilizado en etapas posteriores del flujo de procesamiento.

7.5 Manejo de la estructura de datos de la extracción del nombre de los hablantes

En la sección *Stream de DOCs* se presenta el stream (grafo dirigido) con diversos nodos, entre los cuales se encuentran los nodos 7, 8, 9, 15 y 16. Estos nodos se encargan de gestionar el flujo de la estructura de datos para la extracción de los nombres de hablantes.

Se requiere una estructura de control, ya que es posible que no se identifiquen todos los nombres de hablantes en la primera iteración. Por ello, a veces es necesario realizar varias iteraciones en bucle, garantizando que en las siguientes solo se busquen aquellos nombres que aún no se han identificado. Además, se debe establecer un número máximo de iteraciones para evitar bucles infinitos, junto con otros detalles que se explicarán a continuación.

Librerías utilizadas

La implementación se ha desarrollado íntegramente con Python, utilizando únicamente funcionalidades nativas para el manejo de estructuras de datos (listas, diccionarios, conjuntos) y control de flujo.

Objetivo del flujo de nodos

El objetivo de este conjunto de nodos es identificar a los participantes de una reunión a partir de la transcripción automática y el análisis del vídeo. Para ello, se generan diálogos estructurados con tiempos y confianza media, se preparan parámetros de búsqueda de nombres para cada hablante y, finalmente, se comprueba y actualiza la información hasta que todos los hablantes estén identificados.

Implementación

El proceso comienza con el nodo *7. Preparar diálogos de transcripción*, que recibe como entrada el JSON completo devuelto por Amazon Transcribe. Esta función recorre los elementos presentes en results.items del JSON recibido en la entrada del nodo, filtrando únicamente aquellos que contienen la clave *speaker_label* para identificar el hablante asociado. A medida que avanza el recorrido, se agrupan las palabras pronunciadas por el mismo hablante en bloques continuos, construyendo un texto completo por segmento y registrando los tiempos de inicio y fin.

Un aspecto importante de esta etapa es el cálculo de un campo confidence, que representa la confianza media del segmento. Este valor se obtiene promediando las puntuaciones de confianza individuales que Amazon Transcribe asigna a cada palabra reconocida, lo que permite disponer de una medida global de fiabilidad del bloque de diálogo en lugar de manejar valores dispersos por palabra.

Finalmente, el nodo devuelve una estructura que contiene la lista de diálogos generados, incluyendo el hablante (spk_0, spk_1, etc.), el texto del diálogo, el tiempo de inicio y fin de la intervención, el nivel de confianza y el número total de hablantes detectados.

A continuación, el nodo *8. Preparar parámetros de hablantes* toma la lista de diálogos previamente generada y construye una estructura de datos que servirá como base para el proceso posterior de identificación de nombres. Para cada hablante único detectado en la transcripción, se localiza su diálogo más largo, calculando su duración y determinando el punto medio en formato de marca de tiempo (HH:MM:SS.mmm).

De cada hablante se genera un objeto speaker_json con la siguiente información:

- **num_speaker:** Índice numérico que identifica al hablante en la lista.
- **longest_dialogue:** Diccionario con el tiempo de inicio (start_time) y fin (end_time) de su diálogo más largo.
- **current_timestamp**: Marca de tiempo calculada en el punto medio de ese diálogo más largo.
- **found:** Indicador booleano inicializado en False que señala si el hablante ya ha sido identificado.
- **name:** Cadena de texto inicialmente vacía donde posteriormente se almacenará el nombre del hablante identificado.

Estos objetos se almacenan en una lista llamada result, de modo que cada elemento corresponde a un hablante distinto y contiene toda la información anterior.

Finalmente, la función devuelve un diccionario con la siguiente estructura:

```
{

    "list": result,

    "all_found": 0,

    "count": 1,

    "num_spk": "",

    "total_speakers": total_speakers_count

}
```

Código 21. Estructura de datos de la búsqueda del nombre de hablantes.

En donde cada campo representa lo siguiente:

- **list**: Es la lista result mencionada anteriormente, con la información de todos los hablantes detectados.
- **all_found:** Indicador que señala si todos los hablantes han sido identificados (0 significa que no, 1 significa que sí).
- **count:** Contador que marca el número de iteraciones realizadas para la identificación (inicia en 1).
- **num_spk:** Campo inicialmente vacío que se rellenará posteriormente con el número de hablantes identificados hasta el momento.
- **total_speakers:** Número total de hablantes detectados en la transcripción.

Esta estructura resultante es esencial para las fases siguientes, ya que permite llevar un control preciso de qué hablantes han sido identificados, cuáles no y en qué punto temporal del vídeo se puede buscar su nombre.

El nodo *9. Juntar parámetros* actúa como un enlace dentro del grafo dirigido. Este nodo está configurado para que, en su primera iteración, envíe por su salida los datos obtenidos del nodo 8. En las iteraciones posteriores, en cambio, debe devolver por su salida los datos recibidos del nodo 16.

A continuación, se lleva a cabo todo el proceso de extracción de nombres de hablantes mediante LLMs con capacidades de visión, el cual se explica más adelante. Después, se encuentra el nodo *15. Juntar parámetros*, que también actúa como enlace en el grafo dirigido. En este caso, el nodo está configurado para enviar por su salida un diccionario que contiene, por un lado, el resultado de la extracción de nombres y, por otro, los datos provenientes del nodo 9 que indican el estado de la iteración de la extracción de nombres antes de obtener los nuevos resultados. Esto permite que el siguiente nodo (nodo 16) actualice y valide dichos resultados.

Por último, el nodo *16. Comprobar resultados* gestiona la fase de actualización de datos tras un intento de detección de nombres de hablantes, asegurando la coherencia y completitud del conjunto de resultados.

En primer lugar, se extraen del JSON de entrada los resultados de detección, la lista actual de hablantes (list) y el número de iteración (count). A partir de estos datos se identifican todos los nombres disponibles que podrían asignarse, así como los nombres que ya fueron asociados a hablantes en iteraciones previas.

A continuación, la función analiza los resultados de la detección actual para identificar conflictos, entendidos como casos en los que un mismo nombre se ha asignado a más de un hablante. Estos conflictos se registran y provocan la invalidación de dichas asignaciones, eliminando el nombre y marcando el campo found del hablante como False para permitir que sea reasignado en iteraciones posteriores.

Una vez resueltos los conflictos, la función recopila las nuevas asignaciones válidas detectadas en esta iteración y actualiza la lista de hablantes (updated_list). Cada elemento de esta lista es un objeto que contiene datos como: num_speaker, longest_dialogue, current_timestamp, found y name, explicados anteriormente. Posteriormente, se calcula el número total de hablantes identificados (total_found).

Es importante tener en cuenta que el LLM encargado de extraer el nombre del hablante actual también tiene la tarea de identificar los nombres de todos los participantes en la reunión. Gracias a que el LLM proporciona esta información, si el número de hablantes identificados es igual al total de hablantes menos uno**,** es decir, si falta únicamente un hablante por identificar**,** la función asigna automáticamente el nombre restante que aún no ha sido utilizado.

Si todavía quedan al menos dos hablantes por identificar, la función incrementa el contador de iteraciones (count) y actualiza el campo current_timestamp de los hablantes no encontrados para realizar nuevas búsquedas en otros puntos de su diálogo más largo. El punto temporal usado depende de la iteración:

- Iteración 1: 50% de la duración del diálogo.
- Iteración 2: 25% de la duración del diálogo.

- Iteración 3: 75% de la duración del diálogo.

Este ciclo continúa hasta que se cumpla alguna de las siguientes condiciones:

1. Todos los hablantes han sido identificados (all_found = 1).
2. Se alcanza el límite máximo de iteraciones (más de 3), en cuyo caso el valor all_found pasa a 2 indicando finalización forzada.

Finalmente, la función devuelve un JSON con la estructura siguiente:

```
{

    "list":updated_list, #Lista actualizada de hablantes con sus datos

    "all_found": estado, #0 = continuar, 1 = todos encontrados, 2 =
finalización forzada

    "count": iteracion,  #Número de iteración actual

    "num_spk":total_found #Núm. hablantes identificados hasta ahora

}
```

Código 22. Estructura de datos devuelta al comprobar el resultado de la extracción de los nombres de los hablantes.

En conjunto, este flujo permite transformar la transcripción cruda de Amazon Transcribe en un conjunto de datos estructurados, optimizados y validados, listos para integrarse con el módulo de reconocimiento visual que asigna nombres concretos a cada participante de la reunión.

7.6 Configuración LLM

En la sección *Stream de DOCs* se presenta el stream (grafo dirigido) con diversos nodos, entre los cuales se encuentra el nodo 0, encargado de la configuración de los LLM.

Este nodo cumple una función sencilla pero fundamental: Almacenar y gestionar la información relacionada con el modelo de lenguaje que se utilizará en el procesamiento. Es importante destacar que los modelos empleados corresponden a los ofrecidos por los servicios de AWS.

En este caso, la función detrás del nodo no fue desarrollada por mí, así que principalmente explico los parámetros de esta debido a que lo tuve que configurar para mi trabajo.

Entre los datos almacenados en este nodo se incluyen:

- **ID del modelo:** Identificador único que permite seleccionar y diferenciar el modelo específico dentro de la plataforma AWS.

  En el desarrollo de este trabajo, he optado por utilizar el modelo **Claude Sonnet 4.0**, identificado en AWS como anthropic.claude-sonnet-4-20250514-v1:0. La elección de este modelo se basa en su alto rendimiento y confiabilidad, así como en sus avanzadas capacidades tanto de procesamiento textual como visual. Estas características resultan especialmente necesarias para tareas como la extracción de nombres de hablantes y el análisis de contenidos en pantallas compartidas, que son elementos clave en la funcionalidad desarrollada.

- **Región de AWS:** Especifica la ubicación geográfica del servicio donde se desplegará el modelo.

  En este caso se usa la región us-west-2 de estados unidos. No se usa la ubicación de Europa central debido a que en el proyecto DOCs existe una normativa que especifica usar siempre esta región en los LLM.

- **Máximo de tokens de salida:** Define el límite superior del número de tokens que el modelo puede generar en una sola respuesta, un parámetro crítico para controlar el rendimiento, la calidad y el coste computacional de las inferencias.

  En mi caso, he establecido un límite de 4096 tokens. Este valor se escogió tras realizar varias pruebas empíricas para determinar cuál era el más adecuado. Hay que tener en cuenta que este límite se fijó principalmente en función de la tarea encargada de generar el contenido del acta, ya que es el LLM que genera la mayor cantidad de tokens de salida. Se observó que este valor proporciona un tamaño de acta suficiente y con la información necesaria para cumplir con los requisitos del proyecto.

La correcta configuración y gestión de estos parámetros es clave para optimizar el rendimiento del sistema, adaptando el uso del modelo a las necesidades específicas del proyecto y garantizando una integración eficiente con los servicios de AWS.

7.7 Detección de cambios de escena

El nodo *18. Detección de cambios de escena* se encarga de llamar a la función de detección de cambios de escena, tal como se explicó en el apartado *7.2 Procesamiento multimedia*. En la implementación final se utilizó FFmpeg con un valor de umbral (threshold) de 0.6.

La razón por la que se eligió FFmpeg en lugar de AWS se puede explicar por los resultados obtenidos al comparar ambas tecnologías en el apartado *9. Evaluación comparativa de tecnologías y resultados de transcripción*.

Este nodo devuelve un diccionario con los segmentos donde ocurre un cambio de escena, especificando el inicio y el fin de cada segmento.

7.8 Detección de segmentos de pantalla compartida

El nodo *19. Detección de segmentos de pantalla compartida* es el que se encarga de entre los diferentes fragmentos de la reunión, clasificar aquellos que son de pantalla compartida y los que no.

Librerías utilizadas

Para esta implementación se emplean principalmente dos librerías de Python muy potentes para el procesamiento de imágenes y vídeo:

- **OpenCV (cv2):** Librería especializada en visión por computador que facilita la captura, manipulación y análisis de imágenes y vídeos. Aquí se utiliza para cargar el vídeo, extraer frames en tiempos específicos, convertir imágenes a escala de grises, y aplicar detección de rostros y detección de bordes.
- **NumPy:** Librería fundamental para el cálculo numérico en Python. Se emplea para operaciones sobre matrices e imágenes, como el cálculo del porcentaje de píxeles negros o la cantidad de bordes detectados.

Objetivo del nodo

El objetivo de este nodo es determinar, a partir de segmentos de vídeo, si durante ese tramo se está realizando compartición de pantalla o no. Esta información es crucial para analizar reuniones o grabaciones donde se alterna entre distintas modalidades, afectando la interpretación de la interacción entre participantes.

Para ello, la función analiza ciertos frames representativos del segmento, clasificándolos como “Pantalla compartida” o “Sin compartición”. Si la clasificación no es clara o es contradictoria, se amplía el análisis a más frames para llegar a una conclusión fiable mediante mayoría absoluta.

Implementación

1. Carga y procesamiento del vídeo

Se guarda el video en el almacenamiento temporal, se abre el archivo de vídeo usando OpenCV, y a partir de la marca temporal (en formato "HH:MM:SS.sss") se calcula el momento exacto en segundos para posicionarse en el frame correspondiente.

2. Extracción y análisis de frames representativos

Para cada segmento, se analizan inicialmente 3 frames situados en el 25%, 50% y 75% de la duración del segmento.

- Cada frame se extrae y convierte a escala de grises para facilitar el análisis.
- Se aplica un clasificador de rostros preentrenado (haarcascade_frontalface_default.xml) para detectar cuántos rostros aparecen en la imagen.
- Se calcula la cantidad de bordes (usando el detector Canny) para cuantificar la complejidad visual del frame.

3. Clasificación de cada frame

La regla para clasificar un frame es la siguiente:

- Se calcula un umbral basado en el número de perfiles detectados multiplicado por 4000, más un margen fijo de 4000. Este valor de 4000 se utiliza porque, según un estudio empírico realizado, representa aproximadamente la cantidad promedio de bordes que ocupa cada avatar o usuario participante en la reunión, lo que permite ajustar de forma precisa el umbral para diferenciar entre pantalla compartida y no compartida.

  Además, cabe destacar que los perfiles de los usuarios aparecen tanto cuando hay pantalla compartida como cuando no, pero en situaciones de pantalla compartida se visualizan muchos más elementos en la imagen además de los perfiles, mientras que, si no hay pantalla compartida, únicamente se observan los perfiles de los participantes.

- Si la cantidad de bordes excede este umbral, el frame se considera que muestra una "Pantalla compartida". En caso contrario, se clasifica como "Sin compartición".

4. **Decisión basada en el análisis múltiple**
    - Si los tres frames analizados inicialmente coinciden en la clasificación, esta se adopta como definitiva para el segmento.
    - En caso de discrepancias, se amplía el muestreo a cinco frames para obtener una decisión por mayoría absoluta, aumentando la robustez de la detección y minimizando falsos positivos o negativos.

Funciones clave del nodo

Este nodo llama a una función que hace uso de diversas funciones auxiliares:

- detectar_rostros(frame): Detecta y cuenta los rostros en un frame usando el clasificador Haar Cascade.
- cantidad_bordes(frame): Calcula la cantidad de bordes en un frame usando el filtro Canny.
- clasificar_segmento(frame): Aplica la lógica del umbral para determinar si un frame indica pantalla compartida o no, combinando el conteo de rostros y bordes.
- hora_a_segundos(hora_str): Convierte una cadena de tiempo en formato "HH:MM:SS.sss" a segundos en formato float, facilitando el posicionamiento en el vídeo.
- analizar_frame(video_path, time_str): Dado un path de vídeo y un tiempo, extrae el frame correspondiente y devuelve su clasificación de pantalla compartida o no.

7.9 Preparación de los parámetros de los subgrafos

Los nodos 10, 11, 12 y 13, junto con los nodos 20, 21, 22 y 23, son nodos paralelos en cuanto a funcionalidad, ya que realizan las mismas operaciones, aunque aplicadas a flujos de datos distintos. En ambos casos, su función es preparar los parámetros de entrada para el siguiente nodo: el 14 en un caso y el 24 en el otro, los cuales se encargan de invocar y ejecutar el subgrafo correspondiente.

El nodo *10. Preparar timestamp de frames a extraer*, se encarga de recorrer la estructura de datos de los hablantes explicada anteriormente. Para cada hablante cuyo campo found sea False, es decir, aquellos cuyo nombre aún no ha sido identificado, extrae el valor del campo current_timestamp y lo añade a una lista. Esta lista, que contiene todos los timestamps correspondientes a los hablantes pendientes de identificar, es la que se devuelve al finalizar la ejecución del nodo.

El nodo *20. Extraer frames de cada segmento*, procesa cada segmento detectado como segmento de compartición de pantalla. Para cada segmento, extrae tres frames uniformemente distribuidos a lo largo del mismo. En caso de que la duración del segmento supere los cinco minutos, se extraen tres frames por cada intervalo de cinco minutos para cubrir la mayor cantidad de detalles posible. Al finalizar, este nodo también devuelve una lista con los timestamps correspondientes a los frames extraídos.

Los nodos *11. Longitud lista frames* y *21. Longitud lista frames* realizan la misma función: Devolver el número de elementos de la lista que reciben como entrada, la cual corresponde a la lista de timestamps de los frames a extraer.

Finalmente, los nodos 12, 13, 22 y 23 se encargan de duplicar n veces, donde n es el valor obtenido de los nodos 11 y 21 respectivamente, el elemento recibido por su entrada y devolverlo en una lista con n posiciones.

Esta operación se realiza para que cada uno de estos elementos pueda ser distribuido a la entrada correspondiente de cada rama del subgrafo paralelo.

7.10 Implementación de los subgrafos

En el apartado de diseño se ha presentado la estructura de cada subgrafo correspondiente a los nodos *14. Subgrafo: Extracción de nombre de hablantes* y *24. Subgrafo: Extracción de información de pantalla compartida*.

En ambos casos, el flujo comienza con la extracción de un fotograma del vídeo, utilizando la función de extracción de frames explicada en el apartado *7.2 Procesamiento multimedia*.

El timestamp del fotograma a extraer y el vídeo de origen son proporcionados desde el exterior del subgrafo: en el caso del nodo 14, provienen de los nodos 10 y 13; y en el caso del nodo 24, de los nodos 20 y 23.

El siguiente paso es la extracción de los bytes del fotograma en imagen. Para este proceso no se requiere el uso de librerías externas.

El propósito principal de esta función es acceder al atributo file_bytes del objeto FileData correspondiente al fotograma, el cual contiene la representación binaria del archivo, y empaquetar dichos datos en una estructura consistente en una lista de diccionarios. Este formato de salida está diseñado para integrarse directamente con la posterior llamada al LLM, el cual espera recibir la información de la imagen en binario siguiendo este formato estándar.

A continuación, se realiza la llamada al LLM. Cabe destacar que la función encargada de invocar al modelo no ha sido desarrollada por mí; en mi caso, la tarea consistió en ajustar sus parámetros y elaborar un prompt óptimo y específico para cumplir con los requerimientos concretos de este proyecto.

Los parámetros que tuve que configurar son los siguientes:

- **llm_configuration:** Objeto que contiene toda la configuración del modelo de lenguaje. Esta configuración ya fue explicada en el apartado *7.6 Configuración LLM*.
- **system_prompt:** Texto que define el contexto o instrucciones globales para el modelo, actuando como mensaje del "sistema" que guía su comportamiento.
- **prompt:** Entrada principal de usuario o contenido que se quiere procesar. Es el texto que, junto con el system prompt, conforma la petición al modelo.
- **output_formatter_dict:** Diccionario que define cómo debe formatearse la salida del LLM. Puede incluir reglas de estructura, plantillas o validaciones.
- **text_variables:** Variables de texto que pueden sustituirse dentro del prompt. Esto permite reutilizar la plantilla del prompt con distintos valores dinámicos.
- **images:** Lista de imágenes, en formato binario, que se pasan al LLM si el modelo admite capacidades multimodales. Este es un parámetro opcional.

7.10.1 Extracción del nombre del hablante

En esta configuración del LLM, el objetivo es identificar de manera precisa el nombre del hablante activo en un momento específico de una grabación de Microsoft Teams, utilizando un fotograma extraído como entrada visual. La imagen del fotograma se proporciona al modelo mediante el parámetro **images**, que contiene su representación binaria.

El parámetro **prompt** establece la instrucción principal de la tarea: Extraer el nombre del hablante de la imagen.

Por su parte, el **system_prompt** define un conjunto exhaustivo de directrices especializadas para el análisis de la interfaz de Microsoft Teams para identificar con máxima precisión quién es el hablante activo en ese momento. Este contexto incluye la descripción de la disposición visual típica de los participantes, los indicadores visuales que señalan quién está hablando (marco azul alrededor del icono central o fondo azul en el nombre de la barra lateral), las reglas para determinar casos válidos y errores, y el formato JSON exacto en el que el modelo debe devolver la respuesta, sin añadir texto adicional.

La imagen se divide en dos zonas principales a examinar: La vista central (main view), donde se muestran iconos circulares de los participantes con sus nombres debajo, y la barra lateral derecha (sidebar), que presenta una lista vertical de participantes junto con sus nombres.

Los dos indicadores válidos y de máxima fiabilidad para identificar al hablante son: Un marco azul alrededor de un icono en la vista central, o un fondo azul detrás del nombre en la barra lateral. Ambos indicadores son igualmente válidos y basta con que uno de ellos esté presente de forma única y clara para identificar al hablante.

Se considera un caso válido (found: true) cuando hay un único indicador (marco o fondo) o cuando ambos indicadores coinciden en la misma persona, lo que se considera una doble confirmación. En este caso, se extrae el nombre exacto, manteniendo su capitalización, tildes y espacios, y se establece error_case como null.

Por el contrario, se consideran casos de error (found: false) tres situaciones: no_active_speaker cuando no se detecta ningún indicador, multiple_active_speakers cuando hay indicadores en distintas personas o más de uno sin coincidencia y unclear_visual_indicators cuando la calidad de la imagen no permite identificar los colores o los elementos son ambiguos.

El procedimiento de análisis consiste en determinar si existe vista central y/o barra lateral, buscar marcos y fondos azules para contarlos y asociarlos, aplicar las reglas de coherencia para identificar al hablante y extraer el nombre exacto. Además, se debe obtener el número total de participantes visibles (num_spk) y la lista de sus nombres (spk_names).

El formato de salida debe ser estrictamente un único objeto JSON con las claves found, speaker_name, error_case, num_spk y spk_names, sin explicaciones ni texto adicional.

La estructura **output_formatter_dict** actúa como esquema de validación de la salida generada por el LLM. Especifica los campos esperados en la respuesta, sus tipos de datos y su significado:

- found: Indica si se ha identificado exactamente un hablante activo.
- speaker_name: Nombre completo del hablante identificado o "error" si no se ha podido determinar.

- error_case: Código que describe la causa en caso de no identificar un único hablante.
- num_spk: Número total de participantes visibles en el fotograma.
- spk_names: Lista con los nombres de todos los participantes detectados.

Este diseño garantiza que el modelo no solo identifique al hablante actual, sino que también extraiga información complementaria sobre la composición de la reunión, minimizando errores y evitando conjeturas en situaciones ambiguas.

7.10.2 Extracción de información relevante de la pantalla compartida

En esta configuración del LLM, el objetivo es analizar imágenes (fotogramas) capturadas de una reunión de Microsoft Teams en la que un participante está compartiendo su pantalla. El objetivo es identificar información clave visible en el contenido compartido, como el título de la presentación, el nombre del presentador, la aplicación que se está mostrando y cualquier elemento destacado (por ejemplo, ventanas activas, barra de herramientas, o indicadores de actividad).

La entrada al LLM incluye el parámetro **images**, que contiene el frame correspondiente al momento exacto de la reunión. El análisis debe centrarse únicamente en la parte del contenido compartido, ignorando elementos irrelevantes como los iconos de participantes o la barra lateral, salvo que estos contengan datos directamente relacionados con la presentación o el presentador.

El parámetro **prompt** establece la instrucción principal de la tarea: Extraer información de la pantalla compartida en una reunión de Microsoft Teams de la imagen enviada al modelo.

El **system_prompt** es más directo, claro y sencillo que en el caso anterior. Indica al modelo que debe centrarse en la información visible en la pantalla compartida durante una reunión de Teams. Solicita datos clave como el nombre del presentador, el título, la aplicación o el contenido compartido, y que preste especial atención al contenido textual. También enfatiza que la salida debe ser únicamente en formato JSON, sin explicaciones ni texto adicional.

El **output_formatter_dict** solo contiene un campo llamado *description*, que será un texto libre donde el modelo explica todo lo que ha identificado en el frame. Es un formato sencillo, útil cuando se quiere una salida narrada en lugar de datos estructurados.

7.11 Extraer nombres resultantes

El nodo *17. Extraer nombres resultantes*, se encarga de estructurar correctamente los nombres de los hablantes que han sido identificados, una vez finalizado todo el proceso de extracción.

Librerías utilizadas

La función está implementada en Python y no requiere librerías externas adicionales, apoyándose únicamente en las funcionalidades básicas del lenguaje para manipulación de diccionarios y listas.

Objetivo del nodo

El propósito principal del nodo es procesar la información de los hablantes obtenida tras una etapa previa de detección y reconocimiento, para estructurar un resumen claro y accesible. Esto incluye la creación de un mapeo entre identificadores internos de hablantes y

sus nombres reconocidos, además de generar un resumen textual que facilita la comprensión y posible uso por un modelo de lenguaje o por interfaces de usuario.

Implementación

La función recibe como parámetro un diccionario *speaker_data* que contiene una lista de hablantes con su información asociada. El proceso se desglosa en los siguientes pasos:

1. Extracción de datos iniciales

Se obtiene la lista de hablantes y se determina el número total de hablantes. Si no se provee explícitamente el número total bajo la clave 'num_spk', se calcula a partir del tamaño de la lista.

2. Creación de mapeo de hablantes

Se inicializan dos listas para registrar hablantes identificados y no identificados, así como un diccionario que almacenará el mapeo entre el identificador interno (por ejemplo, "spk_0") y el nombre reconocido o la etiqueta "not_known".

3. Recorrido y clasificación de cada hablante

Para cada elemento en la lista, se extrae la información relevante: Número de hablante (num_speaker), estado de identificación (found) y nombre (name).

- Si el hablante está identificado y tiene un nombre válido distinto de "unknown", se añade al mapeo con su nombre real.
- En caso contrario, se marca como "not_known".

4. Generación del resumen descriptivo

Se construye un texto resumen que indica el total de hablantes detectados, los hablantes identificados y los que aún no se han identificado. Este resumen es útil para alimentar modelos de lenguaje natural.

```
    summary_parts = []

    summary_parts.append(f"Total speakers detected: {total_speakers}")

    if identified_speakers:

        summary_parts.append(f"Identified          speakers:          {',
'.join(identified_speakers)}")

    if unknown_speakers:

        summary_parts.append(f"Unknown            speakers:          {',
'.join(unknown_speakers)}")
```

Código 23. Resumen de hablantes identificados y no identificados.

5. Estructuración del resultado final

Finalmente, se devuelve un diccionario estructurado que incluye:

- El total de hablantes detectados.
- El mapeo entre identificadores y nombres.
- El resumen textual.

- Un estado de identificación con conteos y un indicador booleano que señala si todos los hablantes fueron identificados.

7.12 Corrección de la transcripción y generación del acta de reunión

El nodo *26. Corrección transcripción + Generar acta*, se encarga de utilizar un LLM para corregir posibles errores gramaticales, de puntuación y fonéticos en la transcripción, así como para generar el contenido del acta final.

Este LLM está diseñado para analizar y procesar la transcripción de una reunión, junto con información adicional relevante, como la descripción de la pantalla compartida y detalles sobre los nombres de los participantes. Su tarea principal se divide en dos fases claramente definidas que deben realizarse en orden.

Primero, en TAREA_1, el modelo analiza la transcripción para corregir errores. Esto implica identificar el idioma de la reunión, detectar errores ortográficos y de puntuación, así como corregir posibles palabras mal captadas o transcritas. El objetivo es producir una versión corregida y más precisa del texto original, que servirá como base para la siguiente tarea.

Luego, en TAREA_2, el LLM utiliza esta transcripción corregida para generar un acta de reunión bien estructurada y profesional. El acta debe incluir información clave como la fecha, participantes, propósito de la reunión, un resumen de los temas tratados organizados por puntos de agenda, acuerdos alcanzados, y las acciones a realizar con responsables y fechas límite. Además, debe listar las decisiones importantes con contexto, identificar temas pendientes para futuras reuniones y extraer la fecha y hora de la próxima reunión si esta se menciona.

El modelo puede complementar el acta con tablas, diagramas o esquemas en formato markdown con HTML/SVG cuando sea apropiado y tenga suficiente información. También está autorizado a incluir información relevante adicional que no esté explícitamente en la lista, siempre y cuando esté justificada por la transcripción.

Es importante que el LLM evite inventar información o agregar datos no mencionados en la transcripción. Si algún dato solicitado no está presente, debe indicarlo explícitamente o excluirlo del acta.

Finalmente, el resultado esperado es un único campo JSON llamado acta, que contiene el acta completa en formato markdown, con encabezados, listas y tablas, detallando toda la información estructurada y resumida de la reunión.

Por lo tanto, los parámetros que se le pasan en este caso a la función son los siguientes:

- El objeto con la **configuración del LLM** explicada en el apartado *7.6 Configuración LLM*.
- **system_prompt:** Texto que define las instrucciones para la TAREA_1, encargada de la corrección de la transcripción, y la TAREA_2, responsable de generar el contenido del acta con la posibilidad de incluir gráficos, tablas y diagramas. Es decir, se detallan las instrucciones específicas del trabajo que realiza el nodo descrito anteriormente.
- **prompt:** Explica de forma sencilla que debe corregir la transcripción recibida como parámetro y generar el contenido del acta basándose en dicha transcripción corregida, en los nombres de los hablantes y en la información

relevante extraída de la compartición de pantalla durante la reunión que recibe por parámetro.

- **output_formatter_dict:** Diccionario que especifica que la salida es un único campo JSON llamado acta, que contiene el acta completa en formato markdown.
- **text_variables:** Diccionario que contiene las variables que se pasan al LLM, es decir, la transcripción, los nombres de los hablantes y la descripción de las pantallas compartidas durante la reunión.
- **images:** En este caso, no se utiliza este parámetro, ya que no es necesario y su uso es opcional.

7.13 Ground truth validator

El nodo *27. Ground Truth Validator*, se encarga de validar qué tan correcta, cierta o precisa es la respuesta del LLM que genera el contenido del acta de la reunión en comparación con los datos de entrada, que en este caso incluyen la transcripción, los nombres de los hablantes y la descripción de las pantallas compartidas durante la reunión.

Esta funcionalidad no ha sido desarrollada por mí, pero la he integrado en el flujo de mi grafo dirigido. Su objetivo es validar información comparándola con un texto de referencia ("*ground truth*") y calcular un porcentaje de exactitud junto con una evaluación descriptiva.

Librerías utilizadas

- **concurrent.futures:** Para tareas concurrentes.
- **json y textwrap:** Manejo de datos y formato de cadenas.
- **pydantic:** Definición y validación de esquemas de datos.
- **agentic_core:** Librerías específicas para gestión de agentes LLM, herramientas y utilidades.

Objetivo del nodo

El objetivo es comprobar la exactitud de la información evaluada respecto a un texto de referencia y devolver:

1. Un porcentaje de corrección.
2. Una evaluación con sugerencias de mejora.

Implementación

El proceso consiste en:

1. Establecer una configuración del modelo LLM y conectarlo al proveedor.
2. Iniciar un agente especializado (*GroundTruthValidatorAgent*) con un prompt y un esquema de salida definidos.
3. Ejecutar la validación enviando el texto de referencia y la información a evaluar.
4. Procesar la respuesta del LLM para extraer el porcentaje de corrección y la evaluación.
5. Desconectar el proveedor y devolver el resultado junto con métricas de ejecución.

7.14 Markdown a HTML

Una vez validado que el contenido del acta generado es suficientemente confiable y preciso, el nodo *28. Mrk to HTML* se encarga de convertirlo del formato Markdown a HTML para darle el formato final deseado, con un tratamiento especial para los diagramas SVG incluidos en el texto.

Librerías utilizadas

- **markdown:** Utilizada para realizar la conversión del contenido en formato Markdown a HTML, permitiendo el uso de tablas y bloques de código.
- **re:** Empleada para definir patrones mediante expresiones regulares y localizar bloques de código que contienen SVG.
- **base64:** Utilizada para codificar el contenido SVG en base64, posibilitando su inclusión como imagen embebida en el HTML.
- **html:** Usada para decodificar entidades HTML y limpiar el contenido de caracteres escapados.

Objetivo del nodo

El objetivo de esta función es generar un documento HTML final que integre tanto el texto en Markdown como los diagramas SVG de forma visual, evitando que se muestren como texto plano y garantizando una presentación clara y profesional en las actas generadas automáticamente.

Implementación

1. **Detección de bloques SVG:** Mediante expresiones regulares, se identifican los bloques de código que contienen diagramas SVG.
2. **Procesamiento y limpieza:** El contenido SVG se limpia de caracteres escapados y entidades HTML, asegurando que posea la estructura correcta (<svg> y </svg>).
3. **Codificación en base64:** El SVG validado se convierte en una cadena base64 para poder ser incrustado directamente en un elemento <img> dentro del HTML.
4. **Sustitución en el Markdown:** El bloque original de código SVG se reemplaza por un bloque HTML con estilos predefinidos que contiene el <img> generado.
5. **Conversión final a HTML:** El Markdown ya procesado se convierte a HTML usando la librería *markdown* con extensiones para tablas y bloques de código.
6. **Postprocesado de SVG escapados:** Si después de la conversión quedan diagramas SVG en formato escapado, se detectan y se procesan de la misma forma, asegurando su visualización correcta.
7. **Generación de la plantilla final:** El HTML resultante se inserta en una plantilla con estilos CSS, manteniendo la coherencia visual del documento y optimizándolo para impresión o exportación a PDF.

   En este paso, se añade al final del acta un bloque con fondo oscuro y letra clara que contiene el texto “Documento generado automáticamente”. Esto es importante para concienciar a los usuarios de que el acta fue creada de manera automática, de modo que sean conscientes de que puede contener errores y que

debe ser revisada, además de mantener la transparencia. Así, los usuarios asumen parte de la responsabilidad sobre su validación.

Esta función permite que las actas incluyan diagramas y gráficos de forma integrada y visualmente atractiva, mejorando su legibilidad y utilidad.

7.15 HTML a PDF

El nodo *29. HTML to PDF*, es el encargado de convertir el contenido HTML, obtenido y formateado en el nodo 28, a un archivo PDF. Este proceso permite formalizar el acta de reunión en un formato profesional, fácil de compartir e imprimir, manteniendo la estructura y el estilo definidos previamente.

Librerías utilizadas

- **Aspose.PDF (ap):** Librería principal encargada de la conversión del contenido HTML a PDF y de la manipulación del documento resultante, como la adición de pie de página.
- **tempfile:** Permite la creación de archivos temporales para almacenar el HTML antes de convertirlo.
- **datetime:** Se emplea para generar nombres de archivo con marcas de tiempo en caso de no proporcionarse uno.
- **os:** Utilizada para la eliminación de los archivos temporales creados durante el proceso.
- **io.BytesIO:** Gestiona el contenido binario del PDF en memoria sin necesidad de guardarlo en disco.

Objetivo del nodo

El nodo tiene como finalidad transformar un documento HTML completo en un archivo PDF listo para ser distribuido o almacenado, incluyendo de forma opcional un pie de página personalizado. Esto permite generar documentos formateados, como actas de reunión, directamente desde plantillas HTML.

Implementación

1. **Validación de entrada:** Se comprueba que el HTML recibido no esté vacío.
2. **Generación del PDF base**
    - Se crea un archivo temporal con el HTML.
    - Se configuran los márgenes y opciones de carga usando *HtmlLoadOptions* de *Aspose.PDF*.
    - Se genera el documento PDF en memoria mediante *BytesIO*.
3. **Adición opcional de pie de página**
    - Si se especifica texto para el footer, que se puede especificar mediante un parámetro proporcionado por el usuario, se utiliza *TextStamp* de *Aspose.PDF* para añadirlo centrado en la parte inferior de cada página.
    - Se configura el tamaño de fuente, color y márgenes para garantizar una correcta legibilidad.

- En las actas generadas en este trabajo se añade un pie de página con el siguiente texto: © YEAR Applus+ IDIADA - Creado con DOCs, donde YEAR se sustituye por el año actual.

4. **Asignación de nombre de archivo**
    - Si no se proporciona un nombre, se genera uno automáticamente con el patrón Meeting_minutes_YYYYMMDD_HHMMSS.pdf, donde YYYYMMDD_HHMMSS es la fecha actual.
    - Se asegura que la extensión sea .pdf.
5. **Generación de la respuesta final**
    - Se retorna un diccionario con un objeto FileData que contiene el contenido binario del PDF, el nombre de archivo y metadatos.
    - Se incluyen mensajes de log y métricas para trazabilidad.

Esta implementación asegura la creación de PDFs consistentes y profesionales a partir de HTML, con la flexibilidad de incluir pies de página y gestionar el proceso completamente en memoria para optimizar el rendimiento.

7.16 Gestión de correos electrónicos

Los nodos *31. Conexión al email* y *32. Enviar email*, tal como indica su nombre, se encargan de gestionar la conexión y el envío del correo electrónico, utilizando en este caso Outlook.

Librerías utilizadas

- **smtplib:** Para la conexión a servidores de correo SMTP y envío de mensajes.
- **email y submódulos** (MIMEText, MIMEMultipart, MIMEApplication, MIMEImage, MIMEBase, encoders): Para la construcción y codificación de correos electrónicos en formato MIME, incluyendo cuerpo, cabeceras y adjuntos.
- **hashlib y time:** Para generar identificadores únicos de mensajes y nombres de archivos.
- **datetime:** Para el manejo y formateo de fechas.
- **FileData:** Esquema utilizado para encapsular información de ficheros adjuntos (bytes, nombre y tipo MIME).

7.16.1 Establecimiento de conexión al correo electrónico

Objetivo del nodo

El nodo *31. Conexión al email* es el encargado de crear una instancia de la clase EmailConnector, la cual centraliza la gestión de las conexiones IMAP y SMTP.

Implementación

1. **Parámetros de conexión**

Se recibe el usuario, contraseña, y opcionalmente los servidores IMAP y SMTP. Si estos últimos no se proporcionan, se determinan automáticamente según el dominio del correo

mediante métodos internos (get_imap_server y get_smtp_server), que contienen una tabla de correspondencia para dominios comunes.

2. **Instanciación de EmailConnector**

La clase almacena credenciales, define puertos estándar (IMAP: 993, SMTP: 587) y prepara la conexión.

3. **Gestión de errores y trazabilidad**

Se manejan excepciones para capturar fallos de conexión y se registran mensajes informativos (logs) para auditoría y depuración.

7.16.2 Envío de correos electrónicos con adjuntos

Objetivo del nodo

El nodo *32. Enviar email* permite enviar un correo a través de un servidor SMTP autenticado, incluyendo uno o varios ficheros adjuntos.

Implementación

1. **Construcción del mensaje MIME**
   - Se crea un objeto *MIMEMultipart* para contener todos los elementos del correo.
   - Se asignan cabeceras básicas (From, To, Subject).
   - Se adjunta el cuerpo en texto plano o HTML, según el valor de is_html (True o False).
2. **Gestión de adjuntos**
   - Cada elemento en la lista attachments_data es un objeto FileData que contiene bytes, nombre de archivo y tipo MIME.
   - El método interno *_add_filedata_attachment()* clasifica el adjunto según su tipo (application/pdf, image/*, application/zip, etc.) y lo encapsula usando la clase MIME correspondiente (MIMEApplication, MIMEImage, MIMEBase, etc.).
   - Se añaden cabeceras de disposición para indicar el nombre y tipo del archivo.
3. **Conexión y envío**
   - Se establece una conexión segura al servidor SMTP (*starttls()*).
   - Se autentica el usuario y se envía el mensaje completo con *send_message()*.
   - Se cierra la conexión.
4. **Control de errores y logging**
   - Se capturan posibles fallos de envío (por ejemplo, credenciales incorrectas, adjuntos corruptos o desconexiones).
   - Se registran mensajes de confirmación o error para su trazabilidad.

7.16.3 Mi configuración

En mi caso concreto, la conexión SMTP se ha configurado con los siguientes parámetros:

- Servidor SMTP: smtp.office365.com
- Credenciales: Dirección de correo corporativa y contraseña asociada.

  En el desarrollo del front-end, se deberá implementar un mecanismo para que la contraseña se envíe de forma cifrada o protegida, asegurando que no se transmita en texto plano y garantizando la seguridad de los datos del usuario.

Al enviar el correo, se incluyen los siguientes ajustes específicos:

- Adjunto: Un archivo PDF que contiene el acta generada automáticamente.
- Asunto: "Automated Meeting Report".
- Destinatario: Dirección de correo del usuario que solicita el acta.
- Formato: is_html = true, enviando un cuerpo HTML con diseño personalizado.

El cuerpo HTML incluye:

- Encabezado con el título "Acta Automática Generada" y el logotipo visualmente destacado.
- Mensaje de agradecimiento y explicación del contenido adjunto.
- Enlace directo al asistente de IA AIDA, que permite consultar de forma interactiva información sobre la reunión.
- Sección de cierre con la firma "Equipo de DOCs" y un pie de correo indicando que es un correo automático.

Este formato garantiza que el usuario reciba no solo el documento en PDF, sino también una experiencia visual clara y profesional, con acceso inmediato a herramientas adicionales para el análisis de la reunión.

7.16.4 Resumen de flujo

- *31. Conexión al email*: Configura y crea un objeto EmailConnector con acceso SMTP seguro.
- *32. Enviar email*: Usa esa conexión SMTP para construir un mensaje MIME, añadir el cuerpo y adjuntos, y enviarlo al destinatario.

Este diseño modular separa la gestión de la conexión del proceso de envío, lo que mejora la reutilización del código, facilita la depuración y permite que la lógica de adjuntar archivos sea escalable para soportar nuevos tipos de documentos en el futuro.

7.17 Gestión de la descarga de grabaciones de reuniones desde SharePoint

Los nodos *1. Conexión SharePoint* y *2. Ingesta de grabación de reunión* son los encargados de gestionar la conexión a SharePoint y descargar desde allí los videos de las reuniones para su posterior procesamiento.

Librerías utilizadas

- **msal:** Permite autenticar aplicaciones contra Azure Active Directory (Azure AD) y obtener tokens de acceso para interactuar con Microsoft Graph API.
- **requests:** Se utiliza para realizar peticiones HTTP a la API de Microsoft Graph y descargar archivos.

- **io.BytesIO:** Facilita la manipulación de los archivos descargados en memoria como un flujo de bytes.
- **mimetypes:** Se utiliza para determinar el tipo de contenido de cada archivo según su extensión.
- **asizeof (en métricas):** Permite calcular el tamaño en memoria de los datos descargados.
- **typing:** Para la definición de tipos y anotaciones (List, Dict, Optional) que ayudan a mantener el código seguro y legible.

Objetivo del flujo de nodos

1. Conectar de manera segura a SharePoint usando una aplicación registrada en Azure AD, evitando la necesidad de autenticación individual de cada usuario.
2. Centralizar el acceso a archivos, en particular grabaciones de reuniones de Microsoft Teams, que se almacenan en un directorio organizado por departamento y proyecto.
3. Descargar archivos de manera programática para su posterior procesamiento automatizado.
4. Garantizar control de permisos a través de metadatos del archivo, verificando si un usuario tiene autorización para acceder a una grabación concreta.

7.17.1 Conexión a SharePoint

Implementación

El nodo *1. Conexión SharePoint* establece una sesión segura con SharePoint mediante la clase SharePointConnector. Esta clase se encarga de:

1. **Recepción de credenciales:** La función recibe las credenciales de la aplicación registrada en Azure AD (client_id, client_secret, tenant_id) y los sitios de SharePoint a los que se desea acceder.
2. **Obtención de token de acceso:** Se utiliza MSAL (Microsoft Authentication Library) para autenticar la aplicación mediante *acquire_token_for_client*, generando un token de acceso válido para Microsoft Graph API.
3. **Acceso programático a directorio centralizado:** Esto permite mantener una conexión segura a SharePoint con acceso a un directorio centralizado de grabaciones, evitando que cada usuario tenga que autenticarse individualmente.
4. **Devolución de objeto de conexión:** La función retorna un objeto de conexión que encapsula el token y la información de los sitios, listo para ser utilizado por otras funciones de descarga de archivos.

7.17.2 Descarga de archivos desde SharePoint

Implementación

El nodo *2. Ingesta de grabación de reunión* permite descargar un archivo específico.

1. Recibe como entrada el objeto SharePointConnection generado previamente, junto con el id de reunión, departamento y proyecto del archivo a descargar.
2. Llama al método get_file_bytes de SharepointConnector que realiza:

a. Obtención de metadatos del archivo mediante Microsoft Graph API (/drives/{library_id}/items/{document_id}) para extraer el nombre y verificar permisos de acceso.
b. Descarga del contenido en memoria en forma de BytesIO.
c. Identificación del tipo de contenido usando mimetypes para clasificar el archivo correctamente.

3. Devuelve un objeto FileData con el contenido del archivo y la información asociada.

7.17.3 Problemas de autentificación y permisos

Tal y como se ha explicado en el diseño, se ha decidido centralizar el almacenamiento de grabaciones de reuniones en SharePoint para facilitar el tratamiento y más que nada por problemas de permisos y confidencialidad de la empresa.

A la empresa le cuesta dar permiso tan abiertos y generales como el poder de acceder a SharePoint de un usuario para descargar de allí contenido razón por la cual se tuvo que tomar este enfoque.

Por eso, algunas de las medidas tomadas para mitigar problemas de autenticación y permisos son:

- Se centraliza el almacenamiento de las grabaciones de Teams en un único directorio organizado por departamento y proyecto.
- Una aplicación registrada en Azure AD proporciona acceso programático a este directorio.
- En lugar de autenticar individualmente a cada usuario, la aplicación utiliza Microsoft Graph API para acceder a todas las grabaciones de forma centralizada.
- Los permisos de acceso se determinan a partir de los metadatos del archivo, que incluyen información del usuario que organizó la reunión, permitiendo verificar si un usuario puede acceder a un video específico sin necesidad de autenticación manual.
- La aplicación descarga automáticamente el video correspondiente.

7.18 Crear sesión de AIDA

El nodo *33. Obtener TOKEN*, tal como indica su nombre, se encarga de obtener un token de validación, que verifica al usuario, mientras que el nodo *34. Creación de sesión de AIDA* se encarga de crear una nueva sesión en el asistente inteligente AIDA con el contexto de la transcripción de la reunión que se está tratando.

Librerías utilizadas

- **json:** Para manipular y convertir estructuras de datos JSON a TXT en este caso.
- **requests:** Para realizar llamadas HTTP síncronas al API REST de AIDA (crear sesión, fijar título, vincular archivos).
- **websockets:** Para gestionar comunicación asíncrona con el endpoint de subida de archivos a S3 mediante WebSocket.

- **nest_asyncio y asyncio:** Permiten ejecutar código asíncrono dentro de entornos que no soportan bucles de eventos múltiples.

Objetivo del flujo de nodos

El objetivo de esta función es crear una nueva sesión en el asistente inteligente de la empresa, AIDA y vincular un archivo de transcripción como contexto de dicha sesión. Esto permite que la sesión recién creada tenga acceso directo a la información de la reunión o conversación transcrita, incluyendo los nombres de los hablantes y la transcripción completa.

