COMILLAS
UNIVERSIDAD PONTIFICIA
ICAI

GRADO EN INGENIERÍA EN TECNOLOGÍAS DE TELECOMUNICACIÓN

TRABAJO FIN DE GRADO

Análisis Automatizado de Llamadas en un Call center Mediante Inteligencia Artificial y Herramientas de Procesamiento de Lenguaje Natural

Autor: Beatriz Royo Miralles

Director: Jose Rubio Bernabe

Madrid

Declaro, bajo mi responsabilidad, que el Proyecto presentado con el título

Análisis Automatizado de Llamadas en un Call center Mediante Inteligencia Artificial y Herramientas de Procesamiento de Lenguaje Natural

en la ETS de Ingeniería - ICAI de la Universidad Pontificia Comillas en el curso académico 2024/25 es de mi autoría, original e inédito y no ha sido presentado con anterioridad a otros efectos.

El Proyecto no es plagio de otro, ni total ni parcialmente y la información que ha sido tomada de otros documentos está debidamente referenciada.

Fdo.: Beatriz Royo Miralles Fecha: 23/ 06/2025

Autorizada la entrega del proyecto

EL DIRECTOR DEL PROYECTO

Fdo.: Jose Rubio Bernabe Fecha: 23/ 06/2025

COMILLAS
UNIVERSIDAD PONTIFICIA
ICAI

GRADO EN INGENIERÍA EN TECNOLOGÍAS DE TELECOMUNICACIÓN

TRABAJO FIN DE GRADO

Análisis Automatizado de Llamadas en un Call center Mediante Inteligencia Artificial y Herramientas de Procesamiento de Lenguaje Natural

Autor: Beatriz Royo Miralles

Director: José Rubio Bernabe

Madrid

Agradecimientos

A mi familia, por ser el pilar sobre el que se ha sostenido todo este camino. Gracias por creer en mí incluso cuando yo misma dudaba, por inspirarme y acompañarme en cada paso.

A mis abuelos, Mario y Clara, por enseñarme el valor de lo que dos personas pueden construir juntas con amor, constancia y generosidad.

A mi abuela Inmaculada, por recordarme todos los días de lo que soy capaz.

A mi madre, por hacerme la mujer que soy hoy. Su fuerza, dedicación y ejemplo han sido mi mayor guía. Nada de esto habría sido posible sin ella.

A mi padre, por confiar siempre en mí, incluso en los momentos más inciertos.

A mis amigas de Novelda, por estar presentes toda la vida, por ser esa red invisible que se activa con un simple mensaje, sin importar el lugar ni la hora. Aunque estemos cada una en una punta del mundo, siempre estamos conectadas. Hacéis que 7.000 kilómetros parezcan un suspiro.

A mis amigas de Madrid, mis cuchis, gracias por haber sido mi refugio y mi distracción. Os agradezco de corazón todo lo que habéis hecho por mí, por tenerme siempre presente incluso en la distancia. Me habéis visto crecer y madurar, y me habéis acompañado en los momentos más duros de la carrera y de la vida. Habéis sido un pilar fundamental durante estos años.

A mi gente de Illinois, que me han acompañado durante la creación de este proyecto, con sus ánimos en los días buenos y su apoyo en los días grises. Gracias por haber sido hogar lejos de casa.

A mi director de proyecto, Jose Rubio, por verme capaz de poder hacer un proyecto tan grande como este.

ANÁLISIS AUTOMATIZADO DE LLAMADAS EN UN CALL CENTER MEDIANTE INTELIGENCIA ARTIFICIAL Y HERRAMIENTAS DE PROCESAMIENTO DE LENGUAJE NATURAL

Autor: Royo Miralles, Beatriz.
Director: Rubio Bernabe, Jose
Entidad Colaboradora: cableworld

RESUMEN DEL PROYECTO

El presente proyecto nace de una necesidad real identificada durante mi experiencia directa colaborando con **cableworld**: la falta de herramientas automatizadas para analizar el enorme volumen de llamadas que se gestionan diariamente en su *Call center*. Esta iniciativa combina tecnologías avanzadas de Inteligencia Artificial y Procesamiento de Lenguaje Natural para transformar un proceso manual, lento y limitado en una herramienta estratégica de alto impacto. El sistema diseñado incluye la transcripción automática de audios, la identificación precisa de los interlocutores, el análisis de palabras clave, y la organización sistemática de los resultados en una base de datos PostgreSQL. Gracias a ello, **cableworld** ahora puede generar informes detallados sobre desempeño de agentes, patrones geográficos y tipos de consulta, logrando no solo ahorrar tiempo y reducir errores humanos, sino también reforzar su capacidad analítica para tomar decisiones informadas. Este trabajo representa un avance significativo frente al modelo anterior, que apenas lograba gestionar los cerca de 1.000 audios diarios, y simboliza un paso adelante hacia la modernización tecnológica de la compañía.

Palabras clave: Inteligencia Artificial, Procesamiento de Lenguaje Natural, Automatización, *Call center*, Transcripción de Audio

1. Introducción

A lo largo de este proyecto he podido comprobar cómo el sector de las telecomunicaciones afronta retos cada vez mayores para gestionar con eficacia las interacciones con sus clientes. **cableworld**, en particular, procesa miles de llamadas cada mes en su *Call center*, generando datos que, si se aprovechan adecuadamente, pueden ofrecer información clave para optimizar servicios, detectar problemas y mejorar la experiencia del cliente. Sin embargo, hasta hace poco, esos datos se revisaban de forma manual, usando hojas de cálculo, lo que limitaba mucho las posibilidades de análisis y dificultaba detectar patrones relevantes. Este proyecto surge precisamente para dar respuesta a esa realidad, con el objetivo de implementar una solución innovadora que permita evolucionar hacia un modelo automatizado, preciso y con un verdadero impacto estratégico.

2. Descripción de las tecnologías

El proyecto integra un ecosistema tecnológico avanzado diseñado para automatizar el análisis de llamadas entrantes en **cableworld**. En primer lugar, la API de Enreach

permite acceder diariamente a las grabaciones y metadatos de las llamadas, dando inicio al flujo automatizado. La transcripción de audio se realiza con Vosk [1] , un motor de reconocimiento de voz neuronal, ligero y sin necesidad de conexión a internet, que garantiza privacidad y eficiencia.

Una vez transcritas, las llamadas son analizadas mediante modelos GPT (3.5 Turbo) [2], capaces de extraer emociones, etiquetas clave, resúmenes y clasificaciones temáticas. Este proceso se apoya en técnicas de Procesamiento de Lenguaje Natural [3] , que enriquecen los resultados con detección de intenciones, categorización técnica/comercial y normalización textual.

La arquitectura se orquesta automáticamente a través de *cron jobs* [4]en un servidor Linux, que ejecutan una cadena de scripts en Bash y Python encargados de descargar, procesar, transcribir, analizar y almacenar los datos sin intervención humana. Para ello, se implementó procesamiento multihilo que reduce los tiempos de ejecución diarios a menos de una hora para más de 1.000 llamadas.

Todos los resultados se insertan en una base de datos PostgreSQL, lo que permite su explotación analítica y visualización a través de un *dashboard* web interactivo creado con Streamlit [5] , donde responsables no técnicos pueden consultar métricas filtradas en tiempo real. Herramientas como Postman[6], Pandas, ffmpeg y psycopg2 [7]complementan el sistema asegurando integraciones robustas, conversión de formatos y gestión eficiente de datos.

En conjunto, esta arquitectura modular, escalable y segura transforma por completo la operativa previa basada en hojas de cálculo, sentando las bases para una supervisión moderna basada en datos y adaptada a las necesidades reales del negocio.

3. Estado de la cuestión

Antes de diseñar la solución propuesta, se analizó el contexto inicial de cableworld y las alternativas existentes. El sistema previo era completamente manual, sin transcripción automática ni indicadores objetivos, lo que dificultaba la explotación masiva de datos y generaba una atención al cliente heterogénea.

Tras revisar tecnologías del mercado como Google Cloud[8], Amazon Transcribe [9] o Upbe[10], se comprobó que ninguna ofrecía el equilibrio necesario entre coste, personalización y control de datos. Además, el análisis de volumen reveló que el Call center gestiona más de 1.000 horas de llamadas mensuales, lo que hacía inviable económicamente cualquier solución cerrada.

Por todo ello, se optó por una arquitectura propia basada en tecnologías abiertas: Vosk para transcripción local sin coste, **GPT-3.5** para análisis semántico eficiente, y PostgreSQL [11] para almacenamiento. Esta combinación permitió garantizar escalabilidad, seguridad y adaptación total a las necesidades operativas de la empresa.

4. Justificación del Proyecto

Este capítulo expone las razones por las que se descartaron soluciones comerciales existentes para el análisis de llamadas y se optó por el desarrollo de una herramienta propia. Alternativas como Upbe, Amazon Contact Lens [12] o Google CCAI resultaron inadecuadas por su alto coste, falta de personalización o problemas de privacidad al depender de almacenamiento en la nube.

La solución diseñada combina transcripción local con Vosk y análisis semántico con GPT-3.5, garantizando bajo coste, control total de datos, integración con la infraestructura existente y escalabilidad. Además, se adapta a la limitación técnica de las grabaciones en formato mono, que impide el uso de diarización avanzada.

Desde el punto de vista económico, el sistema desarrollado permite **una reducción de costes superior al 99 % respecto a las soluciones comerciales**, con un mantenimiento mínimo y sin renunciar a la calidad analítica ni a la personalización. Esta diferencia justifica plenamente la viabilidad y sostenibilidad del proyecto.

5. Descripción del Modelo/Sistema/Herramienta

El sistema diseñado para este proyecto constituye una solución automatizada, modular y escalable que permite transformar miles de llamadas en información estructurada y útil, combinando inteligencia artificial, procesamiento de lenguaje natural y herramientas de visualización web. Su implementación en un servidor Linux con tareas programadas mediante cron permite ejecutar diariamente todas las fases del procesamiento sin intervención manual, garantizando así eficiencia operativa, trazabilidad y estabilidad.

El flujo completo se apoya en una secuencia de scripts personalizados:

descarga_audio.sh: obtiene automáticamente las grabaciones del día anterior desde la API de Enreach.

download_ATC.sh: recupera los archivos JSON con metadatos relevantes de cada llamada (número llamado, agente asignado, duración, etc.).

procesar_json.sh: transforma los datos en estructuras legibles y útiles para los módulos posteriores.

unzip_zipped0.sh: localiza y descomprime automáticamente las grabaciones contenidas en archivos .zip.

search_call_agent.sh: organiza las llamadas por agente y localidad, generando un archivo resumen con la información extraída.

transcribir_paralelo.py: convierte los audios en texto mediante modelos de transcripción en paralelo basados en IA.

analizar_todas_llamadas_gpt.py: aplica técnicas de procesamiento del lenguaje natural (NLP) para extraer automáticamente etiquetas, palabras clave, incidencias,

opiniones, emociones y referencias a ventas, e inserta los resultados en una base de datos PostgreSQL.

delete_all.py: elimina las carpetas procesadas para optimizar el uso de espacio y mantener el sistema limpio.

El sistema está diseñado para ser resistente a errores y fácilmente ampliable. Toda la información procesada se almacena en una base de datos estructurada en PostgreSQL, que centraliza el acceso a los resultados y permite consultas analíticas posteriores. Esta base es alimentada directamente por los scripts y es la fuente principal de datos del *frontend* desarrollado en Streamlit, que actúa como panel de control visual.

El panel web permite consultar los resultados de forma interactiva, filtrando por fecha, agente, localidad, tipo de llamada, opinión o estado de resolución y más. Incluye visualizaciones dinámicas como gráficos de evolución, distribución horaria, rankings y mapas de calor.

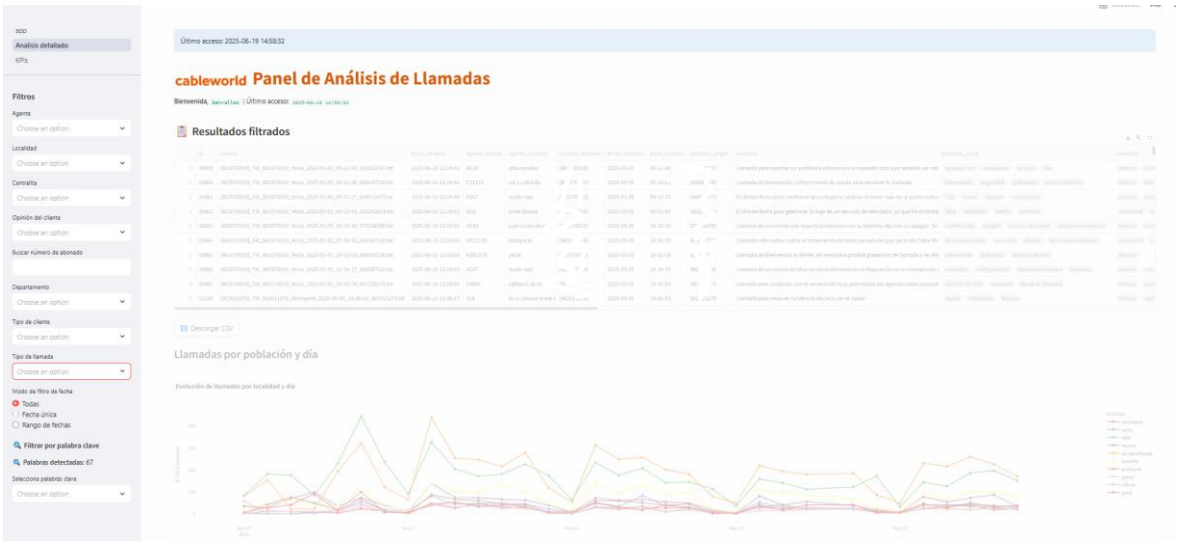


Ilustración 1: Panel interactivo para ver resultados de llamadas

En cuanto a la seguridad, el sistema implementa mecanismos de autenticación de usuarios, cifrado de contraseñas, control de acceso con visibilidad restringida, y cumple parcialmente los requisitos del Esquema Nacional de Seguridad (ENS). Se han establecido roles diferenciados, trazabilidad de accesos y medidas de protección del servidor frente a accesos no autorizados.

La arquitectura modular y automatizada permite que el sistema sea fácilmente adaptable a nuevas fuentes de datos, modelos de análisis o necesidades futuras. A continuación, se presentan la arquitectura general del sistema y el diagrama de casos de uso, que resumen visualmente su funcionamiento y los actores implicados.

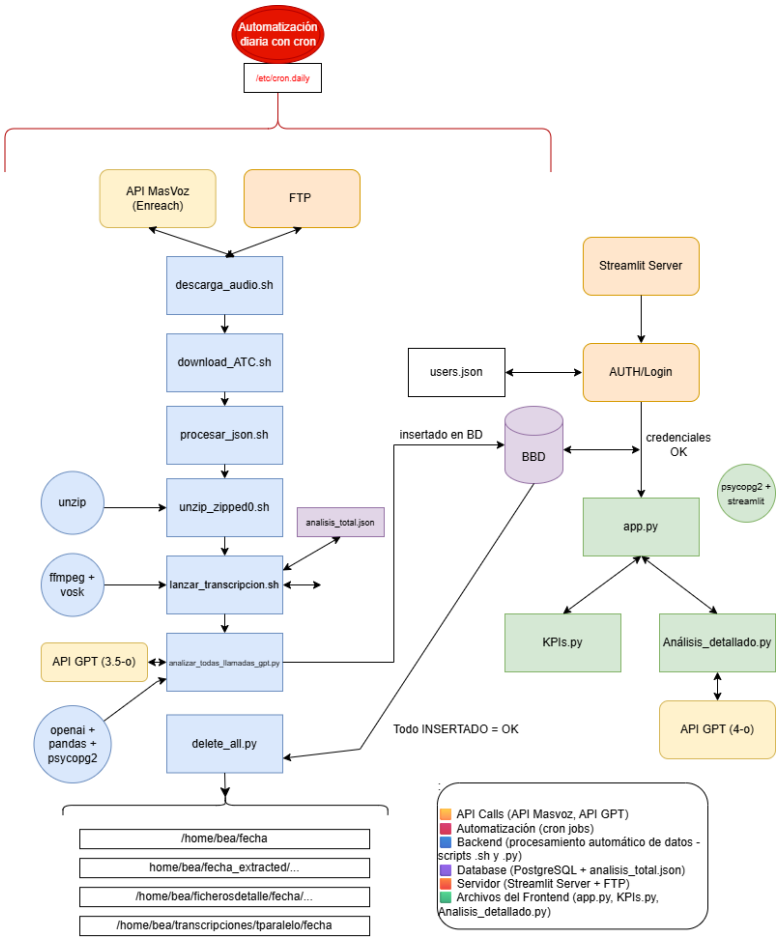


Ilustración 2:Arguitectura del sistema

6. Análisis de resultados

Tras haber detallado la arquitectura y funcionamiento del sistema desarrollado, este capítulo presenta un análisis cuantitativo y cualitativo de los resultados obtenidos tras la puesta en marcha del sistema de transcripción y análisis automatizado de llamadas. Se examinan aspectos clave como el volumen diario de llamadas, la distribución por franjas horarias, la tipología de opiniones expresadas por los clientes, la actividad por agente y las reincidencias.

Los datos analizados proceden directamente de los archivos de audio procesados y de los resultados extraídos mediante técnicas de procesamiento del lenguaje natural (NLP), almacenados en la base de datos PostgreSQL e interpretados a través del panel web.

A modo de ejemplo ilustrativo, se incluye en este capítulo una gráfica representativa del análisis de opiniones, que muestra la distribución de llamadas positivas, negativas y neutras, clasificadas por zona.

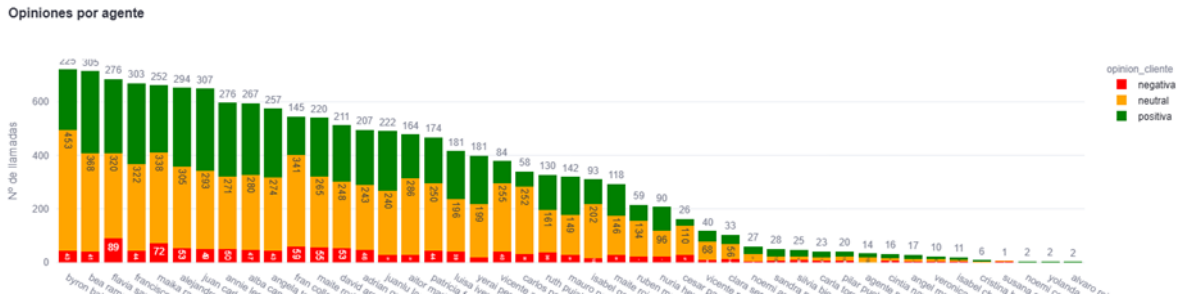


Ilustración 3:Opiniones por agente

Además, se presenta un segundo ejemplo centrado en la gestión operativa del día 29 de abril [13] , posterior al apagón, donde se registró un pico anómalo de llamadas. Este caso sirve para evidenciar la capacidad del sistema para detectar eventos excepcionales, analizar sus causas y cuantificar su impacto

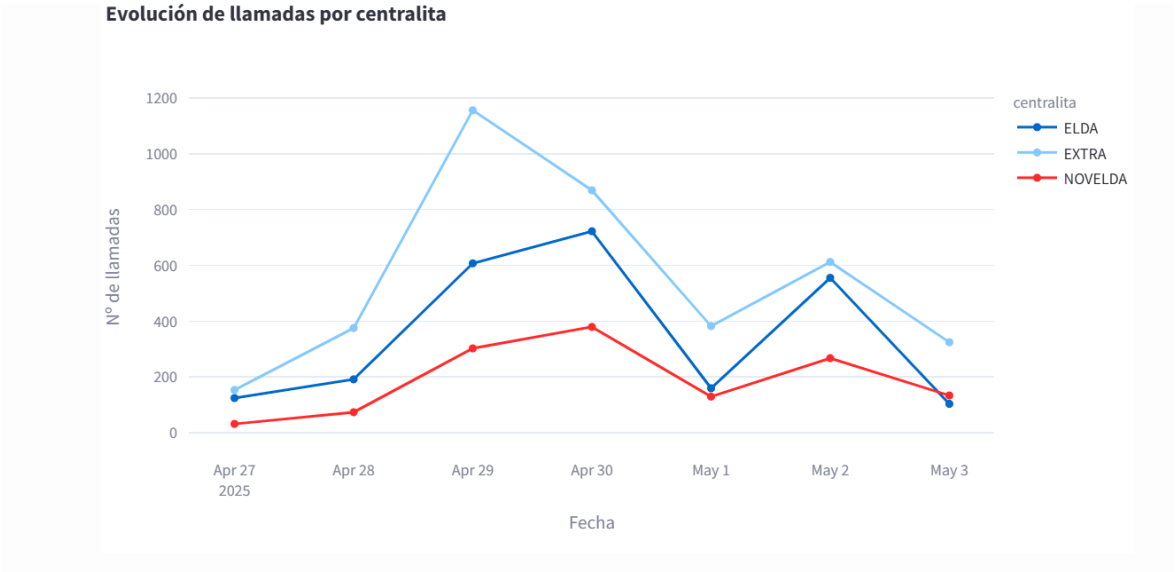


Ilustración 4:Volumen de llamadas época del apagón

7. Conclusiones

El sistema desarrollado ha transformado de forma radical la supervisión del servicio de atención telefónica, permitiendo analizar de forma automatizada más de 22.000 llamadas mensuales mediante técnicas de inteligencia artificial y procesamiento del lenguaje natural. Gracias a su arquitectura modular y al uso de tareas programadas, el sistema funciona de forma completamente autónoma, generando cada día resultados estructurados y visualmente interpretables.

La solución ha permitido detectar patrones operativos, gestionar incidencias en tiempo real y optimizar la asignación de agentes entre centralitas. El análisis de opiniones, reincidencias y palabras clave ha mejorado notablemente la toma de decisiones, reemplazando procesos manuales basados en hojas de cálculo. El panel web interactivo permite a los responsables consultar y filtrar los datos con precisión, lo que ha repercutido en una mayor eficiencia del centro de atención.
Pese a los avances, el sistema presenta limitaciones como la cobertura parcial (no incluye llamadas salientes), la ambigüedad en algunas clasificaciones automáticas y la ausencia de alertas en tiempo real. Aun así, sienta una base sólida para evolucionar hacia un sistema completo de análisis y supervisión operativa en tiempo real.

1. Referencias

[1] Alpha Cephei. (2024). *Vosk Speech Recognition Toolkit*. https://alphacephei.com/vosk/server
[2] OpenAI. (2023). *GPT-3.5 Turbo overview*. https://platform.openai.com/docs/models/gpt-3-5
[3] Jurafsky, D., & Martin, J. H. (2021). *Speech and Language Processing* (3rd ed.). https://web.stanford.edu/~jurafsky/slp3/
[4] Python Software Foundation. (s.f.). *multiprocessing — Process-based parallelism*. https://docs.python.org/3/library/multiprocessing.html
[5] Streamlit Inc. (s.f.). *Streamlit — The fastest way to build and share data apps*. https://streamlit.io/
[6] Postman. (s.f.). *API platform for building and using APIs*. https://www.postman.com/
[7] McKinney, W. (2010). *Data Structures for Statistical Computing in Python*. Proceedings of the 9th Python in Science Conference, 51–56. https://pandas.pydata.org/
FFmpeg. (s.f.). *FFmpeg Documentation*. https://ffmpeg.org/documentation.html
The PostgreSQL Global Development Group. (s.f.). *psycopg2 – PostgreSQL database adapter for Python*. https://www.psycopg.org/
[8] Google Cloud. (2024). *Speech-to-Text API Documentation*. https://cloud.google.com/speech-to-text
[9] Amazon Web Services. (s.f.). *Amazon Transcribe – Automatic Speech Recognition*. https://aws.amazon.com/transcribe/
[10] Upbe. (2024). *Product Overview*. https://www.upbe.ai/
[11] PostgreSQL Global Development Group. (2023). *PostgreSQL Documentation*. https://www.postgresql.org/docs/
[12] Amazon Web Services. (2024). *Amazon Contact Lens for Amazon Connect*. https://aws.amazon.com/connect/contact-lens/
[13] Lombardi, P., & Latona, D. (2025, junio 17). *Miscalculation by Spanish power grid operator REE contributed to massive blackout*. Reuters. https://www.reuters.com/business/energy/investigation-into-spains-april-28-blackout-shows-no-evidence-cyberattack

AUTOMATED CALL CENTER ANALYSIS USING ARTIFICIAL INTELLIGENCE AND NATURAL LANGUAGE PROCESSING TOOLS

Author: Royo Miralles, Beatriz
Supervisor: Rubio Bernabe, Jose
Collaborating Entity: cableworld

ABSTRACT

This project emerged from a real need identified during my direct collaboration with cableworld: the absence of automated tools to analyze the massive volume of calls handled daily by its Call center. This initiative combines advanced technologies in Artificial Intelligence and Natural Language Processing (NLP) to transform a manual, slow, and limited process into a high-impact strategic tool.

The system includes automatic audio transcription, accurate speaker identification, keyword analysis, and systematic organization of results into a PostgreSQL database. As a result, cableworld can now generate detailed reports on agent performance, geographic patterns, and types of inquiries—achieving not only time savings and a reduction in human errors, but also significantly strengthening its analytical capabilities for informed decision-making.

For me, this project represents a significant leap from the previous model, which barely managed to process approximately 1,000 daily audio files, and symbolizes a step forward toward the company's digital transformation.

Keywords: Artificial Intelligence, Natural Language Processing, Automation, Call center, Audio Transcription

1. Introduction

Throughout this project, I have seen firsthand how the telecommunications sector faces increasing challenges in managing customer interactions effectively. cableworld processes thousands of calls each month in its Call center, generating data that if properly harnessed, can provide valuable insights to optimize services, detect problems, and improve customer experience.

However, until recently, these data were reviewed manually using spreadsheets, which severely limited the ability to analyze patterns or detect recurring issues. This project was conceived precisely to address that gap, with the aim of implementing an innovative solution capable of evolving into an automated, precise, and strategically impactful model.

2. Description of the Technologies

The project integrates an advanced technological ecosystem designed to automate the analysis of inbound calls at Cableworld. First, the Enreach API allows daily access to recordings and metadata of the calls, launching the automated workflow.

Audio transcription is performed using Vosk [1] , a neural speech recognition engine that is lightweight, runs offline, and ensures privacy and efficiency. Once transcribed, calls are analyzed using GPT models (3.5 Turbo) [2], capable of extracting emotions, key tags, summaries, and thematic classifications. This process relies on Natural Language Processing [3] techniques, which enrich the results through intent detection, technical/commercial categorization, and text normalization.

The entire architecture is orchestrated automatically through *cron jobs* on a Linux server, which execute a chain of Bash and Python scripts responsible for downloading, processing, transcribing, analyzing, and storing data without human intervention. Multithreaded processing reduces daily execution time to less than one hour for more than 1,000 calls.[4]

All results are inserted into a PostgreSQL database, enabling analytical exploitation and visualization through an interactive web dashboard built with **Streamlit** [5] , where non-technical users can consult filtered metrics in real time. Tools such as Postman [6] , Pandas, ffmpeg, and psycopg2 [7] complement the system by ensuring robust integrations, format conversion, and efficient data management.

Altogether, this modular, scalable, and secure architecture completely transforms the prior spreadsheet-based operation, laying the groundwork for modern, data-driven supervision tailored to real business needs.

3. State of the Art

Before designing the proposed solution, the initial context at cableworld and the existing alternatives were analyzed. The previous system was fully manual, with no automatic transcription or objective indicators, which limited large-scale data exploitation and led to heterogeneous customer service.

After reviewing technologies such as Google Cloud [8], Amazon Transcribe [9], or Upbe [10], none offered the necessary balance between cost, customization, and data control. Furthermore, volume analysis revealed that the Call center handles more than 1,000 hours of calls per month, making any commercial closed-source solution economically unfeasible.

As a result, the project adopted a custom architecture based on open technologies: Vosk for local and cost-free transcription, GPT-3.5 for efficient semantic analysis, and PostgreSQL [11] for secure storage. This combination ensured scalability, security, and full adaptation to the company’s operational requirements.

4. Project Justification

This chapter outlines the reasons why existing commercial solutions for call analysis were discarded in favor of developing a custom tool. Alternatives such as Upbe, Amazon Contact Lens [12], or Google CCAI were deemed inadequate due to their high cost, lack of customization, or privacy concerns related to cloud-based data storage.

The proposed solution combines local transcription using Vosk with semantic analysis via GPT-3.5, ensuring low cost, full data control, seamless integration with the existing infrastructure, and scalability. It also adapts to the technical limitation of mono audio recordings, which prevents the use of advanced speaker *diarization* techniques.

From an economic standpoint, the developed system enables **a cost reduction of over 99% compared to commercial platforms**, with minimal operational requirements and no compromise in analytical depth or system flexibility. This advantage reinforces both the **technical and financial sustainability** of the solution, making it a viable and scalable alternative tailored to the organization's specific needs.

5. Description of the Model / System / Tool

The system designed for this project constitutes an **automated, modular, and scalable solution** that transforms thousands of daily calls into structured, actionable information by combining **artificial intelligence**, **natural language processing**, and **web-based visualization tools**. Its deployment on a **Linux server**, with tasks scheduled via cron, ensures that all processing phases are executed daily without manual intervention, thus guaranteeing **operational efficiency, traceability, and stability**.

The complete workflow is based on a sequence of custom-developed scripts:

- **descarga_audio.sh:** automatically retrieves the previous day's recordings from the Enreach API.
- **download_ATC.sh:** downloads the JSON files containing metadata for each call (dialed number, assigned agent, duration, etc.).
- **procesar_json.sh:** processes and formats the data into readable and usable structures for the subsequent modules.
- **unzip_zipped0.sh**: locates and automatically decompresses the recordings stored in .zip files.
- **search_call_agent.sh**: organizes the calls by agent and location, generating a summary file with the extracted information.
- **transcribir_paralelo.py**: transcribes the audios into text using parallel processing with AI transcription models.
- **analizar_todas_llamadas_gpt.py:** applies natural language processing (NLP) techniques to automatically extract labels, keywords, incidents, opinions, emotions, and references to sales, and inserts the results into a PostgreSQL database.

- **delete_all.py:** deletes processed folders to optimize disk usage and keep the system clean.

The system is designed to be error-tolerant and easily extendable. All processed information is stored in a structured PostgreSQL database, which centralizes access to the results and allows for subsequent analytical queries. This database is populated directly by the scripts and serves as the primary data source for the frontend developed in Streamlit, which functions as a visual control panel.

The web dashboard allows users to interactively query results, filtering by date, agent, location, call type, opinion, or resolution status. It includes dynamic visualizations such as trend graphs, hourly distributions, rankings, and heatmaps.

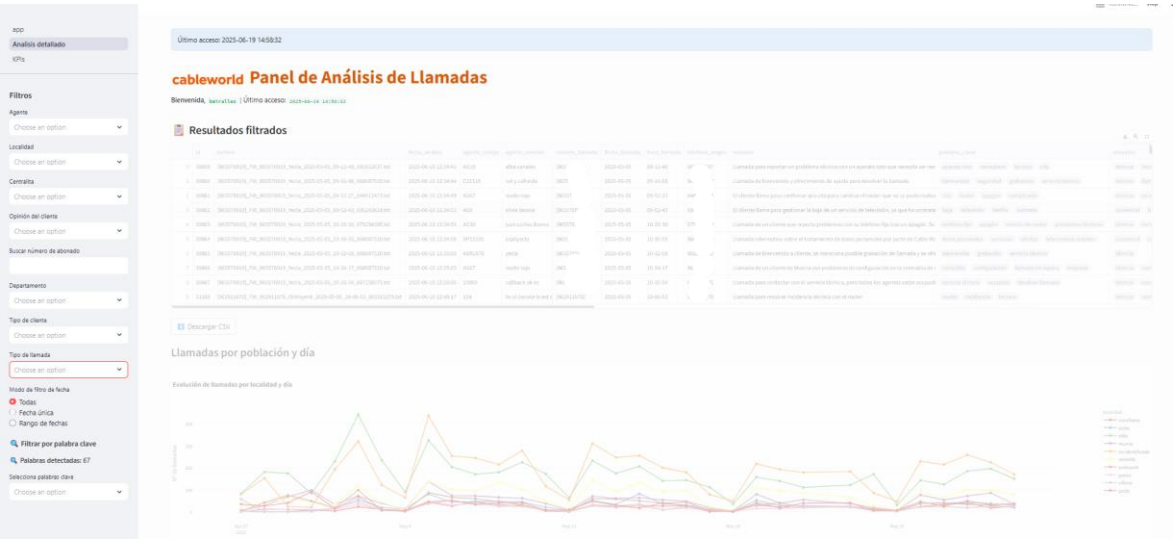


Ilustración 5: Interactive dashboard for reviewing call results

In terms of security, the system implements user authentication, password encryption, and access control with restricted visibility, and partially complies with the requirements of the Spanish National Security Framework (ENS). User roles are clearly defined, access is traceable, and the server is protected against unauthorized access.

The modular and automated architecture ensures that the system is easily adaptable to new data sources, alternative analysis models, or future operational requirements. The following diagrams provide a visual summary of the system's overall architecture and use case interactions.

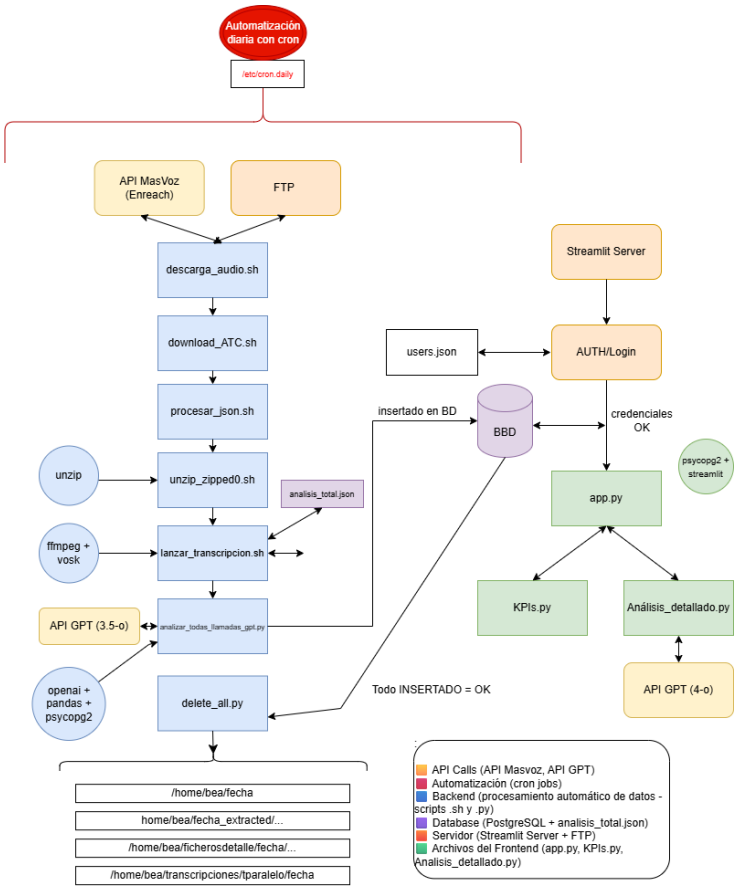


Ilustración 6:System architecture

6. Analysis of Results

After detailing the architecture and functionality of the developed system, this chapter presents a quantitative and qualitative analysis of the results obtained following the deployment of the automated call transcription and analysis system. Key aspects are examined, such as daily call volume, hourly distribution, the types of opinions expressed by customers, agent activity, and recurring contacts.

The data analyzed comes directly from the processed audio files and from the insights extracted using Natural Language Processing (NLP) techniques. These results are stored in a PostgreSQL database and interpreted through the web-based dashboard.

As an illustrative example, this chapter includes a representative chart showing the opinion analysis, which displays the distribution of positive, negative, and neutral calls by region. In addition, a second example focuses on the operational management on April 29 [13], the day after a major outage, when an anomalous peak in calls was recorded. This case demonstrates the system's ability to detect exceptional events, analyze their causes, and quantify their operational impact

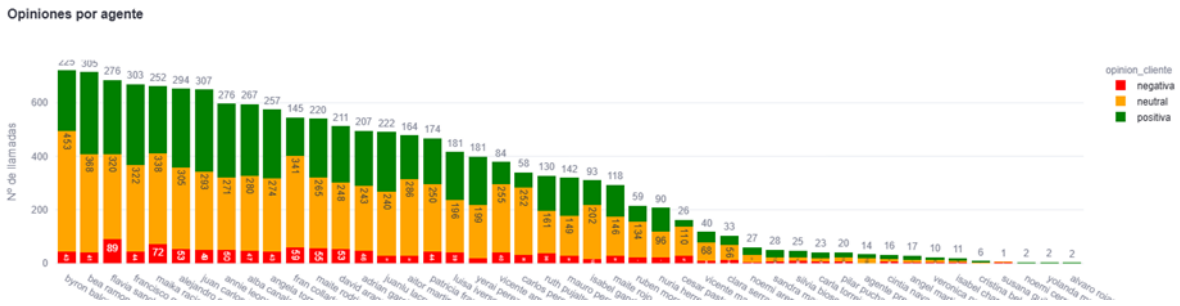


Ilustración 7: Customer opinions by agent

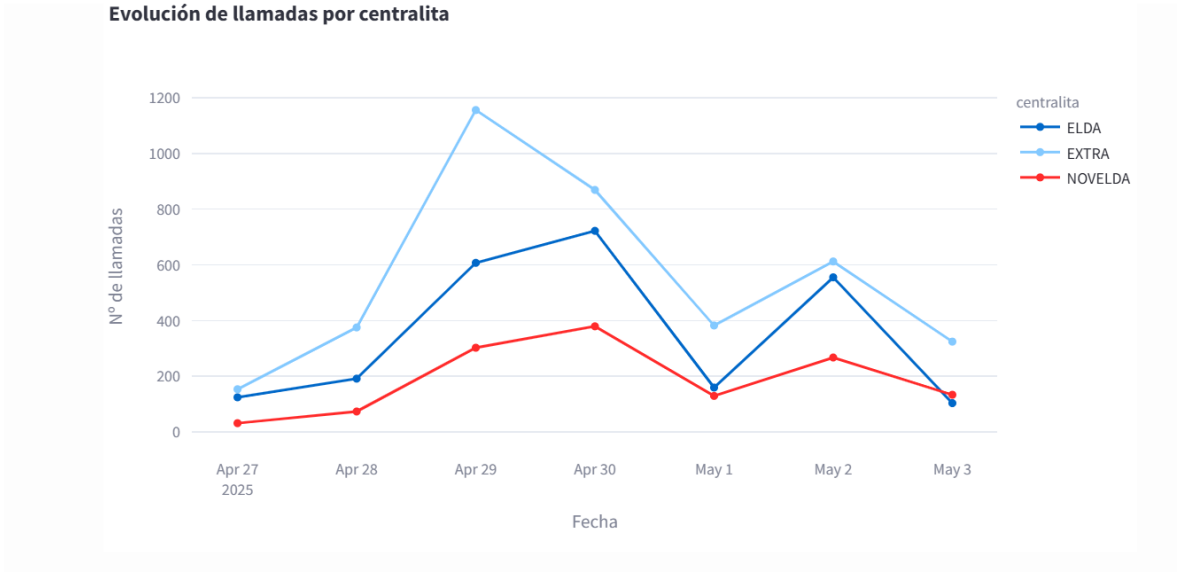


Ilustración 8: Call volume during the outage period

7. Conclusions

The system developed has radically transformed the supervision of the telephone support service, enabling the automated analysis of over 22,000 calls using artificial intelligence and natural language processing. Thanks to its modular architecture and the use of scheduled tasks, the system runs fully autonomously, generating structured and visually interpretable results each day.

The solution has enabled the detection of operational patterns, the real-time management of incidents, and the optimization of agent distribution across switchboards. The analysis of customer opinions, recurrent issues, and keyword trends has greatly improved decision-making, replacing manual spreadsheet-based processes. The interactive web dashboard allows supervisors to filter and consult data with

precision, resulting in a significant increase in the efficiency of the customer service center.

Despite the progress made, the system still presents certain limitations, such as partial coverage (outgoing calls are not yet included), ambiguity in some automated classifications, and the absence of real-time alerting. Nevertheless, it lays a strong foundation for the future development of a comprehensive, real-time operational monitoring system.

8.References

[1] Alpha Cephei. (2024). *Vosk Speech Recognition Toolkit*. https://alphacephei.com/vosk/server

[2] OpenAI. (2023). *GPT-3.5 Turbo overview*. https://platform.openai.com/docs/models/gpt-3-5

[3] Jurafsky, D., & Martin, J. H. (2021). *Speech and Language Processing* (3rd ed.). https://web.stanford.edu/~jurafsky/slp3/

[4] Python Software Foundation. (s.f.). *multiprocessing — Process-based parallelism*. https://docs.python.org/3/library/multiprocessing.html

[5] Streamlit Inc. (s.f.). *Streamlit — The fastest way to build and share data apps*. https://streamlit.io/

[6] Postman. (s.f.). *API platform for building and using APIs*. https://www.postman.com/

[7] McKinney, W. (2010). *Data Structures for Statistical Computing in Python*. Proceedings of the 9th Python in Science Conference, 51–56. https://pandas.pydata.org/

FFmpeg. (s.f.). *FFmpeg Documentation*. https://ffmpeg.org/documentation.html

The PostgreSQL Global Development Group. (s.f.). *psycopg2 – PostgreSQL database adapter for Python*. https://www.psycopg.org/

[8] Google Cloud. (2024). *Speech-to-Text API Documentation*. https://cloud.google.com/speech-to-text

[9] Amazon Web Services. (s.f.). *Amazon Transcribe – Automatic Speech Recognition*. https://aws.amazon.com/transcribe/

[10] Upbe. (2024). *Product Overview*. https://www.upbe.ai/

[11] PostgreSQL Global Development Group. (2023). *PostgreSQL Documentation*. https://www.postgresql.org/docs/

[12] Amazon Web Services. (2024). *Amazon Contact Lens for Amazon Connect*. https://aws.amazon.com/connect/contact-lens/

[13] Lombardi, P., & Latona, D. (2025, junio 17). *Miscalculation by Spanish power grid operator REE contributed to massive blackout*. Reuters. https://www.reuters.com/business/energy/investigation-into-spains-april-28-blackout-shows-no-evidence-cyberattack

Índice de la memoria

Índice de tablas

Índice de Ilustraciones

Índice de códigos

Capítulo 1. INTRODUCCIÓN

En el contexto actual de las telecomunicaciones, los centros de atención telefónica generan un volumen creciente de información valiosa, en su mayoría no estructurada. Transformar estas grabaciones en datos accionables mediante inteligencia artificial abre nuevas oportunidades para mejorar la eficiencia operativa y la toma de decisiones basada en evidencia.

