UAB
Universitat Autònoma de Barcelona

Dipòsit digital de documents de la UAB

This is the **published version** of the bachelor thesis:

Ros Ballester, Mireia. *Sistema inteligente de resolución automática de tickets de soporte mediante IA generativa y búsqueda semántica en BigQuery*. Treball de Final de Grau (Universitat Autònoma de Barcelona), 2026 (Grau en Enginyeria de Dades)

This version is available at https://ddd.uab.cat/record/337362

under the terms of the CC BY-SA license.

Sistema inteligente de resolución automática de tickets de soporte mediante IA generativa y búsqueda semántica en BigQuery

Mireia Ros Ballester

Resumen– El presente trabajo describe el desarrollo de un sistema inteligente para la resolución automática de tickets de soporte empresarial, mediante búsqueda semántica e IA generativa. La arquitectura utiliza Google BigQuery para almacenar un histórico de 28.274 tickets junto con sus representaciones vectoriales, generadas con el modelo `text-multilingual-embedding-002` de Vertex AI. Ante una nueva incidencia, el sistema genera su embedding, recupera el caso histórico más similar aplicando filtros por prioridad y departamento, y toma la solución que dicho caso propone para que el modelo Gemini 2.5 Flash genere a partir de ella una respuesta personalizada y profesional. La evaluación del sistema, realizada sobre un conjunto de test de 5.655 tickets, muestra una similitud coseno promedio de 0,9668, una puntuación de calidad de 8,85/10 y una tasa de acierto del 93 %. Finalmente, se incluye una interfaz web desarrollada en *Streamlit* que permite operar el sistema de forma intuitiva.

Palabras clave– Búsqueda semántica, embeddings, inteligencia artificial generativa, Google BigQuery, tickets de soporte.

Abstract– This work presents the development of an intelligent system for the automatic resolution of enterprise support tickets, using semantic search and generative AI. The architecture uses Google BigQuery to store a historical dataset of 28,274 tickets along with their vector representations, generated with Vertex AI's `text-multilingual-embedding-002` model. When a new incident arrives, the system generates its embedding, retrieves the most similar historical case applying filters by priority and department, and takes the solution proposed by that case so that the Gemini 2.5 Flash model generates from it a personalised and professional response. The evaluation, carried out on a test set of 5,655 tickets, shows an average cosine similarity of 0.9668, a quality score of 8.85/10 and an accuracy rate of 93 %. Finally, a web interface developed in *Streamlit* allows intuitive operation of the system.

Keywords– Semantic search, embeddings, generative artificial intelligence, Google BigQuery, support tickets.

1 Contexto del trabajo

En un entorno empresarial cada vez más digitalizado, la rapidez y eficacia en la gestión de incidencias se ha convertido en un factor crítico para la satisfacción del cliente y la competitividad de las organizaciones [1]. Muchas empresas generan grandes volúmenes de tickets de soporte que requieren atención inmediata y eficiente. Sin embargo, el análisis manual y la resolución de estos tickets es un proceso costoso y propenso a errores [2], especialmente cuando se trata de grandes cantidades de información.

En este trabajo se propone el desarrollo de un sistema inteligente para la gestión automatizada de tickets de soporte, utilizando técnicas avanzadas de inteligencia artificial y análisis de datos. El sistema está basado en la plataforma Google BigQuery, y combina IA generativa [3] y búsqueda semántica por vectores [4] para identificar automáticamente similitudes entre tickets históricos ya resueltos y nuevos tickets que se generen. A partir de esta comparación, el sistema genera automáticamente una respuesta personalizada y profesional lista para ser enviada al cliente, reduciendo así la intervención manual del agente de soporte.

La motivación de este proyecto es demostrar cómo la inteligencia artificial puede optimizar procesos empresariales, aumentar la eficiencia operativa y automatizar tareas repetitivas, reduciendo la carga de trabajo humano y ofreciendo soluciones rápidas y precisas en entornos de soporte.

• E-mail de contacto: mireiarosbr12@gmail.com
• Trabajo tutorizado por: Ana Oropesa Física
• Curso 2026/27

Seguidamente, se presentarán los objetivos del proyecto y la metodología seguida para su desarrollo. A continuación, se revisará el estado del arte y el marco teórico de las tecnologías empleadas. Posteriormente, se detallará de manera exhaustiva el proceso de desarrollo del sistema, incluyendo la preparación de los datos, la generación de embeddings y la implementación de la búsqueda semántica. A continuación, se evaluarán los resultados obtenidos mediante diferentes métricas y se presentará la interfaz web desarrollada para visualizar el funcionamiento completo del sistema. Finalmente, se extraerán las conclusiones del proyecto, destacando los aprendizajes obtenidos y las posibles líneas de mejora.

2 Objetivos

El objetivo general de este proyecto es diseñar e implementar un sistema inteligente capaz de automatizar el ciclo completo de resolución de tickets de soporte en entornos empresariales. Este sistema integrará técnicas de búsqueda semántica [5] sobre grandes volúmenes de datos almacenados en Google BigQuery [6], con el fin de mejorar la eficiencia operativa, reducir la intervención manual y optimizar los tiempos de respuesta.

A su vez, este proyecto tiene los siguientes objetivos específicos:

1. Realizar la búsqueda, recopilación y preparación de los datos, llevando a cabo tareas de limpieza, estructuración y análisis exploratorio de los tickets históricos. Este objetivo se desarrollará en la fase inicial del proyecto, con el fin de garantizar la calidad y consistencia de la información antes de su procesamiento [6].

2. Diseñar la representación vectorial de los tickets, implementando la generación de embeddings que permitan transformar el contenido textual en vectores semánticos almacenados en BigQuery. Este objetivo deberá completarse antes de iniciar la fase de comparación y recuperación de información.

3. Desarrollar el sistema de comparación semántica, implementando un mecanismo de búsqueda basado en similitud vectorial, que permita identificar automáticamente tickets históricos similares a un nuevo ticket generado [7]. Este objetivo constituye el núcleo técnico del sistema.

4. Integrar un modelo de inteligencia artificial generativa que, a partir de la solución del ticket histórico más similar recuperado, genere una respuesta personalizada y automatizada para el usuario final. Este objetivo se desarrollará durante la fase principal de implementación.

5. Evaluar el rendimiento del sistema y proponer posibles mejoras, analizando métricas como la similitud semántica en la recuperación de tickets similares, la calidad de las respuestas generadas y el tiempo de respuesta del sistema en cada una de sus fases [8]. Este objetivo se abordará en la fase final del proyecto.

6. Desarrollar una interfaz web que permita visualizar e interactuar con el sistema completo, integrando todas las fases del ciclo de resolución automática de forma accesible para cualquier usuario. Este objetivo se desarrollará tras la finalización e integración del resto de componentes del sistema.

3 Metodología

El desarrollo de este proyecto se ha estructurado siguiendo un modelo en cascada [9], en el que las distintas fases se abordan de forma secuencial. Este enfoque permite establecer una planificación clara desde el inicio y avanzar de manera progresiva, asegurando que cada etapa quede correctamente validada antes de iniciar la siguiente. Dado que el alcance del sistema y sus objetivos están definidos desde el comienzo, esta metodología resulta adecuada para garantizar coherencia y control en el desarrollo.

