# Práctica 7:

## Orquestación de Modelos de IA - Integración con Gemini

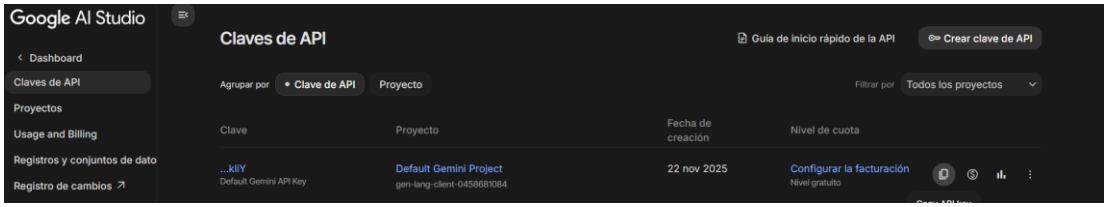
INTEGRACIÓN DE TECNOLOGÍAS Y SERVICIOS INFORMÁTICOS

Daniel Salas Alonso

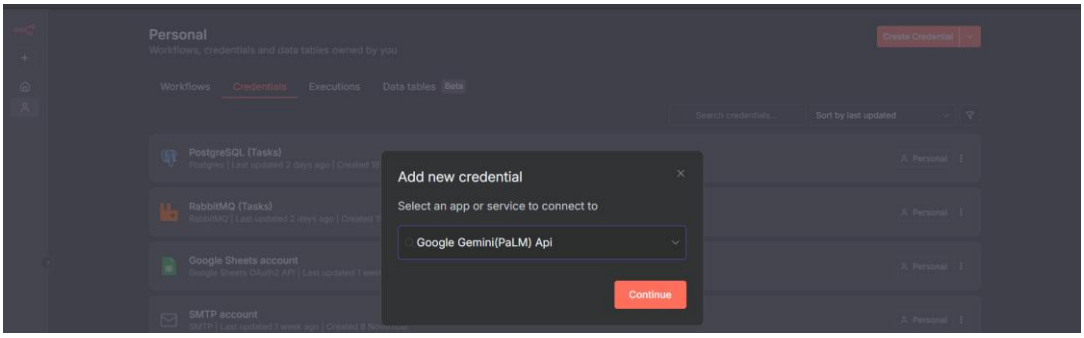
# 3. Desarrollo del Flujo de Trabajo Guiado

## 3.1. Paso 1: Obtener Credenciales de Google AI

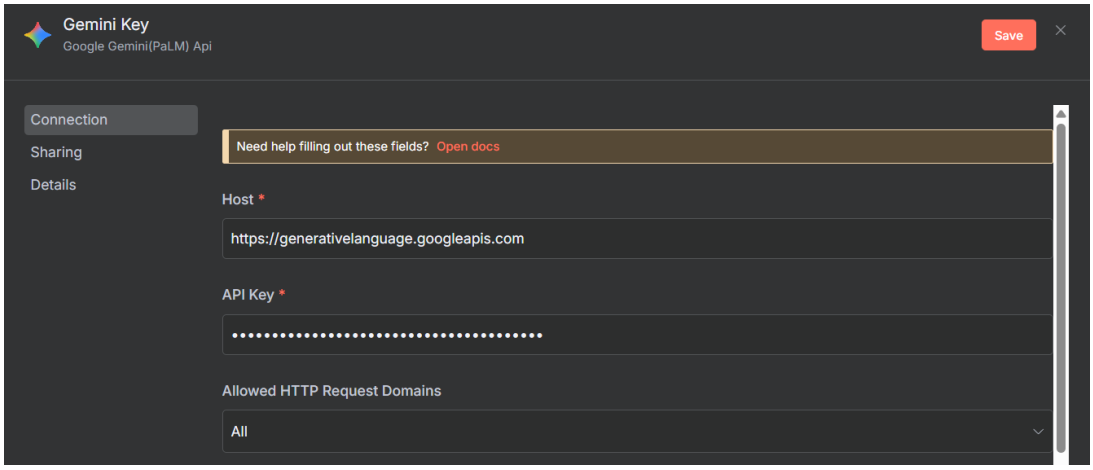
Comenzamos accediendo al panel de Google AI Studio. Donde ya existe una clave de API ("Create API Key") asociada a nuestro proyecto (Default Gemini Project) para permitir la autenticación desde n8n.




Accedemos al panel de credenciales en n8n y seleccionamos "Create Credential". Buscamos y seleccionamos "Google Gemini(PaLM) Api" para configurar la conexión con el servicio de inteligencia artificial.




En la configuración de la credencial, introducimos la API Key obtenida anteriormente en el campo correspondiente y guardamos la credencial para usarla en los flujos posteriores.



## 3.2. Paso 2: Flujo Base - Leer Tareas de PostgreSQL

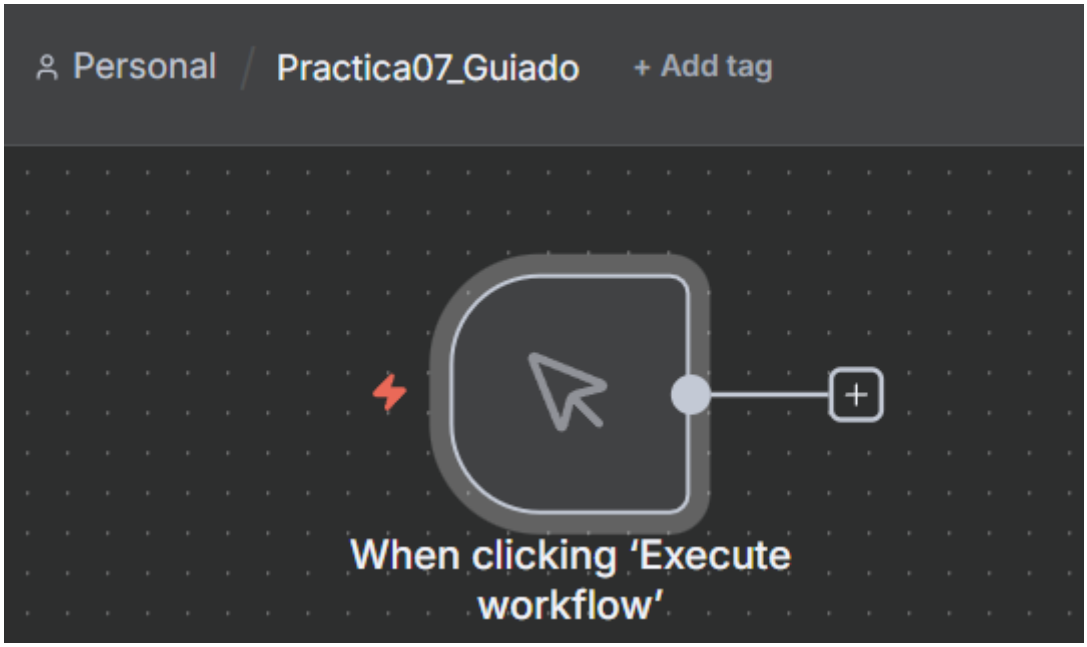
Preparamos el entorno desplegando los microservicios del proyecto Python. Ejecutamos el comando docker-compose up -d en la terminal y verificamos que los servicios se inician correctamente.

```
dsala@TERRAQUE MINGW64 ~/repositorios-master/ITSI/P05/task-manager-service (main)
$ docker-compose up -d
[+] Running 6/6
 ✓ Container task-manager-db              Started                    1.0s
 ✓ Container task-manager-mq              Started                    1.0s
 ✓ Container task-manager-error-handler   Started                    1.2s
 ✓ Container task-manager-worker          Start...                   1.2s
 ✓ Container task-manager-notifier        Sta...                     1.3s
 ✓ Container task-manager-web             Started                    1.3s
```

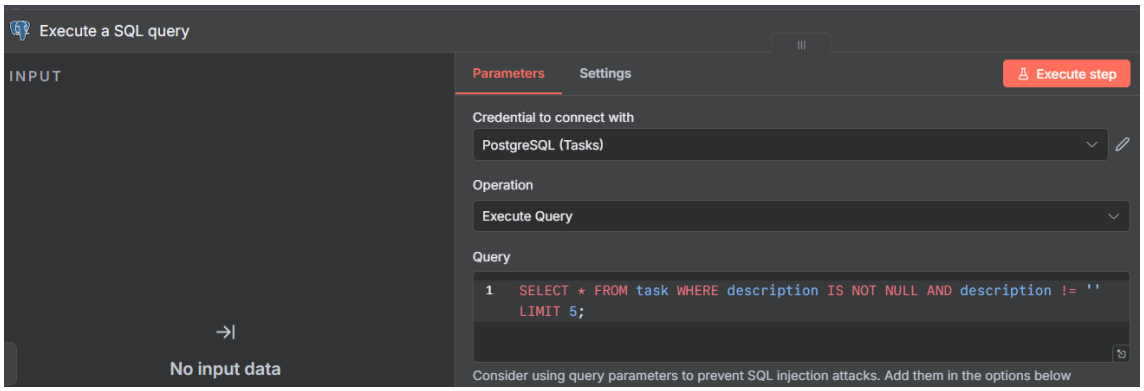
Verificamos que nuestro proyecto n8n este en la misma red que los contenedores de “task-manager-service”.

```
dsala@TERRAQUE MINGW64 ~/repositorios-master/ITSI/P05/task-manager-service (main)
$ docker network connect task-manager-service_default n8n
Error response from daemon: endpoint with name n8n already exists in network task-manag
er-service_default
```

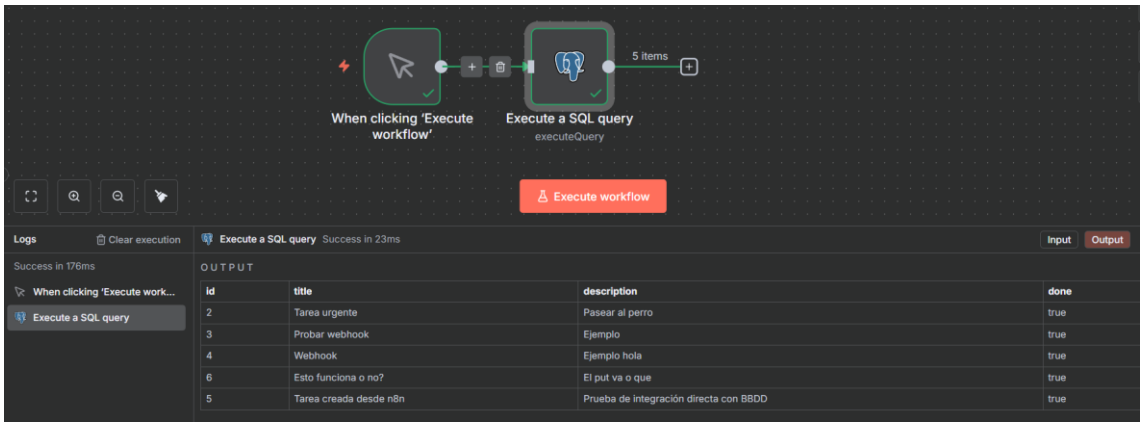
Creamos un nuevo workflow (Practica07_Guiado) y añadimos el nodo inicial "Manual Trigger" para poder disparar el flujo manualmente durante las pruebas.



