**Universidad de Almería**

Máster en Ingeniería Informática

# Integración de Tecnologías y Servicios Informáticos

---

# Práctica 4

Robustez y Modularidad - Manejo de Errores y Sub-Flujos de Trabajo

---

**Autor:** Johan Eduardo Cala Torra

**Fecha:** 8 de diciembre de 2025

Curso 2024-2025

# Índice

# 1. Introducción

Esta práctica representa un avance significativo hacia la construcción de automatizaciones de nivel profesional con n8n. Se introducen dos conceptos fundamentales: el **manejo de errores** para construir flujos robustos y la **modularidad** mediante sub-flujos de trabajo reutilizables.

## 1.1. Objetivos de Aprendizaje

- **Diseño de Flujos Resilientes:** Crear sistemas que detectan, registran y notifican fallos de ejecución.
- **Manejo de Errores Centralizado:** Implementar un flujo dedicado usando el `Error Trigger`.
- **Control de Errores Proactivo:** Utilizar `Stop and Error` para validaciones de negocio.
- **Principios de Modularidad (DRY):** Aplicar "Don't Repeat Yourself" en automatizaciones.
- **Sub-Flujos de Trabajo:** Construir lógica reutilizable con `Execute Workflow`.

# 2. Fundamentos Teóricos

## 2.1. Manejo de Errores: Construyendo Flujos Robustos

En un entorno de producción, cuando un nodo falla (API no disponible, datos incorrectos), la ejecución no debe simplemente detenerse. n8n proporciona un sistema elegante basado en flujos de trabajo especiales.

### 2.1.1. Error Trigger

Es un disparador especial que se activa cuando **otro flujo de trabajo falla**. El flujo que comienza con este nodo se denomina "Error Workflow".

**Datos proporcionados por el Error Trigger:**

- `$json.execution.error.message`: Mensaje de error específico
- `$json.execution.lastNodeExecuted`: Nombre del nodo que falló
- `$json.workflow.name`: Nombre del flujo que falló
- `$json.execution.url`: Enlace directo a la ejecución fallida

### 2.1.2. Stop and Error

Permite detener deliberadamente un flujo cuando los datos no cumplen condiciones de negocio. Lanza un error personalizado que activa el Error Workflow configurado.

## 2.2. Modularidad: Sub-Flujos de Trabajo

Los sub-flujos permiten encapsular lógica reutilizable, evitando duplicación de nodos.

### 2.2.1. Execute Workflow Trigger

Actúa como punto de entrada para un sub-flujo. Espera ser llamado por otro workflow y puede recibir datos del flujo padre.

### 2.2.2. Execute Workflow

Se coloca en el flujo principal para llamar a un sub-flujo. El flujo padre se pausa, envía datos al sub-flujo, espera su finalización y recibe los resultados.

# 3. Flujo de Trabajo Guiado: Sistema de Procesamiento de Imágenes

Este sistema consta de **tres flujos de trabajo interconectados**:

1. **Manejador de Errores**: Registra fallos en Google Sheets
2. **Sub-flujo Descargador de Imágenes**: Descarga imágenes de URLs
3. **Procesador Principal de Imágenes**: Orquesta todo el proceso

## 3.1. Paso 1: Flujo de Trabajo de Errores

Este flujo actúa como "centro de monitoreo" del sistema.

### 3.1.1. Configuración de Google Sheets

Se preparó una hoja con las siguientes cabeceras:

- Timestamp
- Workflow Fallido
- Nodo Fallido
- Mensaje de Error
- URL de Ejecución

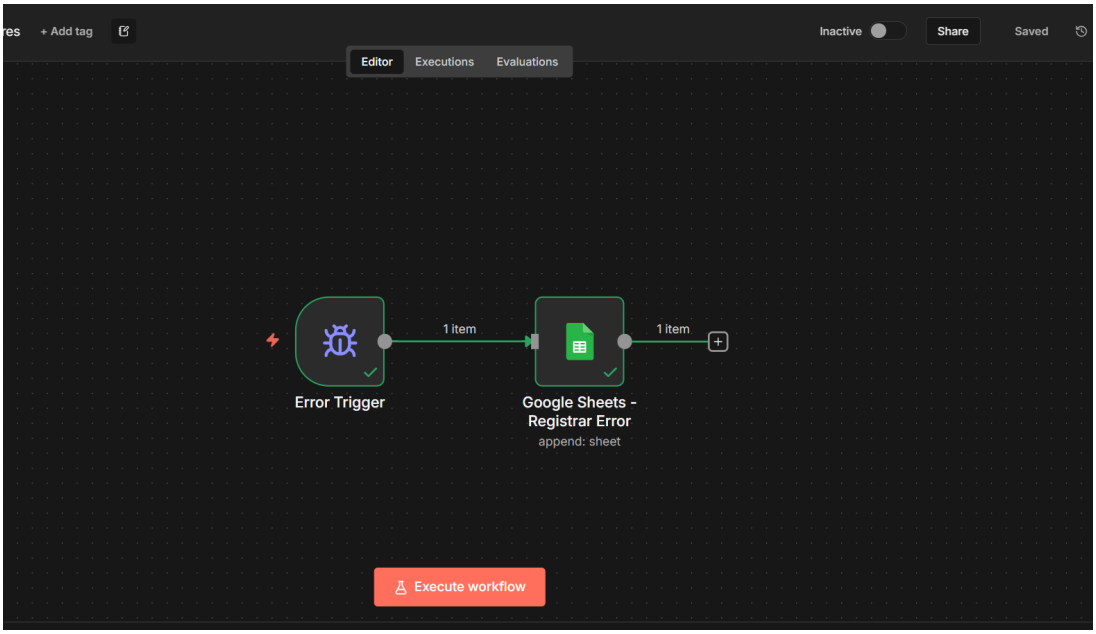


Figura 1: Vista completa del Manejador de Errores

### 3.1.2. Configuración del Error Trigger

El nodo Error Trigger no requiere configuración adicional. Se activa automáticamente cuando un flujo que lo tiene configurado como "Error Workflow" falla.

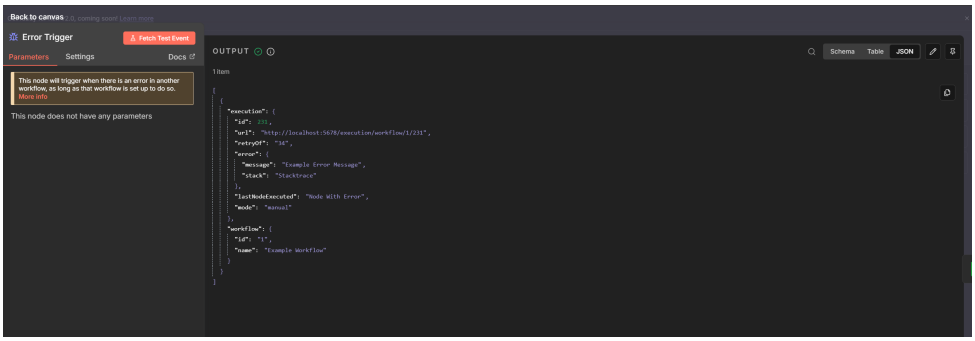


Figura 2: Configuración del Error Trigger

### 3.1.3. Mapeo de Campos en Google Sheets

Se configuró el nodo Google Sheets con las siguientes expresiones:

```
Timestamp:          {{ $now }}
Workflow Fallido:   {{ $json.workflow.name }}
Nodo Fallido:       {{ $json.execution.lastNodeExecuted }}
Mensaje de Error:   {{ $json.execution.error.message }}
URL de Ejecucion:   {{ $json.execution.url }}
```