El proyecto se divide en varias etapas consecutivas: en primer lugar, la preparación y análisis de los datos históricos de tickets; posteriormente, el diseño de la arquitectura del sistema y la generación de representaciones vectoriales; a continuación, la implementación del sistema de búsqueda semántica y la integración del modelo de inteligencia artificial generativa; y finalmente, la evaluación del rendimiento, el análisis de resultados y el desarrollo de la interfaz web de visualización. Esta estructura facilita validar cada componente antes de integrarlo en el sistema completo, reduciendo así posibles errores acumulativos.

Para la planificación y control del proyecto se utilizan dos herramientas complementarias. Por un lado, el diagrama de Gantt [10] permite organizar temporalmente las fases del trabajo, visualizar la duración estimada de cada tarea y establecer los hitos principales, tal y como se muestra en el Apéndice A.1. Por otro lado, Trello [11] se emplea como herramienta de seguimiento operativo, permitiendo desglosar tareas en mayor detalle y controlar su estado de ejecución, tal y como se muestra en el Apéndice A.2. La combinación de ambas herramientas asegura tanto una visión global del proyecto como un seguimiento detallado de su evolución.

4 Estado del Arte

La gestión de tickets de soporte es una necesidad crítica en cualquier organización empresarial moderna [1]. Actualmente, las plataformas más utilizadas en el mercado son Zendesk [12], Jira Service Management [13] y ServiceNow [14], que en conjunto concentran la mayor parte del mercado global de soporte técnico. Estas herramientas permiten centralizar todas las solicitudes en un único sistema, asignarles automáticamente una prioridad, enrutarlas al equipo correspondiente y hacer un seguimiento del proceso hasta su resolución. Sin embargo, a pesar de su amplia adopción, todas ellas comparten una limitación importante: la búsqueda y clasificación de tickets se basa en la coincidencia de palabras clave exactas, lo que impide identificar tickets similares cuando el usuario describe el mismo problema de forma diferente [2]. Esta restricción dificulta la reutilización del conocimiento acumulado en tickets históricos ya resueltos, una oportunidad de mejora que las técnicas más recientes de inteligencia artificial han comenzado a abordar [3].

El sistema propuesto en este trabajo aplica precisamente estas técnicas para comparar el significado de los tickets independientemente de las palabras utilizadas, superando así la limitación de las plataformas actuales basadas en coincidencia exacta de palabras clave.

5 MARCO TEÓRICO

En este apartado se describen los conceptos teóricos fundamentales sobre los que se basa el sistema desarrollado. Su comprensión permite entender las decisiones técnicas tomadas y el funcionamiento del sistema en su conjunto.

5.1. Procesamiento del lenguaje natural

El procesamiento del lenguaje natural (NLP) es una rama de la inteligencia artificial que permite a los ordenadores leer, interpretar y comprender el texto escrito por personas [3]. Los embeddings, tecnología central de este proyecto, son una de las técnicas más avanzadas dentro del NLP, por lo que comprender este campo es fundamental para entender el sistema desarrollado.

5.2. Embeddings y modelos de lenguaje

Un embedding es una representación numérica de un texto en forma de vector, es decir, una lista de números que captura el significado semántico de ese texto [4]. Estos vectores son generados por modelos de lenguaje, sistemas de inteligencia artificial entrenados con enormes cantidades de texto que aprenden a transformar cualquier frase en números que representan su significado [3]. Dos textos que describen el mismo problema pero con palabras diferentes producirán vectores muy similares, mientras que textos sobre temas distintos producirán vectores alejados entre sí. Esta propiedad es la que permite buscar tickets similares por significado y no por palabras exactas.

5.3. Búsqueda vectorial por similitud

La búsqueda vectorial por similitud es una técnica que permite encontrar los vectores más parecidos dentro de un conjunto de datos [5]. Para medir esa similitud se compara la dirección de los vectores en el espacio: cuanto más parecida es su orientación, más similares son los textos que representan. Siendo un valor continuo entre 0 y 1, un valor de 1 indica textos idénticos semánticamente y un valor de 0 indica textos completamente diferentes.

5.4. Google Cloud

Google Cloud es la plataforma de servicios en la nube de Google que proporciona infraestructura, almacenamiento y herramientas de inteligencia artificial accesibles desde cualquier lugar [6]. En este proyecto constituye la base tecnológica principal sobre la que se apoyan todos los servicios utilizados.

5.5. IA generativa aplicada a soporte técnico

La inteligencia artificial generativa es una rama de la IA capaz de generar contenido nuevo, como texto, a partir de información existente [3]. En el contexto del soporte técnico, se utiliza para proponer respuestas automáticas a nuevas incidencias basándose en soluciones históricas similares, reduciendo así la intervención manual de los agentes.

5.6. Retrieval-Augmented Generation

El Retrieval-Augmented Generation (RAG) es la técnica que une la búsqueda semántica con la IA generativa [3]. En lugar de generar una respuesta desde cero, el sistema primero recupera los documentos más similares a la consulta y los utiliza como base para proponer una solución.

6 HERRAMIENTAS Y TECNOLOGÍAS

En este apartado se presentan las herramientas y tecnologías utilizadas para el desarrollo del proyecto, explicando la función de cada una y el motivo por el que ha sido seleccionada.

6.1. Lenguaje y entorno de desarrollo

El lenguaje de programación utilizado en este proyecto es Python, elegido por su gran cantidad de librerías especializadas en análisis de datos e inteligencia artificial, y por su compatibilidad directa con los servicios de Google Cloud, en particular con BigQuery. Como entorno de desarrollo se ha utilizado Visual Studio Code, un editor de código que permite trabajar de forma cómoda y organizada. Dentro de este entorno, el código se ha desarrollado en formato notebook de Jupyter, que permite ejecutar el código por bloques y ver los resultados de forma inmediata, lo que facilita el seguimiento y la validación de cada paso del proceso.

6.2. Librerías Python

Para el desarrollo del proyecto se han utilizado varias librerías de Python, cada una con una función concreta. *Pandas* es la librería utilizada para trabajar con los datos en formato tabla, empleada durante toda la fase de limpieza y preparación del dataset. *Deep-translator* se ha usado para traducir automáticamente los tickets al español a través de Google Translate de forma gratuita. *Google-cloud-bigquery* y *google-cloud-aiplatform* son las librerías oficiales de Google que permiten conectar Python con BigQuery y Vertex AI, facilitando tanto la carga de datos como la generación de embeddings. *Pyarrow* es una librería necesaria para que Python pueda subir tablas de datos correctamente a BigQuery. *Scikit-learn* se ha utilizado para dividir el dataset en los subconjuntos de entrenamiento y test mediante la técnica de train/test split. *Concurrent.futures* es la librería estándar de Python utilizada para paralelizar las consultas durante el cálculo de las métricas de evaluación, reduciendo significativamente el tiempo de ejecución. Por último, *Streamlit* es la librería utilizada para el desarrollo de la interfaz web del sistema, que permite crear aplicaciones interactivas sin necesidad de conocimientos de desarrollo web.

6.3. Servicios Google Cloud

El proyecto se apoya en tres servicios de la plataforma Google Cloud, cada uno con un rol específico dentro del sistema. BigQuery es la base de datos en la nube donde se almacenan los tickets históricos junto con sus embeddings y donde se ejecutan las búsquedas de similitud semántica [6]. Vertex AI proporciona acceso al modelo `text-multilingual-embedding-002`, responsable de transformar el texto de cada ticket en su representación vectorial [5]. Por último, Google Cloud CLI es la herramienta de línea de comandos que permite gestionar y configurar los servicios de Google Cloud directamente desde la terminal, utilizada en este proyecto para autenticar la conexión entre Python y Google Cloud de forma segura [6].