A continuación, conectamos un nodo PostgreSQL configurado con la operación "Execute Query". Seleccionamos la credencial creada previamente e introducimos la consulta SQL propuesta para recuperar las 5 primeras tareas de la base de datos.

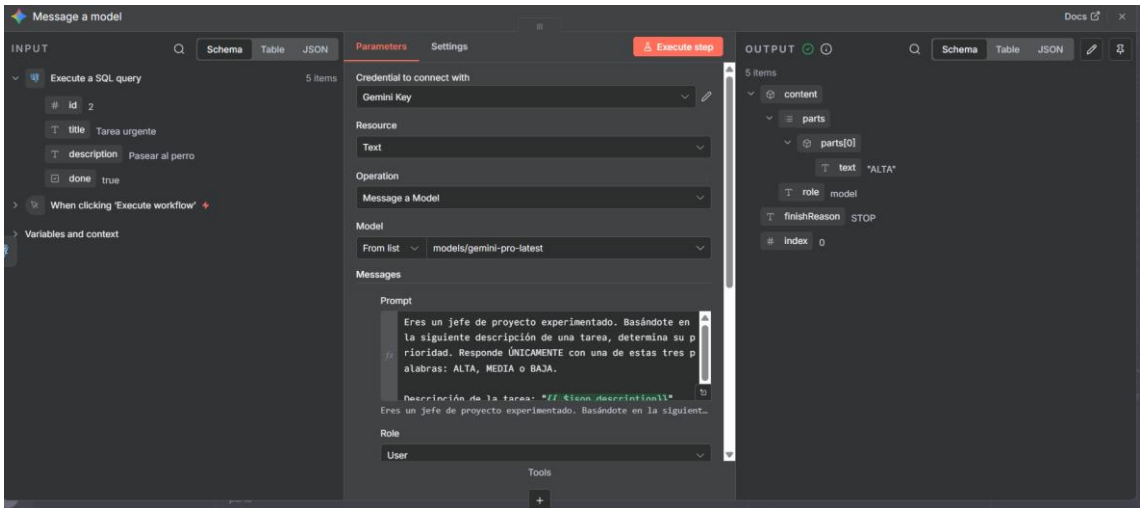



Ejecutamos el flujo y observamos la salida del nodo PostgreSQL. Vemos que nos devuelve un array de 5 ítems con los campos id, title, description y done, confirmando la recuperación de datos exitosa.

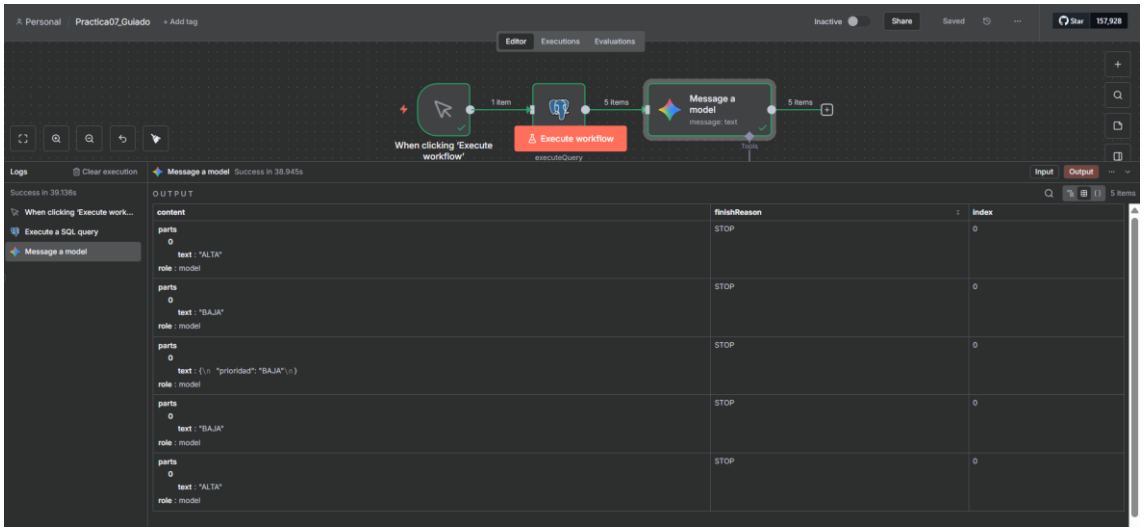


## 3.3. Paso 3: Analizar Tareas con Gemini (Prompt Básico)

Añadimos un nodo "Google Gemini". Seleccionamos la credencial, el modelo e indicamos el recurso Text y la operación “Message a model”. En el campo Prompt redactamos la instrucción para que actúe como un jefe de proyecto y determine la prioridad de la tarea proveniente del nodo anterior.

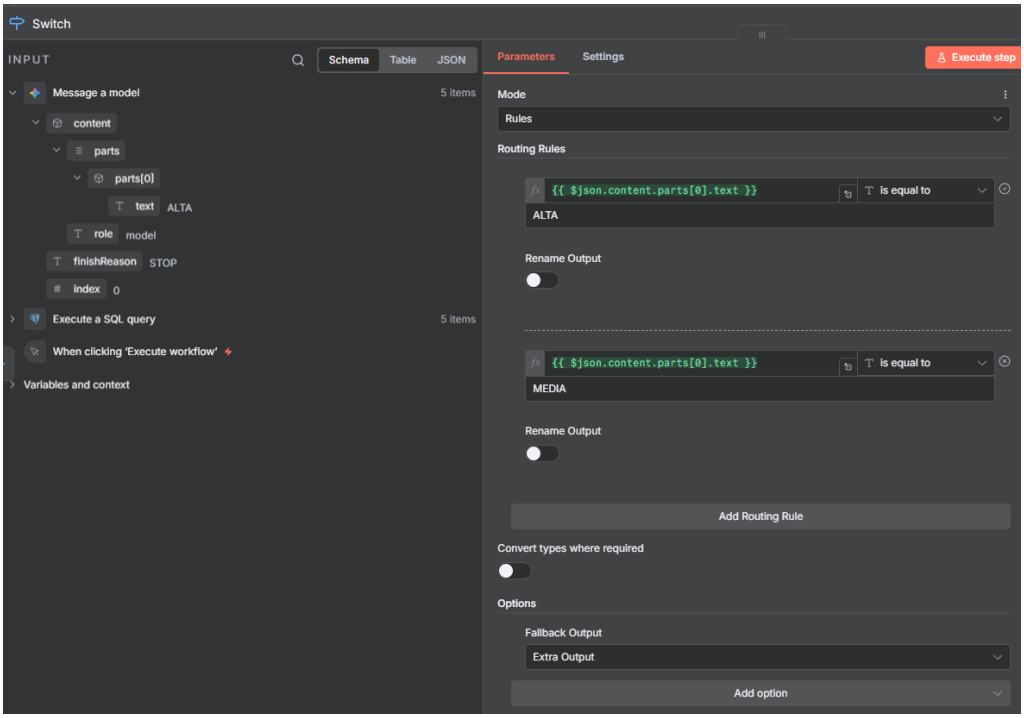



Ejecutamos el nodo y observamos la salida. El modelo ha procesado cada tarea individualmente y ha devuelto una respuesta de texto en el campo content.parts[0].text clasificando la prioridad como "ALTA", "MEDIA" o "BAJA".

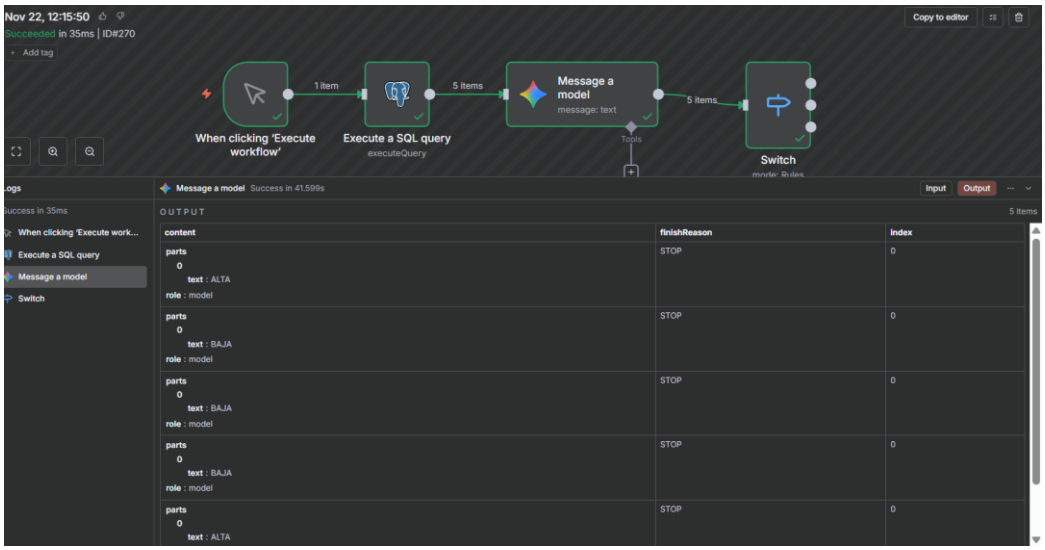


## 3.4. Paso 4: Tomar Decisiones Basadas en la IA

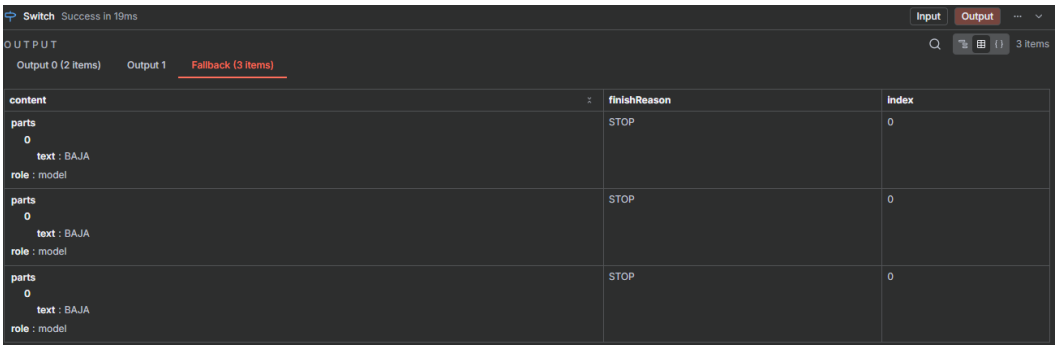
Conectamos un nodo Switch a la salida de Gemini, donde definimos que si la salida del modelo, al procesar la tarea, es igual a "ALTA", el flujo vaya por la salida 0; si es "MEDIA", por la salida 1; y si no es ninguna de las anteriores irá por la rama 2, de esto se encarga el apartado “Fallback Output”, indicando “Extra Output” para que la tarea vaya por una rama diferente en caso de que no cumpla con las condiciones de las ramas anteriores.