Implementación

1. **Procesamiento de la transcripción:**
   a. Se recibe la trascripción de tamaño reducido que se explicó en el apartado *7.4 Reducir tamaño de transcripción*, que ayuda a reducir el tamaño de los datos.
   b. Se transforma la transcripción JSON a un formato TXT, incluyendo:
      i. Un encabezado con los nombres de los hablantes, que se pasan como parámetros.
      ii. La transcripción completa.
      iii. Los segmentos de audio organizados por speaker y tiempo y con la confianza de cada uno.
2. **Obtención del token de autorización:** El nodo *33. Obtener TOKEN* se encarga de obtener el token mediante una llamada a un endpoint interno que devuelve un token válido para el empleado que utiliza la aplicación. Este token se utiliza para autenticar todas las llamadas subsecuentes a AIDA y validar al usuario.
3. **Gestión segura y controlada del acceso de los usuarios:**
   a. Obtención de información del usuario con una llamada GET al middle de AIDA y a partir de la respuesta, el sistema extrae principalmente el rol del usuario para determinar los permisos y capacidades dentro de la aplicación.
   b. Comprobación de interacciones mensuales con una llamada POST al middle de AIDA, para controlar el uso del sistema y respetar los límites de interacciones.
   c. En conjunto, estas llamadas permiten que el sistema gestione de manera segura y controlada el acceso de los usuarios, verificando sus permisos y asegurando que no se excedan las restricciones de interacción mensual.
4. **Creación de sesión y configuración:**
   a. Se hace una llamada POST al endpoint /create_session de AIDA con el token de autorización.
   b. Una vez creada la sesión, se modifica el título de la sesión mediante el endpoint /set_title.

5. **Subida de archivo y vinculación a la sesión:**
   a. Se establece una conexión WebSocket con el endpoint de AIDA que gestiona la subida de archivos a S3, utilizando el token de autorización.
   b. Se envía un mensaje estructurado solicitando subir el archivo (transcription.txt).
   c. Se sube el archivo de transcripción a la URL pre-firmada proporcionada por AIDA.
   d. Se envía un mensaje de confirmación de subida por WebSocket.
   e. Finalmente, se realiza una llamada POST al endpoint /v1/files/link, indicando el id del fichero y de la sesión creada que se han obtenido de las llamadas realizadas anteriormente, para vincular el archivo subido a la sesión recién creada, asegurando que el archivo sirve como contexto de la sesión.
6. **Manejo de errores y validación:**
   a. Se incluyen verificaciones del estado HTTP en cada llamada para asegurar que la sesión se crea correctamente, el título se fija y el archivo se vincula.
   b. Se controlan excepciones específicas de WebSocket para identificar problemas de autorización o conexión.
7. **Resultado:**
   a. Al finalizar la ejecución, la función devuelve un estado "ok" si todo el proceso se completó correctamente, permitiendo que la sesión en AIDA tenga inmediatamente acceso a la transcripción como contexto, optimizando la posterior interacción del asistente inteligente con los datos de la reunión.
   b. En caso de que haya habido error, se devuelve el estado “bad” y una explicación del error.

A continuación, se muestra una figura con cómo se gestionan los archivos en el asistente inteligente AIDA:

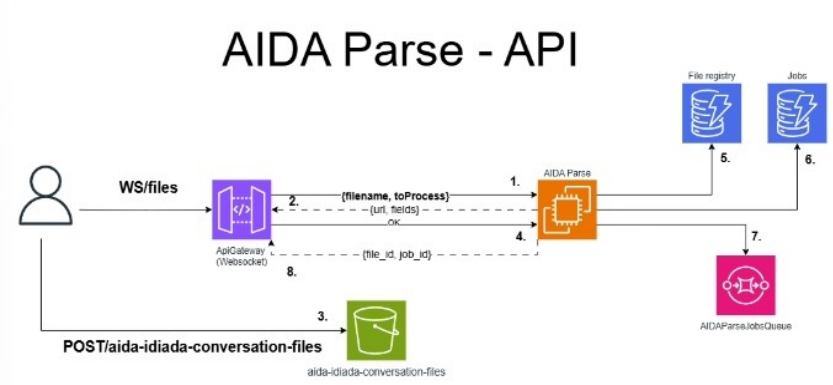


Figura 25. Gestión de archivos por AIDA

7.19 Errores identificados en el engine de DOCs

Durante el desarrollo de este proyecto, he identificado dos errores de casos que no habían sido contemplados, en los cuales el engine encargado de ejecutar el grafo provocaba que la ejecución de los nodos entrara en un bucle infinito hasta alcanzar el máximo número de intentos, deteniéndose finalmente con un error y quedando estancado sin poder completar la ejecución.

Por ello, tuve que investigar y determinar la causa de estos errores para poder continuar con el trabajo. A continuación, se explican ambos errores y cómo se han resuelto.

Cabe destacar que estos errores se encontraron durante el desarrollo del grafo en sí, por lo que los IDs de los nodos que se presentan en el apartado de diseño difieren de las versiones intermedias previas. Por esta razón, se hará referencia a los nodos utilizando letras en lugar de números.

7.19.1 Primer error: Optimización del algoritmo de selección de nodos para evitar deadlocks

El primer error se detectó durante la ejecución de un grafo compuesto por n nodos. En este escenario, el último nodo en ejecutarse era el “e” y, por la lógica de dependencias, se esperaba que el siguiente nodo fuera el “f”. Esto se debía a que los demás nodos todavía no tenían todas sus dependencias completadas, lo que imposibilitaba su ejecución.

Al analizar el comportamiento del engine, se observó que, en lugar de ejecutar “f”, se intentaba ejecutar el nodo “k”. Cada intento fallido generaba un bucle en el que el engine trataba de ejecutar “k”, pero no podía porque le faltaban dependencias críticas.

Cuando el engine entra en deadlock, activa una función diseñada para resolverlo seleccionando un nodo que esté listo para ejecutarse. Sin embargo, esta función presentaba un error debido a que no contemplaba un caso particular de deadlock.

La función de resolución de deadlocks evaluaba los nodos mediante un parámetro denominado *score*, que representaba el grado de preparación de un nodo basado en la proporción de dependencias satisfechas. En este cálculo, el nodo “k” obtenía un score más alto porque tenía 2 de 3 dependencias parcialmente listas, mientras que el nodo “f”, aunque completamente preparado para ejecutarse, no alcanzaba un score superior.

El motivo por el que “f” no era considerado listo es que uno de sus dos parámetros de entrada era opcional y no había sido proporcionado. Aunque el nodo podía ejecutarse sin ese parámetro, la función de cálculo de score no contemplaba que los parámetros opcionales no comprometían la ejecución. Como consecuencia, el score de “f” era 0.5, mientras que el de “k” era 0.6, lo que hacía que el engine eligiera incorrectamente “k” y quedara atrapado en el bucle.

Para solucionar este problema, se modificó la función de resolución de deadlocks de la siguiente manera:

- Se implementó una verificación detallada de la preparación de los nodos, evaluando no solo el número de dependencias satisfechas sino también la obligatoriedad de cada parámetro de entrada.
- Los nodos con entradas faltantes pero opcionales se consideran válidos para su ejecución.

- Se mantiene la prioridad para los nodos que tienen todas sus entradas completamente listas, asegurando un orden lógico de ejecución.
- Con esta mejora, se evita que el engine entre en deadlock debido a una evaluación incorrecta del score, garantizando que los nodos realmente ejecutables sean seleccionados primero.

Esta corrección asegura que la ejecución del grafo siga un flujo consistente y evita bucles infinitos causados por cálculos de preparación incompletos o incorrectos.

7.19.2 Segundo error: Prioridades negativas en la planificación de nodos

Durante la ejecución de un grafo, se identificó un segundo error que provocaba que el engine entrara en un estado de deadlock. El sistema asigna a cada nodo una prioridad numérica comprendida entre 0 y 100, donde 0 representa la máxima prioridad de ejecución.

Durante la depuración del estado de error, se observó que uno de los nodos presentaba una prioridad negativa de -16.66, algo que, por diseño, debería ser imposible. Esta anomalía causaba que el engine intentara ejecutar repetidamente este nodo, el cual no estaba preparado debido a la falta de dependencias, ignorando así otros nodos que sí cumplían las condiciones para ejecutarse.

Cuando el sistema detecta un deadlock, invoca una función de resolución que, en condiciones normales, asigna una prioridad de -1 al nodo listo para ejecutarse, asegurando que se ejecute antes que cualquier otro (dado que no debería existir ninguna prioridad menor a -1). Sin embargo, la presencia de un nodo con prioridad -16.66 rompía esta lógica, ya que el engine lo seleccionaba siempre por tener el valor más bajo.

La causa raíz se identificó revisando y depurando el código: en un punto del algoritmo, se restaban 50 unidades a la prioridad de ciertos nodos con el objetivo de darles preferencia frente a nodos que ya habían sido ejecutados previamente (previniendo posibles bucles de reejecución). Sin embargo, este ajuste no contemplaba el caso en que dicha resta provocara que la prioridad descendiera por debajo de cero.

Para resolver el problema, se introdujo una validación que verifica, antes de restar 50 puntos, si el resultado sería menor que 0. En caso afirmativo, se asigna directamente una prioridad de 0 para evitar valores negativos. En caso contrario, se aplica la resta de 50 normalmente.

Con esta modificación, se garantiza que las prioridades se mantengan dentro de los límites permitidos, evitando así que el engine seleccione nodos no preparados y resolviendo el problema de deadlock en este escenario.

8 Evaluación

Para validar el correcto funcionamiento del sistema, se han diseñado y ejecutado casos de prueba que cubren todos los requisitos funcionales y no funcionales definidos. El objetivo es verificar que cada componente cumple con la funcionalidad esperada, así como garantizar el cumplimiento de criterios de seguridad, rendimiento, escalabilidad y mantenibilidad.

8.1 Diseño de casos de prueba

Se ha seguido una estrategia de pruebas por requisitos, de manera que para cada requisito se ha definido:

- ID de requisito: Correspondiente a la lista de requisitos funcionales y no funcionales.
- Descripción del caso de prueba: Acción o secuencia de acciones necesarias para verificar el requisito.
- Datos de entrada: Parámetros, credenciales, archivos o configuraciones necesarias para ejecutar la prueba.
- Resultado esperado: Comportamiento esperado del sistema.
- Resultado obtenido: Comportamiento real observado durante la ejecución de la prueba.
- Estado: Superada / No superada / Pendiente.

Se ha diseñado una serie de pruebas para asegurar que los requisitos establecidos al inicio se hayan cumplido. Además de estas, se han realizado pruebas específicas y precisas para cada funcionalidad desarrollada.

A cada prueba se le ha asignado un identificador numérico, una descripción del caso de prueba, los datos de entrada, el resultado esperado y el obtenido, así como el estado final.

La diferencia principal radica en que estas pruebas no se han asociado directamente a un requisito concreto, ya que las pruebas orientadas exclusivamente a verificar la lista de requisitos se han realizado por separado. De este modo, estas otras pruebas se enfocan en validar casos de funcionalidad y aspectos internos del desarrollo.

8.2 Relación de pruebas realizadas por requisito funcional

1. Conexión a SharePoint: Prueba de establecimiento de conexión con credenciales de la aplicación Azure AD correctas e incorrectas, verificando mensajes de error y logs.
2. Localización de grabación: Selección manual de ruta y validación de existencia del archivo.
3. Descarga de grabación: Descarga exitosa y validación de integridad del archivo.
4. Notificación de errores de conexión o permisos: Simulación de credenciales de la aplicación Azure AD inválidas y permisos insuficientes.
5. Separación de audio del video: Extracción y validación del archivo de audio resultante.
6. Reducción de ruido: Comparación de la calidad del audio antes y después del proceso. Si el ruido no puede ser detectado por el oído humano, se pueden emplear

herramientas de procesamiento multimedia, como el propio FFmpeg utilizado en el trabajo, para validar si la operación fue exitosa o no.

7. Normalización de audio: Verificación de niveles de volumen uniformes. Igual que en el caso anterior, se puede usar FFmpeg para validar si la operación fue exitosa.
8. Extracción de frames: Obtención de una imagen exacta en un timestamp específico. Se puede verificar que es correcto revisando personalmente el frame extraído y el video del cual fue extraído.
9. Detección de cambios de escena: Identificación correcta de los segmentos, lo cual se puede verificar revisando personalmente el video y comprobando que los segmentos resultantes coincidan con la realidad.
10. Configuración de parámetros de procesamiento multimedia: Verificación de la aplicación de parámetros personalizados.
11. Transcripción y diarización con Amazon Transcribe: Se realiza correctamente la transcripción y la segmentación por hablantes.
12. Vocabulario personalizado en transcripción: Inclusión correcta de términos definidos.
13. Extracción de nombres de speakers: Asociación correcta de nombres personales a los participantes en la reunión.
14. Corrección de transcripción con LLM: Verificación de mejoras gramaticales y semánticas.
15. Generación de acta con LLM: Coherencia y estructura del contenido.
16. Verificación del acta con LLM adicional: Consistencia y reducción de errores si se vuelve a generar el acta de reunión a partir del análisis obtenido por este LLM.
17. Generación de PDF formal: Validación de formato y legibilidad.
18. Creación de sesión en AIDA y carga de contexto: Disponibilidad inmediata de la transcripción en la nueva sesión.
19. Establecer la conexión al email, en este caso Outlook: Validación del flujo y de los parámetros del objeto EmailConnection resultante, verificando que son los correctos.
20. Envío de PDF por correo con enlace a AIDA: Verificación de la correcta recepción del correo, por Outlook, con el archivo adjunto y el enlace, comprobando que este sea funcional.
21. Notificación de errores en cualquier punto: Simulación de fallos y comprobación de alertas.
22. Registro de logs: Comprobación de la trazabilidad y detalle de eventos clave.

8.3 Pruebas de requisitos no funcionales

- Seguridad: Verificación de uso exclusivo de HTTPS/TLS, no almacenamiento de credenciales y eliminación automática de archivos tras el envío.
- Escalabilidad: Ejecución de múltiples solicitudes simultáneas y validación de rendimiento en entornos con carga.

- Rendimiento: Medición de tiempos de descarga, transcripción y generación de resumen, verificando cumplimiento de los límites establecidos.
- Mantenibilidad y monitoreo: Revisión de documentación, cobertura de pruebas unitarias y monitorización de métricas clave.

8.4 Pruebas de funcionalidad y aspectos internos de desarrollo

A continuación, se presentan algunas de las pruebas realizadas para cada funcionalidad.

No se incluyen todas las pruebas ejecutadas, sino una selección representativa que abarca las más relevantes, procurando cubrir distintos tipos de validaciones y escenarios.

El objetivo es mostrar la amplitud y variedad de la cobertura sin sobrecargar el documento con un número excesivo de casos de prueba.

8.4.1 Procesamiento multimedia

A continuación, se presentan algunas de las pruebas realizadas para evaluar las funcionalidades de procesamiento multimedia. Estas pruebas se han ejecutado tanto con FFmpeg como con AWS.

Cabe destacar que, en la práctica, se han evaluado todas las posibilidades de los parámetros de entrada. Se han probado todos los formatos de salida de audio y de imagen de los fotogramas extraídos, tanto con un canal como con dos, y todos los valores posibles de frecuencia de muestreo, etc. Además, se han combinado diferentes opciones entre sí para cubrir la mayor cantidad de escenarios posibles. También se verificó la respuesta del sistema ante valores fuera de rango en cada parámetro, asegurando que se detecten correctamente y que se generen los errores correspondientes.

Se explica todo esto para contextualizar que, a continuación, se resumen únicamente siete de las pruebas realizadas de procesamiento multimedia, con diferentes parámetros de entrada, de la prueba 12 a la 18.

ID	Descripción del caso de prueba	Datos de entrada	Resultado esperado	Resultado obtenido	Estado
1	Configuración de permisos para MediaConvert	Usuario AWS con permisos, JSON inicial de IAM	Usuario con permisos iam:PassRole, mediaconvert:CreateJob, mediaconvert:GetJob y acceso a S3 (s3:GetObject, s3:PutObject)	Usuario configurado correctamente con todos los permisos	OK
2	Creación de rol ARN para MediaConvert	Parámetros: Control del rol de servicio = Crear nuevo rol, Nombre rol = MediaConvert_DOCs_Role, Origen etiquetas = Trabajo	Rol ARN generado con permisos completos para MediaConvert y acceso a S3	Rol ARN generado correctamente	OK
3	Creación del cliente MediaConvert en Python	Sesión Boto3 con aws_access_key_id, aws_secret_access_key y region_name="eu-central-1"	Cliente MediaConvert creado y funcional para realizar llamadas a la API	Cliente creado correctamente y listo para ejecutar jobs	OK

4	Configuración de permisos para Rekognition	Usuario AWS con permisos	Usuario con permisos rekognition: StartSegmentDetection y rekognition: GetSegmentDetection	Usuario configurado correctamente con permisos para Rekognition	OK
5	Creación del cliente Rekognition en Python	Sesión Boto3 con aws_access_key_id, aws_secret_access_key y region_name="eu-central-1"	Cliente Rekognition creado y funcional para realizar llamadas a la API	Cliente creado correctamente y listo para detección de segmentos	OK
6	Configuración de permisos para SQS y SNS	Usuario AWS con permisos	Usuario con permisos: SQS (CreateQueue, DeleteQueue, GetQueueAttributes, SetQueueAttributes, ReceiveMessage, DeleteMessage), SNS (CreateTopic, Subscribe, Unsubscribe, GetTopicAttributes)	Usuario ya contaba con permisos, verificados correctamente	OK
7	Creación del cliente SQS	Sesión Boto3 con credenciales y region_name="eu-central-1"	Cliente SQS creado y funcional para recibir y borrar mensajes	Cliente creado correctamente y listo para operaciones con colas	OK
8	Creación del cliente SNS	Sesión Boto3 con credenciales y region_name="eu-central-1"	Cliente SNS creado y funcional para publicar y suscribirse a tópicos	Cliente creado correctamente y listo para operaciones con tópicos	OK
9	Procesamiento de un archivo de video con MediaConvert	Archivo de video en S3, trabajo MediaConvert configurado con rol ARN	Video procesado correctamente, salida guardada en S3	Video procesado exitosamente y almacenado en S3	OK
10	Detección de segmentos de pantalla con Rekognition	Video procesado por Rekognition, trabajo Rekognition configurado	Segmentos detectados correctamente y resultados disponibles	Segmentos detectados correctamente	OK
11	Notificación de finalización de trabajo con SNS + SQS	Trabajo MediaConvert finalizado, mensaje enviado a SNS y recibido por SQS	Mensaje recibido en SQS confirmando finalización de trabajo	Mensaje recibido correctamente	OK
12	Separar el audio de un video en formato FLAC estéreo a 48 kHz y 24 bits	prueba.mp4, formato=flac, canales=2, sample_rate=48000, bit_depth=24	Archivo de audio en formato FLAC, estéreo, 48 kHz y 24 bits extraído correctamente del video	Archivo generado correctamente con las especificaciones	OK

13	Normalizar audio WAV mono a -23 LUFS, -1 dBTP y 12 LU de rango dinámico	prueba.wav, canales=1, sample_rate=44100, bit_depth=16, I=-23, Tp=-1, LRA=12	Audio WAV mono con volumen ajustado a los parámetros de normalización	El archivo resultante cumple con los parámetros de normalización	OK
14	Reducir ruido en audio MP3 estéreo con frecuencia de corte de 100 Hz, reducción de 15 dB y umbral de ruido -50 dB	prueba.mp3, canales=2, sample_rate=48000, bit_depth=16, highpass=100, noise_reduction=15, noise_floor=-50	Audio limpio con reducción de ruido según los parámetros	El ruido residual está dentro de los límites aceptables	OK
15	Extraer un fotograma en formato JPG del segundo 00:00:15.500 de un video	prueba.mp4, timestamp= 00:00:15.500, formato=jpg	Imagen en formato JPG correspondiente al fotograma indicado	Imagen generada correctamente y corresponde al tiempo especificado	OK
16	Detectar cambios de escena en un video con umbral 0.3	prueba.mp4, shot_threshold=0.3	Lista de segmentos con timestamps que correspondan a cambios de escena reales	Segmentos detectados coinciden con los cambios observados visualmente	OK
17	Separar audio de video con parámetros inválidos (bit_depth=40)	prueba.mp4, formato=wav, canales=2, sample_rate=44100, bit_depth=40	Error por parámetro no válido y no se genera archivo	El sistema rechaza el trabajo y devuelve error de validación	OK
18	Extraer frame con timestamp fuera de la duración del video	prueba.mp4, timestamp = 00:59:59.999, formato=png	Error indicando que el timestamp es inválido	El sistema devuelve mensaje de error y no genera imagen	OK

Tabla 5. Pruebas procesamiento multimedia (FFmpeg y AWS)

8.4.2 Transcripción y diarización

En esta sección se muestran las pruebas para verificar el correcto funcionamiento del nodo que se encarga de realizar la transcripción y diarización.

ID	Descripción del caso de prueba	Datos de entrada	Resultado esperado	Resultado obtenido	Estado
1	Creación del cliente de Amazon Transcribe	session.client("transcribe", region_name="eu-central-1")	Cliente creado correctamente sin errores	Cliente creado correctamente	OK

2	Verificación de permisos de transcripción	Usuario con permisos transcribe: StartTranscriptionJob y transcribe: GetTranscriptionJob	El usuario puede iniciar y consultar trabajos de transcripción	Permisos funcionan correctamente	OK
3	Subida de archivo de audio con nombre adecuado a S3	Audio en objeto FileData	Archivo subido a S3 con nombre UUID + nombre original	Archivo subido correctamente	OK
4	Creación de vocabulario personalizado desde lista	Lista de términos válida	Vocabulario creado correctamente en AWS Transcribe	Vocabulario creado	OK
5	Creación de vocabulario personalizado desde archivo de texto	Archivo FileData con vocabulario	Vocabulario decodificado y creado correctamente	Vocabulario creado	OK
6	Manejo de vocabulario vacío	Lista vacía o archivo sin términos	Error indicando que el vocabulario no puede estar vacío	Error lanzado correctamente	OK
7	Manejo de caracteres especiales no soportados	Lista o archivo con caracteres no válidos	Error capturado y aviso al usuario con referencia a la documentación	Error capturado y aviso mostrado	OK
8	Monitorización del estado del vocabulario	Estado READY	La función continúa la ejecución	Función continúa correctamente	OK
9	Monitorización del estado del vocabulario	Estado FAILED	Error lanzado indicando fallo en vocabulario	Error lanzado correctamente	OK
10	Inicio de trabajo de transcripción	Parámetros: S3 URL, formato audio, bucket, idioma, configuración	Trabajo iniciado correctamente en AWS Transcribe	Trabajo iniciado correctamente	OK
11	Diarización de hablantes	Audio monocanal, ShowSpeakerLabels= True	Texto transcrito separado por hablantes (spk_0, spk_1, …)	Diarización correcta	OK
12	Monitoreo del trabajo de transcripción	Uso de SNS y SQS	Detección de finalización correctamente	Finalizado correctamente	OK

13	Descarga resultados transcripción	Job finalizado	Archivo JSON descargado desde S3 y devuelto como FileData	JSON descargado y devuelto correctamente	OK
14	Eliminación de archivos temporales en S3	Audio original y JSON con la transcripción	Archivos eliminados correctamente	Archivos eliminados	OK
15	Eliminación de vocabulario personalizado	Vocabulario creado	Vocabulario eliminado mediante API	Vocabulario eliminado	OK
16	Manejo de alternativas de transcripción	ShowAlternatives=True, MaxAlternatives=3	Se muestran hasta 3 alternativas por palabra	Alternativas generadas correctamente	OK
17	Transcripción con idioma especificado	language_code válido	Audio transcrito en el idioma indicado	Transcripción correcta	OK
18	Error por idioma no soportado	language_code inválido	Error lanzado indicando idioma no soportado	Error lanzado correctamente	OK

Tabla 6. Pruebas de transcripción y diarización

8.4.3 Reducir tamaño de transcripción

En esta sección se muestran las pruebas para verificar el correcto funcionamiento del nodo que se encarga de reducir el tamaño de la transcripción.

ID	Descripción del caso de prueba	Datos de entrada	Resultado esperado	Resultado obtenido	Estado
1	Verificar que se extraen solo las claves necesarias	Diccionario completo de Amazon Transcribe con results, transcripts, audio_segments, metadata y otras claves auxiliares	Nuevo diccionario solo con transcripts y audio_segments	Nuevo diccionario contiene únicamente transcripts y audio_segments	OK
2	Calcular campo confianza por segmento	Segmentos de audio con palabras y valores de confianza individuales por palabra	Cada segmento incluye un campo confianza con el promedio de las confianzas de las palabras del segmento	Cada segmento tiene confianza calculada correctamente como promedio	OK
3	Seleccionar palabra más confiable en audio_segments	Segmentos con varias alternativas para la misma palabra con diferentes valores de confianza	Se conserva solo la palabra con mayor confianza en cada posición	Se selecciona correctamente la palabra más confiable	OK

4	Conservación del texto completo en transcripts	Texto transcrito completo	transcripts contiene el texto íntegro sin pérdida de información	transcripts contiene todo el texto correctamente	OK
5	Reducción del tamaño del objeto en memoria	Diccionario original grande con múltiples claves y listas de palabras	Diccionario reducido significativamente en tamaño al eliminar claves y datos no esenciales	Diccionario reducido contiene solo transcripts, audio_segments y confianza	OK
6	Manejo de segmentos vacíos	audio_segments vacío o sin palabras	Función devuelve diccionario válido con audio_segments vacío y transcripts intacto	Diccionario reducido contiene audio_segments vacío y transcripts	OK
7	Validar funcionamiento con múltiples hablantes	Segmentos de audio con diarización de 2 o más hablantes	Se mantiene la asignación correcta de hablantes en cada segmento	Asignación de hablantes correcta en audio_segments	OK
8	Validar funcionamiento con confianza máxima 1	Segmentos con valores de confianza entre 0 y 1	Promedio calculado por segmento entre 0 y 1	Promedio calculado correctamente	OK
9	Validar funcionamiento con valores de confianza 0	Segmentos con palabras con confianza 0	Promedio calculado como 0 y no genera errores	Promedio = 0 correctamente calculado	OK

Tabla 7. Pruebas de reducir tamaño de transcripción

8.4.4 Manejo de la estructura de datos de la extracción del nombre de los hablantes

En este apartado se muestran las pruebas para verificar el correcto funcionamiento de los nodos que se encargan de gestionar la estructura de datos usada al extraer el nombre de los locutores.

ID	Descripción del caso de prueba	Datos de entrada	Resultado esperado	Resultado obtenido	Estado
1	Preparar diálogos de transcripción (Nodo 7)	JSON completo de Amazon Transcribe con results.items, incluyendo speaker_label, palabras y confianza	Lista de diálogos por hablante con campos: speaker_label, text, start_time, end_time, confidence, total_speakers	Lista generada correctamente con diálogos agrupados por hablante y confianza promedio	OK

2	Preparar parámetros de hablantes (Nodo 8)	Lista de diálogos generada en Nodo 7	Diccionario con list de objetos speaker_json por hablante, all_found=0, count=1, num_spk="", total_speakers	Diccionario generado con todos los campos correctos y current_timestamp calculado correctamente	OK
3	Nodo 9: Primera iteración de juntado de parámetros	Diccionario del Nodo 8	Salida igual a la entrada que proviene del Nodo 8	Salida coincide correctamente	OK
4	Nodo 9: Iteraciones posteriores	Diccionario proveniente del Nodo 16	Salida igual a la entrada del Nodo 16	Salida coincide correctamente	OK
5	Nodo 16: Actualización de resultados sin conflictos	JSON con nombres detectados, lista de hablantes, count de iteración	Lista de hablantes actualizada, nombres asignados correctamente, all_found actualizado según número de identificados, count incrementado si no todos encontrados	Lista y campos all_found, count, num_spk actualizados correctamente	OK
6	Nodo 16: Detección de conflictos en nombres	El mismo nombre asignado a dos hablantes	Nombres duplicados eliminados, found=False en hablantes afectados, lista actualizada	Conflictos resueltos y lista actualizada correctamente	OK
7	Nodo 16: Asignación automática del último nombre	Todos los nombres menos uno asignados	Último hablante recibe el nombre restante automáticamente	Último nombre asignado correctamente	OK
8	Nodo 16: Manejo de múltiples iteraciones	Algunos hablantes no identificados tras la primera iteración	count incrementado, current_timestamp actualizado según regla (50%, 25%, 75%)	count y current_timestamp actualizados correctamente	OK
9	Nodo 16: Finalización forzada por límite de iteraciones	Más de 3 iteraciones sin identificar todos los hablantes	all_found=2 indicando finalización forzada	all_found=2 correctamente asignado	OK

Tabla 8. Pruebas de la gestión de la estructura de datos de extracción del nombre de hablantes

8.4.5 Detección de cambios de escena

Esta funcionalidad fue evaluada en la sección de pruebas de procesamiento multimedia, específicamente en el apartado *8.4.1 Procesamiento multimedia*, donde se verifican todas las funcionalidades relacionadas con el procesamiento de audio y video.

8.4.6 Detección de segmentos de pantalla compartida

A continuación, se muestran las pruebas para verificar el correcto funcionamiento del nodo que se encarga de detectar si en un segmento determinado se comparte pantalla o no.

ID	Descripción del caso de prueba	Datos de entrada	Resultado esperado	Resultado obtenido	Estado
1	Conversión de marca temporal a segundos	hora_str = "00:05:30.500"	Función hora_a_segundos retorna 330.5 segundos	Retorna 330.5 segundos	OK
2	Conversión a escala de grises	video_path y time_str = 25% duración del segmento	Frame extraído y convertido a escala de grises	Frame extraído y convertido correctamente	OK
3	Detección de rostros en frame	Frame representativo	detectar_rostros retorna número correcto de rostros en la imagen	Núm. de rostros detectados correctamente	OK
4	Cálculo de cantidad de bordes	Frame representativo	cantidad_bordes retorna número correcto de bordes aplicando Canny	Cantidad de bordes calculada correctamente	OK
5	Clasificación de frame según umbral	Frame con num_rostros y num_bordes	clasificar_segmento retorna "Pantalla compartida" o "Sin compartición" según lógica	Clasificación correcta aplicada	OK
6	Clasificación de segmento con 3 frames consistentes	3 frames con misma clasificación	Segmento adoptado según coincidencia de 3 frames	Segmento clasificado correctamente	OK
7	Clasificación de segmento con 3 frames inconsistentes	3 frames con distinta clasificación	Se analiza 2 frames adicionales y se decide por mayoría absoluta	Segmento clasificado según mayoría	OK
8	Manejo de frames sin rostros	Frame con 0 rostros	Umbral calculado = 4000; clasifica correctamente según bordes	Clasificación correcta para sin rostros	OK
9	Manejo de frames con bordes excesivos	Frame con bordes >> umbral	Clasifica correctamente como "Pantalla compartida"	Clasificación correcta	OK

Tabla 9. Pruebas de detección de segmentos de pantalla compartida

8.4.7 Preparación de los parámetros de los subgrafos

En este caso, se realizan pruebas sencillas. Primero, se verifica que los frames extraídos por los nodos 10 y 20 sean los esperados. Luego, se comprueba que la longitud n de las listas obtenidas de estos nodos coincida con la esperada y, finalmente, se asegura que los nodos 12, 13, 22 y 23 devuelvan una lista con n elementos repetidos a partir de los recibidos en la entrada.

8.4.8 Funcionamiento de los subgrafos

A continuación, se muestran las pruebas realizadas para probar que los subgrafos funcionan correctamente y realizan sus tareas como es debido.

ID	Descripción del caso de prueba	Datos de entrada	Resultado esperado	Resultado obtenido	Estado
1	Extracción de fotograma del vídeo	Video como FileData, timestamp	Frame extraído correctamente	Frame extraído correctamente	OK
2	Obtención de bytes del fotograma	Frame extraído con atributo file_bytes	Lista de diccionarios con file_bytes de la imagen	Lista generada correctamente	OK
3	Llamada al LLM para nombre del hablante	Imagen en binario, system_prompt detallado, prompt de extracción de nombre, output_formatter_dict, text_variables	JSON con campos: found, speaker_name, error_case, num_spk, spk_names	JSON generado correctamente según formato esperado	OK
4	Identificación de hablante único mediante marco azul en vista central	Frame con un solo indicador válido	found=true, speaker_name correcto, error_case=null	Correctamente identificado	OK
5	Identificación de hablante único mediante fondo azul en barra lateral	Frame con fondo azul en nombre del participante	found=true, speaker_name correcto, error_case=null	Correctamente identificado	OK
6	Doble confirmación (marco azul + fondo azul coincidente)	Frame con ambos indicadores en la misma persona	found=true, speaker_name correcto, error_case=null	Correctamente identificado	OK
7	Caso no_active_speaker	Frame sin indicadores	found=false, speaker_name="error", error_case= "no_active_speaker"	Correctamente detectado	OK

8	Caso multiple_ active_speakers	Frame con indicadores en varias personas	found=false, speaker_name="error", error_case= "multiple_ active_speakers"	Correctamente detectado	OK
9	Caso unclear_ visual_indicators	Frame con imagen borrosa o colores ambiguos	found=false, speaker_name="error", error_case= "unclear_ visual_indicators"	Correctamente detectado	OK
10	Extracción de información de pantalla compartida	Imagen de pantalla compartida, prompt y system_prompt de pantalla compartida	JSON con campo description describiendo el contenido visible	JSON generado correctamente	OK
11	Exclusión de elementos irrelevantes en análisis de pantalla compartida	Frame con barra lateral e iconos	description incluye solo información relevante de la presentación	Correctamente excluidos los elementos irrelevantes	OK
12	Verificación de formato de salida para pantalla compartida	Output del LLM	JSON con campo description, sin texto adicional	Formato correcto	OK
13	Manejo de frames sin contenido visible	Frame vacío o negro	description vacía o indicando contenido nulo	Correctamente manejado	OK

Tabla 10. Pruebas de los subgrafos

8.4.9 Extraer nombres resultantes

En esta sección se presentan las pruebas realizadas para verificar que los nombres de los hablantes identificados se extraen correctamente y puedan utilizarse adecuadamente posteriormente.

ID	Descripción del caso de prueba	Datos de entrada	Resultado esperado	Resultado obtenido	Estado
1	Procesar diccionario con varios hablantes identificados	speaker_data = {'speakers': [{'num_speaker':0,'found' : True, 'name':'Ana'}, {'num_speaker':1,'found': True, 'name':'Luis'}]}	Diccionario con total_speakers=2, mapping={'spk_0':'Ana', 'spk_1':'Luis'}, resumen textual con ambos nombres, all_identified=True	Igual al esperado	OK

2	Procesar diccionario con hablantes no identificados	speaker_data = {'speakers': [{'num_speaker':0 ,'found':False, 'name':'unknown'}, {'num_speaker':1, 'found':True, 'name':'Luis'}]}	Diccionario con mapping={'spk_0':'not_known', 'spk_1':'Luis'}, resumen textual indicando un desconocido, all_identified=False	Igual al esperado	OK
3	Procesar diccionario vacío de hablantes	speaker_data = {'speakers':[]}	Diccionario con total_speakers=0, mapping vacío, resumen indicando 0 hablantes, all_identified=True	Igual al esperado	OK
4	Procesar diccionario sin clave 'num_spk'	speaker_data = {'speakers': [{'num_speaker':0, 'found':True, 'name':'Ana'}]}	total_speakers calculado automáticamente =1, mapping={'spk_0':'Ana'}, resumen adecuado, all_identified=True	Igual al esperado	OK
5	Procesar diccionario con todos los hablantes no identificados	speaker_data = {'speakers':[{'num_speaker':0, 'found':False, 'name':'unknown'}, {'num_speaker':1, 'found':False, 'name':'unknown'}]}	mapping={'spk_0':'not_known', 'spk_1':'not_known'}, resumen indicando ambos desconocidos, all_identified=False	Igual al esperado	OK
6	Procesar diccionario con nombres parcialmente vacíos o "unknow"	speaker_data = {'speakers':[{'num_speaker':0, 'found':True, 'name':'unknown'}, {'num_speaker':1, 'found':True, 'name':'Luis'}]}	mapping={'spk_0':'not_known', 'spk_1':'Luis'}, resumen textual correcto, all_identified=False	Igual al esperado	OK

Tabla 11. Pruebas de extraer nombres resultantes

8.4.10 Corrección transcripción + Generar acta

En este apartado se presentan las pruebas realizadas sobre la corrección de transcripciones y la generación del contenido del acta.

No se verifica el funcionamiento de las llamadas al LLM, ya que, tal como se explicó en la implementación, no fue desarrollado por mí. En su lugar, se han evaluado únicamente los resultados obtenidos.

Estas pruebas han resultado complejas, ya que requieren un análisis detallado de la entrada, del comportamiento del LLM, de cómo genera la salida y de la verificación de que actúa según lo esperado. A continuación, se resumen algunos fragmentos que evidencian que

el sistema funciona correctamente y que el LLM cumple satisfactoriamente con la tarea encomendada.

ID	Descripción del caso de prueba	Datos de entrada	Resultado esperado	Resultado obtenido	Estado
1	Transcripción breve sin errores	Transcripción: "Reunión de coordinación con Ana y Luis." Nombres: ["Ana","Luis"] Pantalla compartida: "Diagrama de flujo de proyecto"	Acta en JSON con markdown, encabezado, lista de participantes, resumen de tema tratado, sin correcciones necesarias	Acta generada correctamente con formato markdown y contenido coherente	OK
2	Transcripción con errores ortográficos y de puntuación	Transcripción: "Ana discutio el presupuesto final" Nombres: ["Ana"]	Acta con corrección: "Ana discutió el presupuesto final", markdown con lista de participantes, temas, acuerdos, etc.	Acta generada con correcciones reflejadas y estructura correcta	OK
3	Múltiples participantes con información parcial	Transcripción: "Ana: Revisamos el proyecto. Luis: Pendiente el informe" Nombres: ["Ana","Luis"]	Acta con participantes listados, temas tratados por cada persona, acuerdos y pendientes, formato markdown	Acta generada con secciones por participante, resumen, pendientes claramente reflejados	OK
4	Transcripción larga con datos faltantes	Transcripción: "Se discutieron los próximos pasos." Nombres: ["Ana","Luis"] Pantalla: ""	Acta generada correctamente, aunque no haya información sobre la pantalla compartida	Acta generada correctamente	OK
5	Transcripción con fecha y próxima reunión mencionada	Transcripción: "Próxima reunión: 20/08/2025 a las 10am" Nombres: ["Ana","Luis"]	Acta que extraiga correctamente la fecha y hora de la próxima reunión, lista de participantes, resumen de temas, acuerdos y acciones	Acta generada con fecha de próxima reunión incluida correctamente	OK
6	Verificar que no se inventen datos	Transcripción: "Discutimos ajustes al proyecto" Nombres: ["Ana"] Pantalla: ""	Acta que solo incluya información presente en transcripción, nombres o pantalla, sin añadir datos externos	Acta generada sin información inventada, contenido consistente con la transcripción	OK

7	Verificar formato JSON final	Transcripción: Cualquier transcripción válida	Salida JSON con un único campo “acta” que contenga todo en markdown	Acta generada correctamente en JSON con campo “acta”	OK
8	Validar coherencia y consistencia del acta	Transcripción: "Ana: Revisar informes. Luis: Enviar resumen final" Nombres: ["Ana","Luis"]	Acta clara, con secciones de temas por participante, acuerdos y acciones, sin contradicciones	Acta generada correctamente y consistente	OK

Tabla 12. Pruebas de la corrección de la transcripción y la generación del contenido del acta

8.4.11 Ground truth validator

Esta funcionalidad no fue desarrollada por mí, tal como se indicó en el apartado de implementación. Por ello, no se han realizado pruebas de su funcionamiento interno, ya que se asume que fue aprobada por el desarrollador.

Lo que sí hice fue verificar que la respuesta generada fuera coherente con la entrada proporcionada, para confirmar su utilidad en mi proyecto. La evaluación fue positiva, motivo por el cual decidí adoptarla en lugar de desarrollar otro componente similar que realizara la misma tarea que este nodo.

8.4.12 Markdown a HTML

A continuación, se muestran las pruebas realizadas para probar el correcto funcionamiento del nodo que se encarga de convertir el contenido del acta en Markdown a HTML.

ID	Descripción del caso de prueba	Datos de entrada	Resultado esperado	Resultado obtenido	Estado
1	Conversión simple sin SVG	Markdown: # Acta de reunión \nParticipantes: Ana, Luis	HTML con encabezado <h1> y párrafo, sin errores de formato	HTML generado correctamente con estructura <h1> y <p>	OK
2	Conversión con tabla en Markdown	Markdown con tabla de acuerdos	HTML con tabla <table> correctamente estructurada	HTML generado con <table>, <tr>, <td> correcto	OK
3	Inclusión de diagrama SVG válido	Markdown con bloque de código que contiene <svg>...</svg>	Bloque SVG convertido a <img> con contenido codificado en base64 y estilos predefinidos	Imagen SVG embebida correctamente en HTML	OK
4	Limpieza de caracteres especiales en SVG	Markdown con SVG que contiene entidades HTML	SVG limpio y visualizable correctamente en HTML	SVG mostrado sin caracteres corruptos	OK

5	Documento con múltiples SVG	Markdown con dos bloques SVG	Cada SVG convertido a <img> base64 y renderizado correctamente	Todos los SVG mostrados como imágenes embebidas	OK
6	Conversión de Markdown con código que no es SVG	Markdown con bloque de código en Python	El bloque de código se mantiene como <pre><code> en HTML, sin conversión a imagen	Código mostrado en HTML sin alteraciones	OK
7	Inclusión de bloque de advertencia final	Markdown cualquiera	HTML final con bloque "Documento generado automáticamente" con fondo oscuro y letra clara	Bloque añadido correctamente al final del HTML	OK
8	Verificación de coherencia visual final	Markdown con texto, tabla y SVG	HTML final con todos los elementos correctamente renderizados y estilos coherentes	Documento con buena presentación visual	OK

Tabla 13. Pruebas de la conversión de Markdown a HTML

8.4.13 HTML a PDF

A continuación, se presentan las pruebas realizadas para verificar el correcto funcionamiento del nodo encargado de convertir el contenido del acta, previamente transformado a HTML, en un archivo PDF.

ID	Descripción del caso de prueba	Datos de entrada	Resultado esperado	Resultado obtenido	Estado
1	Conversión simple de HTML a PDF	HTML básico con Acta Contenido	PDF generado con título y párrafo, conservando formato	PDF correcto con texto formateado	OK
2	HTML con tablas y estilos CSS	HTML con <table> y estilos en <style>	PDF con tabla formateada igual que en HTML	Tabla y estilos preservados	OK
3	Inclusión de pie de página estándar	HTML simple, parámetro footer con texto "© 2025 Applus+ IDIADA - Creado con DOCs"	PDF con pie de página centrado en todas las páginas	Pie de página correcto y legible	OK
4	Pie de página con año dinámico	HTML simple, footer con placeholder YEAR	PDF con año actual reemplazado por "2025"	Año reemplazado correctamente	OK
5	PDF sin pie de página	HTML simple, sin footer proporcionado	PDF generado sin pie de página	Sin footer y formato intacto	OK

6	Asignación automática de nombre de archivo	HTML simple, sin nombre de archivo	PDF con nombre Meeting_minutes_ YYYYMMDD_ HHMMSS.pdf	Nombre asignado correctamente	OK
7	Conversión de HTML con varias páginas	HTML de 4 páginas	PDF generado con salto de página correcto y pie de página en todas	Documento completo sin cortes	OK
8	Manejo de HTML vacío	HTML vacío	Error controlado o mensaje de validación "HTML vacío"	Mensaje de error mostrado	OK
9	Eliminación de archivos temporales	HTML simple, revisión postproceso	Archivos temporales borrados tras conversión	Sin temporales residuales	OK

Tabla 14. Pruebas de conversión de HTML a PDF

8.4.14 Gestión de correos electrónicos

En esta sección se presentan las pruebas realizadas para verificar la correcta configuración de los parámetros de conexión al correo electrónico, específicamente Outlook en este caso, y el envío del mensaje con el acta adjunta.

ID	Descripción del caso de prueba	Datos de entrada	Resultado esperado	Resultado obtenido	Estado
1	Conexión SMTP exitosa con credenciales correctas	Usuario, contraseña válidos, servidor smtp.office365.com	Conexión establecida sin errores	Conexión realizada correctamente	OK
2	Conexión SMTP fallida por credenciales incorrectas	Usuario válido, contraseña incorrecta	Error de autenticación controlado y mensaje en logs	Error capturado y log registrado	OK
3	Detección automática de servidor SMTP	Usuario usuario@gmail.com, sin servidor especificado	Servidor detectado como smtp.gmail.com y conexión exitosa	Servidor detectado y conexión correcta	OK
4	Envío de correo HTML con asunto y cuerpo personalizado	Destinatario válido, asunto "Automated Meeting Report", cuerpo HTML	Correo recibido con formato HTML correcto	Correo recibido con formato esperado	OK
5	Envío de correo con adjunto PDF	Destinatario válido, adjunto PDF acta.pdf	Correo recibido con adjunto funcional	PDF recibido y accesible	OK

6	Envío de correo con múltiples adjuntos	Lista de FileData con PDF e imagen PNG	Correo recibido con ambos adjuntos intactos	Adjuntos recibidos y legibles	OK
7	Manejo de adjunto corrupto	FileData con bytes incompletos	Error controlado y mensaje de fallo en logs	Error capturado y registrado	OK
8	Envío a destinatario inválido	Dirección de correo malformada	Error controlado en envío y log generado	Error detectado y logueado	OK
9	Uso de conexión segura STARTTLS	Usuario válido, conexión por puerto 587	Conexión cifrada y establecida	Conexión cifrada confirmada	OK
10	Cierre correcto de conexión tras envío	Usuario y servidor válidos	Conexión cerrada sin recursos bloqueados	Cierre confirmado y sin fugas	OK

Tabla 15. Pruebas de la configuración y envío de correos electrónicos

8.4.15 Gestión de la descarga de grabaciones de reuniones desde SharePoint

En este apartado se presentan las pruebas realizadas para verificar la correcta configuración de los parámetros de conexión a SharePoint y la descarga de grabaciones de reunión.

ID	Descripción del caso de prueba	Datos de entrada	Resultado esperado	Resultado obtenido	Estado
1	Conexión a SharePoint con credenciales válidas	client_id, client_secret, tenant_id correctos	Token de acceso válido generado y conexión establecida	Token obtenido y conexión establecida correctamente	OK
2	Conexión a SharePoint con credenciales inválidas	client_id incorrecto	Error de autenticación controlado y registro en logs	Error capturado y log registrado	OK
3	Obtención automática de token de acceso	Credenciales correctas sin token previo	Token válido generado usando MSAL	Token generado correctamente	OK
4	Descarga de grabación con IDs correctos	library_id y document_id existentes y con permisos	Archivo descargado en memoria como BytesIO con metadatos correctos	Archivo recibido, tamaño y tipo validados	OK
5	Descarga de grabación sin permisos de acceso	IDs correctos, pero sin autorización	Error controlado y registro de acceso denegado	Error registrado correctamente	OK

6	Descarga de grabación inexistente	document_id inválido	Error controlado indicando que el archivo no existe	Error capturado y log registrado	OK
7	Identificación correcta del tipo de contenido	Archivo con extensión .mp4	Tipo MIME detectado como video/mp4	Tipo detectado correctamente	OK
8	Verificación de metadatos de grabación	Archivo con organizador y permisos definidos	Acceso concedido si metadatos cumplen criterios	Acceso evaluado correctamente según metadatos	OK

Tabla 16. Pruebas de la configuración de los parámetros y descarga de grabaciones de reuniones desde SharePoint

8.4.16 Crear sesión de AIDA

A continuación, se muestran las pruebas realizadas para probar la correcta creación de la sesión en el asistente inteligente AIDA y que el contexto se carga correctamente con la transcripción.