7 DESARROLLO DEL SISTEMA

En este apartado se describe el proceso completo de desarrollo del sistema, desde la preparación de los datos históricos hasta la implementación de la búsqueda semántica y la generación de respuestas automáticas. El sistema se estructura en dos grandes bloques: el primero, que se ejecuta una única vez al inicio, comprende la preparación y limpieza del dataset, la generación de embeddings y su almacenamiento en Google BigQuery; el segundo es el ciclo de resolución automática, que se activa de forma reiterada cada vez que llega un nuevo ticket, generando su embedding, buscando los tickets históricos más similares y proponiendo automáticamente una solución al cliente.

La figura siguiente muestra de forma visual el flujo completo de las fases de preparación y almacenamiento.

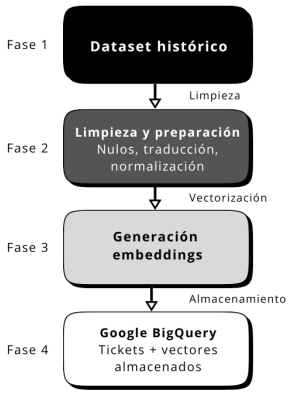


Fig. 1: Fases de preparación de datos y almacenamiento en BigQuery.

7.1. Preparación y limpieza del dataset

El dataset utilizado es el *Multilingual Customer Support Tickets*, disponible en Kaggle [15], un conjunto de datos sintético compuesto por cinco archivos CSV con tickets de soporte empresarial en múltiples idiomas, que incluye tanto el problema del cliente como la respuesta del agente.

De los cinco archivos disponibles se descartan tres: los dos archivos alemanes (`german_normalized`) por no contener la columna `answer` y estar limitados a un único idioma, y el archivo `multilang420k` por ser una versión anterior del `aa_5250version`. Se seleccionan únicamente los dos archivos que comparten la misma estructura de columnas e incluyen la columna `answer`: `dataset-tickets-multi-lang3-4k.csv` (4.000 filas), elegido por ser el único con tickets en español nativo, y `aa_dataset-tickets-multi-lang-5-2-50-version.csv` (28.587 filas), elegido por su mayor cantidad de datos disponibles.

El análisis de calidad confirma la ausencia de duplicados en ambos datasets. Se identifican nulos en columnas críticas: 467 filas sin `subject` y 1 sin `body` en el dataset 1, y 3.838 filas sin `subject` y 7 sin `answer` en el dataset 2. Estas filas se eliminan ya que sin `subject` y `body` no hay texto para generar el embedding, y sin `answer` no hay solución que proponer. Tras la eliminación, el dataset 1 queda en 3.532 filas y el dataset 2 en 24.743.

Se eliminan también las columnas incompatibles entre ambos datasets: `business_type` y `tag_9` del dataset 1, y `version` del dataset 2. Además, las columnas `tag_1` hasta `tag_8`, aunque comunes en ambos datasets, también se eliminan ya que no aportan valor para la búsqueda semántica. Tras estas eliminaciones, ambos datasets quedan con las mismas 7 columnas: `subject`, `body`, `answer`, `type`, `queue`, `priority` y `language`. Los dos datasets se unen mediante concatenación obteniendo un total de 28.275 filas.

El dataset combinado contiene únicamente 714 tickets en español (2,5 % del total), siendo el resto en inglés (14.948), alemán (11.766), francés (428) y portugués (419). Se traducen las columnas `subject`, `body` y `answer` mediante la librería `deep-translator` con Google Translate, procesando los datos por lotes de 100 filas con guardado automático del progreso. El proceso requiere aproximadamente dos días de ejecución continua hasta completar la traducción del 100 % de las filas al español.

Finalmente, se normaliza el texto convirtiendo todo a minúsculas y eliminando caracteres especiales, conservando tildes y puntuación básica por su relevancia semántica. El resultado es el archivo `dataset_final_tickets.csv` con 28.274 filas, 7 columnas, cero nulos y el 100 % del texto en español.

7.2. Configuración de Google Cloud

El primer paso es crear una cuenta en Google Cloud, plataforma que ofrece una prueba gratuita. Una vez dentro, se configura el proyecto renombrándolo como `TFG-tickets-soporte` y se habilitan las APIs de Vertex AI y BigQuery.

Para que Python pueda acceder a estos servicios de forma segura, se crea una cuenta de servicio llamada `tfg-servicio`. Esta cuenta actúa como una identidad dentro de Google Cloud a la que se le asignan tres permisos: Administrador de BigQuery, Administrador de Vertex

AI y Administrador de almacenamiento. Sin estos permisos, Python no puede crear tablas, subir datos ni generar embeddings.

La conexión entre VS Code y Google Cloud se establece mediante Google Cloud CLI. La autenticación se realiza con el método Application Default Credentials (ADC), ejecutando el comando `gcloud auth application-default login` en la terminal. Este proceso abre el navegador, solicita iniciar sesión con la cuenta de Google y guarda automáticamente las credenciales en el ordenador para que Python pueda conectarse a Google Cloud sin necesidad de introducir contraseñas manualmente.

7.3. Generación de embeddings

Una vez configurado Google Cloud y obtenido el dataset limpio, el siguiente paso es generar los embeddings de cada ticket. Para ello se inicializa la conexión con Vertex AI desde Python, especificando el proyecto `TFG-tickets-soporte` y la región `us-central1`. Esta región se elige porque es donde el modelo `text-multilingual-embedding-002` está disponible, ya que no todos los modelos de Vertex AI son accesibles desde todas las regiones.

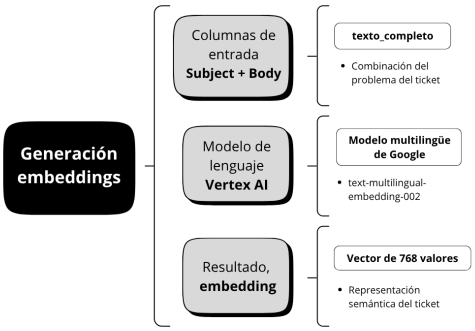


Fig. 2: Proceso de generación de embeddings a partir de las columnas del ticket.

Para generar el embedding de cada ticket, se combinan las columnas `subject` y `body` en una única columna llamada `texto_completo`, que representa el problema completo descrito por el cliente. El modelo analiza el texto y genera internamente 768 valores numéricos, cada uno capturando una dimensión diferente del significado del texto, como el tema, el contexto o el tipo de problema. En conjunto, estos 768 valores forman el embedding del ticket y representan su significado semántico de forma matemática, de modo que dos tickets que describen el mismo problema tendrán embeddings muy parecidos aunque estén escritos con palabras diferentes. La columna `answer` no se convierte en embedding porque no es una columna de búsqueda sino de respuesta: el sistema la recupera como solución una vez encuentra los tickets similares.

El proceso se ejecuta ticket a ticket y, dado el volumen de peticiones a la API, existe el riesgo de que se interrumpa en cualquier momento. Para evitar tener que regenerar todos los embeddings desde cero en caso de interrupción, los embeddings se guardan localmente de forma incremental en el archivo `dataset_con_embeddings.csv` antes de subir los datos a BigQuery, actuando como copia de seguridad del progreso realizado.

7.4. Carga en BigQuery

Se crea en BigQuery el dataset `tickets_soporte` y se sube la tabla con las 28.274 filas y 9 columnas: `subject`, `body`, `answer`, `type`, `queue`, `priority`, `language`, `texto_completo` y `embedding`. Esta tabla constituye la base de datos histórica sobre la que se ejecutarán las búsquedas de similitud semántica.