Cableworld, como operador de telecomunicaciones, gestiona diariamente miles de llamadas a través de su Call center. Sin embargo, carece de un sistema automatizado que permita extraer conocimiento útil de estas interacciones.

Este proyecto propone el desarrollo de una solución escalable y automatizada que transforme las llamadas en datos estructurados, mediante técnicas de transcripción y procesamiento de lenguaje natural (NLP). El objetivo es crear una herramienta estratégica que facilite el análisis semántico automático de conversaciones, integrando los resultados en la infraestructura de datos de la empresa.

1.1 MOTIVACIÓN DEL PROYECTO

En el contexto actual, caracterizado por la creciente digitalización de los procesos empresariales, la capacidad de extraer valor de datos no estructurados representa una ventaja competitiva significativa. En particular, el análisis automatizado de llamadas telefónicas en centros de atención al cliente permite identificar patrones recurrentes, detectar necesidades operativas, evaluar el desempeño del personal y fundamentar la toma de decisiones en indicadores objetivos.

En el caso concreto de Cableworld, el volumen de llamadas gestionadas diariamente a través de la plataforma Enreach dificulta la revisión manual y sistemática de las

conversaciones. La ausencia de mecanismos automáticos de transcripción y análisis limita gravemente la capacidad de explotar esta fuente de información crítica.

Este proyecto se plantea con una doble finalidad: por un lado, dar respuesta a una necesidad técnica concreta de la organización, y por otro, explorar el potencial de las tecnologías de inteligencia artificial para mejorar procesos clave en entornos operativos reales. La solución propuesta se basa en herramientas tecnológicas avanzadas y en el uso de software de código abierto ampliamente validado en el ámbito académico y profesional, lo que garantiza flexibilidad, control total sobre los datos y sostenibilidad económica a largo plazo.

El objetivo es dotar a la empresa de una herramienta estratégica que transforme las llamadas en un activo de alto valor analítico, facilitando una supervisión más inteligente, automatizada y orientada a la mejora continua del servicio.

1.2 DEFINICIÓN DEL PROYECTO

El sistema actual de gestión de llamadas utilizado por cableworld permite almacenar las grabaciones de voz y consultar metadatos básicos (número de origen, destino, duración…), pero no dispone de ningún mecanismo para transcribir automáticamente el contenido ni para realizar un análisis semántico.

Este vacío obliga a depender de procesos manuales para revisar llamadas, lo cual es inviable a gran escala y dificulta la obtención de métricas útiles sobre la actividad del *Call center*.

Además, las soluciones comerciales existentes en el mercado no se adaptan completamente a las necesidades de la empresa: o bien implican costes elevados, o bien suponen una pérdida de control sobre los datos al requerir su envío a la nube.

Por tanto, el problema a resolver consiste en diseñar e implementar un sistema propio que permita:

- Transcribir automáticamente las llamadas almacenadas.
- Analizar semánticamente su contenido (resúmenes, palabras clave, opiniones…).
- Integrarse con la infraestructura existente.
- Preservar la privacidad y garantizar un coste operativo mínimo.

El proyecto aplica técnicas de procesamiento de lenguaje natural (NLP), apoyadas en modelos avanzados como GPT, para analizar las transcripciones de llamadas telefónicas. A través de estas técnicas se extraen automáticamente etiquetas, resúmenes, opiniones, palabras clave y otros metadatos relevantes, que permiten estructurar y explotar de manera eficaz la información contenida en las interacciones con los clientes.

1.3 OBJETIVOS

Para alcanzar este propósito, se definen los siguientes objetivos, que guiarán el desarrollo e implementación del sistema propuesto:

1.3.1 OBJETIVO GENERAL

Diseñar e implementar un sistema automatizado para la transcripción y análisis semántico de llamadas telefónicas en cableworld, utilizando tecnologías de NLP e inteligencia artificial, con un enfoque modular, escalable y de bajo coste.

1.3.2 OBJETIVOS ESPECIFICOS

• Diseñar e implementar un flujo de trabajo automatizado que abarque desde la descarga de grabaciones hasta el almacenamiento estructurado de los resultados, incluyendo la gestión y limpieza del espacio en el sistema.

• Evaluar y comparar distintas tecnologías de transcripción y análisis lingüístico (Google Cloud Speech-to-Text, Amazon Transcribe, Whisper, Gemini, entre otras) en términos de precisión, coste y rendimiento.

• Aplicar técnicas avanzadas de procesamiento de lenguaje natural (NLP) para la extracción automática de etiquetas, resúmenes, opiniones, palabras clave y otros metadatos relevantes a partir de las transcripciones de las llamadas.

• Integrar el sistema de análisis con la base de datos corporativa (PostgreSQL), garantizando la correcta estructuración, trazabilidad y explotación de los datos obtenidos.

• Desarrollar un sistema de análisis semántico que permita la categorización automática de las llamadas según criterios como temática, satisfacción del cliente, reincidencia y resolución.

• Implementar mecanismos de evaluación y validación para cuantificar la fiabilidad y utilidad de los resultados obtenidos mediante las distintas tecnologías y enfoques aplicados.

• Diseñar filtros y criterios automatizados que prioricen las llamadas más relevantes en función de variables como población, agente, tipo de consulta, urgencia o criticidad.

• Optimizar el uso de recursos del sistema mediante estrategias de eliminación automatizada de archivos intermedios y control eficiente del almacenamiento disponible.

• Establecer procedimientos que aseguren la trazabilidad completa de cada llamada, desde la grabación original hasta su análisis final e inserción en la base de datos.

Capítulo 2. DESCRIPCIÓN DE LAS TECNOLOGÍAS

Este proyecto combina varias tecnologías cuidadosamente seleccionadas para automatizar el análisis de llamadas en el *Call center* de cableworld, optimizando el flujo de trabajo diario y permitiendo generar información útil para la toma de decisiones. A continuación, se explican las herramientas principales utilizadas y su papel dentro del sistema.

2.1 API ENREACH

La API de Enreach [1] es fundamental para iniciar todo el proceso, ya que permite realizar peticiones automatizadas para descargar tanto los audios de las llamadas como los archivos JSON que contienen metadatos detallados (como número, agente o departamento). Gracias a esta integración, el sistema accede de forma controlada y segura a la información diaria necesaria para alimentar el flujo de análisis.

Además, si en algún momento se detecta un dato incorrecto, por ejemplo, un número de agente que ya no está operativo, es posible acceder manualmente a la plataforma de Masvoz con las credenciales proporcionadas, y realizar los ajustes necesarios. Esta API proporciona campos clave como el departamento, duración, número de abonado y otros que se describen con mayor detalle en secciones posteriores.

2.2 VOSK

Vosk es una herramienta de reconocimiento automático de voz (ASR) de código abierto [2] [3], basada en redes neuronales profundas, que permite realizar transcripciones de audio con alta precisión y bajo consumo de recursos. A diferencia de muchas soluciones comerciales, Vosk puede ejecutarse completamente en local, lo que garantiza la **privacidad de los datos** y elimina la dependencia de servicios en la nube, un aspecto crítico cuando se manejan conversaciones sensibles de clientes.

Entre sus principales características destacan:

- **Compatibilidad con múltiples idiomas**, incluido el español, sin necesidad de entrenamiento adicional.
- **Reconocimiento en tiempo real y en batch**, con soporte tanto para grabaciones como para audio en vivo.
- **Modelos optimizados para CPU**, lo que permite ejecutarlo en dispositivos de bajo rendimiento (servidores virtuales sin GPU).
- **Interfaz nativa en Python**, lo que facilita su integración en flujos de análisis de datos o automatización.
- **Soporte para diarización básica** y detección de silencio, útil en tareas de preprocesamiento del audio.
- **Consumo de memoria y CPU reducido**, ideal para sistemas que requieren escalabilidad sin grandes costes de infraestructura.

Gracias a estas características, Vosk ha sido clave en este proyecto para transcribir de forma automatizada los audios del *Call center*, posibilitando su posterior análisis semántico. Su uso ha permitido reducir los tiempos de procesamiento, mantener el control total sobre los datos y eliminar costes asociados al uso de APIs externas, todo ello manteniendo una precisión más que aceptable para los fines del sistema.

2.3 (GPT) MODELOS DE LENGUAJE

Para el análisis semántico automatizado de las transcripciones, se ha empleado la **API de OpenAI** [4] con dos modelos diferentes de lenguaje: **GPT-3.5 Turbo** [5] y **GPT-4** [6]. Ambos han sido seleccionados por su capacidad para interpretar lenguaje natural de forma precisa y contextualizada, aunque se han utilizado en fases distintas del proyecto, en función del equilibrio entre coste, velocidad y profundidad del análisis.

2.3.1 GPT-3.5 TURBO: ANÁLISIS DIARIO DE LLAMADAS

El modelo **GPT-3.5 Turbo** ha sido utilizado para analizar diariamente todas las transcripciones de llamadas. Este modelo fue escogido por su excelente relación entre coste y rendimiento, ya que permite:

- Detectar emociones en las llamadas.
- Generar etiquetas automáticas.
- Clasificar opiniones (positiva, neutral, negativa).
- Resumir el contenido semántico.
- Identificar palabras clave y tipologías de consulta.
- Detectar localidades

Este procesamiento se integra dentro del script automatizado **analizar_todas_llamadas_gpt.py**, y su salida se almacena estructuradamente en una base de datos PostgreSQL para su posterior consulta. Todo el flujo es completamente autónomo, escalable y no requiere intervención manual.

2.3.2 GPT-4: CONSULTAS EN LENGUAJE NATURAL DESDE EL DASHBOARD

Por otro lado, el modelo **GPT-4** [7] se ha integrado en la interfaz desarrollada con **Streamlit**, permitiendo a los usuarios realizar consultas directamente en lenguaje natural sobre los datos registrados. Esta funcionalidad está pensada para personal administrativo o de gestión que no tiene conocimientos técnicos avanzados, y que necesita respuestas rápidas sin necesidad de escribir *queries* SQL.

Por ejemplo, el usuario puede preguntar:

"¿Cuáles fueron las llamadas con opinión negativa del lunes pasado en Novelda?" y el sistema, mediante GPT-4, interpreta la intención, traduce la consulta y muestra los resultados automáticamente.

Esta funcionalidad convierte el panel en una herramienta accesible e inteligente, extendiendo las capacidades del análisis automatizado al plano interactivo y exploratorio.

2.4 PROCESAMIENTO DE LENGUAJE NATURAL (PLN)

El procesamiento de lenguaje natural (PLN) en este proyecto no es solo un concepto teórico, sino una capa práctica e integrada que da sentido a todo el análisis. Se emplea para limpiar las transcripciones eliminando introducciones automáticas (como saludos repetidos), normalizar los textos y detectar automáticamente palabras clave relevantes dentro de categorías temáticas. Su principal aporte reside en la generación de un análisis semántico enriquecido gracias al uso de modelos como GPT.

Además, el PLN permite comparar frases comunes para detectar resoluciones o incidencias no resueltas, clasificar emociones y distinguir entre llamadas técnicas y comerciales. También facilita la agrupación de llamadas por temática, la detección de intenciones y la evaluación de la satisfacción del cliente de forma no intrusiva. Este enfoque permite extraer inteligencia accionable a partir del lenguaje espontáneo de los usuarios, algo imposible con métodos manuales o puramente estructurados.

En resumen, el PLN actúa como el puente entre el texto bruto que produce el motor de reconocimiento de voz y los datos analíticos que se almacenan en la base de datos. Esta capa intermedia convierte el lenguaje natural en información estructurada que puede explotarse para mejorar la calidad del servicio y apoyar la toma de decisiones en tiempo real. [8]

2.5 AUTOMATIZACIÓN DIARIA CON CRON JOBS EN LINUX

El sistema emplea **cron jobs**, una herramienta nativa de los sistemas Unix/Linux, para programar la ejecución automática y secuencial de los distintos scripts que conforman el

flujo de trabajo diario. Gracias a esta automatización, se garantiza que las tareas de descarga de audios, extracción de metadatos, descompresión, procesamiento, transcripción, análisis semántico y limpieza de carpetas se realicen de forma autónoma cada día, sin requerir intervención manual.

Esta metodología no solo optimiza el uso de los recursos humanos y técnicos disponibles, sino que también asegura la regularidad, fiabilidad y escalabilidad del sistema en entornos de producción. La automatización permite mantener un ciclo continuo de procesamiento que adapta el sistema a nuevas llamadas entrantes de forma diaria, garantizando así que los responsables del *Call center* dispongan de datos actualizados en tiempo real para la toma de decisiones [9].

2.6 BASE DE DATOS POSTGRESQL

Todos los resultados analizados se almacenan en una base de datos **PostgreSQL**, seleccionada por su **robustez, eficiencia, escalabilidad y compatibilidad con sistemas empresariales**. PostgreSQL es un sistema de gestión de bases de datos relacional de código abierto ampliamente adoptado en entornos de producción debido a su alto rendimiento, soporte de transacciones complejas y su capacidad para manejar grandes volúmenes de datos estructurados.[10]

Esta base de datos es necesaria por varias razones clave:

- **Consultas avanzadas y personalizadas**: permite filtrar resultados por fecha, agente, localidad, tipo de consulta o cliente, lo que facilita la extracción de *insights* específicos de manera rápida.
- **Integración directa con herramientas analíticas**: como *dashboards* en Streamlit o conexiones ODBC con otras plataformas internas.

- **Persistencia histórica**: conserva todos los registros en una estructura optimizada, esencial para llevar a cabo análisis longitudinales, auditorías o comparativas temporales.

- **Fiabilidad y seguridad**: garantiza la integridad de los datos mediante control de transacciones (ACID), autenticación segura y *backups* automatizados.

En el contexto de este proyecto, PostgreSQL permite convertir el análisis semántico de las llamadas en un **repositorio estructurado y consultable** en tiempo real, que sirve como base para la toma de decisiones estratégicas, el seguimiento de incidencias y la evaluación del rendimiento operativo. [11]

2.7 OPTIMIZACIÓN MEDIANTE PROCESAMIENTO EN PARALELO

Dado el elevado volumen de llamadas que deben transcribirse diariamente, se optó por implementar una solución basada en procesamiento paralelo multihilo utilizando la librería *multiprocessing* [12] de Python. El script **transcribir_paralelo.py** distribuye automáticamente la carga entre los diferentes núcleos del servidor, permitiendo procesar múltiples archivos de audio de forma simultánea y eficiente.

Esta estrategia ha supuesto una mejora sustancial en los tiempos de ejecución, reduciendo el procesamiento de más de 1.000 archivos de varias horas a apenas una hora y algunos minutos, dependiendo del número de núcleos disponibles. Esta aceleración ha sido clave para garantizar que el sistema sea escalable, sostenible y operativamente viable a largo plazo.

El flujo paralelo se complementa con el uso de ffmpeg [13] para la conversión previa de los archivos a formato WAV, y el motor de reconocimiento Vosk, que asegura una transcripción robusta y compatible con el procesamiento automatizado posterior. La combinación de herramientas ligeras y procesamiento distribuido convierte esta solución en una arquitectura adaptada a entornos exigentes con recursos computacionales limitados.

2.8 SCRIPTS EN BASH Y PYTHON (RESUMEN FUNCIONAL)

A lo largo del desarrollo del sistema se han implementado diversos scripts que automatizan cada etapa del proceso diario. Estos scripts, escritos en Bash y Python, se ejecutan de forma secuencial y están programados mediante tareas cron en el servidor Linux. Su papel es clave para garantizar la continuidad operativa y la eficiencia del sistema:

- **descarga_audio.sh**: Realiza la descarga automatizada de los audios desde la API de Enreach.
- **download_ATC.sh**: Obtiene los ficheros JSON que contienen los metadatos de las llamadas (agentes, números, etc.).
- **procesar_json.sh**: Extrae y transforma los datos relevantes de esos ficheros para su posterior uso.
- **unzip_zipped0.sh**: Descomprime los archivos .zip que contienen los audios.
- **search_call_agent.sh**: Clasifica los audios por agente y localidad, aunque en la versión final no se integra directamente.
- **transcribir_paralelo.py**: Realiza la transcripción paralela de los audios usando procesamiento multihilo.
- **analizar_todas_llamadas_gpt.py**: Analiza las transcripciones con GPT, genera estructuras JSON y las inserta en la base de datos.
- **delete_all.sh***:* Script para eliminar automáticamente las carpetas ya analizadas, optimizando así el uso del almacenamiento del servidor.

```
00 02 * * * root /bin/bash /etc/cron.daily/descarga_audio.sh
05 02 * * * root /bin/bash /etc/cron.daily/download_ATC.sh
30 02 * * * root /bin/bash /etc/cron.daily/procesar_json.sh
0 05 * * * root /bin/bash /etc/cron.daily/unzip_zipped0.sh
15 05 * * * root /bin/bash /etc/cron.daily/search_call_agent.sh
20 06 * * * root /bin/bash /etc/cron.daily/lanzar_transcripcion.sh
45 08 * * * root python3 /home/bea/scripts/analizar_todas_llamadas_gpt.py
00 16 * * * root python3 /home/bea/scripts/delete_all.py
```

Ilustración 9: Captura desde crontab con el flujo completo de la auutomatización

Gracias a esta estructura modular, el sistema logra una automatización robusta, escalable y fácilmente mantenible, clave para su operatividad diaria sin intervención humana.

2.9 ECOSISTEMA Y HERRAMIENTAS COMPLEMENTARIAS

A lo largo del desarrollo del proyecto, se ha hecho uso de un conjunto de herramientas auxiliares fundamentales que han facilitado tanto la integración del sistema como su operatividad diaria. Estas herramientas, aunque no constituyen el núcleo del sistema, han resultado claves para garantizar su automatización, fiabilidad y mantenibilidad.

2.9.1 SERVIDOR FTP: CREACIÓN Y CONFIGURACIÓN PERSONALIZADA

Uno de los hitos clave del sistema ha sido la **implementación manual de un servidor FTP seguro**, diseñado específicamente para recibir de forma automatizada las grabaciones diarias procedentes de Enreach. Para ello, se desplegó y configuró **vsftpd (Very Secure FTP Daemon** [14] sobre una máquina virtual con Ubuntu. Esta solución fue elegida por su fiabilidad, rendimiento y alto nivel de personalización, permitiendo gestionar usuarios virtuales, restringir permisos de acceso y mantener la integridad de los datos.
A diferencia de soluciones preempaquetadas como Pure-FTPd, vsftpd ofrecía mayor granularidad en la configuración, lo que fue determinante para asegurar una **recepción robusta y segura**. El acceso al servidor se restringe exclusivamente a través de una **VPN L2TP/IPsec**, lo que garantiza que solo dispositivos autorizados puedan transferir datos, cumpliendo así con los requisitos de confidencialidad y protección definidos por la organización. El servidor actúa como **repositorio temporal**, y se vacía periódicamente para evitar congestión y facilitar el mantenimiento operativo. [15]

2.9.2 HERRAMIENTAS DE TESTING DE API (POSTMAN)

Durante la integración inicial con la API de Enreach, se utilizó **Postman** [16] como entorno de pruebas para validar la autenticación, el formato de respuesta y el comportamiento de los distintos *endpoints*. Esta herramienta resultó esencial para depurar errores de conexión, entender la estructura de los JSON devueltos y simular las peticiones antes de automatizarlas mediante scripts. Gracias a esta fase de validación, se minimizó el riesgo de fallos en producción.

Aquí hay un ejemplo de la petición que se mandó para saber el número de llamadas en octubre

Llamadas Octubre 2024

https://manager.masvoz.es/api/rs/call/stats/ATC?accountId=10131&startDate=2024-10-01+08:00:00&endDate=2024-10-31+23:59:59


```
{
        "pivot": 1727762400000,
        "label": "",
        "color": 0,
        "answered": 19813,
        "noanswer": 1662,
        "busy": 156,
        "failed": 32,
        "abandoned": 10709,
        "billsec": 5676771,
        "duration": 5680818
    }
```


Ilustración 10:Postman

2.9.3 LIBRERÍAS AUXILIARES

El sistema integra varias librerías de Python que han resultado críticas para la eficiencia de los procesos:

- **Pandas**: utilizada para la gestión de estructuras tabulares, ha permitido vincular los audios con los datos de los agentes y localidades, generar informes intermedios y aplicar transformaciones para la limpieza y clasificación de datos. [17]

- **psycopg2**: responsable de la conexión directa con la base de datos PostgreSQL, esta librería se ha empleado para insertar los análisis generados por GPT y asegurar transacciones consistentes. [18]
- **ffmpeg**: herramienta de línea de comandos utilizada para convertir archivos de audio al formato .wav, normalizando parámetros como frecuencia y canales para asegurar compatibilidad con Vosk. Su ejecución está automatizada en el flujo de preprocesamiento de audios.
- **multiprocessing**: clave para habilitar la transcripción en paralelo. Esta librería divide automáticamente los archivos entre los núcleos del servidor, reduciendo significativamente el tiempo de ejecución frente a un enfoque secuencial.
- **os, shutil y glob**: permiten recorrer y gestionar la estructura de carpetas del sistema, mover archivos entre rutas, y localizar elementos como audios comprimidos o carpetas UUID específicas. [19]
- **json y csv**: empleadas para leer y exportar los metadatos de las llamadas descargados desde la API, así como para generar resúmenes de relaciones (por ejemplo, agente-localidad) en formatos estructurados. [20]
- **openai**: librería oficial para conectarse a la API de OpenAI, utilizada para enviar las transcripciones a GPT-3.5 (y en el *dashboard* a GPT-4) y recuperar análisis semánticos enriquecidos.
- **datetime y time**: utilizadas para automatizar la detección de fechas, programar tareas cron y nombrar los archivos generados con precisión temporal. [21]
- **re (expresiones regulares)**: imprescindible para extraer datos específicos desde nombres de archivo, como número de abonado, agente, hora, o localidad, y para estructurar correctamente la entrada textual al análisis semántico. [22]

Gracias a esta combinación de librerías, el sistema ha podido desarrollarse de forma modular, eficiente y totalmente automatizada, permitiendo procesar miles de llamadas diarias con mínimos recursos humanos y con un control completo sobre todo el flujo de datos.

Ilustración 11:Algunas de las librerías utilizadas en el proyecto

2.9.4 INTERFAZ WEB DE VISUALIZACIÓN DE RESULTADOS

Con el objetivo de hacer accesible el análisis automatizado a usuarios no técnicos, se ha desarrollado un ***frontend* interactivo** mediante **Streamlit** [23], un *framework* ligero de Python ideal para construir paneles visuales de forma rápida y eficiente. Esta interfaz actúa como nexo entre los datos almacenados en la base de datos PostgreSQL y los responsables de supervisión, facilitando la consulta y comprensión de los resultados generados por el sistema.

El *dashboard* se conecta directamente con la base de datos y permite aplicar filtros dinámicos para obtener visualizaciones personalizadas en tiempo real. Durante la fase de desarrollo, se utilizó **Ngrok** [24] para exponer temporalmente la aplicación sin necesidad de modificar el firewall del servidor, asegurando así un acceso remoto seguro para pruebas. Entre sus funcionalidades principales destacan:

- Filtros por agente, localidad, centralita, fecha, tipo de cliente, tipo de llamada, palabras clave y número de abonado.
- Visualización de evolución temporal de llamadas por localidad, agente o centralita.

- Análisis gráfico de opiniones (positiva, negativa, neutral) con colores personalizados.
- Estadísticas sobre reincidencias y llamadas derivadas.
- Nube de etiquetas y palabras clave más frecuentes.
- Métricas agregadas como duración media por llamada y por agente.
- **Consulta inteligente en lenguaje natural**, alimentada por GPT-4o, que permite al usuario introducir preguntas o criterios de búsqueda en lenguaje cotidiano y recibir respuestas basadas en datos filtrados directamente desde la base.

El despliegue final se realizará en un entorno web seguro, con autenticación individual y cifrado de credenciales, alineado con las recomendaciones del Esquema Nacional de Seguridad (ENS) [25] . Esta capa de visualización garantiza que la explotación del sistema no dependa de conocimientos técnicos en programación ni bases de datos, democratizando así el acceso a la información estratégica contenida en las llamadas.

Esta captura demuestra que el acceso al sistema está restringido mediante credenciales personalizadas y que el diseño se ha adaptado a la identidad corporativa de cableworld.

Ilustración 12:Panel inicio sesión

Esta captura es de la ventana principal del *dashboard*, donde se muestran métricas clave (número de llamadas, reincidencias, duración media, etc.). Esta vista proporciona un resumen ejecutivo en tiempo real para la supervisión diaria.

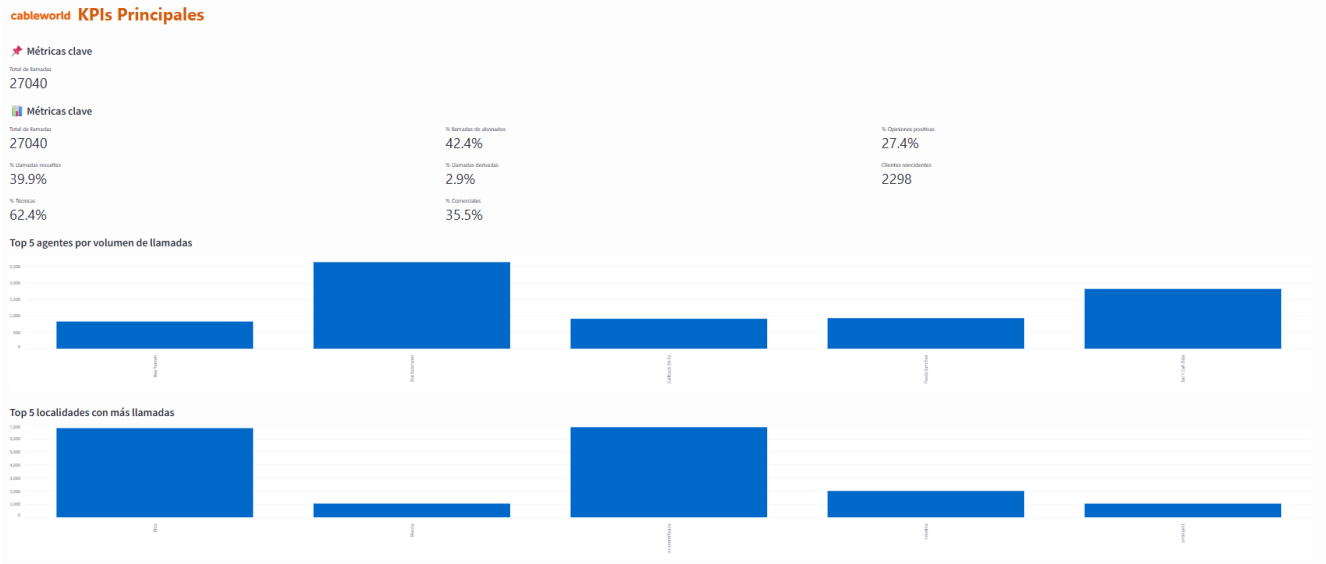


Ilustración 13:Ventana KPIs

Esta captura es de la sección donde se pueden filtrar llamadas por número, agente, localidad o fecha, y consultar análisis individuales (opinión, palabras clave, tipo de llamada, etc.). Esta funcionalidad permite explorar casos concretos y tomar decisiones personalizadas.

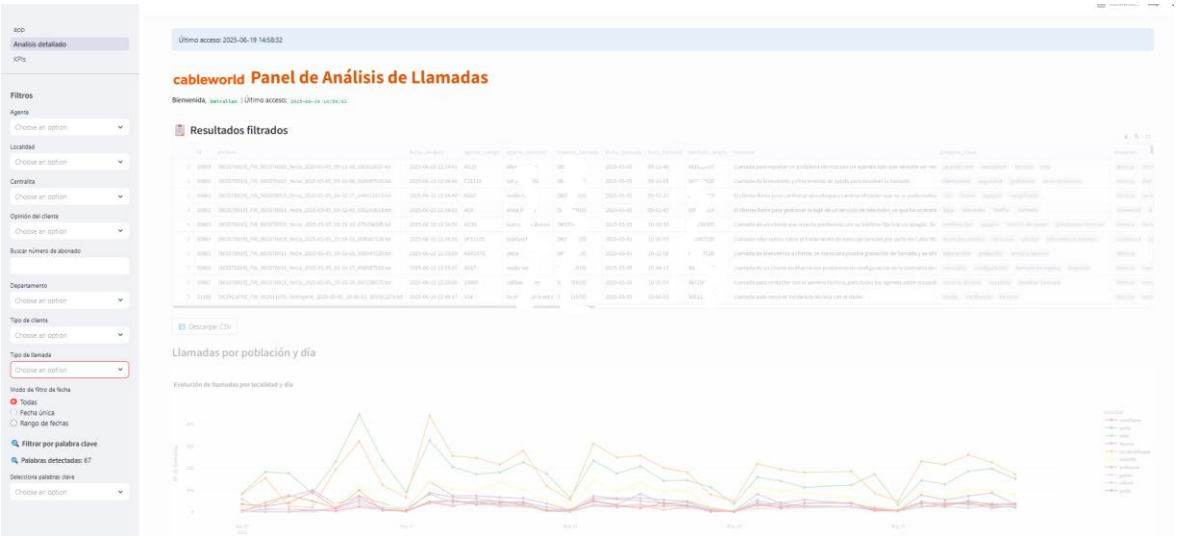


Ilustración 14:Análisis detallado

Capítulo 3. estado de la cuestión

Este capítulo contextualiza la necesidad del proyecto analizando las limitaciones del sistema anterior de cableworld y explorando las tecnologías disponibles en el mercado. El análisis de llamadas se realizaba manualmente, sin transcripciones ni métricas objetivas, lo que dificultaba la explotación de los datos y el control de calidad del servicio.

Se evalúan soluciones comerciales como **Google Speech-to-Text**, **Amazon Contact Lens** o **Upbe**, identificando barreras clave como el coste elevado, la dependencia de la nube o la escasa personalización. También se analiza el volumen real de llamadas que debía soportar el sistema (más de 1.000 horas de audio al mes), lo que refuerza la necesidad de una solución escalable y asequible.

Finalmente, se justifica la elección de una arquitectura propia basada en **Vosk** (transcripción local gratuita) y **GPT-3.5** (análisis semántico de bajo coste), que ofrece el equilibrio óptimo entre rendimiento, coste, control de datos y personalización. Se incluye una comparativa técnica y económica que demuestra su superioridad frente a alternativas comerciales.

3.1 SITUACIÓN ACTUAL EN CABLEWORLD

Hasta el momento, el análisis de llamadas en cableworld se realizaba de forma manual y fragmentada. Las grabaciones se almacenaban sin una estructura unificada, y su revisión quedaba limitada a auditorías internas puntuales o a la escucha directa por parte de los responsables de cada departamento. Esta metodología, aunque funcional para un volumen reducido de interacciones, presentaba múltiples limitaciones a medida que crecía el número de llamadas y se ampliaban los objetivos de análisis.

Entre las principales carencias del sistema previo se encuentran:

Ausencia de transcripción automatizada: las llamadas no se convertían en texto, lo que dificultaba cualquier tipo de análisis textual o semántico.

Falta de indicadores objetivos: no existían métricas cuantificables sobre la calidad del servicio, la satisfacción del cliente, la tipología de las consultas o la actividad de los agentes.

Imposibilidad de explotación masiva: al no estar los datos estructurados, no era viable realizar análisis longitudinales, comparar poblaciones o detectar patrones de comportamiento a lo largo del tiempo.

Dependencia humana: cualquier revisión debía hacerse escuchando manualmente las llamadas, lo que suponía un alto consumo de tiempo y recursos.

Falta de análisis del comportamiento de los agentes: no se evaluaba de manera sistemática cómo actuaban los agentes frente a los diferentes tipos de consulta, ni la efectividad real de las respuestas ofrecidas.

Atención al cliente no homogénea: se detectaban diferencias importantes entre agentes; algunos ofrecían respuestas rápidas, pero no efectivas, derivando en múltiples llamadas del mismo cliente y generando insatisfacción. No existía un control sobre la correcta resolución de los problemas ni sobre la calidad del trato al cliente.

Este contexto evidenció la necesidad de desarrollar una solución automatizada, escalable y fiable que permitiera estructurar, analizar y explotar la información contenida en las llamadas. Además, resultaba esencial incorporar un enfoque orientado a la evaluación del comportamiento y efectividad de los agentes, con el fin de homogeneizar el nivel de atención ofrecido y mejorar la experiencia global del cliente.

3.2 TECNOLOGÍAS EXISTENTES EN EL MERCADO

Como se ha detallado previamente en el capítulo 2, la plataforma **Enreach** actualmente utilizada en cableworld presenta importantes limitaciones en cuanto a la transcripción automática y el análisis avanzado de las llamadas, lo que ha motivado la necesidad de

desarrollar una solución propia que permita extraer valor de estos datos. A partir de este contexto, durante el proyecto se ha llevado a cabo un análisis exhaustivo de las tecnologías disponibles en el mercado para el análisis automatizado de llamadas. Existen múltiples productos y servicios orientados a la transcripción, análisis semántico y explotación de datos en entornos de atención al cliente. Sin embargo, cada uno de ellos presenta limitaciones que, en el caso específico de cableworld, impedían una adopción directa.

3.3 JUSTIFICACIÓN DEL VOLUMEN DE ANÁLISIS

Antes de decidir qué tecnología adoptar, fue importante estimar qué volumen de llamadas se tendría que procesar. A lo largo del proyecto se analizaron los datos históricos de la API de Masvoz, observándose que el *Call center* de Cableworld supera habitualmente las 20.000 llamadas respondidas al mes, con más de 1,2 millones de segundos facturados mensualmente (equivalentes a unas 1.200 a 1.500 horas de conversación real).

No obstante, no todas las llamadas resultan relevantes para el análisis: se excluyen las que solo contienen locuciones automáticas o que son técnicamente fallidas. En base a la experiencia adquirida durante el desarrollo del sistema y las primeras pruebas, se estableció como volumen de referencia **1.000 horas de audio al mes**. Este umbral permite garantizar una cobertura representativa de la actividad del *Call center*, a la vez que mantiene la viabilidad económica del análisis, como se muestra en la comparativa de costes que se expone más adelante.

Mes	***answered***	***billsec (segundos facturados)***	***billsec en horas aprox.***
Octubre 2024	19.813	5.676.771	~1577 h
Noviembre 2024	17.683	5.217.111	~1449 h
Diciembre 2024	17.695	5.209.544	~1447 h
Enero 2025	17.659	5.422.077	~1506 h
Febrero 2025	15.865	4.650.584	~1292 h
Marzo 2025	15.554	4.408.897	~1224 h

Tabla 1: Tabla de minutos sacada de la api Enreach a través de Postman

Para obtener esta información se hizo una solicitud GET a desde Postman a través de la API

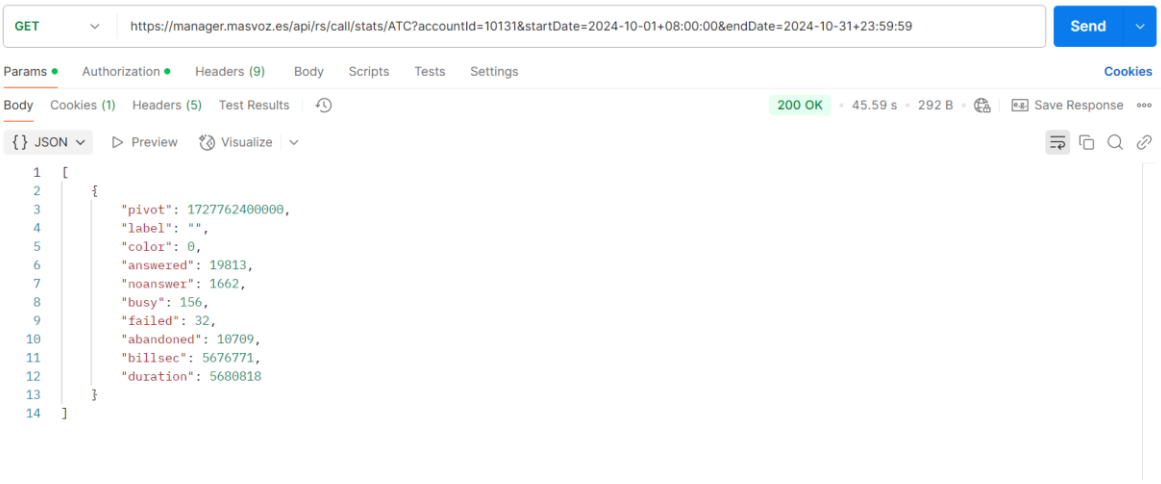


Ilustración 15:Ejemplo solicitud GET postman en el mes de Octubre

Aunque el volumen real supera las 1.200 horas mensuales, se toma como referencia 1.000 horas/mes para facilitar la comparación entre servicios y contemplar un escenario base sostenible.

3.2.1 API DE ENREACH

Como punto de partida, la propia API de Enreach llamada *MasVoz* permite acceder a las grabaciones de las llamadas, así como a ciertos metadatos asociados (números de origen y destino, fechas, duración de la llamada, agente asignado). Sin embargo, la API no ofrece ninguna funcionalidad de transcripción automática ni de análisis semántico o de comportamiento. Además, la explotación de los datos queda completamente a cargo del cliente, sin integraciones predefinidas con herramientas de *business intelligence* o plataformas de análisis. Esta carencia ha sido una de las principales motivaciones para el desarrollo del sistema propuesto.

3.4 PRODUCTOS COMERCIALES DE ANÁLISIS DE VOZ

Durante el desarrollo del proyecto se investigaron y, en algunos casos, se probaron distintas soluciones comerciales que ofrecen funcionalidades de análisis de llamadas. A continuación, se resumen sus principales características y limitaciones:

- **Google Cloud Speech-to-Text**: API que convierte el audio en texto con alta precisión, permitiendo transcripción en varios idiomas y modelos personalizables. Sin embargo, no incluye herramientas integradas para el análisis de palabras clave ni visualización avanzada de resultados, lo que obliga a construir todo el flujo de análisis por separado. [26]

- **Amazon Transcribe**: Servicio de transcripción que permite identificar hablantes en una conversación (*diarización*) y proporciona análisis básicos de sentimientos mediante su integración con Amazon Contact Lens. Una ventaja es que distingue automáticamente entre agente y cliente en grabaciones estéreo, pero para realizar un análisis más profundo y adaptable a los objetivos de cableworld es necesario complementarlo con herramientas adicionales. [27]

- **Upbe**: Software especializado en análisis de conversaciones en *Call centers*. Ofrece detección de palabras clave, análisis de sentimientos, generación de informes automáticos y evaluación del rendimiento de los agentes. No obstante, se trata de una plataforma cerrada con escasa posibilidad de personalización y un coste elevado.[28]

- **Ringover**: Herramienta orientada a la comunicación empresarial, que incluye análisis básico de llamadas mediante inteligencia artificial. Automatiza procesos [29]como el enrutamiento de llamadas y genera *insights* básicos, pero no ofrece análisis en profundidad ni personalización avanzada, elementos clave en este proyecto.

- **Amazon Contact Lens**: Servicio integrado en Amazon Connect que proporciona transcripción, análisis de sentimientos, métricas de conversación y detección de

palabras clave. Su uso en el proyecto permitió comprobar que ofrece buenos resultados en análisis emocional, pero presenta limitaciones cuando las grabaciones no se encuentran en estéreo o tienen voces mezcladas, como ocurre en parte de los audios gestionados por Cableworld [30].

- **Google Cloud Contact Center AI (CCAI)**: Plataforma avanzada que combina transcripción y análisis de interacciones, con funcionalidades como detección de tendencias y recomendaciones en tiempo real. Sin embargo, su uso implica una fuerte dependencia del ecosistema Google Cloud, lo que supone un coste elevado y un nivel de integración poco compatible con la infraestructura interna de la empresa. [31]

- **Azure Cognitive Services - Speech to Text + Conversational Analytics**: Alternativa similar a las anteriores, con buena calidad de transcripción y análisis básico de conversaciones. Presenta limitaciones en la diarización y escasa personalización de etiquetas y modelos. [32]

A nivel técnico, también se comprobó durante el proyecto que soluciones como Google Cloud y Amazon requieren configurar *buckets* de almacenamiento en la nube y una integración relativamente compleja (creación de credenciales, manejo de flujos de datos...). En cambio, soluciones como Vosk permiten un flujo mucho más ágil y flexible para el contexto de cableworld, aspecto que fue clave en la decisión técnica.

Ilustración 16:Algunas de las tecnologías ya existentes

3.4.1 ANÁLISIS COMPARATIVO DE COSTES

Con el objetivo de evaluar la viabilidad económica de las distintas alternativas, se realizó un análisis comparativo tomando como referencia un volumen de **1.000 horas/mes** de audio

procesado, que se considera un escenario equilibrado y sostenible para cableworld. [33] [34] [35]

Solución	***Coste estimado por 1000 horas/mes***	***Limitaciónes principales***
Upbe	> 2.000 €/mes (licencia + consumo)	Plataforma cerrada, sin control de datos
Amazon Contact Lens	~1.200-1.500 €/mes	Coste alto, integración dependiente de AWS
Google Cloud CCAI	> 1.500 €/mes	Dependencia de Google Cloud, coste elevado
Azure Speech Services	~800-1.000 €/mes	Limitaciones en personalización y calidad de diarización
Solución propia (Vosk + GPT-3.5)	< 100 €/mes	Mayor esfuerzo inicial de desarrollo

Tabla 2: Comparativa de costes de componentes de transcripción y análisis IA

Este análisis demostró que ninguna solución comercial ofrecía el equilibrio adecuado entre coste, personalización, integración y control de datos requerido por cableworld. Los costes mensuales de las soluciones "llave en mano" resultaban inasumibles para un procesamiento diario de varias decenas de horas, mientras que las limitaciones en la personalización impedían adaptar el análisis a los objetivos específicos de la empresa.