ID	Descripción del caso de prueba	Datos de entrada	Resultado esperado	Resultado obtenido	Estado
1	Conversión correcta de transcripción JSON a TXT	Transcripción reducida en formato JSON, nombres de hablantes	Archivo TXT con encabezado de hablantes, transcripción y segmentos	Archivo TXT generado correctamente	OK
2	Obtención de token de autorización válido	Credenciales del usuario/empleado válidas	Token válido recibido del endpoint interno	Token obtenido correctamente	OK
3	Error en obtención de token	Credenciales inválidas	Error controlado, mensaje de fallo y sin creación de sesión	Error registrado correctamente	OK
4	Creación de sesión en AIDA con token válido	Token válido	Sesión creada y retorno de session_id válido	Sesión creada y session_id correcto	OK
5	Fallo en creación de sesión por token inválido	Token inválido	Error HTTP controlado, sesión no creada	Error capturado y log registrado	OK
6	Fijar título de sesión correctamente	session_id válido, título definido	Título actualizado en la sesión	Título cambiado con éxito	OK
7	Error al fijar título por session_id inválido	session_id inexistente	Error controlado y registro del fallo	Error capturado correctamente	OK

8	Subida de archivo de transcripción vía WebSocket	session_id válido, token válido, archivo TXT generado	Archivo subido a S3 y confirmación de subida recibida	Subida completada correctamente	OK
9	Fallo en subida de archivo por desconexión WebSocket	Archivo TXT válido, pero conexión interrumpida	Error controlado y subida cancelada	Error registrado correctamente	OK
10	Vinculación correcta de archivo a sesión	session_id válido, file_id correcto	Archivo vinculado a la sesión y accesible como contexto	Archivo vinculado con éxito	OK
11	Fallo en vinculación de archivo a sesión	file_id inválido	Error controlado, archivo no vinculado	Error capturado y log registrado	OK
12	Flujo completo sin errores	Transcripción JSON válida, credenciales correctas, conexión estable	Estado final “ok” con sesión creada y archivo vinculado	Estado “ok” obtenido	OK
13	Flujo completo con fallo en cualquier paso	Error simulado en obtención de token	Estado final “bad” con detalle de error	Estado “bad” y detalle devuelto	OK

Tabla 17. Pruebas de la creación de la sesión de AIDA

8.4.17 Pruebas de integración y construcción del grafo

Entre las pruebas más relevantes realizadas se encuentran aquellas orientadas a la construcción e integración del grafo dirigido que compone el sistema.

Estas pruebas han tenido como objetivo verificar que las funcionalidades implementadas en cada nodo operen correctamente tanto de forma aislada como en conjunto, garantizando así la consistencia y fiabilidad del flujo de procesamiento.

El proceso de construcción se abordó de forma gradual e incremental, siguiendo un enfoque de integración progresiva. En una primera fase, se diseñaron y validaron grafos unitarios de reducido alcance, cada uno orientado a una funcionalidad específica. Algunos ejemplos son:

- Grafos que ejecutan una única operación multimedia, como separación de audio y video, normalización de audio o extracción de fotogramas.
- Grafo destinado exclusivamente a la transcripción y diarización, sin incluir procesamiento multimedia.
- Grafo que, a partir de una transcripción con diarización previamente generada, construye la estructura de datos necesaria para la extracción de nombres de hablantes.
- Subgrafo encargado de identificar al hablante activo a partir de un fotograma suministrado manualmente, con el objetivo de evaluar todos los casos posibles y confirmar el correcto funcionamiento.

En este contexto, se evaluó el comportamiento del LLM recibiendo tanto la imagen en formato parseado como en formato binario, comprobándose que el parseo provoca pérdida de información visual significativa que afecta la precisión en la identificación del locutor.

- Grafo para la creación de una sesión en AIDA.
- Grafo que combina la creación de sesión en AIDA con la adición de contexto.
- Grafo encargado de establecer conexión y enviar correos electrónicos, con y sin archivos adjuntos.
- Grafo para conexión y descarga de grabaciones desde SharePoint.
- Grafo encargado de la corrección de transcripciones obtenidas.
- Grafo responsable de generar el contenido del acta, con y sin corrección previa.

Durante el desarrollo, se construyeron múltiples variantes de cada tipo de grafo, ya que la combinación de nodos podía realizarse de diferentes formas. La aplicación de la regla de nodos atómicos definida en el proyecto DOCs obligó, en algunos casos, a rehacer la estructura varias veces hasta obtener un diseño que cumpliera con los requisitos establecidos.

La integración final se realizó de manera iterativa, siguiendo una lógica de ensamblaje por capas:

1. Integración de grafos de procesamiento multimedia para generar un flujo capaz de procesar el video en su totalidad.
2. Unión de este flujo con el grafo de generación de transcripción y diarización.
3. Integración con el módulo de extracción de nombres de hablantes.
4. Adición progresiva de módulos complementarios, como generación del acta, creación de la sesión de AIDA, envío del correo, etc.

En cada iteración, se ajustaron y estandarizaron los parámetros de entrada y salida, así como los formatos de datos intercambiados entre nodos, asegurando la compatibilidad e interoperabilidad de los componentes.

Este enfoque de desarrollo e integración incremental resultó especialmente efectivo para la detección temprana de errores, la verificación de dependencias entre módulos y la optimización del flujo global. Como resultado, se obtuvo un grafo dirigido con un flujo de procesamiento automatizado, robusto y alineado con los requisitos funcionales y de diseño del sistema.

8.5 Resultados

El 100% de los casos de prueba funcionales planificados han sido ejecutados. Tal como es habitual en un ciclo de desarrollo iterativo, no todas las pruebas obtuvieron un resultado satisfactorio en la primera ejecución. Se identificaron y corrigieron múltiples incidencias durante el proceso. Finalmente, todas las pruebas aquí documentadas, y otras adicionales que no se incluyen en este informe por redundancia o por su escaso aporte de información relevante, han sido superadas con éxito. Esto ha permitido validar que el sistema es funcional en su totalidad, alcanzando una cobertura máxima del código respecto a los criterios establecidos en el plan de pruebas.

En lo que respecta a requisitos no funcionales, el sistema cumple con los criterios de seguridad, rendimiento y mantenibilidad definidos en la fase inicial de especificación. La seguridad se ha verificado mediante validaciones de entrada y control de errores, el rendimiento ha sido evaluado mediante métricas de tiempo de ejecución y consumo de recursos, y la mantenibilidad se ha confirmado gracias a la modularidad de la arquitectura y el cumplimiento de las convenciones de desarrollo definidas.

En el siguiente apartado, se presenta una evaluación comparativa exhaustiva de las dos tecnologías utilizadas en el procesamiento multimedia: FFmpeg y AWS.

Esta comparación se centra en métricas objetivas como: Uso de recursos, rendimiento y coste. Además, se han ejecutado pruebas del comportamiento del pipeline completo utilizando ambas tecnologías.

También se incluye un análisis comparativo de los resultados de transcripción y diarización obtenidos mediante FFmpeg y AWS. Este análisis contempla parámetros como la precisión en la segmentación de hablantes y la tasa de error de palabras, es decir WER[23].

Por lo tanto, en los siguientes apartados se presentarán resultados numéricos detallados derivados de estas evaluaciones, incluyendo tablas comparativas, métricas de rendimiento y gráficos estadísticos. Estos valores constituyen una validación cuantitativa del trabajo realizado, complementando la verificación cualitativa obtenida en las pruebas funcionales.

8.5.1 Muestras de actas generadas

A continuación, se presentan algunas muestras del resultado de las actas generadas automáticamente.

Los apartados generales del documento se presentan con una fuente de mayor tamaño en negrita y con una línea continua debajo. Tras esta línea se redacta la información correspondiente, tal como se muestra en la imagen a continuación.

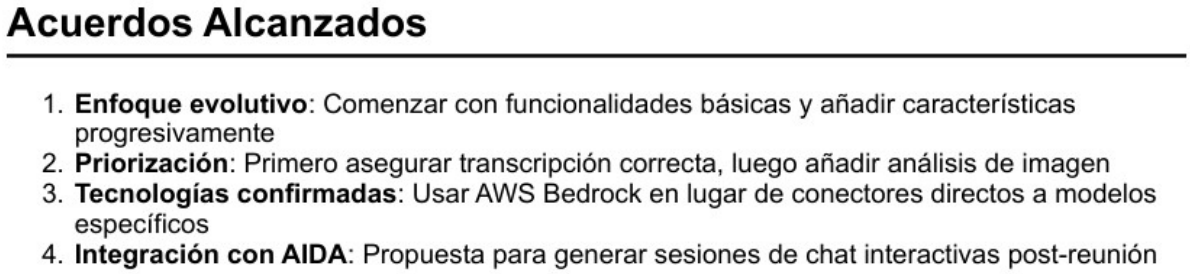


Figura 26. Estilo secciones generales del acta

[23] Word Error Rate

Además de los apartados generales, dentro de estos pueden incluirse subapartados, los cuales pueden ir numerados o no dependiendo del contexto, como se muestra en el ejemplo a continuación.

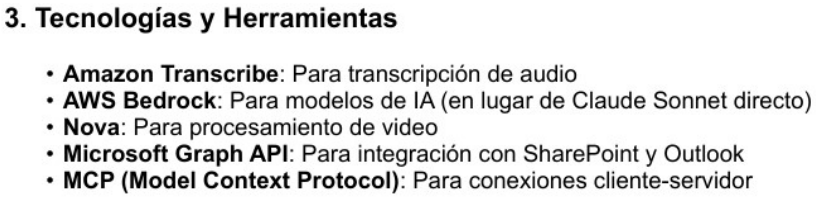


Figura 27. Estilo sección subgeneral numerada del acta

A continuación, se muestra el pie de página del documento.

Figura 28. Pie de página del acta

A continuación, se muestra la nota al final del documento que indica que el documento ha sido generado automáticamente, con el fin de concienciar a los usuarios y garantizar la transparencia.

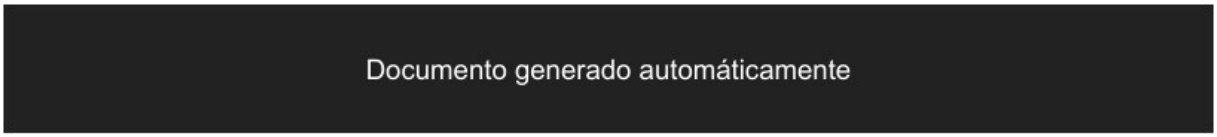


Figura 29. Estilo nota al final del acta indicando que el documento ha sido generado automáticamente

En la sección dedicada a la implementación del proyecto se explicó que las actas generadas automáticamente pueden incluir diagramas cuando sea necesario y la información lo justifique (por ejemplo, al tratar flujos durante la reunión). Estos diagramas se procesan en formato SVG y HTML y posteriormente se insertan en el documento. A continuación, se muestra el estilo de un ejemplo de diagrama generado automáticamente a partir de una reunión.

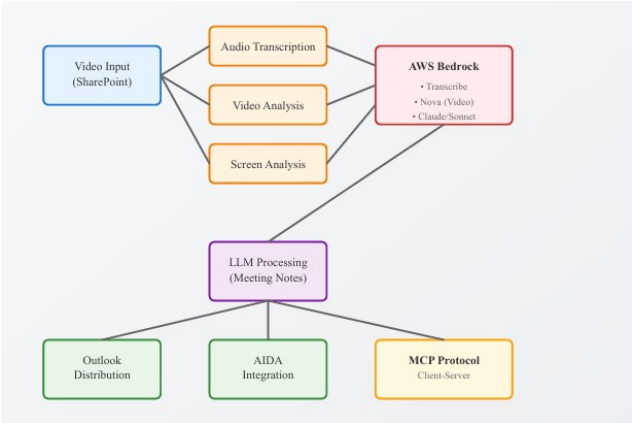


Figura 30. Diagrama generado automáticamente en el acta de reunión

En el trabajo se explicó que se desarrolló un flujo de funcionalidades encadenadas encargado de extraer los nombres de los participantes de la reunión a partir de la interfaz de

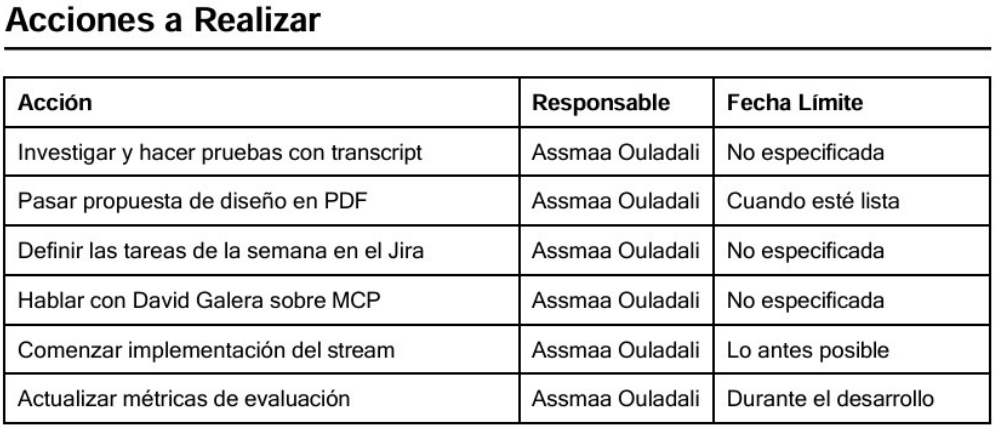
Microsoft Teams. A continuación, se presentan dos casos que evidencian que los nombres fueron extraídos correctamente. Además, en uno de ellos se incluye una tabla, lo que también demuestra la correcta generación de este tipo de elementos.

Acta de Reunión - Proyecto de Transcripción y Análisis de Reuniones

Información General

- **Participantes**:
- Assmaa Ouladali
- Jordi Sanchez
- Wan Li Sun Xu
- **Propósito**: Revisión del diseño de arquitectura y planificación del desarrollo de una herramienta de transcripción y análisis de reuniones

Figura 31. Sección del acta en la que se muestra el listado de participantes de la reunión, con los nombres de cada uno correctamente extraídos.



Acciones a Realizar

Acción	Responsable	Fecha Límite
Investigar y hacer pruebas con transcript	Assmaa Ouladali	No especificada
Pasar propuesta de diseño en PDF	Assmaa Ouladali	Cuando esté lista
Definir las tareas de la semana en el Jira	Assmaa Ouladali	No especificada
Hablar con David Galera sobre MCP	Assmaa Ouladali	No especificada
Comenzar implementación del stream	Assmaa Ouladali	Lo antes posible
Actualizar métricas de evaluación	Assmaa Ouladali	Durante el desarrollo

Figura 32. Sección del acta en la que se muestra una tabla con las tareas a realizar y el responsable de cada una, con su nombre correctamente extraído.

8.5.2 Muestra del correo enviado al usuario por Outlook

En esta sección se muestra el correo que se envía al usuario al finalizar el flujo, con el acta adjunta en PDF y un enlace que redirige al asistente inteligente AIDA, con la sesión creada y contextualizada automáticamente con la información de la reunión, permitiendo al usuario realizar preguntas de manera cómoda e interactiva sobre la reunión.

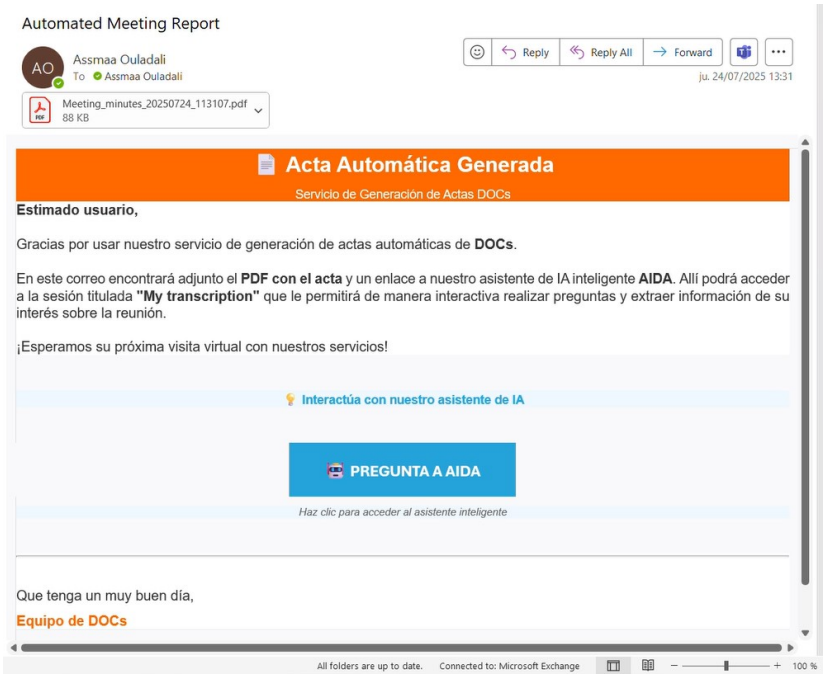


Figura 33. Correo enviado al usuario al finalizar el flujo con los resultados

8.5.3 Muestras de interacción con el asistente AIDA sobre la reunión

En esta sección se muestran algunas de las preguntas realizadas al asistente inteligente AIDA en la sesión ya preparada con el contexto de la reunión, donde se puede observar cómo responde de manera correcta.

Es importante señalar que algunas respuestas se muestran parcialmente por su extensión. No obstante, los fragmentos visibles son suficientes para evidenciar que el sistema funciona correctamente.

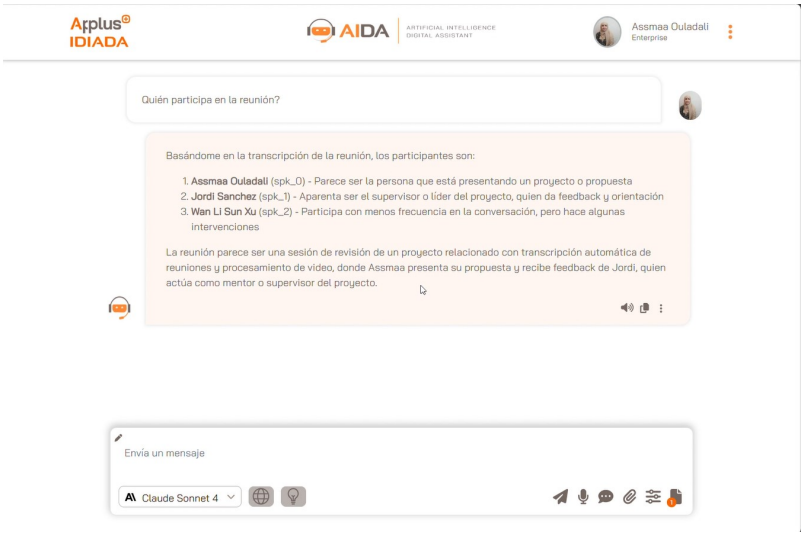


Figura 34. Ejemplo de interacción AIDA – Sesión contextualizada 1

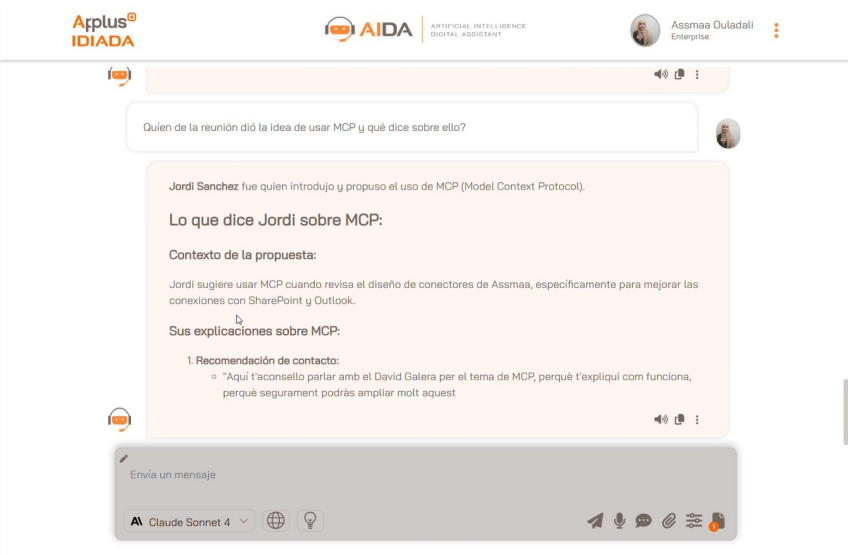


Figura 35. Ejemplo de interacción AIDA – Sesión contextualizada 2

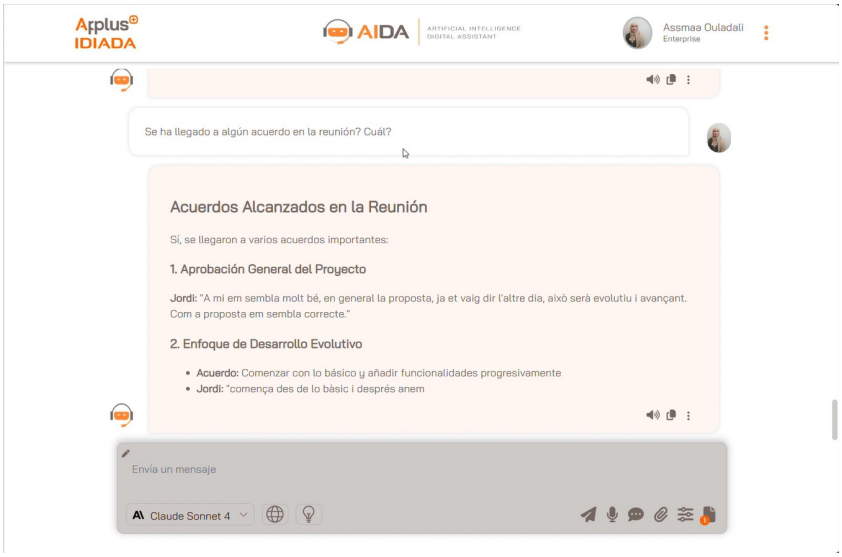


Figura 36. Ejemplo de interacción AIDA – Sesión contextualizada 3

9 Evaluación Comparativa de Tecnologías y Resultados de Transcripción

En este apartado se presenta el esquema de pruebas y las métricas de evaluación utilizadas para comparar las dos tecnologías aplicadas en el procesamiento multimedia: AWS y FFmpeg.

Además, se evaluarán los resultados de transcripción y diarización obtenidos con cada tecnología, considerando parámetros objetivos de calidad y rendimiento.

Finalmente, se mostrarán los resultados cuantitativos y cualitativos obtenidos, junto con las conclusiones derivadas del análisis comparativo.

El objetivo de esta evaluación es determinar la tecnología más adecuada para su adopción en el proyecto, de manera que la solución propuesta a la empresa sea lo más eficiente, precisa y rentable posible. Para ello, se realizará un análisis exhaustivo que permita tomar una decisión fundamentada en datos objetivos.

9.1 Justificación técnica de elecciones previas al análisis empírico

Antes de la ejecución de las pruebas comparativas y la validación empírica del sistema, se adoptaron ciertas decisiones sobre el uso de determinados parámetros, basadas en criterios técnicos y teóricos y en las propiedades intrínsecas de las tecnologías evaluadas. Estas elecciones se fundamentaron en documentación oficial, estándares reconocidos y referencias técnicas, sin depender de resultados experimentales previos dentro del presente proyecto.

Las decisiones son:

- **Tecnología de normalización:** Utilización de FFmpeg para la etapa de normalización de audio, motivada por la implementación del algoritmo EBU R128, frente al algoritmo ITU-R BS.1770 ofrecido por AWS, priorizando la precisión y el estándar de referencia en la industria audiovisual europea.
- **Formato de salida de audio:** Adopción del formato FLAC para los archivos de audio generados, considerando sus ventajas en calidad y compresión sin pérdida.

En las subsecciones siguientes, se expone de forma detallada el fundamento teórico de cada decisión.

9.1.1 Tecnología de normalización

Según la información recopilada durante la investigación realizada, se ha decidido optar por la normalización de audio mediante FFmpeg, aunque esto no descarta la evaluación de AWS como tecnología alternativa para realizar una comparación entre ambas.

La decisión final de usar FFmpeg se fundamenta en que permite emplear el algoritmo EBU R128, mientras que el servicio MediaConvert de AWS no lo ofrece. Para lograr una transcripción automática precisa, se requiere un audio con señal clara, estable y sin sobresaltos de volumen. Aunque ITU-R BS.1770 y EBU R128 comparten el mismo núcleo técnico, ITU-R BS.1770 representa una medición básica, mientras que EBU R128 constituye una evolución orientada a la práctica real del audio moderno [53].

En la Sección 2.1 de la fuente [54] se afirma que:

"*EBU R128 se basa en ITU-R BS.1770, pero introduce mejoras clave para entornos de producción reales: umbrales de 'gating' ajustados, ponderación de frecuencia optimizada, y métricas integradas (LRA) para manejar dinámica.*"

Por su parte, la fuente [55] menciona la capacidad de EBU R128 para combatir la sobrecompresión destructiva, afirmando que permite realizar una normalización basada en la sonoridad percibida, evitando la sobrecompresión que puede derivar de la normalización a nivel máximo utilizada por ITU. Como explica Florian Camerer, ingeniero de sonido de la ORF:

"La lucha por 'Quién es el más ruidoso' desaparece, las mezclas pueden ser más dinámicas, hay menos artefactos de compresión dinámica, como el 'bombeo', ¡y por lo tanto hay un aumento general de la calidad del audio!"

EBU R128 establece -23 LUFS como "el centro del universo de nivelación", basado en la percepción del volumen por el oído humano, lo que garantiza una mejor consistencia vocal y un control más efectivo del ruido de fondo.

Además, aparte del algoritmo, al ejecutar la normalización con AWS usando ITU-R BS.1770 se observa un tiempo de ejecución mayor en comparación con FFmpeg utilizando EBU R128, lo cual constituye un punto adicional a favor de FFmpeg.

9.1.2 Formato de salida del audio

Para este trabajo, se ha optado por el formato de audio que se considera que ofrece la mejor información para realizar la transcripción de manera precisa y eficiente, según las diferentes fuentes académicas y tecnologías de transcripción disponibles en el mercado.

En general, para las tecnologías ASR se recomienda usar formatos de audio sin compresión (WAV) o con compresión sin pérdida (FLAC), mientras que los formatos con pérdida, como MP3 o AAC, reducen la precisión del reconocimiento de voz.

Se ha decidido utilizar FLAC debido a que es una codificación sin pérdida, lo que garantiza que no se pierda información de audio durante la compresión, a diferencia de formatos con pérdida como MP3, que pueden afectar la exactitud del reconocimiento de voz y, por ende, derivar en transcripciones y diarizaciones erróneas [56].

¿Por qué FLAC y no WAV, si ambos son sin pérdida? Existen dos razones principales:

1. **Eficiencia en el tamaño de archivo:** Aunque ambos formatos son sin pérdida y ofrecen la misma calidad de audio, FLAC emplea una compresión inteligente que reduce significativamente el tamaño del archivo. Como se menciona en la fuente: *"Si el tamaño de archivo o la transmisión de datos son importantes para ti, elige FLAC como tu opción de codificación de audio"* [56]. Esto permite una mayor eficiencia en almacenamiento y procesamiento.
2. **Consistencia del formato:** WAV es únicamente un contenedor que puede emplear diferentes codificaciones internas. La fuente advierte que: *"Los archivos de audio .WAV a menudo (pero no siempre) usan una codificación PCM lineal; no supongas que un archivo .WAV tiene una codificación particular hasta inspeccionar su encabezado"* [56]. Con FLAC, se garantiza que siempre se utilizará una codificación sin pérdida específica, asegurando consistencia en la calidad del audio procesado.

9.2 Diseño experimental y métricas

En este apartado se presentan las pruebas diseñadas para evaluar y comparar las tecnologías implementadas para procesamiento de audio y video, así como su efecto sobre la

transcripción y diarización. Se estudiarán distintas métricas de rendimiento, coste y uso de recursos, considerando variables como la duración, la resolución y la calidad de los archivos de entrada. El objetivo es analizar de manera controlada cómo cada tecnología responde a estas variaciones y determinar su eficiencia y precisión en escenarios representativos de uso real.

9.2.1 Datasets de pruebas

En la tabla inferior se muestran las características que variarán para hacer los estudios y las operaciones que afecta cada característica.

Lista de operaciones a evaluar:

1. Reducción de ruido
2. Normalización de audio
3. Extracción del audio de un video
4. Extracción de un frame de un video
5. Detección de cambios de escena

Característica	Variaciones	Propósito	Operación que se ve afectada
Duración	30 seg, 2 min, 5 min, 15 min, 30 min	Evaluar escalabilidad temporal	1, 2, 3, 4, 5
Resoluciones	720p, 1080p, 4K	Impacto en procesamiento	3, 4, 5
Calidad Audio	Limpio y con ruido de fondo	Robustez procesamiento	Estos archivos se usan para evaluar la transcripción y diarización

Tabla 18. Características que se probará su efecto sobre las operaciones de procesamiento multimedia

9.2.1.1 Tecnologías de edición

Para realizar las pruebas presentadas, es necesario contar con videos y audios de entrada con distintas duraciones, resoluciones y formatos. Para obtener estos datos de prueba, se utilizó la herramienta de *Microsoft Clipchamp*, que permite la edición de video y audio, asegurando así que las entradas sean exactamente las requeridas.

Esta herramienta se empleó para recortar los videos a la duración precisa necesaria para cada prueba, así como para garantizar que los videos mantuvieran la misma resolución cuando la métrica estudiada no dependía de ella, o para variar la resolución cuando ésta era el objeto de estudio.

Clipchamp ha resultado fundamental para preparar los datos de entrada con el control necesario, minimizando al máximo la influencia de factores externos o de terceros que puedan afectar la obtención de resultados precisos en las diferentes métricas.

Durante el proceso, al crear videos de diferentes resoluciones, noté que los videos de 720p y 4K tenían un tamaño mucho mayor que los de 1080p, aunque todos dispongan de la

misma duración. Esto parecía anómalo, por lo que investigué la causa utilizando un analizador de metadatos en línea *MediaInfoOnline* [57]. Los resultados fueron los siguientes:

Especificación	1080p	720p	4K
Tamaño	16.3 MB	34.9 MB	133 MB
Bitrate total	456 kb/s	975 kb/s	3,721 kb/s
Bitrate video	260 kb/s	776 kb/s	3,523 kb/s
Frame rate	16 FPS	30 FPS	30 FPS

Tabla 19. Resultados del análisis de los videos con diferentes resoluciones

El análisis mostró que el tamaño más grande del video de 720p se debía principalmente a:

- Bitrate de video: 720p tiene 776 kb/s frente a 260 kb/s de 1080p, es decir, 3 veces más información por segundo.
- Frame rate: 720p tiene 30 FPS frente a 16 FPS de 1080p, casi el doble de cuadros por segundo.
- Configuración de compresión: Ambos usan H.264, pero el 720p estaba menos comprimido, lo que aumenta el tamaño.

Un cálculo rápido confirma la diferencia:

- 720p: 776 kb/s × 30 FPS = 23,280 unidades de datos por segundo
- 1080p: 260 kb/s × 16 FPS = 4,160 unidades de datos por segundo

Esto significa que el 720p procesa 5.6 veces más datos por segundo que el 1080p, explicando su tamaño mayor.

En la tabla de resultados del análisis de los tres videos, aunque no se muestran los cálculos relacionados, se puede observar que el video en 4K también contiene una mayor cantidad de datos, lo que explica su extremadamente gran tamaño.

Para normalizar los archivos y poder compararlos de manera justa, desarrollé un código en Python utilizando FFmpeg que ajusta el bitrate y los FPS de los videos de 720p y 4K para que coincidan con los del video de 1080p:

```
ffmpeg -i 720.mp4 -r 16 -b:v 260k -b:a 192k -c:v libx264 -c:a aac 720_optimizado.mp4
```

Código 24. Comando de FFmpeg para unificar las características de videos con distintas resoluciones.

De esta forma, los datos de entrada se preparan con precisión, minimizando la influencia de factores externos o configuraciones de video que podrían afectar los resultados de las métricas estudiadas.

9.2.2 Métricas de evaluación

9.2.2.1 Rendimiento

Para evitar que otros factores influyan en el rendimiento que se evalúa, se han fijado los parámetros de reducción de ruido, normalización, detección de escena, entre otros, de manera

idéntica tanto en AWS como en FFmpeg, garantizando que estos elementos no afecten las medidas obtenidas.

Por ejemplo, dado que AWS solo puede trabajar con una frecuencia de muestreo mínima de 32000 Hz, FFmpeg también se ha configurado para usar el mismo valor.

Métrica	Explicación	Unidad
Tiempo de procesamiento	Tiempo dedicado a la ejecución efectiva de la operación.	segundos
Throughput	Cantidad de archivos por segundo que se procesan.	archivos/s
Latencia	Tiempo total de procesamiento, incluido todo el overhead.	segundos

Tabla 20. Métricas de rendimiento observadas

9.2.2.2 Coste

Para evaluar el coste, se presentan a continuación las tarifas de AWS correspondientes al uso de los distintos servicios empleados.

En el apartado de los resultados, se calculará el coste a pagar según los resultados que se obtienen.

En el siguiente detalle de costes, se va a considerar que la equivalencia entre el dólar estadunidense y el euro es el siguiente: **0.9€ = 1$**

Coste - Uso genérico AWS

La región en donde están ubicados tanto la EC2 como el S3 es en **Europa central, Frankfurt**, así que los precios que se detallan pertenecen a esta región.

El programa se ejecuta en una **EC2** de AWS, de tipo t3.small con 2 CPUs, 2GB de RAM y hasta 5 Gb de rendimiento de red. La tarifa por hora bajo demanda es de 0,024$.

Para tener una idea de la dimensión del precio, si se mantiene la EC2 encendida durante 24 horas, los 30 días del mes, que equivalen a 720 horas, tiene un costo de 17.28$.

En mi caso, todas las transferencias se realizan dentro de la misma región de AWS. Tanto las transferencias de entrada como de salida tienen un coste de 0,01 $/GB. Sin embargo, las transferencias de datos con S3 dentro de la misma región son gratuitas [58].

Para el almacenamiento, se debe contratar por separado el servicio denominado **EBS**. En el caso de la máquina que utilizo, cuenta con un volumen gp3 de 8 GB de capacidad. Este almacenamiento tiene un coste de 0,0952 $/GB-mes. El precio no es variable, ya que se paga por la cantidad de almacenamiento reservada o alquilada [59].

Por lo tanto, en este caso el coste es constante:

$$8GB * 0{,}0952\ \$/GB\text{-}mes = 0.7616\ \$/mes$$

$$\text{Coste en euros} = 0.6854\ €$$

En el caso de **S3** [60], el bucket que utilizo es de tipo Standard y, dado que no supera los 50 TB, el coste es de 0,0245 $/GB-mes ocupado. A diferencia del almacenamiento EBS, que se paga por capacidad reservada o alquilada, en S3 el coste se calcula en función del uso real. Esto significa que, a medida que se suben archivos, el gasto aumenta proporcionalmente.

Un aspecto importante a considerar es qué ocurre cuando se borran archivos antes de que finalice el mes. En esta implementación, los archivos se eliminan al final del proceso por eficiencia, lo que implica que no permanecen en S3 un mes completo. Sin embargo, AWS cobra por el tiempo que los archivos han permanecido almacenados, aunque sea parcial.

El cálculo del **coste se basa en el tamaño del archivo y el tiempo que permanece en S3**. Por ejemplo, si se sube un archivo de 1 GB y permanece en S3 durante 15 días, el coste se calcularía de la siguiente forma:

$$\text{Coste (1GB por 15 días)} = 1\text{GB} * (15 \text{ días})/(30 \text{ días}) = 0.5\text{GB/mes}$$

En este caso, se paga por haber tenido 1 GB ocupado durante 15 días, no por los 30 días del mes.

En nuestro caso, los archivos pueden permanecer en S3 durante períodos muy cortos, a veces solo segundos o minutos. En este caso, AWS cobra por unidad mínima de 1 hora. Esto significa que, si un archivo se sube y se borra en menos de 1 hora, se cobrará por 1 hora completa.

A parte de pagar el almacenamiento en S3, se debe pagar por las operaciones (PUT, LIST, GET…) que se realizan. Los precios de las operaciones de S3 son los siguientes:

- Subir archivo (PUT): 0,0054$ por 1000 solicitudes. Es decir $5.4*10^{-6}$ $/solicitud.
- Descargar archivo (GET): 0,00043$ por 1000 solicitudes. Es decir $4.3*10^{-7}$ $/ solicitud.
- Eliminar archivo (DELETE): 0,00043$ por 1000 solicitudes. Es decir $4.3*10^{-7}$ $/ solicitud.

En el caso de **SNS** y **SQS**, el uso de SNS es gratuito si las notificaciones se entregan a colas SQS, tal como se indica en la documentación oficial [61]. Por su parte, SQS se paga según la cantidad de solicitudes. Sin embargo, dado que el número de solicitudes en mi caso es muy bajo, inferior a un millón, el coste es cero [62].

Por lo tanto, el uso de estos servicios resulta gratuito y no genera costes adicionales en mi implementación

En la siguiente tabla se resumen todos los precios comentados arriba:

Servicio	Componente	Especificación	Precio USD	Precio EUR
EC2 t3.small	Instancia	2 CPUs, 2GB RAM	0.024$/hora	0.0216€/hora
			17.28$/720 horas/mes	15.552€/720 horas/mes
EBS gp3	Almacenamiento	8 GB reservados	0.0952$/GB/mes	0.0857€/GB/mes
S3 Standard	Almacenamiento	Hasta 50TB	0.0245$/GB/mes	0.0221€/GB/mes
	PUT Operations	Subir archivos	$0.0054/1000 solicitudes	0.00486€/1000 solicitudes
			$5.4*10^{-6}$ $/solicitud	$4.86*10^{-6}$ €/solicitud

Operaciones S3	GET Operations	Descargar archivos	0.00043$/1000 solicitudes	0.000387€/1000 solicitudes
			$4.3*10^{-7}$ $/ solicitud	$3.87*10^{-7}$ €/ solicitud
	DELETE Operations	Eliminar archivos	0.00043$/1000 solicitudes	0.000387€/1000 solicitudes
			$4.3*10^{-7}$ $/ solicitud	$3.87*10^{-7}$ €/ solicitud
Data Transfer	EC2 ↔ S3	Misma región	GRATIS	GRATIS
	Otras transferencias de datos	Dentro región	0.01$/GB	0.009€/GB
SNS y SQS	Uso Gratuito en este caso			

Tabla 21. Resumen costes de uso genérico de AWS

Dado que el coste mensual del EBS es el mismo si se usa FFmpeg como si se usa AWS, no se sumará en los resultados de coste, ya que en este análisis se evalúa únicamente el coste de una operación con cada tecnología. El coste del EBS no depende del número de operaciones realizadas y, además, no varía entre una tecnología y otra.

Coste – Servicios específicos AWS

El uso de los servicios AWS MediaConvert y Rekognition para las funciones estudiadas también conlleva un coste asociado.

En el caso de MediaConvert, el precio varía según la codificación, la resolución y los fotogramas por segundo (fps) del video, pudiendo aumentar considerablemente. Si se procesa únicamente audio, el coste corresponde al precio base multiplicado por 0.4, lo que lo hace más económico que el procesamiento de video.

En mi caso, se priorizan los videos de grabaciones de reuniones de Teams, por lo que los cálculos de coste se basarán en este tipo de archivos.

Los videos descargados de Teams tienen las siguientes características:

- Codificación: AVC
- Resolución: 1080p, es decir HD
- Frames por segundo (fps): 16.00 fps

Los datos del video se pueden obtener fácilmente mediante la herramienta *MediaInfo* [57], que puede instalarse o utilizarse en su versión online, subiendo el video del cual se desea obtener la información.

Dado que en este estudio también se analiza el efecto de modificar la resolución, se detallan los precios para diferentes resoluciones y no solo para HD.

El precio base del servicio MediaConvert es de 0,0085 $/minuto al mes, el cual debe multiplicarse por un factor que varía según la codificación, la resolución y los fps, para obtener el precio final [63].

Resolución	Factor de multiplicación	Precio $/minuto/mes	Precio €/minuto/mes
SD (720p)	X1	0,0085	0.00765
HD (1080p)	X2	0.017	0.0153
4K	X4	0.034	0.0306

Tabla 22. Coste del procesamiento video con AWS MediaConvert según la resolución

Mientras que el precio del procesamiento de audio es el siguiente:

	Factor de multiplicación	Precio $/minuto/mes	Precio €/minuto/mes
audio	X0.4	0.0034	0.0031

Tabla 23. Coste del procesamiento audio con AWS MediaConvert

En el caso de Amazon Rekognition, que se utiliza para detectar cambios de escena y determinar cuándo se comparte la pantalla, se emplea el servicio de *detección de tomas*, cuyo coste es de 0,06 $/min [64].

Servicio	Función	Precio $/min	Precio €/min
Rekognition (Video Analysis)	Segment Detection, Detección de tomas	0.06	0.054

Tabla 24. Coste de detectar cambios de escena mediante Amazon Rekognition

9.2.2.3 Recursos

Las métricas de los recursos utilizados que se evaluarán son las siguientes:

Métrica	Explicación	Unidad
CPU/GPU	% CPU usada (FFmpeg)	% utilización
RAM	RAM usada (FFmpeg)	MB
Almacenamiento	Temporal local (FFmpeg) y temporal S3 (AWS)	MB

Tabla 25. Métricas de recursos

En el caso de los servicios de AWS, estos se ejecutan en la nube y no de manera local, lo que significa que el uso de CPU y RAM locales durante el procesamiento es mínimo, limitado únicamente a realizar llamadas a la API o a recopilar los resultados al final de la operación. Razón por la cual estas métricas, memoria y CPU, solo se observan en FFmpeg.

No fue posible monitorear estas métricas directamente en la consola de la nube, ya que los recursos del entorno de AWS están compartidos y continuamente reciben peticiones de otros procesos del sistema de DOCs, lo que genera datos 'contaminados'. Además, el consumo de recursos en la nube no impacta directamente en el rendimiento de mi sistema local, por lo que no es relevante para el análisis.

9.2.3 Metodología de ejecución

Para garantizar que las mediciones de rendimiento y recursos sean precisas y representativas, se ha definido una metodología estructurada dividida en varias fases. Cada fase tiene un objetivo específico y establece condiciones controladas para minimizar la variabilidad y asegurar resultados consistentes. La siguiente tabla resume estas fases, la configuración utilizada y el tipo de medición realizada:

Fase	Descripción	Configuración	Medición
Preparación	Entorno controlado	Hardware fijo, configuración optimizada	Baseline establecido
Warmup	5 ejecuciones previas	Cache calentado, estado estable	Descartar resultados
Medición	10 repeticiones por caso	Condiciones idénticas	Media

Tabla 26. Metodología de ejecución de las pruebas

El warmup es un concepto clave en benchmarking para obtener mediciones más precisas y confiables. Se refiere al período de "calentamiento" en el que el sistema se ejecuta varias veces antes de recolectar datos, permitiendo que alcance un estado estable y representativo.

Entre los factores que se ven afectados por el warmup se incluyen la caché del sistema operativo, la carga de librerías (como FFmpeg en este estudio) y el establecimiento de conexiones de red. Para minimizar la influencia de estos factores en los resultados finales, se realizaron primero cinco ejecuciones preliminares sin registrar datos, seguidas de diez repeticiones en las que sí se recopilaron las métricas necesarias para el análisis.

9.2.4 Evaluación impacto de transcripción y diarización

En este apartado se va a evaluar cómo afectan los diferentes parámetros de configuración del procesamiento de audio en la transcripción y diarización.

9.2.4.1 FFmpeg vs AWS

Se va a evaluar si se obtienen mejores resultados de transcripción y diarización según si se usa AWS o FFmpeg para el procesamiento audio.

Pipeline	Procesamiento	Transcripción	Métrica
Baseline	Audio Original	Transcripción directa	WER de referencia
Pipeline A	FFmpeg completo	Audio procesado	WER vs baseline
Pipeline B	AWS completo	Audio procesado	WER vs baseline
Análisis	Diferencias específicas	Casos de estudio	WER, CER, tiempo y precisión de diarización

Tabla 27. Comparación y análisi del pipeline con FFmpeg y AWS

El WER o tasa de error de palabras, es una métrica que indica el porcentaje de palabras transcritas incorrectamente en comparación con una referencia correcta. Se calcula

considerando las inserciones, borrados y sustituciones necesarias para transformar la transcripción automática en la referencia. Por ejemplo, un WER de 0,1 significa que aproximadamente el 10% de las palabras transcritas difieren de la transcripción de referencia. Lo que significa que el 90% coincide con la referencia.

El CER (Character Error Rate) o tasa de error de caracteres, es similar al WER, pero se aplica a nivel de caracteres en lugar de palabras. Mide el porcentaje de caracteres incorrectos en la transcripción respecto a la referencia, considerando inserciones, borrados y sustituciones de caracteres individuales. Esta métrica es útil para idiomas con palabras cortas o para evaluar errores menores dentro de palabras, ofreciendo un detalle más fino que el WER. Además, un CER suele ser menor que el WER.

También se evalúa la precisión de la diarización, verificando si los segmentos de cada hablante coinciden con los de referencia.

En la evaluación de la diarización, cada segmento de habla tiene un tiempo de inicio y un tiempo de fin tanto en la referencia como en la transcripción automática. Para determinar si un segmento de un hablante se asigna correctamente, se utiliza un intervalo de tolerancia temporal, que define cuánto puede desplazarse el inicio o el fin del segmento sin considerarlo incorrecto.

Si se establece una tolerancia pequeña, solo se aceptan coincidencias muy precisas, lo que refleja la alta precisión temporal del sistema. En cambio, al aumentar la tolerancia, se permite que los segmentos coincidan incluso cuando hay un pequeño desfase entre la referencia y la transcripción automática.

Sin embargo, una tolerancia demasiado alta puede aumentar los errores. Esto ocurre porque la evaluación compara de manera estricta cada tramo de un hablante. Por ejemplo, si un segmento de “Hablante 2” empieza después de un tramo de “Hablante 0”, y la tolerancia es muy grande, el sistema puede considerar que el tramo de Hablante 2 se superpone con el de Hablante 0, dando como resultado una coincidencia incorrecta. Este efecto produce falsos positivos en la asignación de hablantes, aumentando el error global de diarización.

En resumen, la tolerancia temporal debe elegirse cuidadosamente: Una tolerancia baja refleja precisión estricta, mientras que una tolerancia demasiado alta puede inducir fallos debido a coincidencias erróneas entre segmentos adyacentes de distintos hablantes.

9.2.4.2 Parámetros de configuración a evaluar

Se van a variar los diferentes parámetros de configuración del procesamiento audio y se verá si afecta positiva o negativamente a la transcripción y diarización.