Observamos la estructura del flujo con el nodo Switch conectado. Al ejecutar el flujo completo, vemos cómo los ítems se distribuyen por las diferentes ramas según la decisión tomada por la IA.

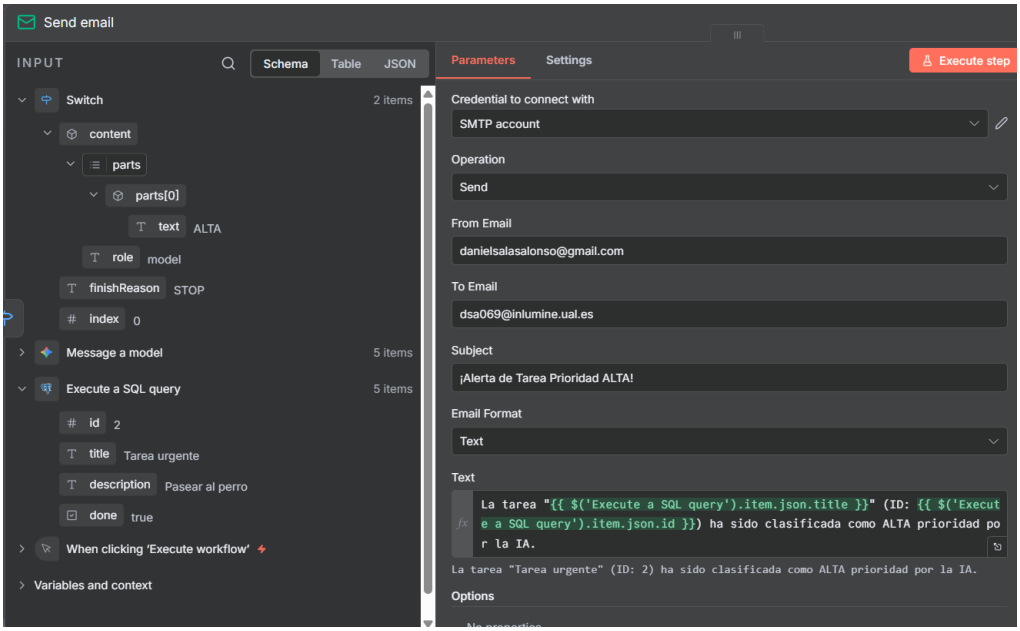


Verificamos la salida del nodo Switch. Observamos que ha filtrado correctamente los ítems, enviando aquellos con prioridad "ALTA" (Output 0) y "BAJA" (Output 1) a sus respectivos caminos.

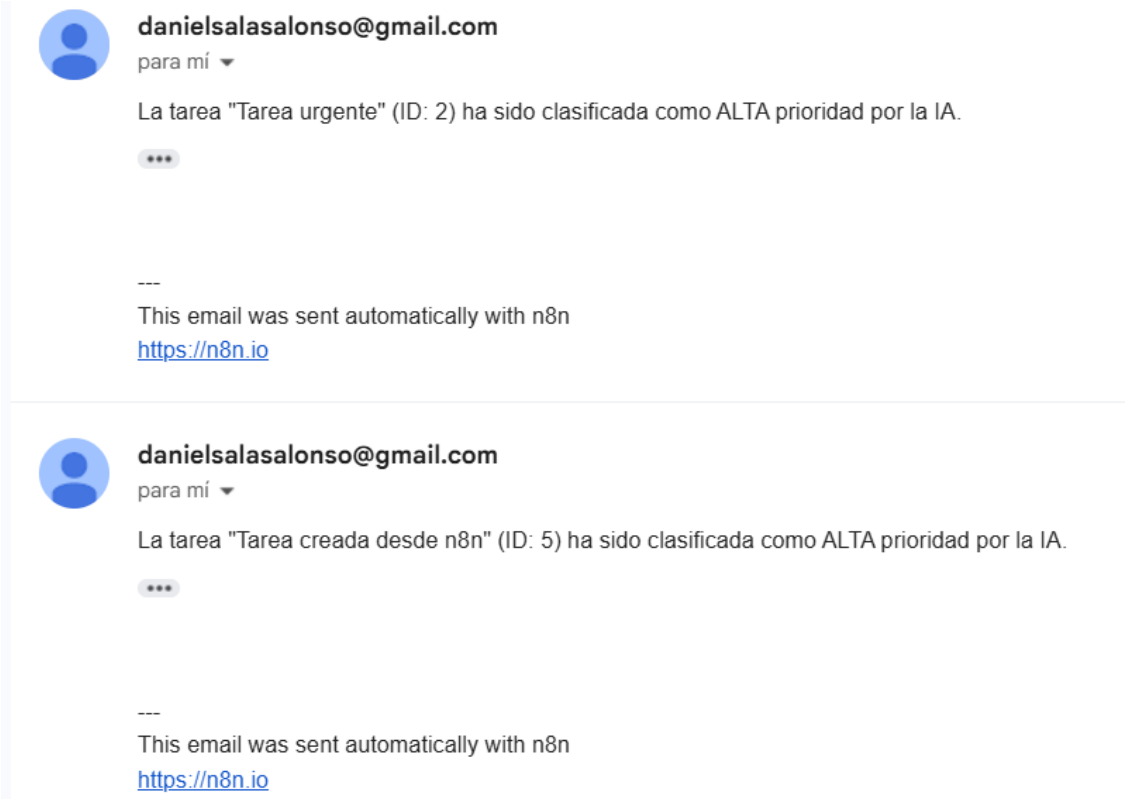

## 3.5. Paso 5: (Simulación) Notificar y Registrar

En la rama de prioridad "ALTA", conectamos un nodo Send Email. Configuramos el correo emisor, destinatario y el asunto indicado. En el cuerpo del mensaje, utilizamos expresiones para incluir dinámicamente el título y el ID de la tarea crítica.



Observamos la ejecución exitosa del flujo de trabajo completo. Los nodos se muestran en verde, indicando que la tarea se leyó de la BBDD, fue analizada por Gemini, filtrada por el Switch como "ALTA" y se envió el correo de notificación.


Verificamos la bandeja de entrada donde recibimos los correos electrónicos. Confirmamos que el asunto y el cuerpo contienen la información correcta de las tareas ("Tarea urgente", "Tarea creada desde n8n") clasificadas como ALTA prioridad por la IA.

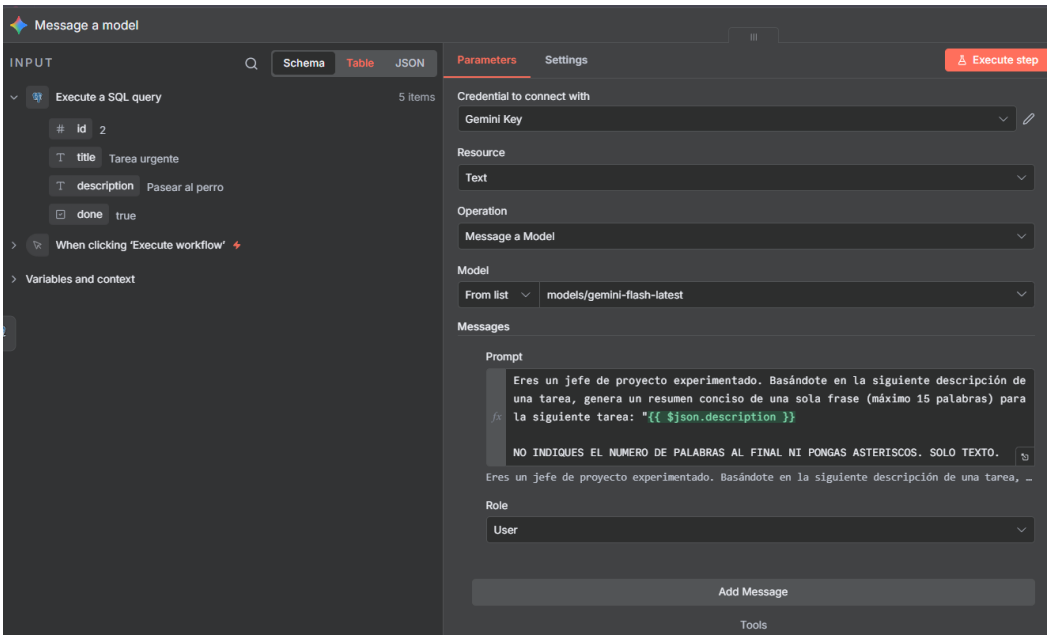


# 4. Ejercicios Propuestos

## 4.1. Ejercicio 1: Generador de Resúmenes (Dificultad: Baja)

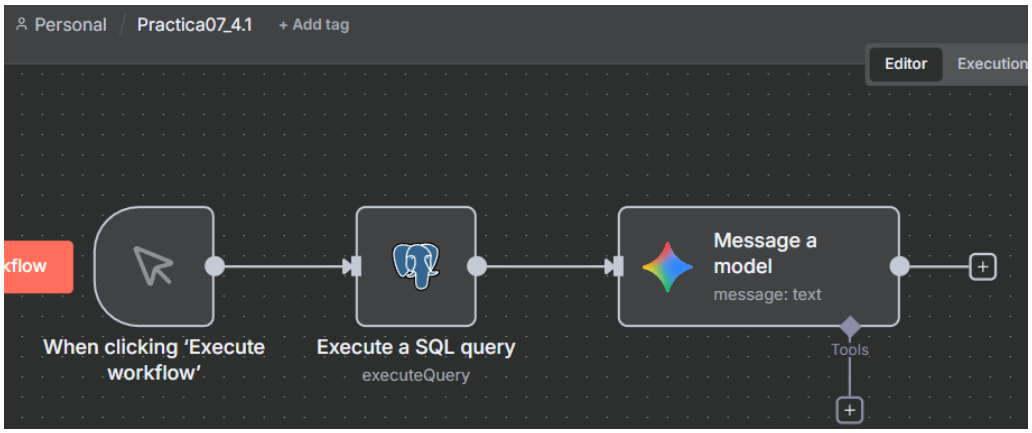
1. *Modifique el Prompt del nodo Google Gemini. El nuevo prompt debe solicitar un resumen de una sola frase para la description de la tarea.*
    - *Ej. Genera un resumen conciso de una sola frase (máximo 15 palabras) para la siguiente tarea: "{{ $json.description }}"*