El análisis de mercado realizado a lo largo del proyecto, complementado con diversas pruebas prácticas, demostró que, si bien existen soluciones avanzadas para el análisis de

llamadas, estas presentan barreras importantes en términos de coste, flexibilidad y control sobre el flujo de datos. En el contexto de cableworld, con la necesidad de:

- controlar completamente los datos sensibles de las llamadas;
- personalizar el análisis (etiquetas propias, palabras clave, métricas específicas);
- integrar la solución en la infraestructura existente (base de datos PostgreSQL, servidores propios);
- y garantizar un coste sostenible en el tiempo;

la opción más adecuada fue el desarrollo de un sistema propio, modular, basado en tecnologías abiertas (**Vosk**, GPT, PostgreSQL) y apoyado puntualmente en servicios en la nube bajo demanda.

Este planteamiento no solo permite satisfacer los requisitos actuales, sino que proporciona una base escalable y flexible para futuras ampliaciones y evoluciones del sistema, tal y como se ha comprobado durante el desarrollo práctico del proyecto.

Además del análisis de soluciones comerciales completas, durante el desarrollo del proyecto también se realizó una evaluación comparativa de los componentes individuales de transcripción y análisis semántico, con el objetivo de seleccionar los más adecuados para una solución propia flexible y optimizada en costes.

En la siguiente tabla se resume esta comparativa, que permitió elegir de forma razonada la combinación **Vosk** (transcripción local gratuita) + **GPT-3.5** (análisis semántico a bajo coste), descartando otras opciones más costosas, menos precisas o integrables en la arquitectura actual.

Proveedor	***Transcripción***	***Separación hablantes***	***Análisis IA***	***Total/ llamada***	***Total mensual***

Google SST	~€0,0056 / 15s	Sí	No	~€0,089	~€268,00
AWS Transcribe	~€0,000372 / s	Sí	No	~€0,089	~€268,00
Whisper (OpenAI o local)	Gratis local / API TBD	Inferido	No	gratuito	Gratuito
Vosk	Gratis local	No	No	Gratuito	Gratuito
GPT-3.5	No	Inferido	~€0,0025	~€0,0025	~€7,53
GPT-4	No	Inferido	~€0,04185	~€0,04185	~€1.255,50
Gemini 1.5 Pro	No(aún)	Inferido	Gratis(actualizado en junio 2025)	Muy Bajo	

Tabla 3: Comparativa de costes para decisión de IA

Asimismo, se valoraron otros criterios técnicos relevantes (coste, precisión, facilidad de integración, capacidad offline...), que reforzaron la elección adoptada para nuestro proyecto:

Criterio	***Google***	***AWS***	***Whisper***	***GPT-3.5***	***GPT-4***	***Vosk***	***Gemini***
Coste	Medio	Medio	Bajo	Muy bajo	Alto	Muy Bajo	Bajo
Precisión	Alta	Alta	Media	Alta (texto)	Muy alta	Media	Alta

Diarización	Sí	Sí	No	Inferido	Inferido	No	Inferido
Fácil de integrar	Alta	Alta	Media	Muy alta	Muy alta	Alta	Alta
Offline disponible	No	No	Sí	No	No	Sí	No

Tabla 4: Matriz de decisión técnica para selección de componentes

Este análisis complementario fue clave para justificar que la solución propuesta no solo es más económica que las plataformas comerciales, sino que también permite un mayor control y personalización en línea con los objetivos específicos de cableworld.

Finalmente, la combinación de **Vosk** para la transcripción local, junto con **GPT-3.5** para el análisis semántico, fue seleccionada como la opción más adecuada para el contexto de cableworld. **Vosk** permite realizar la transcripción de forma completamente offline y sin costes por volumen, garantizando el control total sobre los datos sensibles de las llamadas. Por su parte, **GPT-3.5** proporciona un análisis semántico de alta calidad a un coste muy reducido, lo que permite escalar el sistema a grandes volúmenes de llamadas sin comprometer la viabilidad económica. Además, ambas tecnologías se integran de forma sencilla en la infraestructura existente, lo que facilita la automatización completa del flujo de análisis desarrollado en este proyecto.

Además del uso generalizado de **GPT-3.5** para el análisis semántico, el sistema incorpora de forma puntual el modelo **GPT-4**, exclusivamente en la funcionalidad de consulta en lenguaje natural disponible en el panel de visualización. Esta barra de búsqueda inteligente permite a los responsables formular preguntas complejas y obtener respuestas estructuradas extraídas directamente de la base de datos. Dado su uso esporádico y orientado a facilitar la interpretación de datos ya analizados, el impacto económico de **GPT-4** es marginal en comparación con los beneficios operativos que aporta en términos de accesibilidad e interacción.

Además del ahorro económico directo, la solución propia evita costes ocultos relacionados con la gestión de datos en la nube, licencias cerradas, dependencia de terceros o adaptación de plataformas genéricas.

Asimismo, el uso de **PostgreSQL** como sistema de almacenamiento estructurado no genera ningún coste adicional, al tratarse de una base de datos *open-source* desplegada en servidores propios. Esta elección contribuye directamente a la sostenibilidad económica del proyecto, manteniendo la eficiencia sin comprometer la escalabilidad ni la capacidad analítica.

Capítulo 4. JUSTIFICACIÓN DEL PROYECTO

4.1 POSIBLES ALTERNATIVAS

Actualmente no existe ninguna solución comercial que cumpla de manera integral con los requisitos definidos por cableworld para el análisis automatizado de llamadas. Tanto en el ámbito de la transcripción como en el análisis semántico mediante inteligencia artificial, las soluciones disponibles presentan importantes limitaciones funcionales o implican costes inasumibles.

Si se optara por contratar un sistema cerrado que ofreciera una solución similar, la única alternativa viable sería **Upbe**, con un coste superior a **2.000 €/mes**, además de tratarse de una plataforma cerrada que no ofrece control sobre los datos. Otras opciones, como **Amazon Contact Lens**, presentan un coste más moderado (**~1.200-1.500 €/mes**), pero únicamente cubren parcialmente las necesidades del proyecto, ya que se limitan a la transcripción y análisis básico. Su implementación requeriría rediseñar e integrar de nuevo todo el sistema de almacenamiento, gestión y visualización de datos.

Adicionalmente, las soluciones basadas en plataformas **cloud** implican almacenar los datos de las llamadas en servidores externos, lo cual representa una limitación crítica en términos de privacidad y control de la información sensible, especialmente en un contexto normativo donde la protección de datos es prioritaria.

4.2 SOLUCIÓN OPTADA Y APORTACIONES AL SISTEMA ACTUAL

Ante este panorama, se optó por desarrollar una solución propia que combinase lo mejor de las tecnologías disponibles, garantizando:

- **Coste sostenible**, mediante el uso de componentes *open-source* (Vosk) y servicios de IA de bajo coste (GPT-3.5).

- **Control completo sobre los datos sensibles**, ya que la transcripción se realiza localmente, sin necesidad de enviar audios a la nube.

- **Flexibilidad**, con posibilidad de personalizar etiquetas, palabras clave y métricas específicas según los objetivos de Cableworld.

- **Integración sencilla**, gracias a una arquitectura modular compatible con la infraestructura existente (PostgreSQL, servidores internos).

- **Automatización completa**, ya que todo el flujo puede ejecutarse sin intervención manual, optimizando recursos humanos y técnicos.

En base a estas consideraciones, la combinación de **Vosk** para la transcripción offline y **GPT-3.5** para el análisis semántico fue seleccionada como la solución más adecuada. Vosk permite realizar la transcripción sin costes por volumen, con pleno control de los datos. Por su parte, **GPT-3.5** ofrece un análisis de alta calidad con un coste muy reducido, permitiendo escalar el sistema a grandes volúmenes de llamadas de forma sostenible.

Ambas tecnologías se integran sin dificultad en el sistema actual, lo que ha facilitado el desarrollo de un flujo de análisis automatizado adaptado a las necesidades reales del *Call center*. Esta solución cumple con los objetivos técnicos, económicos y de seguridad definidos al inicio del proyecto, y sienta una base robusta para futuras ampliaciones.

Aunque la solución propuesta se diseñó con criterios de eficiencia económica, durante el desarrollo del proyecto se valoraron también opciones más avanzadas en términos de calidad de análisis, especialmente en la parte de **diarización**. La empresa no habría tenido inconveniente en asumir un mayor coste si ello garantizara una mejora significativa en la separación de interlocutores. Sin embargo, se identificó un problema técnico clave: **las grabaciones proporcionadas por Enreach están en formato mono (un solo canal de audio)**, lo cual imposibilita que sistemas como Amazon Contact Lens o WhisperX detecten correctamente quién habla en cada momento.

Se intentó realizar un cambio manual del formato de audio para forzar la separación, pero esto no fue viable, ya que **la limitación proviene del sistema de grabación de Enreach y no puede modificarse desde Cableworld**. Por tanto, la decisión de optar por un sistema con **análisis basado en texto (GPT-3.5)** y transcripción local sin *diarización* avanzada fue una consecuencia directa de esta restricción técnica externa.

4.3 JUSTIFICACIÓN ECONÓMICA

Para evaluar la viabilidad económica de la solución propuesta, se realizó una comparativa entre los costes estimados de las principales alternativas comerciales y el sistema propio desarrollado.

A continuación, se presenta una estimación de costes mensuales para el sistema diseñado:

Concepto	***Coste mensual estimado***
Transcripción local con Vosk	**0 €**
Análisis semántico con GPT-3.5 (OpenAI API)	~10 **€**
Infraestructura adicional (uso de servidores propios)	**0 €**
Coste total mensual	**~10 €**

Tabla 5:Tabla de coste del sistema actual utilizado

Por contraste, las soluciones comerciales más cercanas en funcionalidad tendrían los siguientes costes:

- **Upbe**: > **2.000 €/mes**
- **Amazon Contact Lens**: **1.200-1.500 €/mes**

- **Google Cloud CCAI**: > **1.500 €/mes**

Estas cifras ponen de manifiesto el ahorro sustancial que supone el sistema desarrollado, reduciendo el coste mensual a menos del **1 %** respecto a las soluciones "llave en mano".

Además del ahorro económico directo, la solución propia evita costes ocultos relacionados con la gestión de datos en la nube, licencias cerradas, dependencia de terceros o adaptación de plataformas genéricas.

Esta justificación económica fue decisiva para validar la solución técnica elegida, ya que permite mantener un sistema escalable, seguro y personalizado, con un presupuesto mensual mínimo.

Capítulo 5. SISTEMA DESARROLLADO

5.1 DISEÑO DEL SISTEMA

Este capítulo ofrece una visión detallada de la estructura y fundamentos del sistema, describiendo la **arquitectura general**, los **casos de uso principales** y los **requisitos** que guían su desarrollo técnico. El propósito es esclarecer cómo interactúan los componentes, qué funciones prestan y qué condiciones de calidad se aseguran.

5.1.1 ARQUITECTURA GENERAL

El sistema está compuesto por los siguientes módulos principales:

- **Módulo de descarga de grabaciones:**
  Automatiza la interacción con la API de Enreach (*Masvoz*) y la descarga de grabaciones desde el servidor FTP, organizando los ficheros localmente por fecha.
- **Módulo de procesamiento de audio:**
  Descomprime los archivos recibidos, convierte los audios a formato WAV cuando es necesario, y ejecuta la transcripción automática mediante el motor Vosk.
- **Módulo de análisis semántico:**
  Procesa las transcripciones con el modelo GPT de OpenAI, extrayendo información estructurada como etiquetas, resumen, opinión del cliente, palabras clave, localidad, entre otros.
- **Módulo de almacenamiento:**
  Persiste los resultados tanto en un archivo consolidado en formato JSON como en la base de datos PostgreSQL, facilitando su explotación posterior.
- **Módulo de automatización:**
  Orquesta la ejecución automática de todo el flujo de procesamiento a través de tareas cron configuradas en el servidor.

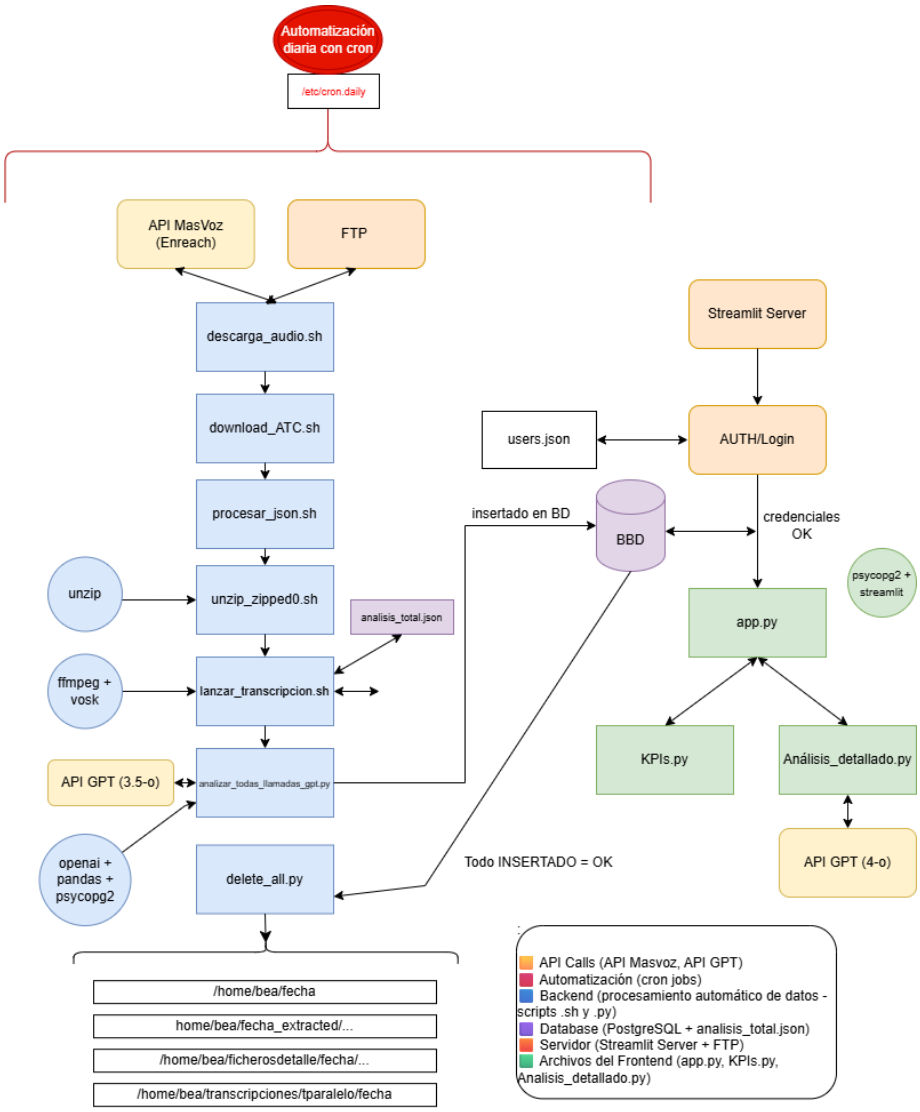


Ilustración 17:Diagrama de la arquitectura del proyecto

La figura representa la interacción entre estos módulos y los actores implicados en el sistema. El flujo completo de procesamiento está totalmente automatizado mediante cron, garantizando que las grabaciones se procesen diariamente de forma robusta y escalable. La arquitectura modular facilita además la optimización del almacenamiento y el mantenimiento de la trazabilidad de los datos procesados.

La interfaz de usuario (*frontend*), desarrollada con **Streamlit**, actúa como punto de acceso para los usuarios autorizados. Permite visualizar y filtrar los datos almacenados en la base de datos PostgreSQL mediante gráficos, tablas y buscadores inteligentes. Está compuesta por dos scripts principales: **KPIs.py**, que presenta los indicadores clave de rendimiento, y **Analisis_detallado.py**, que permite explorar en profundidad cada llamada con métricas y detalles individuales. Ambos módulos se conectan directamente a la base de datos en tiempo real, garantizando que la información mostrada esté siempre actualizada. Esta capa visual simplifica el uso del sistema para perfiles no técnicos y mejora la toma de decisiones basada en datos.

5.1.2 CASOS DE USO

El sistema desarrollado ofrece a los usuarios un conjunto de funcionalidades accesibles a través del panel interactivo **Streamlit**, complementadas con un procesamiento automático diario de las llamadas.

A continuación, se va a mostrar el diagrama de casos de uso:

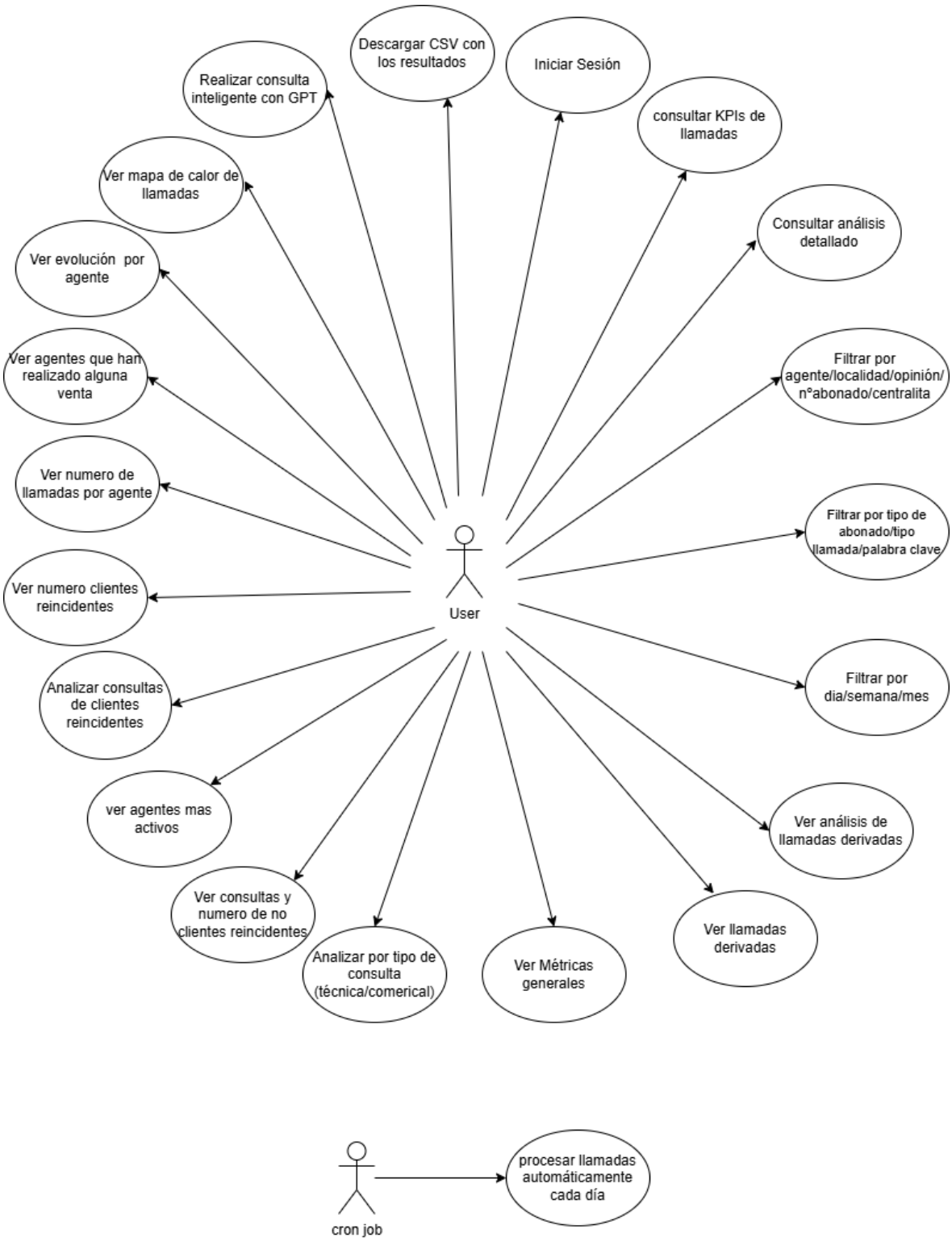


Ilustración 18:Diagrama de casos de uso

- **Iniciar sesión:** El usuario debe autenticarse mediante *login* para acceder al sistema, garantizando así la seguridad de los datos.
- **Consultar KPIs de llamadas:** Acceso a indicadores clave de rendimiento sobre el volumen y características de las llamadas.
- **Consultar análisis detallado:** Visualización en profundidad de las llamadas procesadas, incluyendo todos los campos de análisis.
- **Filtrar por agente, localidad, opinión, número de abonado, centralita, tipo llamada, palabra clave, tipo de abonado:** Filtros avanzados que permiten al usuario segmentar la información de manera personalizada.
- **Ver agentes que han realizado alguna venta:** Análisis específico para identificar agentes con registros de ventas en las llamadas.
- **Ver agentes más activos:** Ranking de agentes en función del número de llamadas gestionadas.
- **Ver clientes reincidentes:** Detección y análisis de abonados y no abonados que han contactado más de una vez.
- **Ver número de llamadas por agente:** Consulta de métricas de volumen por agente.
- **Filtrar por día/semana/mes:** Herramientas de filtrado temporal para analizar tendencias y evolución.
- **Ver evolución por agente:** Gráfico temporal que muestra la actividad de cada agente a lo largo del tiempo.
- **Ver mapa de calor de llamadas:** Visualización de la actividad de las llamadas por agente y fecha.
- **Realizar consulta inteligente con GPT:** Permite al usuario realizar preguntas en lenguaje natural y obtener respuestas contextualizadas basadas en los datos reales.
- **Descargar CSV con los resultados:** Exportación de los resultados filtrados en formato CSV para su posterior análisis o reporte.
- **Ver llamadas derivadas:** Acceso a un listado específico de las llamadas derivadas, su tipología y opinión posterior.
- **Análisis por tipo de consulta (técnica, comercial…):** Distribución de las llamadas según su naturaleza, con porcentajes de resolución por tipo.

- **Tiempo medio por llamada:** Métrica útil para valorar la eficiencia y complejidad de la atención.

Además, el sistema incorpora una **automatización diaria mediante cron job** que ejecuta todo el flujo de procesamiento de llamadas, asegurando que la información del *dashboard* esté actualizada cada día.

5.1.3 REQUISITOS DEL SISTEMA

Requisitos funcionales

Los requisitos funcionales son especificaciones detalladas que describen las capacidades y comportamientos que el sistema debe ofrecer para cumplir sus objetivos y satisfacer las necesidades de los usuarios.

En el contexto de este proyecto, cuyo objetivo es **automatizar el procesamiento, transcripción y análisis semántico de llamadas telefónicas**, así como proporcionar una herramienta visual de explotación de los resultados, estos requisitos aseguran que la solución sea **completa, fiable y alineada con las expectativas de los usuarios de negocio**.

Tabla de requisitos funcionales

Nº	***Requisito funcional***
RF1	El sistema debe ejecutar diariamente todo el flujo (descarga, descompresión, transcripción, análisis, almacenamiento).
RF2	Las grabaciones deben transcribirse automáticamente utilizando Vosk.
RF3	Las transcripciones deben analizarse con GPT-3.5 para extraer etiquetas, opiniones, localidades y palabras clave.

RF4	El sistema debe almacenar los resultados en PostgreSQL y generar un archivo JSON consolidado.
RF5	La aplicación web (*Streamlit*) debe permitir filtrar por agente, localidad, opinión, cliente, etc.
RF6	Los usuarios deben poder visualizar métricas clave (*KPIs*) y acceder al análisis detallado de cada llamada.
RF7	La interfaz debe permitir exportar los datos en formato CSV.
RF8	El sistema debe permitir consultas en lenguaje natural mediante **GPT-4** conectando con la base de datos.
RF9	La autenticación debe estar implementada mediante credenciales cifradas.

Tabla 6: Tabla de requisitos funcionales

Explicación de requisitos funcionales

Para asegurar la operatividad del sistema y la utilidad práctica para los usuarios finales, se definieron una serie de requisitos funcionales concretos:

- **Automatización completa**: El sistema debe obtener diariamente las grabaciones desde la API de Enreach y el servidor FTP, sin requerir intervención humana.
- **Transcripción automática**: Las grabaciones deben ser convertidas a texto mediante Vosk de forma totalmente automatizada.
- **Análisis semántico con IA**: Las transcripciones se analizan con GPT-3.5 para extraer información clave como etiquetas, opinión del cliente, localidad, etc.
- **Persistencia de datos**: Los resultados se almacenan en una base de datos PostgreSQL, permitiendo trazabilidad, consulta histórica y explotación avanzada.

- **Acceso seguro y autenticado**: La interfaz web (Streamlit) requiere autenticación mediante credenciales cifradas.
- **Panel interactivo y filtrado**: Los usuarios deben poder filtrar resultados por agente, localidad, opinión, palabras clave o número de abonado.
- **Visualización de KPIs**: El panel muestra métricas agregadas como duración media, opiniones, reincidencia o volumen de llamadas.
- **Exploración detallada**: Se puede acceder a cada llamada procesada con su análisis individual.
- **Exportación de resultados**: Es posible descargar los datos filtrados en formato CSV para informes o análisis externos.
- **Consulta en lenguaje natural**: A través de GPT-4, el sistema responde a preguntas formuladas en lenguaje natural, accediendo dinámicamente a la base de datos.

Requisitos no funcionales

Los requisitos no funcionales definen las condiciones de calidad que debe cumplir el sistema más allá de sus funcionalidades operativas. En el contexto de este proyecto, estos requisitos garantizan que el sistema sea eficiente, seguro, escalable y fácilmente mantenible, lo cual resulta esencial al tratar con grandes volúmenes de datos sensibles como llamadas telefónicas.

Tabla de requisitos no funcionales

Nº	***Requisito no funcional***
RNF1	*El procesamiento completo debe ejecutarse sin intervención manual (automatización diaria).*

RNF2	*El tiempo total de procesamiento no debe superar las 2 horas por jornada de llamadas.*
RNF3	*El acceso debe protegerse mediante contraseñas hasheadas.*
RNF4	*Los datos deben almacenarse de forma persistente y estructurada.*
RNF5	*La interfaz debe ser accesible vía navegador, sin instalación local.*
RNF6	*El sistema debe garantizar la trazabilidad mediante base de datos y archivos JSON.*
RNF7	*Los archivos temporales deben eliminarse automáticamente tras el procesamiento.*
RNF8	*El sistema debe ser modular y fácilmente mantenible.*
RNF9	*El sistema debe escalar correctamente ante incrementos de volumen sin pérdida de rendimiento.*

Tabla 7: Tabla de requisitos no funcionales

Explicación de requisitos no funcionales

- **Automatización completa**: Todo el flujo diario (descarga, transcripción, análisis e inserción en base de datos) se ejecuta automáticamente mediante tareas cron, sin requerir intervención manual.
- **Tiempo de procesamiento**: El sistema debe procesar todas las llamadas del día en menos de 2 horas, permitiendo disponer de resultados actualizados al comienzo de la jornada.

- **Seguridad de acceso**: Se garantiza el acceso seguro al panel mediante autenticación con contraseñas cifradas (hasheadas), cumpliendo con los principios del Esquema Nacional de Seguridad.
- **Persistencia de datos**: La información extraída se guarda de forma persistente en una base de datos relacional, permitiendo su consulta, trazabilidad y análisis histórico.
- **Accesibilidad web**: Gracias a Streamlit, los responsables pueden consultar los resultados desde cualquier navegador, sin necesidad de instalar software adicional.
- **Trazabilidad**: Cada llamada procesada queda registrada tanto en la base de datos como en archivos estructurados en formato JSON, garantizando el seguimiento y auditoría.
- **Optimización del almacenamiento**: Para reducir el uso de disco, el sistema elimina automáticamente carpetas intermedias y ficheros temporales una vez finalizado el procesamiento.
- **Mantenibilidad**: La modularidad del sistema permite realizar mejoras o ajustes en componentes específicos sin afectar al conjunto.
- **Escalabilidad**: El diseño soporta una carga creciente de llamadas diarias sin comprometer la velocidad ni la calidad del análisis.

5.2 PROCESAMIENTO Y BACKEND

5.2.1 ESTRUCTURA DE SCRIPTS

La siguiente figura muestra la estructura de carpetas y scripts que conforman el *backend* y los componentes de análisis del sistema.

Los scripts ubicados en */etc/cron.daily/* son invocados automáticamente por el cron del sistema, y constituyen el flujo diario automatizado de procesamiento.

Los scripts en */home/bea/scripts/* implementan los módulos de transcripción y análisis de llamadas.

La carpeta */home/bea/scripts/pages/* contiene las páginas de la aplicación de análisis interactiva desarrollada en Streamlit.

Los scripts auxiliares y pruebas se encuentran en */home/bea/scripts/tests/.* Además, se incluyen carpetas dedicadas a los ficheros de detalle obtenidos desde la API de Enreach *(/home/bea/ficherosdetalle/),* a las transcripciones generadas por el motor Vosk *(/home/bea/transcripciones/tparalelo/),* y a los resultados de análisis semántico con GPT *(/home/bea/analisis_gpt/)*

```
./etc/cron.daily/
├── descarga_audio.sh
├── download_ATC.sh
├── procesar_json.sh
├── unzip_zipped0.sh
├── search_call_agent.sh
├── lanzar_transcripcion.sh
├── analizar_todas_llamadas_gpt.py
└── delete_all.py


/home/bea/scripts/
├── analizar_todas_llamadas_gpt.py
├── transcribir_paralelo.py
├── users.json
```

```
├── agentes_elda.csv
├── agentes_novelda.csv
├── palabras_clave.xlsx
├── generarusers.py
├── pages/
└── app.py

/home/bea/scripts/pages/
├── KPIs.py
└── Analisis_detallado.py

/home/bea/scripts/tests/
├── exportardatosbd.py
├── analizar_llamada_gpt.py
├── contactlens-job.json
├── contact_lens_resultado.json
├── prueba_gemini.py
├── descarga_para_analizar_money.sh
└── asignar_agentes.py

/home/bea/ficherosdetalle/
├── {fecha}/
│   ├── output_fecha.txt
│   └── transcription_{fecha}.json

/home/bea/transcripciones/tparalelo/
```

```
└── {fecha}/


/home/bea/temp_wavs/
└── archivo.wav


/home/bea/analisis_gpt/
└── analisis_total.json
|   ├── output_transcripcion_localidad_fecha_hora_0_telefono.txt
|   ├── output_transcripcion_localidad_fecha_hora_1_telefono.txt
```

Tabla 8: Listado de la estructura de carpetas de la solución desarrollada

/etc/cron.daily/

- **descarga_audio.sh**: Automatiza la invocación a la API de Enreach para preparar el lote diario de grabaciones.
- **download_ATC.sh**: Descarga el fichero de detalle *transcription_YYYY-MM-DD.json* con los metadatos de las llamadas.
- **procesar_json.sh:** Procesa el JSON de metadatos para generar un fichero de texto *output_YYYY-MM-DD.txt* con información clave por llamada.
- **unzip_zipped0.sh**: Descomprime los archivos ZIP de grabaciones en formato MP3, dejándolos preparados para la transcripción.
- **search_call_agent.sh:** Copia las grabaciones asociadas a cada agente y genera una relación CSV entre llamadas y agentes/localidades.
- **lanzar_transcripcion.sh:** Lanza el proceso de transcripción automática con Vosk a través del script **transcribir_paralelo.py**.

- **analizar_todas_llamadas_gpt.py**: Ejecuta el análisis semántico de las transcripciones utilizando la API de GPT, generando tanto el **analisis_total.json** como la inserción en base de datos.

- **delete_all.py:** Elimina carpetas y ficheros intermedios para optimizar el uso de almacenamiento.

/home/bea/scripts/

- **analizar_todas_llamadas_gpt.py:** Script principal de análisis semántico de las llamadas.

- **transcribir_paralelo.py**: script encargado de convertir los audios WAV en texto mediante el motor Vosk, utilizando procesamiento paralelo para acelerar la transcripción..

- **users.json:** Fichero que almacena los usuarios y contraseñas hasheadas para la **autenticación** en el panel de análisis.

- **agentes_elda.csv** y **agentes_novelda.csv**: Listados generados de agentes por zona, utilizados para enriquecer el análisis de llamadas.

- **palabras_clave.xlsx**: Documento con las palabras clave de interés para el análisis semántico.

- **generarusers.py**: Script utilizado para generar hashes de contraseñas para users.json.

- **analizar_llamada_gpt.py:** Script auxiliar para análisis de una única llamada.

- **asignar_agentes.py**: Script utilizado en pruebas iniciales para añadir manualmente el campo agente a las llamadas a partir del output TXT.

- **analizar_excel.py**: Script que procesa usuarios_enreach.xlsx y genera automáticamente los CSV de agentes por zona.

- **usuarios_enreach.xlsx**: Excel que contiene los datos de todos los agentes (nombre, código, departamento, estado activo, etc.).
- **exportardatosbd.py**: Script de prueba utilizado para exportar los resultados de la base de datos a CSV.
- **app.py:** Script para lanzar desde streamlit con el inicio de sesión. A partir de este punto empieza el *frontEnd*.
- **descarga_para_analizar_money.sh:** Script para automatizar la solicitud de descargas de múltiples fechas consecutivas (utilizado para pruebas o recuperación de histórico).

/home/bea/scripts/pages/

- **KPIs.py:** Página del panel Streamlit que muestra indicadores clave de las llamadas (KPIs).
- **Analisis_detallado.py:** Página del panel Streamlit que permite el análisis detallado e interactivo de las llamadas.

/home/bea/scripts/tests/

- **contactlens-job.json:** JSON de configuración utilizado en pruebas con Amazon Contact Lens.
- **contact_lens_resultado.json:** Ejemplo de resultado de análisis Contact Lens.
- **prueba_gemini.py:** Script utilizado para comparar resultados de análisis con el modelo Gemini.

/home/bea/ficherosdetalle/

- **transcription_YYYY-MM-DD.json:** Fichero con el detalle de las llamadas proporcionado por la API de Enreach, incluyendo metadatos como número de origen, destino, duración, agente, etc.

- **output_YYYY-MM-DD.txt:** Fichero generado a partir del JSON con un formato simplificado que facilita el cruce de datos con las transcripciones.

/home/bea/transcripciones/tparalelo/

- Carpetas diarias que contienen las transcripciones de las llamadas procesadas con Vosk, organizadas por fecha.

/home/bea/analisis_gpt/

- **analisis_total.json:** Fichero consolidado con el resultado del análisis semántico de todas las llamadas de un día, generado por analizar_todas_llamadas_gpt.py.

/home/bea/temp_wavs/

- Carpeta temporal con los archivos WAV generados durante la transcripción, eliminados automáticamente tras su uso.

5.2.2 FLUJO DE PROCESAMIENTO DE AUDIO

5.2.2.1 Gestión de descargas

Automatiza la obtención de grabaciones desde la **API de Masvoz** y su depósito en un servidor **FTP** para garantizar trazabilidad y accesibilidad.

- **Creación de directorio local**: antes de invocar la API, el script crea de forma dinámica un directorio en el servidor local *(/home/beatriz/ftp/<fecha_previa>)* donde se guardarán los archivos descargados. Esto facilita la organización diaria y evita colisiones de nombres.

```
# 2. Creación de directorio local de trabajo
```

```
work_dir="/home/beatriz/ftp/${current_date}"
mkdir -p "$work_dir"
echo "Directorio local creado: $work_dir"
```

Código 1: Creación de directorio de trabajo

- **Cálculo de la fecha de consulta**: se determina el **día anterior** a la fecha de ejecución con

```
current_date=$(date -d "-1 day" +"%Y-%m-%d")
```

Código 2: Calculo fecha día anterior

de modo que siempre se procesen únicamente las llamadas del día completo inmediatamente anterior.

- **Invocación de la API de Masvoz**:
    - **Endpoint**: https://api.masvoz.es/bulkdownload/prepare
    - **Autenticación**: Basic Auth con credenciales codificadas.
    - **Payload**: incluye accountId (10131), callType (Incoming), rango startDate/endDate, ftpAddress remoto y notificationEmail.

```
response=$(curl --silent --location
'https://api.masvoz.es/bulkdownload/prepare' \
  -H 'Content-Type: application/json' \
  -H 'Authorization: Basic credenciales==' \
  --data-raw "$json_data")
```

Código 3: Invocación de la API de MasVoz

- **Transferencia y ubicación remota**: la respuesta de la API proporciona un bulkId y Masvoz deposita automáticamente los archivos comprimidos en la carpeta FTP correspondiente (ftp://bea:•••••••@ip/<fecha_previa>/).

```
 "ftpAddress": "ftp://user:password@ip_address/${current_date}/",
```

Código 4: Transferencia a FTP

- **Registro local de identificador**: el bulkId se guarda en bulkId.txt dentro del directorio de trabajo para su uso en la fase de descarga activa mediante un cron job separado.

```
bulkId=$(echo $response | jq -r '.bulkId')
echo "bulkId recibido: $bulkId"
echo $bulkId > "$work_dir/bulkId.txt"
```

Código 5: Registro de BulkId

Esto es el mensaje que llega cuando se ejecuta **descarga_audio.sh**

```
root@TFGBea:/etc/cron.daily# ./descarga_audio.sh
Current date: 2025-04-30
El valor de bulkId es: ea6e5d5e-5c41-4a21-960d-87be75fd113c
```

Tabla 9: respuesta al descargar un audio de un día

Tabla de variables clave

Variable	***Descripción***
current_date	Fecha del día anterior en formato YYYY-MM-DD.
work_dir	Ruta local donde se descargan y organizan los archivos.
startDate	Timestamp de inicio del rango (00:00:00 del día anterior).
endDate	Timestamp de fin del rango (23:59:59 del día anterior).
ftpAddress	URL FTP remota con carpeta datada y credenciales para la carga.
bulkId	Identificador único para el lote de descargas.

Tabla 10: Variables clave para la descarga

Flujo completo y control de proceso

El proceso de gestión de descargas constituye la fase preparatoria del flujo de trabajo. En esta etapa, no se realiza aún la descarga local de los archivos de audio, sino que se solicita su generación y se orquesta su colocación automática en el servidor **FTP** remoto.

El identificador *bulkId* recibido permite trazar y recuperar el lote correspondiente en una segunda fase del proceso, automatizada mediante una tarea cron independiente que, de forma periódica, lee el *bulkId* almacenado y transfiere los archivos comprimidos al directorio local de trabajo.

Notificaciones y control de incidencias

Como parte del flujo de proceso, el sistema de **Masvoz** envía automáticamente un correo electrónico de notificación al completar la preparación de la descarga. Dicho correo incluye el *bulkId* correspondiente y un resumen del estado del proceso (éxito o error).

En caso de incidencias durante la transferencia al servidor FTP, el correo proporciona información detallada sobre el error producido, lo que facilita el diagnóstico y la resolución de problemas. Ejemplos típicos de errores incluyen:

- *Can not put test file to directory:* error al intentar escribir en el directorio FTP, habitualmente por permisos insuficientes o rutas inexistentes.

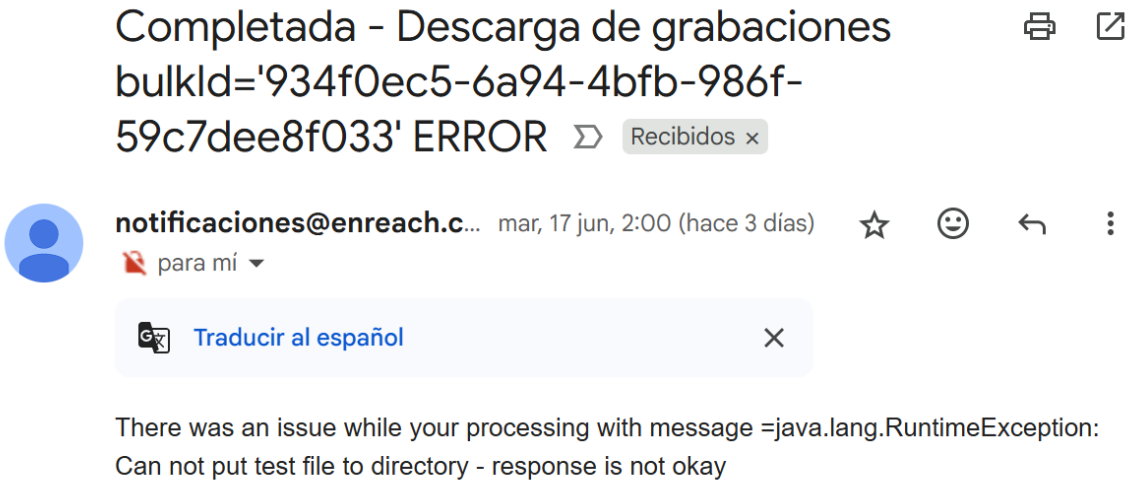


Ilustración 19:Error Descarga(1)

- CopyStreamException: *IOException* caught while copying: error en la transferencia de datos por problemas de red o de permisos.

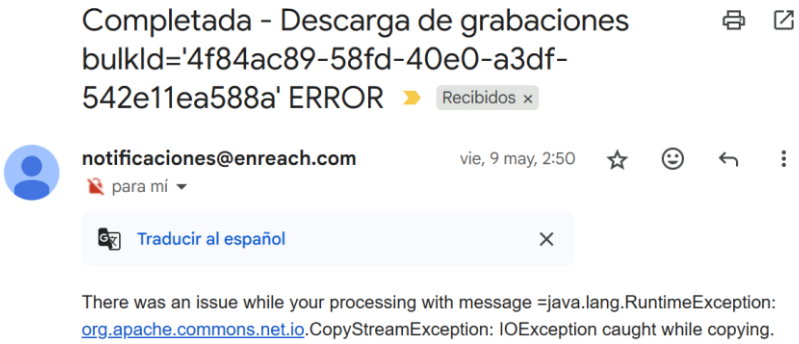


Ilustración 20:Error Descarga(2)

Estos errores suelen estar asociados a excepciones de tipo *IOException*, ya que se producen durante operaciones de entrada/salida en la transferencia de archivos. En particular, la clase *CopyStreamException* forma parte del paquete de excepciones lanzadas cuando hay fallos en la escritura o lectura desde el flujo de datos entre el sistema de **Masvoz** y el servidor FTP.

En entornos automatizados, este tipo de errores puede capturarse como *RuntimeException* si no se gestiona adecuadamente, lo que podría interrumpir la ejecución del cron job.

Cuando la operación es exitosa, el correo confirma la descarga con el número de archivos y el tamaño total transferido.

Ilustración 21:Confirmación Descarga

A continuación, se incluyen ejemplos reales de notificaciones recibidas durante el desarrollo y validación del sistema:

Control de errores y trazabilidad

El script verifica la recepción correcta del *bulkId* y registra todos los eventos relevantes de la ejecución en un fichero de log, facilitando la auditoría y la trazabilidad. Asimismo, la estructura de carpetas organizada por fecha permite una localización rápida y precisa de los archivos correspondientes a cada jornada de trabajo.