Optimización de Sample Rate (Frecuencia de muestreo)

ID	Sample Rate
SR-1	16000 Hz
SR-2	22050 Hz
SR-3	32000 Hz
SR-4	44100 Hz
SR-5	48000 Hz

Tabla 28. Valores de sample rate a probar

Configuración de Canales

Test ID	Canales	Uso Recomendado
CH-1	1 (mono)	Transcripción pura
CH-2	2 (estéreo)	Diarización con separación espacial

Tabla 29. Valores de número de canales a probar

Profundidad de Bits

Test ID	Bit Depth
BD-1	16 bits
BD-2	24 bits
BD-3	32 bits

Tabla 30. Valores de profundidad de bits a probar

Reducción de ruido

Para configurar los parámetros de reducción de ruido, primero se debe fijar el Highpass Frequency (Frecuencia de corte alta), mientras que los parámetros de Noise Reduction (Reducción de ruido) y Noise Floor (Umbral de ruido) se mantienen constantes.

Luego, se varía únicamente el Noise Reduction (Reducción de ruido) y, finalmente, el Noise Floor (Umbral de ruido), manteniendo siempre fijos los parámetros que no se están estudiando y modificando solo el que se analiza.

Parámetro	Mínimo	Máximo
Highpass Frequency	20 Hz	1000 Hz
Noise Reduction	1 dB	50 dB
Noise Floor	-80 dB	-20 dB

Tabla 31. Valor mínimo y máximo de los parámetros de reducción de ruido

Normalización

Después de fijar los parámetros de reducción de ruido, se configuran los parámetros de normalización del audio. Primero se fija el I Value (Integrated Loudness, Loudness Integrada), que define el nivel promedio de loudness del audio a lo largo de todo el archivo. Mientras tanto, los otros parámetros, TP Value (True Peak, Pico Verdadero) y LRA (Loudness Range, Rango de Loudness), se mantienen constantes.

A continuación, se varía únicamente el TP Value (Pico Verdadero), que limita los picos máximos de la señal para evitar distorsión, manteniendo fijos los demás parámetros. Finalmente, se ajusta el LRA (Rango de Loudness), que controla la amplitud dinámica percibida del audio, asegurando que los cambios se estudien de manera independiente y que los demás valores permanezcan constantes.

Parámetro	Mínimo	Máximo
I_Value	-70 LUFS	-5 LUFS

TP_Value	-9 dBTP	0 dBTP
LRA	1 LU	50 LU

Tabla 32. Valor mínimo y máximo de los parámetros de normalización

9.2.4.3 Dataset de datos

Para realizar las pruebas sobre cómo afectan los diferentes parámetros mencionados a la transcripción y la diarización, se han extraído grabaciones de reuniones del dataset *AMI Corpus*, que cuenta con 100 horas de reuniones, incluyendo la transcripción de cada una y la diarización [65].

El hecho de que el dataset incluya tanto transcripción como diarización facilita la evaluación de la calidad de la diarización y transcripciones obtenidas por mi sistema.

De este dataset se han seleccionado grabaciones de reuniones con y sin ruido, así como grabaciones realizadas con dos tipos diferentes de micrófonos:

- Micrófono de diadema (headset): Micrófono diseñado para colocarse en la cabeza con un brazo flexible que sitúa el micrófono cerca de la boca. Es más inmune al ruido ambiental debido a su proximidad a la fuente de sonido.
- Micrófono de solapa (lavalier o lapel mic): Micrófono que se sujeta a la ropa (cerca del pecho o cuello) mediante una pinza. Captura el sonido de forma más natural, pero puede ser más susceptible al ruido de roces o viento, lo que hace que las grabaciones con este tipo de micrófono contengan más ruido en general.

9.2.5 Implementación de las pruebas

9.2.5.1 Implementación de las pruebas de evaluación de métricas en AWS y FFmpeg

Librerías usadas

- **psutil**, para el monitoreo de recursos del sistema
- **FFmpeg**: Procesamiento multimedia local y obtención de métricas ofrecidas por la herramienta.
- **time**, para las mediciones temporales. Se usa time.perf_counter() para medir duración de operaciones y latencia.

Metodología

La metodología de evaluación se fundamenta en una medición integral que combina diferentes enfoques complementarios: La instrumentación directa para obtener métricas precisas del sistema, el monitoreo pasivo de recursos para evaluar el consumo y rendimiento en condiciones reales de ejecución, el análisis de las salidas generadas tanto por FFmpeg como por los servicios de AWS, la realización de pruebas de estrés para valorar la robustez del sistema y, finalmente, la comparación en múltiples entornos con el fin de identificar variaciones de comportamiento y asegurar la validez de los resultados.

Flujo de ejecución

Figura 37. Flujo de ejecución de los tests de FFmpeg y AWS

Implementación

Se ha creado la clase *PerformanceReporter*, especializada en la recolección y análisis de métricas. Además, se han replicado las clases de procesamiento de audio y vídeo, añadiendo en el código los puntos de medición que permiten capturar métricas, las cuales se almacenan y gestionan en el objeto *PerformanceReporter* para su posterior análisis.

Asimismo, se implementa una función central encargada de ejecutar las operaciones un número definido de veces, lo que permite realizar pruebas controladas. En total, se llevan a cabo tres tipos principales de tests:

- Un primer test orientado a la medición general de métricas, incluyendo las relacionadas con el uso de recursos.
- Un segundo test centrado exclusivamente en mediciones temporales, tales como overhead, tiempo de entrada/salida y tiempo de procesamiento puro.
- Un último test diseñado para calcular de forma precisa el throughput. En este caso, se inicia un cronómetro, se ejecuta el procesamiento de n archivos de manera consecutiva y se mide el tiempo total al finalizar, sin registrar métricas intermedias (como recursos o tiempos parciales), lo que permite obtener una estimación exacta del rendimiento global.

9.2.5.2 Implementación de las pruebas de evaluación de transcripción y diarización

Librerías usadas

- **lxml:** Usada para parsear archivos XML con las transcripciones de *AMI Corpus*.
- **jiwer:** Calcula el WER, el CER y permite aplicar transformaciones de texto.
- **textnorm:** Para normalización de texto (espacios, caracteres Unicode, etc.).
- **Otras:** json, re y collections.

Implementación

1. **Extracción y preparación de textos:** Se prepara las transcripciones de AMI y AWS.

   Para AMI, se parsea el archivo XML, con la transcripción, para extraer las palabras con su startTime, se ordenan cronológicamente y se combinan en un único texto. En cambio, para AWS, la transcripción se obtiene directamente del JSON de respuesta en formato completo, sin necesidad de procesamiento adicional.

2. **Normalización de textos:** Antes de comparar, los textos se normalizan para reducir errores artificiales por mayúsculas, acentos o puntuación.

   Esto asegura que, por ejemplo, "Hola," y "hola" se consideren iguales.

3. **Cálculo del WER y CER** utilizando la función dedicada de la librería jiwer.

A continuación, se resumen los pasos implementados para calcular la precisión de la diarización:

1. Se extraen las palabras con marcas de tiempo y speaker de AMI y AWS. Tener en cuenta que se sabe quién es el hablante en AMI corpus debido a que el dataset además de ofrecer las grabaciones de las reuniones ofrece una transcripción dividida en tantos fiches XML como hablantes haya en la reunión.
2. Se extraen las palabras junto con sus marcas de tiempo y el identificador del hablante tanto de AMI como de AWS. Cabe destacar que, en el caso de AMI, se conoce la identidad del hablante, ya que el dataset proporciona las grabaciones de las reuniones junto con la transcripción segmentada en múltiples archivos XML, uno por cada participante. En AWS, la identificación del hablante se realiza mediante etiquetas automáticas asignadas como spk_0, spk_1, etc.
3. Alineación temporal: Se organizan cronológicamente los segmentos de cada hablante para que coincidan con las marcas de tiempo de la transcripción de referencia.
4. Cuenta coincidencias: Calcula cuántas palabras o segmentos de cada hablante automático (los obtenidos por mi sistema) coinciden con cada hablante de referencia, generando un conteo de frecuencias.
5. Inferencia de mapeo de hablantes: Determina la correspondencia más probable entre hablantes automáticos y de referencia a partir de los conteos, asignando cada spk_X al hablante de referencia con más coincidencias.
6. Cálculo de precisión de diarización: Se calcula como la proporción de palabras correctamente asignadas a cada hablante respecto al total de palabras, reflejando la exactitud de la diarización.

9.3 Resultados

A continuación, se presentan los resultados obtenidos al ejecutar las pruebas planificadas y descritas en el apartado anterior.

9.3.1 Métricas de evaluación

En este apartado se presentan los resultados obtenidos al variar las características técnicas mencionadas anteriormente, utilizando tanto AWS como FFmpeg.

Los resultados se organizan por operación de procesamiento multimedia, mostrando para cada una de ellas las métricas de rendimiento, coste y uso de recursos.

Definición de métricas de rendimiento

Para una interpretación precisa de las tablas de resultados, se emplean las siguientes definiciones:

- Tiempo de Procesamiento: Tiempo dedicado exclusivamente a la ejecución de la tarea principal, sin considerar latencias adicionales (sin overhead).
- Latencia Total: Tiempo de Procesamiento más el overhead asociado, incluyendo el tiempo de entrada/salida (I/O).

  Latencia Total = Tiempo Procesamiento + Overhead (incluye tiempo I/O)

- Overhead: Diferencia entre la Latencia Total y el Tiempo de Procesamiento. Incorpora el tiempo de I/O y cualquier otro retardo no propio de la operación principal.

  Overhead (incluye tiempo I/O) = Latencia Total – Tiempo procesamiento

- Eficiencia (%): Relación entre el Tiempo de Procesamiento y la Latencia Total, expresada en porcentaje:

  Eficiencia (%) = (Tiempo Procesamiento / Latencia Total) × 100

9.3.1.1 Tiempo de I/O y overhead

Antes de presentar los resultados, es necesario aclarar un aspecto metodológico clave: La clasificación del tiempo de E/S dentro del overhead. Tras evaluar definiciones técnicas y el comportamiento observado en los datos, se determinó que el tiempo de E/S debe considerarse componente del overhead.

Esta decisión se fundamenta en la definición canónica de overhead en computación: "Todo tiempo o recurso consumido que no contribuye directamente al cómputo útil del programa".

Las operaciones de E/S, incluyendo lectura/escritura de datos, acceso a disco, transferencia de red y esperas asociadas, no constituyen procesamiento activo, sino actividades auxiliares de transferencia o espera. Por lo tanto, califican conceptualmente como overhead.

No obstante, para cuantificar su impacto relativo, los resultados desglosan explícitamente el tiempo de E/S en una columna independiente que se podrá observar en los apartados a continuación.

9.3.1.2 Extracción de un audio de un video

A continuación, se muestran los resultados de la operación de extraer el audio de un video.

Rendimiento

A continuación, se muestra la variación de los valores de las métricas de rendimiento según la duración del video, y consiguientemente del tamaño, y de la resolución.

Duración	FFmpeg (Throughput)	AWS (Throughput)	Factor de Mejora FFmpeg
30 segundos	10.39 archivos/seg	0.14 archivos/seg	74.2x más rápido
2 minutos	4.93 archivos/seg	0.067 archivos/seg	73.6x más rápido
5 minutos	2.61 archivos/seg	0.049 archivos/seg	53.3x más rápido
15 minutos	0.90 archivos/seg	0.015 archivos/seg	60.0x más rápido
30 minutos	0.46 archivos/seg	0.004 archivos/seg	115.0x más rápido

Tabla 33. Extracción del audio de un video: Resultados de throughput y factor de mejora variando la duración del video

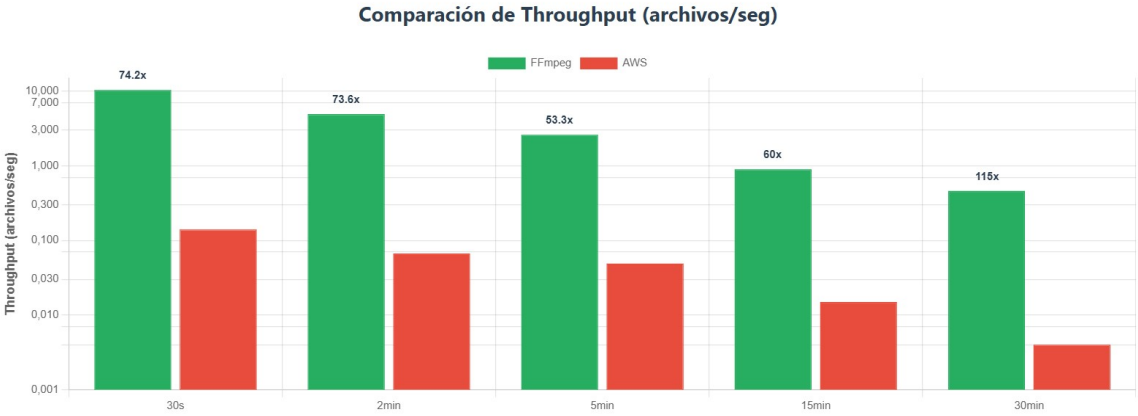


Figura 38. Extracción de audio de un video: Resultados de throughput y factor de mejora variando la duración del video

El throughput (archivos procesados por segundo) muestra un comportamiento esperado: A medida que aumenta la duración de los archivos, y por consiguiente su tamaño, el número de archivos que pueden procesarse por segundo disminuye, debido al mayor coste temporal asociado al procesamiento de un volumen de datos superior.

El factor de mejora presenta un patrón bifásico: Inicialmente decrece y posteriormente comienza a aumentar (74x → 53x → 115x), lo que refleja directamente las diferencias de escalabilidad entre ambas tecnologías. FFmpeg mantiene un crecimiento lineal y predecible de la latencia, mientras que en AWS este crecimiento es exponencial: Al inicio la latencia aumenta de forma más lenta, pero a medida que crece la duración del video, la latencia se incrementa de manera mucho más agresiva. Este comportamiento puede observarse en las gráficas de latencia que se presentan más abajo.

Resolución	FFmpeg (Throughput)	AWS (Throughput)	Factor de Mejora FFmpeg
720p	2,03 archivos/seg	0,02 archivos/seg	101,5x más rápido
1080p	2,37 archivos/seg	0,02 archivos/seg	118,5x más rápido
4K	1,78 archivos/seg	0,02 archivos/seg	89,0x más rápido

Tabla 34. Extracción del audio de un video: Resultados de throughput y factor de mejora variando la resolución del video

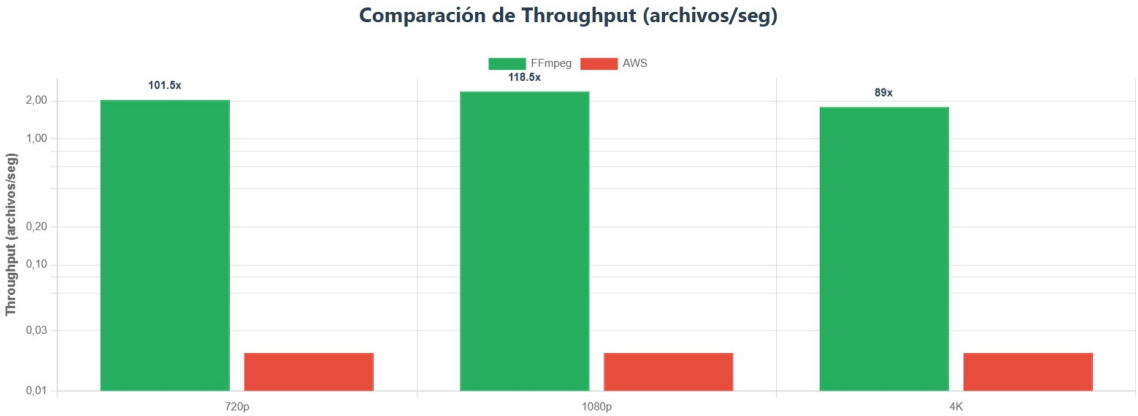


Figura 39. Extracción del audio de un video: Resultados de throughput y factor de mejora variando la resolución del video

Es lógico que el throughput de FFmpeg sea significativamente mayor que AWS porque la extracción local de audio evita los cuellos de botella de red y virtualización, concentrando toda la potencia de procesamiento directamente en la tarea; Aunque los archivos tengan tamaños iguales (5 minutos), las resoluciones más altas (1080p y 4K) requieren mayor potencia de decodificación de video para acceder al flujo de audio, lo que explica las variaciones en el rendimiento (pico en 1080p, caída en 4K), mientras que AWS muestra un throughput constante mínimo (0.02 arch/seg) debido al overhead fijo de transferencia, inicialización de servicios y limitaciones de recursos compartidos en la nube.

Si se compara la variación del throughput con la resolución y con la duración, en esta operación de extracción de audio de un video, se puede afirmar que la duración del video, y por consiguiente el tamaño, impacta más fuertemente en el throughput que la resolución, especialmente en FFmpeg, debido a que AWS se ve limitada por el overhead del tiempo de I/O.

Duración	Tecnología	Latencia Total (s)	Tiempo Procesamiento (s)	Tiempo I/O (s)	Overhead (s)	Eficiencia Procesamiento (%)
30 seg	FFmpeg	0.1077	0,1054	0,0017	0.0023	97.84%
	AWS	7.3199	4.4133	3.3063	3.3312	60.29%
2 min	FFmpeg	0.2251	0.2189	0.006	0.0062	97.23%
	AWS	16.1455	7.1395	9.0060	9.4596	44.2196%
5 min	FFmpeg	0.3829	0.3703	0.0124	0.0127	96.69%
	AWS	20.7764	6.6756	14.1008	14.5870	32.13%
15 min	FFmpeg	1.1150	1.0712	0.04347	0.0438	96.07%
	AWS	69.8311	11.9623	57.8688	58.3631	17.13%
30 min	FFmpeg	2.1736	2.0373	0.0461	0.1359	93.73%
	AWS	347.9219	21.4246	326.4973	327.0455	6.16%

Tabla 35. Extracción del audio de un video: Resultados de métricas de rendimiento variando la duración del video

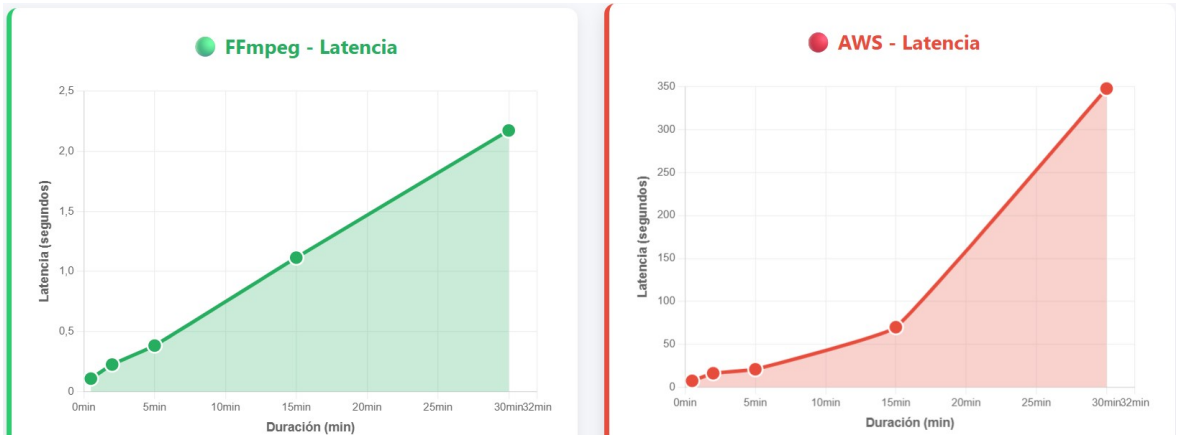


Figura 40. Extracción del audio de un video: Crecimiento de la latencia total variando la duración del video

En las gráficas superiores se puede observar claramente el crecimiento de la latencia tanto en FFmpeg como en AWS respecto a la duración. La latencia de FFmpeg muestra un comportamiento lineal, alcanzando un máximo de 2,17s, mientras que AWS presenta un crecimiento exponencial, con una latencia máxima de 347,92s (5 min y 48 segundos). Este crecimiento exponencial resulta especialmente crítico, ya que aumenta de manera muy rápida, haciendo que el uso de AWS sea significativamente menos eficiente en comparación con FFmpeg.

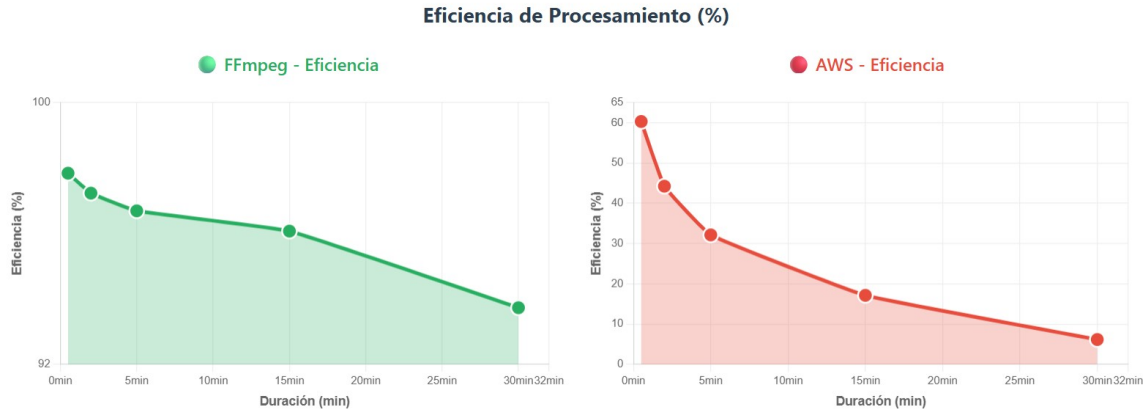


Figura 41. Extracción del audio de un video: Crecimiento de la eficiencia variando la duración del video

Estas gráficas de eficiencia respecto a la duración muestran claramente que la disminución de eficiencia es mucho más pronunciada en AWS, que presenta un decrecimiento exponencial, mientras que en FFmpeg la reducción tiende a ser lineal. Estos resultados son coherentes con el crecimiento lineal de la latencia en FFmpeg y el crecimiento exponencial observado en AWS.

Resolución	Tecnología	Latencia Total (s)	Tiempo Procesamiento (s)	Tiempo I/O (s)	Overhead (s)	Eficiencia Procesamiento (%)
720p	FFmpeg	0,521	0,518	0,014	0,014	97,3%
	AWS	47,179	7,0	40,179	40,565	14,8%
1080p	FFmpeg	0,462	0,467	0,013	0,015	96,8%
	AWS	39,6	6,95	32,65	32,67	17,5%
4K	FFmpeg	0,571	0,505	0,012	0,013	88,5%
	AWS	60,0	7,0	53,0	53,015	11,7%

Tabla 36. Extracción del audio de un video: Resultados de métricas de rendimiento variando la resolución del video

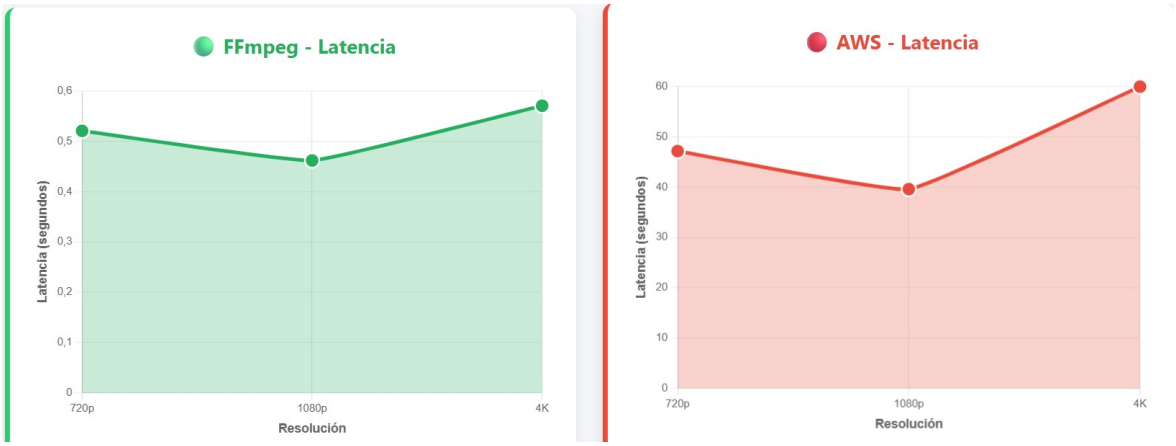


Figura 42. Extracción del audio de un video: Crecimiento de la latencia total variando la resolución del video

FFmpeg muestra un patrón no lineal en la latencia. A 1080p, la latencia es menor, posiblemente porque el uso de CPU es más eficiente y el video de 1080p ocupa menos espacio que el de 720p, lo que permite un mejor aprovechamiento de los recursos disponibles. Estos datos de uso de CPU y almacenamiento se presentan en la Tabla 38. Extracción del audio de un video: Resultado de métricas de recursos variando la resolución del video. En la realidad, la variación de latencia es relativamente pequeña. Las aparentes fluctuaciones en la gráfica se deben a que se están midiendo tiempos muy pequeños.

En 4K, la latencia aumenta debido a que, en muchos casos, es necesario decodificar el video para acceder al audio. Al ser la resolución 4K significativamente mayor, el tiempo de procesamiento se incrementa.

En AWS, el tiempo de procesamiento puro se mantiene prácticamente constante, alrededor de 7 s. La principal variación proviene del tiempo de I/O, que depende del tamaño de los datos, y de la velocidad de la red.

Por tanto, se puede concluir que, tanto en AWS como con FFmpeg, la resolución no afecta directamente, o tiene un efecto limitado, sobre la latencia de procesamiento al extraer el audio de un video.

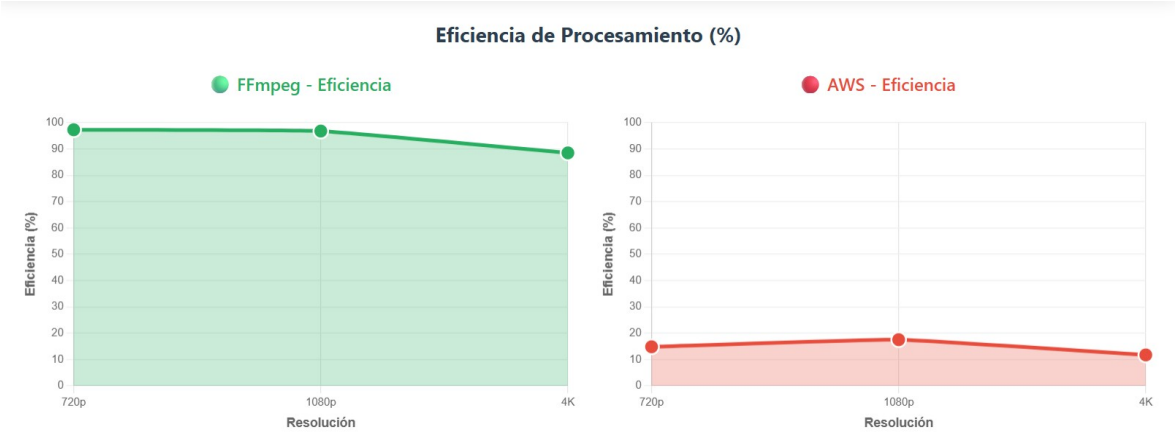


Figura 43. Extracción del audio de un video: Crecimiento de la eficiencia variando la resolución del video

Las gráficas de eficiencia muestran un comportamiento inverso al de la latencia, tal como es de esperar.

La eficiencia observada al variar la resolución de los videos en AWS es considerablemente menor que la obtenida al variar la duración debido a que, en este caso, el video de 5 minutos tiene un tamaño significativamente mayor que el utilizado en el estudio de la duración. Este mayor tamaño incrementa el costo asociado a la transferencia de datos (I/O), lo que provoca que el tiempo de entrada/salida represente la mayor parte de la latencia total. Como resultado, la proporción de tiempo dedicada al procesamiento efectivo es muy pequeña, reduciendo así la eficiencia global del sistema.

A continuación, se muestran las gráficas del desglose de la latencia en tiempo de procesamiento, overhead asociado a operaciones de entrada/salida y overhead de otras operaciones, respecto a la duración y la resolución.

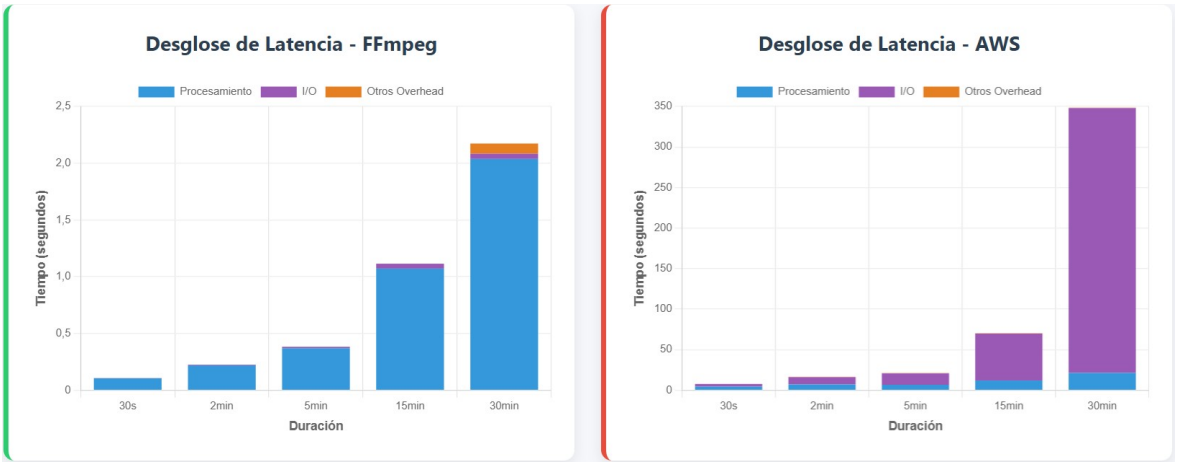


Figura 44. Extracción del audio de un video: Desglose de la latencia variando la duración del video.

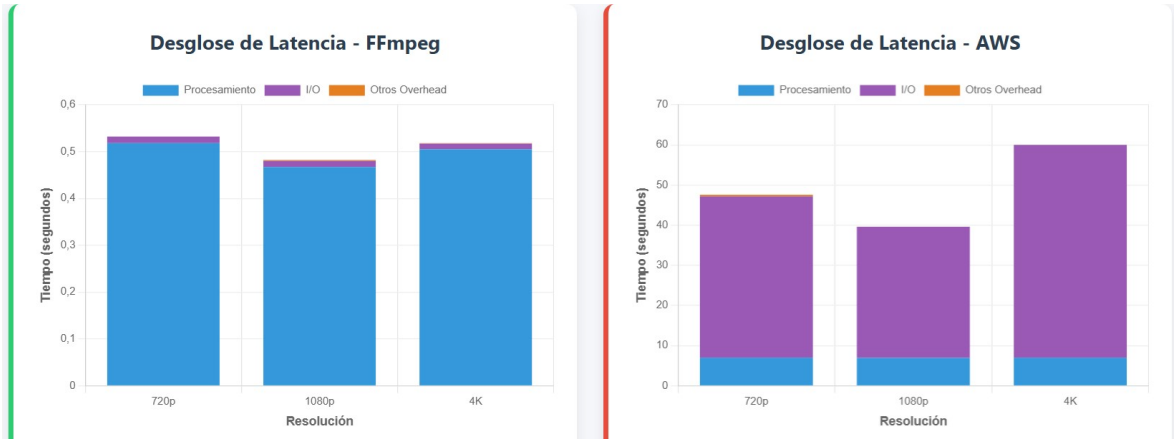


Figura 45. Extracción del audio de un video: Desglose de la latencia variando la resolución del video.

En el caso de FFmpeg, queda claro que la mayor parte de la latencia corresponde al tiempo de procesamiento puro, mientras que en AWS la mayor proporción se debe al tiempo de entrada y salida, lo que explica la baja eficiencia de esta opción.

Recursos

Duración	Uso_CPU (%)	Uso_RAM (%)	Almacenamiento (MB)
30 seg	198.51	23.07	2.18

2 min	196.75	29.10	7.68
5 min	196.95	29.59	16.33
15 min	190.46	33.77	48.82
30 min	199.32	36.11	207.24

Tabla 37. Extracción del audio de un video: Resultado de métricas de recursos variando la duración del video

Resolución	Uso_CPU (%)	Uso_RAM (%)	Almacenamiento (MB)
720p	153,74	30,28	16,24
1080p	185,36	29,79	16,33
4K	196,33	30,37	15.62

Tabla 38. Extracción del audio de un video: Resultado de métricas de recursos variando la resolución del video

En el caso de FFmpeg, el uso de recursos reportado corresponde exclusivamente al consumo generado por el procesamiento multimedia, y no al de toda la función del sistema. Esto se debe a que el objetivo es conocer específicamente los recursos que FFmpeg emplea como herramienta en la ejecución de dichas tareas.

En cambio, para AWS, el procesamiento multimedia se lleva a cabo íntegramente en la nube, por lo que no existe consumo de recursos locales. En este escenario, el sistema únicamente configura el trabajo y lo envía a la nube, razón por la cual no tiene sentido realizar una medición del uso de recursos en el lado de AWS.

Coste – FFmpeg

En el caso de FFmpeg, **el único coste asociado proviene del uso de la instancia EC2** y del almacenamiento EBS, cuyo alquiler se paga mensualmente. No existen costes adicionales por operaciones de lectura y escritura locales en el almacenamiento, ni por el uso de RAM o CPU, ya que el cobro se basa únicamente en el tiempo que la instancia EC2 permanece encendida para ejecutar el programa.

El tiempo de uso facturable de la EC2 coincide con la Latencia Total, es decir, el tiempo que toma ejecutar la función incluyendo todas las operaciones implicadas y el overhead.

Cabe recordar que el precio de la EC2 es de 0,0216 € por hora, lo que equivale aproximadamente a 6×10^{-6} € por segundo.

Duración	Latencia Total	Precio total
30 seg	0,1077 s	$6{,}42\times10^{-7}$ €
2 min	0,2251 s	$1{,}351\times10^{-6}$ €
5 min	0,3829 s	$2{,}297\times10^{-6}$ €
15 min	1,1150 s	$6{,}69\times10^{-6}$ €
30 min	2,1736 s	$1{,}304\times10^{-5}$ €

Tabla 39. Extracción del audio de un video: Resultado métricas de coste FFmpeg variando la duración del video

Resolución	Latencia Total	Precio total
720p	0,521	$3{,}13 \times 10^{-6}$ €
1080p	0,462	$2{,}77 \times 10^{-6}$ €
4K	0,571	$3{,}43 \times 10^{-6}$ €

Tabla 40. Extracción del audio de un video: Resultado métricas de coste FFmpeg variando la resolución del video

La tabla anterior muestra el coste de ejecutar la función según la duración del video.

Coste – AWS

En el caso de AWS, para la función de separar el audio del video, además del uso de EC2 y del almacenamiento, se paga por utilizar MediaConvert y S3.

Para el coste del uso de la EC2:

Duración	Latencia Total	Precio total
30 seg	7,3199 s	$4{,}39 \times 10^{-5}$ €
2 min	16,1455 s	$9{,}69 \times 10^{-5}$ €
5 min	20,7764 s	$1{,}25 \times 10^{-4}$ €
15 min	69,8311 s	$4{,}19 \times 10^{-4}$ €
30 min	347,9219 s	$2{,}09 \times 10^{-3}$ € ≈ 0,002 €

Tabla 41. Extracción del audio de un video: Resultados métricas de coste AWS variando la duración del video, EC2

Resolución	Latencia Total	Precio total
720p	47,179	$2{,}83 \times 10^{-4}$ €
1080p	39,6	$2{,}38 \times 10^{-4}$ €
4K	60,0	$3{,}60 \times 10^{-4}$ €

Tabla 42. Extracción del audio de un video: Resultados métricas de coste AWS variando la resolución del video, EC2

Para el coste del uso de S3:

Recordar:

- Precio almacenamiento por GB: Tal y como se indicó en el apartado de costes de AWS, el servicio S3 aplica un cargo mínimo equivalente a 1 hora de almacenamiento, incluso aunque los archivos no permanezcan ese tiempo. Dado que en nuestro caso ninguna operación supera la hora, siempre se facturará como si se hubiera utilizado una hora completa.

  El coste de 1GB por hora por mes es de: 0.0216€ * 1GB * (1 h)/(30 días) * (1 día)/(24 h) = $3*10^{-5}$ €

- Precio por una operación PUT: $4.86*10^{-6}$ €
- Precio de una operación GET y DELETE: $3.87*10^{-7}$ €
- Cantidad de operaciones PUT: 1 operación

- Cantidad de operaciones GET: 1 operación
- Cantidad de operaciones DELETE: 2 operaciones

El coste total del almacenamiento en S3 al mes de 1 ejecución de la función que separa el audio del video es el siguiente:

Duración	**Cantidad (GB)**	**Coste total almacenamiento (€)**
30 seg	0,002127886	$6{,}38366\times10^{-8}$
2 min	0,007504721	$2{,}25142\times10^{-7}$
5 min	0,01595129	$4{,}78539\times10^{-7}$
15 min	0,047673903	$1{,}43022\times10^{-6}$
30 min	0,202386894	$6{,}07161\times10^{-6}$

Tabla 43. Extracción del audio de un video: Resultados métricas de coste AWS variando la duración del video, almacenamiento S3

Resolución	**Cantidad (GB)**	**Coste total almacenamiento (€)**
720p	0,015860071	4.76×10^{-7}
1080p	0,01595129	4.79×10^{-7}
4K	0,015255184	4.58×10^{-7}

Tabla 44. Extracción del audio de un video: Resultados métricas de coste AWS variando la resolución del video, almacenamiento S3

El coste total de las operaciones de 1 ejecución de la función que separa el audio del video es el siguiente:

Cantidad PUTs	*Coste total PUT (€)*	*Cantidad GETs*	*Coste total GET (€)*	*Cantidad DELETEs*	*Coste total DELETE (€)*	*Coste total (€)*
1	4.86×10^{-6}	1	3.87×10^{-7}	2	7.74×10^{-7}	6.021×10^{-6}

Tabla 45. Extracción del audio de un video: Resultados métricas de coste AWS, operaciones S3

Para el coste del uso de MediaConvert:

Para el caso de MediaConvert, se ha trabajado con videos con resolución de 1080p, por lo que el precio por minuto y mes es de 0.0153€. A continuación, se muestra el coste del procesamiento con MediaConvert de una sola ejecución para cada duración.

Duración	**Coste total (€)**
30 seg	0,00765
2 min	0,0306
5 min	0,0765
15 min	0,2295

30 min	0,459

Tabla 46. Extracción del audio de un video: Resultados métricas de coste AWS variando la duración del video, MediaConvert

En MediaConvert, el precio varía en función de la resolución y la duración del vídeo. En los vídeos utilizados para el estudio del efecto de la resolución en las operaciones, la resolución varía, por lo que es necesario tenerlo en cuenta al calcular el coste.

Los costes por resolución son los siguientes:

- Coste por minuto al mes si se tiene una resolución de 720p: 0.00765 €
- Coste por minuto al mes si se tiene una resolución de 1080p: 0.0153 €
- Coste por minuto al mes si se tiene una resolución de 4K: 0.0306 €

Por lo tanto, los costes de cada vídeo de 5 minutos, según la resolución, son los siguientes:

Resolución	**Coste total (€)**
720p	0.03825
1080p	0,0765
4K	0.153

Tabla 47. Extracción del audio de un video: Resultados métricas de coste AWS variando la resolución del video, MediaConvert

A continuación, se muestra una tabla con el coste de una única ejecución de la función que separa el audio del video. Este coste incluye la suma de los gastos variables derivados del uso de la EC2, el almacenamiento y las operaciones en S3 y el uso de MediaConvert.

Duración	**Coste total 1 ejecución (€)**
30 seg	0,01
2 min	0,03
5 min	0,08
15 min	0,23
30 min	0,46

Tabla 48. Extracción del audio de un video: Resultados métricas de coste AWS variando la duración del video, Total

Resolución	**Coste total 1 ejecución (€)**
720p	0,04
1080p	0,08
4K	0,15

Tabla 49. Extracción del audio de un video: Resultados métricas de coste AWS variando la resolución del video, Total

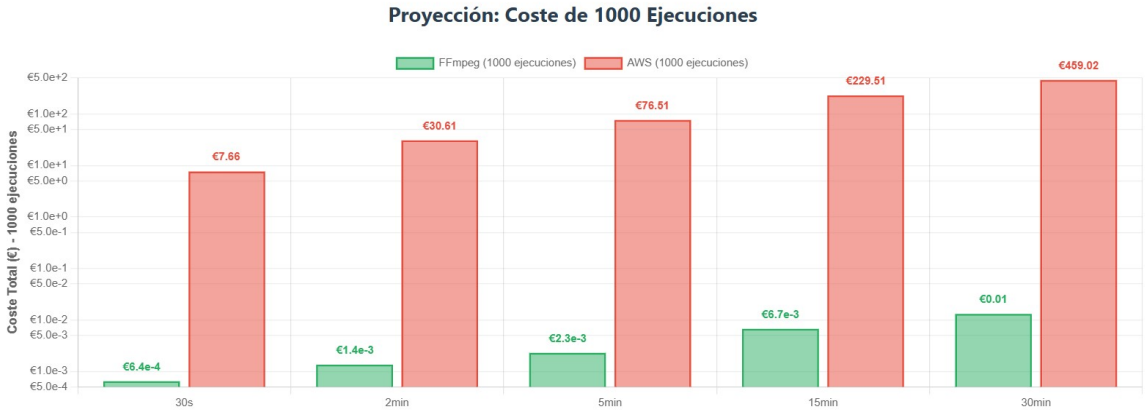


Figura 46. Extracción del audio de un video: Coste total variando la duración del video

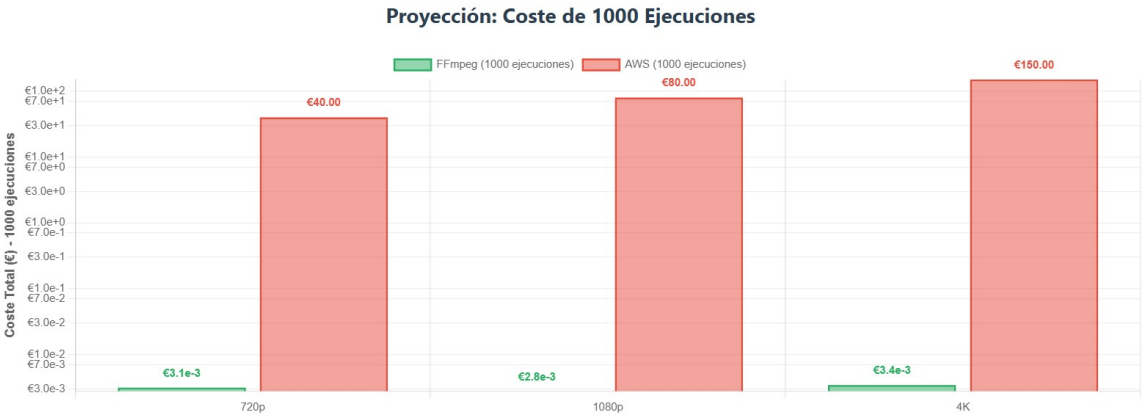


Figura 47. Extracción del audio de un video: Coste total variando la resolución del video

En las gráficas superiores se muestra el coste asociado a 1000 ejecuciones para las distintas duraciones y resoluciones en FFmpeg y AWS.

Se observa que la diferencia entre ambos costes es significativa: FFmpeg resulta mucho más económico que AWS, lo que se traduce en un mayor retorno de inversión (ROI). Al comparar una sola ejecución, la diferencia de coste no resulta relevante, sin embargo, considerando 1000 ejecuciones de un video de 30 minutos, el coste con FFmpeg asciende a solo 0,01€, mientras que con AWS alcanza 459€, evidenciando una diferencia extremadamente elevada.

En el caso de la variación de la resolución, también se aprecia una gran diferencia de coste entre AWS y FFmpeg. Para FFmpeg, el coste se mantiene prácticamente constante y nulo para las distintas resoluciones. Cualquier pequeña variación es prácticamente insignificante y se debe a ligeros cambios en la latencia, que impactan mínimamente en el tiempo de uso de la EC2 y, por ende, en el coste económico.

Sin embargo, no ocurre lo mismo en AWS al variar la resolución. Aunque la latencia de 720p es mayor que la de 1080p, el coste de esta última es superior, lo que indica que el coste de AWS en la extracción de audio aumenta con la resolución. Esto se debe principalmente a MediaConvert, que constituye el componente más costoso de la operación y cuyo precio se incrementa conforme aumenta la resolución del video.

En resumen, el coste de FFmpeg depende principalmente de la duración del video, mientras que en AWS aumenta tanto con la duración como con la resolución.

9.3.1.3 Normalización de audio

A continuación, se muestran los resultados de la operación de normalización de un audio. Se recomienda revisar la primera operación (extracción de audio de un vídeo), donde se explican con mayor detalle aspectos que aquí se omiten deliberadamente para evitar redundancias en la redacción.

Rendimiento

Duración	FFmpeg Throughput (archivos/seg)	AWS Throughput (archivos/seg)	Factor de Mejora FFmpeg
30 segundos	16.57	0.10	165.70x
2 minutos	5.32	0.05	106.40x
5 minutos	2.69	0.02	134.50x
15 minutos	0.97	0.008	121.25x
30 minutos	0.53	0.004	132.50x

Tabla 50. Normalización de audio: Resultados de throughput y factor de mejora variando la duración del video

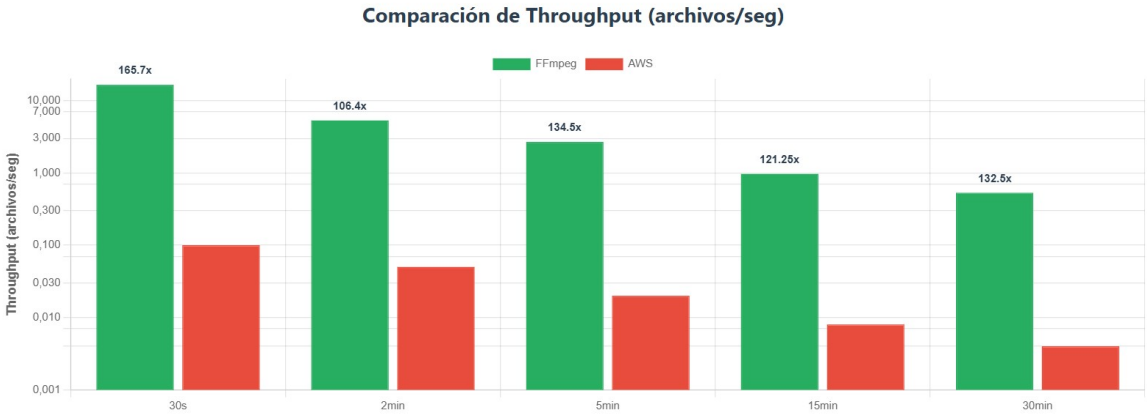


Figura 48. Normalización de audio: Resultados de throughput y factor de mejora variando la resolución del video

El throughput, como se explicó en la operación anterior de separación de audio del video, muestra un comportamiento esperado: La cantidad de archivos procesados por segundo disminuye a medida que aumenta la duración del video.

La diferencia respecto a la operación anterior es que aquí no se observa el comportamiento bifásico del factor de mejora. En este caso, el factor de mejora disminuye de forma consistente, ya que el throughput se reduce progresivamente con la duración del video tanto en AWS como en FFmpeg.