Información del trabajo | Resultados | Visualización | JSON | Detalles de la ejecución | Gráfico de ejecución

Fila	subject	body	answer	queue	priority	texto_completo	embedding
1	asistencia para configurar la im...	estimado soporte técnico de la tienda en línea: necesito ayuda para configurar mi epson ecotank et-4760 para impresión inalámbrica con	estimado nombre, para configurar su epson ecotank et-4760 para impresión inalámbrica, asegúrese de que esté conectado a su wi-fi.	Product Support	high	asistencia para configurar la impresión inalámbrica estimado soporte técnico de la tienda en línea: necesito ayuda para configurar mi	[0.028761347755789797, -0.016517096993062973, 0.006075812962565181, -0.002180701903653145, -0.004281603731215,
2	consulta sobre servicios de con...	estimado servicio de atención al cliente, mi nombre es name, y estoy interesado en obtener más información sobre sus servicios de consultoría en ti,	estimado name, gracias por su consulta sobre nuestros servicios de consultoría en ti. ofrecemos soluciones personalizadas en desarrollo	General Inquiry	medium	consulta sobre servicios de consultoría en ti estimado servicio de atención al cliente, mi nombre es name, y estoy interesado en obtener más	[0.008314495884692078, 0.005476304795595023, 0.05055769405113202, 0.060212101787532872, -0.020397294031271553,
3	problema de conectividad lenta de red informado con el dispositivo cisco router isr4331	hola, le escribo para informar un problema con la conectividad de la red que parece estar afectando nuestra velocidad de internet.	hola nombre, gracias por comunicarse con respecto al problema de red con su enrutador cisco isr4331. comience verificando los	Technical Support	medium	problema de conectividad lenta de red informado con el dispositivo cisco router isr4331 hola, le escribo para informar un problema con la	[0.010729991907347403, -0.040912901551395798, 0.016531012211108208, 0.022603713496622409, -0.031077556372084198,

Fig. 3: Vista previa de la tabla tickets_soporte en Google BigQuery.

A partir de este punto se activa la segunda fase del sistema, correspondiente al ciclo de resolución automática que se repite cada vez que llega un nuevo ticket. La figura siguiente muestra de forma visual este ciclo.

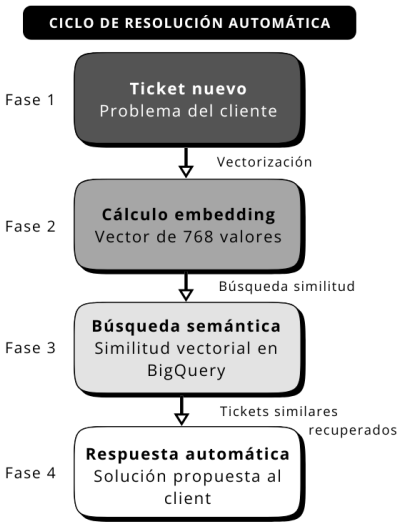


Fig. 4: Ciclo de resolución automática ante un nuevo ticket.

7.5. Desarrollo del sistema de búsqueda semántica

Una vez almacenados los tickets históricos junto con sus embeddings en BigQuery, se implementa el sistema de búsqueda semántica. Este sistema es el núcleo del proyecto: dado un ticket nuevo, genera su embedding y lo compa-

ra con todos los embeddings almacenados para encontrar el ticket histórico más similar.

El proceso comienza generando el embedding del ticket nuevo a partir de su texto, de la misma forma que se hizo con los tickets históricos. A continuación se ejecuta una consulta SQL en BigQuery que calcula la similitud coseno entre el embedding del ticket nuevo y cada uno de los embeddings almacenados en la tabla. La similitud coseno mide el ángulo entre dos vectores: cuanto más parecida es su dirección, más alto es el valor de similitud y más similares son semánticamente los dos tickets. El resultado es el ticket histórico con mayor similitud, del que se recupera la columna `answer` como solución de referencia.

Para mejorar la calidad de los resultados, se aplican dos filtros opcionales en la consulta. El primero es el filtro por nivel de prioridad, que puede ser `low`, `medium` o `high`, y garantiza que la solución propuesta corresponda al mismo nivel de urgencia que el ticket nuevo. El segundo es el filtro por cola o departamento, que puede ser `Technical Support`, `Customer Service`, `Product Support`, `Billing`, `General Inquiry` o `Network Support`, y asegura que la solución provenga del área de soporte más adecuada al tipo de problema. De este modo, si un ticket nuevo tiene prioridad `high` y pertenece a `Technical Support`, el sistema busca únicamente entre los tickets históricos que cumplen ambas condiciones, garantizando que la solución propuesta sea adecuada no solo semánticamente sino también en términos de urgencia y especialización.

Para garantizar que el sistema devuelve siempre una respuesta, se implementa un mecanismo de fallback en tres niveles: en primer lugar se busca con ambos filtros activos; si no se encuentran resultados, se busca únicamente con el filtro de prioridad; y si tampoco hay resultados, se realiza la búsqueda sin ningún filtro. Este mecanismo asegura que el sistema sea robusto ante combinaciones de prioridad y cola poco representadas en el dataset histórico.

7.6. Integración de IA generativa

Una vez implementado el sistema de búsqueda semántica, se observa durante las pruebas que la solución recuperada del ticket histórico más similar es semánticamente correcta pero no está personalizada para el cliente que ha enviado el ticket. Las respuestas históricas son genéricas, dirigidas a un cliente anónimo y basadas en un problema que, aunque similar, no es exactamente igual al nuevo. Para solucionar este problema se integra un modelo de inteligencia artificial generativa que toma como entrada tanto el ticket nuevo del cliente como la solución histórica recuperada, y genera una respuesta completamente personalizada y adaptada al problema concreto.

El modelo elegido es Gemini 2.5 Flash de Google, accesible a través de su API [16]. Se elige este modelo por su integración natural con el ecosistema de Google Cloud ya utilizado en el proyecto, por su capacidad multilingüe y por ofrecer un nivel de calidad suficiente para el caso de uso de soporte técnico. La integración se realiza mediante la librería `google-genai` desde Python, utilizando las mismas credenciales de autenticación configuradas para BigQuery y Vertex AI.

El funcionamiento es el siguiente: una vez recuperado el ticket histórico más similar, se construye un prompt que incluye el problema descrito por el cliente, su nombre, el nivel de prioridad, el departamento y la solución histórica como referencia. Este prompt se envía al modelo, que genera una respuesta profesional, cordial y personalizada en español, dirigida al cliente por su nombre y con pasos concretos para resolver su problema. El modelo tiene instrucciones explícitas de no mencionar que está usando una solución histórica, de dirigirse al cliente de forma profesional y de firmar la respuesta como “Equipo de Soporte Técnico”.

7.7. Ciclo completo de resolución automática

Una vez desarrollados todos los componentes del sistema, el ciclo completo de resolución automática de tickets funciona de la siguiente manera. Cuando llega un nuevo ticket, el sistema genera su embedding combinando el `subject` y el `body` del problema descrito por el cliente. A continuación ejecuta la búsqueda semántica en BigQuery aplicando los filtros de prioridad y cola para encontrar el ticket histórico más similar. Con el ticket similar encontrado y su solución histórica como referencia, Gemini 2.5 Flash genera una respuesta completamente personalizada, adaptada al problema concreto del cliente y firmada por el Equipo de Soporte Técnico. El resultado final es una respuesta profesional, cordial y específica que se propone automáticamente sin intervención humana, cerrando así el ciclo completo de soporte de forma autónoma.