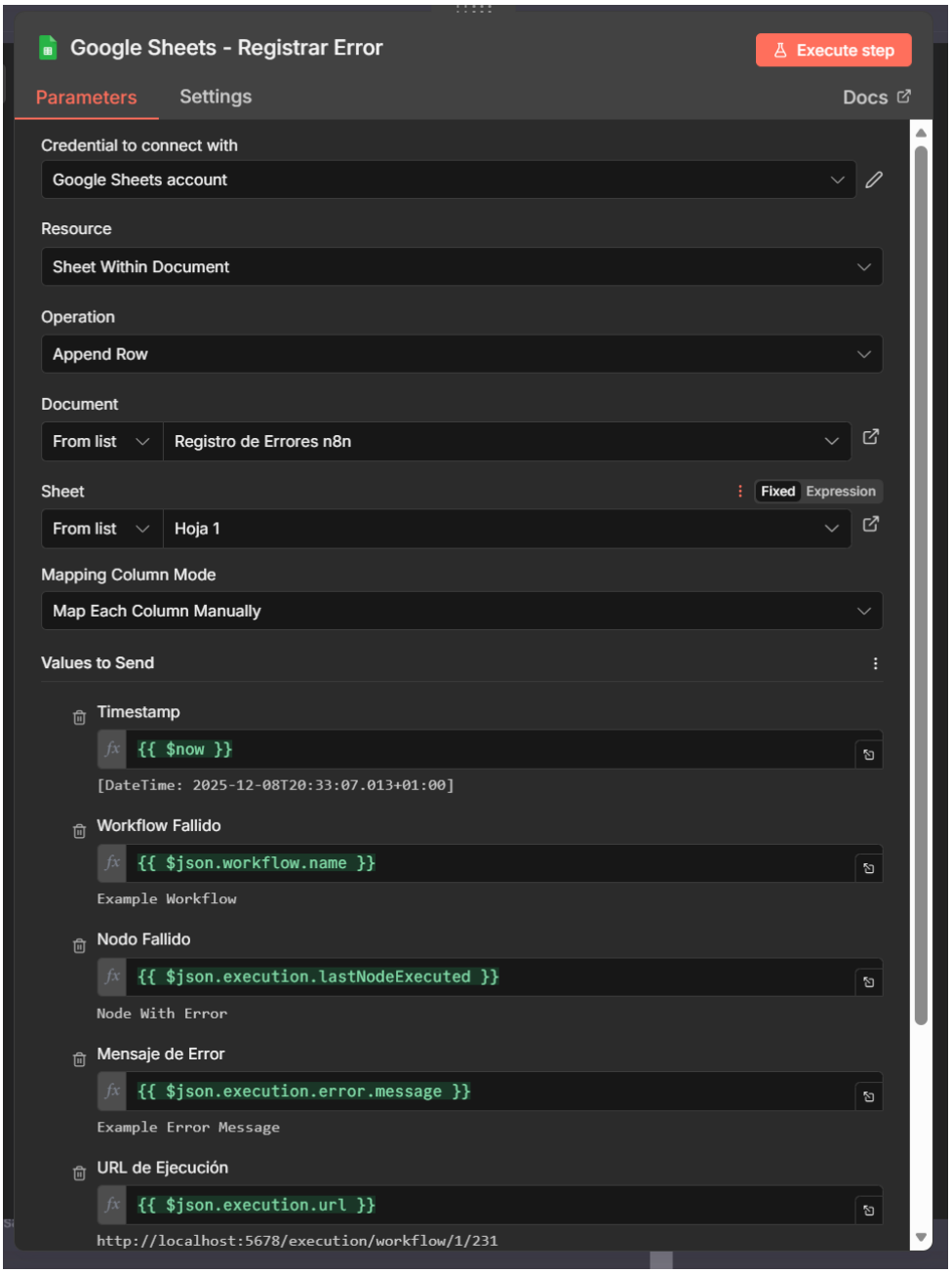


Figura 3: Configuración del nodo Google Sheets para registro de errores

## 3.2. Paso 2: Sub-Flujo Descargador de Imágenes

Este sub-flujo encapsula la lógica de descarga de imágenes para ser reutilizada.

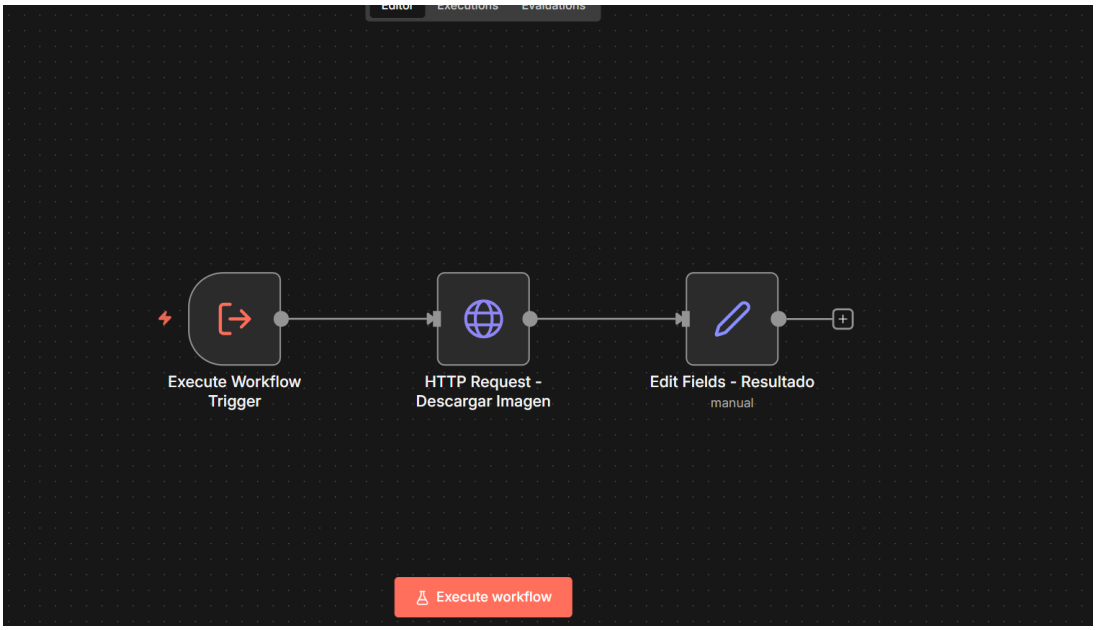


Figura 4: Vista completa del Sub-flujo Descargador de Imágenes

### 3.2.1. Execute Workflow Trigger

Este nodo recibe la URL de la imagen desde el flujo principal:

```
// Datos recibidos del flujo padre:
{ "imageUrl": "https://ejemplo.com/imagen.jpg" }
```

### 3.2.2. HTTP Request - Descarga de Imagen

Configuración del nodo para descargar archivos binarios:

- **URL:** {{ `$json.imageUrl` }}
- **Response Format:** File (para datos binarios)

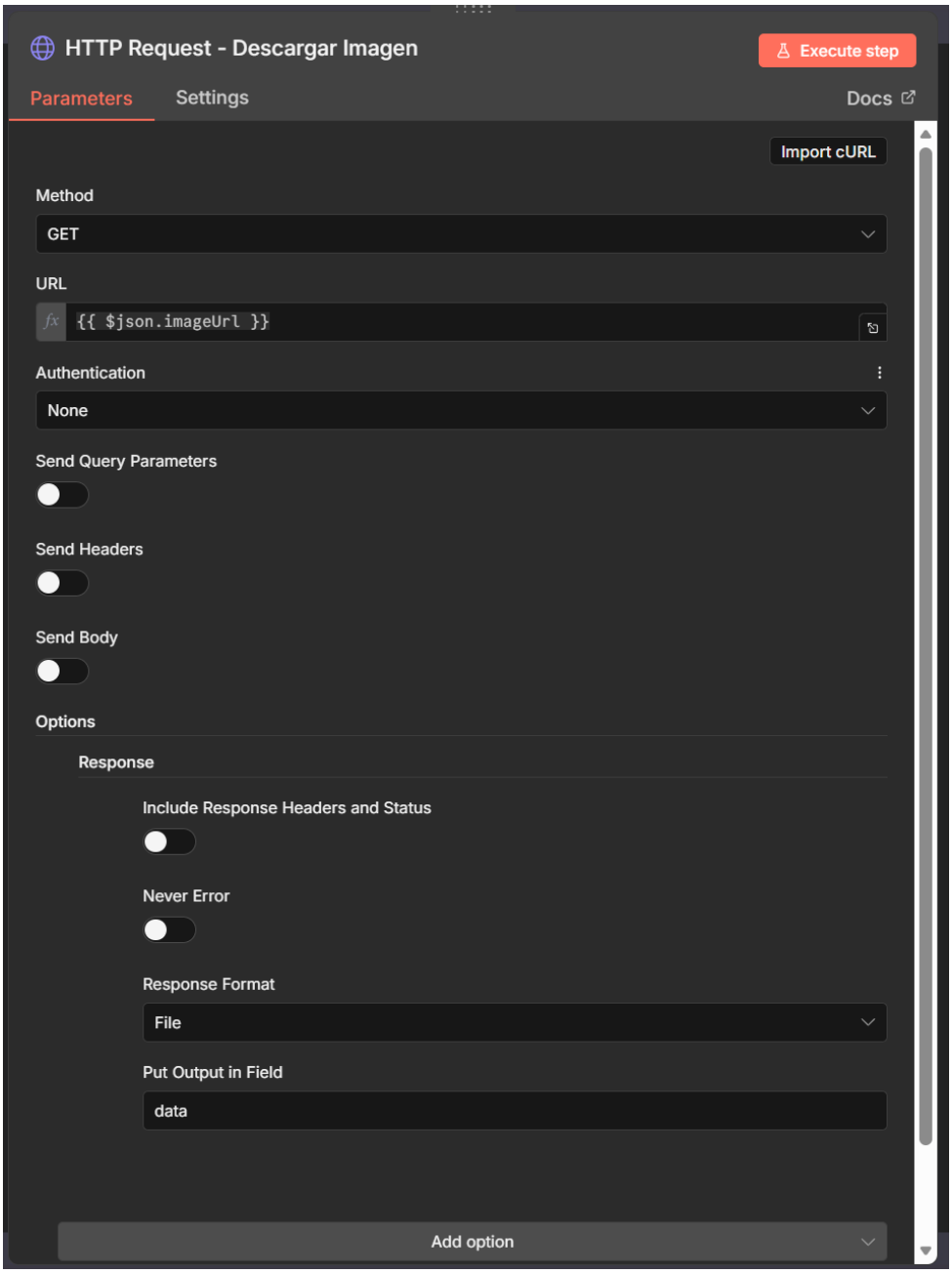


Figura 5: Configuración del HTTP Request para descarga de imágenes

**Nota importante:** Este workflow debe estar **activado** para poder ser llamado.

## 3.3. Paso 3: Flujo Principal - Procesador de Imágenes

Este flujo orquesta todo el proceso y gestiona los errores.