La caída repentina del factor de mejora en el video de 2 minutos, seguida de su recuperación en el video de 5 minutos, se debe a un uso subóptimo de los recursos del sistema en el video de 2 minutos, como se evidencia en la métrica de CPU, que muestra un menor

aprovechamiento de los recursos disponibles. En el resto de las gráficas de latencia y eficiencia se aprecia esta desviación respecto al patrón general en el video de 2 minutos.

Resolución	FFmpeg Throughput (archivos/seg)	AWS Throughput (archivos/seg)	Factor de Mejora FFmpeg
720p	0.1331	0.0207	6.43x
1080p	0.1338	0.0122	10.97x
4K	0.1314	0.0157	8.37x

Tabla 51. Normalización de audio: Resultados de throughput y factor de mejora variando la duración del video

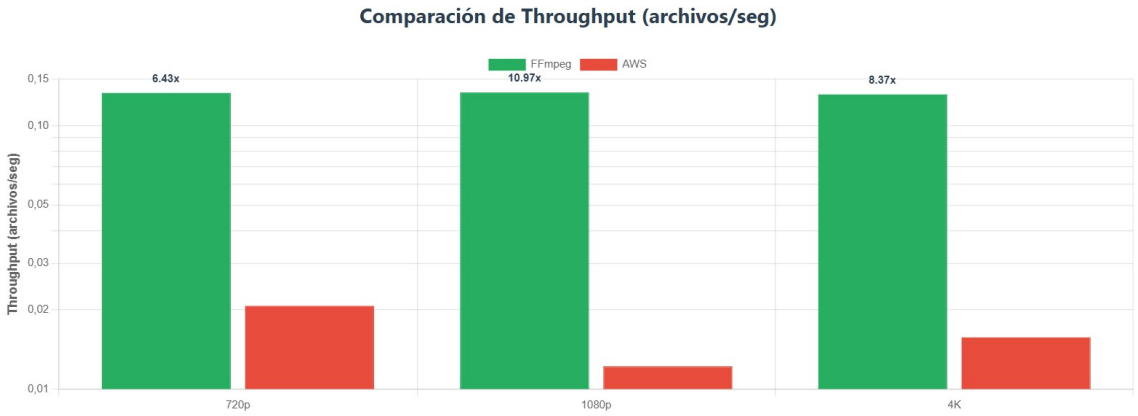


Figura 49. Normalización de audio: Resultados de throughput y factor de mejora variando la resolución del video

Lo que se puede destacar del throughput respecto a la resolución es que, una vez más, parece que este no se ve afectado por la resolución del video, sino por el tamaño del archivo. Este comportamiento era completamente esperado, ya que en la operación de normalización se utiliza el audio extraído del video con la resolución indicada, y desde el inicio se preveía que la resolución no tendría impacto en este proceso, lo cual se confirma con los datos.

Por lo tanto, como se esperaba, la resolución del video original del cual se extrae el audio no afecta el throughput, tanto en FFmpeg como en AWS.

Duración	Tecnología	Latencia Total (s)	Tiempo Procesamiento (s)	Tiempo I/O (s)	Overhead (s)	Eficiencia Procesamiento (%)
30 seg	FFmpeg	0.0808	0.0789	0.0015	0.0019	97.67
	AWS	9.444	6.689	2.735	3.159	70.84
2 min	FFmpeg	0.1933	0.1778	0.0091	0.0095	91.98
	AWS	19.032	12.844	6.188	6.761	67.47
5 min	FFmpeg	0.3613	0.3498	0.0112	0.0114	96.83
	AWS	47.440	25.476	21.964	22.448	53.71

15 min	FFmpeg	1.1130	1.0640	0.0485	0.0489	95.60
	AWS	120.830	55.173	65.122	66.142	45.66
30 min	FFmpeg	2.2466	2.1405	0.1056	0.1061	95.27
	AWS	260.130	97.112	162.520	163.598	37.33

Tabla 52. Normalización de audio: Resultados de métricas de rendimiento variando la duración del video

Figura 50. Normalización de audio: Crecimiento de la latencia total variando la duración del video

A diferencia de la operación anterior, en la que se separaba el audio del video, aquí ambas operaciones presentan un crecimiento lineal de su latencia. Esto indica que la operación de normalización es más eficiente a largo plazo que la de separación de audio en AWS, aunque las latencias siguen siendo considerablemente mayores que las de FFmpeg.

El crecimiento lineal de la latencia en AWS se explica porque, en esta operación, no se procesa video, sino únicamente el audio, lo que hace que el proceso sea menos costoso.

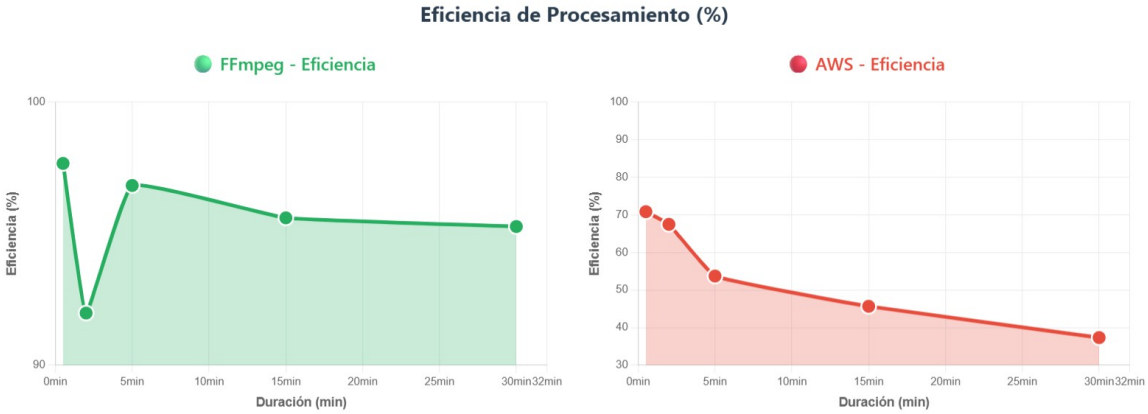


Figura 51. Normalización de audio: Crecimiento de la eficiencia variando la duración del video

La eficiencia, como es habitual, es inversa a la latencia. En este caso, ambas presentan un decrecimiento lineal.

En FFmpeg, se ha mostrado un rango reducido de la escala de eficiencia (de 90 a 100) para visualizar mejor las diferencias entre los valores. En el video de 2 minutos se observa un decrecimiento de eficiencia, atribuible al fenómeno explicado al inicio de la sección sobre el comportamiento del throughput en la normalización del audio. Esto indica un comportamiento

irregular del sistema, evidenciado por un uso subóptimo de los recursos y un menor aprovechamiento de la CPU, que se puede consultar en la Tabla 54. Normalización de audio: Resultado de métricas de recursos variando la duración del video.

Resolución	Tecnología	Latencia Total (s)	Tiempo Procesamiento (s)	Tiempo I/O (s)	Overhead (s)	Eficiencia Procesamiento (%)
720p	FFmpeg	7.49	7.40	0.019	0.090	98.80%
	AWS	47.96	26.09	21.49	21.87	54.40%
1080p	FFmpeg	7.50	7.48	0.012	0.02	99.73%
	AWS	81.14	29.50	51.64	52.18	36.35%
4K	FFmpeg	7.70	7.68	0.015	0.02	99.74%
	AWS	63.08	24.79	38.29	38.81	39.31%

Tabla 53. Normalización de audio: Resultados de métricas de rendimiento variando la resolución del video

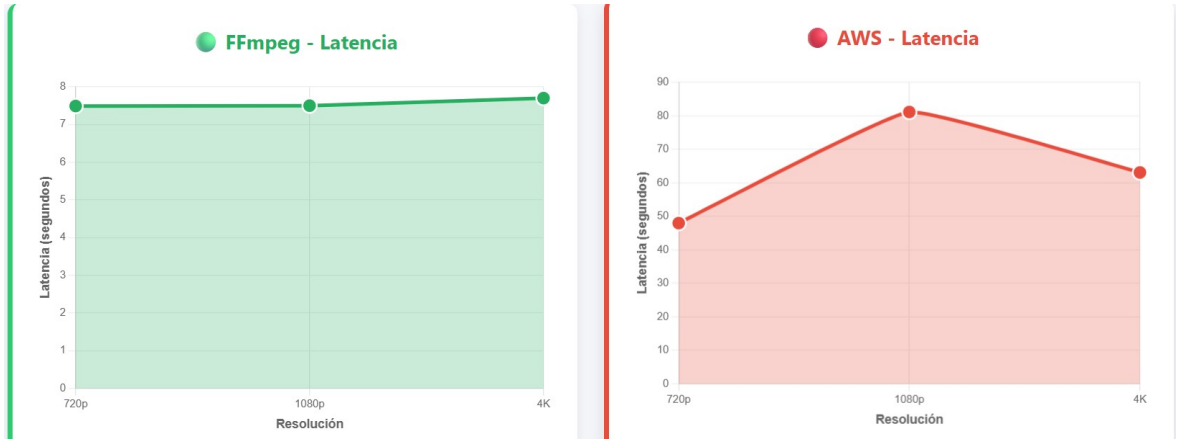


Figura 52. Normalización de audio: Crecimiento de la latencia total variando la resolución del video

En FFmpeg, la latencia varía muy poco, lo cual es razonable, ya que la resolución del video del que se extrae el audio para normalizar no debería afectar la operación.

En el caso de AWS, la fluctuación de la latencia se puede explicar por las diferencias en el tamaño de los distintos audios y la interferencia de la red. El tiempo de procesamiento puro, que se muestra en la tabla, se mantiene ligeramente por encima de 7 s. Lo que realmente varía y provoca cambios en la latencia es el tiempo de entrada y salida asociado a la subida y descarga de los archivos en S3.

Por lo tanto, como se esperaba, la resolución del video original del cual se extrae el audio no afecta el tiempo de procesamiento ni en FFmpeg ni en AWS.

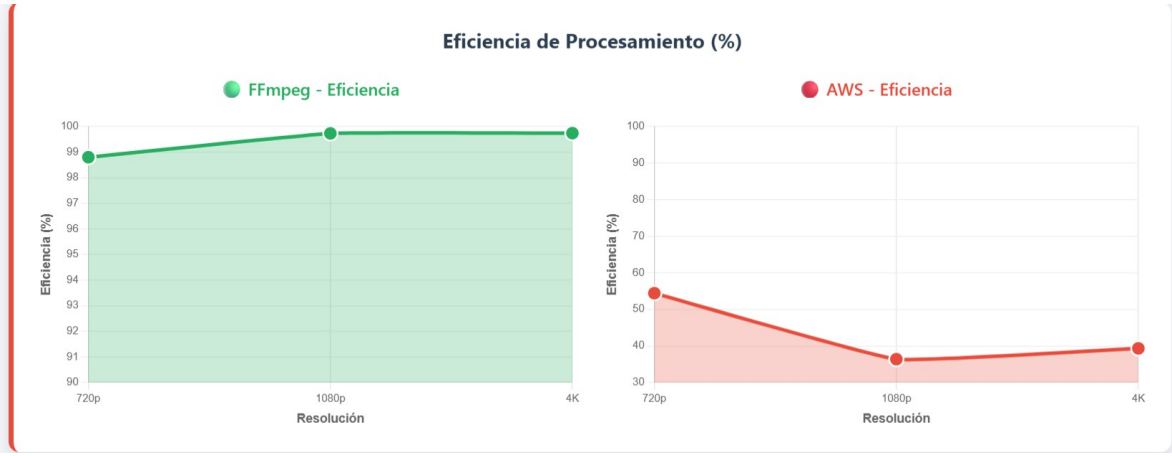


Figura 53. Normalización de audio: Crecimiento de la eficiencia variando la resolución del video

Las gráficas superiores de eficiencia muestran un comportamiento inverso al de la latencia, al igual que en los casos anteriores.

A continuación, se muestran las gráficas del desglose de la latencia en tiempo de procesamiento, overhead asociado a operaciones de entrada/salida y overhead de otras operaciones, respecto a la duración y la resolución.

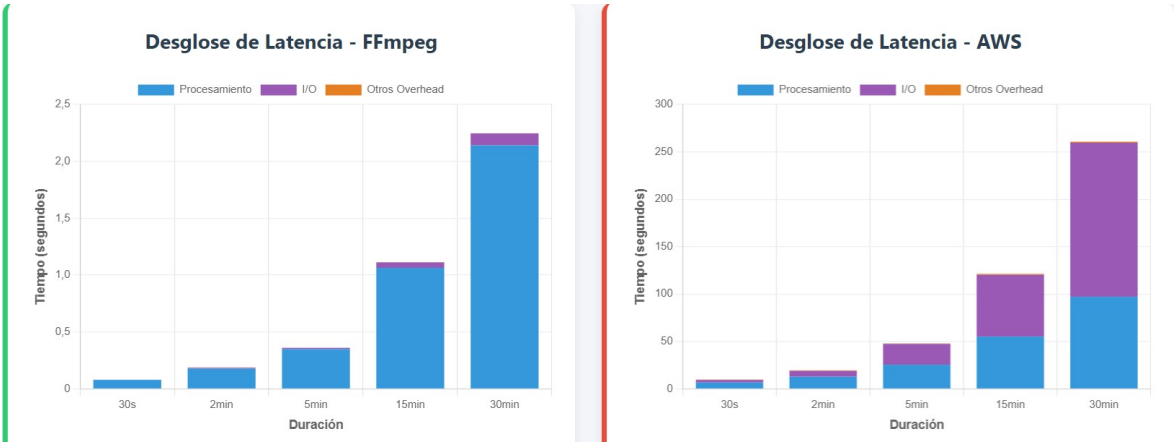


Figura 54. Normalización de audio: Desglose de la latencia variando la duración del video.

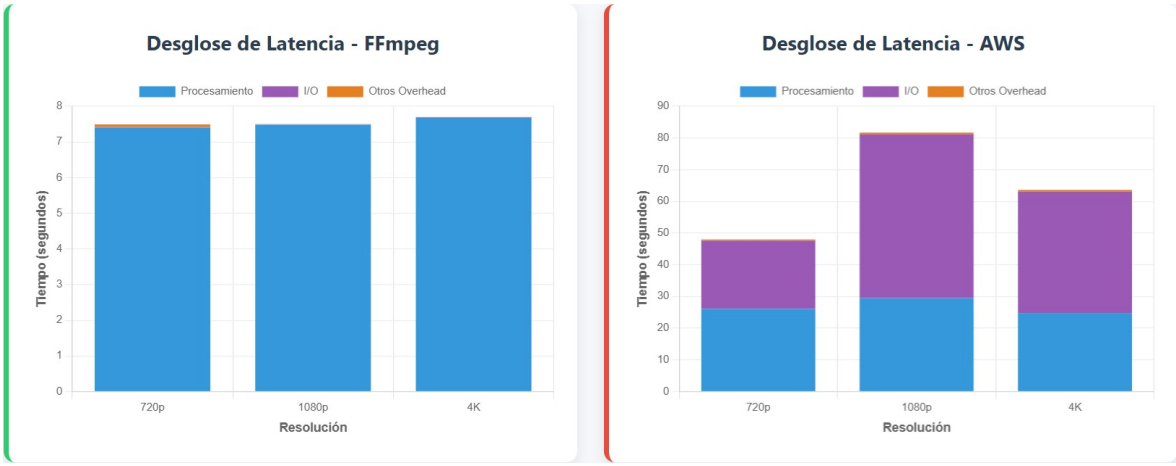


Figura 55. Normalización de audio: Desglose de la latencia variando la resolución del video.

Estas gráficas muestran que prácticamente el 100% de la latencia de FFmpeg se dedica a procesamiento puro, mientras que en AWS una gran parte corresponde a operaciones de entrada y salida, superando en muchos casos el 50% del tiempo total. Esto explica la alta eficiencia de FFmpeg frente a la baja eficiencia de AWS.

Recursos

Duración	Uso_CPU (%)	Uso_RAM (MB)	Almacenamiento (MB)
30 segundos	187.41	12.48	0.62
2 minutos	149.60	21.95	2.51
5 minutos	189.58	21.70	4.83
15 minutos	198.11	22.12	17.11
30 minutos	192.53	22.13	34.18

Tabla 54. Normalización de audio: Resultado de métricas de recursos variando la duración del video

Resolución	Uso_CPU (%)	Uso_RAM (MB)	Almacenamiento (MB)
720p	102.07	75.59	15.03
1080p	111.62	78.94	13.89
4K	120.60	77.06	15.03

Tabla 55. Normalización de audio: Resultado de métricas de recursos variando la resolución del video

Coste – FFmpeg

Cabe recordar que el precio de la EC2 es de 0,0216 € por hora, lo que equivale aproximadamente a 6×10^{-6} € por segundo.

Duración	Latencia Total (s)	Precio total (€)
30 seg	0.0808	4.85×10^{-7}
2 min	0.1933	1.16×10^{-6}
5 min	0.3613	2.17×10^{-6}
15 min	1.1130	6.68×10^{-6}
30 min	2.2466	1.35×10^{-5}

Tabla 56. Normalización de audio: Resultado métricas de coste FFmpeg variando la duración del video

Resolución	Latencia Total (s)	Precio total (€)
720p	7.49	4.49×10^{-5}
1080p	7.50	4.50×10^{-5}
4K	7.70	4.62×10^{-5}

Tabla 57. Normalización de audio: Resultado métricas de coste FFmpeg variando la resolución del video

Coste – AWS

En el caso de AWS, para la función de normalizar un audio, además del uso de EC2 y del almacenamiento, se paga por utilizar MediaConvert y S3.

Para el coste del uso de la EC2:

Duración	Latencia Total (s)	Precio total (€)
30 seg	9.444	5.67×10^{-5}
2 min	19.032	1.14×10^{-4}
5 min	47.440	2.85×10^{-4}
15 min	55.173	3.31×10^{-4}
30 min	97.112	5.83×10^{-4}

Tabla 58. Normalización de audio: Resultados métricas de coste AWS variando la duración del video, EC2

Resolución	Latencia Total (s)	Precio total (€)
720p	47.96	2.88×10^{-4}
1080p	81.14	4.87×10^{-4}
4K	63.08	3.78×10^{-4}

Tabla 59. Normalización de audio: Resultados métricas de coste AWS variando la resolución del video, EC2

Para el coste del uso de S3:

Recordar:

- El coste de 1GB por hora por mes es de: $3x10^{-5}$ €
- Precio por una operación PUT: $4.86x10^{-6}$ €
- Precio de una operación GET y DELETE: $3.87x10^{-7}$ €
- Cantidad de operaciones PUT: 1 operación
- Cantidad de operaciones GET: 1 operación
- Cantidad de operaciones DELETE: 2 operaciones

El coste total del almacenamiento en S3 al mes de 1 ejecución de la función de normalización de un audio es el siguiente:

Duración	Cantidad (GB)	Coste total almacenamiento (€)
30 seg	0.000606403	1.82×10^{-8}
2 min	0.002452736	7.36×10^{-8}
5 min	0.004721355	1.42×10^{-7}
15 min	0.016710711	5.01×10^{-7}

30 min	0.033381615	1.00×10^{-6}

Tabla 60. Normalización de audio: Resultados métricas de coste AWS variando la duración del video, almacenamiento S3

Resolución	Cantidad (GB)	Coste total almacenamiento (€)
720p	0.014681387	4.40×10^{-7}
1080p	0.013561153	4.07×10^{-7}
4K	0.014681387	4.40×10^{-7}

Tabla 61. Normalización de audio: Resultados métricas de coste AWS variando la resolución del video, almacenamiento S3

El coste total de las operaciones en S3 de una ejecución de la función de normalización de audio es el mismo que el de separar el audio de un vídeo, mostrado en la Tabla 45. Extracción del audio de un video: Resultados métricas de coste AWS, operaciones S3, ya que la cantidad de operaciones en S3 realizadas es la misma. Por lo tanto, el coste es de $6{,}021 \times 10^{-6}$ €.

En el caso de MediaConvert, el coste de normalizar un audio varía respecto al de la separación del audio del vídeo, ya que el coste por minuto de procesamiento de audio difiere del de procesamiento de vídeo. El coste de procesamiento de audio por minuto y por mes es de 0,00124 €.

A continuación, se muestra el coste del procesamiento con MediaConvert de una sola ejecución para cada duración.

Duración	Coste total (€)
30 seg	0,00062
2 min	0,00248
5 min	0,0062
15 min	0,0186
30 min	0,0372

Tabla 62. Normalización de audio: Resultados métricas de coste AWS variando la duración del video, MediaConvert

En el caso de la resolución, se han usado audios de 5 minutos, por lo que el precio de 1 ejecución con las diferentes resoluciones cuesta lo mismo todas, ya que no varía con la resolución.

Resolución	Coste total (€)
720p	0,0062
1080p	0,0062
4K	0,0062

Tabla 63. Normalización de audio: Resultados métricas de coste AWS variando la resolución del video, MediaConvert

A continuación, se muestra una tabla con el coste de una única ejecución de la función de normalización de audio. Este coste incluye la suma de los gastos variables derivados del uso de la EC2, el almacenamiento y las operaciones en S3 y el uso de MediaConvert.

Duración	Coste total 1 ejecución (€)
30 seg	0,0007
2 min	0,003
5 min	0,006
15 min	0,02
30 min	0,04

Tabla 64. Normalización de audio: Resultados métricas de coste AWS variando la duración del video, Total

Resolución	Coste total 1 ejecución (€)
720p	0,006
1080p	0,007
4K	0,007

Tabla 65. Normalización de audio: Resultados métricas de coste AWS variando la resolución del video, Total

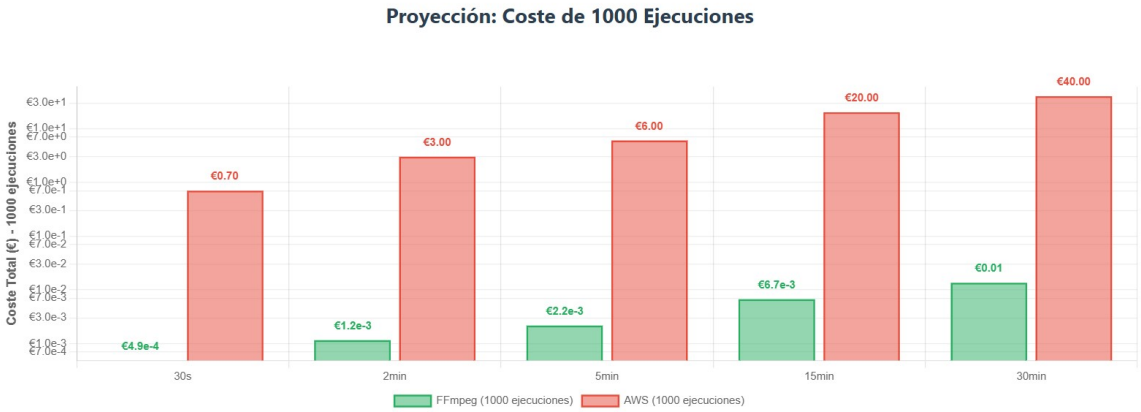


Figura 56. Normalización de audio: Coste total variando la duración del video

Se puede observar que el coste aumenta con la duración tanto en AWS como en FFmpeg, teniendo valores de coste mucho más altos en AWS que en FFmpeg, siguiendo el mismo comportamiento que en la operación de extracción de audio de un video. Para más detalles, se puede consultar la sección correspondiente a esa operación.

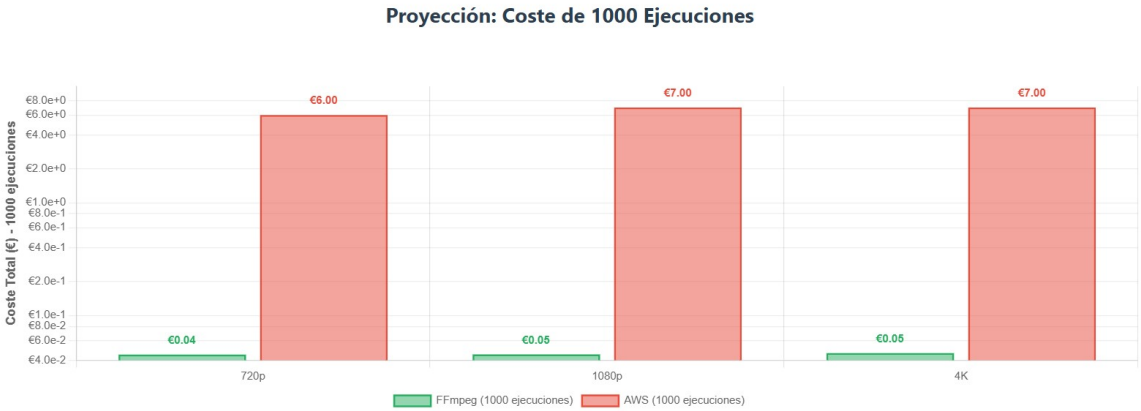


Figura 57. Normalización de audio: Coste total variando la resolución del video

En este caso, a diferencia de la operación de separación de audio del video, el coste de AWS no depende de la resolución del video del cual se extrae el audio para normalizarlo. En la operación anterior de separación de audio, el coste de AWS sí variaba con la resolución. Sin embargo, aquí no ocurre lo mismo, ya que el coste de AWS MediaConvert para el procesamiento de audio no se ve afectado por la resolución.

Por lo que, se puede concluir que el coste de la operación de normalizar un audio se mantiene constante respecto a la variación de la resolución con ambas tecnologías, con la diferencia de que el coste de AWS es mucho más elevado.

9.3.1.4 Detección cambios de escena

A continuación, se muestran los resultados de la operación de detección de cambios de escena en un video. Se recomienda revisar la primera operación (extracción de audio de un vídeo), donde se explican con mayor detalle aspectos que aquí se omiten deliberadamente para evitar redundancias en la redacción.

Rendimiento

Duración	FFmpeg Throughput (archivos/seg)	AWS Throughput (archivos/seg)	Factor de Mejora FFmpeg
30 segundos	1.24	0.07	17.71x
2 minutos	0.40	0.04	10.00x
5 minutos	0.17	0.03	5.67x
15 minutos	0.06	0.02	3.00x
30 minutos	0.02	0.01	2.00x

Tabla 66. Detección cambios de escena: Resultados de throughput y factor de mejora variando la duración del video

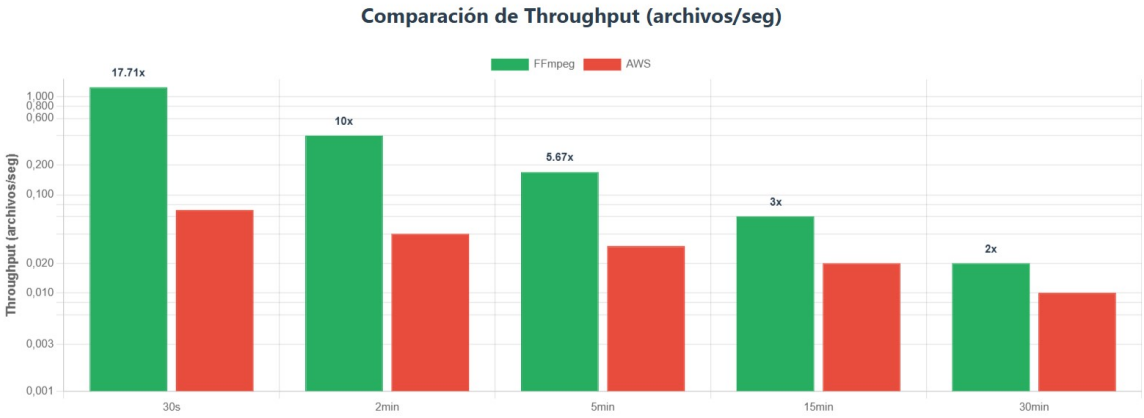


Figura 58. Detección cambios de escena: Resultados de throughput y factor de mejora variando la duración del video

El throughput presenta el comportamiento esperado, en el que la cantidad de archivos procesados por segundo decrece conforme aumenta la duración del video. Al igual que en la operación de normalización de audio, no se observa el patrón bifásico identificado en la operación de extracción de audio del video. En este caso, el factor de mejora disminuye de manera sostenida, reflejando la reducción progresiva del throughput tanto en AWS como en FFmpeg a medida que incrementa la duración del material procesado.

Resolución	FFmpeg Throughput (archivos/seg)	AWS Throughput (archivos/seg)	Factor de Mejora FFmpeg
720p	0.29	0.037	7.84x
1080p	0.17	0.031	5.48x
4K	0.04	0.02	2.00x

Tabla 67. Detección cambios de escena: Resultados de throughput y factor de mejora variando la resolución del video

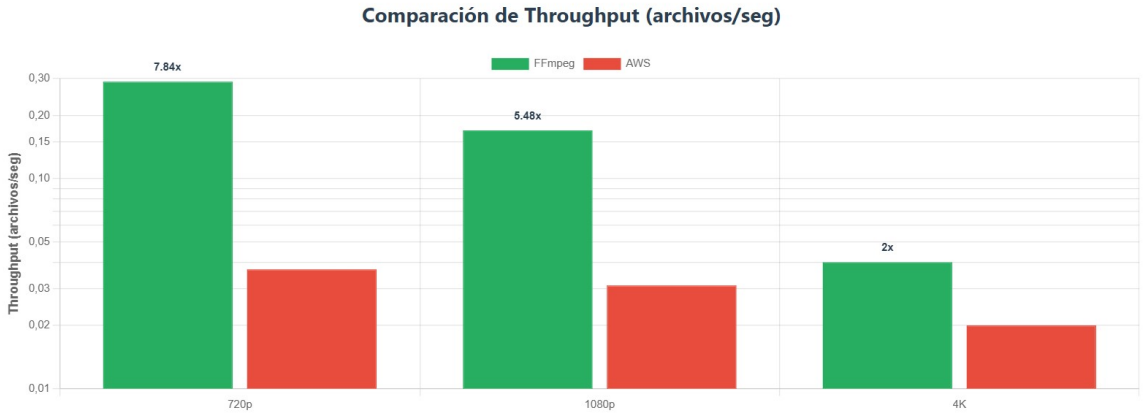


Figura 59. Detección cambios de escena: Resultados de throughput y factor de mejora variando la resolución del video

El throughput de FFmpeg disminuye de forma pronunciada con la resolución debido a la naturaleza intensiva de la detección de cambios de escena. Mientras que en 720p el procesamiento es manejable (unos 1M píxeles/frame), en 1080p la carga se incrementa 2,25 veces (unos 2M píxeles/frame), lo que fuerza comparaciones inter-frame más costosas. En

4K, los casi 8,3M píxeles/frame saturan la CPU y la memoria, generando cuellos de botella en el bus de datos y uso intensivo de paginación.

En AWS, la caída es más moderada (0,037 a 0,02 archivos/s) y se debe principalmente a los overheads de infraestructura más que al límite del procesamiento en sí, como la transferencia de datos, inicialización de servicios en el cloud entre otros.

En la resolución 4K, la diferencia de throughput entre FFmpeg y AWS se reduce (FFmpeg solo 2× más rápido), lo que se explica porque el procesamiento local alcanza los límites físicos de la CPU, mientras que la nube sostiene un rendimiento mínimo gracias a su escalabilidad. Sin embargo, en resoluciones bajas la ventaja de FFmpeg (7,8× en 720p) evidencia que, en tareas de análisis intensivo, la proximidad a los datos y el control del hardware superan a las arquitecturas en la nube cuando los volúmenes de datos no justifican los overheads de transferencia.

Duración	Tecnología	Latencia Total (s)	Tiempo Procesamiento (s)	Tiempo I/O (s)	Overhead (s)	Eficiencia Procesamiento (%)
30 seg	FFmpeg	0.806	0.788	0.0018	0.018	97.77%
	AWS	15.302	13.108	2.194	2.194	85.66%
2 min	FFmpeg	2.504	2.471	0.0050	0.033	98.68%
	AWS	23.184	18.391	4.793	4.794	79.31%
5 min	FFmpeg	5.827	5.797	0.0113	0.030	99.48%
	AWS	31.373	21.906	9.467	9.467	69.81%
15 min	FFmpeg	17.608	17.551	0.0336	0.057	99.68%
	AWS	60.193	35.792	24.401	24.401	59.46%
30 min	FFmpeg	60.285	60.083	0.1686	0.202	99.66%
	AWS	174.832	98.517	76.315	76.315	56.36%

Tabla 68. Detección cambios de escena: Resultados de métricas de rendimiento variando la duración del video

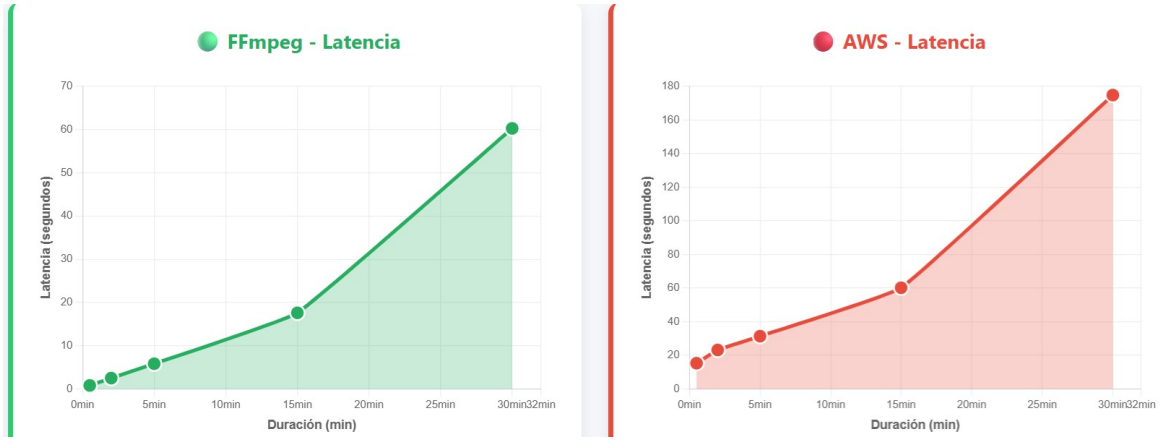


Figura 60. Detección cambios de escena: Crecimiento de la latencia total variando la duración del video

En ambos casos, la latencia muestra un crecimiento similar con tendencia lineal. A medida que aumenta la duración del video, y con ello su tamaño y la cantidad de datos a procesar, se requiere más tiempo para completar la tarea. Aunque AWS continúa presentando tiempos de latencia superiores a los de FFmpeg, también es cierto que FFmpeg alcanza valores de latencia considerablemente más altos que en operaciones previas, lo que evidencia que la detección de cambios de escena es una tarea altamente demandante en términos de recursos computacionales.

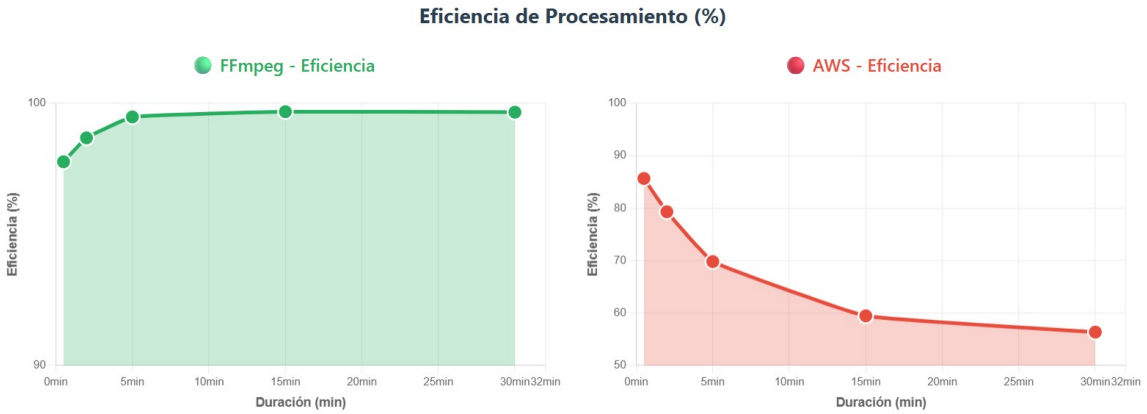


Figura 61. Detección cambios de escena: Crecimiento de la eficiencia variando la duración del video

En FFmpeg, la eficiencia aumenta con la duración del video porque el overhead, que incluye el tiempo de I/O y otras gestiones y preparaciones, no varía significativamente con la duración. Aunque al aumentar la duración del video también crece el tamaño y, por tanto, la cantidad de datos a gestionar en el almacenamiento local, el tiempo de procesamiento puro aumenta considerablemente. Este comportamiento provoca que el impacto relativo del overhead disminuya a medida que la duración se incrementa, generando un aumento de la eficiencia en videos más largos.

En AWS, la eficiencia disminuye porque el overhead del sistema es elevado, principalmente debido a la subida y descarga de archivos a S3, la cual aumenta con la duración y el tamaño de los datos, mientras que el tiempo de procesamiento puro aumenta de manera no tan pronunciada debido a la escalabilidad del procesamiento computacional en entornos cloud. Este comportamiento de la eficiencia es consistente con el observado en las operaciones anteriores explicadas hasta ahora.

Resolución	Tecnología	Latencia Total (s)	Tiempo Procesamiento (s)	Tiempo I/O (s)	Overhead (s)	Eficiencia Procesamiento (%)
720p	FFmpeg	3.402	3.370	0.011	0.032	99.06%
	AWS	27.159	20.816	6.342	6.343	76.65%
1080p	FFmpeg	5.868	5.834	0.015	0.034	99.42%
	AWS	32.122	22.087	10.034	10.035	68.76%
4K	FFmpeg	24.653	24.563	0.012	0.090	99.64%
	AWS	50.245	29.077	21.167	21.168	57.86%

Tabla 69. Detección cambios de escena: Resultados de métricas de rendimiento variando la resolución del video

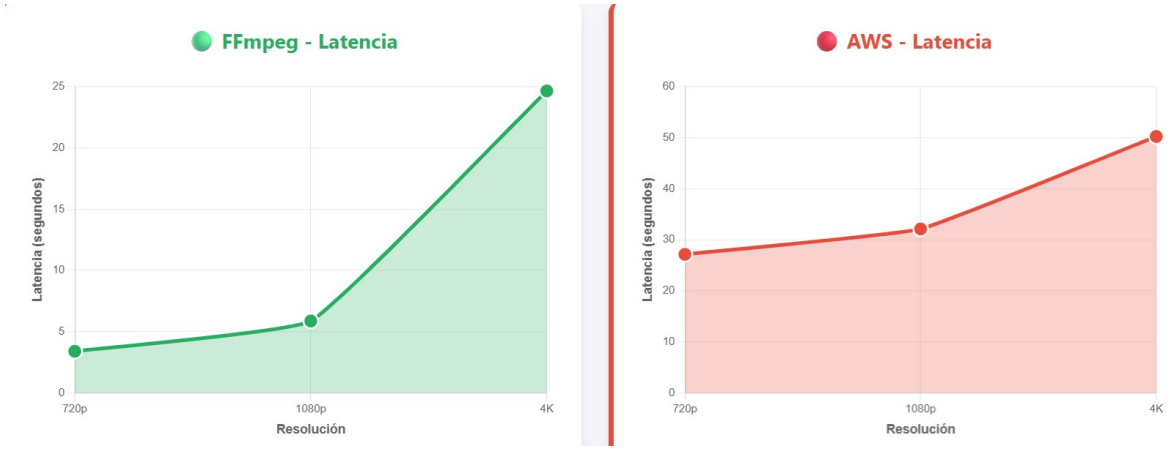


Figura 62. Detección cambios de escena: Crecimiento de la latencia total variando la resolución del video

La latencia aumenta con la resolución, lo cual es esperado, ya que, en la operación de detección de cambios de escena, un mayor número de píxeles por frame implica comparaciones inter-frame más costosas.

Aunque las latencias de AWS son superiores a las de FFmpeg, el incremento en FFmpeg es más pronunciado. Esto se explica por la saturación de los recursos computacionales locales. Al procesar en la máquina disponible, que cuenta con recursos limitados, el tiempo de procesamiento aumenta a medida que la resolución y la carga computacional crecen, como se observó y explicó también en la gráfica del throughput respecto a la resolución.

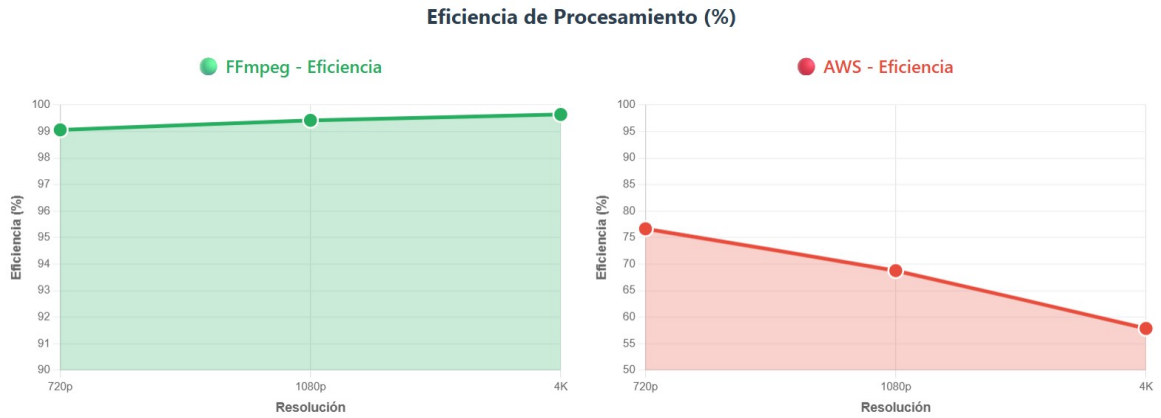


Figura 63. Detección cambios de escena: Crecimiento de la eficiencia variando la resolución del video

La eficiencia muestra el mismo comportamiento que respecto a la duración. En FFmpeg aumenta a medida que crece la resolución, al reducirse el impacto relativo del overhead, mientras que en AWS disminuye debido al alto overhead de subir y descargar archivos, manteniéndose consistente con lo observado en operaciones anteriores.

A continuación, se muestran las gráficas del desglose de la latencia en tiempo de procesamiento, overhead asociado a operaciones de entrada/salida y overhead de otras operaciones, respecto a la duración y la resolución.

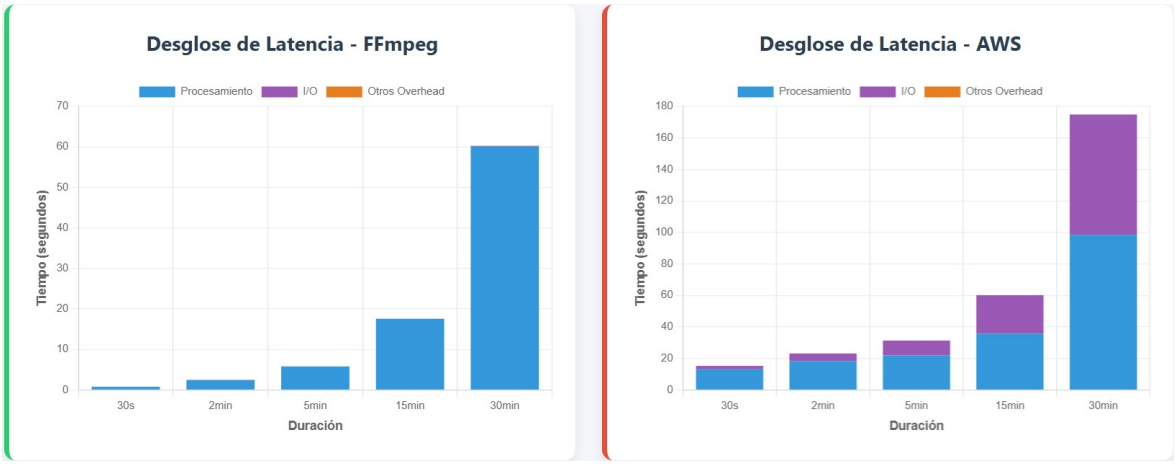


Figura 64. Detección cambios de escena: Desglose de la latencia variando la duración del video.

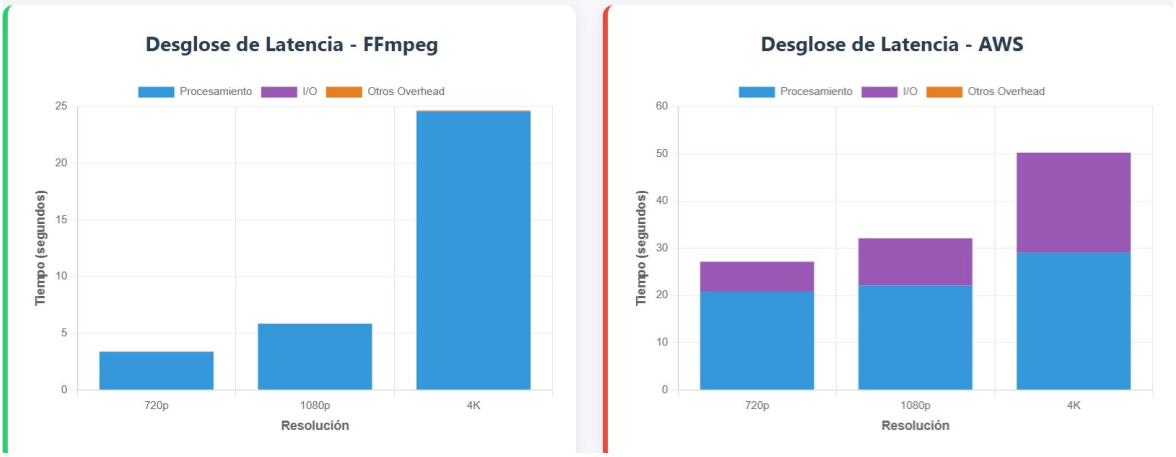


Figura 65. Detección cambios de escena: Desglose de la latencia variando la resolución del video.

Como siempre, en FFmpeg la mayor parte de la latencia corresponde a procesamiento puro. En AWS, durante la operación de detección de cambios de escena, el overhead de entrada y salida es menor que en otros casos, aunque claramente puede aumentar significativamente cuando se procesan archivos de gran tamaño.

Recursos

Duración	Uso_CPU (%)	Uso_RAM (MB)	Almacenamiento (MB)
30 segundos	178.88	188.88	2.18
2 minutos	183.43	210.56	7.68
5 minutos	188.24	191.13	16.33
15 minutos	193.12	193.40	48.83
30 minutos	198.90	204.19	207.24

Tabla 70. Detección cambios de escena: Resultado de métricas de recursos variando la duración del video

Resolución	Uso_CPU (%)	Uso_RAM (MB)	Almacenamiento (MB)
720p	184.92	123.49	16.24
1080p	197.79	189.63	16.33
4K	199.82	600.58	15.62

Tabla 71. Detección cambios de escena: Resultado de métricas de recursos variando la resolución del video

Coste – FFmpeg

Cabe recordar que el precio de la EC2 es de 0,0216 € por hora, lo que equivale aproximadamente a 6×10^{-6} € por segundo.

Duración	Latencia Total (s)	Precio total (€)
30 seg	0.806	4.84×10^{-6}
2 min	2.504	1.50×10^{-5}
5 min	5.827	3.50×10^{-5}
15 min	17.608	1.06×10^{-4}
30 min	60.285	3.62×10^{-4}

Tabla 72. Detección cambios de escena: Resultado métricas de coste FFmpeg variando la duración del video

Resolución	Latencia Total (s)	Precio total (€)
720p	3.402	2.04×10^{-5}
1080p	5.868	3.52×10^{-5}
4K	24.563	1.47×10^{-4}

Tabla 73. Detección cambios de escena: Resultado métricas de coste FFmpeg variando la resolución del video

La tabla anterior muestra el coste de ejecutar la función según la duración y resolución del video.