8 Evaluación y resultados

Para medir la exactitud y el rendimiento del sistema desarrollado se definen tres métricas que evalúan diferentes aspectos del ciclo de resolución automática. Con el fin de garantizar la validez de los resultados, el dataset se divide previamente en dos subconjuntos mediante la técnica de train/test split: un 80 % de los tickets (22.619 filas) se carga en BigQuery como base de datos histórica de entrenamiento, y el 20 % restante (5.655 filas) se reserva como conjunto de test, es decir, como tickets que el sistema no ha visto previamente y sobre los que se evalúa su capacidad de búsqueda y resolución. Esta división se realiza de forma proporcional por nivel de prioridad, garantizando que la proporción 80/20 se mantiene dentro de cada nivel, es decir, que tanto el conjunto de train como el de test contienen una representación equilibrada de tickets de prioridad `low`, `medium` y `high`. Esta división garantiza que las métricas reflejan el comportamiento real del sistema ante tickets nuevos y no vistos, evitando así cualquier sobreajuste u overfitting en la evaluación. Las tres métricas se calculan sobre 100 tickets extraídos aleatoriamente del conjunto de test, lo que permite comparar los resultados de forma directa y coherente.

8.1. Similitud semántica por nivel de prioridad

Esta métrica evalúa la eficiencia de la primera fase del ciclo de resolución automática: la búsqueda semántica. Concretamente, mide si el sistema es capaz de encontrar, dentro del conjunto de tickets históricos almacenados en BigQuery correspondiente al conjunto de train, un ticket suficientemente similar al nuevo ticket recibido. Para ello, cada uno de los 100 tickets seleccionados aleatoriamente del conjunto de test pasa por el proceso de generación de embeddings y se busca en BigQuery el ticket del train cuyo embedding es más cercano al suyo, obteniendo un valor de similitud coseno que indica cómo de parecidos son semánticamente ambos tickets. Para comprobar la fiabilidad de este proceso los resultados se analizan agrupándolos por nivel de prioridad, es decir, low, medium y high, de forma que se puede verificar si la calidad de la búsqueda es consistente independientemente del nivel de urgencia del ticket.

Cabe recordar que el valor de similitud coseno oscila entre 0 y 1, siendo 1 el valor máximo que indica que dos tickets son semánticamente idénticos y 0 que no tienen ninguna relación. Se establece un umbral mínimo de 0,9 porque por encima de este valor se considera que los dos tickets describen el mismo tipo de problema con suficiente certeza como para que la solución histórica recuperada sea relevante. Por debajo de este umbral, la similitud podría deberse a coincidencias parciales y no a una verdadera similitud semántica. Los resultados obtenidos son los siguientes:

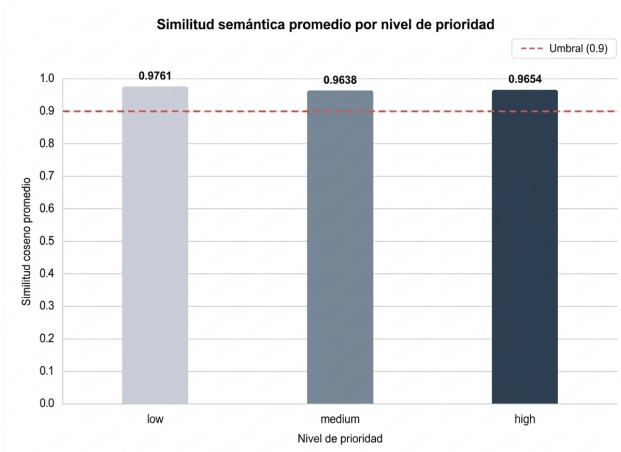


Fig. 5: Similitud semántica promedio por nivel de prioridad.

Los resultados muestran una similitud media de 0,9761 para `low`, 0,9638 para `medium` y 0,9654 para `high`. Los tres valores se encuentran muy por encima del umbral mínimo establecido, lo que demuestra que el sistema encuentra tickets históricos muy similares de forma consistente para los tres niveles de prioridad, confirmando que los embeddings generados capturan correctamente el significado semántico del texto.

8.2. Precisión de la solución generada

Esta métrica evalúa si la solución final que proporciona el sistema es realmente adecuada para resolver el problema del cliente. A diferencia de la similitud coseno, que mide parecido semántico entre textos, esta métrica mide la calidad real de la respuesta generada comparando el resultado final producido por Gemini con la respuesta real del ticket de test, almacenada en la columna `answer`.

Para evaluar la calidad de las respuestas se utiliza Gemini como juez externo, ya que es un modelo de lenguaje avanzado capaz de comprender el significado de dos textos y determinar si resuelven el mismo problema. A cada par de respuestas, la generada por el sistema y la real del ticket de test, se le pide a Gemini que las evalúe del 1 al 10 analizando si abordan el mismo tipo de incidencia, si los pasos de resolución propuestos son equivalentes y si el nivel de detalle y la orientación de la solución son comparables. Una puntuación de 1 indica que las respuestas son completamente diferentes y no tienen relación, mientras que una puntuación de 10 indica que resuelven exactamente el mismo problema de la misma forma. Se establece un umbral mínimo de 7 porque por encima de este valor se considera que la respuesta generada es suficientemente adecuada para resolver el problema del cliente, aunque no sea idéntica a la respuesta real. Al igual que en la métrica anterior, los 100 valores obtenidos se agrupan por nivel de prioridad para verificar que la calidad de las respuestas generadas es consistente independientemente de si el ticket es de prioridad `low`, `medium` o `high`. Los resultados obtenidos se muestran en la figura siguiente:

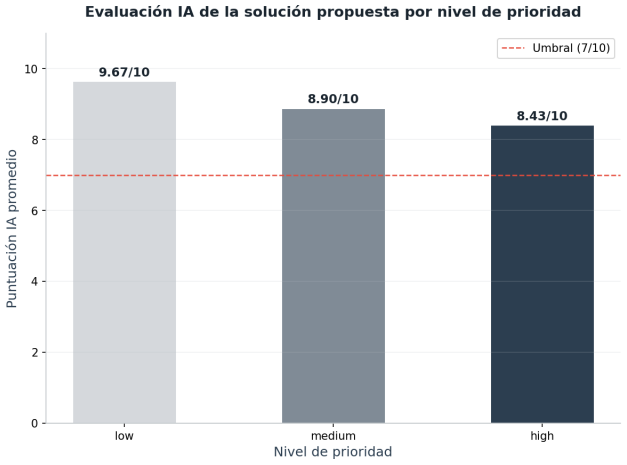


Fig. 6: Evaluación IA de la solución propuesta por nivel de prioridad.

Los resultados muestran una puntuación media de 9,67/10 para `low`, 8,90/10 para `medium` y 8,43/10 para `high`, todos muy por encima del umbral mínimo establecido de 7/10. Esto demuestra que el sistema no solo encuentra tickets semánticamente similares sino que además genera respuestas que resuelven el mismo tipo de problema, validando así la calidad del ciclo completo de resolución automática.

8.3. Tiempo de respuesta del sistema

Esta métrica analiza el tiempo que tarda el sistema en completar el ciclo completo de resolución de un ticket, desde que llega el problema del cliente hasta que se genera la respuesta final. Se elige para identificar en qué fase del proceso se concentra el mayor tiempo de ejecución.