Conclusión

Este mecanismo automatizado garantiza que, de forma robusta y controlada, cada jornada queden preparados en el sistema tanto los identificadores de lote como la estructura de carpetas necesarias para la posterior descarga activa y el procesamiento integral de las llamadas.

5.2.2.2 Procesamiento local de las descargas

Una vez que la API de **Masvoz** ha preparado el lote de grabaciones y ha depositado los archivos en el servidor **FTP** remoto, el sistema implementa un proceso automatizado de descarga activa que transfiere diariamente dichos archivos al servidor local para su posterior procesamiento.

Automatización mediante cron job

Para garantizar la descarga puntual y sin intervención manual, se ha configurado una tarea automática (cron job) que se ejecuta diariamente a las 2:00. Esta tarea:

- Lee el identificador *bulkId* previamente almacenado en el directorio de trabajo.
- Se conecta al servidor FTP remoto utilizando las credenciales configuradas.
- Descarga el contenido correspondiente al *bulkId* en la carpeta local creada previamente.
- Registra en un log el resultado de la operación (éxito o error), así como los archivos descargados.

El cron job se ha diseñado para ser robusto frente a errores de red o de conectividad, con reintentos automáticos en caso de fallo.

Estructura de carpetas locales

Los archivos descargados se organizan localmente siguiendo una estructura de carpetas por fecha*: /home/bea/YYYY-MM-DD/*

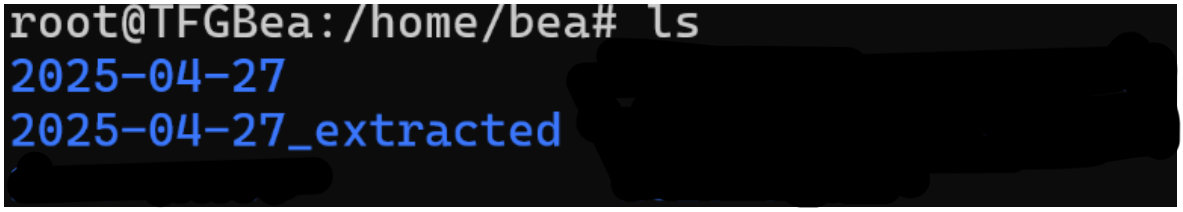


Ilustración 22: Carpetas descargadas en el directorio

Esta organización facilita el seguimiento temporal de las llamadas y su procesamiento secuencial en las etapas posteriores del sistema.

Control de logs y errores

El sistema registra en un log dedicado:

- La fecha y hora de cada intento de descarga.
- El resultado de la conexión al **FTP**.
- La lista de archivos efectivamente descargados.
- Los posibles errores detectados (conectividad, permisos, archivos faltantes, etc.).

Este registro permite auditar el proceso y simplifica la detección de anomalías en las descargas.

Verificación de integridad

Aunque el sistema de **Masvoz** no proporciona actualmente *checksums* automáticos, se implementa una verificación básica de integridad local mediante:

- Comprobación del número de archivos recibidos vs. notificados en el correo de confirmación.
- Comparación del tamaño total recibido con el tamaño informado.
- Verificación de la descompresión correcta de los ficheros ZIP descargados.

Estas comprobaciones permiten detectar rápidamente problemas de integridad o descargas incompletas antes de iniciar las fases de conversión y transcripción.

Descompresión de los ficheros descargados

El proceso de descompresión se realiza mediante un script específico (**unzip_zipped0.sh**), que recorre los archivos ZIP descargados, los descomprime en una carpeta estructurada por fecha, y deja los audios en formato MP3 preparados para su conversión y análisis posterior.

```
# Buscar todos los archivos .zip en el directorio y subdirectorios
find "$search_dir" -type f -name '*.zip' | while read -r zip_file; do
    echo "Descomprimiendo $zip_file"

    # Crear un directorio para los archivos descomprimidos
    output_dir="${output_base_dir}/$(basename "${zip_file%.zip}")"
    mkdir -p "$output_dir"

    # Descomprimir el archivo sin confirmar el reemplazo de archivos
    unzip -o "$zip_file" -d "$output_dir"
done
```

Código 6: Descompresión de los ficheros para poder analizarlos

Este paso garantiza que los audios estén disponibles de forma inmediata para la siguiente etapa del sistema.

La operación de descompresión queda registrada en el log de proceso, incluyendo:

- el nombre del fichero ZIP procesado,
- el número de archivos extraídos,
- posibles incidencias detectadas durante la operación.

La estructura resultante tras la descompresión sigue el siguiente esquema:

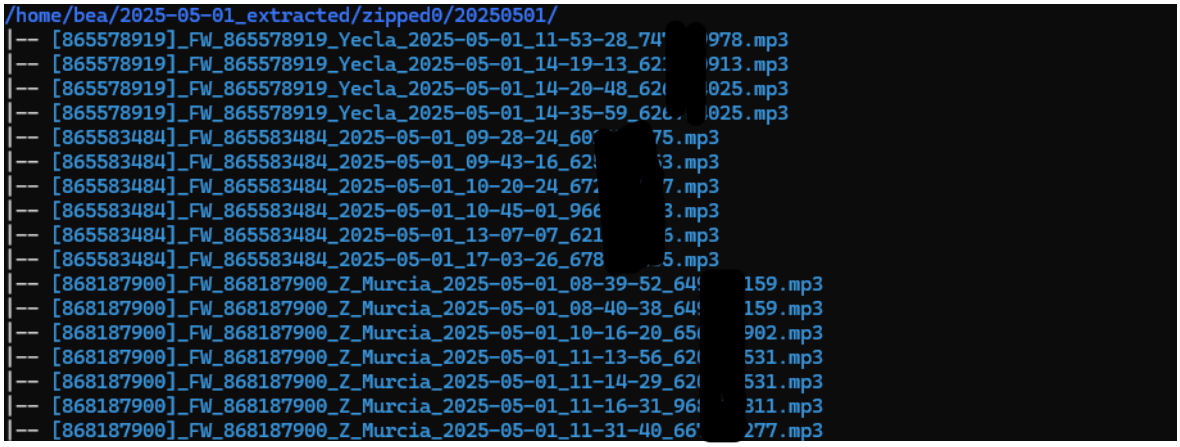


Ilustración 23:Ejemplo de estructura de carpeta tras la descompresión de audios.

5.2.2.3 Conversión de audios MP3 a WAV

Las grabaciones descargadas desde la plataforma de **Masvoz** se reciben en formato comprimido MP3. Sin embargo, para que el sistema de transcripción automática pueda procesarlas correctamente, es necesario convertirlas previamente al formato PCM WAV, que es el requerido por el motor de reconocimiento de voz utilizado (Vosk).

```
    print(f"[START] {filename}")
    subprocess.run([
        "ffmpeg", "-y", "-i", mp3_path,
        "-ar", "16000", "-ac", "1", "-acodec", "pcm_s16le",
        temp_wav
    ], stdout=subprocess.DEVNULL, stderr=subprocess.DEVNULL)
```

Código 7: Conversión audios MP3 a WAVs

Esta conversión se realiza de forma automática dentro del propio proceso de transcripción, implementado en el script **transcribir_paralelo.py**. Durante la ejecución del script, cada archivo MP3 se convierte temporalmente a WAV utilizando la herramienta ffmpeg, con los siguientes parámetros de conversión:

- Frecuencia de muestreo: **16 kHz**
- Canal de audio: **mono (1 canal)**

- Formato de codificación: **PCM firmado de 16 bits (pcm_s16le)**

Los archivos WAV generados se almacenan temporalmente en el directorio `/home/bea/temp_wavs/`, creado dinámicamente durante la ejecución. Una vez completada la transcripción de cada audio, el correspondiente archivo WAV temporal es eliminado, de modo que el sistema no acumula ficheros intermedios y optimiza el uso del espacio en disco.

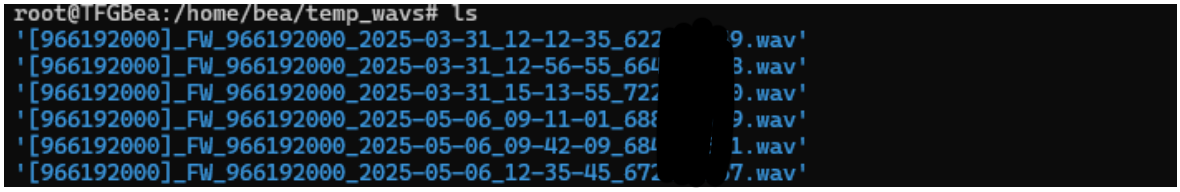


Ilustración 24:Carpeta temporal utilizada para la conversión automática de audios

Este enfoque garantiza una conversión eficaz y controlada, plenamente integrada en el flujo automatizado del sistema, sin necesidad de procesos manuales adicionales.

5.2.3 LLAMADAS A VOSK Y TRANSCRIPCIONES

La transcripción automática de las grabaciones constituye uno de los componentes clave del sistema desarrollado. Para ello se ha optado por utilizar el motor de reconocimiento de voz **Vosk**, una solución de código abierto que permite realizar el proceso de transcripción de forma local, sin necesidad de conexión a servicios externos ni costes adicionales por volumen de uso.

El sistema emplea el modelo preentrenado en español **`vosk-model-es-0.42`**, que ofrece un equilibrio adecuado entre velocidad y precisión para el dominio de llamadas telefónicas.

5.2.3.1 Proceso de transcripción

La transcripción se implementa en el script **transcribir_paralelo.py,** que integra las siguientes etapas:

1. Conversión automática del archivo **MP3** a formato WAV PCM.

2. Carga del modelo Vosk correspondiente.

```
model = vosk.Model(model_path)
```

Código 8: carga modelo Vosk

3. Procesamiento del archivo WAV mediante el reconocedor de Vosk (*KaldiRecognizer*), que opera a nivel de flujo de audio (*streaming*).

```
wf = wave.open(temp_wav, "rb")
rec = vosk.KaldiRecognizer(model, wf.getframerate())
```

Código 9: carga KaldiRecognizer

4. Construcción progresiva del texto transcrito a partir de los resultados parciales y finales obtenidos.

```
transcription = ""
while True:
    data = wf.readframes(4000)
    if len(data) == 0:
        break
    if rec.AcceptWaveform(data):
        transcription += json.loads(rec.Result()).get("text", "") + "\n"

transcription += json.loads(rec.FinalResult()).get("text", "")
```

Código 10: Construcción de texto

5. Guardar el resultado en txt

```
with open(output_path, "w") as f:
    f.write(transcription)
```

Código 11: Guardado de transcripción en un txt

El proceso se ejecuta en paralelo aprovechando las capacidades multicore del servidor, con un número de procesos configurable (hasta 8 procesos en paralelo). Esto permite reducir significativamente el tiempo total de transcripción de grandes volúmenes de grabaciones.

5.2.3.2 Paralelización del proceso

Dado el volumen de grabaciones a procesar diariamente, se implementó un enfoque de transcripción paralelizada que permite reducir considerablemente los tiempos de ejecución.

El script **transcribir_paralelo.py** aprovecha la librería estándar de Python *multiprocessing,* mediante el uso de un **pool de procesos concurrentes**. De este modo, el sistema distribuye la carga de transcripción de los diferentes archivos de audio entre varios núcleos de CPU de forma simultánea.

El número de procesos se adapta dinámicamente a la capacidad del servidor (hasta un máximo de 8 procesos en la configuración actual):

```
    print(f"[DONE] {filename}")

if __name__ == "__main__":
    files = os.listdir(input_folder)
    num_procesos = min(cpu_count(), 8)
    print(f"Procesando {len(files)} archivos con {num_procesos} procesos...")
```

Código 12: Proceso de paralelización con 8 procesos

Este enfoque permite procesar múltiples llamadas en paralelo, logrando así una transcripción diaria escalable y adecuada a las necesidades del sistema.

Ejecutando el proceso en segundo plano

Para el procesamiento de grabaciones correspondientes a días pasados o volúmenes especialmente altos, el sistema permite ejecutar la transcripción en segundo plano mediante el uso de *nohup*. De este modo, es posible lanzar el proceso de transcripción de larga duración sin que se interrumpa en caso de cierre de sesión.

En pruebas realizadas, la transcripción en serie (sin paralelización) requería entre **6 y 7 horas** para un día completo de grabaciones. Con la optimización actual basada en transcripción paralela, el tiempo de procesamiento se ha reducido a aproximadamente **2 horas** para el mismo volumen de datos, lo que permite garantizar la disponibilidad diaria de los resultados.

Parámetro	***Valor / Configuración***
Motor de transcripción	Vosk
Modelo empleado	vosk-model-es-0.42
Frecuencia de muestreo (WAV)	16 kHz
Canales	Mono (1 canal)
Codificación WAV	PCM firmado de 16 bits (pcm_s16le)
Procesamiento paralelo	Sí (multiprocessing Pool)
Núm. máximo de procesos en paralelo	8
Tiempo medio en serie (día completo)	6-7 horas
Tiempo medio en paralelo (día completo)	~2 horas
Ejecución en segundo plano	Sí (nohup)

Tabla 11: Características de Vosk y rendimiento en este proyecto

5.2.4 FICHEROS DETALLE Y POST-PROCESADO JSON

El flujo completo del sistema requiere combinar de forma estructurada la información proveniente de la plataforma de **Masvoz** (metadatos de las llamadas) con las transcripciones de audio obtenidas mediante **Vosk**. Esta integración es esencial para enriquecer el análisis

final de cada llamada, permitiendo asociar a cada registro campos como el número de abonado, número marcado, duración o servicio asociado, que no están presentes en la transcripción pura.

Este proceso se apoya en la generación de **ficheros de detalle intermedios** en formato JSON y TXT, que permiten alimentar posteriormente el análisis con **GPT** de forma estructurada.

5.2.4.1 Descarga de metadatos: download_ATC.sh

El primer paso de esta fase consiste en ejecutar el script **download_ATC.sh**, que lanza una llamada adicional a la API de Masvoz para obtener un fichero JSON detallado de metadatos asociado a las llamadas del día procesado.

El endpoint utilizado devuelve un fichero de la forma:

```
/home/bea/ficherosdetalle/YYYY-MM-DD/transcription_YYYY-MM-DD.json
```

Este JSON contiene información clave sobre cada llamada, incluyendo:

- Número de abonado (si disponible)
- Número de destino marcado
- Número de origen
- Fecha y hora de la llamada
- Duración en segundos
- Notas o etiquetas asociadas (campo note)
- Otros metadatos relevantes.

*5.2.4.2 **Procesamiento del fichero de detalle: procesar_json.sh***

A continuación, el script **procesar_json.sh** automatiza la extracción de los campos más relevantes del JSON de Masvoz. Para ello, recorre las entradas del fichero y genera un fichero intermedio en formato TXT más sencillo de parsear:

```
/home/bea/ficherosdetalle/YYYY-MM-DD/output_YYYY-MM-DD.txt
```

Esto es un ejemplo de o que encontraríamos en una línea de este fichero

```
[<número_abonado>]_<número_destino>_<fecha>_<hora>_<número_origen>  |  DEPARTAMENTO:  <nombre_agente>  |  DURATION: <duración_segundos>
```

```
[101743]_[966192000]_FW_966192000_2025-05-07_14-55-52_686XX5X50 | DEPARTAMENTO: AG51_Cesar_X| DURATION: 256

[]_[966192000]_FW_966192000_2025-05-07_14-57-07_665XX653  | DEPARTAMENTO: AG49_Byron_Y | DURATION: 135
```

Este fichero simplificado sirve como referencia cruzada que permite:

- asociar fácilmente cada archivo de audio a su llamada correspondiente
- identificar el agente que gestionó la llamada
- recuperar el número de abonado (cuando la llamada sea de un abonado) cuando no es un abonado, el campo estará vacío, como podemos ver en el archivo de AG49_Byron_Y

- Obtener el número de teléfono del cliente para poder hacer seguimiento de incidencia a posteriori en caso que no sean abonados.
- obtener la duración de la llamada en segundos
- permitir que esta información enriquezca el análisis posterior

Integración con el análisis GPT: analizar_todas_llamadas_gpt.py

En la última etapa del flujo, el script **analizar_todas_llamadas_gpt.py** consume tanto:

- las transcripciones de audio en formato .txt generadas en la fase de transcripción
- como el fichero *output_YYYY-MM-DD.txt* generado en esta etapa.

Durante el procesamiento, cada entrada de transcripción se enriquece con la información estructurada extraída del TXT:

- agente asociado
- número de abonado
- número marcado
- duración
- hora exacta de la llamada
- día de llamada
- teléfono de origen.

El resultado consolidado de esta integración se almacena en:

```
/home/bea/analisis_gpt/analisis_total.json
```

Este fichero JSON es el que finalmente se utiliza tanto para alimentar la base de datos PostgreSQL como para proporcionar datos al panel interactivo de análisis en Streamlit.

```
+-----------------------------------------------------------+
|
| API Masvoz (metadatos JSON)
|     ↓
| download_ATC.sh       →   transcription_YYYY-MM-DD.json
|     ↓
| procesar_json.sh      →   output_YYYY-MM-DD.txt
|     ↓
| Transcripción Vosk    →   transcripciones TXT
|     ↓
| analizar_todas_llamadas_gpt.py
|     ↓
| analisis_total.json   +   inserción en PostgreSQL (BD)
|
+-----------------------------------------------------------+
```

Tabla 12: flujo de proceso

Conclusión

Este mecanismo de ficheros de detalle y post-procesado JSON permite garantizar que el análisis de cada llamada sea completo y contextualizado, integrando tanto la transcripción de la conversación como los metadatos operativos de la llamada. De este modo, se consigue una explotación de datos más rica y precisa en las etapas posteriores de análisis y visualización.

5.2.5 ANÁLISIS DE TRANSCRIPCIONES Y EXTRACCIÓN AUTOMÁTICA DE INFORMACIÓN

El sistema desarrollado incorpora un módulo avanzado de análisis lingüístico de las transcripciones obtenidas, cuyo objetivo es estructurar la información contenida en cada llamada y facilitar su posterior explotación.

A partir de los ficheros .txt generados en la fase de transcripción, se ejecuta un proceso automatizado de análisis semántico mediante la integración de la API de **OpenAI (GPT-3.5-turbo)**.

5.2.5.1 Proceso de análisis

El análisis se implementa en el script **analizar_todas_llamadas_gpt.py**, que recorre secuencialmente todas las transcripciones de una fecha determinada. Para cada transcripción, el sistema:

1. Elimina frases de introducción estándar que no aportan contenido relevante.

```
frases_locucion = [
    "le damos la bienvenida a cableworld",
    "no has dicho ninguna poblaci  n. te transfiero",
    "por motivos de calidad y por su seguridad, esta llamada puede grabarse",
    "gracias por llamar a cablewell",
    "marque cero y le devolveremos la llamada",
    "para aver  as urgentes que no pueden esperar, marque uno",
    "nuestro horario de atenci  n telef  nica en festivos es de las 10 a las 14",
]
```

Código 13: Frases locución a borrar para que no cuente como transcripción a analizar

```
def limpiar_intro(texto):
lineas = texto.strip().splitlines()

# Buscar la primera línea que NO contenga frases de la locución
start_index = 0
for i, linea in enumerate(lineas):
linea_lower = linea.lower()
if not any(f in linea_lower for f in frases_locucion):
```

```
start_index = i
break

# Devolver el texto desde esa línea
return "\n".join(lineas[start_index:])
```

Código 14: función para limpiar intro

2. Construye un *prompt* específico que guía a GPT en la identificación de información clave.

   El *prompt* utilizado está diseñado para guiar al modelo GPT en la extracción de información clave, devolviendo un resultado estructurado en formato JSON. Además, se establecen reglas explícitas para validar campos como la localidad (limitada a un listado predefinido), el resultado de la llamada, y la clasificación de etiquetas y tipo de consulta. A continuación, se muestra un fragmento representativo del *prompt*:

```
prompt = f"""
Eres un sistema que analiza llamadas de un Call center de
telecomunicaciones.

Analiza la siguiente transcripción y responde exactamente en formato
JSON.

{
  "resumen": "Resumen breve de 2 a 4 líneas explicando el objetivo y
desarrollo de la llamada",
  "palabras_clave": ["keyword1", "keyword2", "keyword3"],
  "etiquetas": ["comercial", "técnica", "portabilidad", "router",
"abonado", "consulta", "incidencia"],
  "tipo_llamada": "comercial" o "técnica",
  "tipo_incidencia": "router", "móvil", "cobertura", "facturación",
// puede estar vacío si no aplica
  "respuesta_agente": "Breve descripción crítica sobre cómo actuó el
agente",
  "resultado": "resuelta" | "derivada" | "sin solución" |
"pospuesta",
  "opinion_cliente": "positiva" | "negativa" | "neutral",
  "motivo_opinion": "Explicación del por qué de esa opinión del
cliente",
```

```
  "localidad": "Nombre exacto de la localidad si está en la lista, o
'no identificada'",
  "venta_realizada": true | false | "" // true si se ha confirmado la
venta, false si se ha intentado pero no logrado, "" si no se menciona
}


Transcripción:
\"\"\"{{contenido}}\"\"\"
"""
```

Código 15:Fragmento representativo del prompt

3. Envía la transcripción limpia a la API de GPT.

```
response = client.chat.completions.create(
    model="gpt-3.5-turbo",
    messages=[{"role": "user", "content": prompt}],
    temperature=0.4
)
return response.choices[0].message.content
```

Código 16: LLamada a la api de GPT

4. Recoge la respuesta de GPT en formato estructurado (JSON).

```
respuesta = analizar_con_gpt(contenido_limpio)

try:
    datos = json.loads(respuesta)
except Exception as e:
    print(f"[ERROR] JSON inválido: {e}")
    continue
```

Código 17: Respuesta de GPT en formato JSON

5. Almacena el resultado en un archivo consolidado **analisis_total.json** y, simultáneamente, inserta los datos correspondientes en la base de datos **PostgreSQL**.

   5.1 Guardado en archivo **analisis_total.json**

```
analisis_total.append(datos)
# (el guardado en el archivo completo se realiza al final del script)
```

Código 18: Guardado en analisis_total.json

Este archivo permite mantener una copia consolidada de los resultados diarios y sirve como respaldo trazable.

5.2 Inserción en **PostgreSQL**

Simultáneamente, los campos generados son insertados en la tabla "analisis_llamadas" de PostgreSQL:

```
cursor.execute(insert_query, (
datos["resumen"],
datos["palabras_clave"],
json.dumps(datos["palabras_clave_categorizadas"],
ensure_ascii=False),
datos["etiquetas"],
...
datos["duracion_segundos"]
))
conn.commit()
```

Código 19: Inserción en PostgreSQL

Este almacenamiento estructurado en base de datos permite consultas eficientes y visualización posterior a través del *dashboard* de análisis.

5.2.5.2 Diseño del prompt

El *prompt* utilizado en el proceso de análisis ha sido cuidadosamente diseñado para garantizar que el modelo GPT responda de forma coherente y estructurada, en un formato JSON predefinido que facilite su tratamiento automático. Este enfoque asegura que los datos generados puedan ser insertados directamente en la base de datos y explotados de forma uniforme.

El diseño del *prompt* contempla los siguientes elementos clave:

- **Control de localidad:** Se proporciona un listado exhaustivo de localidades válidas. Si no se menciona ninguna localidad incluida en el listado, se especifica "localidad": "no identificada". La IA debe detectar correctamente la localidad si está mencionada en la transcripción. En caso de que se cite una localidad fuera de la lista (por ejemplo, "Barcelona") y no se identifique ninguna otra localidad válida en la conversación, el campo se establecerá como "no identificada". Además, si no se menciona ninguna localidad en la conversación, se prioriza la información contenida en el nombre del archivo de audio (que puede incluir la localidad). Si tampoco se identifica en este último, el campo se marca igualmente como "no identificada".

- **Control de categorías y formato estructurado**: Se definen explícitamente los campos que deben incluirse en la respuesta JSON: resumen, palabras clave, etiquetas, tipo de llamada, tipo de incidencia, resultado, opinión del cliente, motivo de la opinión, localidad y si hubo o no una venta. Esto garantiza que el modelo devuelva respuestas uniformes, analizables automáticamente, y evita la invención de estructuras o valores fuera de lo esperado.

Resultado de la consulta:

Se definen explícitamente las categorías válidas para el campo "resultado": "resuelta", "derivada", "pospuesta", "sin solución", u otros casos controlados.

- Se considera **"pospuesta"** cuando la IA detecta en la conversación que queda alguna acción pendiente, como el envío de un técnico, la necesidad de realizar una nueva llamada o una gestión que queda pendiente ("te llamamos", "vamos a comprobar y te llamamos", etc.).
- Se considera **"derivada"** cuando la llamada es transferida a otro departamento. Esto suele ocurrir, por ejemplo, cuando el número al que se llama es un número general y se deriva a técnico o comercial en función del motivo, o en el caso de llamadas para gestionar bajas, que se derivan al departamento de Calidad.

Además de la clasificación inicial efectuada por el modelo GPT, el sistema incorpora un **mecanismo automático de corrección semántica del resultado** basado en la detección de frases clave en la transcripción:

- Si el texto incluye expresiones como *"ya funciona"*, *"lo arreglaste"*, *"está solucionado"*, el sistema fuerza el resultado como **"resuelta"**.
- Si contiene frases como *"no va"*, *"sigue sin funcionar"*, *"me voy"*, *"cambio de compañía"*, se clasifica como **"sin solución"**.

Este doble enfoque, combinando inteligencia artificial con reglas semánticas explícitas, **aumenta la robustez del análisis y reduce errores** en la categorización de llamadas.

```
frases_resuelta = ["ya funciona", "está solucionado", "ha vuelto", "lo
arreglaste"]
frases_no_resuelta = ["sigue sin funcionar", "no va", "me voy", "cambio
de compañía"]
frases_pospuesta = ["te llamamos", "mañana te llamo", "nos pondremos en
contacto"]

for frase in frases_resuelta:
    if frase in texto_completo:
        datos["resultado"] = "resuelta"

for frase in frases_no_resuelta:
    if frase in texto_completo and datos.get("resultado") != "resuelta":
        datos["resultado"] = "sin solución"

for frase in frases_pospuesta:
    if frase in texto_completo:
        datos["resultado"] = "pospuesta"
```

Código 20: Validación del campo resultado

5.2.5.3 Estructura de la respuesta

Se define una estructura estándar en formato JSON que incluye los siguientes campos:

Campo	***Descripción***

resumen	Resumen breve (2-4 líneas) explicando el objetivo y desarrollo de la llamada.
palabras_clave	Lista de palabras clave relevantes mencionadas en la conversación.
etiquetas	Lista de etiquetas generales aplicables (comercial, técnica, portabilidad, etc.).
tipo_llamada	"comercial" o "técnica".
tipo_incidencia	Subtipo de incidencia si aplica (router, móvil, cobertura, facturación, etc.).
respuesta_agente	Breve descripción crítica de cómo actuó el agente.
resultado	"resuelta", "derivada", "sin solución", "pospuesta".
opinion_cliente	"positiva", "negativa", "neutral".
motivo_opinion	Justificación de la opinión expresada por el cliente.
localidad	Nombre exacto de la localidad identificada, o "no identificada".
venta_realizada	true, false o vacío si no se menciona.

5.2.5.4 Control de invención

Se instruye al modelo para que no genere localidades inventadas ni categorías fuera de las definidas.

En este control se establece la siguiente prioridad:

1) Dar preferencia a la localidad que se mencione en la **locución de la llamada**.

2) Si no se menciona localidad en la conversación, se recurre a la información contenida en el **nombre del archivo** (por ejemplo: _Yecla_...).
3) Si no se identifica localidad en ninguno de los dos elementos anteriores, se establece "localidad": "no identificada".
4) Los archivos de Murcia había una Z, en vez del nombre entonces se hizo una extracción a parte.

```
if "_z_" in datos.get("archivo", "").lower():
    datos["localidad"] = "murcia"
else:
    try:
        partes_archivo = datos["archivo"].replace(".txt", "").split("_")
        for i in range(len(partes_archivo)):
            if partes_archivo[i].lower() in ["fw", "ca", "caw", "z"]:
                localidad_extraida = partes_archivo[i + 2].lower()
                if localidad_extraida not in exclusiones and not
localidad_extraida.isdigit():
                    datos["localidad"] = localidad_extraida
    except Exception as e:
        print(f"[WARN] Falló la extracción de localidad desde el
archivo: {e}")
```

Código 21: Validación de localidad por nombre de archivo

5.2.5.5 Consistencia con la base de datos

La estructura JSON generada ha sido diseñada para ser mapeada directamente a la estructura de la base de datos PostgreSQL implementada.

De este modo, se garantiza que cada campo pueda insertarse de forma homogénea y que los datos extraídos sean explotables posteriormente en los cuadros de mando y en los análisis agregados.

```
cursor.execute(insert_query, (
    datos["resumen"],
    datos["palabras_clave"],
    json.dumps(datos["palabras_clave_categorizadas"], ensure_ascii=False),
    datos["etiquetas"],
    datos["tipo_llamada"],
```

```
    ...
    datos["duracion_segundos"]
))
```

Código 22: mapeo a base de datos

5.2.5.6 Control de errores y fallos de respuesta de GPT

Dado que el análisis semántico se basa en la interacción con un modelo de lenguaje a través de la API de **OpenAI**, es fundamental implementar mecanismos de control ante posibles fallos de conexión, tiempos de espera o respuestas mal formateadas. Para ello, el script **analizar_todas_llamadas_gpt.py** encapsula la llamada a la API en un bloque *try/except* que:

- Detecta y registra cualquier excepción producida al invocar el modelo.
- Captura errores de estructura si la respuesta de GPT no se puede interpretar como un objeto JSON válido.
- Retorna un objeto vacío con la estructura esperada para evitar que el proceso falle.

```
try:
    response = client.chat.completions.create(
        model="gpt-3.5-turbo",
        messages=[{"role": "user", "content": prompt}],
        temperature=0.4
    )
    return response.choices[0].message.content
except Exception as e:
    print(f"[ERROR] Error al llamar a GPT: {e}")
    return '{"resumen": "", "palabras_clave": [], "etiquetas": [],
"tipo_llamada": "", "tipo_incidencia": "", "respuesta_agente": "",
"resultado": "", "opinion_cliente": "", "motivo_opinion": "",
"localidad": "", "venta_realizada": ""}'
```

Código 23: Control de errores y fallos

5.2.5.7 Prevención de duplicados

Para evitar procesar transcripciones que ya han sido analizadas en ejecuciones anteriores, el script implementa un mecanismo de filtrado previo a partir de los registros almacenados en la base de datos. Antes de iniciar el análisis de cada archivo .txt, se consulta la tabla "analisis_llamadas" para obtener la lista de nombres de archivos ya insertados:

```
cursor.execute("SELECT archivo FROM analisis_llamadas;")
archivos_ya_insertados = set(row[0] for row in cursor.fetchall())
```

Código 24:Consulta para obtener lista de nombres

A continuación, durante la iteración de ficheros, el script ignora automáticamente aquellos que ya existen en la base de datos:

```
if archivo in archivos_ya_insertados:
    print(f"[SKIP] {archivo} ya está insertado.")
    continue
```

Código 25: Evitar duplicados

5.2.6 PRUEBAS UNITARIAS Y BENCHMARKING

Con el objetivo de garantizar la calidad y fiabilidad del sistema desarrollado, se han realizado diferentes pruebas unitarias y comparativas. Estas pruebas abarcan tanto la validación de los componentes individuales como el análisis de rendimiento y precisión de los motores de transcripción y análisis semántico.

Adicionalmente, se ha evaluado el impacto de la calidad de las grabaciones sobre el procesamiento, comparando resultados obtenidos a partir de grabaciones de la plataforma Enreach con grabaciones de la centralita corporativa antigua (grabaciones de alta fidelidad).

5.2.6.1 Comparativa de motores de transcripción

Como parte de la validación del sistema, se realizaron pruebas exhaustivas con distintos motores de transcripción automática, incluyendo soluciones de código abierto y motores comerciales de los principales proveedores **cloud**:

- Google Cloud Speech-to-Text
- Amazon Transcribe
- Azure Speech-to-Text
- Whisper (OpenAI WhisperX)
- Vosk (motor de producción)
- GPT-3.5-turbo (análisis semántico post-transcripción)
- Gemini AI (pruebas de *diarización* complementaria sobre transcripciones de Google) [36]

5.2.6.2 Metodología de prueba

Las pruebas se han basado en audios reales de clientes, con el objetivo de evaluar:

Las pruebas se efectuaron utilizando audios reales de llamadas, tanto procedentes de **Enreach** como de la centralita antigua.

El objetivo fue evaluar:

- Precisión de la transcripción (palabras, números, nombres)
- Fidelidad temporal
- Robustez frente a degradaciones de audio
- Capacidad de *diarización* (cuando aplica)

5.2.6.3 Resultados

Las pruebas se efectuaron tanto con grabaciones recientes obtenidas de la plataforma Enreach, como con grabaciones históricas de la centralita corporativa anterior. Estas últimas,

al contar con una calidad de audio superior, permitieron evaluar el comportamiento de los motores en condiciones ideales y contrastar los efectos de la degradación en las grabaciones actuales. Todas estas pruebas están más detalladas en el Anexo II.

Motor	***Precisión transcripción***	***Diarización hablantes***	***Robustez en baja calidad***
Google Cloud Speech-to-Text	Muy alta	Limitada	Media
Amazon Transcribe	Alta	Buena	Media
Azure Speech-to-Text	Baja (omisión de fragmentos)	Variable	Baja
Whisper (WhisperX)	Media-alta	Media (errores en cambio de speaker)	Alta
Vosk (producción)	Media-alta	No integrada	Muy alta (local y robusto)

Tabla 13: Resultados de probar las IAs

Comparativa entre complejidad de integración de motores cloud vs local

Motor	***Precisión***	***Integración***	***Dependencias***	***Coste***
Vosk	Media-alta	Muy fácil	Ninguna (local)	0 €
Google Cloud	Alta	Alta complejidad (bucket + *polling*)	GCP	Alto

Amazon Transcribe	Alta	Alta complejidad (S3+ jobs)	AWS	Alto
WhisperX	Alta	Media	Local + Python deps	0 €

Tabla 14:Comparativa entre complejidad de integración de motores coud vs local

Comparativa de análisis semántico (Vosk + GPT)

Dado que el sistema desarrollado integra un módulo de análisis semántico mediante la API de **GPT**, se evaluó también la calidad y consistencia del pipeline completo:

1. Transcripción → **Vosk**
2. Post-procesado + análisis semántico → **GPT-3.5-turbo**

Se realizaron pruebas comparando:

- Resultados de **Vosk** puro → texto plano de baja riqueza semántica.
- Resultados de **Vosk** + **GPT** → transcripción limpia + etiquetas + clasificación de intención + extracción de palabras clave.

Comparativa pipeline:

Pipeline	***Calidad del resumen***	***Extracción de etiquetas***	***Consistencia semántica***
Vosk solo (TXT plano)	Baja (sin resumen)	No disponible	Limitada
Vosk + GPT (actual sistema)	Muy alta	Completa y coherente	Muy alta

Tabla 15: Comparativa pipeline

Pruebas unitarias de los scripts

Se desarrollaron pruebas unitarias ad-hoc sobre los siguientes componentes:

- **test_conversion.py**
  Verificación de la conversión MP3 → WAV (ffmpeg).
- **test_procesar_json.py**
  Validación de la correcta extracción de campos desde los JSON de Enreach.
- **test_segmentacion.py**
  Verificación de la segmentación correcta en frases para el análisis con GPT.
- **test_vosk_vs_whisper.py**

Comparativa de WER y latencia entre Vosk y Whisper.

Motor	*WER (%)*	*Tiempo medio (s)*
Vosk	8,5	12,3
Whisper (base)	7,2	45,0
Google ASR	9,1	10,0

Tabla 16: Comparativa de Motores en WER y tiempo medio.

Consideraciones sobre la complejidad de integración

Durante el proceso de pruebas se observó que las soluciones cloud (**Google Cloud Speech-to-Text** y **Amazon Transcribe**), si bien ofrecían una precisión elevada, implicaban una complejidad adicional en su integración:

- Requerían la creación y gestión de un *bucket* de almacenamiento (GCS en Google Cloud, S3 en AWS).
- Era necesario subir previamente los ficheros de audio al *bucket*.
- Las APIs de transcripción se basaban en un modelo asíncrono: tras la petición, era necesario monitorizar el estado del *job* y descargar el resultado de forma diferida.
- Todo ello suponía un flujo de integración más complejo y menos inmediato.

Por el contrario:

- La transcripción con **Vosk** se realiza íntegramente en local, sin necesidad de subida previa de ficheros ni dependencias cloud.
- El análisis semántico con **GPT-3.5-turbo** se realiza directamente mediante llamada a la API de **OpenAI**, con resultados síncronos y formato JSON ya optimizado para su integración en la base de datos.

Este análisis llevó a optar por una arquitectura híbrida local/**cloud**, en la que:

- **Vosk** proporciona una transcripción robusta y eficiente.
- **GPT-3.5** completa el análisis semántico avanzado.

Validación de integración

Adicionalmente se comprobó que:

- El script **analizar_todas_llamadas_gpt.py** genera correctamente el fichero **analisis_total.json**.
- Los datos insertados en PostgreSQL respetan la estructura de la base de datos.
- Los cuadros de mando en **Streamlit** funcionan correctamente y son consistentes con los datos transcritos y analizados.

5.3 BASE DE DATOS Y ALMACENAMIENTO

El sistema desarrollado incorpora un componente de persistencia basado en una base de datos relacional **PostgreSQL**.

El objetivo de esta base de datos es proporcionar un almacenamiento estructurado y eficiente para los resultados del análisis de llamadas, facilitando su explotación posterior mediante cuadros de mando interactivos y consultas analíticas.

El diseño del esquema permite almacenar de forma normalizada la información clave extraída en el proceso de análisis, manteniendo la trazabilidad completa de cada llamada: metadatos, etiquetas, opiniones del cliente, clasificación de resultados, palabras clave, y otros campos relevantes.

Este enfoque garantiza:

- la disponibilidad inmediata de los datos para su visualización en el *dashboard*,
- la posibilidad de realizar análisis históricos y comparativos,
- y la integración con otros sistemas de la empresa a través de consultas SQL estándar.

5.3.1 MODELO Y ESQUEMA EN POSTGRESQL

El modelo de datos se ha diseñado con un esquema sencillo y optimizado para el análisis de llamadas.
El sistema almacena cada análisis de llamada como un registro en la tabla "analisis_llamadas", que contiene tanto los metadatos de la llamada como los resultados del análisis semántico realizado mediante **GPT**.

La siguiente figura muestra el diagrama entidad-relación (ER) de la tabla principal:

Tabla 17: Diagrama ER

A continuación, se detalla la estructura de la tabla y la función de cada uno de sus campos:

Campo	***Tipo de dato***	***Descripción***
Id	SERIAL PRIMARY KEY	Identificador único
fecha_analisis	DATE	Fecha en la que se realizó el análisis
Archivo	TEXT	Nombre del archivo de la llamada

numero_abonado	TEXT	Número de abonado (si existe)
numero_llamado	TEXT	Número marcado en la llamada
telefono_origen	TEXT	Teléfono desde el que se realiza la llamada
fecha_llamada	DATE	Fecha de la llamada
hora_llamada	TEXT	Hora de la llamada
agente_nombre	TEXT	Nombre del agente
agente_codigo	TEXT	Código del agente
Zona	TEXT	Zona (ELDA / NOVELDA)
Departamento	TEXT	Departamento asignado
Centralita	TEXT	Centralita de la llamada
Resumen	TEXT	Resumen de la llamada
palabras_clave	JSONB	Palabras clave extraídas
Etiquetas	TEXT[] (ARRAY)	Etiquetas asignadas
tipo_llamada	TEXT	Tipo de llamada (comercial / técnica)
tipo_incidencia	TEXT	Subtipo de incidencia

respuesta_agente	TEXT	Evaluación de la actuación del agente
Resultado	TEXT	Resultado de la llamada
opinion_cliente	TEXT	Opinión expresada por el cliente
motivo_opinion	TEXT	Motivo de la opinión
Localidad	TEXT	Localidad identificada
venta_realizada	BOOLEAN	Indicador de venta realizada
Resuelto	TEXT	Indicador de resolución (sí / no / pospuesta / derivada)
duracion_segundos	INTEGER	Duración total de la llamada en segundos
palabras_clave_categorizadas	JSONB	Palabras clave categorizadas por temas

Tabla 18: Elementos en la base de datos

El modelo de datos se ha diseñado siguiendo una arquitectura simplificada, con el objetivo de facilitar las consultas analíticas y la integración directa con el panel de visualización implementado en **Streamlit**.

Se ha optado por almacenar la información semántica (resumen, etiquetas, palabras clave, tipo de incidencia, etc.) junto con los metadatos técnicos de la llamada en una única tabla "analisis_llamadas".
Este enfoque permite realizar consultas rápidas, sin necesidad de *joins* entre múltiples tablas, y facilita la construcción de visualizaciones avanzadas en tiempo real.

Aunque algunos campos podrían ser normalizados (por ejemplo, agentes o zonas), se ha priorizado un modelo autocontenido y optimizado para el análisis exploratorio de los datos. El diseño actual es extensible y permite la incorporación de nuevas tablas auxiliares en futuras evoluciones del sistema, en caso de que se requiera un modelo relacional más completo.

La tabla "analisis_llamadas" constituye la fuente de datos principal utilizada por los cuadros de mando interactivos y por los procesos automatizados de *reporting*.

5.3.2 GESTIÓN FTP Y ARTEFACTOS

El sistema automatizado de análisis de llamadas se apoya en un flujo de integración que combina el uso de la API de **Masvoz** (**Enreach**) con un servidor **FTP** intermedio para la transferencia y almacenamiento temporal de los artefactos (grabaciones de audio).

Este mecanismo de intercambio mediante **FTP** garantiza una separación entre las operaciones de la API y los procesos de transcripción y análisis, y permite una mayor trazabilidad en el flujo de datos.

5.3.2.1 Proceso de descarga de artefactos

1. El script **descarga_audio.sh** se encarga de automatizar la solicitud de generación de los lotes de descarga a la API de **Masvoz**.