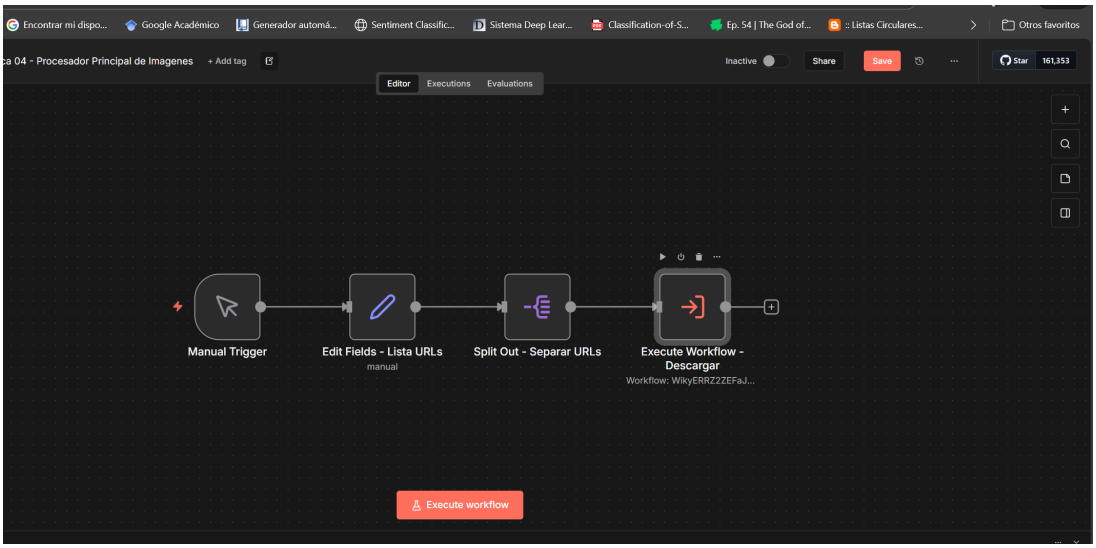


Figura 6: Vista completa del Procesador Principal de Imágenes

### 3.3.1. Configuración del Error Workflow

En Settings del workflow, se selecciona el "Manejador de Errores" como Error Workflow.

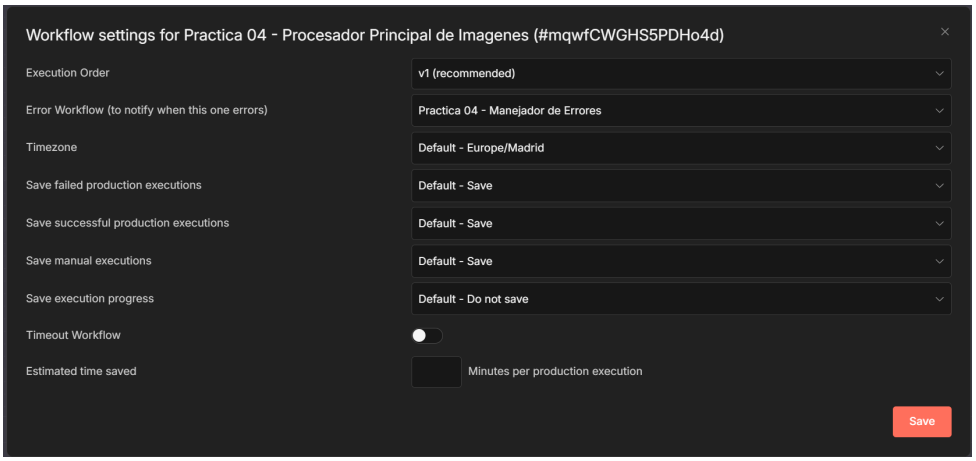


Figura 7: Configuración del Error Workflow en Settings

### 3.3.2. Simulación de Datos de Entrada

Se crea una lista de URLs incluyendo una inválida para forzar un error:

```
[
  {"imageUrl": "https://picsum.photos/200"},
  {"imageUrl": "https://picsum.photos/300"},
  {"imageUrl": "https://esta-url-no-existe-error.xyz/imagen.jpg"}
]
```

### 3.3.3. Split Out y Execute Workflow

- **Split Out:** Separa el array en items individuales
- **Execute Workflow:** Llama al sub-flujo por cada URL

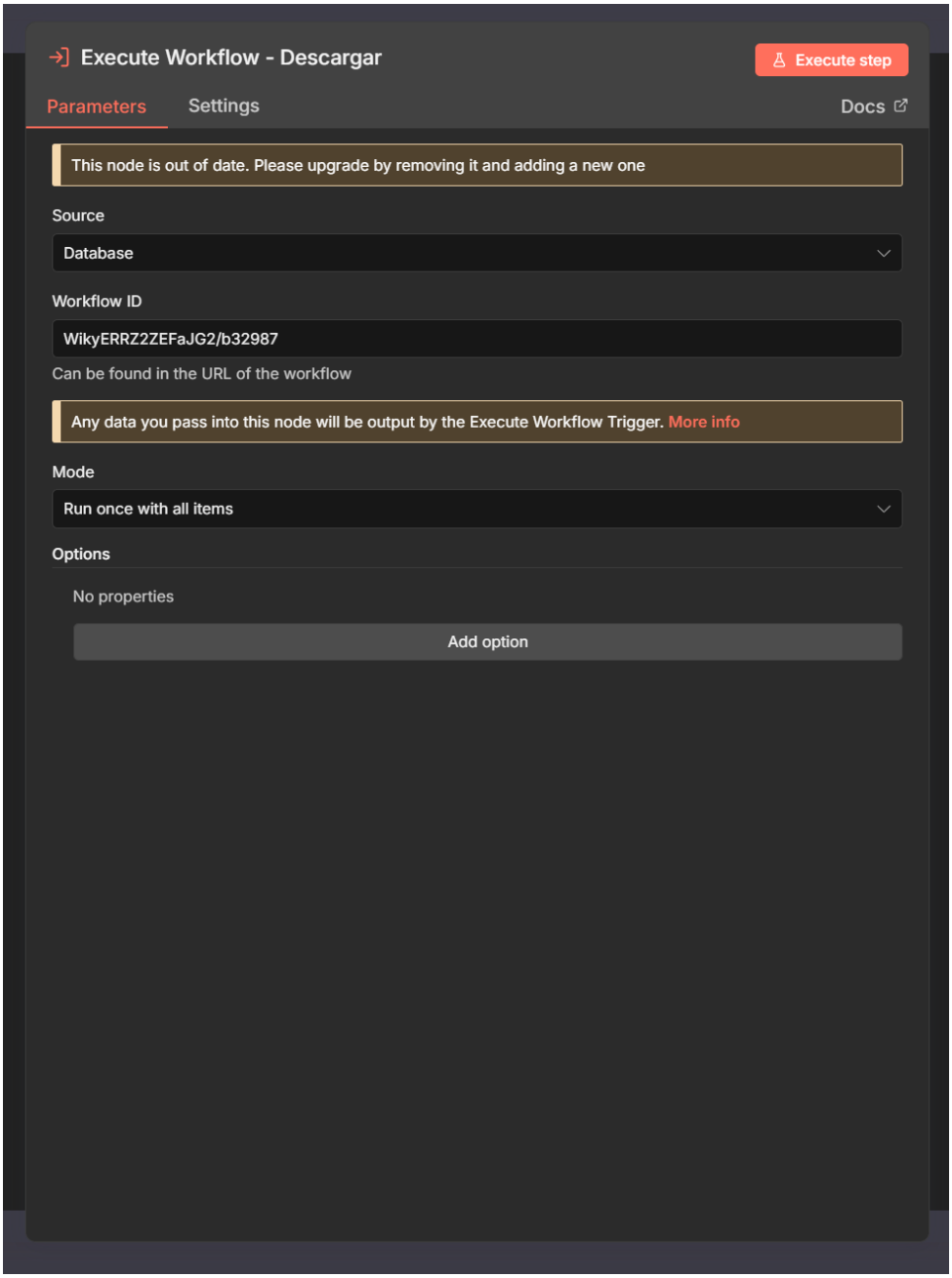


Figura 8: Configuración del nodo Execute Workflow

### 3.3.4. Resultado de la Ejecución

Al ejecutar, el flujo falla en la URL inválida y el error queda registrado en Google Sheets.