Creamos una copia del flujo anterior y modificamos el nodo de Gemini. Cambiamos el Prompt para solicitar un resumen conciso basado en la descripción de la tarea. Instruimos al modelo para que no incluya texto adicional.



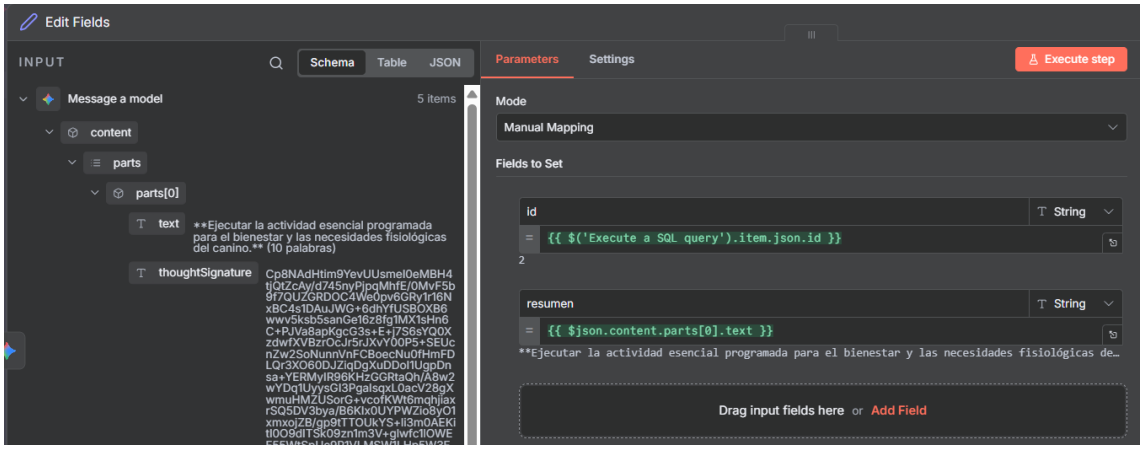

2. *Elimine el nodo Switch.*

Vemos la estructura simplificada del ejercicio. Hemos eliminado el nodo Switch y el nodo Send Email para este caso.

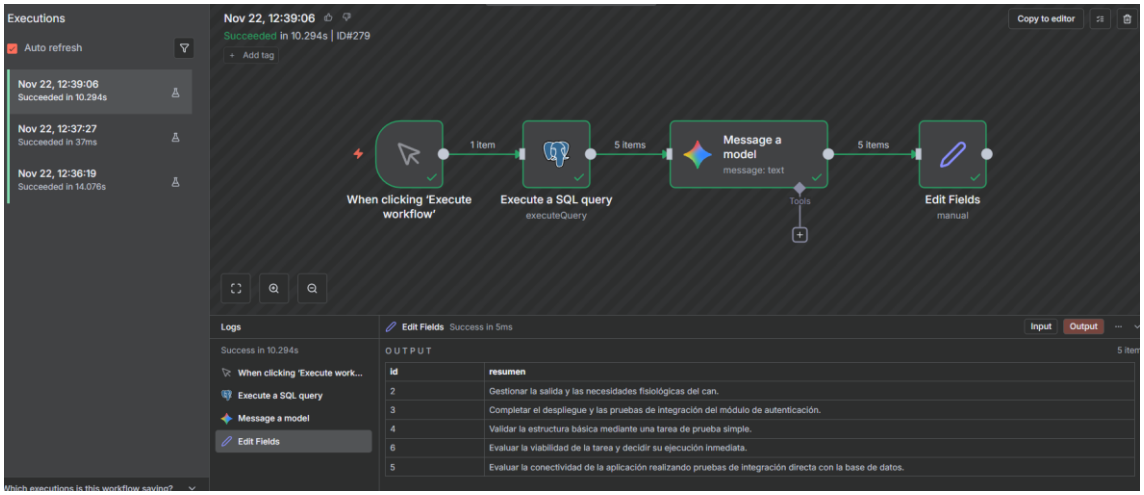


3. *Conecte un nodo Edit Fields (Set) que guarde la salida de Gemini ({{ $json.text }}) en un nuevo campo llamado resumen.*

Añadimos un nodo Edit Fields. Mapeamos el id de la tarea original y creamos un nuevo campo resumen asignándole el valor generado por la IA.



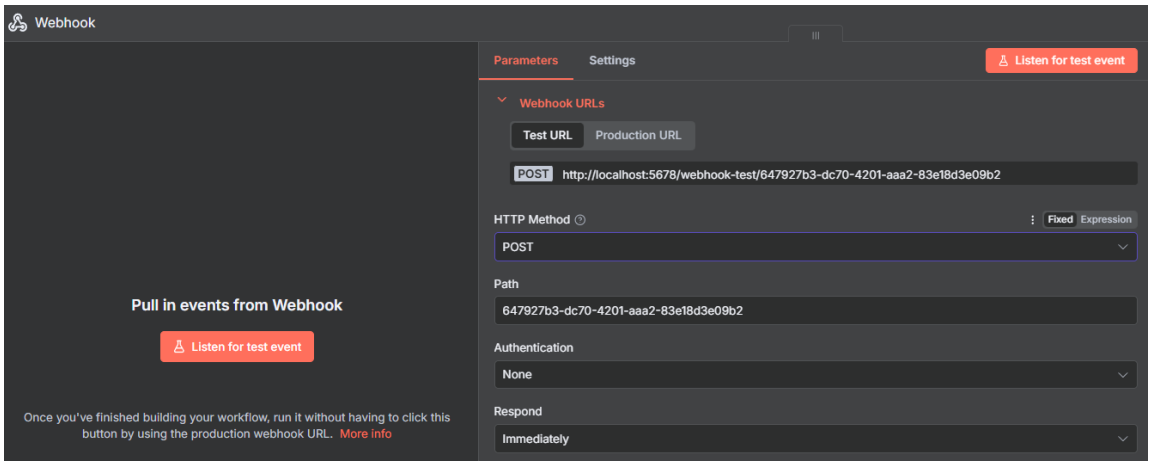

Ejecutamos el flujo y observamos la tabla de resultados. Vemos una lista limpia con el id y el resumen corto generado para cada tarea, confirmando que la IA ha sintetizado correctamente las descripciones.



## 4.2. Ejercicio 2: Extracción de Entidades y Formato JSON (Dificultad: Media)

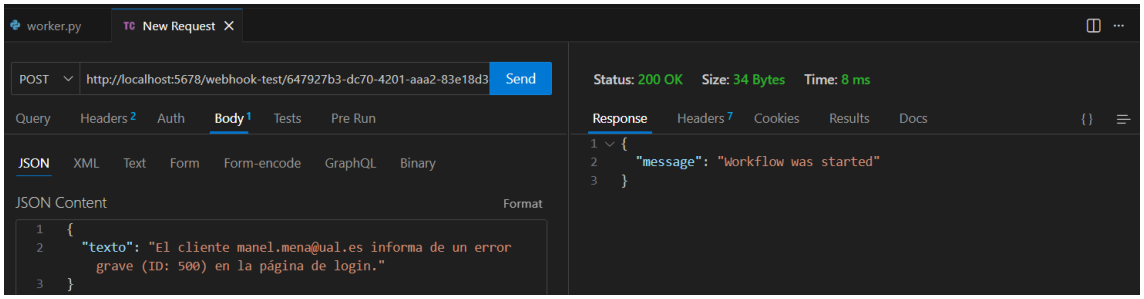
1. *Inicie un nuevo flujo con un Webhook Trigger.*

Iniciamos un nuevo flujo con un nodo Webhook. Configuramos el método HTTP a POST y copiamos la URL de prueba para recibir los datos del reporte de incidencia.



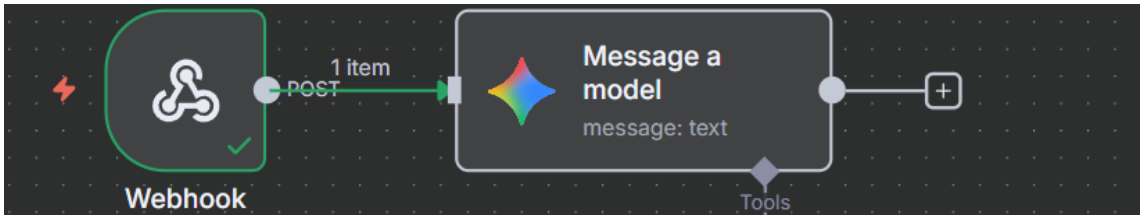

2. *Simule una entrada de datos (ej. con curl o Postman) con un JSON como: {"texto": "El cliente manel.mena@ual.es informa de un error grave (ID: 500) en la página de login."}*

Realizamos una petición enviando un JSON con un campo texto que contiene la descripción del problema propuesto.