2. Una vez que la API ha preparado el lote, las grabaciones comprimidas en formato ZIP son depositadas automáticamente en el servidor **FTP** remoto, en una carpeta datada.

Ejemplo de estructura en el FTP remoto:

```
/2025-06-08/
└── llamadas_2025-06-08.zip
```

El script **download_ATC.sh** automatiza la descarga de los metadatos asociados a las llamadas en formato JSON (*transcription_YYYY-MM-DD.json*), que se almacenan localmente en:

```
/home/bea/ficherosdetalle/YYYY-MM-DD/

transcription_YYYY-MM-DD.json

output_YYYY-MM-DD.txt
```

La transferencia de las grabaciones comprimidas al servidor local se realiza mediante el script **descarga_audio.sh** en combinación con el cron del sistema, que invoca de forma periódica el proceso de descarga

Script	***Función***
descarga_audio.sh	Solicita generación del lote a la API y descarga el ZIP desde FTP.
download_ATC.sh	Descarga el fichero de detalle de llamadas (JSON + TXT).
procesar_json.sh	Procesa el fichero JSON para extraer los campos clave y generar el fichero TXT.
unzip_zipped0.sh	Descomprime automáticamente los ficheros ZIP descargados.

Tabla 19:Scripts involucrados en la gestión de artefactos y flujo FTP

5.3.2.2 Artefactos generados

El proceso completo genera de manera estructurada los siguientes artefactos:

- **Grabaciones de audio:**

Localizadas en */home/bea/YYYY-MM-DD_extracted/zipped0/YYYYMMDD/*, en formato MP3 tras la descompresión.

- **Metadatos de llamadas:**

Ficheros *transcription_YYYY-MM-DD.json* y *output_YYYY-MM-DD.txt* en */home/bea/ficherosdetalle/YYYY-MM-DD/.*

- **Artefactos temporales:**

Ficheros WAV temporales en */home/bea/temp_wavs/,* eliminados automáticamente tras el proceso de transcripción.

- **Resultados consolidados:**

Fichero **analisis_total.json** generado tras el análisis con GPT en **/home/bea/analisis_gpt/.**

Tipo de artefacto	***Ruta / Estructura***	***Descripción***
Grabaciones de audio	/home/bea/YYYY-MM-DD_extracted/zipped0/YYYYMMDD/	Audios en formato MP3 obtenidos tras la descompresión de las grabaciones descargadas.
Metadatos de llamadas	/home/bea/ficherosdetalle/YYYY-MM-DD/ transcription_YYYY-MM-DD.json → output_YYYY-MM-DD.txt	Ficheros JSON y TXT que contienen los metadatos de las llamadas recibidas. Sirven como referencia para el análisis posterior.
Artefactos temporales	/home/bea/temp_wavs/	Archivos WAV temporales generados para la transcripción con Vosk. Estos ficheros se

		eliminan automáticamente tras completar la transcripción.
Resultados consolidados	/home/bea/analisis_gpt/analisis_total.json	Fichero JSON consolidado que contiene los resultados del análisis semántico de las llamadas. Este fichero es el que finalmente se inserta en la base de datos y alimenta el panel de análisis de Streamlit.

Este diseño de ficheros y estructura permite mantener una alta trazabilidad en el proceso, facilitando auditorías y comprobaciones posteriores. Asimismo, la eliminación automática de artefactos temporales garantiza una gestión eficiente del almacenamiento en el servidor.

5.3.2.3 Ejemplo de estructura local resultante

```
/home/bea/

/home/bea/YYYY-MM-DD_extracted/zipped0/YYYYMMDD/

/home/bea/ficherosdetalle/YYYY-MM-DD/transcription_YYYY-MM-
DD.json

/home/bea/ficherosdetalle/YYYY-MM-DD/output_YYYY-MM-DD.txt

/home/bea/temp_wavs/  [temporales eliminados]
```

```
/home/bea/analisis_gpt/analisis_total.json
```

5.3.2.4 Justificación del uso de FTP

El uso de un servidor **FTP** intermedio responde a los siguientes objetivos:

- **Desacoplamiento:** Separar el flujo de generación de ficheros (**Masvoz** API) del procesamiento local.
- **Tolerancia a fallos:** Permitir reintentos en la descarga sin necesidad de repetir la generación completa del lote.
- **Trazabilidad:** Garantizar un registro completo de los artefactos descargados por fecha.
- **Interoperabilidad:** Adaptarse a las restricciones de la API de Masvoz, que deposita los ficheros por diseño en un FTP gestionado. [37]

El sistema incorpora además controles de logs y verificación de integridad (número de ficheros, tamaño total recibido) para asegurar la fiabilidad del proceso de descarga.

5.4 ORQUESTACIÓN Y AUTOMATIZACIÓN DEL SISTEMA

El sistema completo ha sido diseñado para funcionar de forma totalmente automatizada, sin intervención manual diaria. Esta orquestación se consigue mediante una combinación de scripts programados y tareas planificadas a través del servicio **cron** de Linux, que permiten ejecutar los procesos en horarios definidos y en el orden correcto.

5.4.1 AUTOMATIZACIÓN MEDIANTE TAREAS PROGRAMADAS (CRON JOBS)

Para garantizar que el flujo de trabajo se ejecute cada día con puntualidad y sin intervención, se han definido *cron jobs* que ejecutan los siguientes scripts:

Script	***Frecuencia Hora Función***

descarga_audio.sh	Diaria	2:05	Solicita grabaciones a la API y descarga los audios desde el FTP.
download_ATC.sh	Diaria	02:05	Descarga el fichero de metadatos transcription_YYYY-MM-DD.json.
procesar_json.sh	Diaria	02:30	Procesa el JSON para generar el archivo output.txt.
unzip_zipped0.sh	Diaria	05:00	Descomprime los archivos ZIP descargados.
transcribir_paralelo.py	Diaria	06:20	Transcribe las llamadas usando Vosk en paralelo.
analizar_todas_llamadas_gpt.py	Diaria	08:45	Analiza semánticamente las transcripciones y guarda los resultados.
delete_all.py	Diaria	16:00	Elimina todos los archivos descargados y creados para liberar espacio.

Tabla 20: Archivos utilizados con Cron Jobs

Estos procesos están encadenados de forma cronológica para asegurar que cada etapa finalice antes de comenzar la siguiente.

5.4.2 GESTIÓN DEL ALMACENAMIENTO Y ELIMINACIÓN DE ARCHIVOS TEMPORALES

El sistema ha sido diseñado para mantener un uso eficiente del almacenamiento local, evitando la acumulación innecesaria de archivos intermedios. Una vez finalizada la transcripción de las llamadas, los archivos de audio temporales en formato WAV generados durante el proceso (almacenados en */home/bea/temp_wavs/*) son eliminados automáticamente. Esta limpieza está integrada dentro del propio script de transcripción (**transcribir_paralelo.py**), de modo que cada proceso borra sus ficheros una vez procesado el audio correspondiente.

Además, los artefactos comprimidos descargados desde el servidor FTP son descomprimidos en una carpeta estructurada por fecha *(/home/bea/YYYY-MM-DD_extracted/),* y estos directorios pueden ser eliminados manualmente o a través del cron tras validar que todo el proceso de análisis ha concluido correctamente y los resultados han sido insertados en la base de datos. De esta forma, se garantiza que el sistema no acumula archivos que ya no son necesarios, liberando espacio en disco de forma periódica y controlada.

Esta estrategia asegura un balance entre trazabilidad de datos y uso racional del almacenamiento, sin necesidad de herramientas externas adicionales.

Directorio / Archivo	***Contenido***	***Eliminación***
/home/bea/temp_wavs/	Audios WAV temporales generados para transcripción	Automática tras procesar cada archivo (en el script)
/home/bea/YYYY-MM-DD_extracted/	Audios MP3 descomprimidos	Eliminación automática con cron Jobs

/home/bea/analisis_gpt/analisis_total.json	Análisis completo de llamadas (formato JSON)	Conservado como resultado final, no se elimina, ya que se va transcribiendo cada día
/home/bea/ficherosdetalle/YYYY-MM-DD/	Metadatos de llamadas (JSON + TXT)	Conservados para trazabilidad, eliminables con cron jobs
/home/bea/scripts/*.log	Logs del sistema y de transcripción	Conservados para auditoría, limpieza manual opcional

Tabla 21:Politica de eliminación de los archivos

5.5 PANEL DE ANÁLISIS Y EXPLOTACIÓN DE RESULTADOS

Para facilitar la explotación de los resultados generados por el sistema, se ha desarrollado una interfaz interactiva mediante **Streamlit**, una herramienta de código abierto que permite construir aplicaciones web en Python de forma rápida, eficaz y visual.

El panel actúa como cuadro de mando, ofreciendo una vista global y detallada del comportamiento de las llamadas recibidas, clasificadas y analizadas semánticamente. Todos los datos se extraen directamente de la base de datos PostgreSQL, garantizando su consistencia con los resultados diarios procesados por el sistema.

Funcionalidades principales

Indicadores clave (KPIs):

- Número total de llamadas procesadas por día, agente o localidad.
- Distribución por zona (Elda / Novelda).
- Porcentaje de llamadas resueltas, derivadas, pospuestas o sin solución.
- Opiniones del cliente (positiva, negativa, neutral) y motivos asociados.
- Número de ventas realizadas en llamadas comerciales.
- Métricas resumen: total de llamadas, duración media, % de llamadas de abonados, localidades más frecuentes, tipos de llamada.

```
st.subheader("📊 Métricas resumen")
st.metric("Total llamadas", len(df_filtrado))
media_duracion = df_filtrado["duracion_segundos"].mean()
minutos = int(media_duracion // 60)
st.metric("Duración media", f"{minutos} min")
```

Código 26: Ejemplo de una métrica en el panel de KPIs.py

Análisis por agente:

- Número de llamadas por agente y evolución temporal.
- Porcentaje de opiniones positivas, negativas y neutras por agente.
- Duración media de llamadas por agente (en minutos).
- Visualización de actividad diaria por agente y tipo de llamada.
- Detección de llamadas derivadas gestionadas por cada agente.

```
df_heatmap["fecha"] =
pd.to_datetime(df_heatmap["fecha_llamada"]).dt.date
heatmap_data = df_heatmap.groupby(["fecha",
"agente_nombre"]).size().unstack(fill_value=0).T
fig = px.imshow(heatmap_data, text_auto=True)
st.plotly_chart(fig, use_container_width=True)
```

Código 27: Ejemplo de un gráfico para analizar volumen por agente

Análisis por localidad y centralita:

- Evolución de llamadas por localidad y día.
- Clasificación automática por centralita (Elda / Novelda).
- Comparativa temporal de opiniones por centralita.
- Visualización de etiquetas más frecuentes (router, móvil, técnica, etc.).

```
llamadas_por_dia = df_filtrado.groupby(["fecha",
"localidad"]).size().reset_index(name="llamadas")
fig = px.line(llamadas_por_dia, x="fecha", y="llamadas",
color="localidad")
st.plotly_chart(fig)
```

Código 28: Ejemplo de análisis por localidad

```
df["centralita"] = df["zona"]
if df["centralita"].str.lower().eq("elda").any():
    df["centralita"] = df["centralita"].str.upper()
```

Código 29: Clasificación automática por centralita

Clientes reincidentes:

- Identificación de abonados con más de una llamada.
- Listado top 10 de abonados reincidentes.
- Análisis de tiempo medio entre llamadas de un mismo cliente.
- Top 10 teléfonos no abonados que han contactado repetidamente.

```
reincidentes = df["numero_abonado"].value_counts()
reincidentes = reincidentes[reincidentes > 1]
```

```
st.dataframe(reincidentes.head(10))
```

Código 30: Ejemplo de localizar Clientes reincidentes

Consultas avanzadas y filtros:

- Filtro por fecha (día concreto, rango personalizado o todas las descargadas).
- Filtro por agente, departamento o zona.
- Filtro por tipo de llamada (comercial / técnica).
- Búsqueda por número de abonado: historial completo de llamadas asociadas.
- Filtro por palabras clave (ej. "baja", "router", "movistar").

```
# Filtro por agente
agente = st.sidebar.multiselect("Agente",
options=sorted(df["agente_nombre"].unique()))
df = df[df["agente_nombre"].isin(agente)]

# Filtro por palabras clave
palabras_seleccionadas = st.sidebar.multiselect("Palabras",
palabras_unicas)
df = df[df["palabras_clave_categorizadas"].apply(contiene_palabra)]
```

Código 31:Ejemplo de filtros

Visualizaciones temporales:

- Distribución media de llamadas por hora del día.
- Evolución de llamadas según tipo, opinión y agente.
- Mapa de calor de llamadas por franja horaria y tipo.

Consulta inteligente mediante GPT:

El sistema permite realizar preguntas en lenguaje natural mediante una búsqueda semántica optimizada con GPT, facilitando el análisis avanzado de patrones o detección de llamadas con condiciones específicas.

```
consulta_usuario = st.text_input("Escribe una pregunta sobre las llamadas")
if consulta_usuario:
    prompt_filtrado = [{
        "role": "system",
        "content": "Eres un asistente que interpreta preguntas sobre
llamadas y devuelve un JSON con filtros."
    },
    {
        "role": "user",
        "content": f"Interpreta esta pregunta y devuelve solo el JSON:
'{consulta_usuario}'"
    }]
    respuesta = client.chat.completions.create(model="gpt-4o",
messages=prompt_filtrado)
    filtros_dict = json.loads(respuesta.choices[0].message.content)
```

Código 32: Consulta GPT en la searchbar

5.5.1 DISEÑO UI/UX

El diseño del panel ha sido orientado a la simplicidad, la claridad visual y la eficiencia en el acceso a los datos. Se han utilizado componentes nativos de **Streamlit** (gráficas, tablas, selectores) con una estética coherente y colores consistentes con la identidad visual de la organización.

La navegación es **intuitiva y accesible incluso para perfiles no técnicos**, y la interfaz responde de forma dinámica a los filtros aplicados. La información se presenta de forma jerarquizada, priorizando la interpretación visual y la toma de decisiones basada en datos.

5.5.2 ARQUITECTURA DEL PANEL

El *dashboard* se estructura en dos módulos principales, implementados en los scripts **KPIs.py** y **Analisis_detallado.py**, respectivamente:

- **KPIs.py:** muestra indicadores clave de rendimiento de forma resumida y accesible. Incluye métricas globales y gráficas agregadas.
- **Analisis_detallado.py**: permite el análisis pormenorizado de llamadas, con opciones de filtrado avanzadas, búsquedas individuales, y visualización del contenido analizado por la IA.

Ambos módulos se integran en el archivo principal **app.py**, que gestiona la autenticación, la navegación entre secciones, y la conexión a la base de datos PostgreSQL.

Ilustración 25: Ejemplo de navegación entre ventanas

5.5.3 TECNOLOGÍAS UTILIZADAS

El desarrollo se ha realizado íntegramente en Python, empleando las siguientes herramientas y librerías:

- Streamlit para la interfaz web.
- Pandas y Plotly para el tratamiento y visualización de datos.
- SQLAlchemy para la conexión con PostgreSQL.
- OpenAI API (GPT-3.5-Turbo) para las consultas semánticas.
- bcrypt y users.json para la autenticación con contraseña cifrada.

SISTEMA DE AUTENTICACIÓN

El acceso al panel está protegido mediante un sistema de *login* de usuario. Las credenciales están gestionadas a través de un archivo users.json, que almacena nombres de usuario y contraseñas cifradas mediante *hashing* seguro (SHA256). Cada usuario accede a una cuenta individual, permitiendo una trazabilidad clara de los accesos al sistema.

Ilustración 26: Control de inicio de sesión al navegar

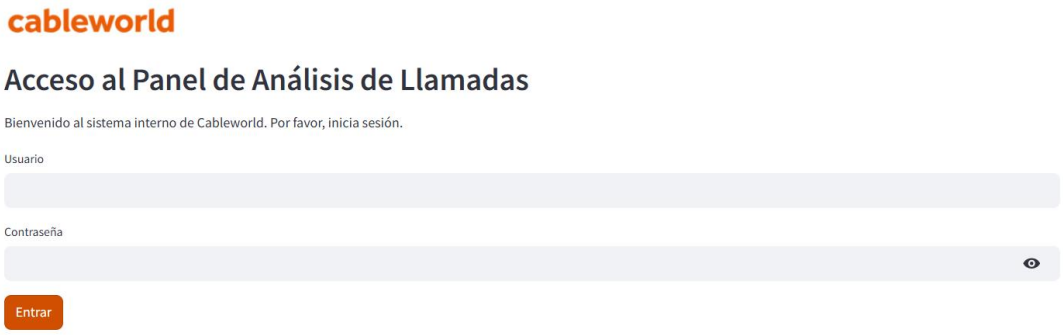


Ilustración 27: Panel inicial para iniciar sesión

Cuando un usuario intenta acceder al panel sin haberse autenticado previamente, el sistema bloquea la navegación y solicita el inicio de sesión. Esta interfaz de acceso permite introducir las credenciales y acceder de forma segura al panel, asegurando que cada interacción quede registrada y asociada a un usuario concreto.

5.5.5 INTERACCIÓN CON LA BASE DE DATOS

El *dashboard* se conecta directamente con la base de datos PostgreSQL, desde donde extrae en tiempo real los resultados del análisis de llamadas (almacenados previamente por el script **analizar_todas_llamadas_gpt.py**). Esta conexión permite consultar:

- Resúmenes diarios de actividad.
- Estadísticas por agente, zona, tipo de llamada o localidad.
- Opiniones de clientes y motivos.
- Etiquetas más frecuentes y palabras clave mencionadas.
- Información sobre clientes reincidentes.

La información se actualiza dinámicamente en función de los filtros aplicados por el usuario.

```
# Conexión a PostgreSQL
from sqlalchemy import create_engine
import pandas as pd

engine =
create_engine("postgresql://usuario:contraseña@localhost:5432/llamadas")
query = "SELECT * FROM analisis_llamadas WHERE fecha_llamada >= '2025-06-
01'"
df = pd.read_sql(query, engine)
```

Código 33: Consulta de datos analizados desde la base de datos PostgreSQL

Este enfoque permite que el panel visualice siempre los datos más recientes disponibles, ofreciendo estadísticas actualizadas en tiempo real y adaptadas a los filtros definidos por el usuario.

5.5.6 FUNCIONALIDADES DEL PANEL

El *dashboard* está diseñado para ofrecer una experiencia interactiva, basada en filtros personalizables, visualizaciones dinámicas y consultas directas sobre los datos almacenados en la base de datos. Las funcionalidades se agrupan en los siguientes bloques:

5.5.6.1 Filtros y segmentación de datos

El usuario puede aplicar filtros combinados para acotar la información mostrada en el panel. Estos filtros permiten extraer patrones relevantes de forma ágil y precisa:

- **Por fecha:** permite seleccionar un día concreto, un rango de fechas o visualizar todas las llamadas disponibles.
- **Por agente**: segmenta las métricas por agente comercial o técnico.
- **Por localidad o zona:** filtra por poblaciones concretas o por zona operativa (Elda / Novelda).
- **Por departamento:** útil para agrupar llamadas por áreas organizativas.
- **Tipo de llamada:** comercial o técnica.
- **Tipo de opinión:** positiva, negativa o neutral.
- **Estado de resolución:** llamadas resueltas, derivadas, pospuestas o sin solución.
- **Por número de abonado o número de teléfono**: permite buscar clientes concretos y visualizar su historial de consultas.

5.5.6.2 Aplicación de filtros combinados

La herramienta permite combinar estos filtros para realizar análisis complejos. En la Ilustración se muestra un ejemplo en el que se filtran únicamente las llamadas **negativas**, de **tipo técnico**, **atendidas por la agente Maika**, en la **tercera semana de mayo** y procedentes de **Yecla**, una localidad con alta recurrencia de incidencias.

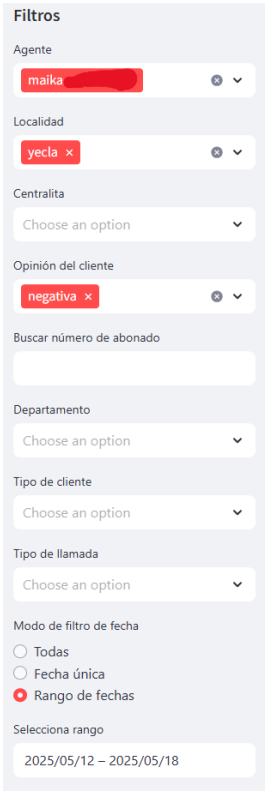


Ilustración 28: Ejemplo de la aplicación de filtros

Seguimiento de abonados recurrentes

Otra funcionalidad destacada del sistema es el seguimiento individualizado de abonados con múltiples llamadas. Por ejemplo, en la **Ilustración de la tabla** se detecta que un abonado ha realizado **nueve llamadas** en la misma semana, lo que puede ser indicativo de una incidencia no resuelta o de una gestión ineficaz.

En la **Ilustración 30**, se muestra cómo, al introducir su número de abonado en el filtro, se puede visualizar rápidamente el tipo de consultas realizadas y el tratamiento recibido, permitiendo así evaluar la calidad del soporte proporcionado y tomar decisiones correctivas si fuera necesario.

Ilustración 29:Ejemplo de buscar por numero de abonado

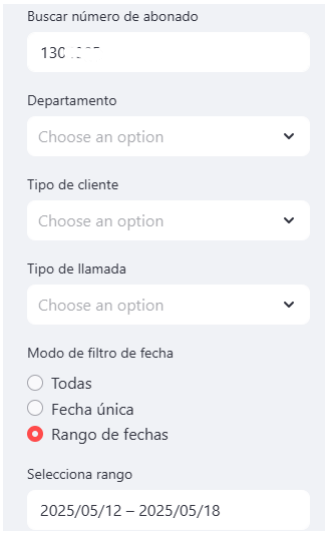


Ilustración 30:Ejemplo aplicamos filtro por número de abonado

5.5.6.3 Visualizaciones y gráficos interactivos

Todas las visualizaciones mencionadas en este apartado se encuentran integradas en el *dashboard* y se presentan en detalle en el capítulo 6, junto con el análisis correspondiente. A continuación, se muestra una vista general del panel interactivo con filtros aplicados y representación gráfica dinámica.

Las gráficas generadas en el panel permiten interpretar visualmente el comportamiento y desempeño del sistema. Entre las más destacadas se incluyen:

- **Número total de llamadas** por día, localidad, agente o zona.
- **Distribución global de opiniones** de los clientes.
- **Opiniones por agente**, clasificadas por tono y frecuencia.
- **Evolución temporal** de llamadas por tipo de opinión o por centralita.

- **Etiquetas más frecuentes** detectadas por la IA (por ejemplo: router, móvil, incidencia).
- **Etiquetas detectadas según el Excel:** Los directivos de la empresa proporcionaron un Excel para palabras que ellos considerasen clave según la categoría. Se encuentra como palabras_clave_categorizadas.
- **Distribución horaria** de llamadas (media diaria y por tipo de llamada).
- **Actividad individual por agente**, por día y acumulada.
- **Ventas realizadas** en llamadas comerciales.
- **Duración media de llamadas** por agente.
- **Análisis de palabras con tono negativo**, expresado como porcentaje.
- **Clientes reincidentes**: evolución mensual, tiempo medio entre llamadas, top de abonados y no abonados reincidentes.
- **Evolución de llamadas derivadas,** para controlar desviaciones del flujo normal de atención.
- **Métricas resumen**, como:
    - Total de llamadas.
    - Duración media de las llamadas.
    - Porcentaje de llamadas realizadas por abonados.
- Localidades más frecuentes y tipo de llamadas.

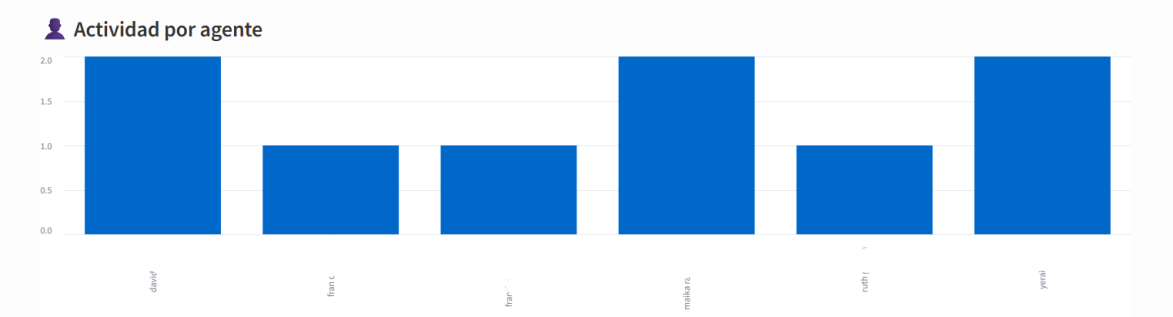


Ilustración 31:Ejemplo de Actividad por agente

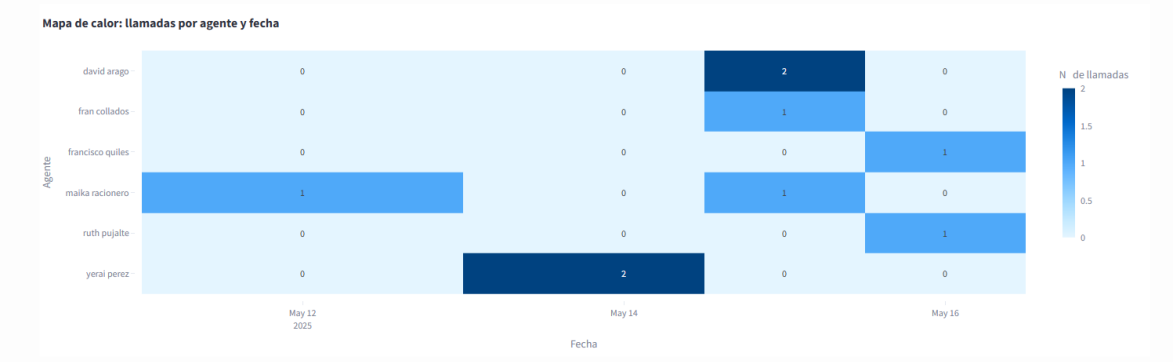


Ilustración 32:Ejemplo de resultados a los filtros aplicados en 5.5.6.1

A través del módulo **Analisis_detallado.py**, el usuario puede realizar un seguimiento exhaustivo de llamadas individuales:

- Ver todas las llamadas asociadas a un número de abonado.
- Comprobar el tipo de llamada, la opinión recogida, palabras clave, resolución, duración y agente implicado.
- Visualizar el resumen y etiquetas generadas por el modelo de lenguaje.
- Identificar si el cliente ha llamado varias veces, y en qué fechas.

motivo_opinion	localidad	centralita	departamento	zona	nombre_real	palabras_clave_categorizadas
El cliente recibió información y acordó una visita para solucionar el problema con el v	ontinyent	ELDA	SAT	ELDA	Maika	{"A. TECNICA":["fallo","mal"],"FIBRA":["fibra","internet",
El cliente parece estar satisfecho con la explicación del agente y la ayuda ofrecida.	ontinyent	ELDA	SAT	ELDA	YERAI	{"A. TECNICA":["mal"],"FIBRA":["internet","router"],"MO
El cliente parece estar conforme con la gestión realizada hasta el momento.	ontinyent	ELDA	SAT	ELDA	YERAI	{"MOVIL":["datos"],"SERVICIO":["calidad"]}
El cliente espera una pronta solución a su problema con la señal de las videocámaras	ontinyent	ELDA	SAT	ELDA	Davic	{"A. TECNICA":["funciona"],"FIBRA":["router"],"MOVIL":[
El cliente recibió la información necesaria, pero quedó con dudas sobre la configurac	ontinyent	ELDA	SAT	ELDA	Maika	{"FIBRA":["router"],"MOVIL":["datos"],"SERVICIO":["cali
El cliente está frustrado porque a pesar de haber realizado llamadas anteriores, el problema persiste.	ontinyent	ELDA	SAT	ELDA	Fran	{"A. TECNICA":["funciona"],"FIBRA":["router"],"MOVIL":[
El cliente espera que se solucione el problema con las cámaras de vigilancia.	ontinyent	ELDA	SAT	ELDA	David	{"A. TECNICA":["funciona","mal"],"FIBRA":["router"],"MO
El cliente espera que el departamento técnico resuelva el problema.	ontinyent	ELDA	SAC	ELDA	Ruth	{"FIBRA":["router"],"MOVIL":["datos"],"SERVICIO":["cali
El cliente parece estar frustrado por la situación pero acepta la ayuda ofrecida por el	no identifi	ELDA	SAT	ELDA	Francisco	{"A. TECNICA":["funciona"],"FIBRA":["router"],"MOVIL":[

Ilustración 33: Detalle de los filtros aplicados en 5.2.6.2

4. Consulta inteligente con IA

El sistema incorpora un campo de consulta natural basado en **GPT-3.5-Turbo**, que permite realizar preguntas complejas directamente sobre los datos, como por ejemplo:

¿Cuántas llamadas desde Aspe han sido comerciales y han terminado en venta positiva?

¿Qué agentes han gestionado más incidencias relacionadas con WiFi este mes?

Este motor semántico permite interpretar la intención del usuario y devolver resultados ajustados a los filtros inferidos, sin necesidad de conocimientos técnicos o SQL.

Ilustración 34: Consulta inteligente con GPT

Usabilidad y escalabilidad

El diseño del panel se ha centrado en ofrecer una experiencia de usuario intuitiva y eficiente, permitiendo a perfiles no técnicos interpretar los datos sin necesidad de acceder directamente a la base de datos. El sistema es fácilmente escalable y modular, permitiendo añadir nuevas visualizaciones o filtros conforme evolucionen los requisitos.

5.6 MEDIDAS DE SEGURIDAD

Dado que el sistema permite el acceso a información sensible derivada de conversaciones reales con clientes, se han implementado diversas medidas de seguridad orientadas a garantizar la confidencialidad, integridad y trazabilidad de los datos procesados. Estas medidas cubren tanto el acceso al panel de análisis como la protección del servidor en el que se ejecuta la plataforma.

5.6.1 AUTENTICACIÓN DE USUARIOS

El acceso al panel está restringido mediante un sistema de *login* individual. Cada usuario dispone de un nombre de usuario y una contraseña personal, almacenados en un archivo

users.json. Las contraseñas se encuentran cifradas utilizando hashing seguro mediante el algoritmo SHA256, impidiendo su recuperación en texto plano. [38] [39] [40]

Este mecanismo permite:

- Asegurar que solo usuarios autorizados acceden al sistema.
- Asignar sesiones individuales a cada usuario.
- Trazar el historial de accesos, facilitando la auditoría posterior.

5.6.2 CONTROL DE ACCESOS Y TRAZABILIDAD

cada usuario accede al sistema con sus propias credenciales, lo que permite una trazabilidad clara de:

- Quién ha accedido al sistema.
- Cuándo se ha accedido.
- Desde qué IP o dispositivo (si se activa el registro extendido).

Ilustración 35:Pantalla de bienvenida con trazabilidad de acceso y control de intentos fallidos

Ilustración 36: Usuario no encontrado : Acceso denegado

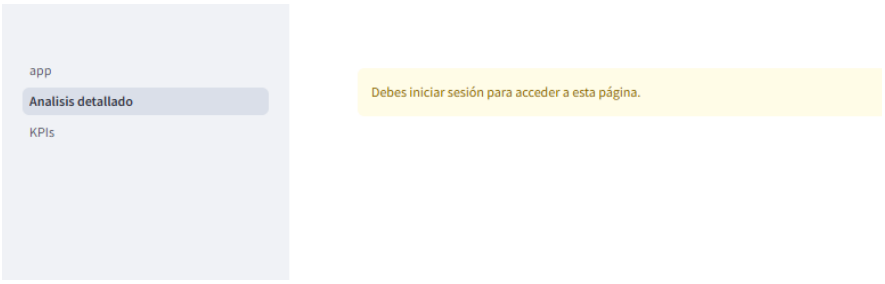


Ilustración 37: Requisito de identificarse para navegar por la página

Este enfoque es coherente con las buenas prácticas en protección de datos y auditoría exigidas por el Esquema Nacional de Seguridad (ENS).

5.6.3 PROTECCIÓN DEL SERVIDOR

El servidor donde se aloja el sistema ha sido configurado siguiendo un enfoque de **defensa en profundidad**, combinando distintos mecanismos de seguridad para reducir la superficie de ataque y garantizar su integridad. Las medidas implementadas son:

- **Acceso restringido mediante VPN L2TP/IPsec**: la conexión al servidor está limitada únicamente a través de una VPN segura, lo que impide el acceso desde direcciones IP externas no autorizadas.
- **Cortafuegos UFW configurado estrictamente**: se ha desplegado un firewall mediante `ufw`, cuya política por defecto es **denegar todas las conexiones entrantes**, permitiendo solo los puertos necesarios como SSH, los servicios internos del sistema y las conexiones VPN.

```
root@TFGBea:/home/bea/scripts# sudo ufw status verbose
Status: active
Logging: on (low)
Default: deny (incoming), allow (outgoing), disabled (routed)
New profiles: skip

To                         Action      From
--                         ------      ----
21/tcp                     ALLOW IN    Anywhere
8501/tcp                   ALLOW IN    Anywhere
8502/tcp                   ALLOW IN    Anywhere
8504                       ALLOW IN    Anywhere
22/tcp (OpenSSH)           ALLOW IN    Anywhere
8080                       ALLOW IN    Anywhere
21/tcp (v6)                ALLOW IN    Anywhere (v6)
8501/tcp (v6)              ALLOW IN    Anywhere (v6)
8502/tcp (v6)              ALLOW IN    Anywhere (v6)
8504 (v6)                  ALLOW IN    Anywhere (v6)
22/tcp (OpenSSH (v6))      ALLOW IN    Anywhere (v6)
8080 (v6)                  ALLOW IN    Anywhere (v6)
```

Ilustración 38: Estados de cortafuegos UFW mostrando puertos permitidos y política de acceso

- **Protección contra ataques por fuerza bruta con Fail2ban**: el sistema dispone de fail2ban monitorizando el acceso por SSH. Este servicio analiza los intentos de autenticación fallidos en los logs del sistema y bloquea temporalmente las IP sospechosas.

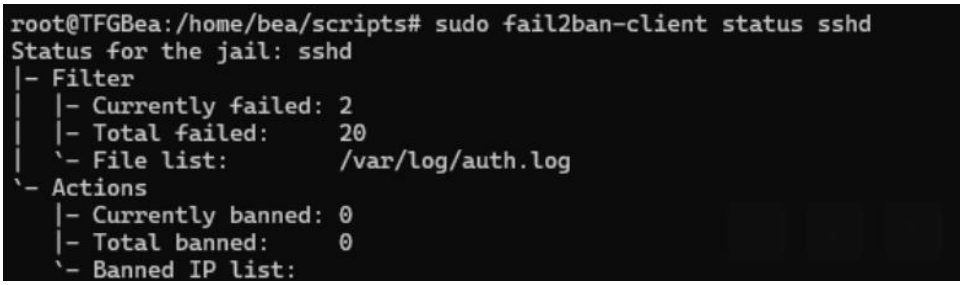


Ilustración 39: Estado de Fail2Ban mostrando el numero de intentos fallidos y archivo log monitorizado

- **Supervisión activa de logs de acceso**: los registros del sistema permiten detectar conexiones exitosas y fallidas, así como acciones ejecutadas por los usuarios autenticados. En los logs del 22 de junio se registró el acceso de la usuaria beatriz al sistema a través de SSH y posterior ejecución de comandos administrativos mediante sudo

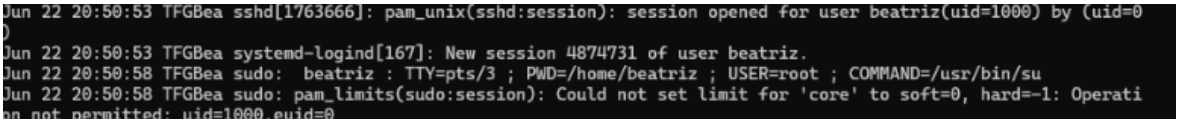


Ilustración 40:Registro de acceso vía SSH de la usuaria beatriz, reflejado en los logs del sistema.

5.6.4 SEGURIDAD DE DATOS Y CIFRADO

- Las contraseñas de usuarios no se almacenan en texto plano.
- La conexión con la base de datos PostgreSQL puede cifrarse utilizando SSL (opcional, si se habilita acceso remoto).
- No se almacenan datos de clientes reales más allá de números de teléfono parciales y el contenido anonimizado de las llamadas.
- Las transcripciones completas no se almacenan en la base de datos, sino como archivos .txt externos, lo que facilita el control de acceso y la depuración.

5.6.5 CUMPLIMIENTO DEL ESQUEMA NACIONAL DE SEGURIDAD (ENS)

Durante el desarrollo se han tenido en cuenta las recomendaciones básicas del ENS, especialmente en cuanto a:

- Autenticación robusta: contraseña con longitud mínima, uso de mayúsculas, números y símbolos.
- Cifrado de credenciales.
- Accesos individuales y trazabilidad.
- Segmentación de red mediante VPN.
- Posibilidad de ampliar a autenticación en dos pasos (2FA) en versiones futuras.

Estas medidas garantizan que el sistema pueda ser usado de forma segura en un entorno corporativo o académico que requiera protección frente a accesos no autorizados y exposición de datos sensibles.

5.7 PROBLEMAS TÉCNICOS Y MEJORAS APLICADAS

Durante el desarrollo del sistema se identificaron diversos problemas técnicos, errores de origen de datos y limitaciones derivadas del entorno de producción real. Este apartado documenta las incidencias más relevantes y las soluciones adoptadas en cada caso.

5.7.1 ERROR EN LA ELIMINACIÓN DE ARCHIVOS TEMPORALES .WAV

- **Síntoma**: el script lanzaba una excepción *FileNotFoundError* al intentar eliminar archivos .wav que no existían.
- **Causa**: algunas llamadas no generaban correctamente el archivo .wav temporal tras la conversión desde .mp3.
- **Solución**: se añadió una verificación con os.path.exists() antes de eliminar el archivo:

```
if os.path.exists(temp_wav):
os.remove(temp_wav)
```

Código 34: Verificación antes de eliminar un archivo

5.7.2 FECHA INCORRECTA EN LA BASE DE DATOS

- **Síntoma**: algunas filas en la base de datos mostraban como "fecha_llamada" la fecha de análisis (fecha del sistema).
- **Causa**: cuando no se encontraba la clave de relación (relaciones[clave]), el script tomaba por defecto la fecha actual.
- **Solución**: se modificó el script para que extraiga siempre la fecha real a partir del nombre del archivo o del transcription.json.

5.7.3 LOCALIDADES NO IDENTIFICADAS

- **Síntoma:** hasta un **70 % de las llamadas** aparecían inicialmente sin localidad identificada.
- **Causa:** cuando GPT no lograba extraer la localidad directamente del audio (por baja calidad, interrupciones o respuestas ambiguas), el script recurría al nombre del archivo. Si este también era genérico (por ejemplo, `Z_murcia_...` o simplemente `Z`), se clasificaba como `no identificada`. También se detectaron errores por **variaciones ortográficas**, como `San Vicente` vs `San Vicent`, o nombres duplicados como `elda Elda`.
- **Solución técnica:** se mejoró el sistema de fallback para extraer localidades del nombre de archivo de forma más precisa, se añadió normalización de tildes y se unificaron variantes lingüísticas.

```
}
[DEBUG] archivo: [966319975]_FW_966319975_Villena_2025-05-02_20-17-09_699824700.txt → clave: 2025-05-02_20-17-09_699824700 → duración
: None
 ^f^r Buscando agente AG17 en Excel...
[INFO] Normalizando centralita: ELDA  ^f^r ELDA
[INFO] Localidad corregida desde el archivo: villena
```

Tabla 22: Correción de localidad a través del nombre del archivo

Medida organizativa: se informó a los agentes de atención telefónica de la importancia de **preguntar siempre explícitamente la localidad al cliente**. En caso de no escucharla con claridad, se les recomendó **volver a preguntar**, con el fin de **reducir el número de llamadas sin localidad identificada y mejorar la calidad del análisis automático**.

5.7.4 ERRORES POR DATOS AMBIGUOS EN EL EXCEL DE AGENTES

- Síntoma: el agente asignado a algunas llamadas no coincidía con el agente real que había gestionado la llamada.
- **Causa técnica:** parte de la información procedente del Excel original contenía **claves duplicadas o ambiguas**, lo que llevaba a asignaciones incorrectas o nulas en algunos casos. Además, **los datos proporcionados desde Enreach tenían inconsistencias**: en la plataforma de Enreach existen dos fuentes distintas para la información de agentes:

  Config/Agentes

  Cuentas/Gestión de usuarios

  Según se informó internamente por parte del director del equipo técnico , **al crear los usuarios en Enreach se cruzaron erróneamente los datos con los que tenía AG66**, generando desajustes.
- **Solución técnica:** se decidió dejar de depender del Excel de agentes como fuente principal y utilizar **directamente los datos extraídos del archivo transcription.json**, mucho más robustos y alineados con la información real registrada por el sistema.
- **Solución organizativa:** Un técnico de sistemas revisará manualmente toda la lista de agentes y **abrirá un caso en Enreach** para corregir los errores detectados, ya que desde la web actual no es posible editar directamente esta información

5.7.5 PROBLEMAS DE DIARIZACIÓN POR CALIDAD DE AUDIO

- **Síntoma:** no se podía aplicar *diarización* automática en muchas llamadas.
- **Causa:** los audios proporcionados por Enreach estaban en **formato mono**, y para aplicar diarización se requiere **audio estéreo** (un canal por interlocutor).
- **Solución propuesta:** se ha solicitado a Enreach la grabación en estéreo, y se ha documentado la necesidad de esta mejora técnica en la plataforma externa.

5.7.6 PROBLEMAS DE RENDIMIENTO AL TRANSCRIBIR EN PARALELO

- **Síntoma:** durante la transcripción de múltiples llamadas, la CPU se saturaba completamente y la máquina virtual se bloqueaba.
- **Solución:** se aumentaron los recursos de la máquina virtual (núcleos y RAM) y se ajustó el número de procesos paralelos del script.