Para calcularla se seleccionan 100 tickets aleatorios del conjunto de test y se cronometra cada una de las tres fases del sistema por separado: la generación del embedding del ticket nuevo mediante Vertex AI, la búsqueda del ticket más similar en BigQuery y la generación de la respuesta personalizada por Gemini. Se calcula el tiempo promedio de cada fase sobre los 100 tickets, con los siguientes resultados:

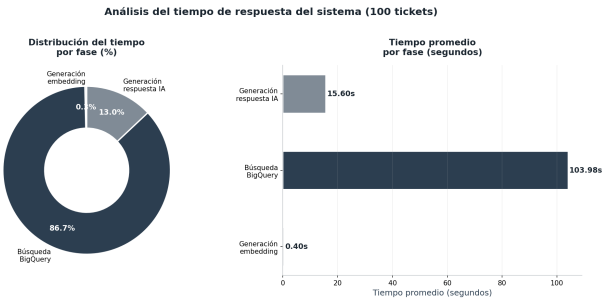


Fig. 7: Análisis del tiempo de respuesta del sistema por fase (100 tickets).

Los resultados muestran que la generación del embedding tarda una media de 0,40 segundos, la generación de la respuesta con Gemini 15,60 segundos, y la búsqueda en BigQuery 103,98 segundos, representando esta última el 86,7 % del tiempo total del ciclo. El tiempo total promedio de resolución de un ticket es de 119,98 segundos, siendo la búsqueda en BigQuery la fase que concentra la mayor parte del tiempo de ejecución

9 Dashboard de visualización

Con el objetivo de hacer el sistema accesible a cualquier usuario sin conocimientos técnicos, se desarrolla una interfaz web interactiva mediante Streamlit que permite introducir un ticket de soporte y obtener una respuesta personalizada de forma visual e intuitiva.

La interfaz se estructura de la siguiente manera:

1. **Barra de estadísticas** — En la parte superior se muestra información general del sistema: el número de tickets históricos disponibles en BigQuery (22.619), correspondientes al conjunto de train utilizado como base de datos histórica, las dimensiones del modelo de embeddings (768), el modelo de IA generativa utilizado (Gemini 2.5) y la base de datos vectorial empleada (BigQuery).

2. **Formulario de entrada** — El usuario introduce el nombre del cliente, describe el problema en un campo de texto libre, selecciona el nivel de prioridad (`low`, `medium` o `high`) y elige el departamento al que pertenece el ticket entre las siguientes opciones: `Technical Support`, `Customer Service`, `Product Support`, `Billing`, `General Inquiry` y `Network Support`.

3. **Botón de envío** — Al pulsar el botón de enviar, el sistema inicia el ciclo completo de resolución automática.

4. **Barra de progreso** — Muestra en tiempo real el estado del proceso, indicando en cada momento en qué fase se encuentra: generación del embedding, búsqueda en BigQuery o generación de la respuesta con Gemini.

5. **Ticket similar encontrado** — Se muestra el asunto del ticket histórico más similar encontrado en BigQuery junto con su valor de similitud coseno, que indica el grado de parecido semántico entre el ticket nuevo y el caso histórico recuperado.

6. **Respuesta generada** — Se muestra la respuesta personalizada generada por Gemini a partir de la solución del ticket histórico más similar, dirigida al cliente por su nombre y con pasos concretos para resolver su problema.

El resultado visual de la interfaz puede consultarse en el Anexo A.3, donde se muestra un ejemplo de su funcionamiento con una consulta de cliente aleatoria.

10 Conclusiones

Este trabajo ha permitido desarrollar un sistema inteligente capaz de automatizar el ciclo completo de resolución de tickets de soporte, integrando técnicas de búsqueda semántica sobre grandes volúmenes de datos almacenados en Google BigQuery con un modelo de inteligencia artificial generativa. El sistema cumple con todos los objetivos planteados al inicio del proyecto: se ha preparado y limpiado un dataset de 28.274 tickets en español, se han generado y almacenado sus representaciones vectoriales en BigQuery, se ha implementado un sistema de búsqueda semántica con filtros por prioridad y cola, se ha integrado Gemini para personalizar las respuestas y se ha desarrollado una interfaz web accesible para cualquier usuario.

Los resultados obtenidos en la evaluación demuestran que el sistema funciona con alta precisión. La similitud coseno promedio de 0,9668 confirma que el modelo de embeddings encuentra tickets históricos semánticamente muy similares a los nuevos tickets de test. La evaluación de la calidad de las respuestas generadas por Gemini, comparadas con las respuestas reales de los tickets, obtiene una puntuación media de 8,85/10, con una tasa de acierto del 93 %, lo que valida la calidad del ciclo completo de resolución automática.

Durante el desarrollo del proyecto surgieron diversas dificultades técnicas. La más significativa fue la traducción del dataset, que requirió aproximadamente dos días de ejecución continua debido al volumen de tickets y a las limitaciones de la API gratuita de Google Translate. También se encontraron problemas con la disponibilidad de los modelos de Gemini en Vertex AI según la región, lo que obligó a utilizar la API directa de Gemini AI Studio como solución alternativa. Por último, la librería utilizada para la generación de embeddings presenta un aviso de retirada previsto para junio de 2026, lo que requerirá una migración a la nueva librería en el futuro.

Como líneas de mejora futura, la más prioritaria es la implementación de un índice vectorial nativo en BigQuery, que

reduciría el tiempo de búsqueda actual de aproximadamente 104 segundos a milisegundos, haciendo el sistema viable en entornos de producción reales. Otras mejoras incluyen la migración a la nueva librería `google-genai` antes de su retirada, la ampliación del dataset con tickets reales de empresa para mejorar la calidad de las respuestas, y la incorporación de más departamentos y niveles de prioridad para cubrir un mayor rango de casos de soporte.

A nivel personal, este proyecto ha supuesto un aprendizaje muy completo en el ámbito de la ingeniería de datos y la inteligencia artificial aplicada. Ha permitido trabajar con tecnologías reales de Google Cloud en un caso de uso empresarial concreto, comprendiendo desde la preparación y limpieza de datos hasta la integración de modelos de lenguaje avanzados y el despliegue de una interfaz web funcional. El resultado final demuestra que la inteligencia artificial puede automatizar procesos empresariales repetitivos de forma eficiente, reduciendo la carga de trabajo humano y ofreciendo soluciones rápidas y personalizadas.

Declaración de uso de inteligencia artificial

En el presente trabajo se han utilizado herramientas de inteligencia artificial como apoyo en tres ámbitos concretos: la mejora de la redacción y coherencia textual, el desarrollo de la interfaz web en Streamlit, y la comprensión y configuración de los servicios de Google Cloud. En todos los casos, el contenido, las decisiones de diseño y el código resultante han sido supervisados, comprendidos y validados por mi. La IA ha actuado en todo momento como herramienta de apoyo, nunca como sustituto del trabajo propio.

Agradecimientos

En primer lugar, quiero agradecer a Ana Oropesa, tutora de este trabajo, por su orientación y consejos a lo largo de todo el proyecto, y por permitirme encaminarlo hacia mis intereses desde el principio.

A mis compañeras de universidad Julia y Martina, por acompañarme durante toda esta etapa y por las clases de Google Cloud que resultaron totalmente imprescindibles para el desarrollo de este trabajo.

A mi mejor amiga Mar, por su ayuda en toda la parte visual del proyecto. Su buen gusto y sus conocimientos han hecho posible que los esquemas y el diseño quedaran lo mejor posible.