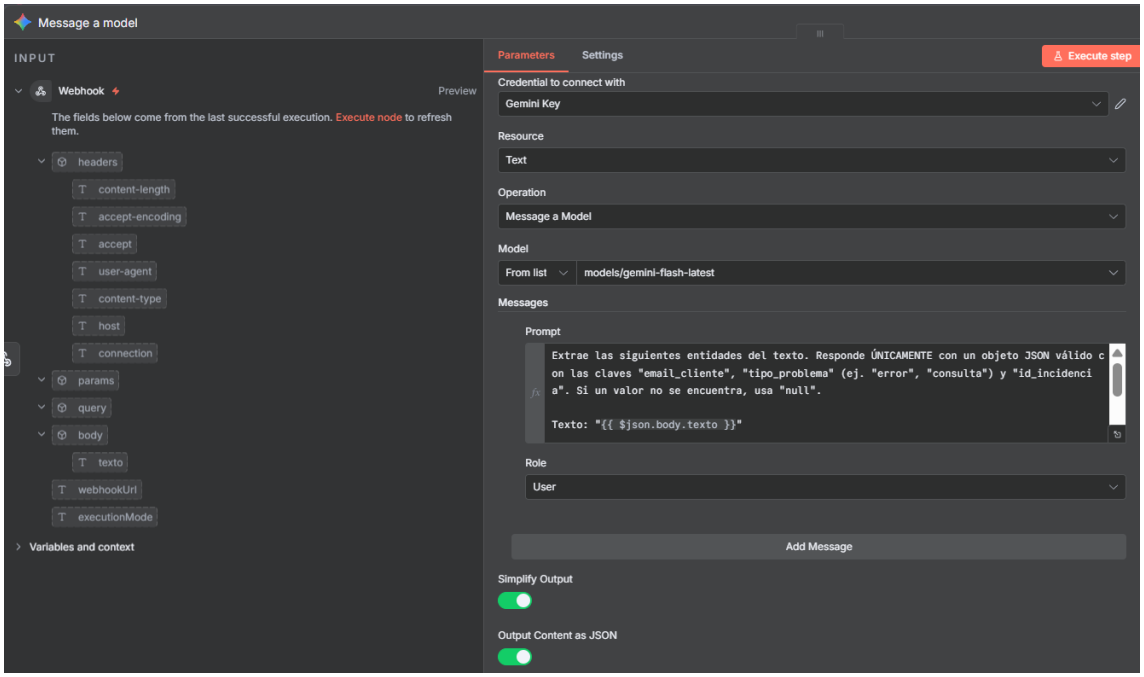

3. *Añada un nodo Google Gemini.*

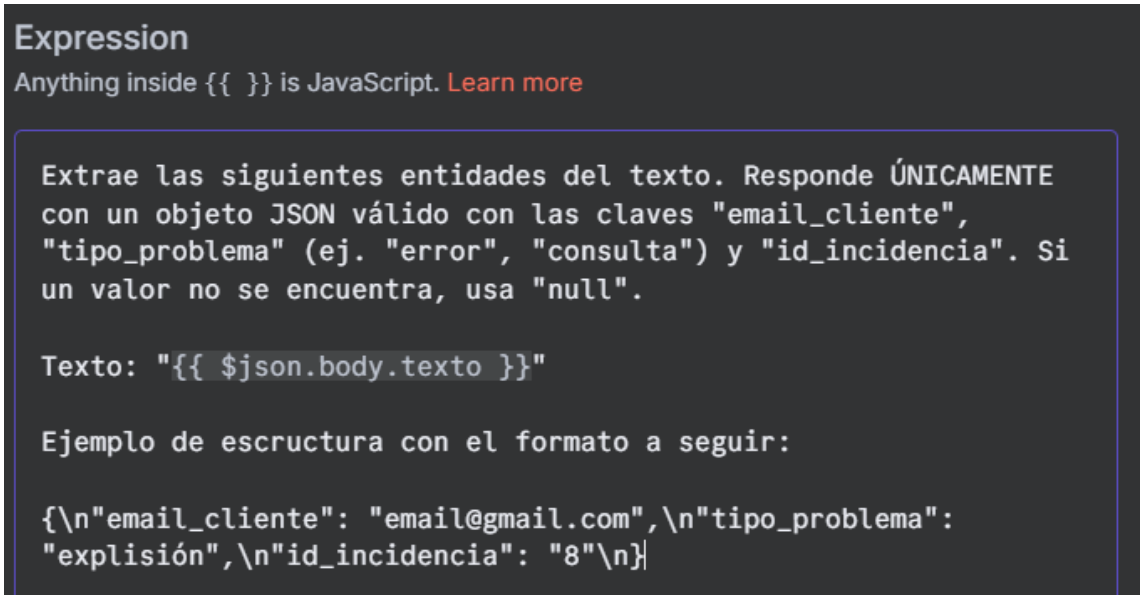
Añadimos el nodo Gemini y observamos la estructura del flujo hasta ahora.



4. *Diseñe un Prompt que le pida a Gemini que extraiga el email, el tipo de problema y la ID, y que responda únicamente en formato JSON. [source,text] ---- Extrae las siguientes entidades del texto. Responde ÚNICAMENTE con un objeto JSON válido con las claves "email_cliente", "tipo_problema" (ej. "error", "consulta") y "id_incidencia". Si un valor no se encuentra, usa "null".*

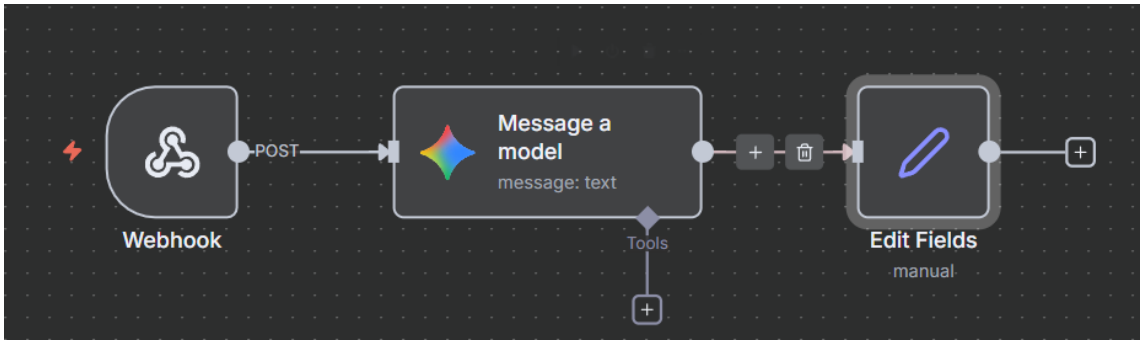
Configuramos el Prompt de Gemini. Le pedimos que extraiga entidades ("email_cliente", "tipo_problema", "id_incidencia") y que responda ÚNICAMENTE con un objeto JSON válido. Activamos la opción "Output Content as JSON" para que devuelva siempre un resultado, que, aunque en texto plano, tenga estructura JSON.



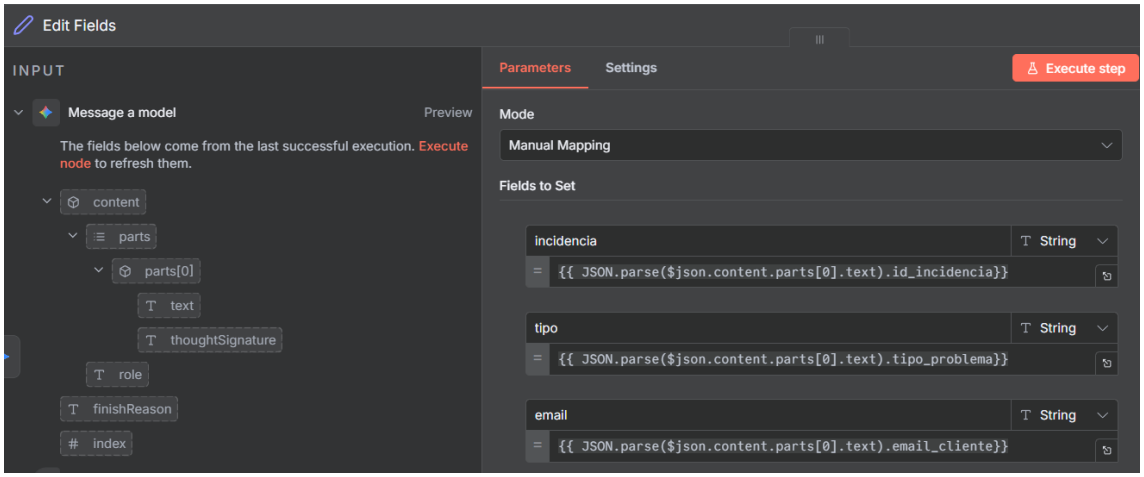
5. *Añada un nodo `Edit Fields (Set)` después de Gemini.*

Añadimos el nodo Edit Fields y observamos la estructura del flujo hasta ahora.



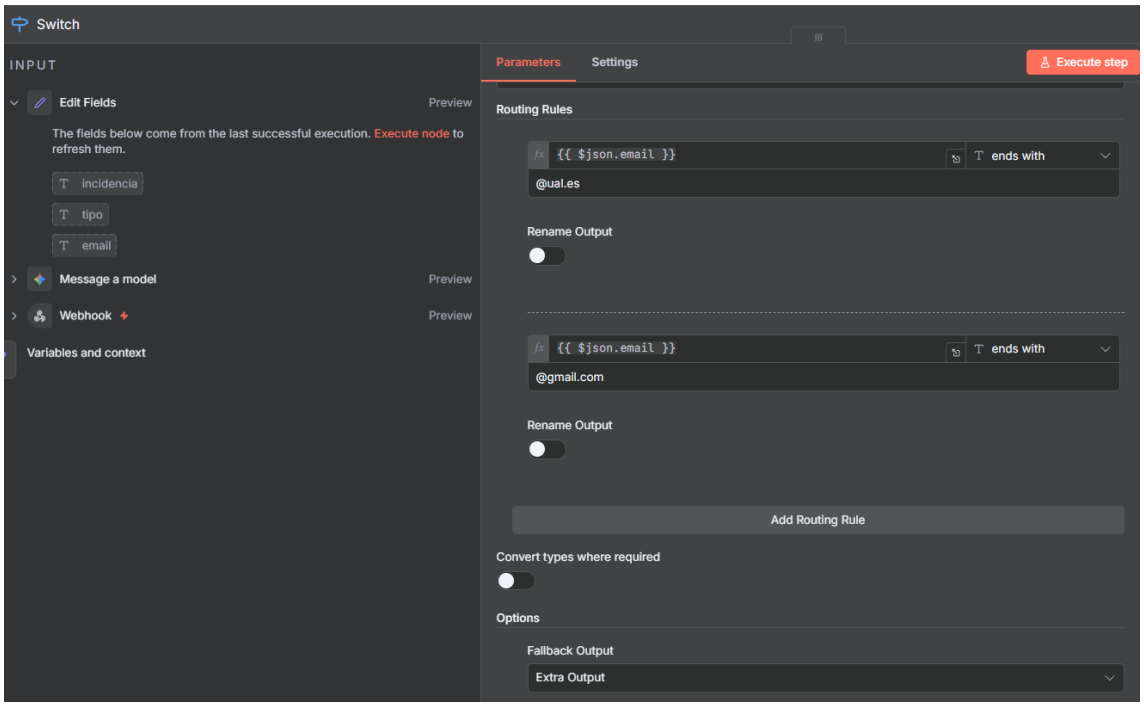

6. *Configure el nodo para *parsear* (analizar) el texto JSON de la IA y crear campos separados:*

En el nodo Edit Fields, utilizamos expresiones “{{ JSON.parse(...) }}” para convertir la respuesta de texto plano de Gemini en campos estructurados (incidencia, tipo, email) utilizables por n8n.