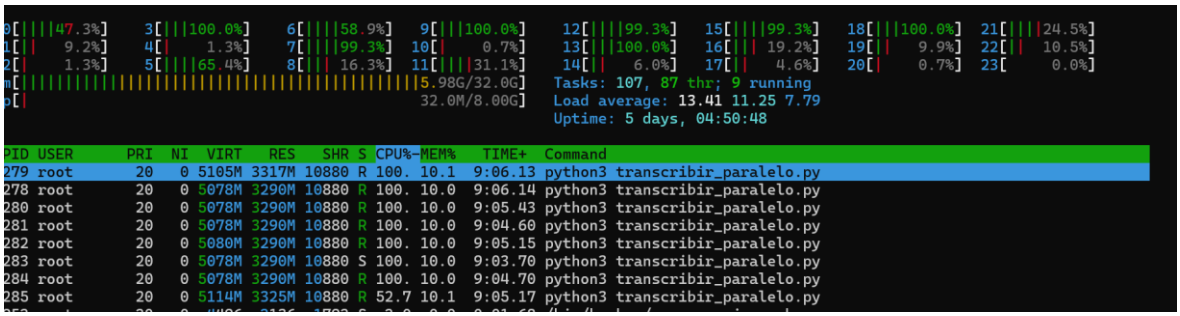


Ilustración 41: CPU sin saturar al aumentar la capacidad

En el momento de mayor carga, el sistema llegó a lanzar 9 procesos concurrentes de transcripción mediante el script **transcribir_paralelo.py**, ocupando el 100 % de los núcleos disponibles de la máquina virtual.

Esta ejecución provocaba una saturación de CPU que exigió ampliar los recursos del sistema para evitar bloqueos y mejorar la eficiencia del procesamiento en paralelo

5.7.7 INCONSISTENCIAS EN CAMPOS CATEGÓRICOS Y UNIFICACIÓN DE VALORES

- **Síntoma:** algunos valores categóricos como la opinión del cliente o el resultado de la llamada aparecían con variaciones ortográficas o semánticas, dificultando su análisis estadístico.
- **Causa:** los modelos de IA generaban variantes como neutra y neutral, o no identificado y no identificada, lo que fragmentaba los resultados al agrupar por esos campos.

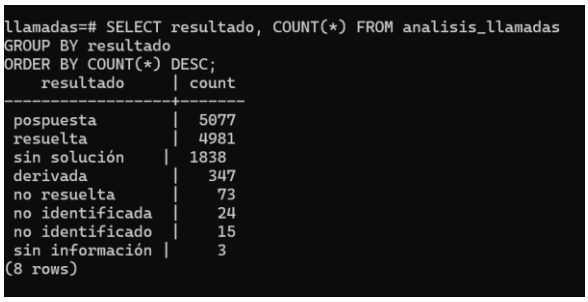


Ilustración 42:Fallo en "no identificado" y "no identifcada"

- **Solución**: se procedió a unificar estos valores mediante consultas SQL directamente sobre la base de datos **PostgreSQL**. Se aplicó *LOWER(TRIM(...))* para eliminar diferencias por mayúsculas, espacios o género gramatical.

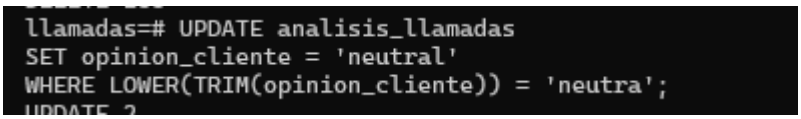


Ilustración 43:Fusión de "neutra" y "neutral"

Impacto: esta limpieza permitió mejorar significativamente la claridad de los gráficos y agrupaciones, como se observa en la visualización “Opiniones por agente” donde ya se reflejan solo las tres categorías consolidadas: positiva, negativa y neutral.

Capítulo 6. ANÁLISIS DE RESULTADOS

Este capítulo presenta una evaluación detallada de los resultados obtenidos tras la implementación completa del sistema automatizado de análisis de llamadas. Se pretende comprobar si se han alcanzado los objetivos definidos en fases previas del proyecto, valorando la eficacia de las técnicas de procesamiento del lenguaje natural (NLP) empleadas, la utilidad del *dashboard* interactivo para la supervisión operativa y la calidad de los datos recopilados.

Asimismo, se examinan los patrones detectados a partir de los datos analizados, desglosados por agente, localidad, tipo de llamada y opinión del cliente, entre otros factores. Finalmente, se analizan las mejoras implementadas durante el proceso, los errores detectados y corregidos, así como el impacto real de la solución desarrollada sobre el volumen de información procesada y su utilidad para la toma de decisiones.

6.1 VOLUMEN Y DISTRIBUCIÓN DE LLAMADAS

Durante el periodo analizado (mayo de 2025), el sistema procesó un total de **22.057 llamadas entrantes**, con una **media diaria de 735,2 llamadas**. Estas interacciones se distribuyen entre las dos centralitas principales: **Elda** y **Novelda** dimensionadas de acuerdo cada una al volumen que reciben. Además, se detectaron llamadas gestionadas en **turnos de guardia**, particularmente durante **fines de semana y franjas nocturnas**, lo que evidencia la actividad del sistema más allá del horario laboral habitual.

Para facilitar este análisis, desde el propio panel interactivo se seleccionó el **rango completo del mes de mayo** mediante el filtro de fechas del *frontend* (ver Ilustración 44), obteniendo así las métricas agregadas que se muestran en la Ilustración 45.

Nota importante: Aunque el presente análisis cubre la totalidad del mes de mayo, se ha decidido tratar por separado el impacto del **apagón del 28 de abril**. Dado que dicho evento

provocó una alteración significativa y puntual del servicio, su análisis se recoge al final del capítulo en un bloque específico ("Antes y después del apagón"), evitando así distorsionar las tendencias generales del mes.

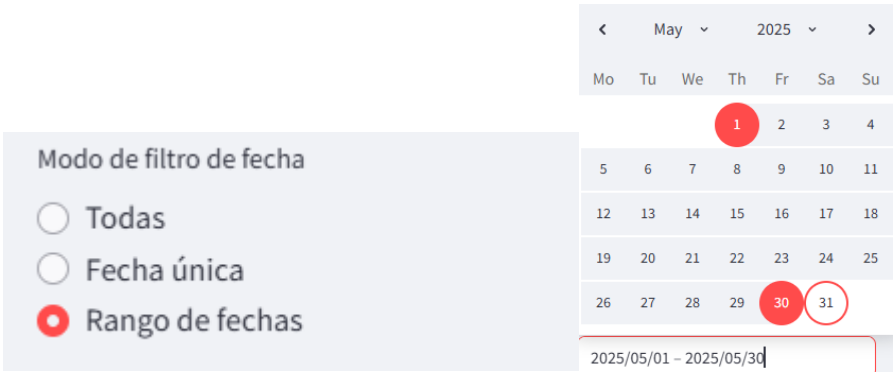


Ilustración 44:Seleccionar Fecha por Rango (mes de mayo)

Estas son las comprobaciones de todas las llamadas entrantes al *Call center* de cableworld

Ilustración 45:Métricas sacadas desde el FrontEnd

6.1.1 EVOLUCIÓN DE LLAMADAS POR CENTRALITA

Esta distribución permite observar los siguientes patrones:

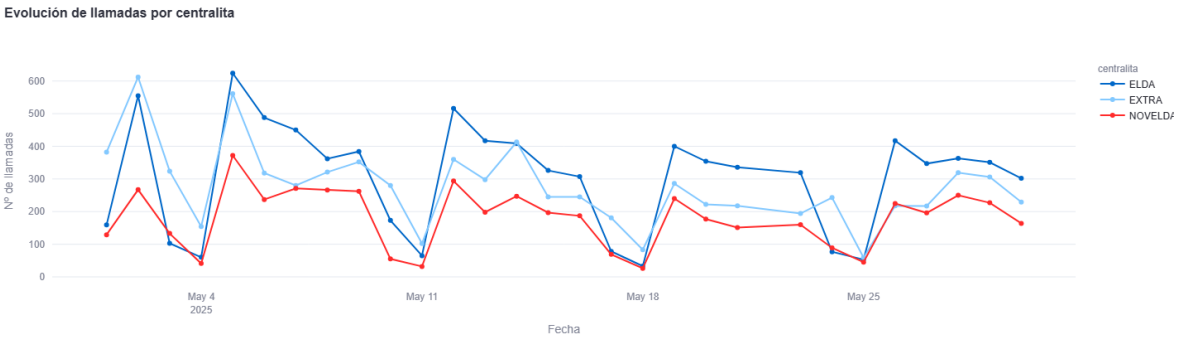


Ilustración 46: Gráfico de evolución de llamadas por centralita

Esta distribución permite identificar varios patrones relevantes:

- **Picos de carga los lunes y martes**: El mayor volumen de llamadas se concentra en los primeros días laborables de la semana. Esta tendencia puede deberse a:
  - La acumulación de incidencias durante el fin de semana.
  - Mayor disponibilidad de los clientes para contactar tras comprobar el funcionamiento de los servicios en casa.
  - Decisiones pospuestas en familia que derivan en contrataciones o consultas al inicio de semana.
- **Regularidad semanal**: Se observa un patrón cíclico con repuntes regulares al inicio de cada semana (especialmente los lunes) y una tendencia descendente hacia los fines de semana. Esto refuerza la necesidad de **dimensionar la plantilla por franjas semanales**, no solo diarias.
- **Días valle**: En varias semanas, los viernes y especialmente los sábados presentan un claro descenso de actividad. Este comportamiento puede ser aprovechado para programar tareas de *backoffice*, formación o mantenimiento interno de sistemas.

6.1.2 EVOLUCIÓN DE LLAMADAS POR LOCALIDAD Y DÍA

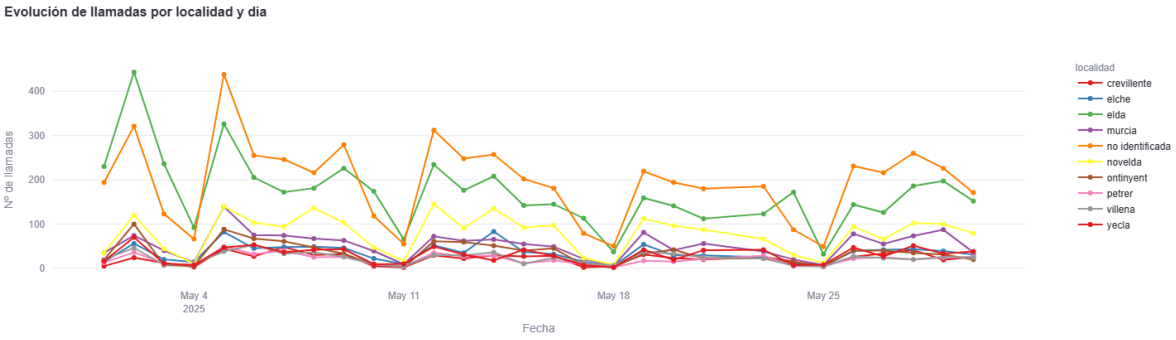


Ilustración 47:Llamadas por localidad y día

La evolución diaria del volumen de llamadas también se ha desglosado por localidad, permitiendo identificar con precisión **zonas de mayor actividad**, **picos de demanda** y **comportamientos diferenciales según el territorio**.

Los principales hallazgos son los siguientes:

1. Localidades con mayor volumen de llamadas

En las localidades de **Elda** y **Novelda** destacan claramente como los focos con mayor número de llamadas. Esta concentración es coherente con su peso histórico dentro del despliegue de cableworld: ambas fueron las primeras zonas en disponer del servicio (Elda y Novelda desde 1988), lo que explica su elevada **penetración de mercado**, ya que superan el **75 % de las viviendas**.

En fechas concretas, como el **2 y el 5 de mayo**, se registran picos que superan las **400 llamadas diarias** en Elda, lo cual coincide con la recuperación de servicio tras incidencias masivas (como el apagón del 28 de abril) o con el retorno al horario laboral tras el fin de semana.

2.Comportamientos dispares por localidad

Se observan comportamientos regulares y estables en localidades como **Elche, Ontinyent, Villena y Yecla**, con volúmenes moderados y sin oscilaciones bruscas. Esto puede deberse a que el servicio en estas zonas es más reciente y la **cultura de contacto telefónico con atención al cliente aún no está tan instaurada** como en las zonas históricas.

Por otro lado, se detecta que **Murcia** y el grupo de llamadas clasificadas como **"no identificadas"** mantienen volúmenes medios-altos. En el caso de Murcia, se trata de una expansión reciente con fuerte implantación. En cuanto a las llamadas no clasificadas, se profundiza a continuación.

3.Llamadas sin localidad identificada: problema y mejora

Durante las primeras semanas del análisis se detectó un número elevado de llamadas etiquetadas como **"no identificada"**. Esto dificultaba el análisis por zona, ya que no se podía saber con precisión desde dónde llamaba el cliente.

Las principales causas eran:

- **El cliente no menciona su localidad** durante la llamada, o lo hace de forma ambigua.
- **Las llamadas entrantes por bots o en horario de guardia** (fines de semana o noches) no incluyen esta información.
- En algunos casos, los **nombres de archivo de los audios** no aportaban información fiable o estaban mal etiquetados.

Desde principios de junio, se ha dado instrucción clara a los agentes para que **pregunten y repitan la pregunta sobre la localidad si no se entiende bien**, con el fin de mejorar la identificación geográfica. Aunque no todos lo cumplen al 100 %, la situación ha mejorado de forma notable.

Además, se han aplicado mejoras técnicas en los scripts de análisis y una tabla de normalización de nombres para **unificar localidades escritas de forma distinta** (por ejemplo, "Monóvar" y "Monovar").

6.2 OPINIONES DEL CLIENTE

El sistema no solo permite conocer si una llamada ha sido resuelta técnicamente, sino también **cómo ha percibido el cliente la atención recibida**. Esta valoración se clasifica automáticamente en opiniones **positivas, neutras o negativas**, utilizando técnicas de procesamiento de lenguaje natural (NLP) aplicadas sobre las transcripciones.

Opiniones por agente

El sistema permite analizar no solo la resolución técnica de las llamadas, sino también la percepción del cliente tras cada interacción, clasificada en opinión positiva, neutral o negativa. Esta información se ha extraído mediante técnicas de procesamiento de lenguaje natural aplicadas a las transcripciones, y representa uno de los indicadores más valiosos del sistema.

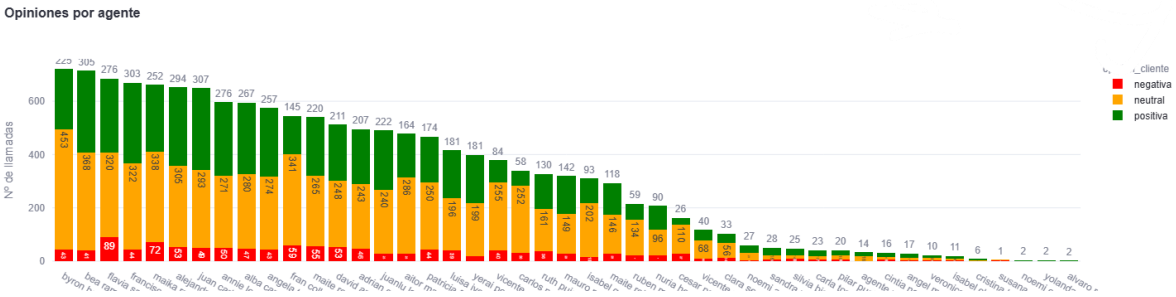


Ilustración 48:Gráfico de opiniones por agente

Se observa que los agentes con mayor volumen de llamadas tienden a concentrar un mayor número absoluto de opiniones negativas y neutras, lo que podría estar relacionado con sobrecarga operativa en jornadas de alta demanda. En cambio, algunos agentes con menor carga muestran porcentajes proporcionalmente más altos de opiniones positivas, lo que sugiere una mayor calidad en la atención individualizada.

Hallazgos principales:

- Los agentes con mayor volumen de llamadas tienden a acumular más opiniones neutras o negativas. Esto puede deberse a una sobrecarga operativa en días de alta demanda.
- Agentes con menor carga suelen mostrar un mayor porcentaje de opiniones positivas, lo que podría asociarse a una atención más pausada o personalizada.
- Las opiniones neutras aparecen con frecuencia cuando el cliente no recibe una solución inmediata, o se le indica que recibirá una llamada posterior con la resolución.

Por ejemplo: si el agente gestiona una incidencia técnica fuera del horario del SAT, lo habitual es que se le posponga la respuesta. En esos casos, el cliente no expresa ni satisfacción ni insatisfacción clara, resultando en una opinión neutra.

6.2.1 RANKING DE AGENTES SEGÚN VALORACIÓN DEL CLIENTE

Desde el panel interactivo es posible **filtrar por agentes** y visualizar directamente aquellos que acumulan más opiniones **positivas o negativas**, lo cual permite una evaluación rápida del rendimiento individual.

- La **Ilustración 49** muestra el ranking de agentes con mayor número de valoraciones **positivas**. Destacan perfiles pertenecientes tanto a la centralita de Elda como a la de Novelda, lo que indica una **calidad homogénea en la atención**, independientemente de la ubicación.

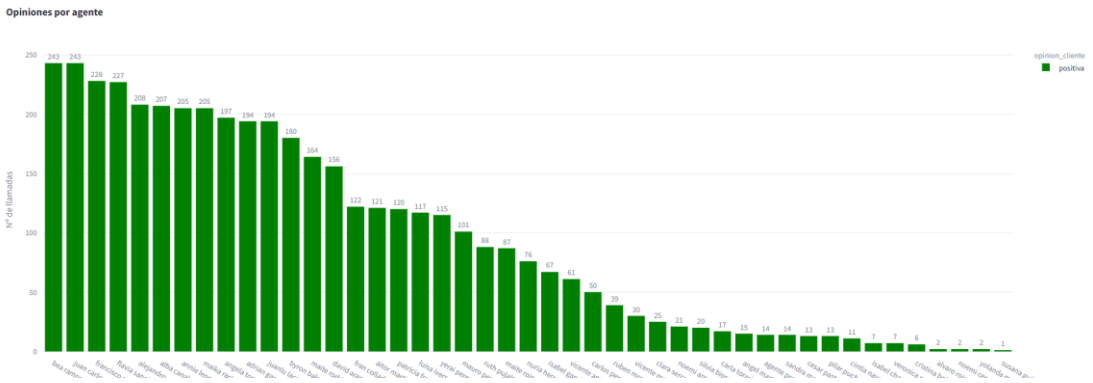


Ilustración 49:Ranking agentes con más valoraciones positivas

- La **Ilustración 50** recoge, en cambio, a los agentes con mayor número de opiniones **negativas**, permitiendo enfocar acciones de mejora específicas

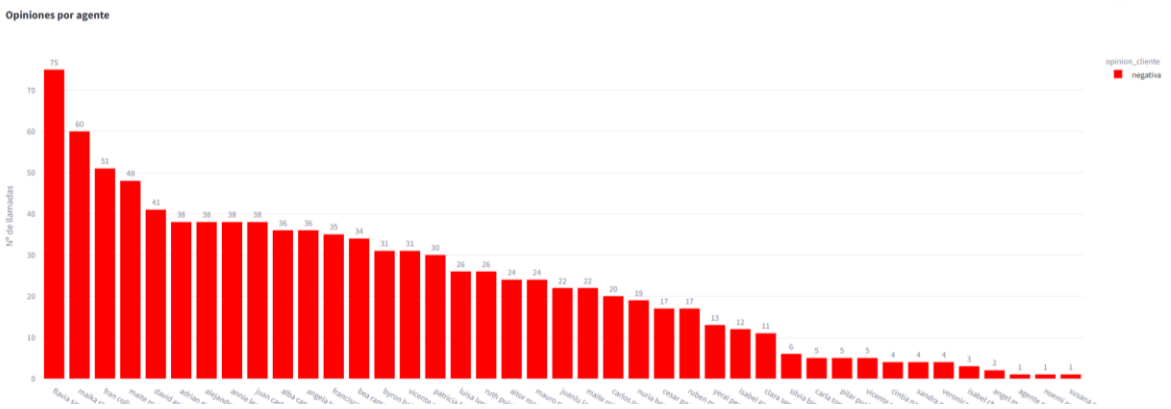


Ilustración 50:Ranking agentes con mas valoraciones negativas

Estos rankings se han utilizado para orientar acciones formativas internas, reforzar turnos con mejor percepción o redistribuir carga operativa en días críticos.

6.2.2 COMPARATIVA ELDA VS NOVELDA

Se ha realizado un análisis comparado entre las dos centralitas principales: Elda y Novelda, considerando tanto el volumen de llamadas como la distribución de opiniones.

- En la ilustración se muestra que el comportamiento es muy similar en ambas sedes: aproximadamente un 50 % de opiniones neutras, menos del 10 % negativas y el resto positivas.

Esta simetría refuerza la idea de que la calidad del servicio es consistente en ambas ubicaciones, aunque también revela ciertos aspectos comunes a mejorar.

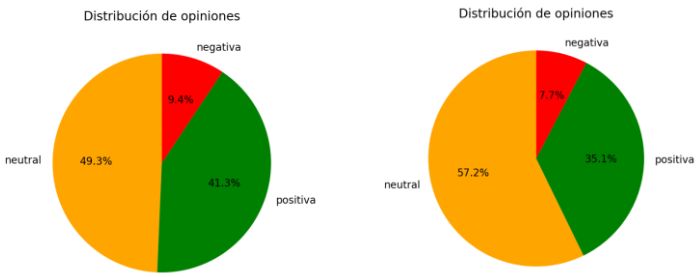


Ilustración 51:Distribución de opiniones Elda vs Novelda

Tras un análisis cualitativo, se concluye que la mayoría de las **opiniones neutras** se corresponden con llamadas en las que la resolución **se pospone**. Esto ocurre, por ejemplo, cuando el agente gestiona una incidencia **fuera del horario del SAT** o necesita validar información adicional, y se compromete a llamar posteriormente.

6.2.3 DISTRIBUCIÓN TEMPORAL DE OPINIONES DEL CLIENTE

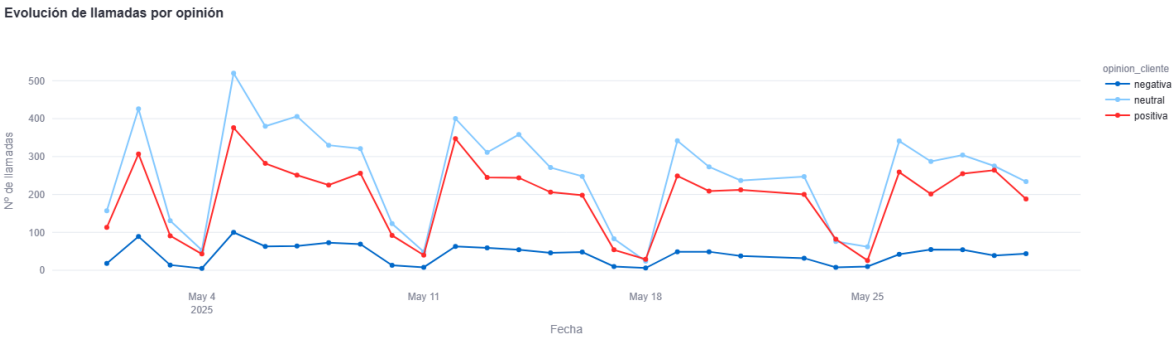


Ilustración 52: Evolución de llamadas por opinión

La Ilustración 52 muestra la evolución diaria de opiniones extraídas de las transcripciones mediante análisis semántico, clasificadas como positivas, neutrales o negativas.

Durante el mes de mayo se observa un patrón **constante**:

- Las opiniones neutras son mayoritarias, con varios picos que superan las 500 llamadas diarias.
- Las positivas siguen una tendencia paralela, pero con menor volumen.
- Las negativas se mantienen bajas, aunque con repuntes los lunes o días con mayor carga operativa.

¿Qué valor aporta este gráfico?

Aunque no muestra un cambio radical, sí confirma **tres fenómenos importantes** detectados gracias al sistema:

- **Correlación directa entre volumen y percepción**: En días de mayor entrada de llamadas (como los lunes o después de festivos), caen las opiniones positivas y aumentan las neutras o negativas.
- **Estabilidad en la atención**: A pesar de la carga, las opiniones negativas se mantienen siempre por debajo del 10 %, lo que muestra una atención sólida incluso en contextos complejos.
- **Hipótesis validada sobre opiniones neutras:** Gracias al análisis posterior con los agentes, se ha confirmado que la mayoría de las opiniones neutras corresponden a llamadas pospuestas, es decir, cuando el cliente no recibe una respuesta definitiva en el momento (por ejemplo, fuera del horario del SAT).

Mejoras aplicadas

Aunque la gráfica no muestra un "antes y después", **ha servido como base para implementar varios cambios:**

- Se reforzaron los turnos en lunes y jornadas críticas.

- Se trasladó a los agentes la **instrucción de dejar clara la fecha de retrollamada,** para evitar insatisfacción implícita.
- Se propuso la futura integración de llamadas salientes al sistema, para completar la percepción real del cliente (actualmente no se analizan).

6.3 ACTIVIDAD OPERATIVA POR AGENTE

El sistema desarrollado permite medir y visualizar de forma precisa la carga de trabajo de cada agente durante el periodo analizado. La Ilustración 53 muestra el número total de llamadas gestionadas por agente a lo largo del mes de mayo.

Se observan diferencias significativas:

- Algunos agentes han gestionado más de 550 llamadas en el mes, como *David*, *Flavia*, *Bea* o *Byron*, lo que equivale a más de 25 llamadas diarias en media.

- Otros agentes aparecen con volúmenes muy reducidos, lo que puede deberse a roles específicos (formación, bajas, soporte interno o cobertura parcial).

Este análisis permite extraer tres conclusiones clave:

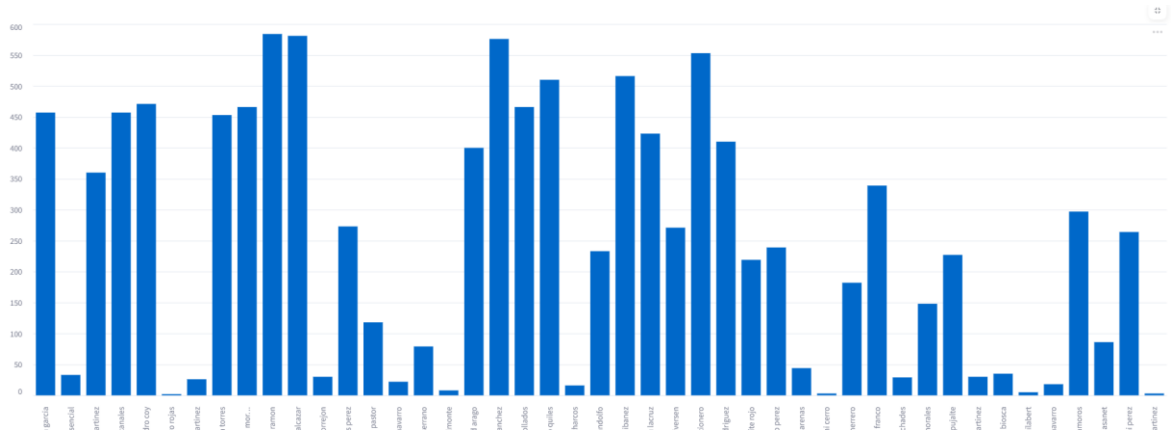


Ilustración 53: Actividad operativa por agente

1. Detección de desequilibrios

La distribución desigual de la carga sugiere que algunos agentes podrían estar asumiendo un volumen excesivo de llamadas, lo que **aumenta el riesgo de fatiga operativa** y puede impactar en la calidad de la atención.

2. Relación entre carga y satisfacción

Al cruzar estos datos con las opiniones recogidas, se observa que **una mayor carga no siempre se traduce en peores valoraciones**. De hecho, algunos agentes con alto volumen mantienen niveles aceptables de opiniones positivas, lo que permite **identificar perfiles de alta eficiencia**.

3.Optimización de plantilla

El análisis de los datos de llamadas ha permitido detectar diferencias en la carga de trabajo entre zonas y tipos de consulta, lo que ha servido como base para **ajustar la asignación de tareas y perfiles de forma más eficiente**.

En lugar de aplicar cambios estructurales, se ha apostado por **fomentar la polivalencia entre los agentes**, de modo que parte del equipo comercial ha recibido **formación básica**

en resolución de incidencias técnicas. Esta estrategia permite **reforzar temporalmente al equipo técnico en momentos puntuales de alta demanda**, sin necesidad de modificar la estructura organizativa.

Gracias a este enfoque, se mejora la capacidad de respuesta del servicio, se aprovechan mejor los recursos existentes y se favorece un entorno de trabajo más dinámico y colaborativo.

Este tipo de visualizaciones y métricas permite a los responsables del *Call center:*

- Evaluar el rendimiento individual con objetividad.
- Detectar agentes con potencial para formación cruzada o promoción.
- Identificar cuellos de botella o posibles mejoras en la planificación de turnos.

Además, el panel interactivo implementado en el *dashboard* permite **filtrar por agente y fecha**, para realizar seguimientos individuales diarios o semanales, y detectar rápidamente cambios de tendencia o incidencias particulares.

6.3.1 DURACIÓN MEDIA POR LLAMADA

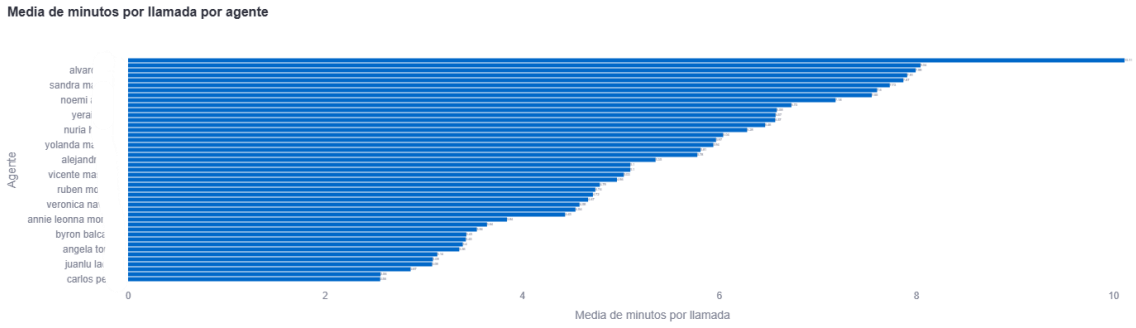


Ilustración 54:Media de minutos por llamada por cada agente

Además del volumen total de llamadas gestionadas, el sistema permite calcular la duración media por agente, un indicador fundamental para evaluar la eficiencia operativa y el tipo de atención prestada.

La **Ilustración** muestra la media de minutos por llamada durante el mes de mayo. Se observan diferencias significativas entre perfiles:

- **Álvaro**, **Sandra** y **Noemí** encabezan el ranking con una media superior a los 8 minutos por llamada, lo cual sugiere que gestionan consultas más complejas, posiblemente de carácter técnico, o que dedican más tiempo a asegurar una resolución completa.
- En el extremo opuesto, agentes como **Carlos**, **Juanlu** o **Ángela** presentan tiempos medios cercanos a los 3 minutos, lo que puede estar relacionado con:
    - Una mayor proporción de llamadas comerciales más breves.
    - Derivaciones tempranas a otros departamentos.
    - Un estilo de atención más ágil y directo.

Este análisis permite contrastar el rendimiento operativo con las opiniones del cliente, ya que una duración más larga no siempre implica una mejor percepción, y una llamada breve puede ser altamente efectiva si se resuelve al primer intento.

Conclusión:

La duración media es útil para detectar

- **Ineficiencias o cuellos de botella** si los tiempos son excesivos.
- **Agentes que podrían estar sobrecargados** con casos complejos.
- **Buenas prácticas** en agentes que equilibran tiempo y satisfacción, ideales para formación interna.

6.3.2 EVOLUCIÓN DIARIA POR AGENTE

El sistema desarrollado permite visualizar en detalle la carga diaria de trabajo de cada agente, lo que aporta una visión dinámica y operativa del rendimiento individual y colectivo.

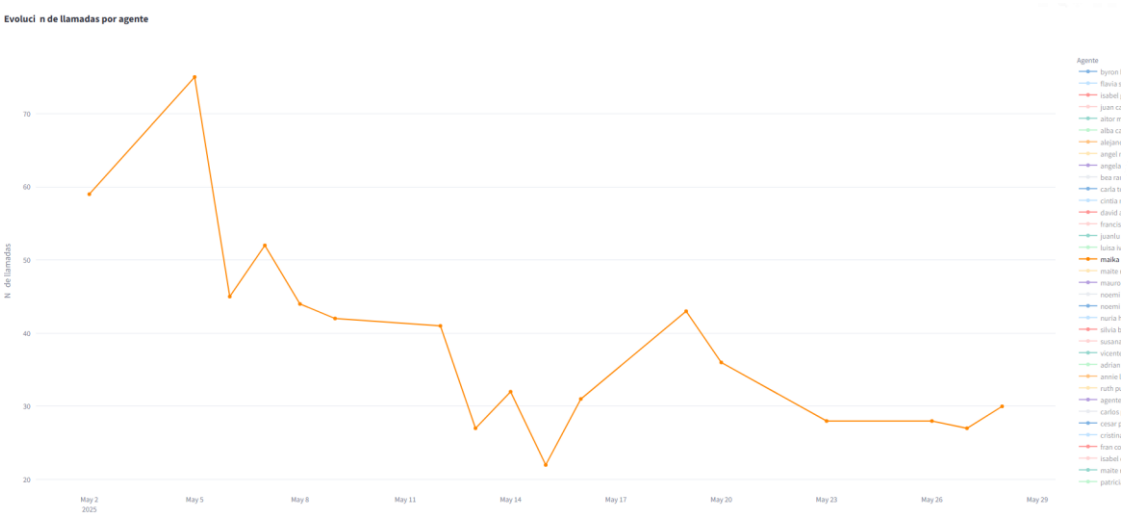


Ilustración 55:Evolución de llamadas por agente (Individual)

La **Ilustración** muestra la evolución de llamadas gestionadas diariamente por la agente **Maika**, una de las más activas del mes. En los primeros días de mayo, Maika gestiona entre **60 y 75 llamadas diarias**, volumen que desciende progresivamente hasta estabilizarse en torno a **30-40 llamadas** a partir de la segunda quincena. Esta tendencia descendente puede reflejar:

- Un **reajuste en la distribución de turnos**.
- Una **reorganización interna** del equipo.
- Una **reducción en el volumen de llamadas entrantes** en su zona.

Este tipo de visualización ha demostrado ser útil para detectar sobrecargas, prever descensos de rendimiento y adaptar la planificación de turnos según la evolución operativa.

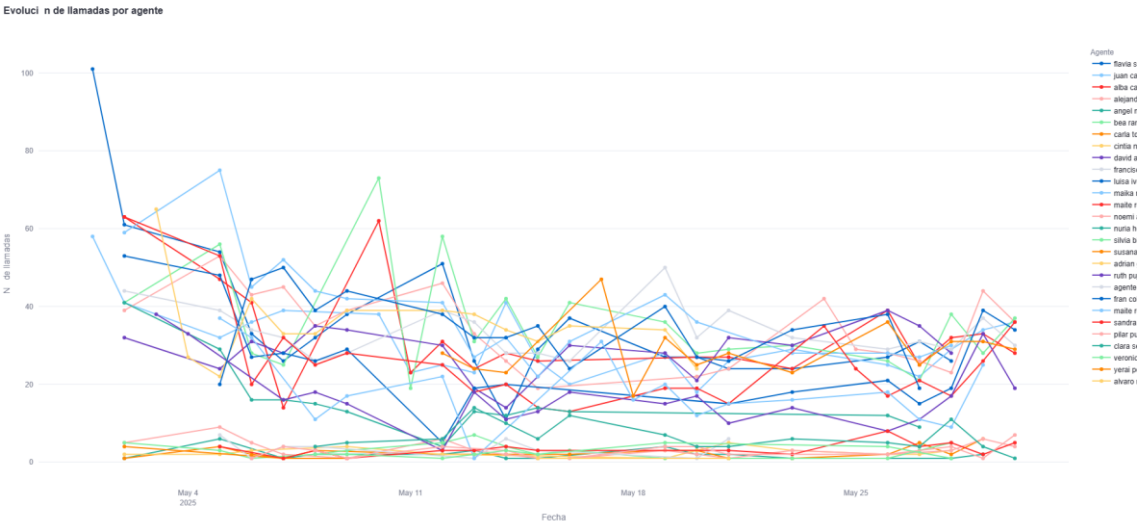


Ilustración 56:Evolución diaria de los agentes de la centralita de Elda

Por otro lado, esta Ilustración representa la evolución diaria de llamadas de los principales agentes de la **centralita de Elda** (por ejemplo, **Flavia**, **Alejandro**, **Francisco**, entre otros). Se observan picos notables —como el del **2 de mayo,** (día posterior a un festivo), en el que Flavia superó las **100 llamadas gestionadas**— seguidos de una tendencia de estabilización. Este comportamiento evidencia cómo el sistema es capaz de reflejar:

- El **impacto de eventos críticos** (como caídas de servicio o incidencias técnicas).
- La **capacidad de adaptación del equipo** tras situaciones de alta demanda.
- La **eficacia del reparto dinámico de carga** entre agentes.

Aplicación práctica del análisis:

a estas métricas se han podido justificar decisiones como:

- **El traspaso de agentes de una centralita a otra**, en función del volumen real detectado.
- **La planificación de refuerzos específicos en días pico**, como los lunes tras el fin de semana.
- **La detección de perfiles con alta rotación de llamadas**, útiles para entrenar en eficiencia u orientar a llamadas breves.

Este análisis no solo mejora el conocimiento interno de los ritmos operativos del *Call center*, sino que permite intervenir de forma **proactiva y fundamentada** en la mejora del rendimiento y la experiencia del cliente.

6.4 LLAMADAS DERIVADAS

Durante el mes de mayo, el sistema identificó un total de **623 llamadas derivadas**, es decir, casos en los que el agente inicial no pudo resolver la consulta o incidencia y fue necesario transferir la llamada a otro departamento o nivel de soporte.

6.4.1 OPINIONES TRAS LA DERIVACIÓN

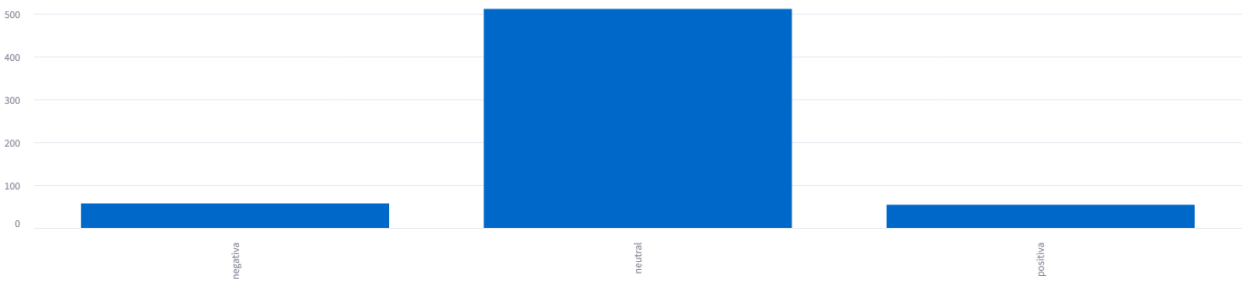


Ilustración 57:opiniones de las llamadas derivadas

Tal como se muestra en la gráfica, la distribución de opiniones del cliente tras una derivación presenta el siguiente patrón:

- Opiniones **neutras** (mayoría): Más del 80 % de las llamadas derivadas acaban clasificadas como neutrales, lo cual indica que en muchos casos la persona queda a la espera de una solución futura o no puede valorar aún el resultado.

- Opiniones **negativas**: Unas 60 llamadas reflejan insatisfacción por parte del usuario, probablemente debido a la falta de seguimiento, demoras o frustración por no resolver su consulta en el primer contacto.

- Opiniones **positivas**: Representan una minoría, aunque permiten deducir que algunas derivaciones sí finalizan con una gestión efectiva y satisfactoria.

6.4.2 ETIQUETAS MÁS FRECUENTES

El análisis semántico de estas llamadas revela que la mayoría están relacionadas con problemas de carácter técnico, seguido de consultas e incidencias. También aparecen en menor medida términos como *router*, comercial o calidad.

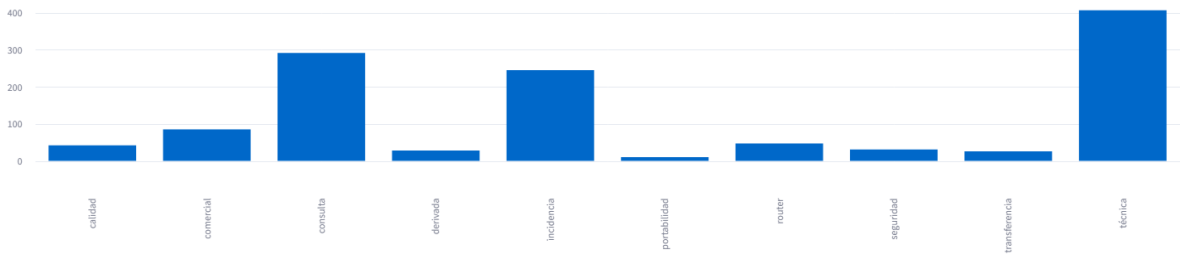


Ilustración 58:Etiquetas más frecuentes de las llamadas derivadas

Esto confirma que:

- Muchas derivaciones se deben a confusiones iniciales entre llamadas **comerciales** y **técnicas**: por ejemplo, usuarios que llaman por un fallo en el *router*, pero son atendidos inicialmente por un agente comercial sin herramientas para solucionarlo.

- La transferencia suele estar justificada, pero deja margen de mejora en términos de comunicación con el cliente y seguimiento posterior.

- Etiquetas como **portabilidad** o **transferencia** tienen escasa aparición, lo que sugiere que esos procesos se resuelven habitualmente en el primer contacto.

Este análisis pone de manifiesto una **oportunidad clara de optimización operativa**:

- **Establecer un flujo de seguimiento** tras cada derivación, especialmente en casos no resueltos.
- **Formar a los agentes comerciales** para que puedan realizar una primera clasificación más precisa, evitando derivaciones innecesarias.
- **Marcar en el sistema si la incidencia fue resuelta tras la derivación**, lo que permitiría analizar su eficacia real en futuras versiones del sistema.