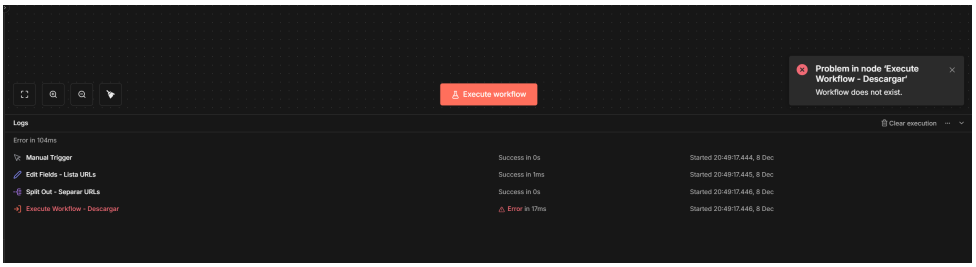


Figura 9: Ejecución fallida mostrando el nodo en rojo

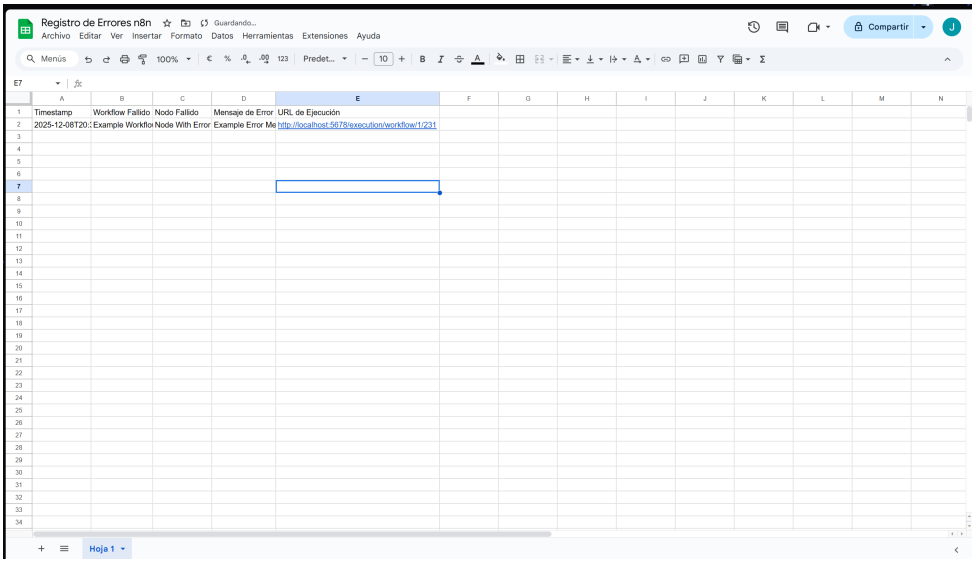


Figura 10: Error registrado automáticamente en Google Sheets

# 4. Ejercicio 1: Notificación de Errores Mejorada

**Objetivo:** Mejorar el "Manejador de Errores" para que, además de registrar en Google Sheets, envíe una notificación por correo electrónico.

**Dificultad:** Baja

## 4.1. Diseño de la Solución

Se modifica el workflow "Manejador de Errores" añadiendo un nodo Gmail en paralelo con el nodo de Google Sheets. Ambos nodos reciben los datos del Error Trigger simultáneamente.

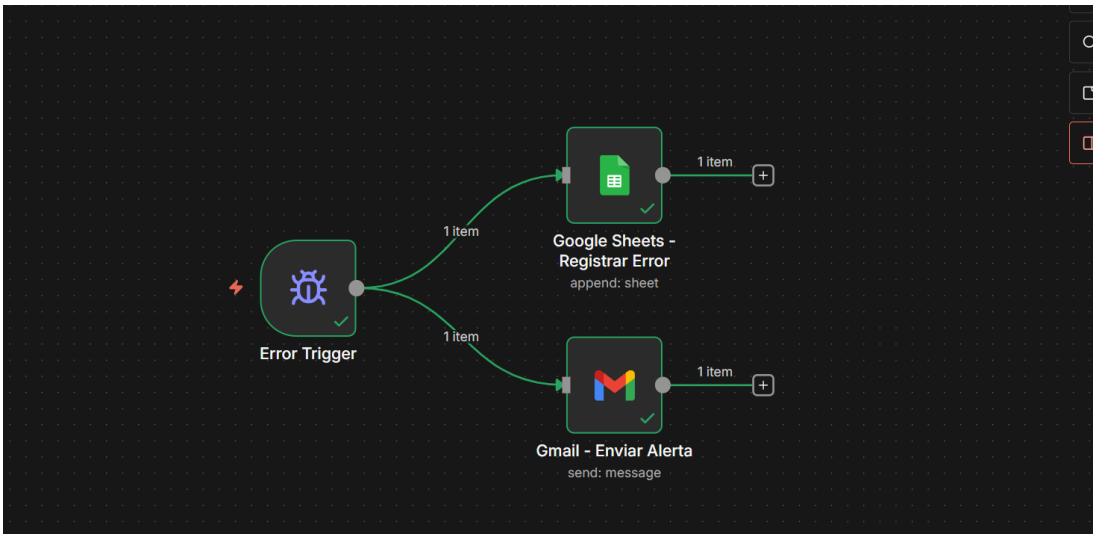


Figura 11: Workflow del Manejador de Errores Mejorado con notificación por email

## 4.2. Configuración del Nodo Gmail

Se configuró el nodo Gmail con los siguientes parámetros:

- **Send To:** Dirección de correo del administrador
- **Subject:** `¡Alerta de Error en n8n! - {{ $json.workflow.name }}`

**Cuerpo del mensaje:**

```
Se ha producido un error en n8n:

Workflow Fallido: {{ $json.workflow.name }}
Nodo que Fallo: {{ $json.execution.lastNodeExecuted }}
Mensaje de Error: {{ $json.execution.error.message }}
Fecha/Hora: {{ $now }}

Ver ejecucion: {{ $json.execution.url }}
```