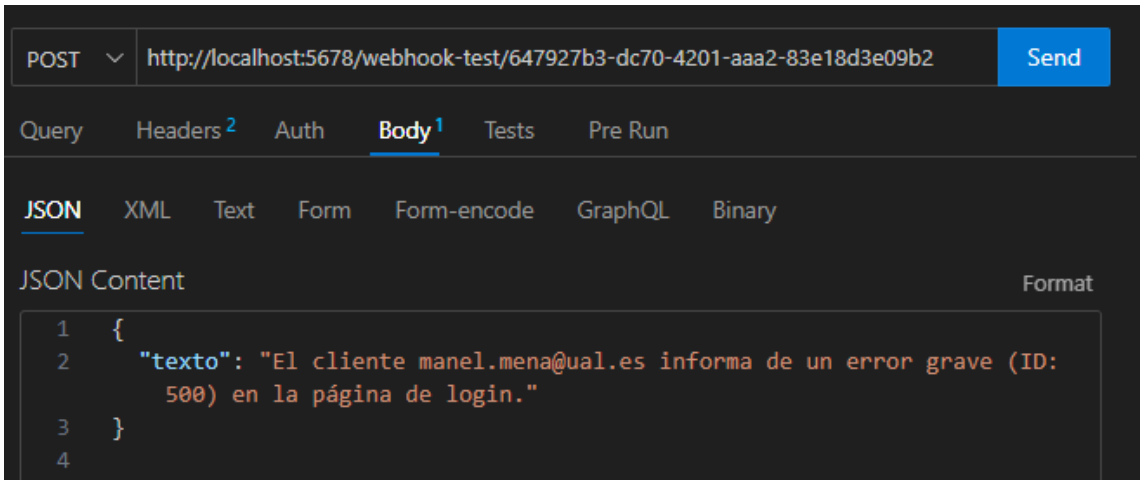


7. *Añada un nodo `Switch` que enrute el flujo basándose en el campo `email` (ej. si contiene `@ual.es`).*

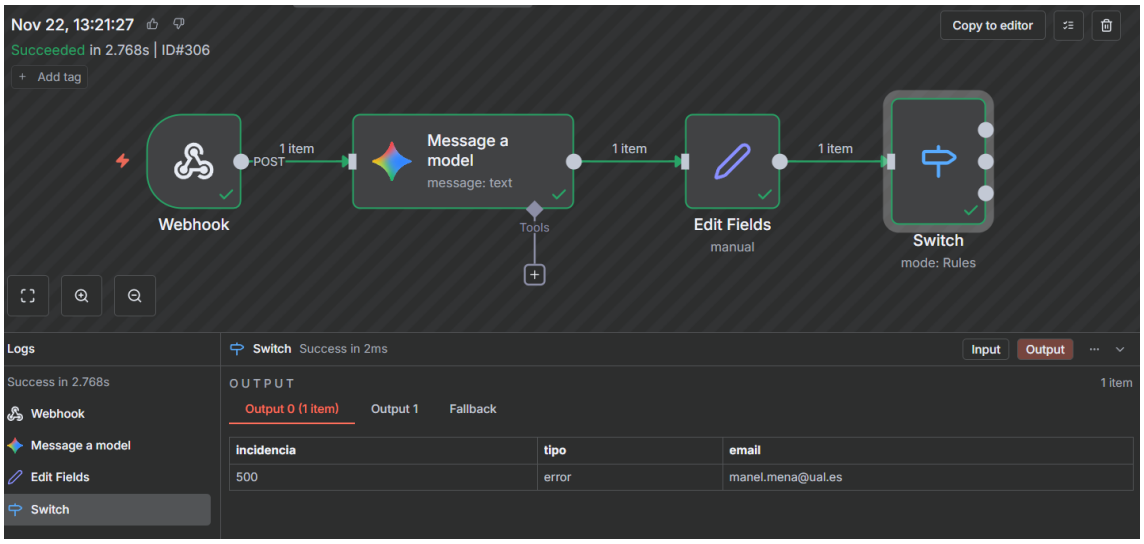
Configuramos el nodo Switch para enrutar basándonos en el dominio del correo. Establecemos las siguientes reglas: si el campo email termina con (ends with) “@ual.es”, el flujo irá por la salida 0; si termina con “@gmail.com” irá por la salida 1; y si no posee ninguna de estas terminaciones irá por la rama por defecto 2, creada con “Fallback Output”.



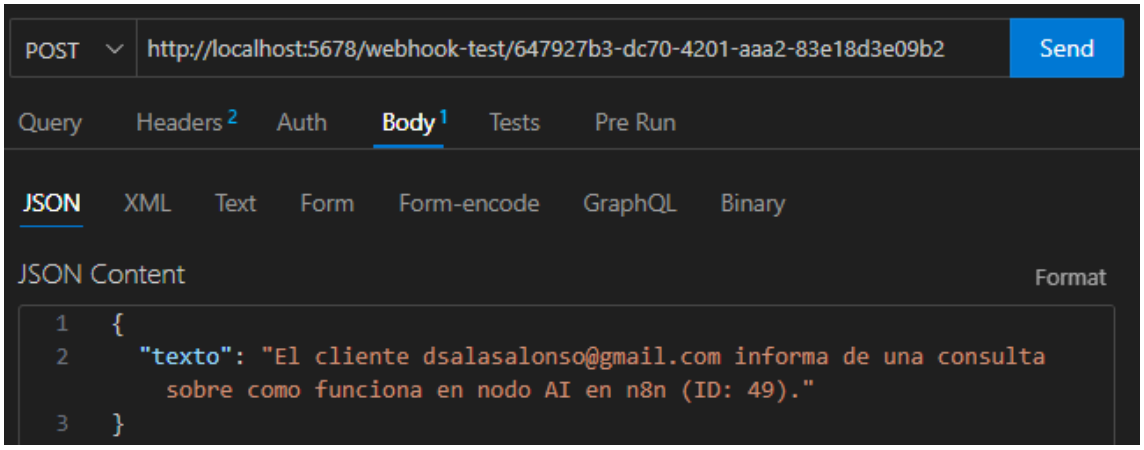

Realizamos la primera prueba enviando una petición al Webhook. En el JSON enviamos un texto que simula un error grave con el ID 500, reportado por el cliente manel.mena@ual.es. El objetivo es verificar si el flujo detecta correctamente el dominio corporativo, el ID y el tipo de petición.



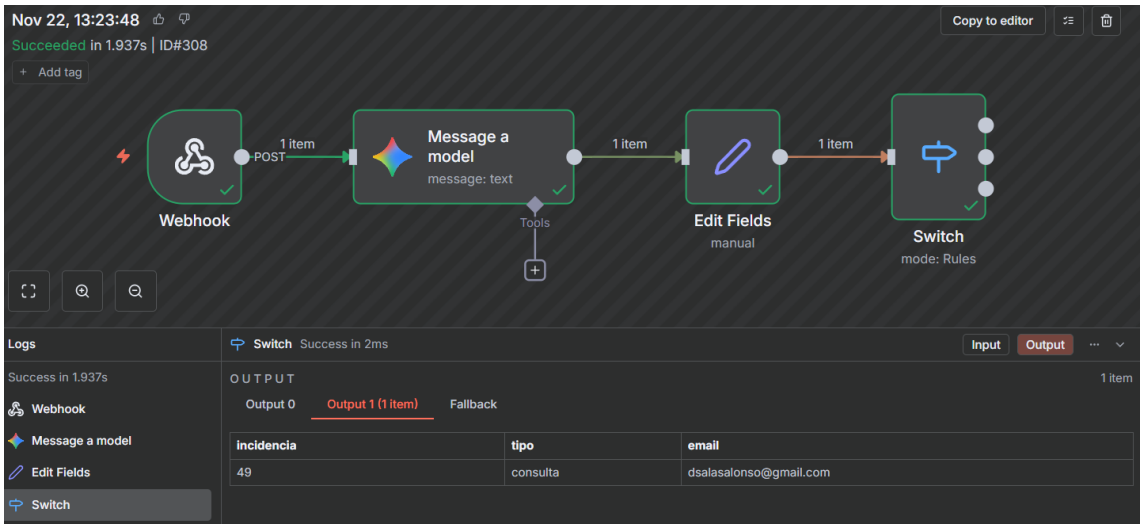
Observamos la ejecución del nodo Switch. Vemos que el flujo se ha dirigido correctamente por la Output 0. Esto confirma que el nodo ha validado que el correo extraído (manel.mena@ual.es) cumple la condición de terminar en @ual.es. Además, verificamos que Gemini ha extraído correctamente la incidencia: 500 y el tipo: error.