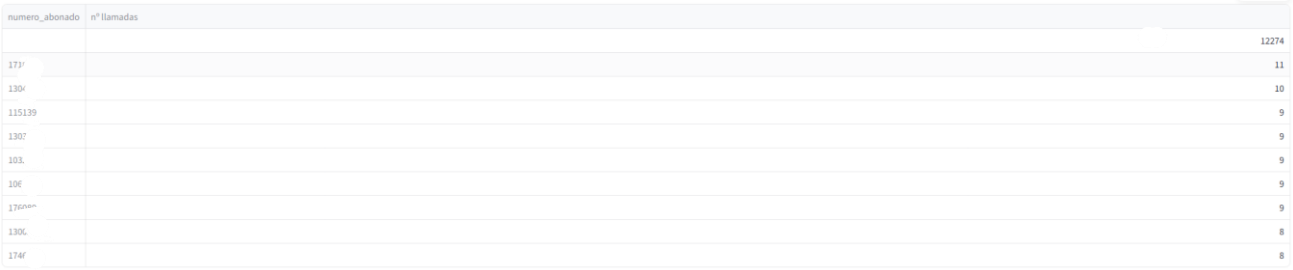
En resumen, aunque el sistema automatizado permite identificar con precisión este tipo de llamadas, su análisis muestra que aún hay espacio para mejorar la **satisfacción percibida** por el cliente en estos casos.

6.5 CLIENTES REINCIDENTES

Uno de los indicadores clave del sistema desarrollado ha sido la capacidad para detectar **clientes reincidentes**, es decir, aquellos que realizan **más de una llamada durante el mismo periodo**. Esta funcionalidad permite identificar patrones de resolución incompleta, incidencias persistentes o incluso oportunidades de mejora en el soporte post-llamada.

6.5.1 VOLUMEN TOTAL Y TOP 10

Durante el mes de mayo se recolectaron estos top abonados con el mayor número de llamadas reincidentes.



numero_abonado	nº llamadas
	12274
173	11
130	10
115139	9
1307	9
103.	9
106	9
176	9
130.	8
174	8

Ilustración 59: Clientes con más llamadas reincidentes

Este tipo de seguimiento permite detectar:

- Casos de **incidencias crónicas** o mal gestionadas.
- Posibles bucles de llamada o asignaciones erróneas.
- Necesidad de intervención preventiva en cuentas con alta reincidencia.

6.5.2 TIEMPO MEDIO ENTRE LLAMADAS

Se ha calculado el **tiempo medio entre contactos** de clientes reincidentes, obteniendo un valor de **84,8 horas** (aproximadamente 3,5 días). Este dato es útil para:

- Detectar ventanas críticas en las que intervenir antes de que el cliente vuelva a llamar.
- Valorar la eficacia de la resolución en primera llamada.
- Medir el impacto real de las acciones correctivas.

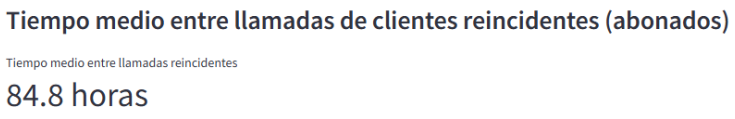


Ilustración 60: Cálculo según los filtro aplicados

6.5.3 EVOLUCIÓN DIARIA DE REINCIDENCIAS

La Ilustración de este apartado, muestra la evolución diaria de reincidentes. Los picos más pronunciados se observan los lunes y martes, coincidiendo con días de mayor carga operativa. Esto sugiere que:

- Muchos usuarios que llaman en fines de semana o guardias vuelven a contactar si no obtuvieron respuesta inmediata.
- Tras un corte importante (como el **apagón del 28 de abril**), se concentran reclamaciones en días posteriores. El 2 de mayo, por ejemplo, supuso el día con mayor volumen de reincidentes.

Ilustración 61:Evolución diaria de clientes reincidentes

6.5.4 REINCIDENCIA EN NO ABONADOS

Además de los clientes registrados, el sistema ha sido capaz de detectar patrones de reincidencia entre **usuarios no abonados**, es decir, números de teléfono que han realizado múltiples llamadas sin estar asociados a una cuenta de cliente activa.

Durante el mes de mayo, se identificaron **687 teléfonos de no abonados** con más de una llamada registrada. El caso más extremo realizó **25 llamadas en el mismo mes**, la mayoría de ellas relacionadas con temas técnicos como **problemas de cobertura o datos móviles**.

6.5.4.1 Posibles causas de reincidencia

Este tipo de comportamiento puede tener distintas interpretaciones:

- **Usuarios en proceso de contratación** que experimentan incidencias antes de completar el alta o la instalación.
- **Llamadas realizadas por familiares** del titular o desde otros dispositivos no vinculados al contrato.
- **Contactos recurrentes desde oficinas comerciales o empresas colaboradoras** que no figuran como abonados, pero gestionan llamadas en nombre de clientes.

6.5.4.2 Panel de seguimiento de llamadas

El sistema permite **filtrar y seleccionar cada número de teléfono** desde el panel interactivo, accediendo a todas las llamadas asociadas a ese contacto.

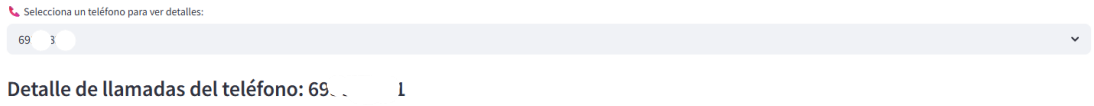


Ilustración 62:Captura del panel interactivo para enseñar como se puede seleccionar las llamadas

Una vez seleccionado el número, se despliega automáticamente el **detalle cronológico de todas las llamadas realizadas**, incluyendo:

- Fecha y hora
- Tipo de consulta (comercial, técnica…)
- Opinión del cliente tras cada interacción
- Palabras clave detectadas

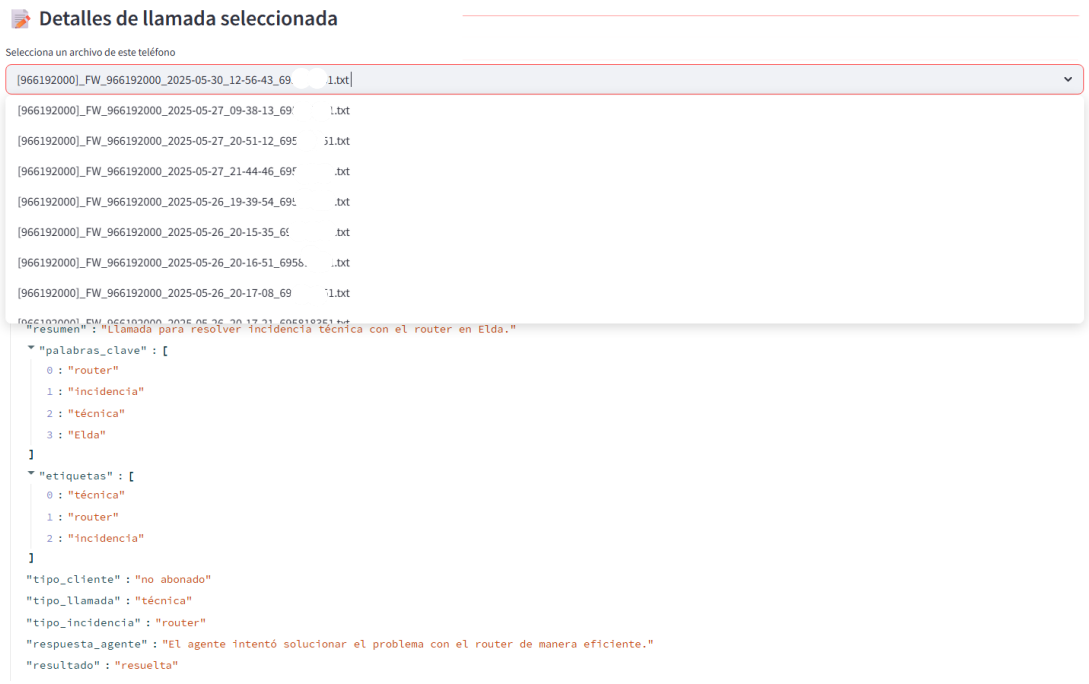


Ilustración 63:Seguimiento completo de llamadas por número no abonado

6.5.4.3 Implicaciones estratégicas

La reincidencia de no abonados debe considerarse como una **fuente de información comercial clave**, ya que:

- Permite **detectar oportunidades de captación** en usuarios con interés activo pero dudas técnicas.
- Ofrece una imagen clara de la **calidad de atención prestada a potenciales clientes**.
- Aporta datos útiles para campañas de **fidelización o contacto posterior**, especialmente si se integran con el CRM o sistemas de marketing.

Después de seleccionar se abre todas las llamadas con el detalle de cada una con este numero

6.5.5 PALABRAS CLAVE Y TIPO DE CONSULTA

El panel interactivo permite, además, aplicar un filtro por número específico para visualizar el detalle de todas sus llamadas. A través de este seguimiento, se puede analizar también el contenido semántico de las conversaciones, extrayendo las **palabras clave más mencionadas**.

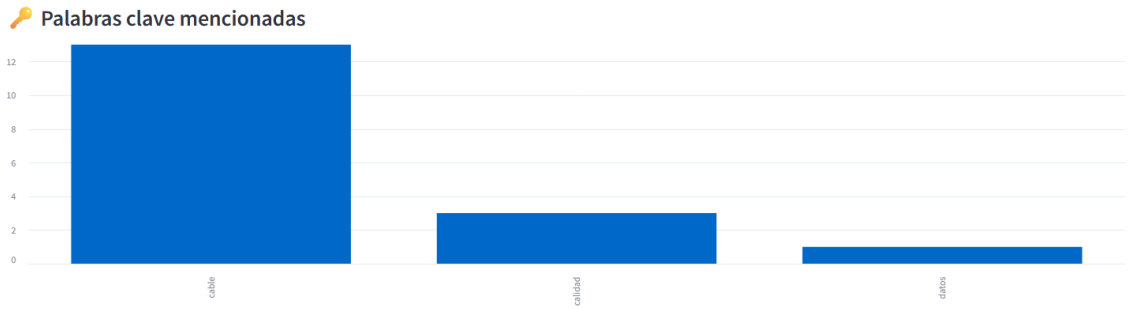


Ilustración 64:Filtro por número de teléfono y detalle semántico de llamadas

Gracias a este análisis, se ha observado que las llamadas reincidentes por parte de **no abonados** giran en torno a un conjunto de términos técnicos comunes, entre los que destacan:

- "Datos"
- "Cobertura"
- "Router"
- "Calidad"
- "Portabilidad"

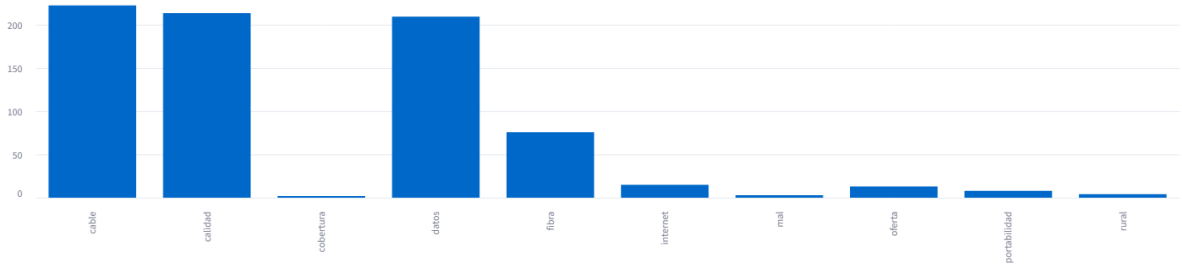


Ilustración 65:Palabras clave más frecuentes en llamadas de no abonados reincidentes

Este patrón refuerza la idea de que muchos de estos contactos están relacionados con problemas técnicos reales, aunque no provengan de clientes registrados. También permite identificar tendencias sobre los motivos más frecuentes de interés o reclamación antes de contratar.

En cuanto a la tipología de llamada, el 80 % de las comunicaciones corresponden a consultas de tipo técnico, mientras que el análisis de sentimiento revela que más del 50 % de las opiniones asociadas son positivas o neutras.

Esto sugiere que, aun sin poder ofrecer una solución completa al no tratarse de clientes formales, el sistema de atención mantiene una percepción general satisfactoria, lo cual puede influir positivamente en la conversión futura.

Conclusión: la monitorización semántica aplicada a no abonados ofrece información valiosa para reforzar la estrategia comercial, anticipar dudas técnicas frecuentes y mejorar la experiencia del usuario desde el primer contacto.

6.6 DISTRIBUCIÓN DE LLAMADAS POR HORA

El análisis horario de las llamadas entrantes revela un patrón muy claro: la mayoría de las interacciones se concentran en la franja de mañana. Tal como se observa en la distribución horaria de llamadas totales, el volumen de llamadas comienza a incrementarse a partir de las **8:00 h**, alcanzando su pico alrededor de las **10:00–11:00 h**, con más de 80 llamadas por hora. A partir de las 13:00 h, el número de llamadas desciende de forma significativa, cayendo a menos de la mitad entre las 14:00 y las 16:00 h.

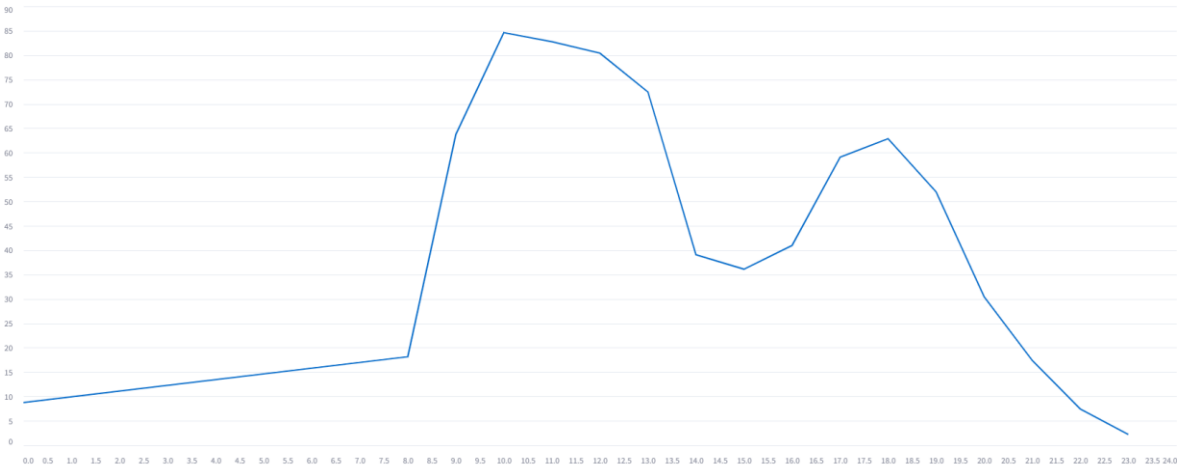


Ilustración 66:Distribución horaria de llamadas totales

Este comportamiento tiene implicaciones claras para el dimensionamiento del personal del *Call center*. Se recomienda establecer turnos escalonados o reforzar el personal en las horas punta de la mañana, mientras que en las horas valle (especialmente entre las 14:00 y las 16:00 h), los agentes podrían desempeñar tareas complementarias como *backoffice*, seguimiento de tickets abiertos o *retrollamadas* planificadas. De este modo, se evita recurrir a jornadas partidas que podrían afectar negativamente a la conciliación laboral.

El diagrama de abajo desglosa además el volumen horario por tipo de llamada. Se observa que las **consultas técnicas** representan el mayor volumen en todas las franjas, especialmente entre las 9:00 y las 13:00 h. Por otro lado, las llamadas de tipo **comercial** tienen una presencia más moderada y estable a lo largo del día.

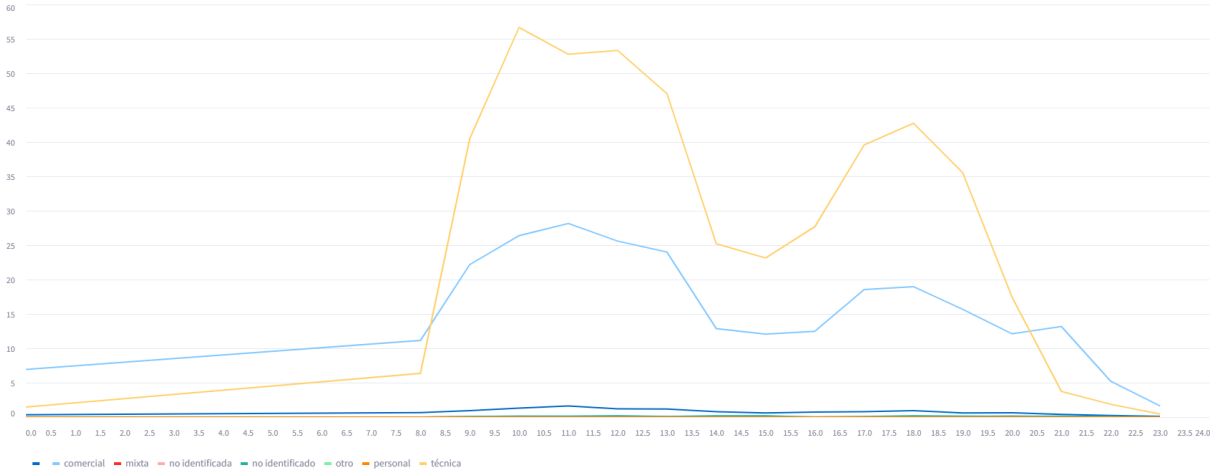


Ilustración 67:Distribución horaria por tipo de llamada

Este análisis tiene un valor organizativo clave, ya que permite ajustar con precisión la composición del equipo por perfil (técnico vs. comercial) en cada franja horaria. Por ejemplo, aunque actualmente la plantilla está compuesta por 13 agentes técnicos y 12 comerciales, esta ligera diferencia queda justificada por la mayor carga de trabajo técnico en horas críticas y por el hecho de que las llamadas técnicas presentan una duración media significativamente superior a las comerciales, según los datos operativos facilitados por la responsable del área.

Además, muchos agentes comerciales han sido formados específicamente en la resolución de incidencias técnicas comunes, lo que les permite apoyar al equipo técnico en momentos puntuales de saturación, especialmente cuando se producen picos de incidencias o averías.

Cabe añadir que una parte importante de las llamadas comerciales entrantes no finaliza con una contratación telefónica, ya que muchos usuarios prefieren formalizar estos trámites presencialmente en las oficinas. Por ello, optimizar la asignación de personal según el tipo de consulta y el comportamiento del cliente puede mejorar notablemente tanto la eficiencia operativa como la calidad del servicio percibido.

6.7 ANÁLISIS DE CONSULTAS INTELIGENTES CON GPT

Este análisis permite interpretar preguntas formuladas en lenguaje natural y traducirlas automáticamente a filtros estructurados aplicados sobre la base de datos real de llamadas. Esta capacidad ha sido posible gracias a la integración de un modelo GPT adaptado al dominio específico.

El sistema responde a preguntas directas como *"¿Qué agente recibió más llamadas en Elda?"* o *"¿Cuántas llamadas comerciales hay?"*, generando automáticamente los criterios de filtrado y devolviendo una respuesta basada en los datos reales disponibles.

A continuación, se describen algunos ejemplos representativos extraídos del sistema:

- En la consulta *"¿Qué agente recibió más llamadas en Elda?"*, el modelo interpretó correctamente la acción como una búsqueda de máximo y filtró por localidad igual a *Elda*. La respuesta identificó al agente con mayor volumen de llamadas, acompañado de la cifra total (6.631 llamadas).

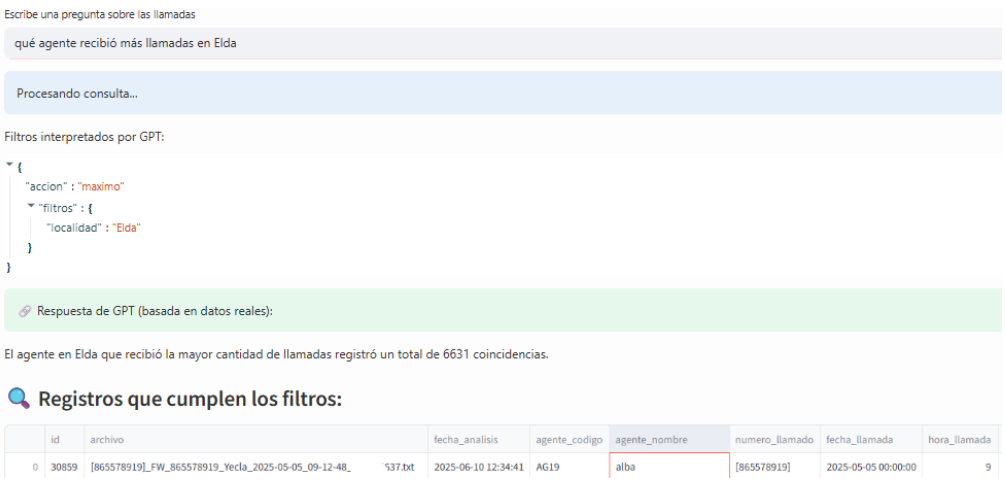


Ilustración 68: Consulta (1) GPT

Después de realizar esta consulta se abre el detalle de todas las llamadas que ha hecho el agente.

- En la consulta *"¿Cuántas llamadas comerciales hay?"*, el modelo reconoció la intención de contar llamadas con el campo tipo_llamada igual a *comercial*. El resultado arrojó un total de 9.605 coincidencias.

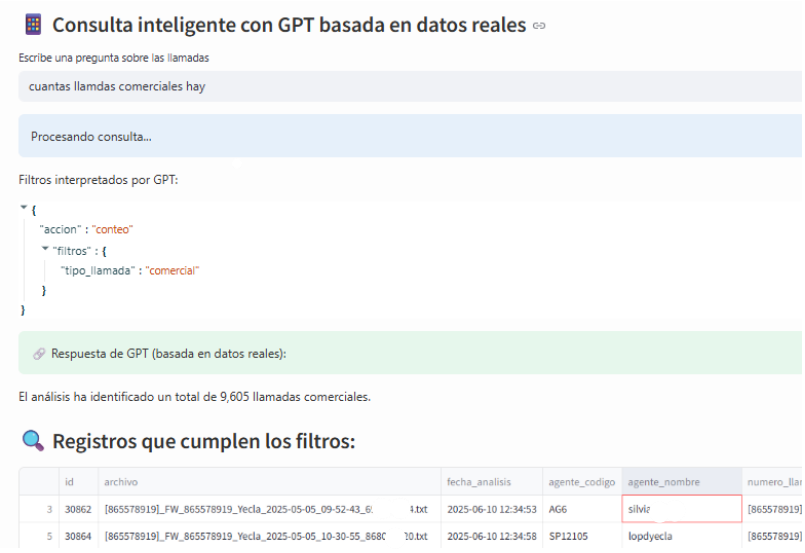


Ilustración 69: Consulta (2) GPT

- En la pregunta *"¿Cuántas llamadas positivas hubo en Monóvar?"*, se aplicaron correctamente filtros de opinión positiva y localidad Monóvar, con un resultado de 127 llamadas positivas.

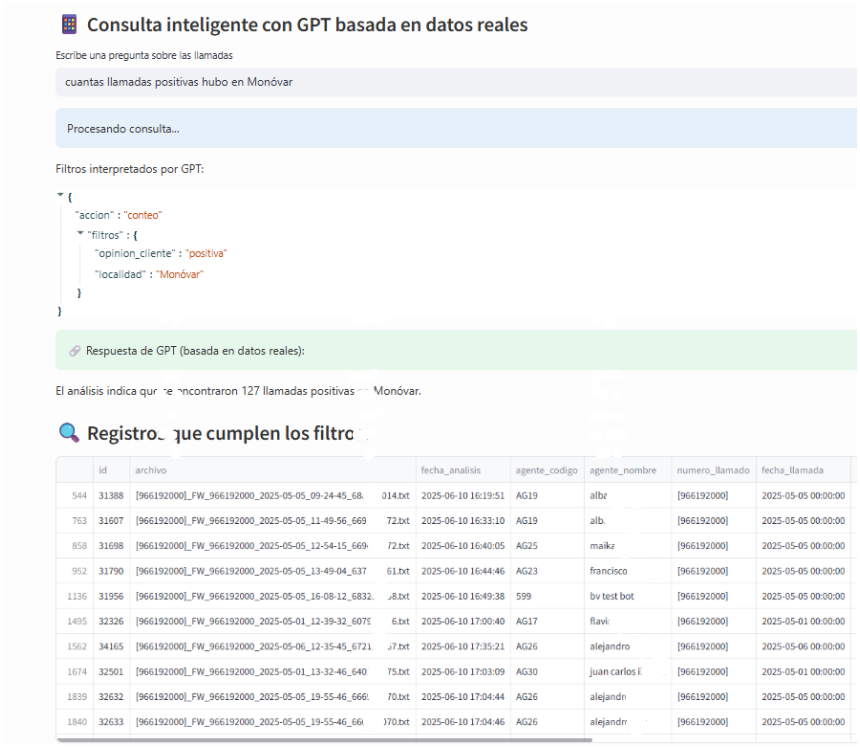


Ilustración 70: Consulta (3) GPT

- Finalmente, al consultar por las llamadas del *29 de abril de 2025*, el modelo devolvió los registros correspondientes a esa fecha, con un total de 2.065 coincidencias.

Ilustración 71: Consulta (4) GPT

Estos resultados evidencian la capacidad del sistema para comprender consultas naturales, traducirlas a lógica computacional y generar respuestas precisas, facilitando el análisis sin necesidad de conocimientos técnicos avanzados por parte del usuario final.

La interfaz también permite visualizar directamente los registros filtrados, lo que contribuye a la trazabilidad y validación de los resultados ofrecidos por la inteligencia artificial.

6.8 EJEMPLO DE ANÁLISIS DETALLADO

6.8.1 SEGUIMIENTO DE REINCIDENCIA DE UN ABONADO

Entonces copio el número de abonado en el panel de filtros y veo que tipo de consultas ha tenido y el grado de atención que se le ha dado, ya que puede ser por una mala gestión de agentes

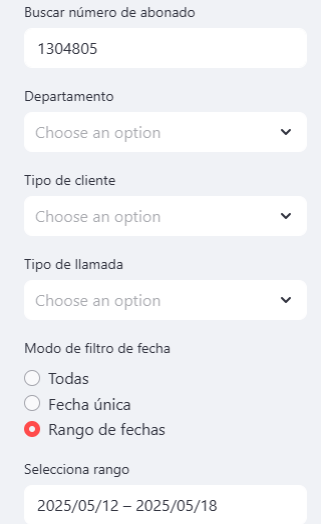


Ilustración 72:Ejemplo aplicamos filtro por número de abonado

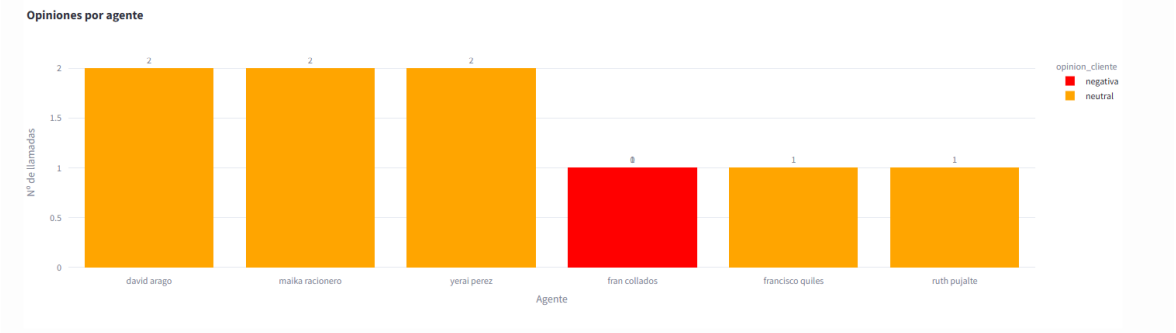


Ilustración 73:Resultados a los filtros aplicados en opiniones por agente

Aquí vemos que las consultas en todos los agentes se han realizado de manera neutral menos en un agente que ha recibido una valoración negativa. También vemos que no hay ninguna positiva. Así que, si queremos indagar más, con los mismos filtros que hemos puesto se puede ver el detalle de llamadas.

6.8.2 SEGUIMIENTO DE UNA INCIDENCIA EN GUARDIA

Una de las funcionalidades clave del panel desarrollado es la capacidad de aplicar filtros avanzados para identificar y analizar eventos concretos, como picos de incidencias o anomalías detectadas en jornadas específicas. A continuación, se presenta un caso real que demuestra cómo la plataforma permite realizar un seguimiento preciso de una incidencia ocurrida durante una guardia.

Identificación de la guardia con mayor incidencia

1. Se aplica el filtro en el panel interactivo para seleccionar exclusivamente las llamadas gestionadas por agentes de guardia:

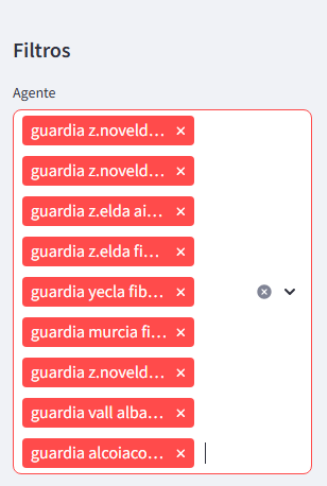


Ilustración 74:Filtro para seleccionar todas las guardias

2.En el mapa de calor de llamadas por agente y fecha, se observa que el **2 de mayo** presenta una concentración anómala de incidencias:

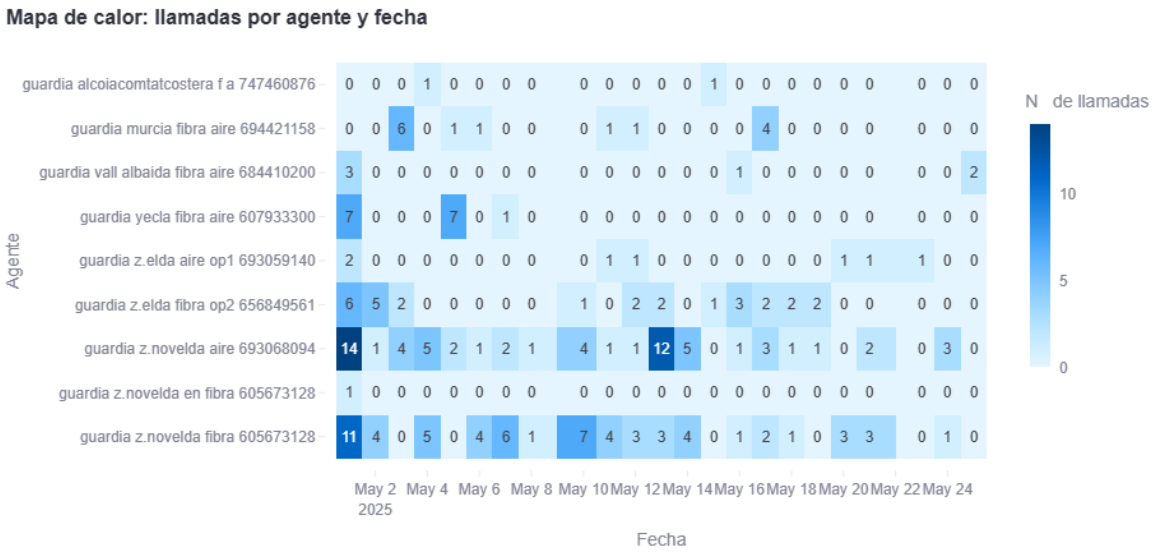


Ilustración 75:Mapa de calor llamadas por agente y fecha

3.Para investigar el origen del problema, se selecciona manualmente la **fecha del 1 de mayo**, anterior al pico, desde el selector de fechas del panel:

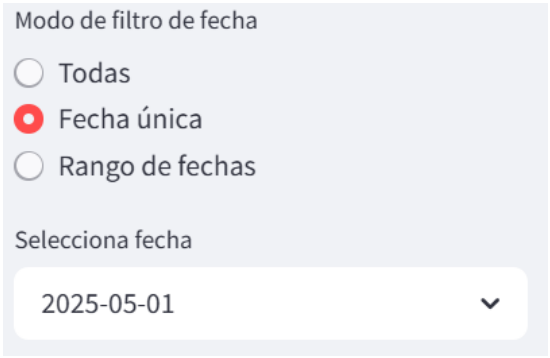


Ilustración 76:Selección de fecha en el panel interactivo

- El panel muestra en detalle **todas las llamadas de guardia registradas ese día**, facilitando el análisis por agente, tipo de llamada, etiquetas y opinión del cliente.

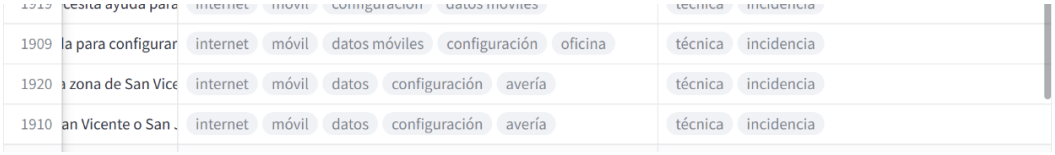


Ilustración 77:Panel con todos los detalles de la llamada (solo se han mostrado las etiquetas)

Causa detectada

Este comportamiento se atribuyó a una consecuencia posterior al **apagón masivo del 28 de abril**, el cual afectó a diferentes zonas del país. Aunque se realizaron reajustes internos y cambios operativos durante los días siguientes, se comprobó que algunas zonas no se habían reseteado correctamente, provocando fallos residuales de conectividad móvil.

Es importante destacar que el análisis se centró a partir del 1 de mayo, ya que previamente se modificaron los criterios de anotación utilizados por los agentes, lo cual podría haber afectado a la homogeneidad de los datos. Aun así, este caso ilustra cómo el sistema permite detectar fallos persistentes tras eventos externos y facilita el cruce entre volumen de llamadas, localidad afectada y tipo de problema reportado.

Conclusión

Este ejemplo demuestra la utilidad real de la plataforma como herramienta de supervisión operativa. Permite no solo reaccionar ante incidencias, sino anticipar su impacto y comprobar su evolución. Además, los filtros por fecha, localidad, etiquetas y tipo de llamada brindan una base sólida para la toma de decisiones informadas por parte de los responsables del servicio.

6.9 ANOMALÍA POR CAÍDA DE SERVICIO EN EL PAÍS (DÍA DEL APAGÓN)

El **lunes 28 de abril de 2024**, [41] se produjo un apagón eléctrico generalizado en diversas regiones de España, lo que afectó gravemente la capacidad de los abonados para contactar con el servicio de atención telefónica. Este evento provocó una **caída abrupta en el volumen de llamadas**, interrumpiendo el patrón regular de distribución horaria que presentan los lunes ordinarios.

En la **Ilustración 78,** correspondiente a un lunes habitual del mes de mayo, se observa una distribución constante y coherente con la dinámica diaria del servicio: un incremento progresivo desde primeras horas de la mañana, con picos entre las **9:00 y las 13:00**, y una disminución paulatina por la tarde.

Sin embargo, en la **Ilustración 79,** correspondiente al 28 de abril, se aprecia una **anomalía muy marcada**: tras un inicio aparentemente normal, el volumen de llamadas cae drásticamente a partir de las **13:00 h**, manteniéndose en niveles mínimos hasta el cierre del servicio. Este comportamiento, totalmente atípico, fue provocado por la interrupción del suministro eléctrico que impidió a muchos clientes realizar llamadas, y también afectó la operatividad de algunas instalaciones.

Para evitar distorsiones en los indicadores generales, se ha aplicado un **tratamiento específico** a esta fecha:

- Ha sido **excluida de los promedios semanales y horarios**, para no alterar las medias representativas.

- Se ha **etiquetado como día atípico** en el sistema, conservando la información para fines de control de calidad y detección de eventos.

Este caso demuestra la utilidad del sistema desarrollado no solo para detectar tendencias y patrones, sino también para **identificar y aislar eventos exógenos** que alteran significativamente el comportamiento del servicio.

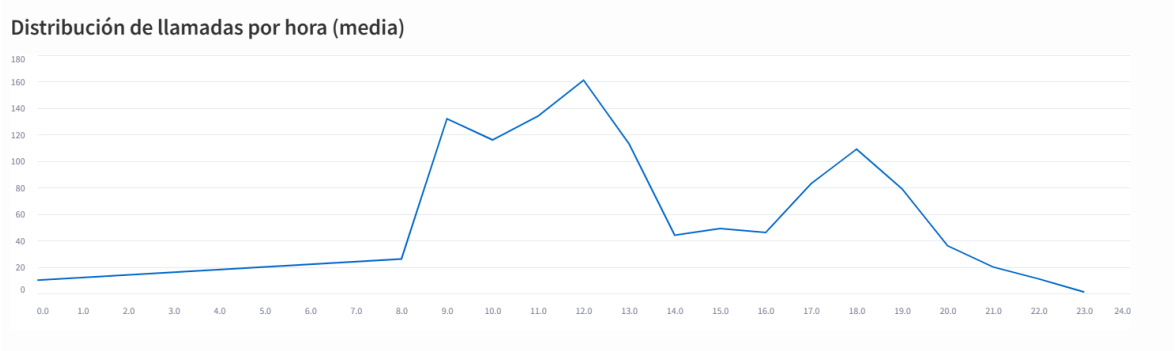


Ilustración 78:Distribución horaria de llamadas en un lunes nomal (mayo 2025)

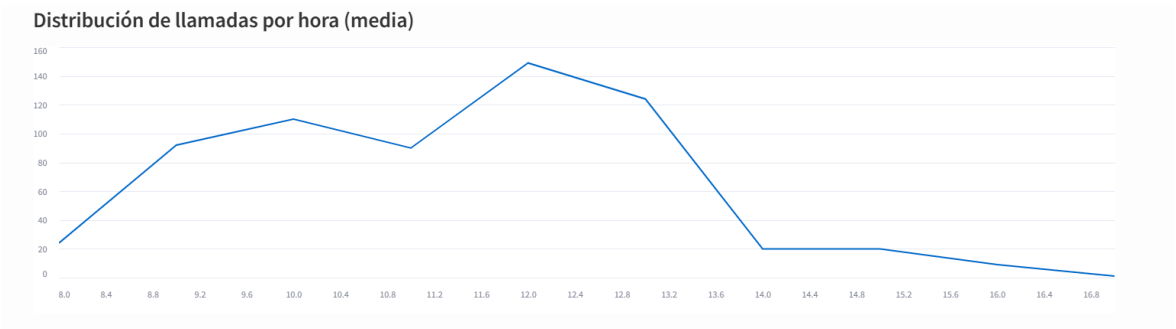


Ilustración 79:Distribución horaria de llamadas el lunes 28 de abril de 2025

Descenso abrupto de llamadas a partir del mediodía debido al apagón nacional.

6.9.1 GESTIÓN DE INCIDENCIAS TRAS EL APAGÓN (29 DE ABRIL)

Dentro del patrón semanal descrito, destaca un caso excepcional ocurrido el **martes 29 de abril de 2024**, cuando se registró un volumen de llamadas anormalmente elevado como consecuencia directa del **apagón nacional** que tuvo lugar el día anterior.

Este fenómeno se explica por la **acumulación de incidencias no reportadas durante el lunes**, ya que a partir de las 12:00 h del 28 de abril, gran parte del país incluida la zona operativa de cableworld, sufrió un corte generalizado de suministro eléctrico. Aunque las instalaciones de cableworld permanecieron operativas gracias a generadores auxiliares, los clientes **no disponían de electricidad ni conectividad**, lo que imposibilitó el contacto telefónico o el uso del servicio de internet.

Una vez restablecido el suministro en la mayoría de zonas, el martes 29 se produjo una **sobrecarga significativa de llamadas**, especialmente a partir de las 9:00 h. Esta situación afectó de forma particular al equipo técnico, que recibió múltiples consultas relacionadas con el restablecimiento de servicios.

Según el análisis de los **resúmenes generados automáticamente** por el sistema de transcripción y análisis semántico, se identificaron **276 llamadas** con menciones explícitas al término *"apagón"*, reflejando preocupaciones como pérdida de señal de televisión, desconexión del *router* o dudas sobre la recuperación completa del servicio.

Este episodio subraya la **importancia de contar con herramientas analíticas** capaces de identificar eventos correlacionados en el tiempo y priorizar automáticamente la atención en función del motivo de consulta. En este caso, la detección de palabras clave como *"apagón"*,

"recuperación" o *"antenas"* permitió segmentar rápidamente las incidencias relacionadas con el evento crítico y mejorar la respuesta operativa.

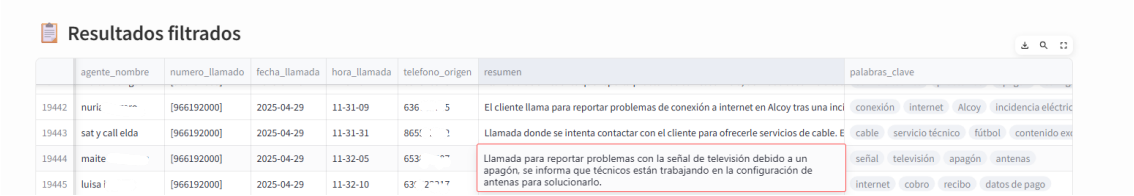


Ilustración 80:Resumen de una llamada al buscar apagón en el buscador

Ilustración 81:276 resultados al buscar la palabra apagón el 29 de abril

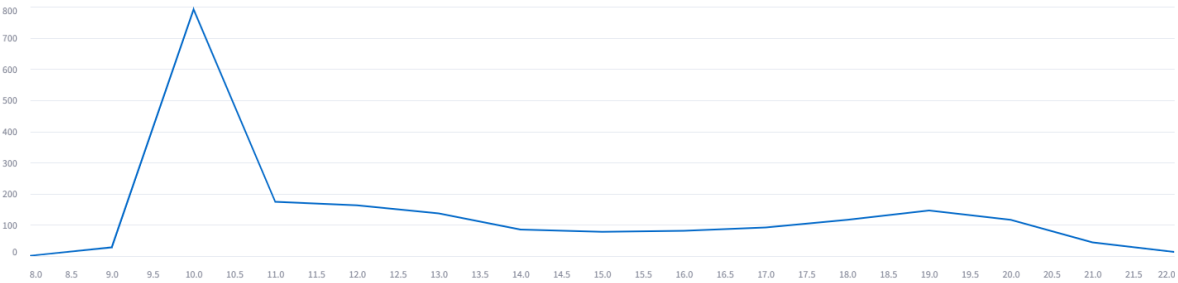


Ilustración 82: Distribuición horaria de llamadas el 29 de abril de 2025

Se observa una concentración elevada de incidencias con mención explícita al apagón. Aumento de volumen a primera hora del día debido al efecto acumulado de incidencias no resueltas.