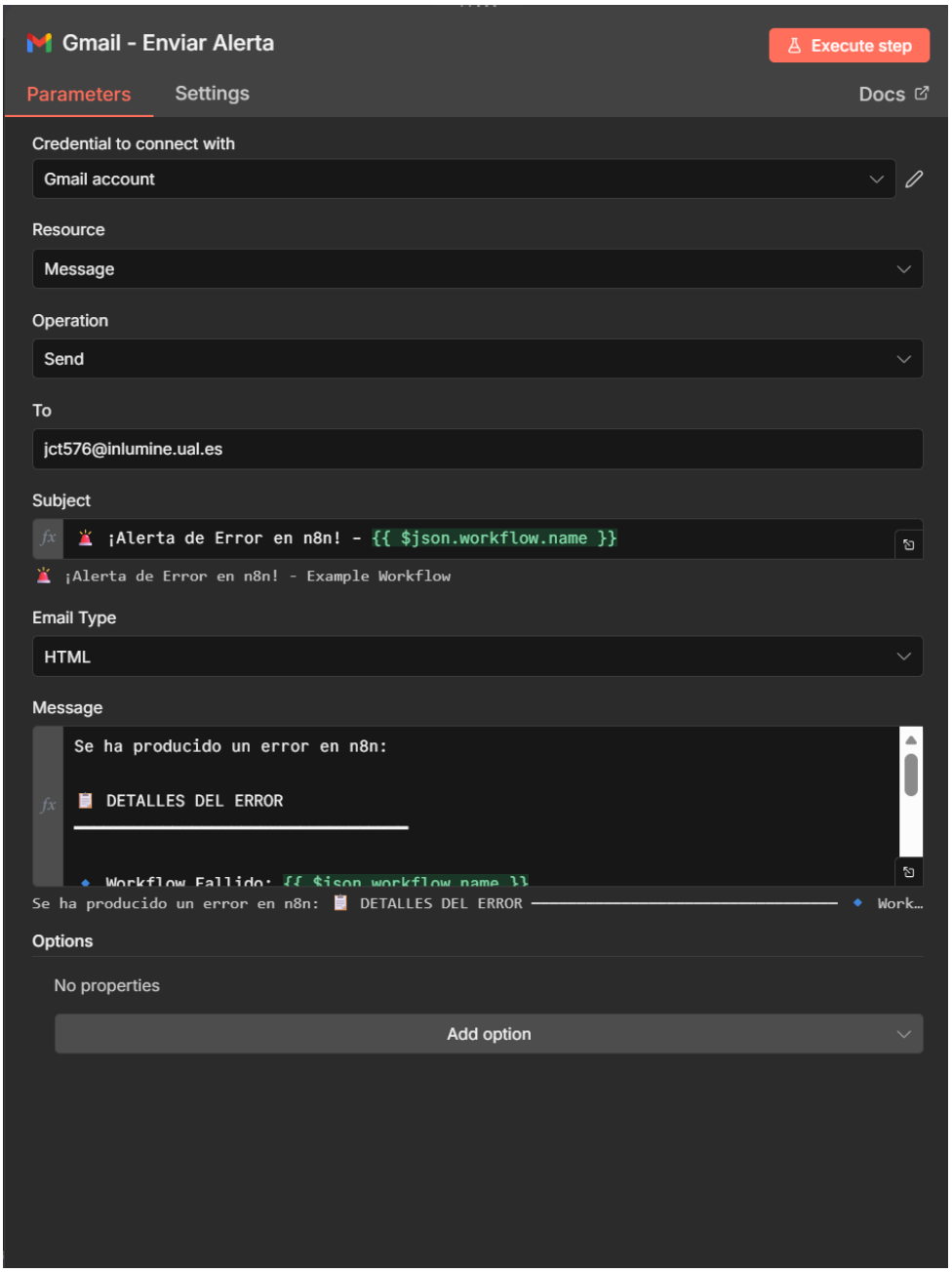


Figura 12: Configuración del nodo Gmail para alertas de error

## 4.3. Resultado

Al producirse un error en cualquier workflow configurado, se recibe automáticamente un correo electrónico con los detalles del fallo.

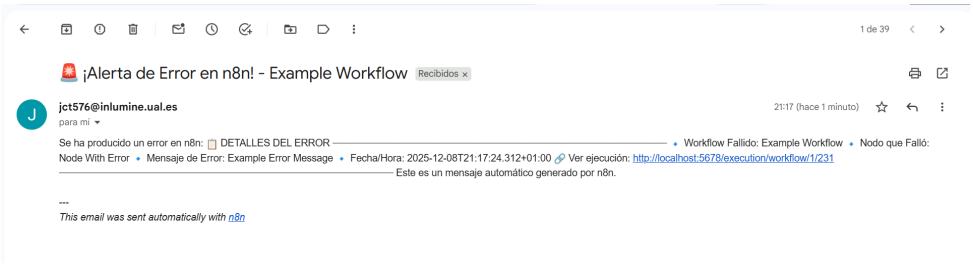


Figura 13: Correo de alerta recibido en la bandeja de entrada

# 5. Ejercicio 2: Sub-Flujo de Validación de Datos

**Objetivo:** Crear un sub-flujo reutilizable que valide la estructura de datos de un producto y lance un error si faltan campos obligatorios.

**Dificultad:** Media

**API a utilizar:** Fake Store API (`https://fakestoreapi.com/products/1`)

## 5.1. Arquitectura del Sistema

Este ejercicio requiere dos workflows:

1. **Sub-flujo Validador de Productos:** Recibe un producto y verifica que tenga los campos obligatorios (`title` y `price`).
2. **Flujo Principal:** Obtiene un producto de la API, elimina el campo `price` para forzar el error, y llama al validador.

## 5.2. Sub-flujo: Validador de Productos

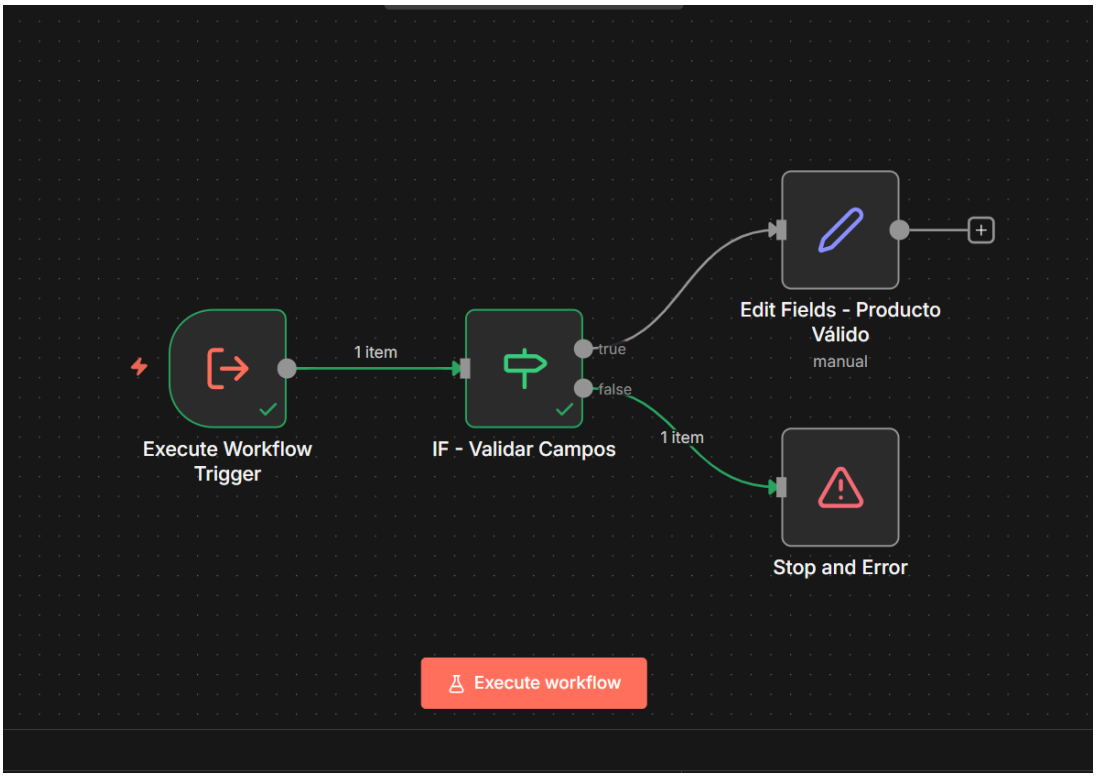


Figura 14: Sub-flujo Validador de Productos

### 5.2.1. Configuración del Nodo IF

El nodo IF verifica dos condiciones con operador AND:

- `$json.title` → is not empty (el título existe y no está vacío)
- `$json.price` → exists (el precio existe)