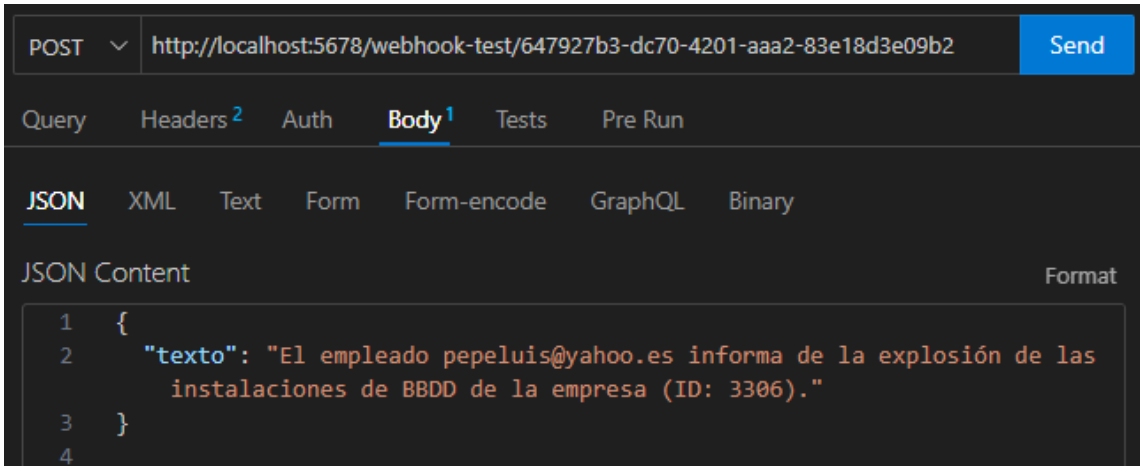

Para la segunda prueba, modificamos los datos del JSON. Ahora enviamos una consulta sobre el funcionamiento de la IA con el ID 49, proveniente del correo dsalasalonso@gmail.com.



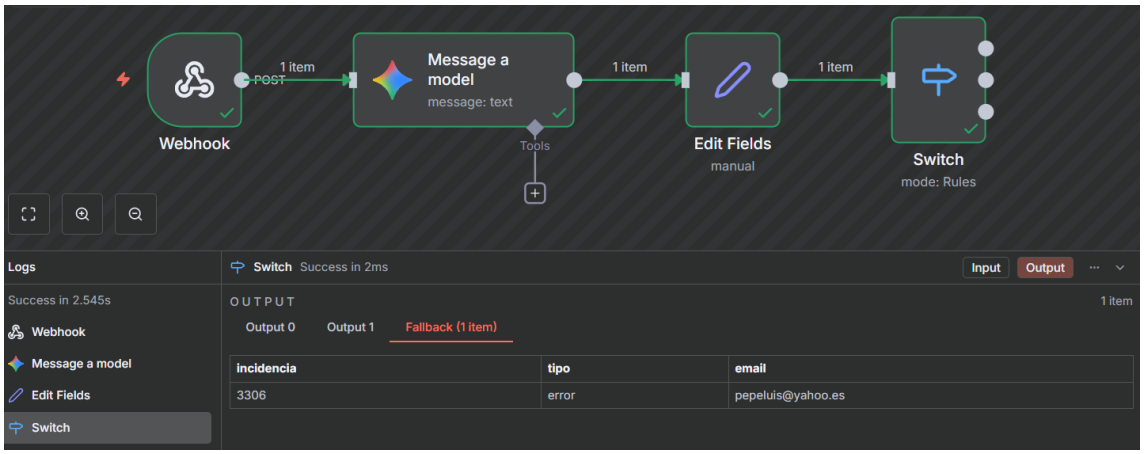
Verificamos la salida en n8n. En este caso, el nodo Switch ha enrutado el flujo por la Output 1. Esto indica que la regla definida para los correos que terminan en @gmail.com ha funcionado correctamente. Los datos extraídos (incidencia: 49, tipo: consulta) también son correctos y están listos para ser procesados por esta rama específica.




Finalmente, realizamos una petición enviando un reporte de una explosión en las instalaciones con ID 3306, proveniente de un empleado con correo pepeluis@yahoo.es.



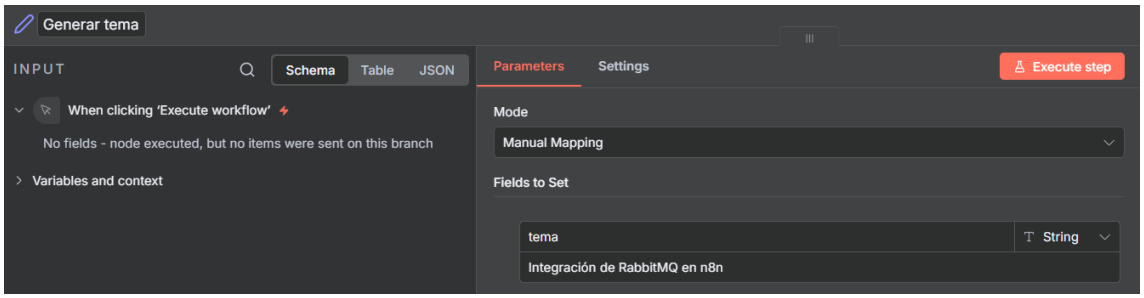
Observamos que el nodo Switch ha dirigido el flujo hacia la rama Fallback. Esto sucede porque el dominio @yahoo.es no cumple ninguna de las condiciones anteriores. A pesar de ello, la extracción de entidades mediante IA ha sido exitosa, identificando la incidencia: 3306 y el tipo: error dentro del informe.



## 4.3. Ejercicio 3: Cadena de IA (AI Chain) (Dificultad: Alta)

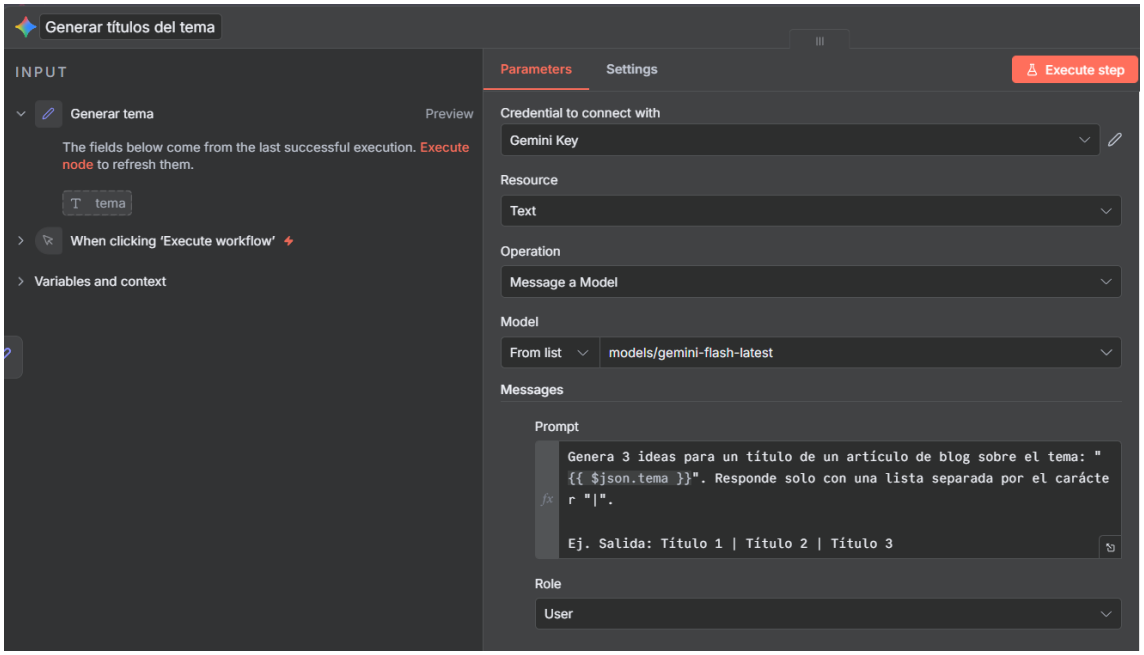
1. *Inicie con un Manual Trigger + Edit Fields (Set). Defina un campo tema (ej. {"tema": "Integración de RabbitMQ en n8n"}).*

Iniciamos este nuevo flujo con un nodo Manual Trigger, seguido de un Edit Fields donde definimos una variable “tema” con el valor "Integración de RabbitMQ en n8n".



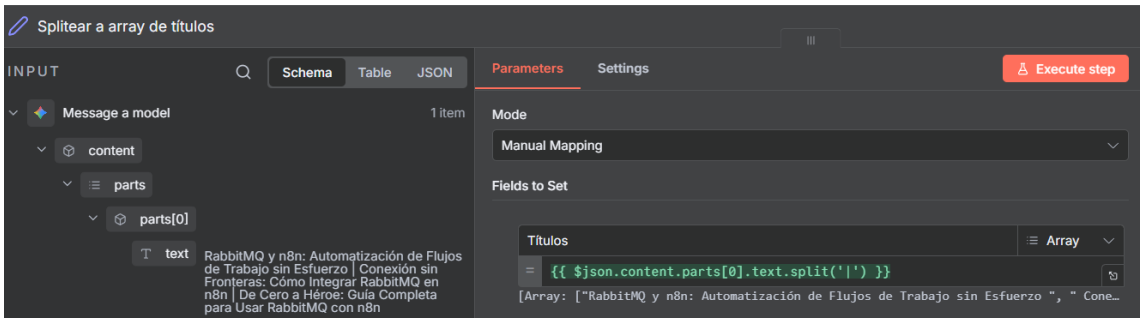

2. *Gemini Node 1 (Brainstorm):*
    - *Prompt: Genera 3 ideas para un título de un artículo de blog sobre el tema: "{{ $json.tema }}". Responde solo con una lista separada por el carácter "|".*
    - *Ej. Salida: Título 1 | Título 2 | Título 3*

Configuramos el primer nodo Gemini. Establecemos las credenciales y el modo “Message a Model” .El Prompt solicita generar 3 ideas para títulos de blog sobre el tema definido anteriormente, separados por el carácter |.