Coste – AWS

En el caso de AWS, para la función de detección de cambios de escena en un video, además del uso de EC2 y del almacenamiento, se paga por utilizar AWS Rekognition y S3.

Para el coste del uso de la EC2:

Duración	Latencia Total (s)	Precio total (€)
30 seg	15.302	9.18×10^{-5}
2 min	23.184	1.39×10^{-4}
5 min	31.373	1.88×10^{-4}
15 min	60.193	3.61×10^{-4}

30 min	174.832	1.05×10^{-3}

Tabla 74. Detección cambios de escena: Resultados métricas de coste AWS variando la duración del video, EC2

Resolución	Latencia Total (s)	Precio total (€)
720p	27.159	1.63×10^{-4}
1080p	32.122	1.93×10^{-4}
4K	50.245	3.01×10^{-4}

Tabla 75. Detección cambios de escena: Resultados métricas de coste AWS variando la resolución del video, EC2

Para el coste del uso de S3:

Recordar:

- El coste de 1GB por hora por mes es de: $3x10^{-5}$ €
- Precio por una operación PUT: $4.86x10^{-6}$ €
- Precio de una operación GET y DELETE: $3.87x10^{-7}$ €
- Cantidad de operaciones PUT: 1 operación
- Cantidad de operaciones GET: 0 operaciones
- Cantidad de operaciones DELETE: 1 operación

Dado que en la operación actual de detección de cambios de escena se utilizan los mismos archivos de vídeo que en la extracción de audio, la duración y resolución de los archivos es idéntica, por lo que los costes también lo son. Estos costes se pueden consultar en las tablas Tabla 43. Extracción del audio de un video: Resultados métricas de coste AWS variando la duración del video, almacenamiento S3 y Tabla 44. Extracción del audio de un video: Resultados métricas de coste AWS variando la resolución del video, almacenamiento S3.

El coste total de las operaciones de 1 ejecución de la función que detecta cambios de escena es el siguiente:

Cantidad PUTs	*Coste total PUT (€)*	*Cantidad GETs*	*Coste total GET (€)*	*Cantidad DELETEs*	*Coste total DELETE (€)*	*Coste total (€)*
1	4.86×10^{-6}	0	0	1	3.87×10^{-7}	5.247×10^{-6}

Tabla 76. Detección cambios de escena: Resultados métricas de coste AWS, operaciones S3

Para el coste del uso de AWS Rekognition:

En este caso, en lugar de usar MediaConvert, se usa AWS Rekognition. El precio por minuto es de 0.054€, independientemente de la resolución usada. A continuación, se muestra el coste del procesamiento con AWS Rekognition de una sola ejecución para cada duración.

Duración	Coste total (€)
30 seg	0.027
2 min	0.108

5 min	0.27
15 min	0.81
30 min	1.62

Tabla 77. Detección cambios de escena**:** Resultados métricas de coste AWS variando la duración del video, Rekognition

En el caso de la resolución, se han utilizado vídeos de 5 minutos, por lo que el precio de una ejecución es el mismo para todas las resoluciones, ya que en AWS Rekognition no varía en función de esta.

Resolución	**Coste total (€)**
720p	0.27
1080p	0.27
4K	0.27

Tabla 78. Detección cambios de escena: Resultados métricas de coste AWS variando la resolución del video, Rekognition

A continuación, se muestra una tabla con el coste de una única ejecución de la función de detección de cambios de escena en un video. Este coste incluye la suma de los gastos variables derivados del uso de la EC2, el almacenamiento y las operaciones en S3 y el uso de AWS Rekognition.

Duración	**Coste total 1 ejecución (€)**
30 seg	0,03
2 min	0,11
5 min	0,27
15 min	0,81
30 min	1,62

Tabla 79. Detección cambios de escena: Resultados métricas de coste AWS variando la duración del video, Total

Resolución	**Coste total 1 ejecución (€)**
720p	0,27
1080p	0,27
4K	0,27

Tabla 80. Detección cambios de escena: Resultados métricas de coste AWS variando la resolución del video, Total

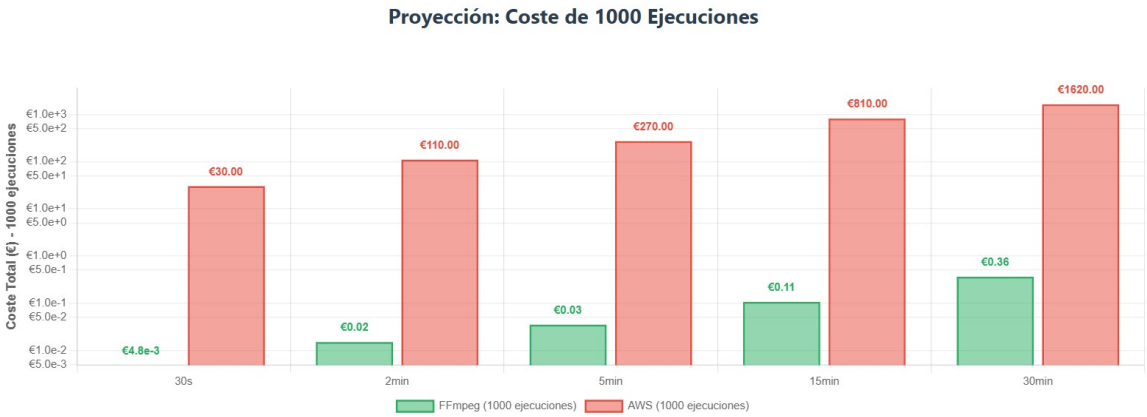


Figura 66. Detección cambios de escena: Coste total variando la duración del video

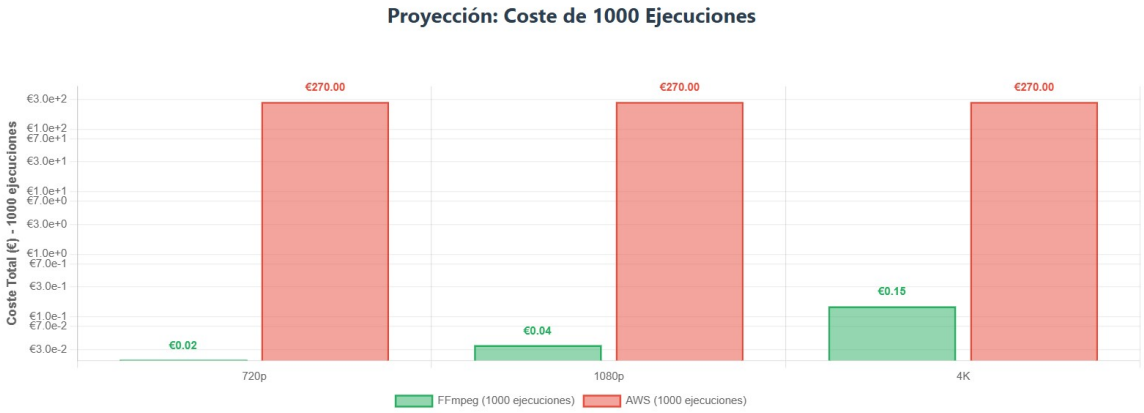


Figura 67. Detección cambios de escena: Coste total variando la resolución del video

Igual que en los otros casos, AWS presenta un coste mucho más elevado que FFmpeg.

Al comparar el coste respecto a la duración, se observa que la operación de detección de cambios de escena es significativamente más cara que las operaciones anteriores de separar el audio de un video o normalizar un audio. Este incremento se debe a que esta operación conlleva mayor uso de la EC2 por la elevada latencia y utiliza AWS Rekognition, cuyo coste es considerablemente mayor que el de MediaConvert.

En relación con el coste asociado a la resolución del vídeo, se observa que en AWS el precio se mantiene constante, dado que Rekognition no modifica sus tarifas en función de la resolución del contenido procesado. Este servicio constituye, además, el principal componente del coste total. Por el contrario, el uso de FFmpeg sí se ve afectado por la resolución, ya que un incremento en la resolución conlleva un incremento en el tiempo de procesamiento que prolonga la ejecución en la instancia EC2, cuyo coste de uso representa el principal componente del gasto al trabajar con FFmpeg, elevando así el coste económico final.

9.3.1.5 Extracción de un frame de un video

A continuación, se muestran los resultados de la operación de extraer un frame (fotograma) de un video. Se recomienda revisar la primera operación (extracción de audio de un vídeo), donde se explican con mayor detalle aspectos que aquí se omiten deliberadamente para evitar redundancias en la redacción.

Rendimiento

Duración	FFmpeg Throughput (archivos/seg)	AWS Throughput (archivos/seg)	Factor de Mejora FFmpeg
30 segundos	3.07	0.088	34.89x
2 minutos	3.82	0.067	57.01x
5 minutos	3.54	0.041	86.34x
15 minutos	3.72	0.014	265.71x
30 minutos	2.35	0.0045	522.22x

Tabla 81. Extracción de un frame de un video: Resultados de throughput y factor de mejora variando la duración del video

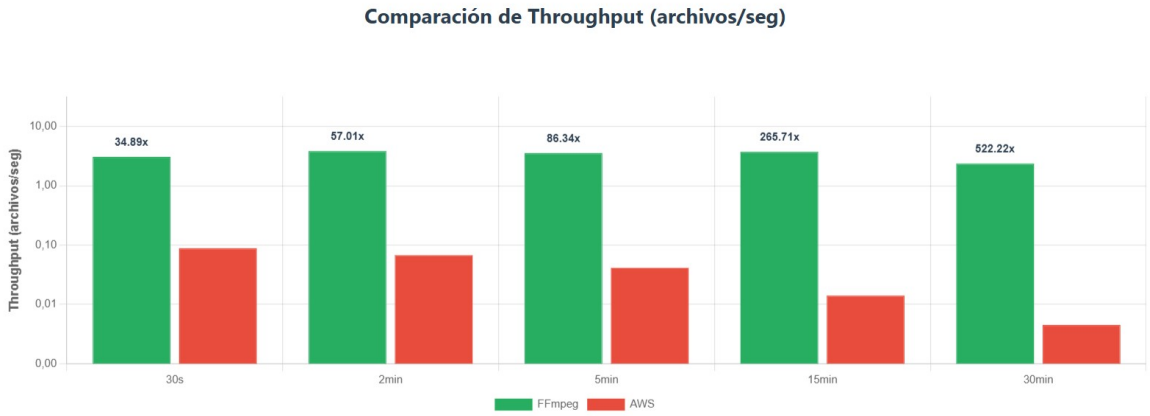


Figura 68. Extracción de un frame de un video: Resultados de throughput y factor de mejora variando la duración del video

El throughput disminuye a medida que aumenta la duración del video, ya que al haber más datos que procesar, la cantidad de información procesada por segundo se reduce.

A diferencia de las otras operaciones, en la extracción de frames del video el factor de mejora aumenta con la duración. Esto se debe a que el throughput de FFmpeg disminuye a un ritmo mucho más lento que el de AWS. Este comportamiento también se refleja en las latencias, donde FFmpeg muestra variaciones menores en comparación con AWS.

Resolución	FFmpeg Throughput (archivos/seg)	AWS Throughput (archivos/seg)	Factor de Mejora FFmpeg
720p	8.13	0.07	121.71x
1080p	5.85	0.04	148.40x
4K	2.83	0.02	127.60x

Tabla 82. Extracción de un frame de un video: Resultados de throughput y factor de mejora variando la resolución del video

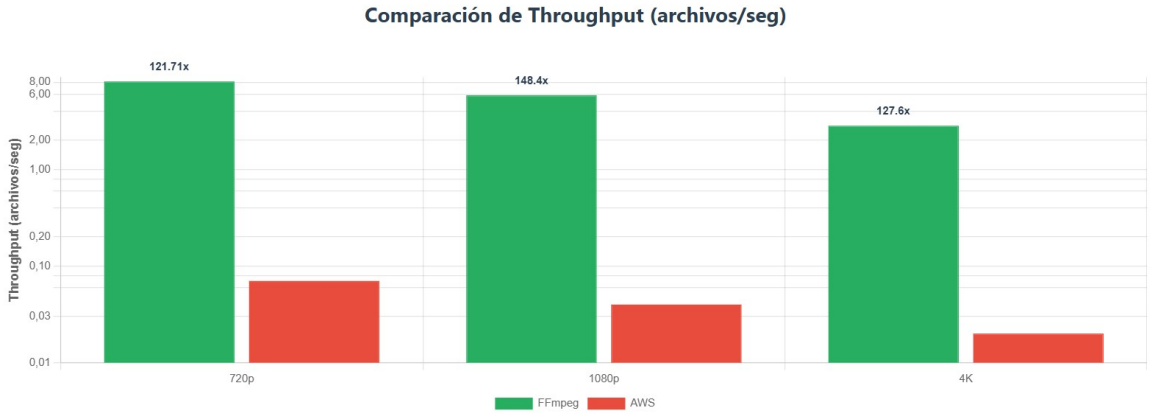


Figura 69. Extracción de un frame de un video: Resultados de throughput y factor de mejora variando la resolución del video

En cuanto a la resolución, a medida que esta aumenta, el throughput disminuye, es decir, se procesan menos archivos por segundo. Esto es completamente lógico, ya que una mayor resolución incrementa la carga computacional debido al aumento de píxeles, lo que requiere más tiempo para completar la operación y, por lo tanto, reduce la cantidad de archivos que se pueden procesar en un mismo período de tiempo.

Duración	Tecnología	Latencia Total (s)	Tiempo Procesamiento (s)	Tiempo I/O (s)	Overhead (s)	Eficiencia Procesamiento (%)
30 seg	FFmpeg	0.350	0.293	0.0027	0.057	83.71%
	AWS	10.412	6.353	4.059	4.892	61.02%
2 min	FFmpeg	0.251	0.215	0.0105	0.036	85.66%
	AWS	14.190	6.596	7.594	8.215	46.49%
5 min	FFmpeg	0.276	0.245	0.0173	0.031	88.77%
	AWS	23.797	11.012	12.785	13.535	46.27%
15 min	FFmpeg	0.223	0.179	0.0363	0.044	80.27%
	AWS	71.205	31.129	40.076	40.774	43.72%
30 min	FFmpeg	0.389	0.186	0.1979	0.203	47.81%
	AWS	221.571	74.420	147.151	147.967	33.59%

Tabla 83. Extracción de un frame de un video: Resultados de métricas de rendimiento variando la duración del video

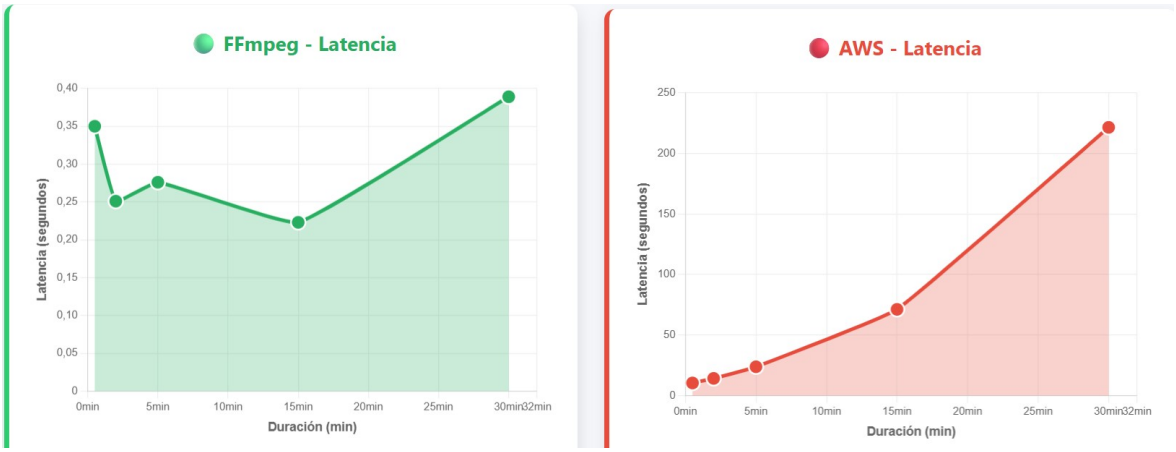


Figura 70. Extracción de un frame de un video: Crecimiento de la latencia total variando la duración del video

En FFmpeg, la latencia se mantiene bastante constante (0.223s – 0.389s), influida principalmente por el overhead de entrada/salida y la gestión del almacenamiento local, mientras que el tiempo de procesamiento puro no depende de la duración del video. Los altibajos observados responden más a factores de optimización y administración del sistema que a la operación en sí, pues es imposible obtener resultados idénticos en todas las mediciones. Esto permite concluir que la duración del video no afecta de manera significativa la operación de extracción de un frame. Además, al emplearse almacenamiento local, el impacto de la entrada/salida sobre la latencia total es leve.

En AWS, por el contrario, la latencia crece significativamente conforme aumenta la duración del video, mostrando un comportamiento cercano al exponencial. Esto marca una diferencia fundamental: Mientras que en FFmpeg la operación se mantiene estable, en AWS la duración del video sí condiciona directamente el tiempo de procesamiento.

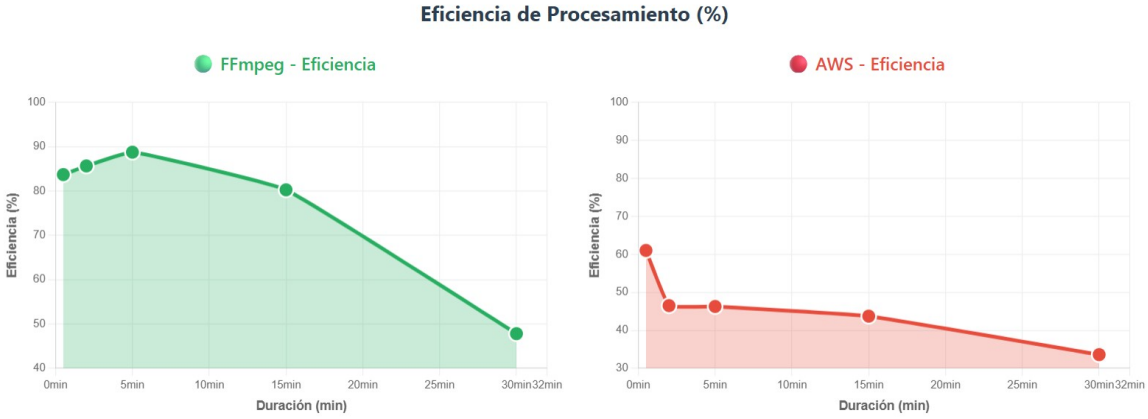


Figura 71. Extracción de un frame de un video: Crecimiento de la eficiencia variando la duración del video

En FFmpeg, se observa una caída de la eficiencia en videos de mayor duración debido al tiempo de entrada y salida, mientras que en videos cortos los valores presentan pocas fluctuaciones. En AWS, la eficiencia disminuye de manera más pronunciada debido al overhead asociado a la subida y descarga de archivos desde S3, ya que el uso de la red amplifica el impacto en comparación con el almacenamiento local de FFmpeg.

La mayor eficiencia observada en el archivo de 5 minutos se explica por un uso más óptimo de los recursos del sistema, como la CPU y la RAM, tal como se muestra en la tabla de recursos más abajo.

Resolución	Tecnología	Latencia Total (s)	Tiempo Procesamiento (s)	Tiempo I/O (s)	Overhead (s)	Eficiencia Procesamiento (%)
720p	FFmpeg	0.118	0.110	0.006	0.008	93.22%
	AWS	14.945	7.294	7.083	7.651	48.81%
1080p	FFmpeg	0.171	0.158	0.011	0.013	92.40%
	AWS	25.382	11.477	13.380	13.905	45.22%
4K	FFmpeg	0.353	0.319	0.008	0.034	90.37%
	AWS	44.958	31.322	13.031	13.636	69.65%

Tabla 84. Extracción de un frame de un video: Resultados de métricas de rendimiento variando la resolución del video

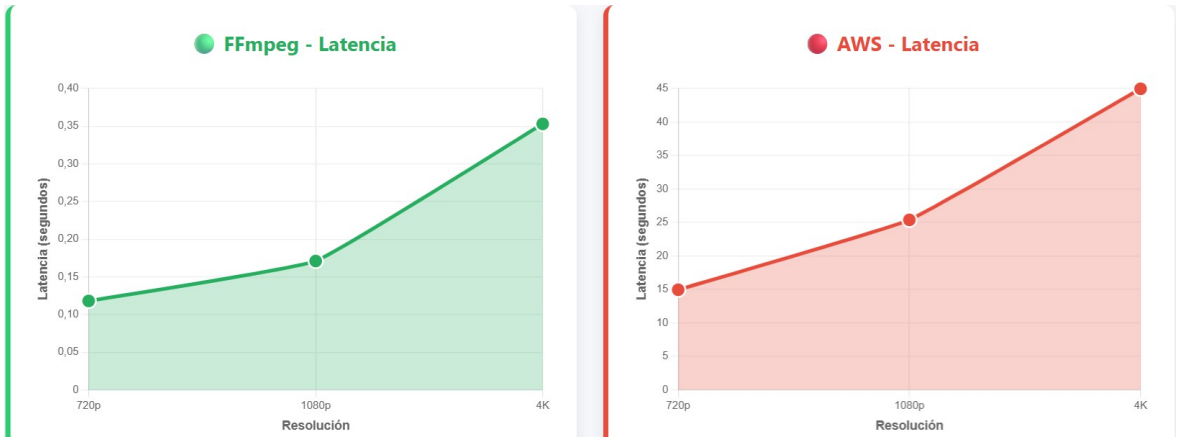


Figura 72. Extracción de un frame de un video: Crecimiento de la latencia total variando la resolución del video

En cuanto a la resolución, tanto la latencia como el tiempo de procesamiento aumentan a medida que esta se incrementa. Esto es totalmente esperado, al igual que en el caso del throughput respecto a la duración, ya que una mayor resolución eleva la carga computacional por el aumento de píxeles, lo que requiere más tiempo para completar la operación.

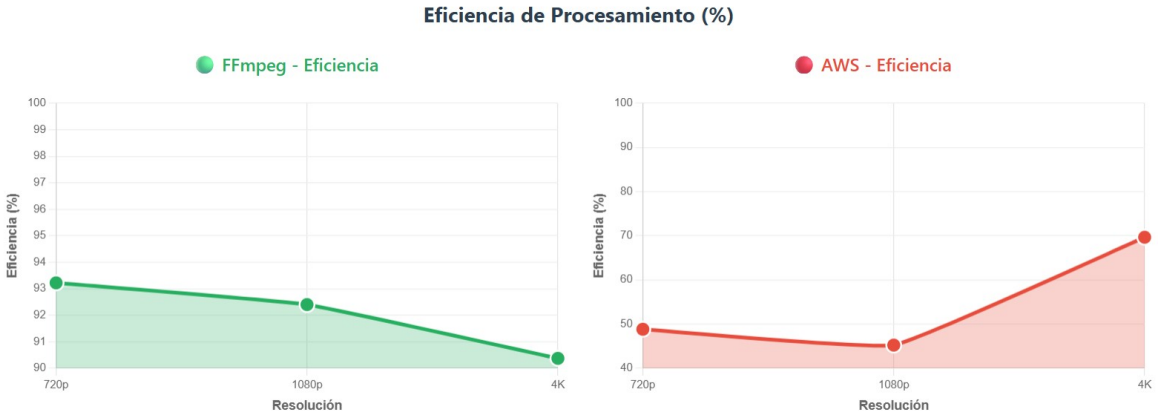


Figura 73. Extracción de un frame de un video: Crecimiento de la eficiencia variando la resolución del video

La eficiencia en FFmpeg se mantiene alta (>90%), aunque disminuye ligeramente con la resolución. Esto se debe a que, aunque el tiempo de procesamiento puro aumenta con la resolución, también lo hace el overhead. Por ejemplo, al manejar el frame extraído en la RAM

o en el disco local, se genera un overhead adicional, además, el frame ocupa más espacio a medida que la resolución aumenta.

En el caso de AWS, en resoluciones bajas, la eficiencia cae drásticamente (45-49%) debido a que el procesamiento real es rápido pero dominado por I/O y overhead. Subir un video completo a S3 solo para extraer un frame resulta desproporcionado. En cambio, en 4K, la eficiencia mejora (69,65%) porque el procesamiento puro representa una mayor proporción del tiempo total frente al I/O y al overhead.

Por lo tanto, la resolución afecta la eficiencia en la extracción de un frame de un video, tanto en FFmpeg como en AWS. Esto es completamente esperable, ya que anteriormente se observó que la latencia también se ve influida por la resolución del video.

A continuación, se muestran las gráficas del desglose de la latencia en tiempo de procesamiento, overhead asociado a operaciones de entrada/salida y overhead de otras operaciones, respecto a la duración y la resolución.

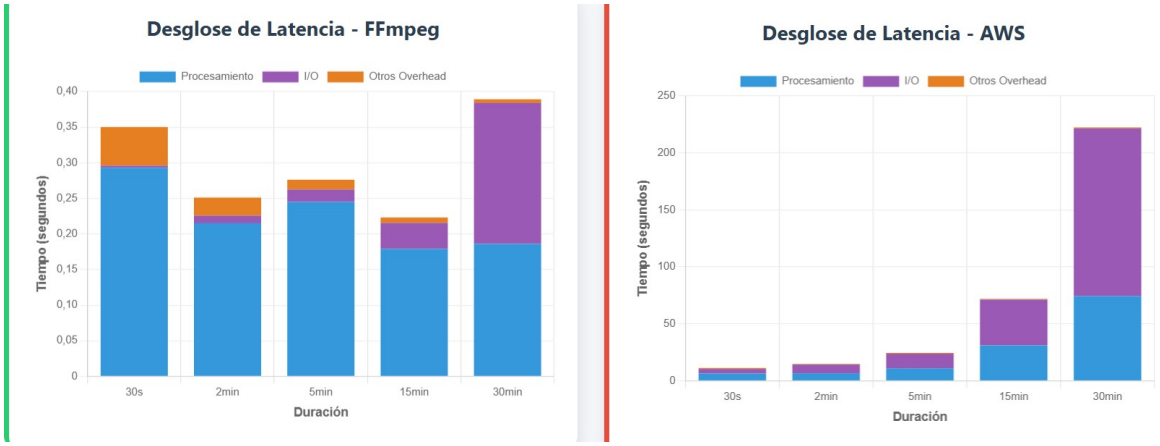


Figura 74. Extracción de un frame de un video: Desglose de la latencia variando la duración del video.

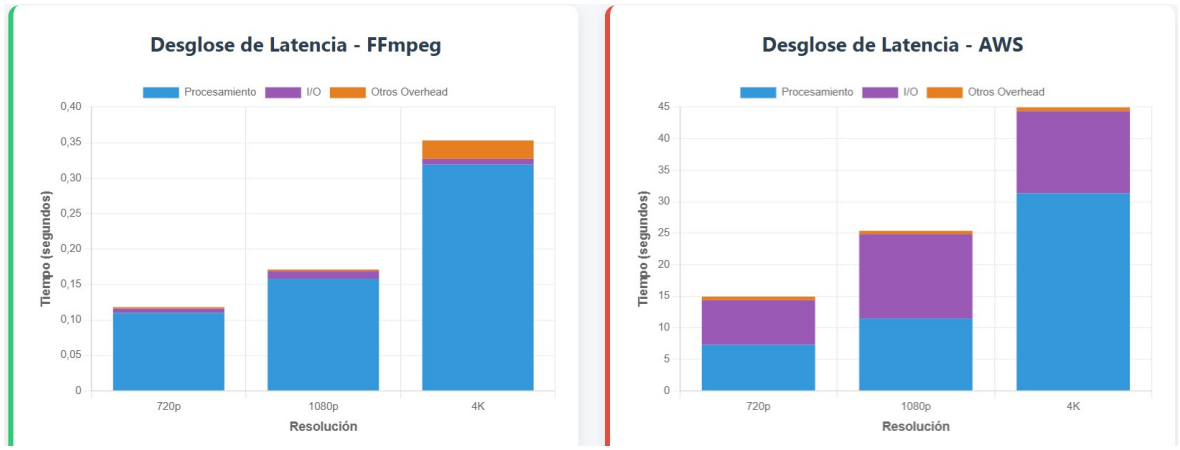


Figura 75. Extracción de un frame de un video: Desglose de la latencia variando la resolución del video.

En estas gráficas se puede observar el impacto del overhead de entrada y salida y del tiempo de procesamiento en FFmpeg y AWS, de manera similar a lo visto en operaciones anteriores. En AWS, el tiempo de entrada y salida se ve especialmente afectado por la transferencia de archivos a través de la red.

Recursos

Duración	Uso_CPU (%)	Uso_RAM (MB)	Almacenamiento (MB)
30 segundos	146.31	172.72	2.18
2 minutos	173.43	190.53	7.68
5 minutos	189.34	210.56	16.33
15 minutos	198.12	193.42	48.83
30 minutos	198.91	204.48	207.24

Tabla 85. Extracción de un frame de un video: Resultado de métricas de recursos variando la duración del video

Resolución	Uso_CPU (%)	Uso_RAM (MB)	Almacenamiento (MB)
720p	185.75	107.45	16.24
1080p	181.93	160.85	16.33
4K	196.93	533.39	15.62

Tabla 86. Extracción de un frame de un video: Resultado de métricas de recursos variando la resolución del video

Coste – FFmpeg

Cabe recordar que el precio de la EC2 es de 0,0216 € por hora, lo que equivale aproximadamente a 6×10^{-6} € por segundo.

Duración	Latencia Total (s)	Precio total (€)
30 seg	0.350	2.10×10^{-6}
2 min	0.251	1.51×10^{-6}
5 min	0.276	1.66×10^{-6}
15 min	0.223	1.34×10^{-6}
30 min	0.389	2.33×10^{-6}

Tabla 87. Extracción de un frame de un video: Resultado métricas de coste FFmpeg variando la duración del video

Resolución	Latencia Total (s)	Precio total (€)
720p	0.118	7.08×10^{-7}
1080p	0.171	1.03×10^{-6}
4K	0.353	2.12×10^{-6}

Tabla 88. Extracción de un frame de un video: Resultado métricas de coste FFmpeg variando la resolución del video

La tabla anterior muestra el coste de ejecutar la función según la duración y resolución del video.

Coste – AWS

En el caso de AWS, para la función de extraer un frame de un video, además del uso de EC2 y del almacenamiento, se paga por utilizar MediaConvert y S3.

Para el coste del uso de la EC2:

Duración	Latencia Total (s)	Precio total (€)
30 seg	10.412	6.25×10^{-5}
2 min	14.190	8.51×10^{-5}
5 min	23.797	1.43×10^{-4}
15 min	71.205	4.27×10^{-4}
30 min	221.571	1.33×10^{-3}

Tabla 89. Extracción de un frame de un video: Resultados métricas de coste AWS variando la duración del video, EC2

Resolución	Latencia Total (s)	Precio total (€)
720p	14.945	8.97×10^{-5}
1080p	25.382	1.52×10^{-4}
4K	44.958	2.70×10^{-4}

Tabla 90. Extracción de un frame de un video: Resultados métricas de coste AWS variando la resolución del video, EC2

Para el coste del uso de S3:

Recordar:

- El coste de 1GB por hora por mes es de: $3x10^{-5}$ €
- Precio por una operación PUT: $4.86x10^{-6}$ €
- Precio de una operación GET y DELETE: $3.87x10^{-7}$ €
- Cantidad de operaciones PUT: 1 operación
- Cantidad de operaciones GET: 1 operación
- Cantidad de operaciones DELETE: 2 operaciones

Dado que en la operación actual de extracción de un frame (fotograma) de un video se utilizan los mismos archivos de vídeo que en la extracción de audio, la duración y resolución de los archivos es idéntica, por lo que los costes también lo son. Estos costes se pueden consultar en las tablas Tabla 43. Extracción del audio de un video: Resultados métricas de coste AWS variando la duración del video, almacenamiento S3 y Tabla 44. Extracción del audio de un video: Resultados métricas de coste AWS variando la resolución del video, almacenamiento S3.

El coste total de las operaciones en S3 de una ejecución de la función que extrae un fotograma de un vídeo es el mismo que el de separar el audio de un vídeo, mostrado en la Tabla 45. Extracción del audio de un video: Resultados métricas de coste AWS, operaciones S3, ya que la cantidad de operaciones en S3 realizadas es la misma. Por lo tanto, el coste es de $6{,}021 \times 10^{-6}$ €.

Para el coste del uso de MediaConvert, como que depende de la duración y resolución del video, también son los mismos costes calculados en la operación de separar el audio del

video en las tablas Tabla 46. Extracción del audio de un video: Resultados métricas de coste AWS variando la duración del video, MediaConvert y Tabla 47. Extracción del audio de un video: Resultados métricas de coste AWS variando la resolución del video, MediaConvert.

A continuación, se muestra una tabla con el coste de una única ejecución de la función que extrae un frame de un video. Este coste incluye la suma de los gastos variables derivados del uso de la EC2, el almacenamiento y las operaciones en S3 y el uso de MediaConvert.

Duración	Coste total 1 ejecución (€)
30 seg	0,01
2 min	0,03
5 min	0,08
15 min	0,23
30 min	0,46

Tabla 91. Extracción de un frame de un video: Resultados métricas de coste AWS variando la duración del video, Total

Resolución	Coste total 1 ejecución (€)
720p	0,04
1080p	0,08
4K	0,15

Tabla 92. Extracción de un frame de un video: Resultados métricas de coste AWS variando la resolución del video, Total

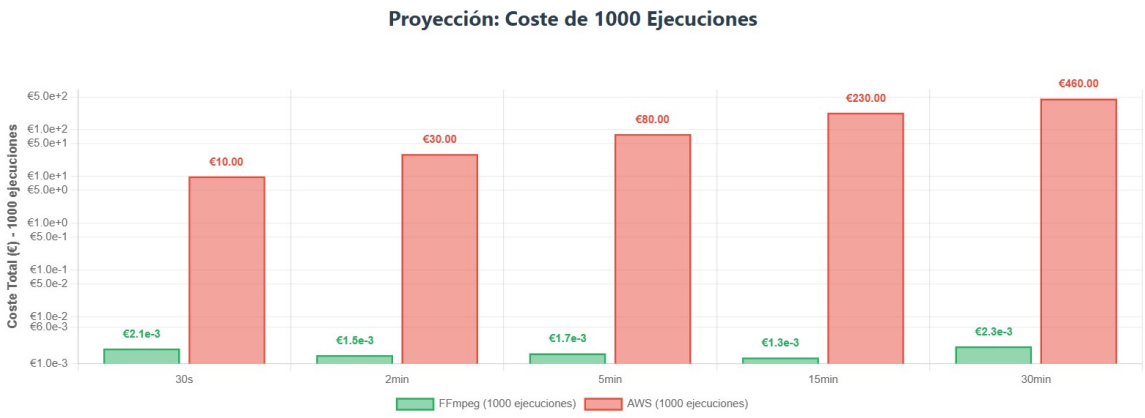


Figura 76. Extracción de un frame de un video: Coste total variando la duración del video

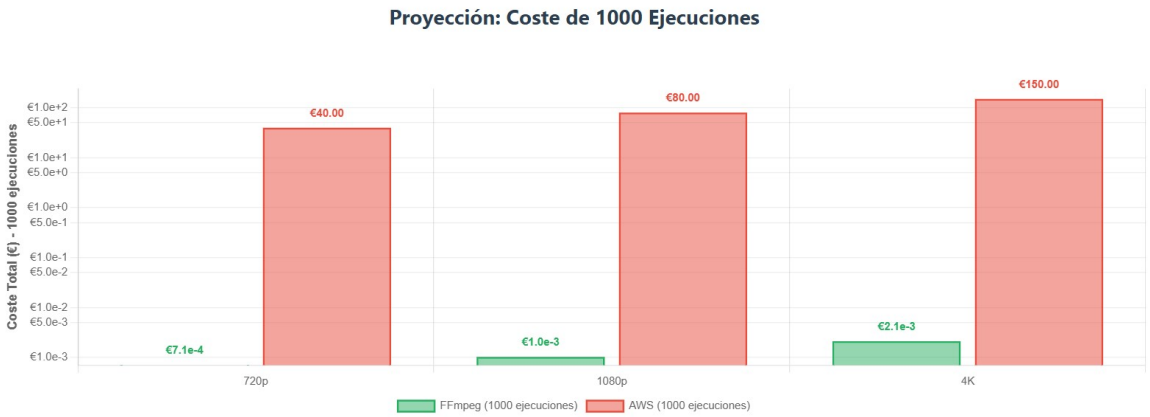


Figura 77. Extracción de un frame de un video: Coste total variando la resolución del video

En cuanto a los costes, se puede aplicar lo mismo que se explicó en la operación de extraer el audio de un video. En resumen, FFmpeg resulta mucho más económico. El coste de ambos, AWS y FFmpeg, aumenta con la duración del video, ya que al incrementarse la latencia se requiere más tiempo de uso de la instancia EC2. Además, en el caso de AWS se utiliza MediaConvert. En AWS, el coste también crece con la resolución, dado que MediaConvert cobra por minuto de procesamiento y este se incrementa a medida que aumenta la resolución.

La diferencia con la operación de separar el audio de un video es que, en FFmpeg, el coste también aumenta con la resolución. Al extraer un frame, con un mayor número de píxeles implica mayor carga computacional, más tiempo de procesamiento y, por lo tanto, un mayor uso de la instancia EC2, lo que eleva el coste económico.

9.3.1.6 Reducción de ruido de un audio

En este caso, tal y como se explicó en el desarrollo del proyecto, no se ha implementado esta funcionalidad con AWS, ya que no existe ningún servicio de AWS que realice esta operación.

Por esta razón, en este apartado se presentarán únicamente los resultados de FFmpeg, sin realizar ninguna comparación con AWS.

Rendimiento

Duración	Tecnología	Latencia Total (s)	Tiempo Procesamiento (s)	Tiempo I/O (s)	Overhead (s)	Eficiencia Procesamiento (%)
30 seg	FFmpeg	6.50	6.26	0.20	0.24	96.31%
2 min	FFmpeg	22.38	21.82	0.49	0.56	97.50%
5 min	FFmpeg	52.17	50.79	1.29	1.38	97.35%
15 min	FFmpeg	115.58	112.36	3.05	3.22	97.21%
30 min	FFmpeg	189.07	180.42	8.34	8.65	95.42%

Tabla 93. Reducción de ruido de un audio: Resultados de métricas de rendimiento variando la duración del video

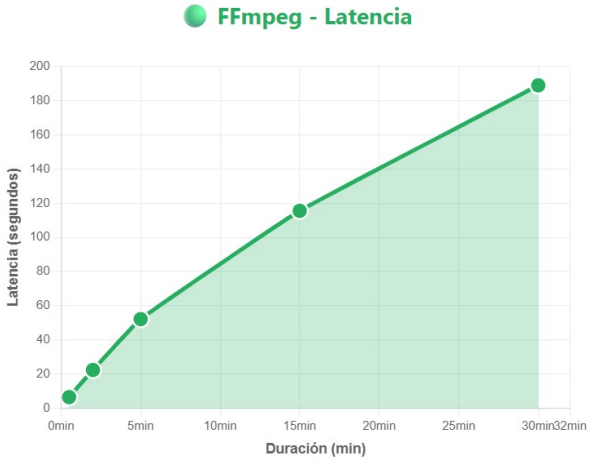


Figura 78. Reducción de ruido de un audio: Crecimiento de la latencia total variando la duración del video

La latencia incrementa con la duración del audio, ya que los archivos más largos contienen un mayor número de muestras a analizar y mayor cantidad de ruido a reducir. Además, el aumento del tamaño del archivo incrementa el tiempo de entrada/salida y el overhead general del sistema. Esta operación presenta el mayor costo temporal entre todas las evaluadas hasta ahora, estableciendo un tiempo máximo en todas las duraciones analizadas.

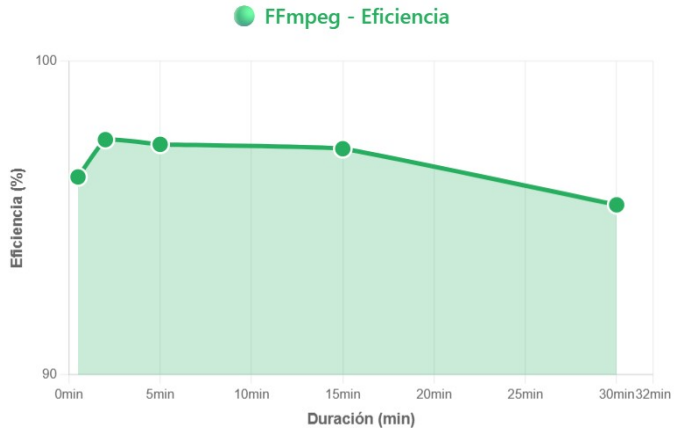


Figura 79. Reducción de ruido de un audio: Crecimiento de la eficiencia variando la duración del video

La eficiencia presenta una disminución, similar a la observada en la mayoría de las operaciones analizadas, aunque con un ritmo relativamente lento. Este decrecimiento se debe a que el procesamiento local de volúmenes mayores de datos exige más recursos computacionales. Dado que los recursos disponibles son limitados, al aumentar tanto el tiempo de procesamiento como el overhead, este último se ve incrementado de manera más significativa al acercarse a la saturación del sistema.

Estos resultados permiten concluir que la duración, y por ende el tamaño, del audio sí impacta en la latencia y el tiempo de procesamiento en la operación de reducción de ruido.

Resolución	Tecnología	Latencia Total (s)	Tiempo Procesamiento (s)	Tiempo I/O (s)	Overhead (s)	Eficiencia Procesamiento (%)
720p	FFmpeg	2.895	2.887	0.008	0.008	99.72%

1080p	FFmpeg	2.889	2.880	0.008	0.009	99.69%
4K	FFmpeg	2.903	2.896	0.007	0.007	99.76%

Tabla 94. Reducción de ruido de un audio: Resultados de métricas de rendimiento variando la resolución del video

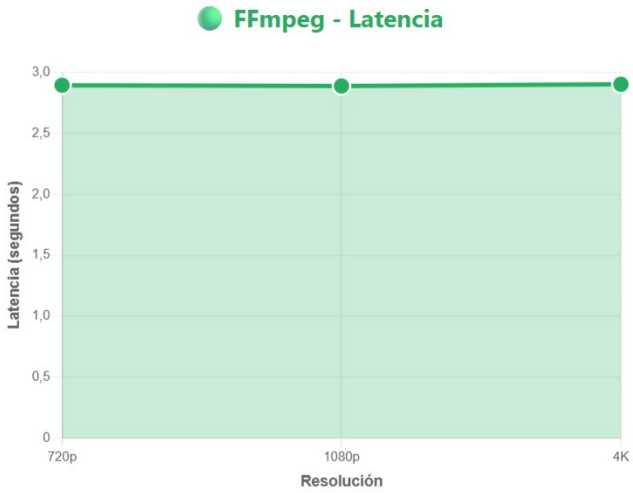


Figura 80. Reducción de ruido de un audio: Crecimiento de la latencia total variando la resolución del video

Al variar la resolución del video del cual proviene el audio, la latencia se mantiene prácticamente constante, como se puede observar en la gráfica y en la tabla de resultados.

Esto se refleja también en la eficiencia, que muestra un comportamiento estable en la gráfica inferior.

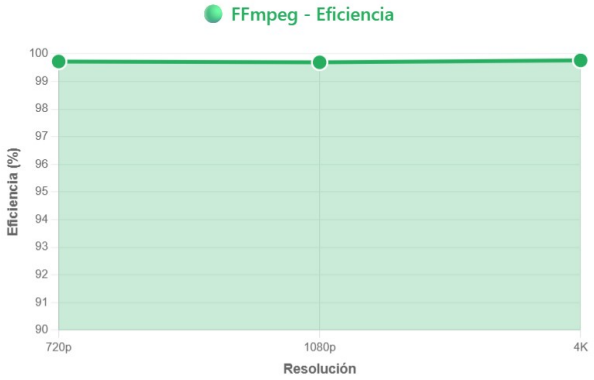


Figura 81. Reducción de ruido de un audio: Crecimiento de la eficiencia variando la resolución del video

Por lo tanto, se puede concluir que, tal como se esperaba, la reducción de ruido no se ve afectada por la resolución original del video, sino que por el tamaño y ruido del audio resultante.

Recursos

Duración	Uso_CPU (%)	Uso_RAM (MB)	Almacenamiento (MB)
30 segundos	199.2	553.30	2.18
2 minutos	183.43	554.54	7.68

5 minutos	190.24	557.34	16.33
15 minutos	181.12	562.62	48.83
30 minutos	198.90	575.11	207.24

Tabla 95. Reducción de ruido de un audio: Resultado de métricas de recursos variando la duración del video

Resolución	Uso_CPU (%)	Uso_RAM (MB)	Almacenamiento (MB)
720p	145.55	34.65	16.24
1080p	157.87	34.32	16.33
4K	155.68	34.46	15.62

Tabla 96. Reducción de ruido de un audio: Resultado de métricas de recursos variando la resolución del video

Coste – FFmpeg

Cabe recordar que el precio de la EC2 es de 0,0216 € por hora, lo que equivale aproximadamente a 6×10^{-6} € por segundo.

Duración	Latencia Total (s)	Precio total (€)
30 seg	6.50	3.90×10^{-5}
2 min	22.38	1.34×10^{-4}
5 min	52.17	3.13×10^{-4}
15 min	115.58	6.93×10^{-4}
30 min	189.07	1.13×10^{-3}

Tabla 97. Reducción de ruido de un audio: Resultado métricas de coste FFmpeg variando la duración del video

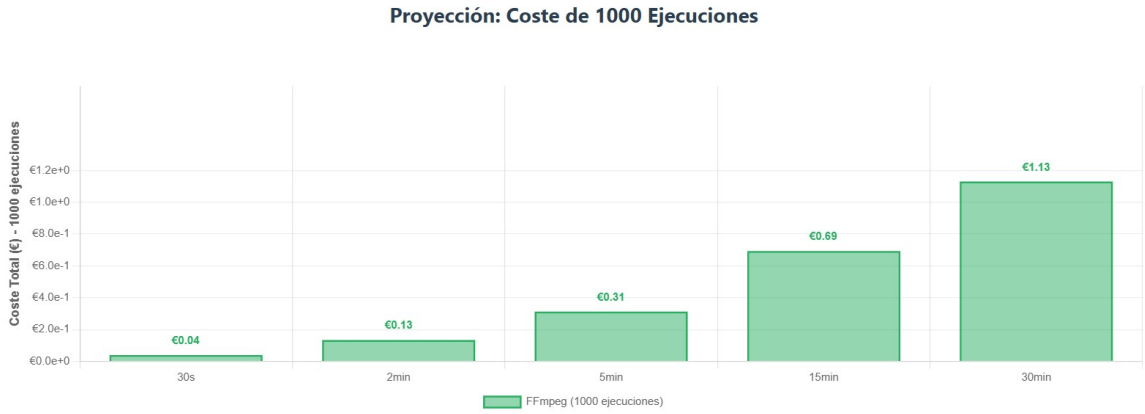


Figura 82. Reducción de ruido de un audio: Coste total variando la duración del video

La gráfica muestra claramente cómo el coste aumenta con la duración del video. Esto se debe a que el archivo es más grande y, como se ha observado, la latencia crece considerablemente con la duración, lo que mantiene la instancia EC2 encendida por más tiempo y, por lo tanto, incrementa el coste.