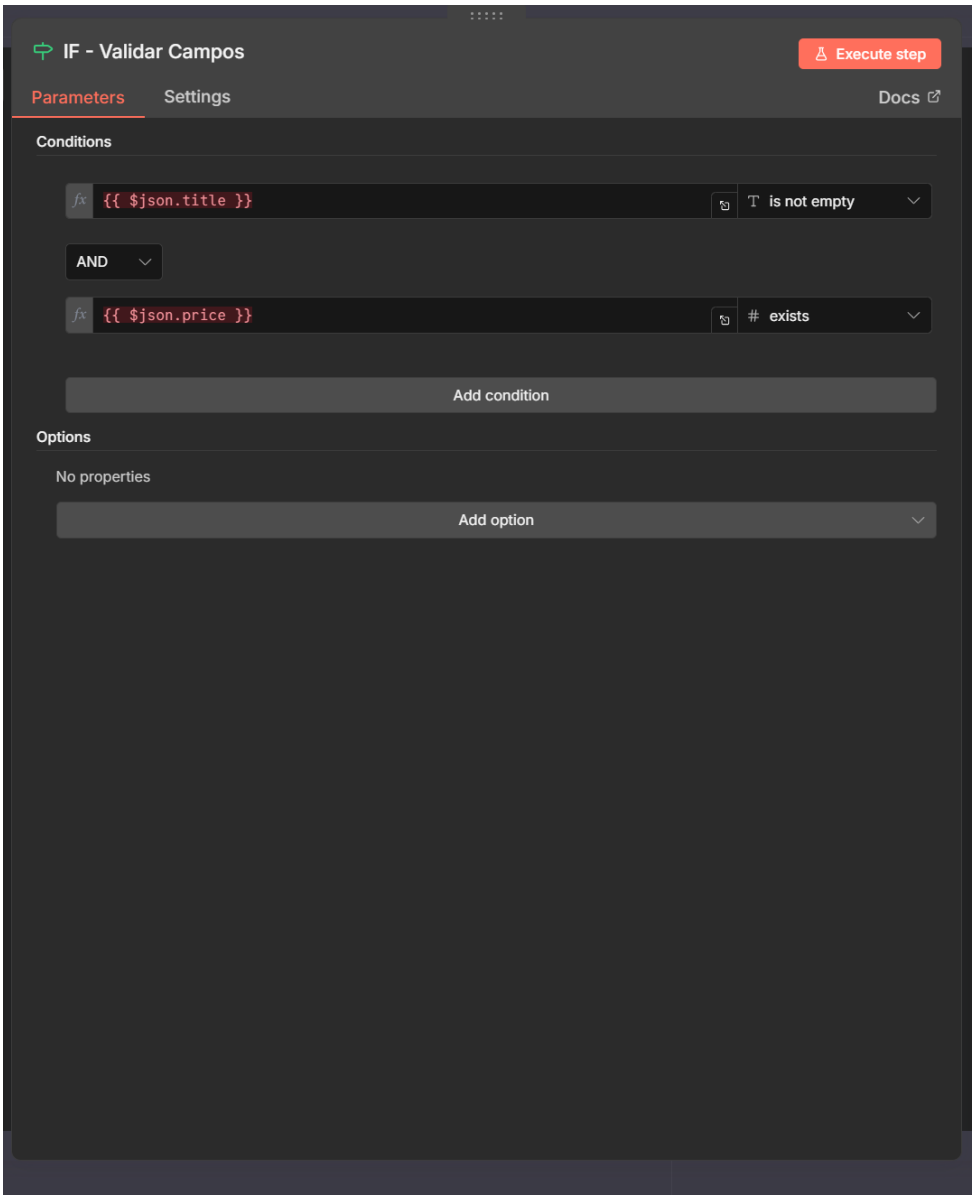


Figura 15: Configuración del nodo IF con condiciones AND

### 5.2.2. Configuración del Stop and Error

Si la validación falla (rama FALSE), se ejecuta el nodo Stop and Error con un mensaje personalizado:

```
Validacion fallida: Faltan campos de producto obligatorios (title
    o price)
```

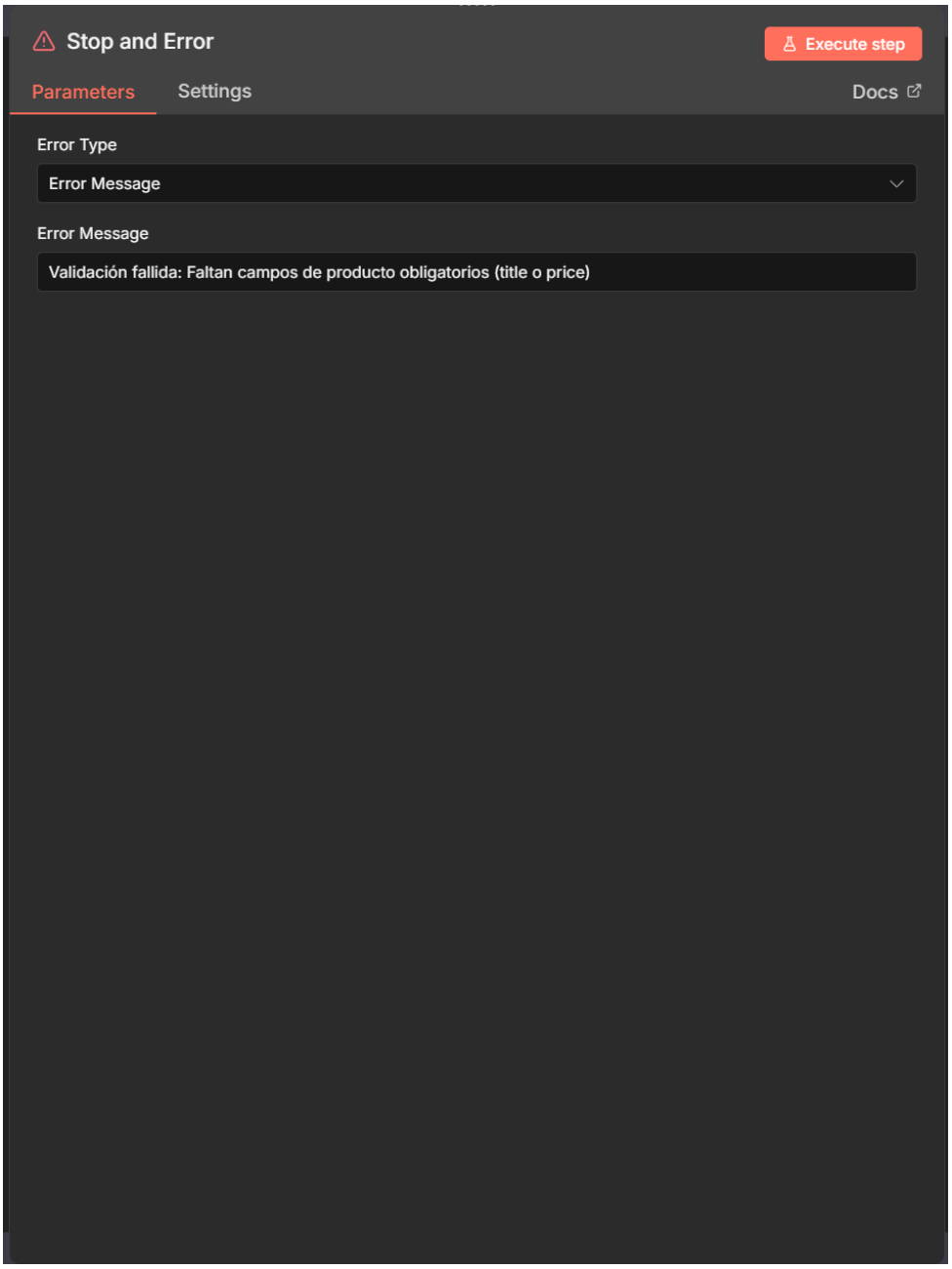


Figura 16: Configuración del nodo Stop and Error

## 5.3. Flujo Principal de Productos

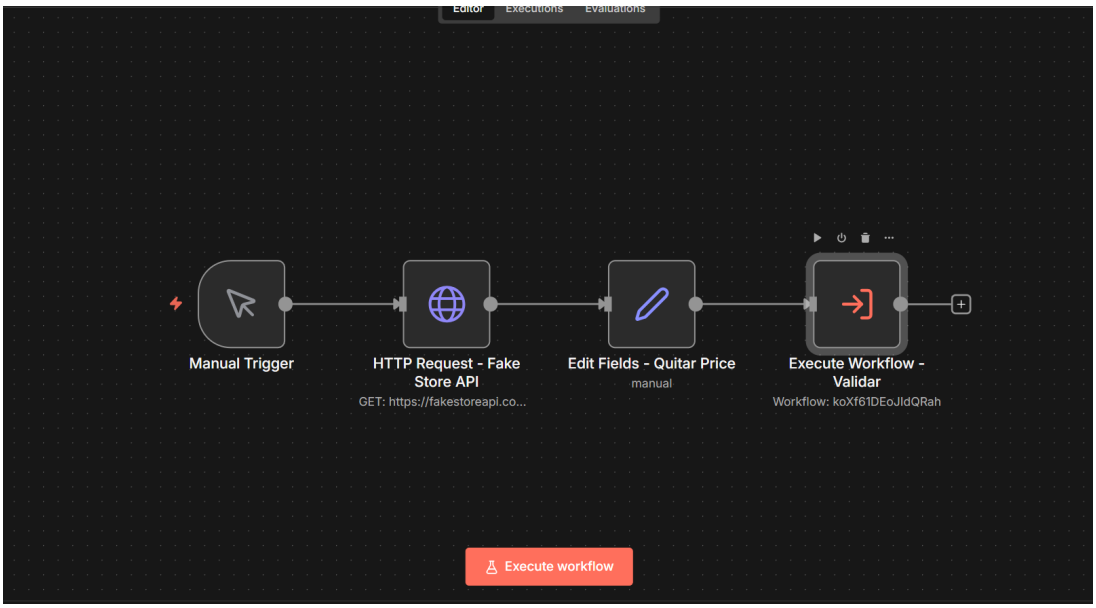


Figura 17: Flujo Principal que llama al validador

### 5.3.1. Simulación del Error

El nodo "Edit Fields" elimina intencionalmente el campo `price` para forzar que la validación falle:

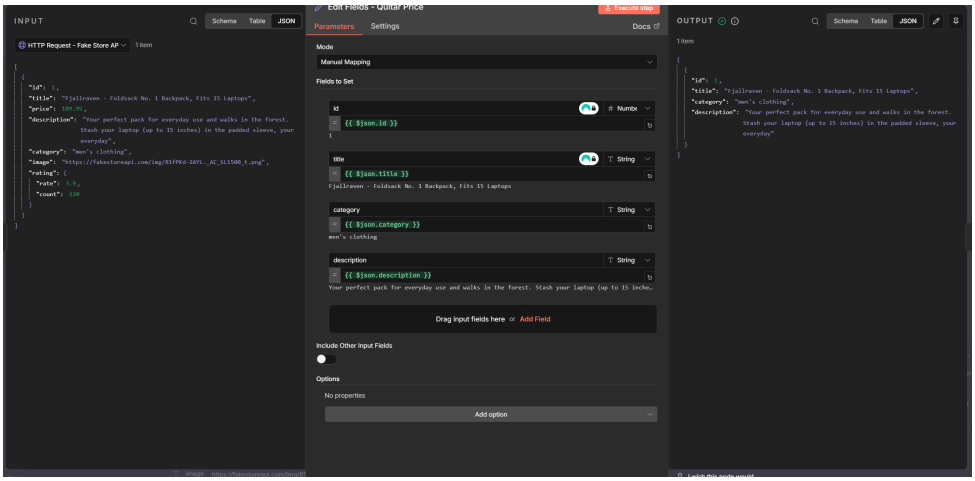


Figura 18: Edit Fields configurado para excluir el campo price

## 5.4. Resultado de la Ejecución

Al ejecutar el flujo principal, el sub-flujo validador detecta la ausencia del campo `price` y lanza el error personalizado, que es capturado por el Manejador de Errores Mejorado.

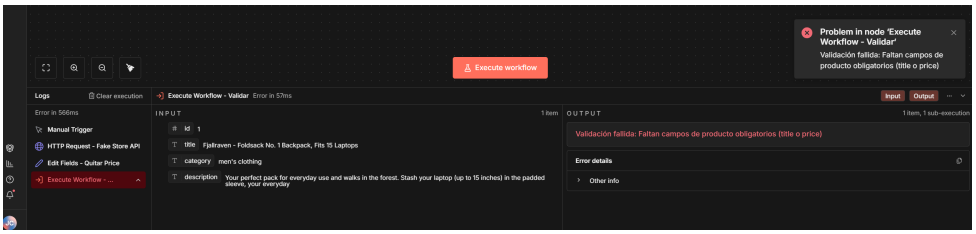


Figura 19: Ejecución fallida mostrando el error de validación

# 6. Ejercicio 3: Orquestador de Tareas Dinámicas

**Objetivo:** Crear un flujo principal que decide qué tarea realizar y llama al sub-flujo apropiado.

**Dificultad:** Alta

**APIs a utilizar:**

- Bored API: `https://bored-api.appbrewery.com/random`
- Public Holidays API: `https://date.nager.at/api/v3/PublicHolidays/2025/ES`

## 6.1. Arquitectura del Sistema

Este ejercicio implementa un patrón de orquestación donde un flujo principal decide dinámicamente qué sub-flujo ejecutar basándose en una variable de control. Se requieren tres workflows:

1. **Sub-flujo Obtener Actividad:** Consulta la Bored API para obtener una actividad aleatoria.
2. **Sub-flujo Obtener Festivos:** Consulta la API de festivos de España 2025.
3. **Orquestador Principal:** Utiliza un nodo Switch para decidir qué sub-flujo llamar.

## 6.2. Sub-flujo: Obtener Actividad

Este sub-flujo consulta la Bored API y formatea la respuesta con información sobre la actividad sugerida.

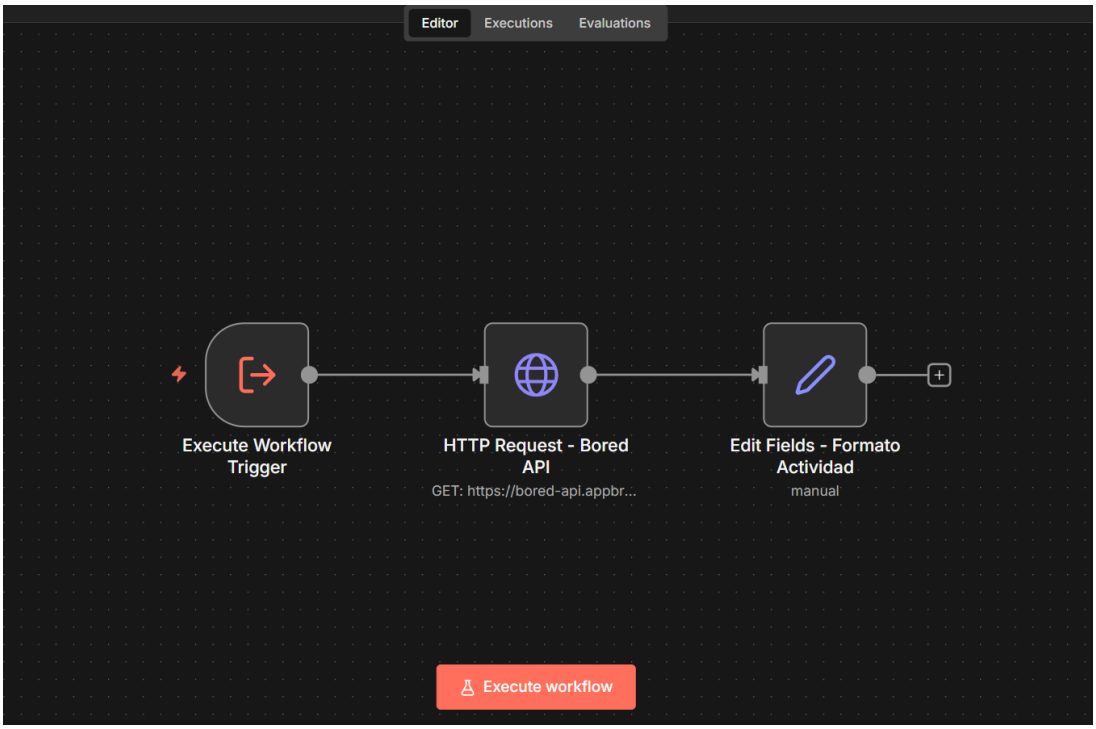


Figura 20: Sub-flujo para obtener actividad aleatoria de Bored API

**Configuración del HTTP Request:**

- **URL:** `https://bored-api.appbrewery.com/random`
- **Method:** GET

## 6.3. Sub-flujo: Obtener Festivos

Este sub-flujo consulta la API de festivos públicos y devuelve la lista de días festivos de España para 2025.

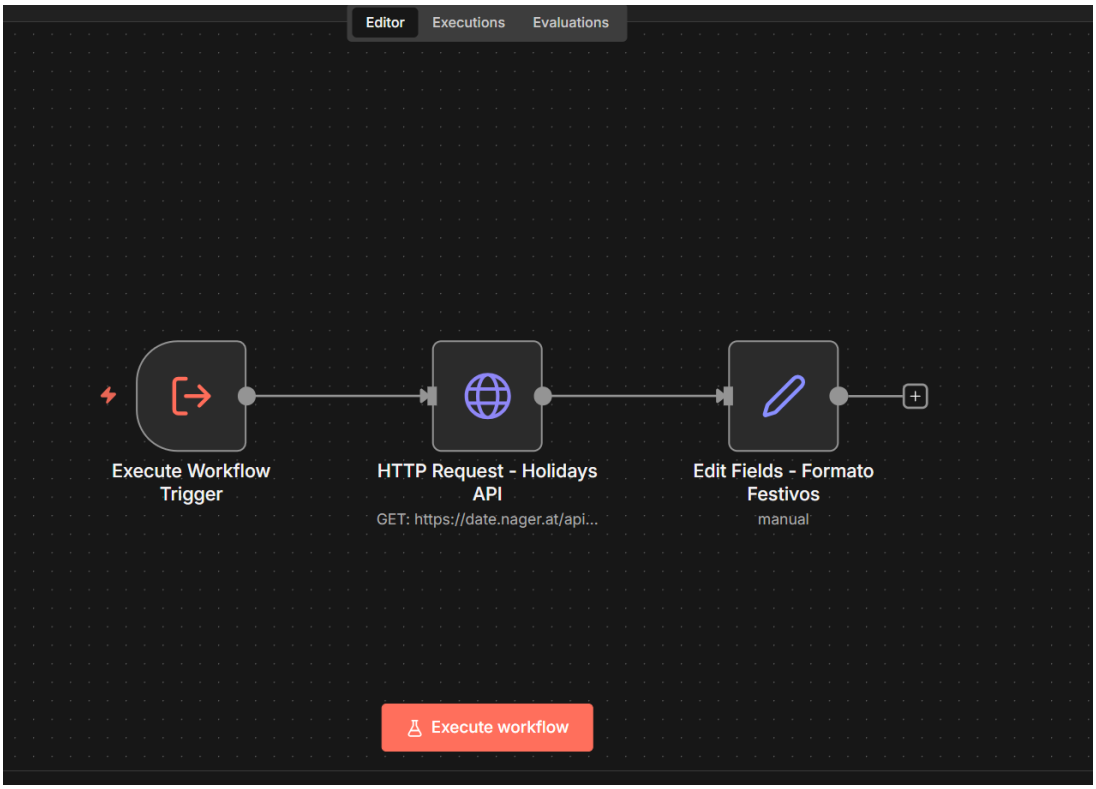


Figura 21: Sub-flujo para obtener festivos de España 2025

**Configuración del HTTP Request:**

- **URL:** `https://date.nager.at/api/v3/PublicHolidays/2025/ES`
- **Method:** GET

## 6.4. Orquestador Principal

El flujo principal utiliza un nodo Switch para evaluar la variable de control y dirigir la ejecución al sub-flujo correspondiente.