El impacto operativo también se refleja en la **asignación de carga entre agentes y servicios técnicos**. El mapa de calor de llamadas por agente y fecha, muestra que **el servicio técnico general Sat y Call elda gestionó más de 500 incidencias en un solo día**, evidenciando el **papel clave de este recurso compartido** para absorber la carga generada por el apagón.

Este tipo de agrupaciones permite redirigir llamadas automáticamente a personal técnico disponible, lo que resultó determinante para **restablecer el servicio en condiciones críticas**.

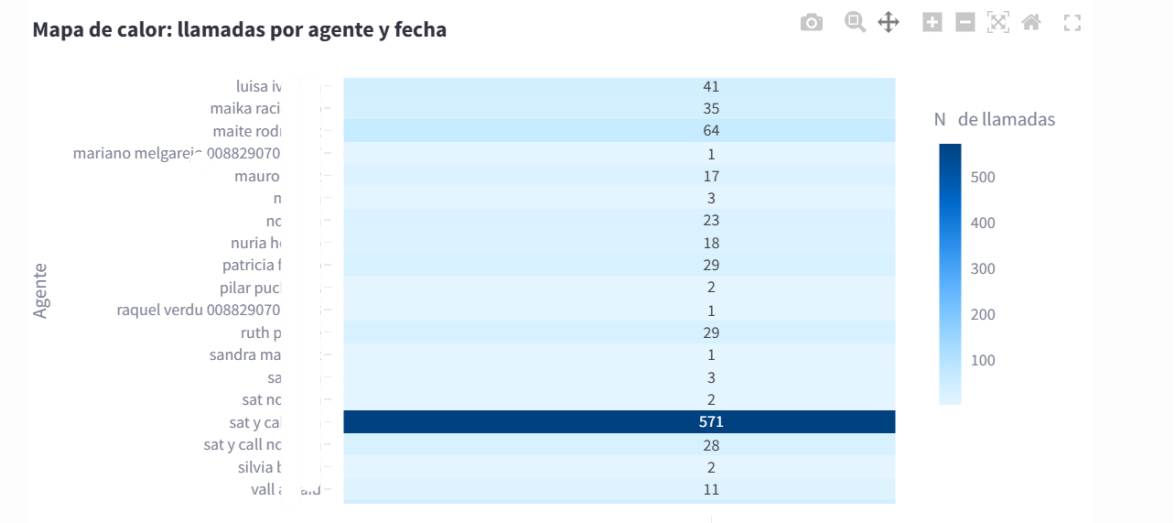


Ilustración 83:Mapa de calor de volumen de agentes y llamadas

En conjunto, el volumen de llamadas de esa jornada fue **muy superior al habitual**, como se aprecia en la evolución de llamadas por centralita. Si bien algunas llamadas fueron atendidas directamente por agentes, otras fueron redirigidas o almacenadas temporalmente en la **centralita EXTRA**, que agrupa llamadas en espera o fuera del flujo principal. Esta agrupación refleja el uso puntual de recursos alternativos para gestionar una situación crítica, aunque el **dato clave es el incremento total de la demanda** en todas las áreas.

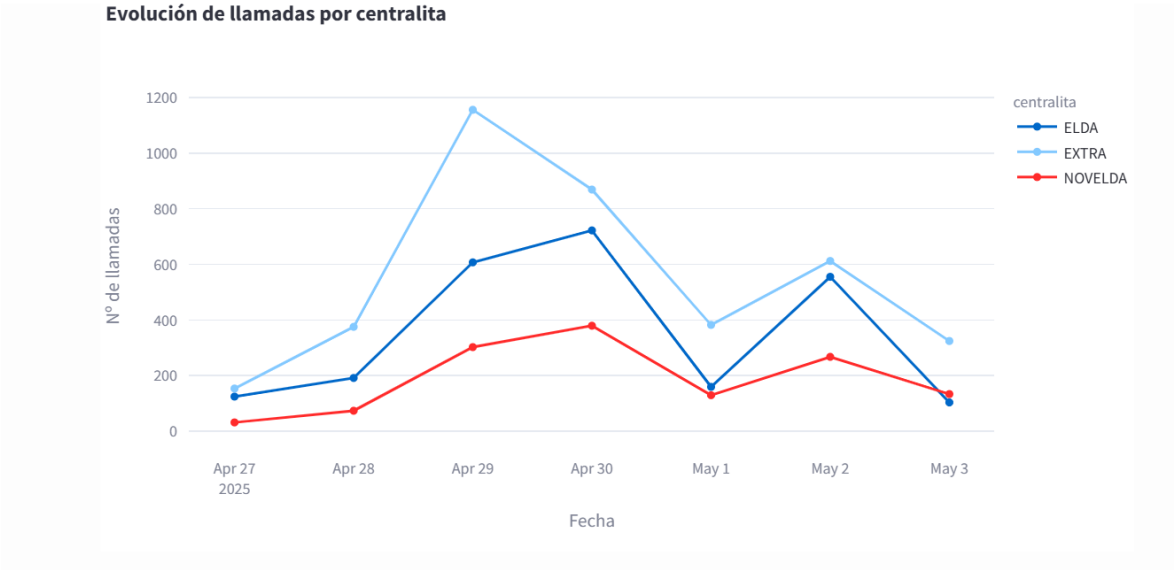


Ilustración 84: Evolución de llamadas por centralita (del 27 de abril al 3 de mayo)

Capítulo 7. CONCLUSIONES Y TRABAJOS FUTUROS

7.1 VALOR APORTADO AL NEGOCIO

La implementación de este sistema ha supuesto un punto de inflexión en la gestión operativa del servicio de atención telefónica. Se ha conseguido transformar más de 22.000 llamadas en información estructurada y visualmente interpretable, permitiendo detectar patrones, identificar incidencias en tiempo real y optimizar la distribución de carga entre agentes y centralitas.

A diferencia del enfoque anterior, donde la supervisión se realizaba de forma manual mediante hojas de cálculo (Excel) y comprobaciones puntuales, el nuevo sistema automatizado permite que cada día se incorporen de forma autónoma nuevas llamadas al panel de análisis. Gracias a este proyecto, la supervisión es continua, sin necesidad de intervención diaria, y con un nivel de detalle y fiabilidad mucho mayor que el de los métodos tradicionales.

Esto ha permitido al equipo supervisor:

- Detectar sobrecargas y ajustar turnos de trabajo con antelación.
- Reforzar la atención en días y franjas críticas (guardias, lunes, horarios pico).
- Reasignar agentes entre departamentos (por ejemplo, de comercial a) según el volumen real.
- Segmentar adecuadamente las llamadas entre SAC (atención comercial) y SAT (soporte técnico).
- Evaluar la calidad del servicio con base en opiniones y reincidencias extraídas automáticamente.

Todo ello ha mejorado notablemente la eficiencia del centro de atención al cliente, permitiendo un seguimiento diario y escalable de las operaciones. Además, la posibilidad de aplicar filtros en el panel interactivo otorga a los responsables la capacidad de obtener *insights* personalizados en segundos, mejorando la toma de decisiones en todos los niveles operativos.

7.2 LIMITACIONES

Pese a los avances logrados, el sistema presenta aún algunas limitaciones relevantes:

- **Dependencia del comportamiento humano**: la ausencia de información sobre la localidad en ciertas llamadas, especialmente durante guardias o en interacciones automatizadas con *bots*, provoca registros sin clasificar. Actualmente se está trabajando en la formación de los agentes para que insistan en preguntar y confirmar la localidad del llamante cuando no se indica de forma espontánea.
- **Cobertura parcial del flujo de llamadas**: el sistema analiza exclusivamente las llamadas entrantes. Las llamadas salientes, especialmente importantes en el ámbito comercial, no están todavía integradas, ya que la API de Enreach no permite su descarga. Se han iniciado gestiones para obtener los permisos necesarios y ampliar el análisis.
- **Ambigüedad en las opiniones neutras**: muchas opiniones clasificadas como neutras no reflejan desinterés o neutralidad real, sino situaciones en las que la respuesta al cliente ha quedado pospuesta. Esta ambigüedad complica la interpretación automática de estos casos.
- **Falta de alertas automáticas**: el sistema aún no dispone de mecanismos de notificación en tiempo real para advertir sobre incidencias o picos de carga anómalos. La incorporación de esta funcionalidad permitiría una reacción más ágil del equipo supervisor.
- **Resultados no auditados en tiempo real**: aunque el sistema realiza un análisis semántico automatizado de gran valor, no se ha implementado aún una validación

humana para los casos críticos. Esto implica que, en ciertas situaciones, puede ser necesario revisar manualmente los resultados para confirmar su precisión.

7.3 LÍNEAS FUTURAS Y PROYECCIÓN ESTRATÉGICA

El sistema desarrollado sienta una base sólida para evolucionar hacia un modelo de supervisión operativa inteligente, adaptable y escalable. A corto plazo, se prevé la incorporación de llamadas salientes al flujo de análisis, así como la integración de alertas automáticas que permitan detectar incidencias críticas en tiempo real. Del mismo modo, se contempla implementar un sistema de validación manual para aquellos casos donde la ambigüedad semántica pueda comprometer la precisión del análisis.

En un horizonte más amplio, el sistema podría integrarse con el **ERP** corporativo, lo que permitiría enriquecer las transcripciones con información adicional sobre clientes, servicios contratados o histórico de interacciones. Esta conexión habilitaría nuevas posibilidades como la segmentación avanzada, la detección de patrones de abandono o el análisis proactivo de necesidades, todo ello sin intervención manual.

Desde una perspectiva estratégica, esta herramienta deja de ser una simple solución técnica para convertirse en un **activo clave de transformación digital**. Permite a la empresa disponer de indicadores objetivos, trazables y automatizados sobre el funcionamiento del *Call center*, facilitando una **toma de decisiones más rápida, fundamentada y escalable**. Además, el sistema resulta fácilmente extrapolable a otros departamentos o incluso a otras empresas del sector, especialmente aquellas que gestionan grandes volúmenes de interacciones no estructuradas.

Finalmente, el proyecto se alinea con los objetivos de innovación y eficiencia tecnológica recogidos en la Agenda 2030. La **contribución a los Objetivos de Desarrollo Sostenible (ODS)** se detalla en el anexo I

Capítulo 8. BIBLIOGRAFÍA

[1] Enreach. (2023). *Enreach API Documentation*. https://developer.enreach.com/

[2] Alpha Cephei. (2024). *Vosk Speech Recognition Toolkit*. https://alphacephei.com/vosk/server

[3] Toolify. (2024). *Cómo usar Vosk: la biblioteca de reconocimiento de voz sin conexión para Python*. https://www.toolify.ai/es/ai-news-es/cmo-usar-vosk-la-biblioteca-de-reconocimiento-de-voz-sin-conexin-para-python-622064

[4] OpenAI. *Introduction to the Chat Completions API*. https://platform.openai.com/docs/guides/chat

[5] OpenAI. (2023). *GPT-3.5 Turbo overview*. https://platform.openai.com/docs/models/gpt-3-5

[6] OpenAI. (2024). *GPT-4 overview*. OpenAI. https://platform.openai.com/docs/models/gpt-4

[7] OpenAI. (2023). *GPT-4 Technical Report*. https://openai.com/research/gpt-4

[8] Jurafsky, D., & Martin, J. H. (2021). *Speech and Language Processing* (3rd ed.). https://web.stanford.edu/~jurafsky/slp3/

[9] Nemeth, E., Snyder, G., Hein, T. R., Whaley, B., & Mackin, D. "UNIX and Linux System Administration Handbook (5th ed.)". Pearson, 2017.

[10] PostgreSQL Global Development Group. (2023). *PostgreSQL Documentation*. https://www.postgresql.org/docs/

[11] IBM. (2023). *What is PostgreSQL?*. https://www.ibm.com/docs/en/cloud-paks/cp-data/4.0?topic=databases-postgresql

[12] Python Software Foundation. *multiprocessing — Process-based parallelism*. Python 3.12 documentation. https://docs.python.org/3/library/multiprocessing.html

[13] FFmpeg. *FFmpeg Documentation*. https://ffmpeg.org/documentation.html

[14] Smith, C. (2019). *vsftpd: Secure and Fast FTP Server for Linux*. Linux Journal. https://www.linuxjournal.com/content/vsftpd-secure-and-fast-ftp-server-linux

[15] Canonical. (2023). *Community Help Wiki – vsftpd*. https://help.ubuntu.com/community/vsftpd

[16] Postman. *API platform for building and using APIs*. Recuperado de: https://www.postman.com/

[17] McKinney, W. (2010). *Data Structures for Statistical Computing in Python*. Proceedings of the 9th Python in Science Conference, 51–56. https://pandas.pydata.org/

[18] The PostgreSQL Global Development Group. *psycopg2 – PostgreSQL database adapter for Python*. https://www.psycopg.org/

[19] Python Software Foundation. *os — Miscellaneous operating system interfaces*. En *Python 3.12.3 documentation*. https://docs.python.org/3/library/os.html

[20] Python Software Foundation. *csv — CSV File Reading and Writing*. https://docs.python.org/3/library/csv.html

[21] Python Software Foundation. *datetime — Basic date and time types*. En *Python 3.12.3 documentation*. https://docs.python.org/3/library/datetime.html
Python Software Foundation. *time — Time access and conversions*. https://docs.python.org/3/library/time.html

[22] Python Software Foundation. *re — Regular expression operations*. En *Python 3.12.3 documentation*. https://docs.python.org/3/library/re.html

[23] Streamlit Inc. *Streamlit — The fastest way to build and share data apps*. https://streamlit.io/

[24] Ngrok. *Ngrok — Secure introspectable tunnels to localhost*. https://ngrok.com/

[25] Esquema Nacional de Seguridad. (2022). *Guía CCN-STIC 825 - Requisitos del ENS: Implementación de medidas de seguridad en sistemas de información*. Centro Criptológico Nacional (CCN-CERT). https://www.ccn-cert.cni.es/es/

[26] Google Cloud. (2024). *Speech-to-Text API Documentation*. https://cloud.google.com/speech-to-text

[27] Amazon Web Services. *Amazon Transcribe – Automatic Speech Recognition*. https://aws.amazon.com/transcribe/

[28] Upbe. (2024). *Product Overview*. https://www.upbe.ai/

[29] Ringover. (2024). *Ringover Product Overview*. https://www.ringover.com/

[30] Amazon Web Services. (2024). *Amazon Contact Lens for Amazon Connect*. https://aws.amazon.com/connect/contact-lens/

[31] Google Cloud. *Contact Center AI*. https://cloud.google.com/contact-center-ai

[32] Microsoft Azure. *Speech to Text – Azure Cognitive Services*. https://azure.microsoft.com/en-us/products/cognitive-services/speech-to-text/

[33] OpenAI. *.Pricing for GPT models*. https://openai.com/pricing

[34] Google Cloud. *Speech-to-Text API Pricing*. https://cloud.google.com/speech-to-text/pricing

[35] AWS.*Amazon Transcribe Pricing*. https://aws.amazon.com/transcribe/pricing

[36] Google. (2024).*Gemini – Google AI Studio*. Disponible en: https://aistudio.google.com/welcome

[37] Masvoz API. (2025). *Call Statistics API*. https://manager.masvoz.es/api/rs/call/stats/ATC

[38] Instituto Nacional de Ciberseguridad (INCIBE). (2024). *Recomendaciones para la gestión segura de contraseñas*. Recuperado de https://www.incibe.es

[39] OWASP Foundation. (2023). *Authentication Cheat Sheet*. Open Web Application Security Project. https://cheatsheetseries.owasp.org/cheatsheets/Authentication_Cheat_Sheet.html

[40] NIST (National Institute of Standards and Technology). (2017). *Digital Identity Guidelines: Authentication and Lifecycle Management (SP 800-63B)*. Recuperado de https://pages.nist.gov/800-63-3/sp800-63b.html

[41] Lombardi, P., & Latona, D. (2025, junio 17). *Miscalculation by Spanish power grid operator REE contributed to massive blackout*. Reuters. https://www.reuters.com/business/energy/investigation-into-spains-april-28-blackout-shows-no-evidence-cyberattack

ANEXOS

ANEXO I :ALINEACIÓN DEL PROYECTO CON LOS ODS

El presente proyecto se alinea con los Objetivos de Desarrollo Sostenible (ODS) definidos por las Naciones Unidas en la Agenda 2030, en tanto que promueve un uso responsable y estratégico de la tecnología para mejorar la eficiencia operativa, la toma de decisiones basada en datos y la sostenibilidad del servicio de atención al cliente en el sector de las telecomunicaciones.

A través de la automatización inteligente del análisis de llamadas, el uso de modelos de IA, la implementación de procesos seguros y la optimización de recursos humanos y técnicos, este sistema contribuye de manera concreta y medible a los siguientes ODS:

ODS 9. Industria, Innovación e Infraestructura

El sistema desarrollado representa un ejemplo claro de innovación tecnológica aplicada a procesos industriales, al reemplazar procedimientos manuales (como el análisis de llamadas con hojas de cálculo) por un flujo automatizado, basado en tecnologías de vanguardia como la inteligencia artificial, el procesamiento del lenguaje natural (NLP), la programación paralela y la visualización web interactiva.

Esta modernización permite no solo mejorar la infraestructura digital interna de la empresa, sino también garantizar la resiliencia de los servicios críticos, como la atención técnica durante emergencias. Casos como la gestión del volumen extraordinario de llamadas tras el apagón del 29 de abril son ejemplo de cómo una infraestructura inteligente puede adaptarse, responder y escalar ante situaciones imprevistas.

ODS 11. Ciudades y Comunidades Sostenibles

El proyecto también contribuye a la sostenibilidad urbana y comunitaria al mejorar la capacidad de respuesta del sistema de atención técnica y comercial, especialmente en localidades donde los servicios de telecomunicaciones son esenciales para el acceso a la información, la educación, la seguridad y la conectividad social.

Mediante el análisis automatizado de más de 22.000 llamadas, el sistema ha permitido identificar de forma proactiva problemas recurrentes por zona, reorganizar la carga entre agentes y centralitas, y mejorar la eficiencia del soporte técnico. Esto se traduce en una mejora directa de la calidad del servicio percibido por el usuario final y en una reducción de los tiempos de resolución de incidencias, elementos fundamentales para el desarrollo de comunidades más resilientes y cohesionadas.

ODS 16. Paz, Justicia e Instituciones Sólidas

Desde el punto de vista institucional, el sistema refuerza principios clave de buena gobernanza tecnológica al incorporar mecanismos de seguridad, trazabilidad y transparencia. El uso de contraseñas cifradas, la autenticación por usuario y el control de accesos diferenciados por rol permiten proteger la integridad de la información sensible, al tiempo que se garantiza la responsabilidad en el uso de los datos procesados.

Además, al centralizar todos los resultados en una base de datos estructurada, accesible únicamente para personal autorizado, se promueve un modelo operativo basado en datos auditables, decisiones objetivas y procesos documentados, en línea con los valores que promueve el ODS 16.

Este proyecto no solo cumple su función técnica dentro de un contexto empresarial, sino que demuestra cómo una solución tecnológica diseñada con criterios de eficiencia, escalabilidad y ética puede contribuir activamente a los retos sociales y ambientales globales. La posibilidad de replicar esta arquitectura en otros contextos —como servicios de emergencia, atención médica, administración pública o empresas con grandes volúmenes de interacción ,abre nuevas vías para extender su impacto y seguir promoviendo un desarrollo sostenible, justo y digitalmente inteligente.

ANEXO II: COMPARATIVA ENTRE IAS PARA LA TRANSCRIPCIÓN DE AUDIOS

1.INTRODUCCIÓN.

En este documento se analiza el rendimiento de diferentes modelos de inteligencia artificial en la transcripción de audios, evaluando su precisión, capacidad de diarización (identificación de hablantes) y calidad general de la transcripción. Se han realizado pruebas con **Google Cloud Speech-to-Text, Amazon Transcribe, Azure Speech-to-Text y Gemini de Google**, comparando los resultados obtenidos en distintos audios.

El objetivo es determinar cuál de estas herramientas ofrece la mejor combinación de **precisión, segmentación de hablantes y fidelidad al audio original**, con base en diferentes pruebas realizadas con audios reales. Para ello, se han identificado las principales fortalezas y debilidades de cada servicio, incluyendo problemas detectados como omisión de fragmentos, errores en la diarización o dificultades en la interpretación de ciertos términos.

En primer lugar voy a analizar paso a paso cada transcripción y la diferenciación entre los diferentes hablantes.

2. PRUEBAS

2.1 prueba1.wav (centralita vieja)

1. *AWS*

{"jobName":"transcripcion-prueba1","accountId":"120569625568","status":"COMPLETED","results":{"transcripts":[{"transcript":"a día diez de febrero del dos mil veinticinco procedemos a realizar la grabación de la llamada para la baja del contrato número uno tres cero cero dos cero tres. Dígame, por favor el nombre completo y el DNI Juan Luís. Mhm. El DNI** N de Navarra. Vale, Juan Luis,

actualmente tiene contratado el servicio Fibra de trescientos megas,

por el cual Sol paga veinticuatro con noventa y nueve del cual nos

solicita la baja total. Está de acuerdo? De acuerdo. Vale.

Le informamos que la baja será efectiva el día veintiocho de

febrero del dos mil veinticinco, fecha en la que serán desactivados.

Recuerde que los equipos son en sesión. Tiene que entregarlos en

cualquiera de nuestras oficinas o conservarlos para su recogida.

De acuerdo, Juan Luis. Vale, de acuerdo. Vale, pues ya estaría todo.

Juan Luis, finalizamos la grabación."}],

*Está subrayado las partes en las que luego hablaré

2. Google Cloud

"transcript": "a día 10 de febrero del 2025 procedemos a realizar la grabación de la llamada para la baja del contrato número 1300203 dígame por favor el nombre completo y el DNI Juan Luis Burguete Martínez el DNI **** de Navarra vale Juan Luis actualmente tiene contratado el servicio fibra de 300 megas por el cual paga 24.99 del cual nos solicita la baja total está de acuerdo de acuerdo del 2025 fecha en la que serán desactivados recuerde que los equipos son en sesión tiene que entregarlos en cualquiera de nuestras oficinas o conservarlos para su recogida de acuerdo cuando vale de acuerdo vale pues ya estaría todo Juan Luis finalizamos la grabación"

3. Whisper

beatriz@TFGBea:~/ftp/download/2025-02-05/d926d087-8790-4e0a-b350-4a822d81bdc2/extracted_files/20250205/converted_wavs$ cat prueba1_whisper.txt

[00:00.000 --> 00:11.500] A día 10 de febrero del 2025 procedemos a realizar la grabación de la llamada para la baja del contrato número 1300203.

[00:11.840 --> 00:14.160] Dígame, por favor, el nombre completo y el DNI.

[00:14.760 --> 00:23.700] Juan Luis. El DNI es 8…. de Navarra.

[00:23.700 --> 00:32.940] Vale, Juan Luis. Actualmente tiene contratado el servicio fibra de 300 megas, por el cual paga 24,99, del cual nos solicita la baja total.

[00:33.380 --> 00:34.020] ¿Está de acuerdo?

[00:34.580 --> 00:35.100] De acuerdo.

[00:35.700 --> 00:42.060] Vale, le informamos que la baja será efectiva el día 28 de febrero del 2025, fecha en la que serán desactivados.

[00:42.560 --> 00:48.020] Recuerde que los equipos son en cesión, tiene que entregarlos en cualquiera de nuestras oficinas o conservarlos para su recogida.

[00:48.100 --> 00:48.800] ¿De acuerdo, Juan?

[00:48.840 --> 00:49.740] Vale, de acuerdo.

[00:50.180 --> 00:52.820] Vale, pues ya estaría todo, Juan Luis. Finalizamos la grabación.

// y ahora, si identifica speakers

00:01.870 --> 00:11.469

[SPEAKER_02]: A día 10 de febrero del 2025 procedemos a realizar la grabación de la llamada para la baja del contrato número 1-3-00**.

00:12.171 --> 00:14.215

[SPEAKER_02]: Dígame por favor el nombre completo y el DNI.

00:15.157 --> 00:16.980

[SPEAKER_00]: Juan Luis.

00:18.022 --> 00:23.794

[SPEAKER_00]: El DNI 10 es 85….. de Navarra.

00:24.736 --> 00:30.962

[SPEAKER_02]: Vale, Juan Luis, actualmente tiene contratado el servicio fibra de 300 megas por el cual paga 24,99 y el cual nos solicita la baja total.

00:30.982 --> 00:31.763

[SPEAKER_02]: ¿Está de acuerdo?

00:31.803 --> 00:32.204

[SPEAKER_02]: De acuerdo.

00:32.224 --> 00:42.034

[SPEAKER_02]: Vale, le informamos que la baja será efectiva el día 28 de febrero del 2025, fecha en la que serán desactivados.

00:42.654 --> 00:48.100

[SPEAKER_02]: Recuerde que los equipos son encesión, tiene que entregarlos en cualquiera de nuestras oficinas o conservarlos para su recogida.

00:48.120 --> 00:49.902

[SPEAKER_00]: ¿De acuerdo, Juan Luis?

00:49.922 --> 00:50.262

[SPEAKER_02]: Vale, de acuerdo.

00:50.282 --> 00:53.005

[SPEAKER_02]: Vale, pues ya sería todo, Juan Luis, finalizamos la grabación.

Observación: En este caso, el sistema no identifica correctamente a los interlocutores a lo largo de toda la transcripción. Se observa una inconsistencia en la asignación de etiquetas de hablante (speaker labels), ya que inicialmente se asigna SPEAKER_00 al operador u operadora, pero posteriormente esta etiqueta se intercambia con SPEAKER_02, generando confusión. Esta falta de coherencia compromete la fiabilidad de la **diarización**, especialmente en contextos donde la identificación precisa de los interlocutores es crítica.

4. Google cloud para transcribir y luego analizar con Gemini

beatriz@TFGBea:~/ftp/download/2025-02-05/d926d087-8790-4e0a-b350

-4a822d81bdc2/extracted_files/20250205/converted_wavs$ cat diarizacion_prueba1.srt

00:00:00,000 --> 00:00:12,000

[SPEAKER_1]: A día 10 de febrero del 2025, procedemos a realizar la grabación de la llamada para la baja del contrato número 1300203. Dígame, por favor, el nombre completo y el DNI.

00:00:12,000 --> 00:00:20,000

[SPEAKER_2]: Juan Luis Burguete Martínez. El DNI es 85 301 377 N de Navarra.

00:00:20,000 --> 00:00:35,000

[SPEAKER_1]: Vale, Juan Luis. Actualmente tiene contratado el servicio fibra de 300 megas, por el cual paga 24.99, del cual nos solicita la baja total. ¿Está de acuerdo?

00:00:35,000 --> 00:00:38,000

[SPEAKER_2]: De acuerdo.

00:00:38,000 --> 00:00:49,000

[SPEAKER_1]: Del 2025, fecha en la que serán desactivados. Recuerde que los equipos son en sesión, tiene que entregarlos en cualquiera de nuestras oficinas o conservarlos para su ecogida.

00:00:49,000 --> 00:00:52,000

[SPEAKER_2]: De acuerdo.

00:00:52,000 --> 00:00:55,000

[SPEAKER_1]: ¿Cuándo...? Vale, de acuerdo.

00:00:55,000 --> 00:00:59,000

[SPEAKER_1]: Vale, pues ya estaría todo, Juan Luis. Finalizamos la grabación.

Conclusión de la Prueba 1 (centralita antigua):

Entre las tres soluciones analizadas, Google Cloud Speech-to-Text ha demostrado ser la más precisa en términos de transcripción literal, especialmente en la interpretación de números y valores numéricos, que reproduce correctamente sin errores de forma.

Por su parte, Whisper muestra un buen rendimiento en segmentación temporal, pero presenta deficiencias en la identificación de hablantes, lo que limita su utilidad en escenarios donde la diarización es relevante.

Como alternativa, se ha evaluado la combinación de transcripción con Google Cloud seguida de análisis de hablantes con Gemini, lo cual ha ofrecido mejores resultados en la asignación de interlocutores. A nivel estadístico, Gemini presenta una mayor consistencia que WhisperX en la diarización, por lo que sería la opción preferente en este contexto.

Se prevé replicar este enfoque en pruebas posteriores, utilizando el audio extraído de la plataforma Enreach (grabación "mi reina") en lugar de grabaciones provenientes de la centralita antigua.

5. Microsoft Azure

python3 transcribe_azure.py

Transcripción:

A día 10 de febrero del 2025 procedemos a realizar la grabación de la llamada para la baja del contrato número 1300203 dígame por favor el nombre completo y el DNI Juan Luis Burguete Martínez, el DNI es 8 ** n de Navarra vale Juan Luis actualmente tiene contratado el servicio fibra de 300 megas, por el cual sol paga 24,99. FALTA INFORMACION QUE NO HA TRANSCRITO

Se intentó **aumentar el volumen del archivo de audio original**, dado que se detectó que el nivel medio de decibelios era relativamente bajo. La hipótesis inicial era que Azure podría requerir un **umbral mínimo de volumen** para procesar correctamente la entrada de audio.

No obstante, tras la amplificación del sonido, **la transcripción resultante seguía siendo incompleta**.

```
[Parsed_volumedetect_0 @ 0x55e13f2c7100] n_samples: 425600
[Parsed_volumedetect_0 @ 0x55e13f2c7100] mean_volume: -25.
7 dB
[Parsed_volumedetect_0 @ 0x55e13f2c7100] max_volume: -2.0
dB
[Parsed_volumedetect_0 @ 0x55e13f2c7100] histogram_1db: 3
[Parsed_volumedetect_0 @ 0x55e13f2c7100] histogram_2db: 8
[Parsed_volumedetect_0 @ 0x55e13f2c7100] histogram_3db: 42
[Parsed_volumedetect_0 @ 0x55e13f2c7100] histogram_4db: 96
[Parsed_volumedetect_0 @ 0x55e13f2c7100] histogram_5db: 19
2
[Parsed_volumedetect_0 @ 0x55e13f2c7100] histogram_6db: 23
7
```

Ilustración 85: Intentando subir el volumen para poder analizar mejor el audio

Ejemplo del resultado tras amplificar el audio:

*"A día 10 de febrero del 2025 procedemos a realizar la grabación de la llamada para la baja del contrato número 13*2** dígame por favor el nombre completo y el DNI, Juan Luis Burguete Martínez, el DNI [...] 7 7 N de Navarra."*

Dado que el recorte se producía justo en el momento en que el/la operador/a iba a responder, se realizó también una prueba adicional consistente en **eliminación de silencios del archivo**, sin que ello mejorara significativamente el resultado.

Resultado tras eliminar silencios:

"El DNI [...] 7 N de Navarra."

En resumen, **Azure Speech-to-Text no consigue transcribir el audio completo** y pierde información relevante, incluso después de aplicar ajustes en el preprocesado. No se ha llegado a ejecutar la transcripción con un audio procedente directamente de Enreach, debido a dos factores:

1. La **calidad de audio es inferior** en esas grabaciones, lo que previsiblemente afectaría negativamente al rendimiento del sistema.
2. Se presentaron **dificultades para localizar el archivo** exacto en el sistema.

(Nota: El script utilizado para realizar esta prueba se denomina transcribe_azure.py.)

2.2 Audio Enreach

Prueba primer audio transcrito de google cloud. Y utilizando Gemini. (audio extraido de ENREACH)

transcripcion_elche_sanvicente_diarized.txt [SPEAKER_01] le damos la bienvenida que hable buena por favor dile a tu población Elche por motivos de calidad y por su seguridad está llamada puede grabarse para más información sobre la política de datos personales pulse 9 gracias le informamos sobre el tratamiento que se realizará de sus datos personales puede saltar esta locución en cualquier momento marcando la tecla almohadilla los datos personales facilitados por usted durante esa llamada serán tratados por las empresas del grupo que si desea hablar con ese vicio técnico pulse o diga uno para otras consultas manténgase a la espera gracias

[SPEAKER_02] centro muy bueno buenas tardes MI REINA que quería saber si me han cortado el internet porque yo mandé un mensaje que si me podían esperar hasta el día y el chico me dijo que te las mandé a ti te ha contratado a nombre de mi hijo Jaime ** dime el teléfono de tu hijo el DNI ***

[SPEAKER_01] televisión está cortado por impago no y que no hay internet aquí en casa voy a apagar el equipo y ahí

[SPEAKER_02] vale cariñoo

Conclusión

Los resultados obtenidos reflejan que la detección de hablantes no es fiable cuando se utilizan grabaciones provenientes de la plataforma Enreach. Las principales dificultades se

presentan en la asignación consistente de los interlocutores, con errores frecuentes en la identificación del operador/a y del cliente.

En contraste, los audios extraídos de la centralita antigua (Prueba 1 y Prueba 2) presentan una calidad notablemente superior, lo que facilita tanto la transcripción como la diarización.

A la vista de estos resultados, la principal fuente de error parece estar en el origen de los audios y no necesariamente en el modelo empleado. Una posible línea de mejora sería probar otro software o motor de diarización más robusto, si bien el tiempo y recursos disponibles para esta fase del proyecto son limitados, por lo que deberá evaluarse su viabilidad en función de prioridades y calendario.

2.3 prueba2.wav

- **Aws**

transcripcion-prueba2.json

{"jobName":"transcripcion-prueba2","accountId":"120569625568",

"status":"COMPLETED","results":{"transcripts":[{"transcript":

"sobretodo cuando quiero poner Netflix me da error y tengo que

apagar televisión, router, etcétera y volverlo a encender para

que se pueda conectar. Vale. Se lo comento al técnico para que

lo tenga en cuenta. Vale, ahora, de todas formas, ahora ahora le

explico a usted. Vale, Vale, perfecto. Gracias. Hasta ahora."}],"

- **Google cloud**

"transcript": "sobre todo cuando quiero poner Netflix me da error y tengo que apagar televisión router etcétera y volverlo a encender para que se pueda conectar"

"transcript": " se lo comento al técnico para que lo tenga en cuenta vale ahora vamos a pasar ahora le explica usted vale vale perfecto gracias ahora"

En esta prueba, el modelo de Google Cloud no ha introducido correctamente los signos de puntuación, o directamente no los ha aplicado. Esta carencia puede suponer una limitación en análisis futuros que requieran una segmentación precisa del discurso.

- **Amazon Web Services (AWS):**

Se ha detectado una omisión en la transcripción: el fragmento "vamos a pasar" no ha sido recogido. Asimismo, existe ambigüedad en la frase resultante, que genera dudas entre "ahora le explico a usted" y "ahora le explica usted". Por contexto, la interpretación más lógica parece ser la segunda opción.

- **Microsoft Azure:**

En la frase *"Sobre todo cuando quiero poner Netflix me da error y tengo que apagar televisión, router, etcétera y volverlo a encender para que se pueda conectar"*, el modelo ha omitido la segunda parte del mensaje, perdiendo así información relevante del testimonio. Esta pérdida de contenido reduce la fidelidad de la transcripción final.

3. RESUMEN DE LAS DIFERENTES IAS Y COMPARATIVAS

PRUEBA 1 (Centralita Vieja)

IA	*Transcripción Resumida*	*Errores Detectados*
AWS	Transcribe correctamente la mayor parte del texto, pero hay algunas imprecisiones en la numeración y puntuación.	No mantiene el formato de los números correctamente, omite algunos detalles menores.

Google Cloud	La transcripción es precisa y mantiene el formato original de los números.	Se detectan errores menores en la segmentación del texto.
Whisper	Buena transcripción y separación por tiempo.	No identifica bien a los hablantes, cambia de etiqueta de *speaker* de manera inconsistente.
Google Cloud + Gemini	Separa correctamente los hablantes y mejora la estructuración.	Mejora la diarización, pero aún se pueden encontrar imprecisiones en la separación de turnos de habla.

Tabla 23:Conclusiones Prueba1.wav

Conclusiones de Prueba1.wav

- Google Cloud es el más preciso en la transcripción de números y texto.
- Whisper ofrece buena segmentación temporal, pero no es fiable en la identificación de hablantes.
- AWS tiene imprecisiones en la numeración.
- La combinación de Google Cloud y Gemini mejora la diarización, siendo la mejor opción en esta prueba.

Prueba2.wav (Centralita Vieja)

- Nota: En esta prueba se han realizado menos test por disponer de menor cantidad de información que en la Prueba 1.

IA	*Transcripción Resumida*	*Errores Detectados*

AWS	Captura la mayor parte del contenido.	Se omite "vamos a pasar", falta claridad en la separación de turnos.
Google Cloud	Transcribe correctamente, pero separa la transcripción en dos partes.	No mantiene la puntuación correctamente.
Azure	Se come la segunda parte de la transcripción.	No logra capturar el audio completo.

Tabla 24:Conclusiones prueba2.wav

Prueba con Enreach

IA	***Transcripción Resumida***	***Errores Detectados***
Google Cloud + Gemini	Captura la mayor parte del audio.	No detecta bien a los hablantes.
WhisperX	Transcribe correctamente.	Errores en la asignación de hablantes.
Azure	No se probó por problemas con el audio.	No se encontró el audio adecuado.

Tabla 25: Enreach prueba de audio

Conclusiones generales

1. Google Cloud es la mejor opción para transcripciones limpias y precisas.
2. Whisper es bueno en la segmentación temporal, pero no fiable en diarización.
3. Gemini mejora la diarización cuando se usa con Google Cloud.
4. Azure es poco fiable, omite partes del audio.
5. El audio de Enreach es más difícil de transcribir, afectando la calidad de todas las IAs probadas.

4. COMANDOS IMPORTANTES Y ERRORES

- **Google Cloud Speech to text.**
  - **Comandos principales utilizados:**

```
## Verificar si el archivo está en el bucket de Google Cloud
gsutil ls gs://prueba-tfg-1739318298/


# Ejecutar transcripción de audio largo
gcloud ml speech recognize-long-running \
  gs://prueba-tfg-1739318298/966192001_CAW_966192001_Elche_SanVicente_2025-02-05_14-17-27_688209020.wav \
  --language-code=es-ES \
  --format=json > transcripcion_elche_sanvicente.json
```

◦ **Errores comunes:**

- **Error:** NOT_FOUND: No such object
  - **Causa:** El archivo no estaba correctamente subido al bucket.
  - **Solución:** Se verificó la existencia del archivo con gsutil ls y se corrigió la ruta.

- **Error:** InvalidArgument: sample_rate_hertz (16000) must match WAV header (8000)
  - **Causa:** El sample rate del archivo de audio no coincidía con el configurado en Google Cloud.
  - **Solución:** Se verificó el formato con ffprobe y se ajustó en el código Python.
- **Error:** Speaker Label no asignado correctamente (solo Speaker 0)
  - **Causa:** Google Cloud no estaba aplicando correctamente la diarización.
  - **Solución:** Se probó con WhisperX y Gemini para detectar hablantes.

- **Amazon Transcribe**
  - **Comandos utilizados**

```
# Iniciar transcripción en Amazon Transcribe
aws transcribe start-transcription-job \
  --transcription-job-name "transcripcion-prueba1" \
  --language-code es-ES \
  --media MediaFileUri=s3://prueba-tfg-aws/prueba1.wav \
  --output-bucket-name prueba-tfg-aws
```

◦ **Errores comunes:**

- **Error:** The URI that you provided doesn't point to an S3 object
  - **Causa:** El archivo no estaba correctamente subido o la ruta en S3 era incorrecta.
  - **Solución:** Se verificó con aws s3 ls s3://prueba-tfg-aws/.
- **Error:** Job already exists
  - **Causa:** Se intentó crear un trabajo con el mismo nombre.
  - **Solución:** Se cambió el nombre del job añadiendo una marca de tiempo.

- **Error:** Output bucket name incorrecto
  - **Causa:** No se configuró correctamente el bucket de salida.
  - **Solución:** Se especificó el bucket correcto en --output-bucket-name.

- **Azure Speech to Text**
  - **Comandos**

```
# Ejecutar transcripción en Azure
  python3 transcribe_azure.py
```

◦ **Errores comunes:**

- **Error**: SPXERR_FILE_OPEN_FAILED
  - **Causa**: El archivo de audio no estaba accesible o tenía un formato incorrecto.
  - **Solución**: Se verificó con ffprobe y se convirtió a otro formato con ffmpeg.
- **Error**: No se detecta el audio correctamente
  - **Causa**: El volumen del audio era muy bajo.
  - **Solución**: Se analizó con ffmpeg -af "volumedetect" y se amplificó.
- **Error**: Clave de API incorrecta
  - **Causa**: La clave de Azure no era válida o no estaba bien configurada.
  - **Solución**: Se generó una nueva clave en el portal de Azure.

4.Gemini AI (Google)

- **Comandos**

```
# Listar los modelos disponibles en Gemini
python3 modelosgemini.py

# Ejecutar el script de análisis de hablantes con Gemini
python3 pruebagemini.py
```

◦ **Errores comunes:**

- **Error:** module 'google.generativeai' has no attribute 'chat'
  - **Causa:** Se usó un método incorrecto (chat en lugar de generate_content).
  - **Solución:** Se corrigió el código con model.generate_content(prompt).
- **Error:** 404 models/gemini-pro is not found
  - **Causa:** El modelo no estaba disponible en la API utilizada.
  - **Solución:** Se listaron modelos disponibles y se seleccionó gemini-1.5-pro.

5. COMPARACION RESULTADOS

IA	*Precisión en Transcripción*	*Detección de Hablantes*	*Facilidad de Uso*	*Problemas Comunes*
Google Cloud	Alta	Baja (Speaker 0 en todos)	Media	Errores en sample rate y diarización
Amazon Transcribe	Media	Media (mejor que Google)	Alta	Problemas con rutas de S3

Azure Speech	Media	Alta	Baja	Problemas con formato de audio
Gemini AI	No transcribe (solo análisis)	Muy Alta (mejor diarización)	Media	Errores con modelos y claves API

Tabla 26:Resultados finales de comparación de IAs

6. CONCLUSIÓN FINAL

Después de todas las pruebas:

- **Para transcripción pura**, **Google Cloud** es la mejor opción, pero su **diarización no es fiable**.
- **Amazon Transcribe** es **más fácil de usar** y ofrece **mejor diarización que Google**, pero depende de S3.
- **Azure Speech** es **bueno en diarización**, pero es más **difícil de configurar**.
- **Gemini AI no transcribe**, pero es **el mejor para diarización**.