Y en especial, a mi familia, por su paciencia infinita escuchándome leer este documento una y otra vez.

Gracias a todos.

Referencias

[1] Atlassian. 2024. What is IT Service Management (ITSM)? [online] Available at: `https://www.atlassian.com/itsm` [Accessed 6 March 2026].

[2] IBM. What is IT Service Management (ITSM)? [online] Available at: `https://www.ibm.com/topics/itsm` [Accessed 6 March 2026].

[3] IBM. What is Generative AI? [online] Available at: `https://www.ibm.com/think/topics/generative-ai` [Accessed 6 March 2026].

[4] Google Cloud. Vector Search Overview. [online] Available at: `https://cloud.google.com/alloydb/docs/ai/vector-search-landing` [Accessed 6 March 2026].

[5] Google Cloud. Vector similarity search for AI applications. [online] Available at: `https://cloud.google.com/vertex-ai/docs/vector-search/overview` [Accessed 6 March 2026].

[6] Google Cloud. BigQuery Documentation and Data Analysis. [online] Available at: `https://cloud.google.com/bigquery/docs` [Accessed 6 March 2026].

[7] IBM. Data preparation for AI and analytics. [online] Available at: `https://www.ibm.com/think/topics/data-preparation` [Accessed 6 March 2026].

[8] IBM. Artificial intelligence and analytics performance metrics. [online] Available at: `https://www.ibm.com/analytics` [Accessed 6 March 2026].

[9] Sommerville, I. 2016. *Software Engineering*. 10th Edition. Pearson.

[10] Kerzner, H. 2017. *Project Management: A Systems Approach to Planning, Scheduling, and Controlling*. 12th Edition. Wiley.

[11] Trello. Trello Guide: Manage Projects and Tasks. [online] Available at: `https://trello.com/guide` [Accessed 6 March 2026].

[12] Zendesk. 2025. Zendesk Customer Service Software. [online] Available at: `https://www.zendesk.es` [Accessed April 2026].

[13] Atlassian. 2025. Jira Service Management. [online] Available at: `https://www.atlassian.com/software/jira/service-management` [Accessed April 2026].

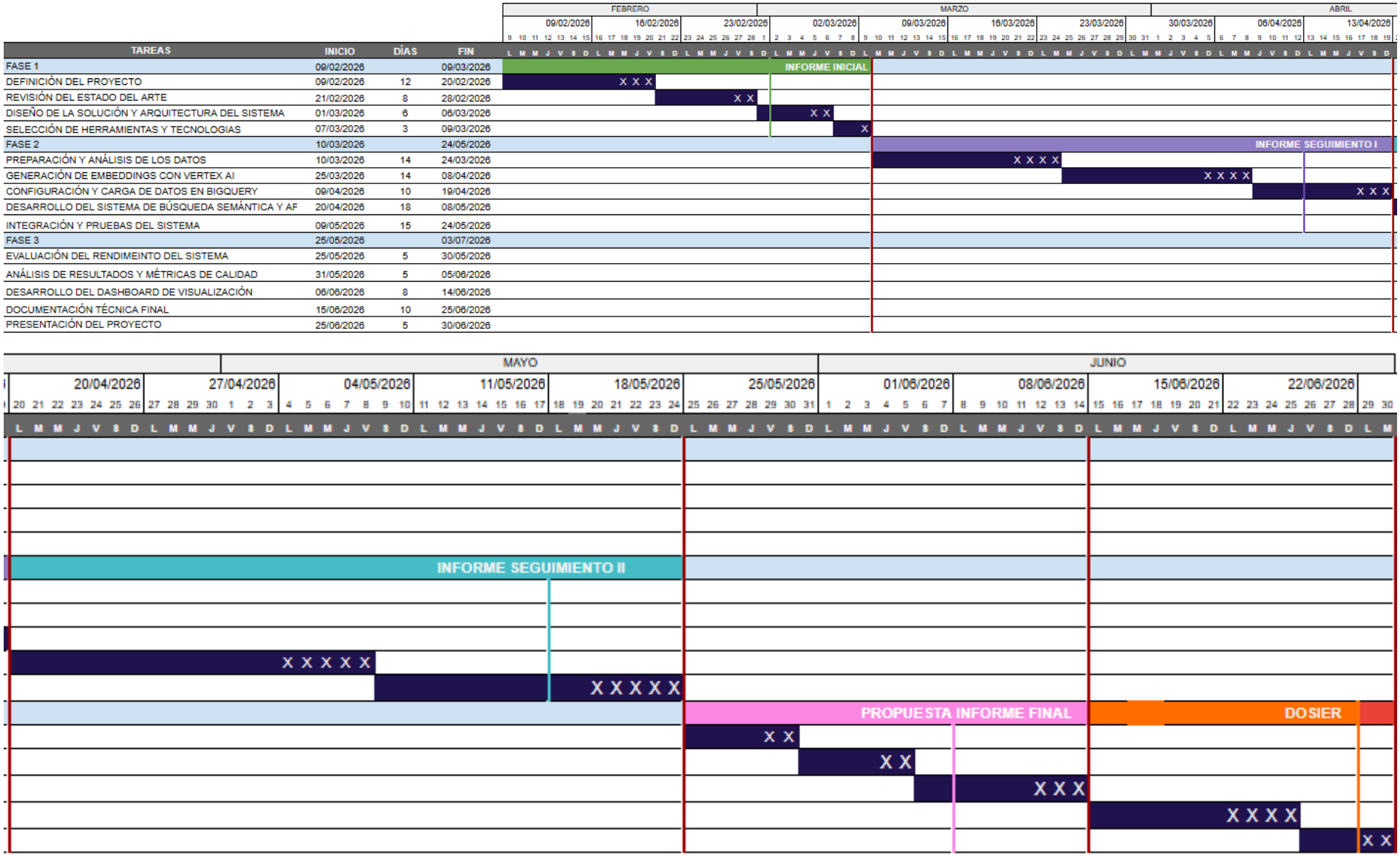
[14] ServiceNow. 2025. IT Service Management. [online] Available at: `https://www.servicenow.com/products/itsm.html` [Accessed April 2026].

[15] Bueck, T. 2024. Multilingual Customer Support Tickets. [online] Available at: https://www.kaggle.com/datasets/tobiasbueck/multilingual-customer-support-tickets [Accessed March 2026].

[16] Google. 2025. Gemini API Documentation. [online] Available at: https://ai.google.dev/gemini-api/docs [Accessed May 2026].

APÉNDICE

A.1. Diagrama de Gantt



A.2. Tablero de Trello

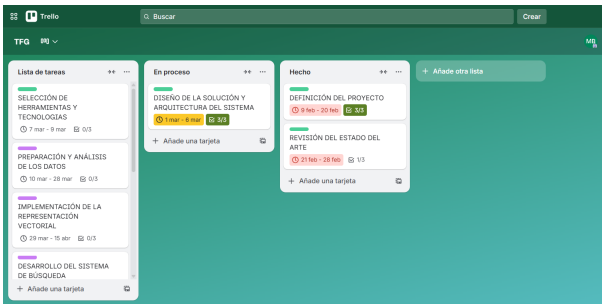


Fig. 8: Tablero de Trello utilizado para la planificación y seguimiento de tareas.