Figura 22: Orquestador Principal con Switch y llamadas a sub-flujos

### 6.4.1. Variable de Control

El nodo "Edit Fields - Variable de Control" define qué tarea ejecutar:

```
{
  "tarea": "actividad"  // o "festivos"
}
```

### 6.4.2. Configuración del Switch

El nodo Switch evalúa el campo `tarea` y tiene dos salidas:

- **Salida 1 (actividad):** Se activa cuando `$json.tarea === .ªctividad"`
- **Salida 2 (festivos):** Se activa cuando `$json.tarea === "festivos"`

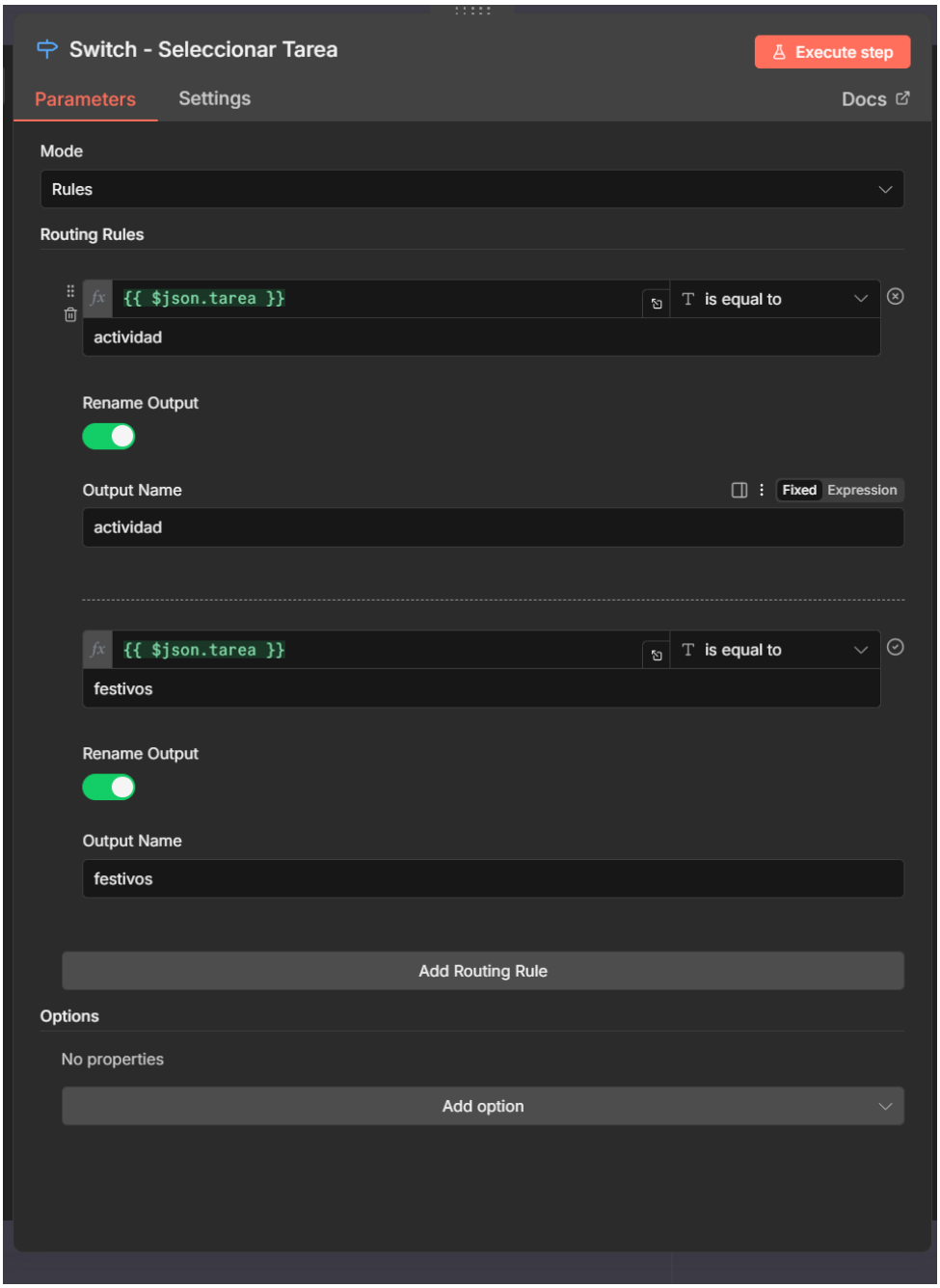


Figura 23: Configuración del nodo Switch con las dos condiciones

### 6.4.3. Nodo Merge

El nodo Merge en modo “Append” consolida las salidas de ambas ramas, pasando los datos de la rama que se haya ejecutado.

## 6.5. Resultados de Ejecución

### 6.5.1. Prueba con tarea = “actividad”

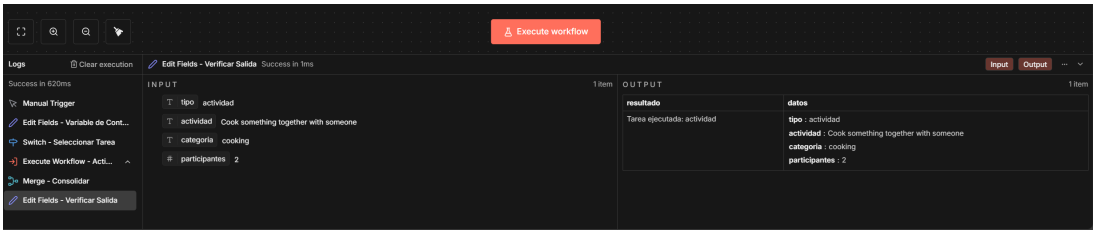


Figura 24: Ejecución del orquestador con tarea=.ªctividad”@

### 6.5.2. Prueba con tarea = “festivos”


Figura 25: Ejecución del orquestador con tarea=”festivos”@

### 6.5.3. Resultado Final

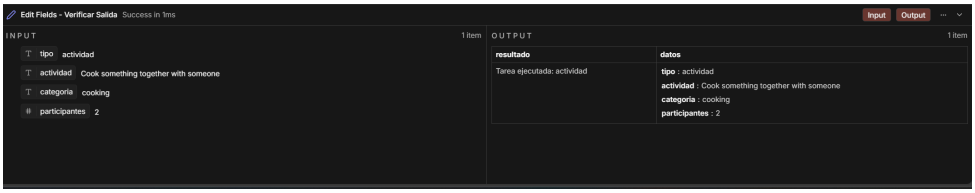


Figura 26: Nodo Edit Fields mostrando el resultado de la tarea ejecutada

# 7. Conclusiones

En esta práctica se han alcanzado los siguientes objetivos:

1. **Manejo de errores centralizado** mediante Error Trigger y Error Workflow
2. **Control proactivo de errores** con el nodo Stop and Error
3. **Modularidad con sub-flujos** usando Execute Workflow y Execute Workflow Trigger
4. **Principio DRY** aplicado a automatizaciones reutilizables
5. **Registro automático de fallos** en Google Sheets

## 7.1. Lecciones Aprendidas

- **Error Workflow vs Error en nodo:** El Error Workflow captura errores de todo el flujo, mientras que el manejo por nodo es más granular.
- **Sub-flujos deben estar activos:** Un workflow con Execute Workflow Trigger debe estar activado para poder ser llamado.
- **Paso de datos entre flujos:** El flujo padre pasa automáticamente el item actual al sub-flujo.
- **Stop and Error para validaciones:** Permite convertir condiciones de negocio en errores controlados.
- **Modularidad mejora mantenibilidad:** Los sub-flujos permiten actualizar lógica en un solo lugar.