Resolución	Latencia Total (s)	Precio total (€)
720p	2.895	1.74×10^{-5}
1080p	2.889	1.73×10^{-5}
4K	2.903	1.74×10^{-5}

Tabla 98. Reducción de ruido de un audio: Resultado métricas de coste FFmpeg variando la resolución del video

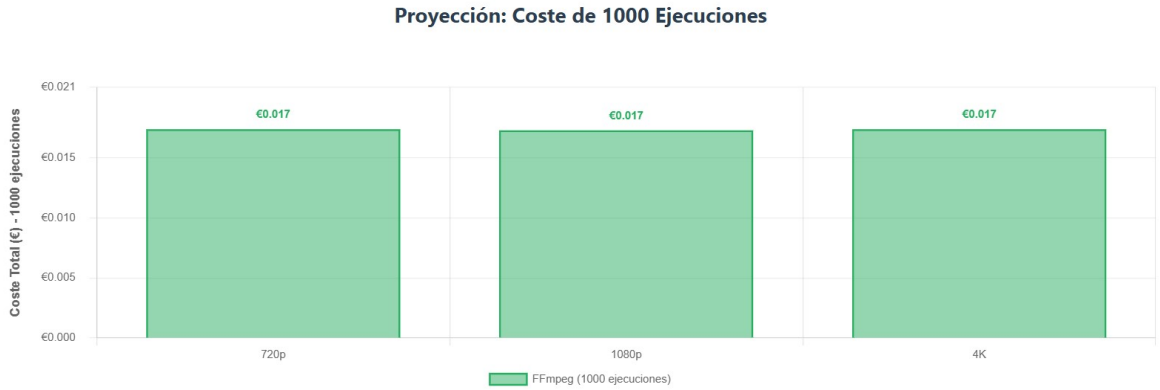


Figura 83. Reducción de ruido de un audio: Coste total variando la resolución del video

En el caso del coste respecto a la resolución, se puede observar que no varía, lo cual es totalmente lógico por las mismas razones explicadas previamente sobre la latencia y la eficiencia en la reducción de ruido.

9.3.1.7 Resumen y conclusiones

En este apartado se presentan y resumen las conclusiones más relevantes, previamente explicadas y mostradas en las secciones anterior junto con los resultados numéricos obtenidos.

9.3.1.7.1 Resumen del análisis realizado

Comportamiento de las métricas en AWS y FFmpeg respecto la variación de la duración del video/audio:

Operación	Métricas Clave	Comportamiento en FFmpeg	Comportamiento en AWS	Factores Determinantes
Separar audio del video	Throughput	Decrece suavemente con duración	Caída drástica	Tamaño archivo y escalabilidad lineal vs. exponencial
	Latencia	Lineal (predecible)	Crecimiento exponencial	Overhead I/O AWS (S3) vs. procesamiento directo
	Eficiencia	Decrecimiento suave (lineal)	Decrecimiento exponencial	Saturación progresiva recursos locales (FFmpeg) vs. overhead I/O crítico (AWS)

	Coste	Mínimo (€0.01/1000 ejecuciones. 30 min)	Muy alto (€459/1000 ejecuciones. 30min)	Coste marginal local vs. precios por uso AWS (MediaConvert)
Normalizar audio	Throughput	Decrece con duración	Caída consistente	Reducción progresiva del throughput en FFmpeg y AWS
	Latencia	Lineal	Lineal (valores absolutos altos)	Procesamiento eficiente audio vs. overhead S3
	Eficiencia	83-99%, decrecimiento muy lento	Decrecimiento lineal	Amortización overhead (FFmpeg) vs. I/O dominante >50% (AWS)
	Coste	Prácticamente Cero	Aumento lineal con duración	Precios por minuto AWS de MediaConvert
Detección cambios de escena	Throughput	Decrecimiento progresivo con duración	Decrecimiento progresivo con duración	Carga computacional creciente en FFmpeg + Factor mejora de FFmpeg respecto AWS se reduce mucho en duraciones largas
	Latencia	Comportamiento lineal con tendencia exponencial en duraciones largas (alta por carga CPU)	Comportamiento lineal con tendencia exponencial en duraciones largas (valores absolutos muy altos)	Operaciones por píxel (FFmpeg) + AWS Rekognition
	Eficiencia	Aumenta con duración	Decrece (casi exponencial) con duración por el tamaño del archivo	Overhead relativo ↓ en larga duración (FFmpeg) vs. I/O ↑ (AWS)
	Coste	Bajo	Muy alto (Rekognition)	CPU local vs. coste por análisis AWS
Extraer frame de un video	Throughput	Decrecimiento leve con duración	Decrecimiento acelerado con duración	Resistencia FFmpeg vs. dependencia I/O crítica en AWS. Factor de mejora de FFmpeg va en crecimiento con duración.
	Latencia	Bastante estable	Comportamiento cercano al exponencial con duración	I/O local mínimo vs. transferencia S3 dominante
	Eficiencia	Decrece por mayor tamaño del archivo y frame, que hay que gestionar en memoria o disco.	61% a 33.5%, Decrece más rápidamente	Gestión buffers memoria (FFmpeg) + serialización contenedores (AWS)

	Coste	Prácticamente cero	Muy alto (MediaConvert)	Almacenamiento local vs. almacenamiento S3 + coste minuto AWS

Tabla 99. Resumen métricas FFmpeg y AWS según la variación de la duración

Comportamiento de las métricas en AWS y FFmpeg respecto la variación de la resolución:

Operación	**Métrica**	**Comportamiento FFmpeg (Resolución)**	**Comportamiento AWS (Resolución)**	**Factores Clave**
Separar audio del video	Throughput	Máximo en 1080p, y bastante constante en general	Constante mínimo (0.02 arch/s)	Decodificación video (FFmpeg) vs. overhead fijo transferencia (AWS)
	Latencia	Constante	T. procesamiento puro estable (~7s), I/O depende de tamaño	Resolución no afecta prácticamente
	Eficiencia	Ligera caída en 4K (88.5%)	Poca variación, debido a tamaños parecidos de los archivos de entrada	Saturación RAM/CPU (FFmpeg) vs. I/O dominante (AWS)
	Coste	Marginal (prácticamente 0)	Aumenta con resolución (MediaConvert, precio por minuto y resolución)	Hardware existente vs. precio por resolución
Normalizar audio	Throughput	Estable (no afectado por resolución del video de origen del audio)	Estable (no afectado por resolución del video de origen del audio)	Audio independiente de resolución video de origen del audio
	Latencia	Estable (variación <1%)	T. procesamiento puro estable (~7s), I/O fluctuante	Procesamiento eficiente vs. red AWS
	Eficiencia	>98% (alta en todas resoluciones)	Va en decrecimiento por tiempo de I/O	Pipeline local vs. serialización AWS
	Coste	Prácticamente nulo	Constante (no depende de resolución)	Servicios audio AWS sin costo por resolución
Detección cambios escenas	Throughput	Caída drástica: 720p(0.29)→4K(0.04 arch/s)	Caída moderada: 720p(0.037)→4K(0.02 arch/s)	Carga píxeles $O(n^2)$ (FFmpeg) vs. overhead infraestructura (AWS)
	Latencia	Aumento pronunciado (limitación recursos del hardware)	Aumento moderado (escalado recursos pero overhead I/O)	Operaciones/frame (FFmpeg) vs. transferencia datos (AWS)

	Eficiencia	Crece debido a la ↓ impacto overhead respecto T. procesamiento puro que ↑ con resolución.	Decrece por el overhead de transferencia de datos (red)	Uso CPU local vs. optimización Rekognition
	Coste	Aumenta con resolución (tiempo EC2)	Constante (Rekognition sin costo por resolución)	Coste computación local vs. precio fijo por análisis
Extraer frame	Throughput	Decrecimiento por tener ↑ pixeles a ↑ resolución.	Decrecimiento por tener ↑ pixeles a ↑ resolución.	Acceso directo a keyframes (FFmpeg) vs. transferencia completa (AWS)
	Latencia	Crecimiento por tener ↑ pixeles a ↑ resolución.	Crecimiento por tener ↑ pixeles a ↑ resolución.	Decodificación píxeles (FFmpeg) vs. Servicios cloud + red (AWS)
	Eficiencia	Ligera disminución (93% a 90%)	Mejora en 4K, debido a que el % del T. procesamiento puro es ↑ que overhead.	Overhead gestión memoria/disco (FFmpeg) vs. proporción procesamiento (AWS)
	Coste	Aumenta con resolución (tiempo EC2)	Aumento significativo (MediaConvert, precio por minuto y resolución)	Tiempo procesamiento local vs. Precio por minuto y resolución AWS

Tabla 100. Resumen métricas FFmpeg y AWS según la variación de la resolución

9.3.1.7.2 Conclusiones

En este apartado se presentan las conclusiones generales sobre el impacto de la variación de duración y resolución en las operaciones, mostrando la comparación entre AWS y FFmpeg, así como la recomendación final sobre cuál de las dos tecnologías es la mejor opción según los resultados obtenidos.

Impacto de la Duración

- Throughput: Disminuye en todas las operaciones al aumentar la duración (A mayor tamaño, mayor tiempo de procesamiento por archivo y, por lo tanto, menor cantidad de archivos procesados por segundo).
- Latencia
    - FFmpeg: Crecimiento prácticamente siempre lineal (predecible y eficiente).
    - AWS: Crecimiento exponencial o aproximadamente exponencial en operaciones con video (ej: separar audio), especialmente crítico en archivos largos (30 min).
- Coste en AWS: Aumenta drásticamente con la duración, mientras que en FFmpeg es muy bajo e, incluso en muchos casos, casi nulo.

Impacto de la Resolución

- Throughput
    - FFmpeg: Se ve más afectado (ej: detección de escenas en 4K satura CPU/memoria).
    - AWS: Menor impacto (limitado por overhead de I/O y red, no por potencia).
- Latencia
    - FFmpeg: Aumenta en tareas intensivas que analizan el contenido del video y, por lo tanto, la resolución tiene un impacto en ellas (detección de escenas, extracción de frames).
    - AWS: Menor impacto (el procesamiento puro es constante, la variación depende de I/O).
- Coste
    - FFmpeg: El coste sigue siendo bajo, aunque aumenta ligeramente al analizar videos de alta resolución, debido al mayor tiempo de procesamiento, mayor uso de la EC2 y, por lo tanto, mayor coste económico.
    - AWS: Aumenta con la resolución en operaciones de video y que usan MediaConvert (ej: separar audio de video, extraer frames), pero no en otras (ej: normalizar audio, detección cambios de escena que usa Rekognition).

Comparación FFmpeg vs AWS

- Rendimiento (Throughput/Latencia)
    - FFmpeg: Superior en todas las operaciones (excepto detección de escenas en 4K, pero sigue dando tiempos competitivos con AWS).

      Esto es gracias a que evita overheads de red/virtualización y aprovecha mejor los recursos locales. La proximidad de los datos ofrece ventaja.
    - AWS: Alto overhead de I/O (alcanza valores superiores del 50% de la latencia es transferencia S3).
- Eficiencia: FFmpeg mantiene un comportamiento estable y predecible mientras que AWS tiene gran dependencia del I/O y overhead, disminuyendo eficiencia.
- Coste:
    - FFmpeg: Mínimo (solo incluye el coste de la CPU local, que se traduce en el coste de la EC2).
    - AWS: Mucho más costoso, ya que se suman el coste de la EC2, el almacenamiento y las operaciones en S3, además de depender de servicios caros como MediaConvert o Rekognition.

Recomendaciones por operación

- Extracción del audio de un video: **FFmpeg**, debido a que AWS tiene latencia exponencial y coste prohibitivo. Resolución no afecta a FFmpeg.

- Normalización de audio: **FFmpeg**, debido al procesamiento ligero de FFmpeg. AWS no mejora latencia, añade coste I/O innecesario y tiene un coste muy alto.
- Detección de cambios de escena en un video:
    - **FFmpeg**: Óptimo en resoluciones medias/bajas
    - **AWS**: Mejor en 4K o resoluciones altas (FFmpeg satura recursos locales, AWS sostiene rendimiento mínimo).
    - FFmpeg ofrece un rendimiento adecuado incluso con videos en 4K, ya que la latencia obtenida en las pruebas fue menor que la de AWS, que pierde tiempo en operaciones de I/O. Sin embargo, FFmpeg puede volverse ineficiente con archivos de muy alta calidad y gran tamaño. AWS, por su parte, resulta económicamente muy costoso debido a servicios como Rekognition. En general, FFmpeg es la opción recomendada. AWS solo sería conveniente para archivos grandes si se cuentan con recursos económicos suficientes, o si no se dispone de hardware local con GPU, escenario en el que FFmpeg sigue siendo superior.
- Extracción de un frame de un video: **FFmpeg** debido a su latencia constante (no depende de duración), AWS tiene coste elevado y se reduce la eficiencia con la resolución.

Ganador global: **FFmpeg** es la opción ganadora para todas las operaciones por ser más rápida, eficiente y económica, salvo casos específicos donde la nube sea la única opción para manejar grandes volúmenes en paralelo.

9.3.2 Evaluación impacto de transcripción y diarización

En esta sección se mostrará cómo se han establecido los parámetros generales de audio (frecuencia de muestreo, número de canales y profundidad de bits), así como los parámetros de reducción de ruido y de normalización de audio. Para determinar estos valores, se estudió el WER de la transcripción obtenida al variar cada parámetro individualmente, manteniendo las demás constantes, así como la precisión de la diarización obtenida. Esto permitió identificar la mejor combinación de parámetros para lograr una transcripción y una diarización lo más precisas posible.

La precisión de la diarización se representa con la precisión global y precisión por Speaker o hablante.

1. Precisión global: Es el porcentaje ponderado por el número de segmentos en toda la reunión, calculada como Precisión Global = (Total segmentos correctos en toda la reunión / Total segmentos) x 100.

   Refleja el rendimiento general del sistema.

2. Precisión por speaker: Es un promedio no ponderado de las precisiones individuales de cada speaker, calculada como Precisión speakerX = (Prec. spkX reunión_1+Prec. SpkX reunión_2+Prec. spkX reunión_3+Prec. spkX reunión_4) / 4

   Muestra el comportamiento por participante.

Al final de la sección se realiza una comparación entre el pipeline de AWS y FFmpeg, en la que se analizan el WER y el CER de la transcripción obtenida, la precisión de la diarización y el tiempo de ejecución de cada uno.

Como recordatorio del significado de las métricas y para facilitar la interpretación de los resultados, el **Word Error Rate (WER)** es una métrica que mide el porcentaje de errores en una transcripción comparada con el texto original. Dichos errores pueden ser palabras omitidas, cambiadas o añadidas. Por ejemplo, si se obtiene un **WER de 0.2**, significa que aproximadamente un **20% de las palabras son incorrectas**, mientras que el **80% restante se han transcrito correctamente**.

De manera similar, el **Character Error Rate (CER)** evalúa los errores, pero a nivel de caracteres en lugar de palabras.

En la práctica, **la normalización de los textos** (por ejemplo, homogenizar mayúsculas/minúsculas, eliminar signos de puntuación no lingüísticos, unificar espacios, tratar diacríticos, o acordar formatos para números, horas, siglas y símbolos) ayuda a que el WER refleje mejores errores lingüísticos reales y no solo diferencias de formato. No obstante, **no existe un estándar global de normalización** para todos los dominios. Hay múltiples maneras válidas de representar la misma información ("10:30" vs. "10.30", "RR. HH." vs. "Recursos Humanos", "IA" vs. "inteligencia artificial", "50 %" vs. "cincuenta por ciento", etc.).

En este proyecto se aplicó a **ambos textos** (referencia e hipótesis) **una normalización general y consistente**, suficiente para comparar sistemas de manera justa, pero sin llegar a un pipeline de normalización profundamente específico del dominio. Esto es debido a que **la prioridad es la comparabilidad entre WERs**, no la minimización absoluta de la métrica. Aun así, es razonable esperar que, tras un estudio más exhaustivo y la adopción de reglas de normalización avanzadas alineadas con AMI corpus (p. ej., mapeo sistemático de siglas, reglas de numerales y horas), los valores de WER podrían descender de forma apreciable respecto a los aquí reportados.

9.3.2.1 Grabaciones de reuniones usadas en el estudio

Como se explicó anteriormente en el diseño de las pruebas, se utilizarán reuniones (videos y audios) extraídas del dataset *AMI Corpus*. También se mencionó que algunas grabaciones se realizaron con micrófono de diadema (headset) y otras con micrófono de solapa (lavalier o lapel mic).

Se han seleccionado cuatro reuniones: Dos con micrófono de diadema y dos con micrófono de solapa. En cada tipo de micrófono, una reunión corresponde a un entorno organizado (con menos ruido de fondo) y la otra a un entorno con mayor ruido de fondo.

De este modo, será posible calcular un promedio y determinar la combinación óptima de parámetros para un conjunto diverso de reuniones con distintos niveles de ruido, en lugar de limitarse a un solo tipo de grabación.

9.3.2.2 Frecuencia de muestreo

En este apartado se estudiará cómo afecta la frecuencia de muestreo a la transcripción y diarización.

En primer lugar, se analiza cómo afecta a la transcripción, mostrando en la siguiente tabla el WER obtenido en cada reunión según la configuración de la frecuencia de muestreo.

Frec. de muestreo (Hz) / Reunión	16000	22050	32000	44100	48000
Reunión_1	0,281	0,292	0,293	0,293	0,293
Reunión_2	0,273	0,284	0,286	0,286	0,286
Reunión_3	0,308	0,312	0,311	0,31	0,31
Reunión_4	0,366	0,374	0,373	0,375	0,374
Media	**0.307**	0.316	0.316	0.316	0.316

Tabla 101. Tasa de error de palabras (WER) en la transcripción de reuniones según la frecuencia de muestreo

16000 Hz tiene el menor WER promedio, lo que indica mejor precisión en transcripción a esta frecuencia.

Para frecuencias ≥ 22,050 Hz, el WER se estabiliza en 0.316, sin mejoras significativas al aumentar la frecuencia. Aunque todas las frecuencias de muestreo se redondean a 0,316, existen diferencias mínimas entre ellas que no resultan apreciables, como por ejemplo valores del orden de 0,3155 frente a 0,3157.

A continuación, se analiza cómo afecta la frecuencia de muestreo a la diarización, mostrando en la siguiente tabla la precisión global y por Speaker (promedio de las 4 Reuniones) según la configuración de la frecuencia de muestreo.

Frec. de muestreo (Hz) / Speaker	16000	22050	32000	44100	48000
Speaker_0	69.49%	73.42%	76.93%	72.88%	72.65%
Speaker_1	90.59%	93.84%	93.04%	93.25%	93.24%
Speaker_2	96.24%	91.99%	92.88%	92.27%	92.65%
Speaker_3	89.67%	90.23%	90.71%	89.91%	89.47%
Precisión global	93.64%	94.15%	**94.56%**	93.81%	93.87%

Tabla 102. Precisión de la diarización según la frecuencia de muestreo

La frecuencia de muestreo de 32 kHz proporciona la máxima precisión global de diarización (94,56%), aunque la diferencia respecto a 16 kHz (93,64%) es menor al 1%, lo que hace que ambas frecuencias sean prácticamente equivalentes para este propósito.

Dado que **16 kHz** presenta el menor WER y genera archivos de menor tamaño, se considera **la opción más equilibrada**, optimizando tanto la precisión de la transcripción como la eficiencia del procesamiento.

9.3.2.3 Número de canales

En esta sección se estudiará cómo afecta el número de canales a la transcripción y diarización.

En primer lugar, se analiza cómo afecta a la transcripción, mostrando en la siguiente tabla el WER obtenido en cada reunión según la configuración del número de canales.

Número canales / **Reunión**	**1**	**2**
Reunión_1	0,281	0,878
Reunión_2	0,273	0,882
Reunión_3	0,308	0,952
Reunión_4	0,366	0,918
Media	**0.307**	0.908

Tabla 103. Tasa de error de palabras (WER) en la transcripción de reuniones según el número de canales

Un solo canal tiene un WER significativamente menor (0.307 vs 0.908), lo que indica una transcripción 3 veces más precisa que con dos canales.

La diferencia es consistente en todas las reuniones (WER en 2 canales siempre >0.878). Los dos canales introducen distorsión o interferencia que empeora drásticamente el reconocimiento de voz.

A continuación, se analiza cómo afecta el número de canales a la diarización, mostrando en la siguiente tabla la precisión global y por Speaker (promedio de las 4 Reuniones) según la configuración del número de canales.

Número canales / **Speaker**	**1**	**2**
Speaker_0	69.49%	89.54%
Speaker_1	90.59%	85.07%
Speaker_2	96.24%	96.66%
Speaker_3	89.67%	91.06%
Precisión global	93.64%	**93.93%**

Tabla 104. Precisión de la diarización según el número de canales

En la precisión global, el uso de dos canales supera ligeramente al de un solo canal. Si se observan las precisiones individuales por hablante, en general son muy similares, con la excepción del hablante 0, cuyo porcentaje al usar un canal es más bajo. Esto puede explicarse porque el speaker 0 tiene menos segmentos, lo que facilita obtener un porcentaje reducido.

Existe una razón técnica por la cual el uso de dos canales mejora la diarización. Se reduce el solapamiento de voz (overlap). Cuando dos hablantes hablan simultáneamente, la señal estéreo permite separar parcialmente los componentes frecuenciales asociados a cada dirección de origen, lo que disminuye los errores en segmentos con voz superpuesta, entre otros beneficios.

No obstante, considerando que el uso de dos canales introduce una gran desventaja en la precisión de la transcripción y que la mejora en la diarización respecto a un solo canal es marginal, **la recomendación es emplear un único canal** en el sistema. De este modo se garantiza un equilibrio óptimo entre calidad de transcripción y precisión en la diarización. Además, en este caso particular, las reuniones de Microsoft Teams que se transcriben ya se graban en formato mono y no estéreo, lo que refuerza aún más la elección de un solo canal.

9.3.2.4 Profundidad de bits

En este apartado se estudiará cómo afecta la profundidad de bits a la transcripción y diarización. Solo se analizarán las profundidades de 16 y 24 bits (no se considerará la de 32 bits), ya que, como se explicó anteriormente, por motivos técnicos y teóricos se utilizará el formato de audio FLAC, el cual solo admite estas dos profundidades.

En primer lugar, se analiza cómo afecta a la transcripción, mostrando en la siguiente tabla el WER obtenido en cada reunión según la configuración de la profundidad de bits.

Profundidad de bits / **Reunión**	**16 bits**	**24 bits**
Reunión_1	0,253	0,254
Reunión_2	0,258	0,258
Reunión_3	0,334	0,333
Reunión_4	0,374	0,374
Media	**0.305**	**0.305**

Tabla 105. Tasa de error de palabras (WER) en la transcripción de reuniones según la profundidad de bits

Ambas profundidades de bits tienen el mismo WER promedio y las diferencias en los valores individuales son mínimas (≤0.001) y no afectan el resultado global.

Por lo tanto, elegir 16 bits o 24 bits no impacta la precisión de transcripción, pero como recomendación se puede optar por 16 bits para ahorrar almacenamiento sin sacrificar calidad.

A continuación, se analiza cómo afecta la profundidad de bits a la diarización, mostrando en la siguiente tabla la precisión global y por Speaker (promedio de las 4 Reuniones) según la configuración de la profundidad de bits.

Profundidad de bits / **Speaker**	**16 bits**	**24 bits**
Speaker_0	89.63%	90.36%
Speaker_1	89.18%	88.82%
Speaker_2	88.27%	88.22%
Speaker_3	90.87%	89.92%
Precisión global	**90.65%**	90.28%

Tabla 106. Precisión de la diarización según la profundidad de bits

Los 16 bits mantienen ventaja sobre los 24 bits, tanto en la precisión global como en la mayoría de las mediciones por hablante.

Teniendo en cuenta que las profundidades de 16 y 24 bits ofrecen una precisión de transcripción equivalente, y que la configuración de 16 bits presenta una ligera ventaja en la precisión de diarización, además de reducir el tamaño de los archivos, optimizando así el almacenamiento y acelerando el procesamiento, **la recomendación técnica es utilizar una profundidad de 16 bits.**

9.3.2.5 Reducción de ruido

En este apartado se analiza la optimización de los parámetros de reducción de ruido: Frecuencia de corte (highpass_freq), nivel de reducción (noise_reduction) y umbral de ruido (noise_floor).

Se evaluó el WER obtenido en transcripciones para distintas configuraciones de estos parámetros, seleccionando los rangos de valores que mostraron mejores resultados preliminares.

Dado que los audios presentan ruido heterogéneo y variable, se calculó el WER promedio sobre los cuatro audios muestreados de las reuniones. Los parámetros que minimicen este WER promedio se implementarán en el producto final.

Frecuencia de corte (highpass_freq)

A continuación, se presentan los valores de WER obtenidos en las transcripciones para distintas frecuencias de corte.

Frecuencia de corte (Hz)	**20**	**77**	**100**	**135**	**150**	**155**	**163**	**192**	**210**	**250**	**500**	**750**	**1000**
Reunión_1	0,348	0,35	0,343	0,331	0,322	0,322	0,318	0,32	0,317	0,31	0,32	0,33	0,364
Reunión_2	0,356	0,357	0,356	0,352	0,35	0,349	0,351	0,352	0,349	0,352	0,357	0,37	0,38
Reunión_3	0,264	0,257	0,253	0,244	0,243	0,241	0,241	0,238	0,235	0,235	0,236	0,244	0,267
Reunión_4	0,263	0,258	0,256	0,251	0,251	0,25	0,248	0,245	0,243	0,238	0,238	0,244	0,265
Media	0,30775	0,3055	0,302	0,2945	0,2915	0,2905	0,2895	0,28875	0,286	**0,28375**	0,28775	0,297	0,319

Tabla 107. Tasa de error de palabras (WER) en la transcripción de reuniones a diferentes frecuencias de corte

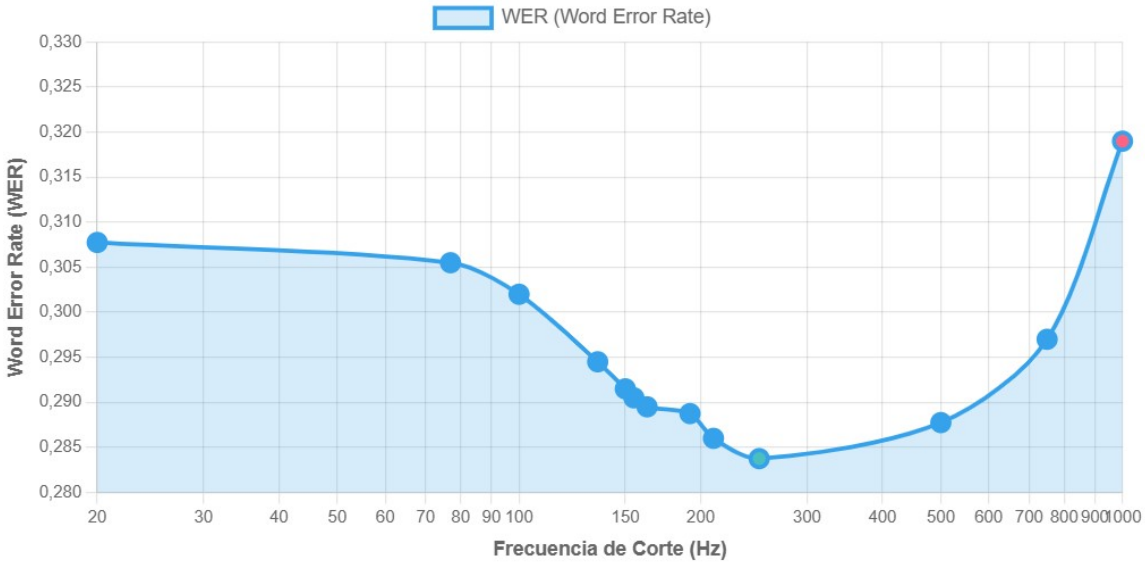


Figura 84. Tasa de error de palabras (WER) en la transcripción de reuniones a diferentes frecuencias de corte

Valor mínimo: A una frecuencia de corte de 250 Hz

Nivel de reducción de ruido (noise_reduction)

A continuación, se presentan los valores de WER obtenidos en las transcripciones para distintos niveles de reducción de ruido.

Noise reduction (dB)	**1**	**13**	**19**	**22**	**23**	**24**	**25**	**26**	**38**	**50**
Reunión_1	0,251	0,236	0,236	0,236	0,235	0,234	0,236	0,239	0,244	0,242
Reunión_2	0,255	0,24	0,237	0,235	0,233	0,235	0,236	0,237	0,243	0,244
Reunión_3	0,328	0,31	0,311	0,315	0,318	0,319	0,319	0,322	0,321	0,317
Reunión_4	0,369	0,353	0,348	0,353	0,357	0,357	0,358	0,361	0,37	0,371
Media	0,30075	0,28475	**0,283**	0,28475	0,28575	0,28625	0,28725	0,28975	0,2945	0,2935

Tabla 108. Tasa de error de palabras (WER) en la transcripción de reuniones a diferentes niveles de reducción de ruido

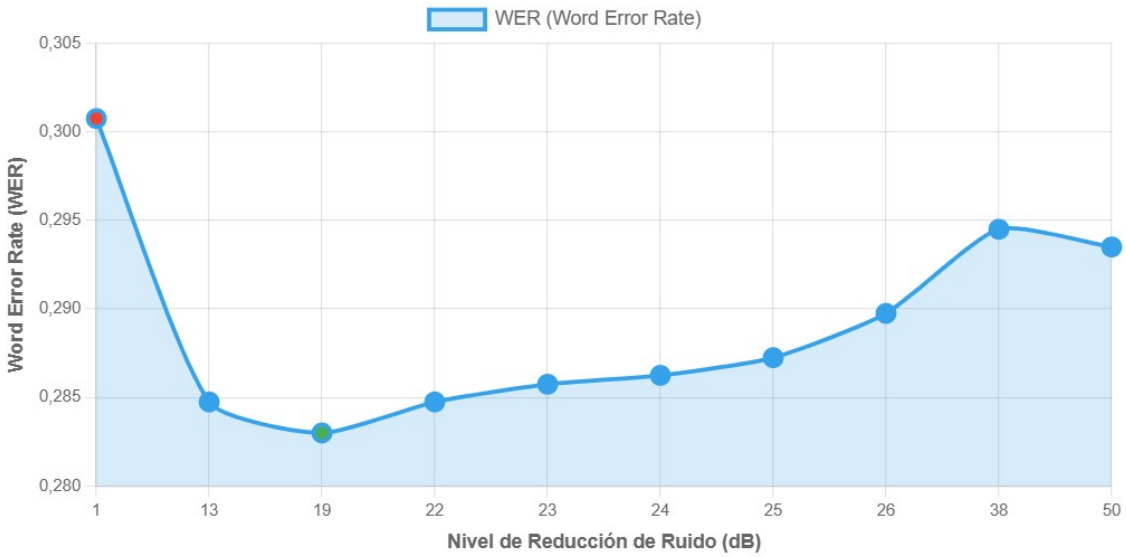


Figura 85. Tasa de error de palabras (WER) en la transcripción de reuniones a diferentes niveles de reducción de ruido

Valor mínimo: Nivel de reducción de ruido de 19 dB.

Suelo de ruido (noise floor)

A continuación, se presentan los valores de WER obtenidos en las transcripciones para distintos suelos de ruido.

Noise floor (dB)	**-80**	**-65**	**-50**	**-43**	**-39**	**-35**	**-20**	**-80**
Reunión_1	0,25	0,252	0,238	0,235	0,234	0,232	0,252	0,25
Reunión_2	0,252	0,25	0,24	0,236	0,237	0,236	0,252	0,252
Reunión_3	0,335	0,32	0,313	0,308	0,319	0,328	0,347	0,335
Reunión_4	0,366	0,348	0,348	0,352	0,362	0,365	0,384	0,366
Media	0,30075	0,2925	0,28475	**0,28275**	0,288	0,29025	0,30875	0,30075

Tabla 109. Tasa de error de palabras (WER) en la transcripción de reuniones a diferentes suelos de ruido

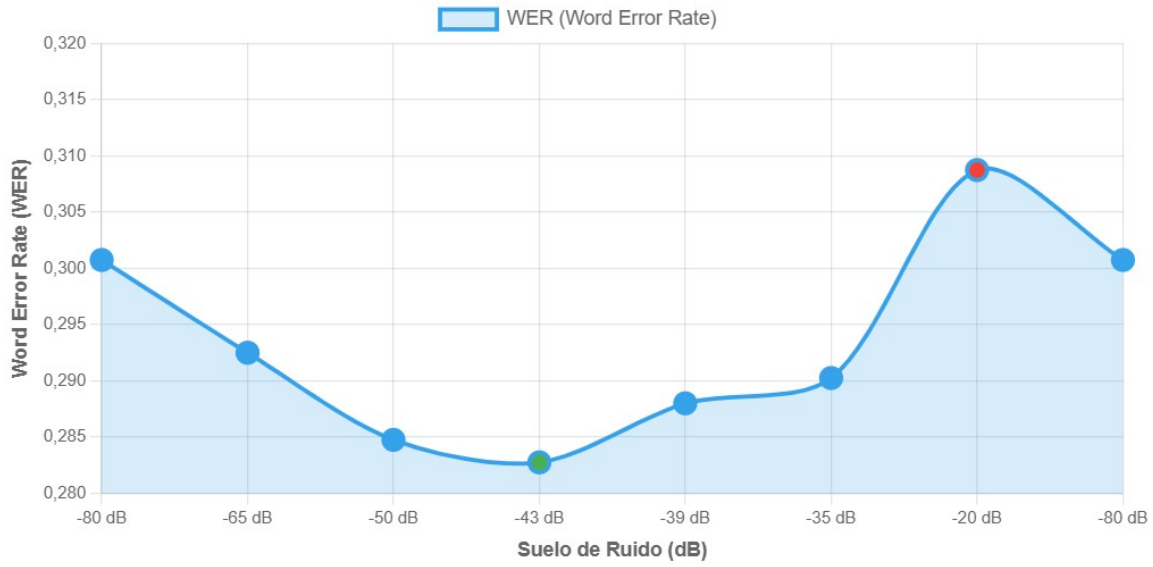


Figura 86. Tasa de error de palabras (WER) en la transcripción de reuniones a diferentes suelos de ruido

Valor mínimo: Umbral o suelo de ruido a -43 dB.

9.3.2.6 Normalización de audio

En este apartado se analiza la optimización de los parámetros de normalización de audio: Loudness integrada (I_VALUE), pico verdadero (TP_VALUE) y rango dinámico (LRA).

De manera análoga a la operación previa de reducción de ruido, se evaluó el WER en transcripciones para distintas configuraciones de estos parámetros, calculando el WER promedio sobre los cuatro audios muestreados de las reuniones. Los valores que minimicen este WER promedio se implementarán en el producto final.

Loudness integrada (I_VALUE)

A continuación, se presentan los valores de WER obtenidos en las transcripciones para distintos valores del I_VALUE.

I_value (LUFS)	**-70**	**-52**	**-43**	**-36**	**-34**	**-30**	**-23**	**-16**	**-5**
Reunión_1	0,425	0,312	0,276	0,245	0,241	0,238	0,235	0,238	0,285
Reunión_2	0,438	0,325	0,284	0,252	0,248	0,244	0,241	0,245	0,292
Reunión_3	0,445	0,331	0,289	0,256	0,252	0,248	0,245	0,249	0,298
Reunión_4	0,455	0,34	0,295	0,26	0,255	0,25	0,242	0,253	0,305
Media	0,44075	0,327	0,286	0,25325	0,249	0,245	**0,24075**	0,24625	0,295

Tabla 110. Tasa de error de palabras (WER) en la transcripción de reuniones a diferentes I_VALUE

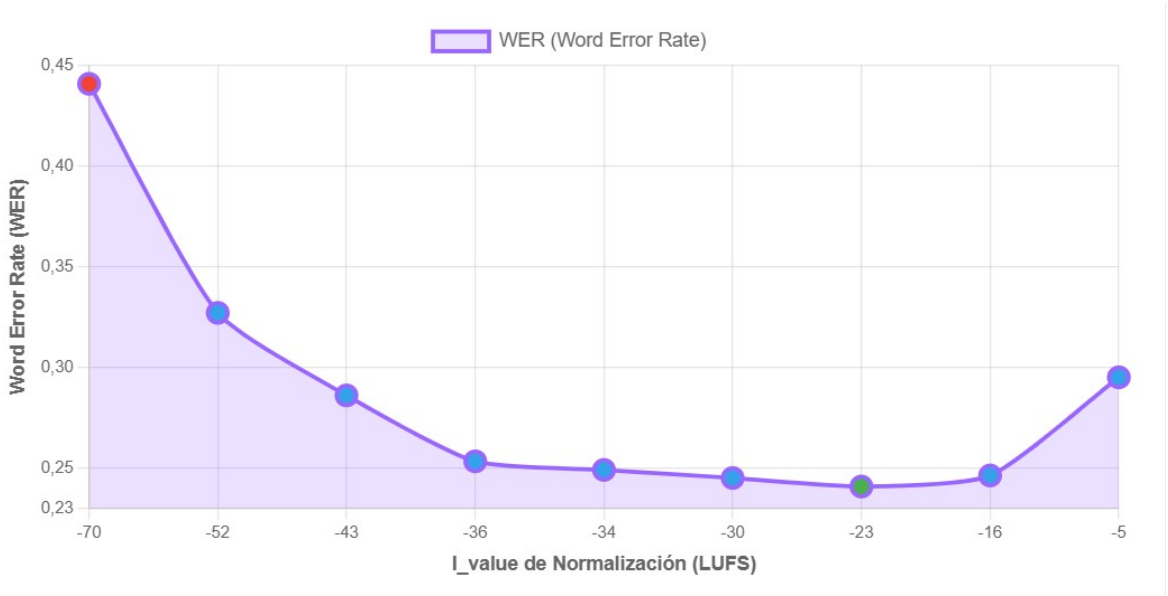


Figura 87. Tasa de error de palabras (WER) en la transcripción de reuniones a diferentes I_VALUE

Valor mínimo: -23 LUFS

Pico verdadero (TP_VALUE)

A continuación, se presentan los valores de WER obtenidos en las transcripciones para distintos valores del TP_VALUE.

TP_value (dBTP)	**-9**	**-7**	**-4,5**	**-2**	**0**
Reunión_1	0,248	0,242	0,238	0,235	0,234
Reunión_2	0,252	0,246	0,242	0,241	0,24
Reunión_3	0,255	0,249	0,245	0,244	0,243
Reunión_4	0,258	0,252	0,248	0,237	0,245
Media	0,25325	0,24725	0,24325	**0,23925**	0,2405

Tabla 111. Tasa de error de palabras (WER) en la transcripción de reuniones a diferentes TP_VALUE

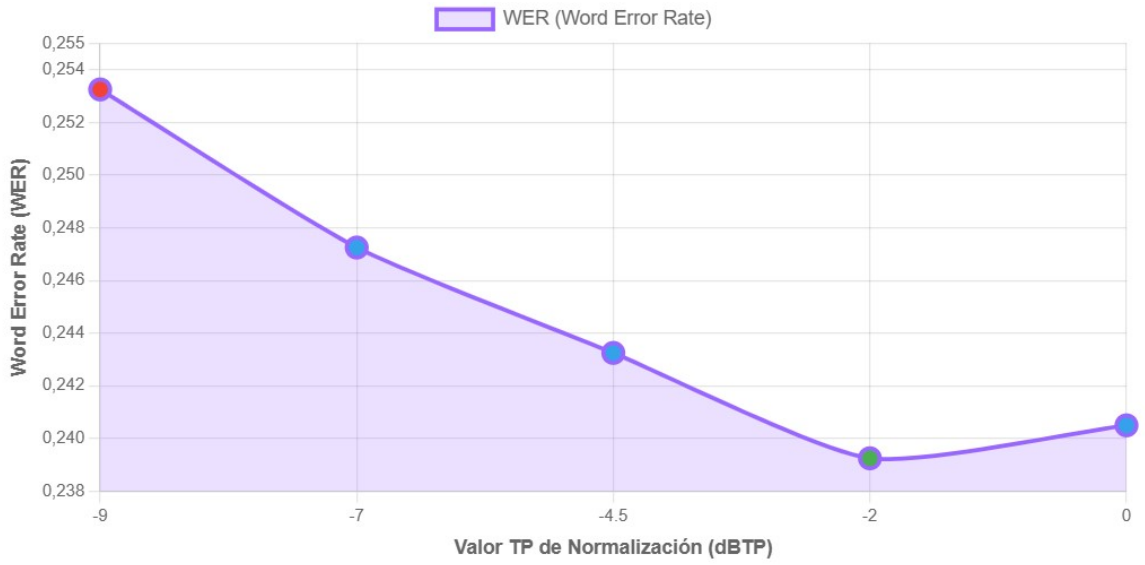


Figura 88. Tasa de error de palabras (WER) en la transcripción de reuniones a diferentes TP_VALUE

Valor mínimo: -2 dBTP.

Rango dinámico (LRA)

A continuación, se presentan los valores de WER obtenidos en las transcripciones para distintos valores del LRA.

LRA (LU)	**1**	**7**	**11**	**13**	**25**	**31**	**38**	**50**
Reunión_1	0,245	0,238	0,235	0,234	0,233	0,234	0,236	0,239
Reunión_2	0,248	0,241	0,238	0,237	0,236	0,237	0,239	0,242
Reunión_3	0,251	0,244	0,241	0,24	0,239	0,24	0,242	0,245
Reunión_4	0,253	0,246	0,243	0,242	0,235	0,243	0,245	0,248
Media	0,24925	0,24225	0,23925	0,23825	**0,23575**	0,2385	0,2405	0,2435

Tabla 112. Tasa de error de palabras (WER) en la transcripción de reuniones a diferentes valores de LRA

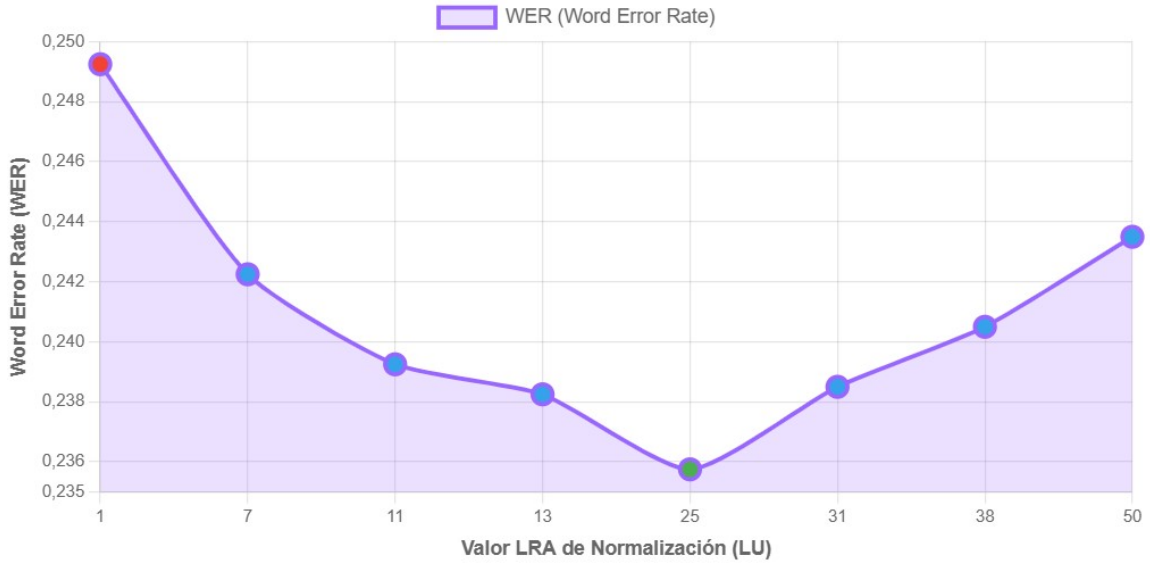


Figura 89. Tasa de error de palabras (WER) en la transcripción de reuniones a diferentes valores de LRA

Valor mínimo: 25 LU.

9.3.2.7 FFmpeg vs AWS

Se han ejecutado tres pipelines: El primero corresponde al baseline, en el que se realiza la transcripción sin aplicar ningún procesamiento de audio, el segundo aplica todo el procesamiento de audio mediante FFmpeg y el tercero utiliza AWS, salvo en la reducción de ruido, que se llevó a cabo con FFmpeg al no estar disponible esta funcionalidad en AWS.

Se configuraron los procesos y parámetros en FFmpeg y AWS de la manera más similar posible para lograr una mayor consistencia entre ambos pipelines.

A continuación, se muestra el WER obtenido en cada pipeline para cada reunión.

Pipeline / **Reunión**	**Baseline**	**FFmpeg**	**AWS**
Reunión_1	0,317	0,232	0,234
Reunión_2	0,326	0,234	0,236
Reunión_3	0,412	0,333	0,352
Reunión_4	0,447	0,368	0,387
Media	0.376	**0.292**	0.302

Tabla 113. FFmpeg vs AWS: Tasa de error de palabras (WER) en la transcripción

FFmpeg tiene mejor rendimiento (media más baja), mientras que AWS muestra mayor variabilidad, con peor rendimiento general.

También cabe mencionar que, en la práctica, los bajos valores de WER obtenidos con AWS se deben a la reducción de ruido aplicada con FFmpeg y que, al omitir este paso, el WER llega a acercarse a valores de 0,5.

Ambas tecnologías muestran una mejora significativa en comparación con el pipeline baseline, el cual no aplica ningún procesamiento previo al audio antes de la transcripción.

A continuación, se muestra la precisión de la diarización obtenida en cada pipeline.

Tecnología / Speaker	FFmpeg	AWS
Speaker_0	81,83%	75,86%
Speaker_1	91,52%	87,95%
Speaker_2	87,14%	84,74%
Speaker_3	88,79%	91,31%
Precisión global	**90,35%**	86,20%

Tabla 114. FFmpeg vs AWS: Precisión de diarización

Respecto a la precisión global, FFmpeg alcanza un 90,35%, mientras que AWS se queda en 86,20%, lo que indica que, en promedio, FFmpeg ofrece un desempeño superior en la asignación correcta de segmentos a los hablantes.

FFmpeg supera a AWS en tres de los cuatro hablantes (Speaker_0, Speaker_1 y Speaker_2). Solo en Speaker_3 AWS obtiene un valor superior.

Por lo tanto, se puede concluir que FFmpeg muestra una mayor consistencia y precisión general en la diarización, aunque AWS puede superar a FFmpeg en casos puntuales de hablantes individuales. Esto sugiere que FFmpeg es la opción más confiable para mantener un alto rendimiento global, mientras que AWS podría ser competitivo en situaciones específicas.

A continuación, se muestra el CER obtenido en cada pipeline para cada reunión.

Tecnología / Reunión	FFmpeg	AWS
Reunión_1	0,104	0,105
Reunión_2	0,105	0,106
Reunión_3	0,150	0,158
Reunión_4	0,166	0,174
Media	**0,131**	0,136

Tabla 115. FFmpeg vs AWS: Tasa de error de caracteres (CER) en la transcripción

La comparación de CER entre FFmpeg y AWS muestra un rendimiento bastante similar en las reuniones iniciales, con diferencias mínimas (0,001 en las reuniones 1 y 2). Sin embargo, a medida que aumenta la complejidad del audio (reuniones 3 y 4), FFmpeg mantiene consistentemente valores de CER más bajos que AWS, con una brecha que llega hasta 0,008 en la reunión 4. En términos generales, la media confirma esta tendencia: 0,131 para FFmpeg frente a 0,136 para AWS. Esto indica que, aunque ambos sistemas son comparables en condiciones más limpias, FFmpeg ofrece mayor precisión en escenarios con mayor dificultad acústica.