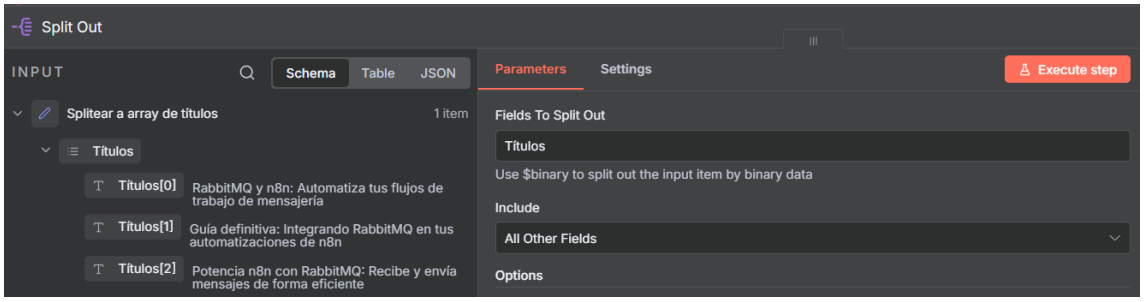
3. *Use un nodo Edit Fields (Set) o Code para dividir (split) la respuesta por el | y convertirla en un array. (Pista: {{ $json.text.split('|') }})*

A continuación, utilizamos un nodo Edit Fields para transformar la cadena de texto en un array. Usamos la expresión {{ $json.content.parts[0].text.split('|') }} para dividir la respuesta y configuramos que la salida debe estar en formato Array.



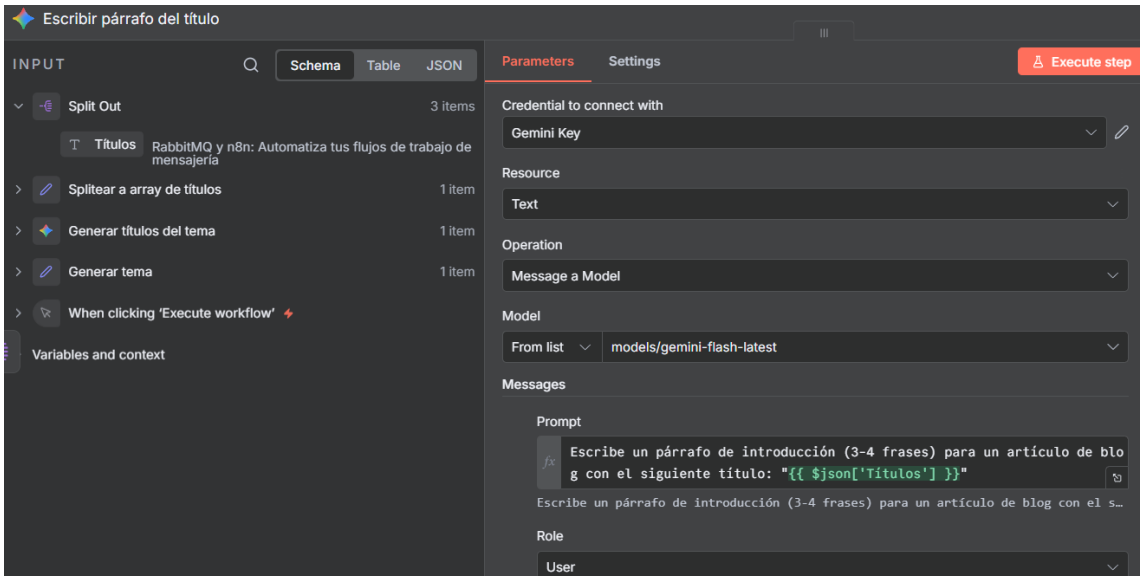

4. *Use un nodo Split Out (P2) para dividir el array en ítems individuales.*

Conectamos un nodo Split Out. Lo configuramos para iterar sobre el array de Títulos, separando cada título en un ítem individual para su procesamiento secuencial.



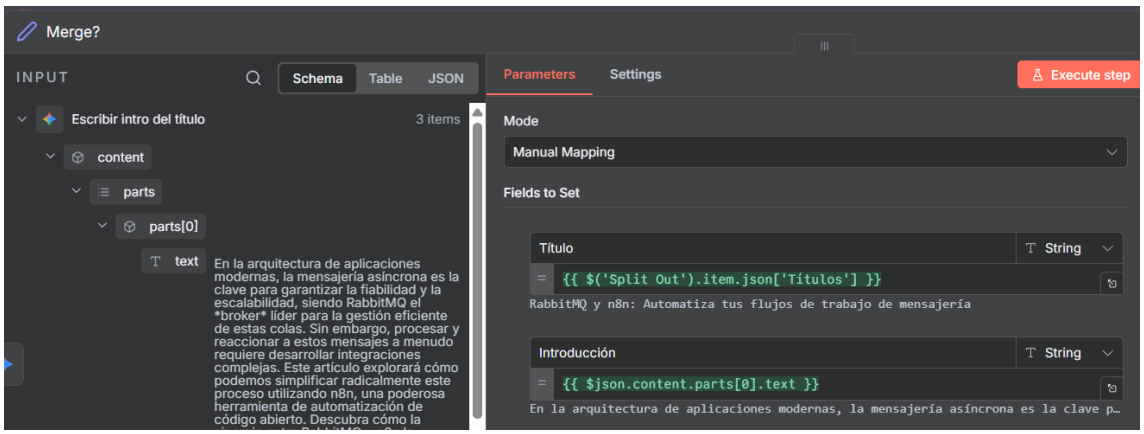
5. *Gemini Node 2 (Expand):*
    - *Este nodo recibirá cada título individualmente.*
    - *Prompt: Escribe un párrafo de introducción (3-4 frases) para un artículo de blog con el siguiente título: "{{ $json.value }}" (Suponiendo que Split Out genera el campo value).*

Añadimos un segundo nodo Gemini. Esta vez, el Prompt solicita escribir un párrafo de introducción para el artículo basándose en el título recibido del nodo anterior. Este nodo se ejecutará tantas veces como títulos haya en el array.

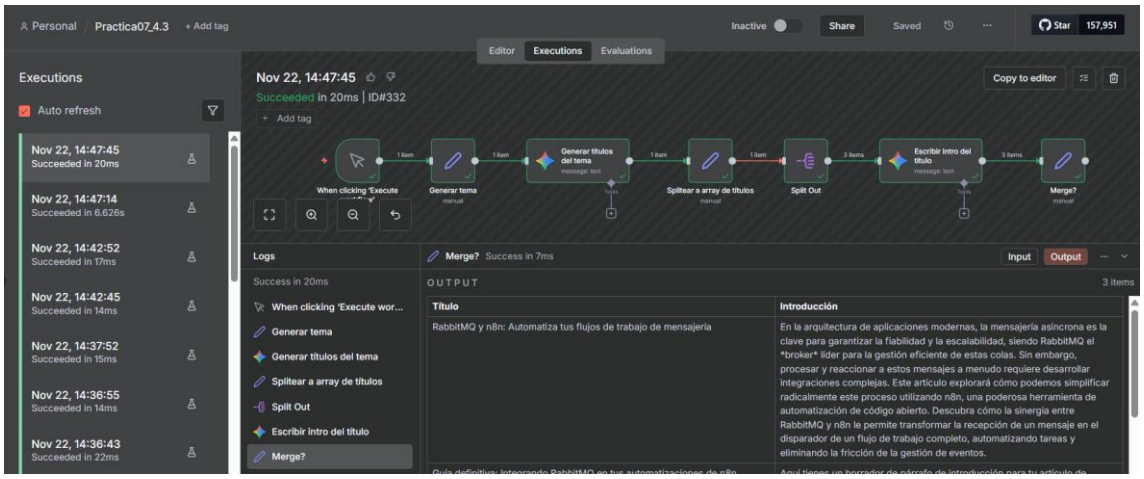



6. *Conecte un nodo Merge (P3) al final para recopilar los 3 párrafos generados.*

Al final del flujo, utilizamos un nodo Edit Fields para estructurar la salida final. Como el flujo es lineal para cada ítem generado por el Split Out, no necesitamos fusionar ramas. En su lugar, utilizamos este nodo para limpiar y formatear el objeto JSON final, asignando el "Título" (recuperado del contexto del Split Out) y la "Introducción" (generada por el segundo nodo Gemini) en un formato legible.



Observamos la ejecución completa de la cadena de IA. Vemos cómo el flujo genera el tema, crea múltiples títulos, los divide y genera un resumen específico para cada uno, resultando en 3 ítems finales con título e introducción listos para usar.





```
[
{
"Título": "RabbitMQ y n8n: Automatiza tus flujos de trabajo de mensajería ",
"Introducción": "En la arquitectura de aplicaciones modernas, la mensajería asíncrona es la clave para garantizar la fiabilidad y la escalabilidad, siendo RabbitMQ el *broker* líder para la gestión eficiente de estas colas. Sin embargo, procesar y reaccionar a estos mensajes a menudo requiere desarrollar integraciones complejas. Este artículo explorará cómo podemos simplificar radicalmente este proceso utilizando n8n, una poderosa herramienta de automatización de código abierto. Descubra cómo la sinergia entre RabbitMQ y n8n le permite transformar la recepción de un mensaje en el disparador de un flujo de trabajo completo, automatizando tareas y eliminando la fricción de la gestión de eventos."
},
{
"Título": "Guía definitiva: Integrando RabbitMQ en tus automatizaciones de n8n ",
"Introducción": "Aquí tienes un borrador de párrafo de introducción para tu artículo de blog:\n\nRabbitMQ se ha consolidado como el sistema de mensajería de referencia para manejar tareas asíncronas y distribuir cargas de trabajo, garantizando que ninguna tarea crucial se pierda. Si utilizas n8n para orquestar tus flujos de trabajo de automatización, integrar RabbitMQ puede desbloquear un nuevo nivel de escalabilidad y fiabilidad, especialmente cuando manejas un alto volumen de eventos. Esta guía definitiva te mostrará paso a paso cómo conectar, enviar y consumir mensajes de RabbitMQ directamente dentro de tus *workflows* de n8n, transformando la forma en que tus automatizaciones gestionan la comunicación entre servicios."
},
```

```
{
"Título": "Potencia n8n con RabbitMQ: Recibe y envía mensajes de forma eficiente",
"Introducción": "Aquí tienes un borrador de introducción (3-4 frases):\n\n\"n8n es una herramienta de automatización increíblemente flexible, pero cuando se enfrenta a flujos de trabajo de alta concurrencia o necesita una comunicación asíncrona robusta, es esencial integrarla con un sistema de colas de mensajes. RabbitMQ es el bróker de mensajes estándar de facto que ofrece durabilidad y escalabilidad para manejar cualquier volumen de datos. En esta guía, exploraremos cómo podemos potenciar significativamente nuestras automatizaciones de n8n, configurándolo para que tanto reciba mensajes entrantes de RabbitMQ como publique mensajes salientes de manera eficiente.\""
}
]
```