A.3. Interfaz de usuario

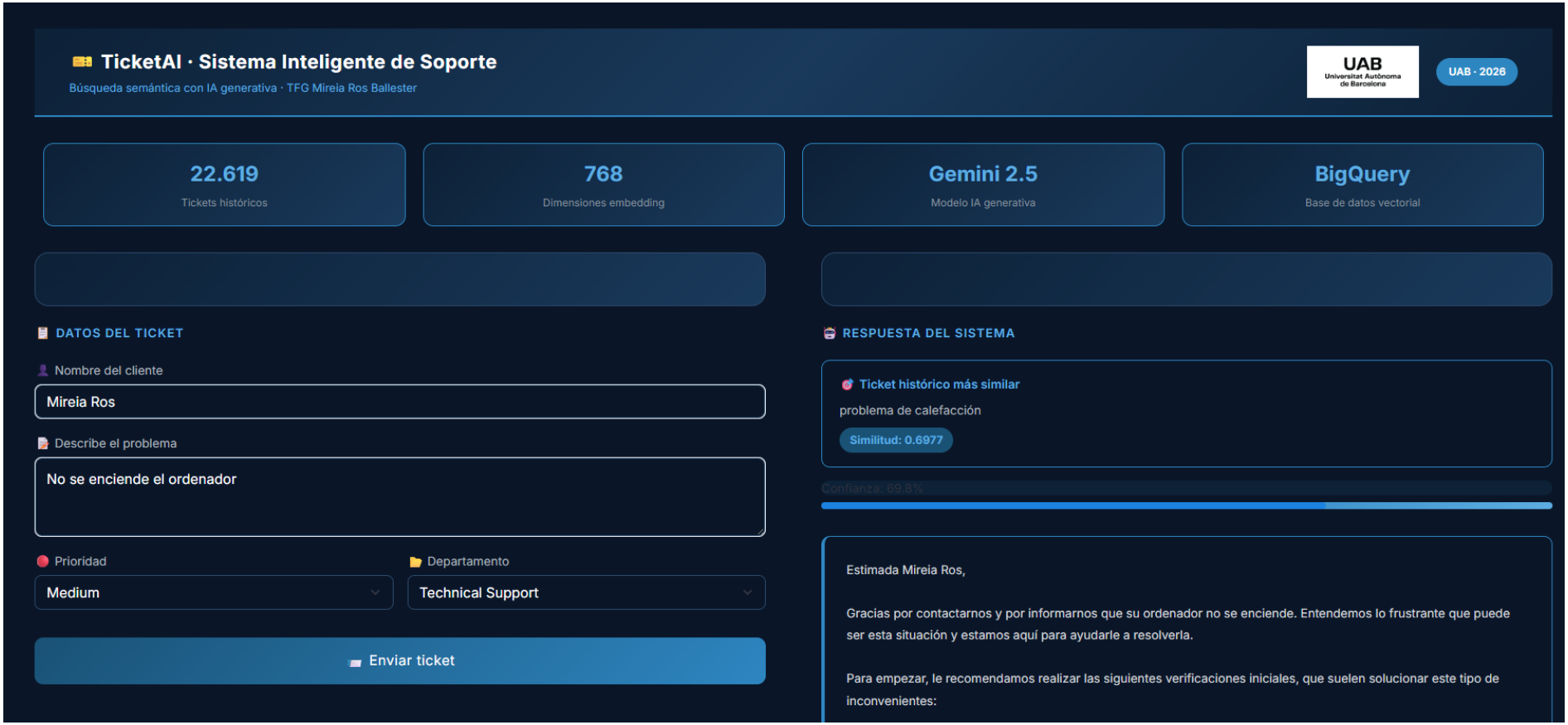


Fig. 9: Vista general de la interfaz web con el formulario de entrada y la respuesta generada.

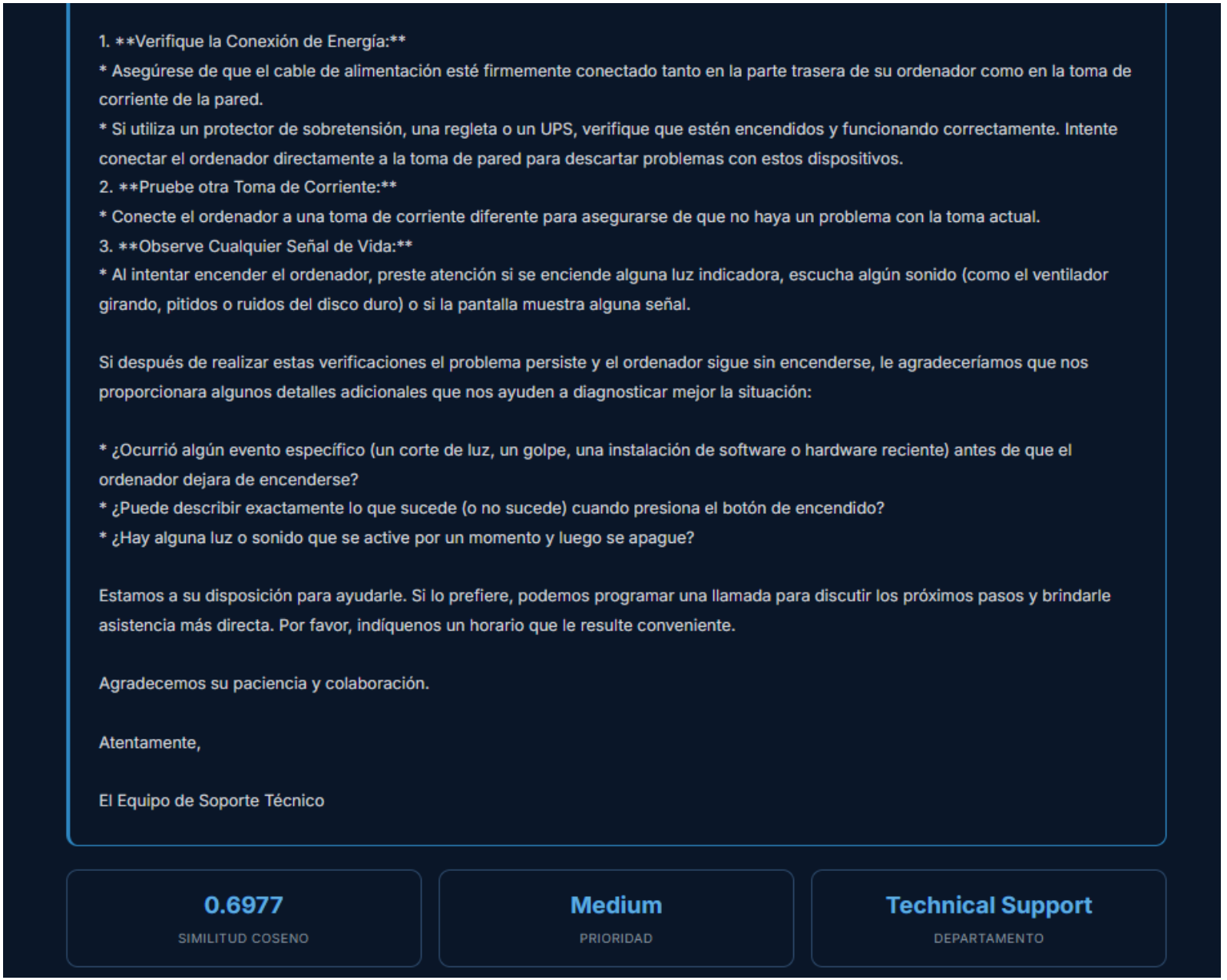


Fig. 10: Detalle de la respuesta personalizada generada por Gemini y métricas de resolución.