9.3.2.7.1 Tiempos de procesamiento

En este caso, para evaluar los tiempos de procesamiento y el porcentaje de ciertas etapas del pipeline del proyecto, se han utilizado cuatro reuniones distintas a las empleadas previamente para evaluar la transcripción y la diarización. Esto se debe a que, en aquella evaluación, era importante considerar reuniones con y sin ruido para analizar su efecto en la transcripción. En cambio, en esta ocasión, se seleccionaron reuniones de Microsoft Teams con diferentes duraciones y tamaños para evaluar el rendimiento del sistema en distintas condiciones.

A continuación, se presentan las reuniones que se han usado para el estudio:

- Reunión 1: Reunión de 3 minutos y con un tamaño de 8MB.
- Reunión 2: Reunión de 15.5 minutos y con un tamaño de 44MB.
- Reunión 3: Reunión de 18 minutos y con un tamaño de 37.8MB.
- Reunión 4: Reunión de 21 minutos y con un tamaño de 40.3MB.

Para evaluar el coste temporal, se realizaron múltiples ejecuciones del pipeline para cada reunión, y las tablas que se muestran a continuación presentan la media de todas estas ejecuciones.

Tiempos y porcentajes del pipeline usando FFmpeg:

Reunión	**Tiempo total**	**Ingesta (%)**	**Procesamiento multimedia (%)**	**Transcripción (%)**	**Generación acta (%)**	**Porcentaje restante (%)**
Reunión 1	1 min y 26.5 s	2.44	0.61	36.59	20.73	39.63
Reunión 2	2 min y 49 s	16.04	5.35	40.11	16.04	22.46
Reunión 3	3 min y 14.3 s	13.71	7.61	35.53	17.77	25.38
Reunión 4	3 min y 23 s	9.90	12.38	42.08	19.80	15.84

Tabla 116. Tiempos y porcentajes del pipeline con FFmpeg

Los resultados muestran que, a medida que aumenta la duración y el tamaño de las reuniones, el tiempo total de ejecución del pipeline también se incrementa de manera proporcional. La mayor parte del tiempo se destina a la transcripción y la generación del acta, representando entre un 35% y 42% y entre un 16% y 20% respectivamente. La ingesta y el procesamiento multimedia consumen porcentajes menores, generalmente inferiores al 13%. El resto del tiempo corresponde al resto de funcionalidades que realiza el grafo y tareas internas del sistema. Esto indica que las etapas críticas en cuanto a tiempo son la transcripción y la generación del acta, mientras que la ingesta y el procesamiento multimedia tienen un impacto relativamente menor en el rendimiento total.

Cabe destacar que la ingesta se ve más influenciada por el tamaño del archivo que por su duración, lo cual es un comportamiento esperado. Esto se observa en que la ingesta de la Reunión 2 representa un mayor porcentaje del pipeline que en las Reuniones 3 y 4. Además,

al observar los valores absolutos, el tiempo de ingesta de la Reunión 2 es superior al de la Reunión 3.

En la tabla inferior se presentan los tiempos y porcentajes del pipeline medidos usando AWS.

Reunión	Tiempo total	Ingesta (%)	Procesamiento multimedia (%)	Transcripción (%)	Generación acta (%)	Porcentaje restante (%)
Reunión 1	1 min y 60 s	1.98	19.80	29.70	16.83	31.68
Reunión 2	3 min y 57 s	12.66	32.91	31.65	12.66	10.13
Reunión 3	4 min y 27 s	10.11	33.71	26.22	13.11	16.85
Reunión 4	4 min y 51 s	6.87	36.43	29.21	13.75	13.75

Tabla 117. Tiempos y porcentajes del pipeline con AWS

Los resultados muestran que, a medida que aumentan la duración y el tamaño de las reuniones, también crece el tiempo total de ejecución del pipeline. En este caso, la etapa más costosa es el procesamiento multimedia, que concentra entre un 20% y 36% del tiempo total, seguido de la transcripción (26–32%) y la generación del acta (13–17%). La ingesta tiene un impacto menor, más ligado al tamaño del archivo que a la duración, mientras que el porcentaje restante corresponde a las otras operaciones del sistema.

En FFmpeg, el tiempo total de ejecución fue menor en todas las reuniones (1–3 min), incluso en las de mayor duración. La mayor carga se concentra en la transcripción y la generación del acta, pero aun así logra mantener los tiempos globales más bajos. En cambio, con AWS, aunque el procesamiento se distribuye entre varias etapas, el tiempo total es más elevado (2–5 min), principalmente porque el procesamiento multimedia consume entre un 20% y un 36% del pipeline. En conclusión, **FFmpeg resulta más eficiente en términos de tiempo de ejecución**, mientras que AWS introduce una mayor sobrecarga en el procesamiento multimedia, lo que lo hace más lento bajo las mismas condiciones.

10 Análisis de implicaciones sociales, ambientales y éticas

El desarrollo de cualquier sistema de inteligencia artificial e integración de sistemas, como el aquí propuesto, trasciende lo puramente técnico. Conlleva una serie de implicaciones sociales, ambientales y éticas que deben ser identificadas, analizadas y gestionadas de forma responsable. Este capítulo tiene como objetivo realizar una reflexión crítica sobre el proyecto de generación automática de actas, examinándolo bajo los prismas de la igualdad, el medio ambiente, la responsabilidad social y la ética profesional.

10.1 Igualdad de género y diversidad

El principio de igualdad de trato y no discriminación es fundamental. En el contexto de este proyecto, es crucial evaluar si el sistema pudiese perpetuar o amplificar sesgos existentes, particularmente de género.

- **Identificación de potenciales discriminaciones:** El principal riesgo identificado reside en las herramientas de IA utilizadas, concretamente en las fases de diarización (identificación de hablantes) y transcripción de Amazon Transcribe. Si los modelos en los que se basan estos servicios han sido entrenados con datasets que no son representativos de la diversidad de voces (tonos, acentos, timbres), podrían presentar un rendimiento desigual. Por ejemplo, un modelo entrenado predominantemente con voces masculinas podría cometer más errores al transcribir o identificar a hablantes femeninas, lo que constituiría un sesgo de género algorítmico. Esto tendría un efecto directo en la calidad y precisión del acta generada, pudiendo infrarrepresentar o distorsionar las contribuciones de determinadas personas.
- **Medidas de mitigación adoptadas:** Para combatir este riesgo, el proyecto se plantea con una conciencia clara del problema. En primer lugar, se seleccionan servicios cloud de AWS (Transcribe) que, según su documentación, se esfuerzan por emplear datasets diversos y cuantificar sus sesgos [66]. En segundo lugar, el audio se normaliza siempre antes de la transcripción, lo que disminuye la probabilidad de que aparezcan dichos sesgos.
- **Marco legislativo:** Esta aproximación es coherente con el espíritu de la Ley Orgánica 3/2007, de 22 de marzo, para la igualdad efectiva de mujeres y hombres, que promueve la integración del principio de igualdad en todos los ámbitos, incluido el tecnológico.

10.2 Sostenibilidad y medio ambiente

La huella ambiental de los servicios digitales, especialmente aquellos que implican un alto consumo de procesamiento como el tratamiento de audio y vídeo y la IA, es un aspecto crítico a considerar.

- **Problemas ambientales:** El principal impacto indirecto del proyecto deriva del uso intensivo de servicios en la nube de AWS (EC2, Bedrock (LLMs), S3, Transcribe, etc.). Los data centers que albergan estos servicios consumen grandes cantidades de energía eléctrica y recursos de refrigeración, contribuyendo a la huella de carbono si la energía no proviene de fuentes renovables.

- **Desarrollo sostenible aplicado:** Para aplicar una perspectiva de sostenibilidad, la arquitectura del sistema se ha diseñado siguiendo principios de eficiencia computacional. Se usan servicios funciones serverless de AWS que solo consumen recursos durante el tiempo exacto de ejecución, evitando el desperdicio energético de servidores encendidos de forma permanente. Además, el flujo está optimizado para procesar las grabaciones de la forma más rápida posible, minimizando el tiempo de CPU utilizado. A nivel de estrategia, se prioriza el uso de servicios gestionados por AWS, ya que esta compañía ha anunciado su compromiso de alcanzar cero emisiones netas de carbono para 2040 y alimentar sus operaciones con un 100% de energía renovable para 2025 [67]. Al construir sobre esta infraestructura, el proyecto se beneficia indirectamente de estos objetivos de sostenibilidad corporativa.

  Además, durante el análisis comparativo entre las tecnologías FFmpeg y AWS, se decidió utilizar FFmpeg debido a su mayor eficiencia en el procesamiento multimedia. Esta elección no solo optimiza el rendimiento, sino que también reduce significativamente el consumo computacional, lo que disminuye las emisiones de CO_2. Además, al procesar los datos localmente en lugar de depender de servicios en la nube, se mitiga el impacto ambiental asociado al consumo de agua y a los sistemas de refrigeración de los centros de datos.

10.3 Responsabilidad social

El proyecto no surge en un vacío, sino que responde a necesidades de una comunidad empresarial concreta y tiene el potencial de impactar en el bienestar de sus miembros.

- **Problemáticas sociales abordadas:** El sistema aborda directamente la problemática de la carga administrativa excesiva y el estrés laboral. Liberar a los profesionales de tareas repetitivas y de bajo valor añadido como la redacción manual de actas les permite dedicar su tiempo a labores más creativas, estratégicas o de mayor satisfacción personal, contribuyendo a una mejor conciliación de la vida laboral y personal y a un mayor bienestar en el entorno de trabajo.
- **Accesibilidad:** Al disponer de actas automáticas y consultas interactivas mediante el asistente AIDA, se facilita el acceso a la información a personas que, por cuestiones de disponibilidad o limitaciones personales, no puedan asistir a la reunión en directo.

10.4 Ética

Como ingeniera informática, la ética debe ser el pilar sobre el que se construye cualquier solución, especialmente una que maneja información sensible.

- **Principios éticos clave:** Los conceptos éticos centrales que guían este proyecto son:
  - **Confidencialidad**: Es el principio rector. El sistema está diseñado para que los datos de las reuniones (audio, vídeo, transcripción) nunca salgan del ecosistema controlado por la empresa (Microsoft 365 y AWS bajo su cuenta) y no sean utilizados para entrenar modelos externos.
  - **Transparencia**: Los usuarios deben y son informados de que la reunión será procesada automáticamente.

- **Consecuencias de las decisiones técnicas:** La decisión de no usar APIs de LLMs públicas (como ChatGPT de OpenAI) y en su lugar optar por un modelo desplegado de forma privada o servicios de AWS con cláusulas contractuales estrictas, es una decisión ética con consecuencias técnicas (mayor coste, mayor complejidad) y de negocio (control total). Esta elección prioriza la protección de la información corporativa sensible (secretos comerciales, estrategias, datos personales) sobre la comodidad o el coste, evitando riesgos de filtración o uso indebido de los datos por parte de terceros. Esta postura es deontológicamente sólida y está alineada con normativas como el **Reglamento General de Protección de Datos (RGPD).**

11 Conclusiones

El presente Trabajo de Fin de Grado ha permitido el diseño, desarrollo e implementación de un sistema integral para la generación automática de actas de reuniones de Microsoft Teams, abordando desde la descarga de grabaciones en SharePoint hasta la entrega de un documento formal en PDF y la integración con un asistente inteligente contextual. Los resultados obtenidos demuestran que la solución cumple con los objetivos planteados, combinando técnicas avanzadas de procesamiento multimedia, inteligencia artificial e integración de servicios en la nube.

La arquitectura basada en grafos de DOCs ha resultado ser un marco idóneo para orquestar un flujo de procesamiento tan complejo y multifacético. La modularidad inherente a los streams permitió descomponer el problema en tareas manejables y reutilizables, facilitando tanto el desarrollo como las pruebas y el mantenimiento. La implementación de operaciones de audio y video con dos tecnologías distintas (FFmpeg y AWS) no solo enriqueció el abanico de opciones disponibles, sino que permitió realizar un análisis comparativo riguroso en términos de rendimiento, coste y precisión, cuyos resultados aportan información valiosa para la toma de decisiones en futuras implementaciones.

El uso de LLMs ha sido fundamental para dotar al sistema de capacidades avanzadas de comprensión y generación de lenguaje natural. La corrección automática de transcripciones, la extracción contextualizada de nombres de participantes a partir de video y la generación estructurada de actas son claros ejemplos de cómo la IA transforma un proceso manual y propenso a errores en uno automatizado y fiable. La integración con Amazon Bedrock y Transcribe demostró ser robusta y escalable, aunque también puso de relieve la importancia de un diseño cuidadoso de prompts y de la validación de resultados para minimizar alucinaciones y garantizar la veracidad del output.

La integración con el ecosistema Microsoft 365 (SharePoint, Outlook) y con el asistente AIDA cierra el ciclo de automatización, ofreciendo una experiencia de usuario fluida y productiva. El sistema no solo libera a los profesionales de tareas repetitivas de documentación, sino que también amplía las posibilidades de interacción y consulta de la información mediante sesiones contextualizadas.

Como parte de la **evaluación de la aportación del TFG a mi formación personal**, puedo afirmar que la realización de este Trabajo de Fin de Grado ha representado un hito significativo en mi formación como ingeniera, permitiéndome integrar y aplicar conocimientos adquiridos a lo largo de la carrera en un proyecto de gran envergadura y con un claro impacto en el entorno empresarial. A nivel técnico, he consolidado competencias en el desarrollo de arquitecturas distribuidas y basadas en microservicios, el uso avanzado de servicios cloud en AWS (S3, SQS, SNS, MediaConvert, Rekognition, Transcribe, Bedrock), el procesamiento de señales multimedia con FFmpeg y la integración de APIs REST complejas como Microsoft Graph.

He profundizado en el campo de la inteligencia artificial, adquiriendo experiencia práctica en el uso de LLMs para tareas de comprensión y generación de lenguaje, así como en técnicas de visión por computador para el análisis de video. Asimismo, he desarrollado habilidades clave en ingeniería de software como el diseño de sistemas modulares y extensibles, escritura de código mantenible y bien documentado, implementación de pruebas exhaustivas y gestión de versiones.

Más allá de lo técnico, este TFG me ha permitido desarrollar competencias transversales esenciales como la capacidad de abordar problemas complejos de forma estructurada, la

gestión eficiente del tiempo y los recursos en un proyecto de larga duración, la comunicación efectiva con tutores y equipos de trabajo, y la adaptabilidad para enfrentar desafíos técnicos y tomar decisiones fundamentadas.

Además, el proceso de investigación me llevó a explorar tecnologías y protocolos emergentes que, si bien no fueron aplicados en la solución final, enriquecieron significativamente mi panorama técnico. Un claro ejemplo es el Model Context Protocol (MCP), un protocolo moderno y open source que investigué para una posible integración. MCP está diseñado para estandarizar la comunicación entre aplicaciones y servidores de contexto, actuando como un puente que permite a las herramientas, como los asistentes de IA, acceder de forma segura y dinámica a datos en tiempo real, código, documentos y APIs de diversos orígenes (bases de datos, repositorios, etc.). Aunque al final no se incorporó al diseño, el estudio de MCP me proporcionó una comprensión profunda de los desafíos y las soluciones modernas en el ámbito de la orquestación de contextos para la IA, un conocimiento que sin duda será valioso para mis futuros desarrollos en el campo de los agentes inteligentes y las aplicaciones augmentadas por IA.

Ha sido una oportunidad única para trabajar en un entorno real, con requisitos empresariales exigentes en materia de seguridad, escalabilidad y eficiencia, lo que me ha acercado a la realidad del sector y ha enriquecido mi perfil profesional de manera invaluable.

Lo más relevante es que **este proyecto no culmina con la entrega de este TFG**, sino que constituye la base sobre la que seguiré trabajando en la empresa para desarrollar una solución aún más completa y poderosa. El siguiente paso fundamental será la implementación de un sistema de gestión de accesos y persistencia de datos que supere el actual paradigma de procesamiento efímero.

La arquitectura futura permitirá almacenar cada transcripción en Amazon S3 y gestionarla mediante una base de datos que actúe como índice central, asociando metadatos y permisos de acceso basados en los asistentes de la reunión. A través de un servidor MCP personalizado, el asistente AIDA podrá consultar de forma segura estas transcripciones, realizar búsquedas semánticas y generar resúmenes, evolucionando hacia un sistema con memoria organizacional a largo plazo bajo un modelo estricto de seguridad y privacidad.

En definitiva, todo este futuro desarrollo será posible gracias a la sólida base construida durante este TFG. Más allá de los conocimientos y competencias que me ha aportado en lo personal, este trabajo trasciende lo académico para convertirse en un proyecto con verdadero impacto en el mundo empresarial, capaz de aportar beneficios tangibles a los usuarios y de evolucionar hacia una solución de gran valor.

12 Recursos utilizados

El desarrollo de este proyecto ha requerido la utilización de un conjunto diverso de recursos software, bibliotecas, servicios en la nube y herramientas de desarrollo. A continuación, se detallan los más relevantes:

Software y bibliotecas

- **FFmpeg (v5.1):** Herramienta fundamental para el procesamiento multimedia local. Se compiló e instaló directamente desde el código fuente en el entorno Docker para garantizar el acceso a la última versión estable y a todos los filtros y codecs necesarios (como los filtros de normalización loudnorm y de reducción de ruido), los cuales no siempre están disponibles o están desactualizados en las versiones de los gestores de paquetes de los sistemas operativos. El proceso de instalación en el Dockerfile consistió en descargar el binario estático precompilado, mover los ejecutables ffmpeg y ffprobe a la ruta del sistema y asignar los permisos de ejecución correspondientes.
- **Bibliotecas python:** Se utilizó un amplio ecosistema de librerías para diferentes funcionalidades:
    - **boto3:** SDK oficial de AWS para interactuar sus servicios.
    - **markdown==3.8:** Para la conversión del texto del acta desde formato Markdown a HTML.
    - **websockets==15.0.1:** Para la gestión de conexiones asíncronas.
    - **nest_asyncio==1.6.0:** Para permitir la ejecución de event loops asíncronos en entornos anidados, como Jupyter notebooks.
    - **psutil==7.0.0:** Para la monitorización de recursos del sistema.
    - **jiwer==3.1.0:** Para el cálculo de métricas de evaluación de transcripciones.
    - **lxml==5.4.0:** Para parsear los documentos en XML de las transcripciones de referencia extraídas de *AMI Corpus*.
    - **normalize==2.1.0:** Para tareas de normalización de texto.
    - **opencv-python==4.9.0.80:** Librería especializada en visión por computador. Se usa para determinar si en un fragmento se comparte o no pantalla.
    - **numpy==2.2.5:** Se emplea para operaciones sobre matrices e imágenes, como el cálculo de la cantidad de bordes detectados en un frame.
    - **aspose-pdf==25.3.0:** Se usa para manejar documentos PDF.

Otras librerías estándar como json, os, re, subprocess, datetime, uuid, tempfile, smtplib, email, entre otras fueron usadas extensivamente.

Plataformas y servicios en la nube

- **Amazon Web Services (AWS)**: Se requirió una cuenta activa de AWS con licencia de uso para consumir los siguientes servicios:
    - **Amazon S3:** Almacenamiento de archivos.

    - **Amazon SQS y SNS:** Gestión de colas y notificaciones para la espera de la finalización de jobs.
    - **AWS MediaConvert:** Servicio para procesamiento de audio y video (extracción de audio, normalización, extracción de frames).
    - **Amazon Rekognition:** Servicio para la detección de cambios de escena en videos.
    - **Amazon Transcribe:** Servicio para la transcripción automática de audio y diarización.
    - **Amazon Bedrock:** Servicio para acceder a LLMs como Claude de Anthropic para la generación y corrección de contenido.
- **Microsoft 365:** Se utilizaron sus servicios y APIs para la integración:
    - **Microsoft Graph API:** Para la conexión y descarga de grabaciones desde SharePoint.
    - **Microsoft Teams:** Origen de las grabaciones de reuniones.
    - **SharePoint:** Repositorio central donde se almacenan automáticamente las grabaciones.
    - **Outlook:** Servicio de correo electrónico utilizado para el envío automatizado de las actas finales.

Herramientas de desarrollo y testing

- **Docker y Docker Desktop:** Se utilizó Docker para contenedizar la aplicación y garantizar un entorno de ejecución consistente e independiente del sistema operativo anfitrión. Docker Desktop fue la herramienta utilizada para gestionar los contenedores en el sistema de desarrollo, que fue Windows.
- **Git:** Sistema de control de versiones utilizado para el almacenamiento, gestión y trackeo de todos los cambios en el código fuente a lo largo del proyecto. Se empleó la plataforma de hosting de repositorios GitLab para el trabajo colaborativo y la integración.
- **Microsoft Clipchamp:** Herramienta de edición de video utilizada para preparar, recortar y manipular algunos de los videos y audios de prueba, permitiendo simular diferentes condiciones.
- **AMI Corpus:** Corpus público de grabaciones de reuniones utilizado para obtener muestras de audio con y sin ruido, esenciales para realizar pruebas de transcripción y diarización.
- **MediaInfo:** Herramienta utilizada para analizar los metadatos técnicos de los archivos multimedia de prueba (codec, tasa de bits, frecuencia de muestreo, etc.), lo que fue crucial para la configuración correcta de los procesos de FFmpeg y AWS.

Referencias en línea

Todas las páginas web de recursos en línea consultadas para el desarrollo de este proyecto se encuentran debidamente citadas en la siguiente sección de Referencias.

Referencias

[1] J. Devlin, M.-W. Chang, K. Lee y K. Toutanova, «BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding,» arXiv, 24 May 2019.

[2] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer y V. Stoyanov, «RoBERTa: A Robustly Optimized BERT Pretraining Approach,» arXiv, 26 Jul 2019.

[3] M. Lewis, Y. Liu, N. Goyal, M. Ghazvininejad, A. Mohamed, O. Levy, V. Stoyanov y L. Zettlemoyer, «BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension,» de 58th Annual Meeting of the Association for Computational Linguistics (ACL 2020), Online, Jul 2020.

[4] D. Moreira, I. Cruz, K. Gonzalez, A. Quirumbay, C. Magallan, T. Guarda, A. Andrade y C. Castillo, «Análisis del Estado Actual de Procesamiento de Lenguaje Natural,» RISTI - Revista Ibérica de Sistemas e Tecnologías de Información, nº E42, pp. 126-136, 2021

[5] S. Latif, A. Zaidi, H. Cuayáhuitl, F. Shamshad, M. Shoukat, M. Usama y J. Qadir, «Transformers in Speech Processing: A Survey,» arXiv, 2023.

[6] H. Kheddar, M. Hemis y Y. Himeur, «Automatic Speech Recognition using Advanced Deep Learning Approaches: A survey,» arXiv, 2024.

[7] A. Gulati, J. Qin, C.-C. Chiu, N. Parmar, Y. Zhang, J. Yu, W. Han, S. Wang, Z. Zhang, Y. Wu y R. Pang, «Conformer: Convolution-augmented Transformer for Speech Recognition,» arXiv, 16 May 2020.

[8] C. Lee, S. Britto y K. Diwan, «Evaluating the Impact of Artificial Intelligence (AI) on Clinical Documentation Efficiency and Accuracy Across Clinical Settings: A Scoping Review,» Cureus, vol. 16, nº 11, p. e73994, 19 Nov 2024.

[9] S. Liang, «Otter.ai - Transcripción de voz a texto con IA, grabación y toma de notas,» Otter.ai, Inc., 2016. [En línea]. Available: https://otter.ai. [Último acceso: 2025].

[10] K. y. Z. S. Jain, «Fireflies.ai - AI Meeting Assistant for Teams,» Fireflies.ai, Inc., 2019. [En línea]. Available: https://fireflies.ai. [Último acceso: 2025].

[11] R. White, «Fathom - AI Meeting Assistant & Recorder,» Fathom Video, Inc., 2021. [En línea]. Available: https://fathom.video. [Último acceso: 2025].

[12] B. Tossell, «There's an AI for That - The Largest AI Tools Directory,» There's an AI for That, 2022. [En línea]. Available: https://theresanaiforthat.com/ai. [Último acceso: 2025].

[13] G. Abel, «G2: Business Software and Services Reviews,» G2.com, Inc., 2012. [En línea]. Available: https://www.g2.com. [Último acceso: 2025].

[14] Futurepedia, «Futurepedia - The Largest AI Tools Directory,» Futurepedia, 2022. [En línea]. Available: https://www.futurepedia.io. [Último acceso: 2025].

[15] Google Cloud, «Turn speech into text using Google AI,» Google, [En línea]. Available: https://cloud.google.com/speech-to-text?hl=en. [Último acceso: 2025].

[16] Amazon Web Services, «Amazon Transcribe, Automatically convert speech to text and gain insights,» Amazon.com, [En línea]. Available: https://aws.amazon.com/transcribe/. [Último acceso: 2025].

[17] Microsoft Corporation, «View live transcription in Microsoft Teams meetings,» Microsoft Corporation, [En línea]. Available: https://support.microsoft.com/en-us/office/view-live-transcription-in-microsoft-teams-meetings-dc1a8f23-2e20-4684-885e-2152e06a4a8b. [Último acceso: 2025].

[18] Microsoft corporation, «What's new in Microsoft Teams,» Microsoft corporation, [En línea]. Available: https://support.microsoft.com/en-us/office/what-s-new-in-microsoft-teams-d7092a6d-c896-424c-b362-a472d5f105de. [Último acceso: 2025].

[19] Microsoft Corporation, «Frequently asked questions about Copilot in Microsoft Teams,» Microsoft Corporation, 31 Jul 2024, [En línea]. Available: https://support.microsoft.com/en-us/office/frequently-asked-questions-about-copilot-in-microsoft-teams-e8737767-4087-4ae6-b1d8-10264152b05a. [Último acceso: 2025].

[20] Microsoft Corporation, «Manage Microsoft 365 Copilot in Teams meetings and events,» Microsoft Corporation, 30 Abr 2025. [En línea]. Available: https://learn.microsoft.com/en-us/microsoftteams/copilot-teams-transcription. [Último acceso: 2025].

[21] Microsoft Corporation, «Find the right Microsoft Teams for your business,» Microsoft Corporation, [En línea]. Available: https://www.microsoft.com/en-us/microsoft-teams/compare-microsoft-teams-business-options. [Último acceso: 2025].

[22] Microsoft Corporation, «Microsoft Teams Premium,» Microsoft Corporation, [En línea]. Available: https://www.microsoft.com/en-us/microsoft-teams/premium#tabs-oc2831_tab2. [Último acceso: 2025].

[23] AWS, «Amazon Transcribe Pricing,» AWS, [En línea]. Available: https://aws.amazon.com/transcribe/pricing/. [Último acceso: 2025].

[24] D. Cheung, «Meta Llama 2 vs. OpenAI GPT-4,» A Medium Corporation, 5 Dic 2023. [En línea]. Available: https://medium.com/@meetdianacheung/meta-llama-2-vs-openai-gpt-4-785589efe15e. [Último acceso: 2025].

[25] Anthropic, «Claude 3.5 Sonnet,» Anthropic, 21 Jun 2024. [En línea]. Available: https://www.anthropic.com/news/claude-3-5-sonnet. [Último acceso: 2025].

[26] P. Goel, «An Overview of Llama 2: Open Foundation and Fine-Tuned Chat Models,» Medium, 10 Dic 2023. [En línea]. Available: https://medium.com/@pradeepgoel/an-overview-of-llama-2-open-foundation-and-fine-tuned-chat-models-955677da69a6. [Último acceso: 2025].

[27] S. Pichai y D. Hassabis, «Our next-generation model: Gemini 1.5,» Google The Keyword, 15 Feb 2024. [En línea]. Available: https://blog.google/technology/ai/google-gemini-next-generation-model-february-2024/#sundar-note. [Último acceso: 2025].

[28] C. Mann y A. Kirkovska, «Claude 3 Opus vs GPT-4: Task Specific Analysis,» Vellum AI, 8 Abr 2024. [En línea]. Available: https://www.vellum.ai/blog/claude-3-opus-vs-gpt4-task-specific-analysis. [Último acceso: 2025].

[29] A. Dewan y S. Subramanian, «Get the most from Amazon Titan Text Premier,» AWS, 26 Ago 2024. [En línea]. Available: https://aws.amazon.com/blogs/machine-learning/get-the-most-from-amazon-titan-text-premier/. [Último acceso: 2025].

[30] OpenAI, «Introducing GPT-4.1 in the API,» OpenAI, 14 Abr 2025. [En línea]. Available: https://openai.com/index/gpt-4-1/. [Último acceso: 2025].

[31] Google Cloud, «Detect different speakers in an audio recording,» Google. [En línea]. Available: https://cloud.google.com/speech-to-text/docs/multiple-voices. [Último acceso: 2025].

[32] NVIDIA Corporation, «IA de Habla - Proporciona interfaces basadas en la voz para tus aplicaciones de IA conversacional,» NVIDIA Corporation. [En línea]. Available: https://www.nvidia.com/es-la/ai-data-science/solutions/speech-ai/. [Último acceso: 2025].

[33] M. Elshenawy, S. Khalifa y S. Badawi, «Boost Meeting Productivity with AI-Powered Note-Taking and Summarization,» NVIDIA Corporation, 29 Nov 2023. [En línea]. Available: https://developer.nvidia.com/blog/boost-meeting-productivity-with-ai-powered-note-taking-and-summarization/. [Último acceso: 2025].

[34] AWS, «meeting.ai Optimizes Meeting Productivity with Amazon Bedrock and Claude 3,» AWS. [En línea]. Available: https://aws.amazon.com/solutions/case-studies/meeting-ai. [Último acceso: 2025].

[35] N. Kanda, X. Xiao, Y. Gaur, X. Wang, Z. Meng, Z. Chen y T. Yoshioka, «Transcribe-to-Diarize: Neural Speaker Diarization for Unlimited Number of Speakers using End-to-End Speaker-Attributed ASR,» Microsoft Corporation, May 2022. [En línea]. Available: https://www.microsoft.com/en-us/research/publication/transcribe-to-diarize-neural-speaker-diarization-for-unlimited-number-of-speakers-using-end-to-end-speaker-attributed-asr/. [Último acceso: 2025].

[36] J. Mack, J. Singh Bhoot, I. Pantazi, C. Williams, T. Ehrenborg y M. Borsky, «CAIMAN-ASR: Pushing the Boundaries of Low-Latency Streaming Speech Recognition,» Myrtle.ai, 11 Abr 2025. [En línea]. Available: https://myrtle.ai/resources/caiman-asr-pushing-the-boundaries-of-low-latency-streaming-speech-recognition/. [Último acceso: 2025].

[37] Way With Words, «Speech-to-Text Data Preparation: Best Practices and Techniques,» Way With Words. [En línea]. Available: https://waywithwords.net/resource/speech-to-text-data-preparation-practice/. [Último acceso: 2025].

[38] Medium Corporation, «Audio Pre-Processings For Better Results in the Transcription Pipeline,» Medium Corporation, 29 Sep 2024. [En línea]. Available: https://medium.com/@developerjo0517/audio-pre-processings-for-better-results-in-the-transcription-pipeline-bab1e8f63334. [Último acceso: 2025].

[39] A. Dagar, «Speaker Diarization,» Skit.ai, 21 Jul 2020. [En línea]. Available: https://tech.skit.ai/speaker-diarization/. [Último acceso: 2025].

[40] D. Cazzani, «"Who Said That?" A Technical Intro to Speaker Diarization,» Cisco Systems, Inc., 30 Nov 2021. [En línea]. Available: https://blogs.cisco.com/developer/speakerdiarization01. [Último acceso: 2025].

[41] S. Aggarwal, «Best Prompts to Summarize Meeting Transcripts Using ChatGPT,» Bliro, 20 Feb 2025. [En línea]. Available: https://www.bliro.io/en/blog/best-prompts-to-summarize-meeting-transcripts-using-chatgpt. [Último acceso: 2025].

[42] Gladia, «How to summarize audio using Whisper ASR and GPT 3.5,» Gladia, 6 Nov 2023. [En línea]. Available: https://www.gladia.io/blog/how-to-summarize-audio-using-whisper-asr-and-gpt-3-5. [Último acceso: 2025].

[43] M. Payne, «Harnessing GPT-4 for Meeting Summarization: Zero-Shot and Aspect-Based Approaches,» Width.ai, 21 Ago 2023. [En línea]. Available: https://www.width.ai/post/gpt-4-for-meeting-summarization. [Último acceso: 2025].

[44] R. Majdoddin, «Transcription and diarization (speaker identification),» GitHub, 6 Oct 2022. [En línea]. Available: https://github.com/openai/whisper/discussions/264. [Último acceso: 2025].

[45] A. Neumiller, J. Cai, C. Lott, E. Costa Filho, C. Schuett y H. Su, «Boost productivity with video conferencing transcripts and summaries with the Amazon Chime SDK Meeting Summarizer solution,» AWS, 4 Jun 2024. [En línea]. Available: https://aws.amazon.com/blogs/machine-learning/boost-productivity-with-video-conferencing-transcripts-and-summaries-with-the-amazon-chime-sdk-meeting-summarizer-solution/. [Último acceso: 2025].

[46] Y. Yamsanwar y S. Hirt, «Summarize call transcriptions securely with Amazon Transcribe and Amazon Bedrock Guardrails,» AWS, 17 Oct 2024. [En línea]. Available: https://aws.amazon.com/blogs/machine-learning/summarize-call-transcriptions-securely-with-amazon-transcribe-and-amazon-bedrock-guardrails/. [Último acceso: 2025].

[47] AWS, «AWS Elemental MediaConvert API Reference – Jobs» AWS, 29 Ago 2017. [En línea]. Available: https://docs.aws.amazon.com/mediaconvert/latest/apireference/jobs.html. [Último acceso: 2025].

[48] AWS, «MediaConvert User Guide - Creating a job,» AWS, [En línea]. Available: https://docs.aws.amazon.com/mediaconvert/latest/ug/creating-a-job.html. [Último acceso: 2025].

[49] AWS, «Using the Amazon Rekognition Segment API» AWS, [En línea]. Available: https://docs.aws.amazon.com/rekognition/latest/dg/segment-api.html. [Último acceso: 2025].

[50] AWS, «Amazon Rekognition - StartSegmentDetection» AWS, [En línea]. Available: https://docs.aws.amazon.com/rekognition/latest/APIReference/API_StartSegmentDetection.html. [Último acceso: 2025].

[51] FFmpeg, «ffmpeg Documentation» FFmpeg, [En línea]. Available: https://ffmpeg.org/ffmpeg-all.html. [Último acceso: 2025].

[52] AWS, «Amazon Transcribe - Character sets for custom vocabularies and vocabulary filters» AWS, [En línea]. Available: https://docs.aws.amazon.com/transcribe/latest/dg/charsets.html. [Último acceso: 2025].

[53] EBU TECHNICAL MEDIA TECNOLOGY & INNOVATION, «EBU R 128 – the EBU Loudness Recommendation» EBU TECHNICAL MEDIA TECNOLOGY & INNOVATION. [En línea]. Available: https://docslib.org/doc/628081/ebu-r-128-the-ebu-loudness-recommendation. [Último acceso: 2025].

[54] EBU - TECHNOLOGY & INNOVATION, «Practical guidelines for EBU R 128 (Loudness),» EBU - TECHNOLOGY & INNOVATION, 21 Nov 2023. [En línea]. Available: https://tech.ebu.ch/publications/tech3343. [Último acceso: 2025].

[55] G. Holzmann, «Audio loudness measurement and normalization with EBU R128 (Calm Act, ATSC A/85),» 2 Ago 2012. [En línea]. Available: https://us1.auphonic.com/blog/2012/08/02/loudness-measurement-and-normalization-ebu-r128-calm-act/. [Último acceso: 2025].

[56] Google Cloud, «Introducción a la codificación de audio para Speech-to-Text,» Google. [En línea]. Available: https://cloud.google.com/speech-to-text/docs/encoding?hl=es-419. [Último acceso: 2025].

[57] MediaInfo, «MediaInfoOnline - MediaInfo in your browser!», MediaInfo, [En línea]. Available: https://mediaarea.net/MediaInfoOnline. [Último acceso: 2025].

[58] AWS, «Precios de las instancias bajo demanda de Amazon EC2,» AWS, [En línea]. Available: https://aws.amazon.com/es/ec2/pricing/on-demand/. [Último acceso: 2025].

[59] AWS, «Precios de Amazon EBS,» AWS, [En línea]. Available: https://aws.amazon.com/es/ebs/pricing/. [Último acceso: 2025].

[60] AWS, «Precios de Amazon S3,» AWS, [En línea]. Available: https://aws.amazon.com/es/s3/pricing/. [Último acceso: 2025].

[61] AWS, «Precios de Amazon SNS,» AWS, [En línea]. Available: https://aws.amazon.com/es/sns/pricing/. [Último acceso: 2025].

[62] AWS, «Precios de Amazon SQS,» AWS, [En línea]. Available: https://aws.amazon.com/es/sqs/pricing/. [Último acceso: 2025].

[63] AWS, «Precios de AWS Elemental MediaConvert,» AWS, [En línea]. Available: https://aws.amazon.com/es/mediaconvert/pricing/. [Último acceso: 2025].

[64] AWS, «Precios de Amazon Rekognition,» AWS, [En línea]. Available: https://aws.amazon.com/es/rekognition/pricing/. [Último acceso: 2025].

[65] AMI corpus, «AMI corpus download,» AMI corpus, [En línea]. Available: https://groups.inf.ed.ac.uk/ami/download/. [Último acceso: 2025].

[66] M. Kearns, A. Roth, «Responsible AI in the wild: Lessons learned at AWS,» Amazon science, 16 Nov 2023, [En línea]. Available: https://www.amazon.science/blog/responsible-ai-in-the-wild-lessons-learned-at-aws. [Último acceso: 2025].

[67] Amazon, «Delivering on net-zero carbon by 2040,» Amazon, [En línea]. Available: https://sustainability.aboutamazon.com/. [Último acceso: 2025].

Anexos

Anexo A: Puesta en marcha y mantenimiento

El proyecto se ha integrado como una funcionalidad adicional dentro de la plataforma existente DOCs, a la cual los usuarios finales acceden a través de su interfaz web. De este modo, no es necesario que el usuario realice ninguna instalación ni configuración técnica, ya que puede utilizar la herramienta directamente desde su navegador.

Cabe señalar que, para poder acceder a la plataforma y utilizar sus funcionalidades, entre las cuales se encuentra el proyecto presentado en este documento, los usuarios finales deben adquirir la licencia correspondiente.

Las tareas de mantenimiento clave son las siguientes:

- **Gestión de dependencias:** Las dependencias de Python se listan en requirements.txt. Para actualizarlas, se debe modificar este archivo y reconstruir la imagen Docker.
- **Monitorización:** Se debe monitorizar el sistema de logging para monitorizar las métricas de coste y uso en la consola de AWS para evitar gastos inesperados.
- **Actualización de FFmpeg:** Si en el futuro se requiere una nueva versión de FFmpeg, se debe modificar la URL en el Dockerfile y reconstruir la imagen.
- **Gestión de credenciales:** Las claves de AWS y los secretos de Azure AD tienen una caducidad. Es crucial contar con un procedimiento seguro para su rotación y actualización.
- **Resolución de problemas comunes:**
    - Error de permisos en AWS: Verificar las políticas de IAM del usuario.
    - FFmpeg no encuentra un filtro: Asegurarse de que la instalación se realizó desde los binarios estáticos como se indica en el Dockerfile.
    - Conexión rechazada por SharePoint: Verificar que las credenciales de la aplicación de Azure AD sean correctas y que los permisos necesarios de la API hayan sido concedidos por un administrador.

Anexo B: Funcionalidades desarrolladas, pero no integradas en la versión final

A lo largo del desarrollo se implementaron distintas características que, por diversos motivos ya expuestos en el cuerpo del trabajo, finalmente no fueron incorporadas en la solución final.

Entre ellas destacan dos en particular: La eliminación de silencios en archivos de audio y la invocación de un LLM a partir de entrada en formato vídeo.

En la plataforma DOCs, entorno en el que se integró el proyecto, ya existían mecanismos para interactuar con LLMs a través de texto e imágenes, lo que únicamente requirió configurar su uso para adaptarlo a las necesidades planteadas. No obstante, el soporte para vídeo no estaba contemplado, por lo que en este caso tuve que implementar personalmente la lógica que permitiera realizar dicha llamada.

En este apartado se describe de manera resumida la implementación de estas dos funcionalidades.

Eliminación de silencios en un audio

La técnica de eliminación de silencios no se incorporó en la solución final, dado que introduce limitaciones importantes en el proceso de diarización. Aunque su aplicación resulta ventajosa en el ámbito de la transcripción, al reducir la duración del archivo, optimizar el rendimiento computacional y descartar pausas carentes de contenido lingüístico, en el caso de la diarización provoca el efecto contrario. La supresión de estas pausas altera la continuidad temporal del audio, lo que dificulta la detección precisa de los cambios de locutor y complica la segmentación y asignación correcta de los fragmentos a cada participante.

A continuación, se resume la implementación de esta función mediante FFmpeg y AWS MediaConvert. Es importante señalar que, en ambos casos, la función mantiene la misma estructura de flujo de procesamiento que las demás funciones multimedia descritas en este trabajo, como, por ejemplo, la normalización de audio.

Ambas implementaciones, FFmpeg y AWS, comparten los mismos parámetros de entrada que son los siguientes:

- File_data: El audio de entrada, objeto de tipo FileData, al cual hay que aplicar la eliminación de silencios.
- output_format: Formato de salida del audio. Por defecto ".flac".
- channels: Número de canales.
- sample_rate: Frecuencia de muestreo del audio de salida en Hz.
- bit_depth: Profundidad de bits del audio de salida.
- silence_threshold: Valor en dB que representa el umbral por debajo del cual se considera silencio.
- silence_duration: Duración mínima requerida para considerar un período como silencio (en segundos).

FFmpeg

En la implementación se emplearon distintas librerías de Python que facilitaron tanto el procesamiento como la gestión de datos. La librería **tempfile** permitió crear directorios temporales para almacenar archivos durante la ejecución, mientras que **subprocess** se utilizó para lanzar de manera externa los comandos de FFmpeg. El manejo de rutas y operaciones básicas del sistema se resolvió con **os**, y el análisis de la salida generada por FFmpeg se apoyó en expresiones regulares a través de **re**. Finalmente, el uso de **FileData**, una clase que define la forma en que los archivos se representan y manipulan dentro del sistema, garantizando una gestión estructurada y coherente de la información.

El **flujo de procesamiento** se puede resumir en los siguientes pasos:

1. Validación de parámetros.
2. Configuración de formatos: Se establece la configuración según el formato y profundidad de bits seleccionados.
3. Creación del entorno temporal: Se crea un directorio temporal para el procesamiento.
4. Guardado del archivo de entrada: Los bytes del audio se guardan en un archivo temporal.

5. Construcción del comando FFmpeg: Se genera el comando completo con todos los parámetros necesarios, incorporando el **filtro silenceremove** de FFmpeg con su configuración específica.
6. Ejecución del proceso: Se ejecuta FFmpeg con el filtro silenceremove.
7. Lectura del resultado: El audio procesado se lee y devuelve como objeto FileData.

AWS

Al igual que en el caso anterior, en esta implementación se utilizaron librerías como **os**, que facilita el acceso a variables de entorno, y **FileData**, empleada para definir la estructura con la que se representan y gestionan los archivos dentro del sistema. Además, se incorporaron otras herramientas específicas: **time** para controlar los tiempos de espera en la verificación de procesos en **AWS MediaConvert**, **uuid** para generar identificadores únicos de los archivos temporales en S3, y el **cliente de aws** para interactuar directamente con los servicios en la nube. A ello se suman librerías personalizadas, que reúne funciones auxiliares para la carga, eliminación y recuperación de ficheros en S3, completando así la integración con el entorno cloud.

El **flujo de procesamiento** se puede resumir en los siguientes pasos:

1. Validación de Parámetros.
2. Subida a S3: Almacena el archivo audio de entrada temporalmente en S3 para su procesamiento.
3. Configuración del trabajo de MediaConvert: Prepara la configuración específica para eliminación de silencios, mediante el filtro ***silenceremove***.
4. Ejecución del Trabajo: Inicia y monitoriza (mediante SNS y SQS) el trabajo de procesamiento.
5. Descarga de Resultados: Recupera el audio procesado desde S3.
6. Limpieza: Elimina los archivos temporales de S3.
7. Se devuelve el resultado recuperado desde S3 como objeto FileData.

Invocación de un LLM con entrada de vídeo

Se desarrolló la lógica para invocar un LLM a partir de vídeo, utilizando AWS Bedrock y sus modelos Nova, recientemente habilitados para entradas audiovisuales. No obstante, esta funcionalidad no se incorporó al proyecto final, ya que el modelo presenta limitaciones significativas como la no capacidad de leer texto en los vídeos y sus capacidades generales resultan insuficientes para los objetivos requeridos. Por ello, se optó por soluciones alternativas más adecuadas para el procesamiento y análisis multimedia.

Librerías usadas

En esta implementación se utilizaron varias librerías de Python para facilitar la manipulación y el envío de los vídeos al LLM mediante AWS Bedrock. Al igual que en apartados anteriores, se empleó **subprocess** para ejecutar comandos de FFmpeg y FFprobe y así preparar los segmentos de vídeo. Además, se usaron librerías estándar como **base64** para codificar los vídeos antes de su transmisión y **json** para la serialización y deserialización de las solicitudes y respuestas hacia Bedrock. Por otra parte, aws permitió configurar el cliente

para interactuar con el servicio **Bedrock Runtime**, mientras que **tempfile** se utilizó para crear directorios temporales durante el procesamiento de los segmentos. Finalmente, **FileData** sirvió para representar y gestionar de manera estructurada los archivos de vídeo dentro del sistema.

Parámetros principales

- file_data: El video de entrada, objeto de tipo FileData, el cual se pasará y procesará con el LLM.
- prompt: Texto personalizado para guiar el análisis del LLM.

Flujo de procesamiento

1. Segmentación del video: Dado que el LLM tiene un límite en el tamaño de entrada, los vídeos de gran duración o tamaño deben dividirse en fragmentos más pequeños para poder procesarlos correctamente.
2. Preparación del Contenido: Se detecta automáticamente el formato del vídeo, se codifica en Base64 porque así lo requiere la API del LLM y se construye el mensaje adecuado para ser enviado al LLM.
3. Configuración del LLM: Se define el comportamiento del modelo mediante system prompts, se ajustan los parámetros de inferencia (topK, topP, temperatura y máximo tokens de salida) para el análisis de vídeo y se especifica el id del modelo Nova de AWS Bedrock.
4. Procesamiento y Respuesta: Se realiza la invocación del modelo de Bedrock, se procesa la respuesta obtenida y se extrae y formatea el contenido analítico para su posterior uso.

Parámetros de inferencia del LLM

Los parámetros de inferencia determinan cómo el modelo genera sus respuestas a partir de una entrada. TopK controla el número de palabras candidatas más probables que el modelo considera en cada paso, limitando la selección a las K más altas. TopP (muestreo de núcleo) define un umbral acumulativo de probabilidad, permitiendo al modelo elegir entre las palabras más plausibles hasta que se alcance esa probabilidad. Temperatura ajusta la aleatoriedad de la salida donde valores bajos producen respuestas más deterministas y coherentes, mientras que valores altos aumentan la creatividad. Finalmente, el máximo de tokens de salida limita la longitud de la respuesta generada, evitando que el modelo produzca textos excesivamente largos. Estos parámetros se combinan para equilibrar precisión, creatividad y control sobre la generación de la respuesta.