Memoria Práctica 6

Jaime Parra Jiménez

INTEGRACIÓN DE TECNOLOGÍAS Y SERVICIOS INFORMÁTICOS

Universidad de Almería

Correo: jpj451@inlumine.ual.es

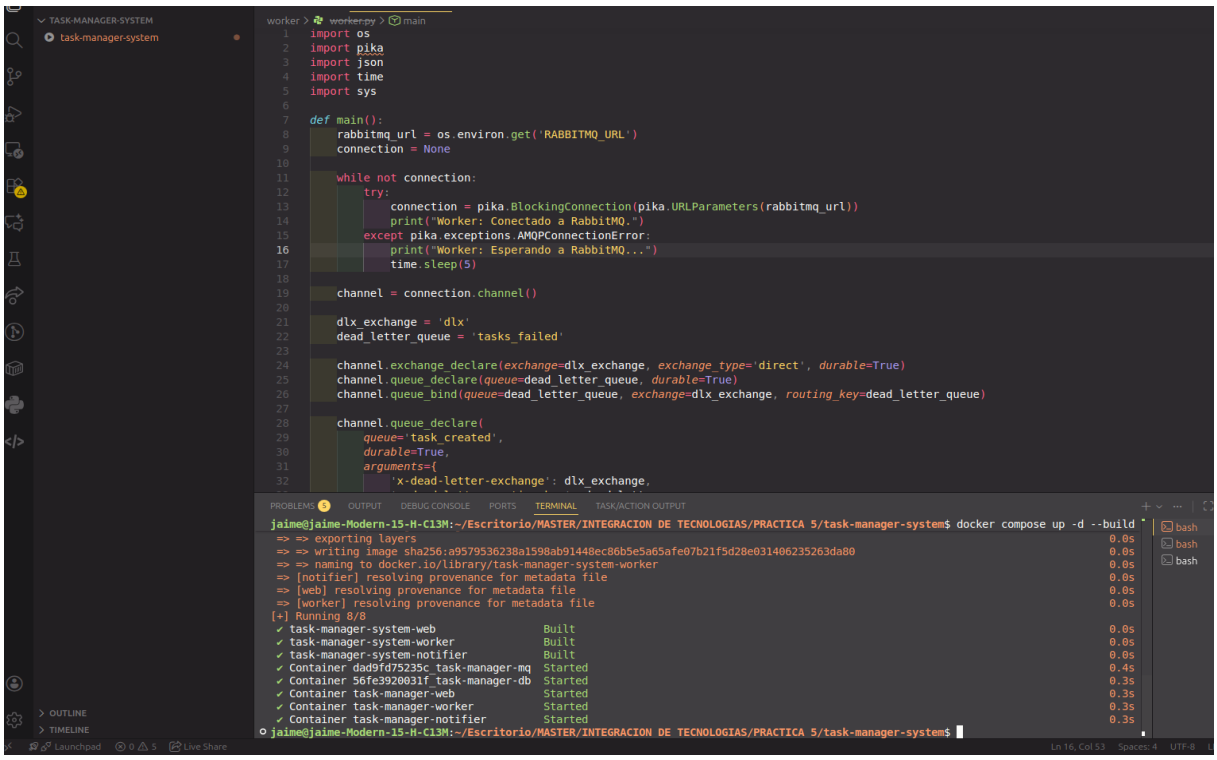
27 de noviembre de 2025

Índice general

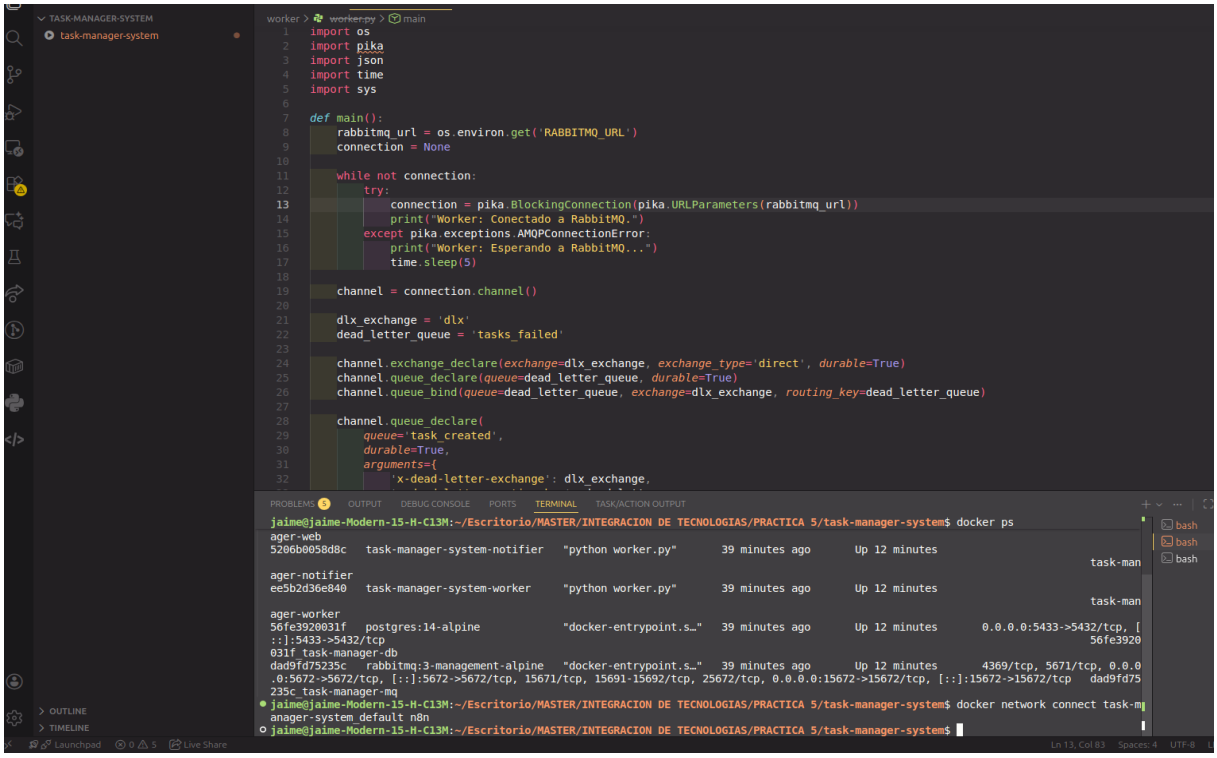
Capítulo 1

Ejercicio Guiado 1

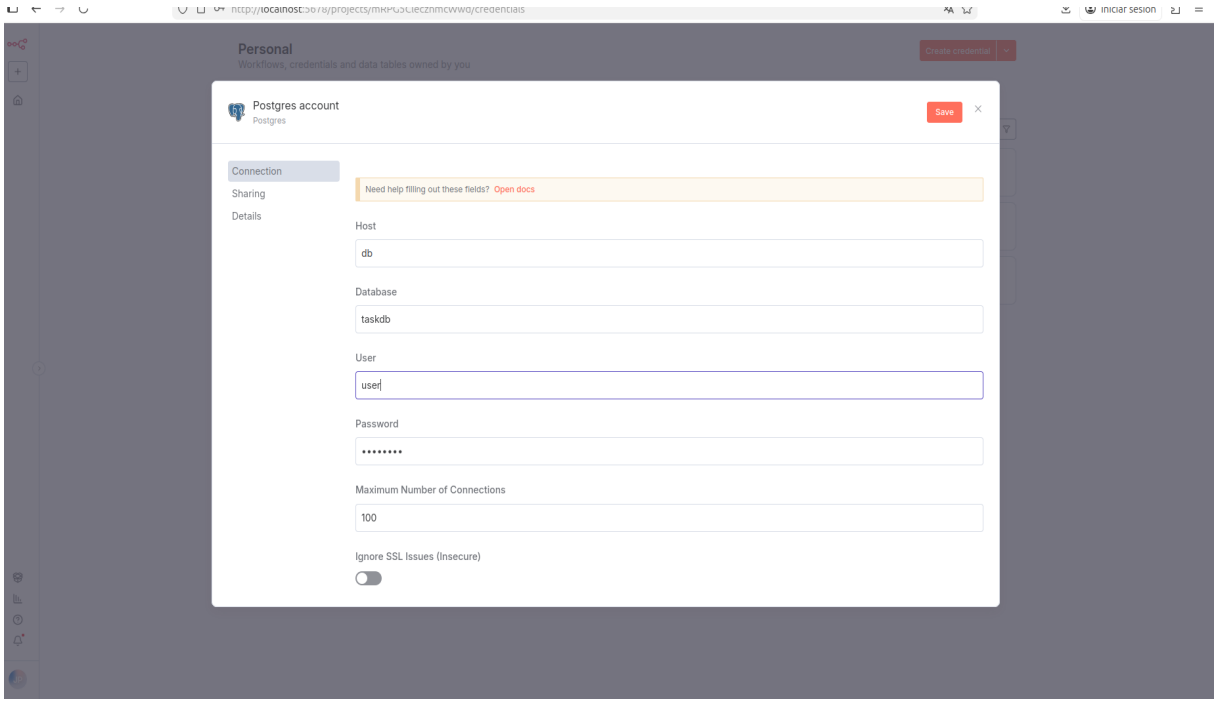
El objetivo del ejercicio guiado es conectar n8n el sistema de la práctica 5. n8n leerá datos de la base de datos PostgreSQL, y luego actuará como productor y consumidor de mensajes de RabbitMQ. El primer paso será preparar el entorno de microservicios. Para ello se debe iniciar la pila de servicios en segundo plano usando el comando docker-compose up -d –build.



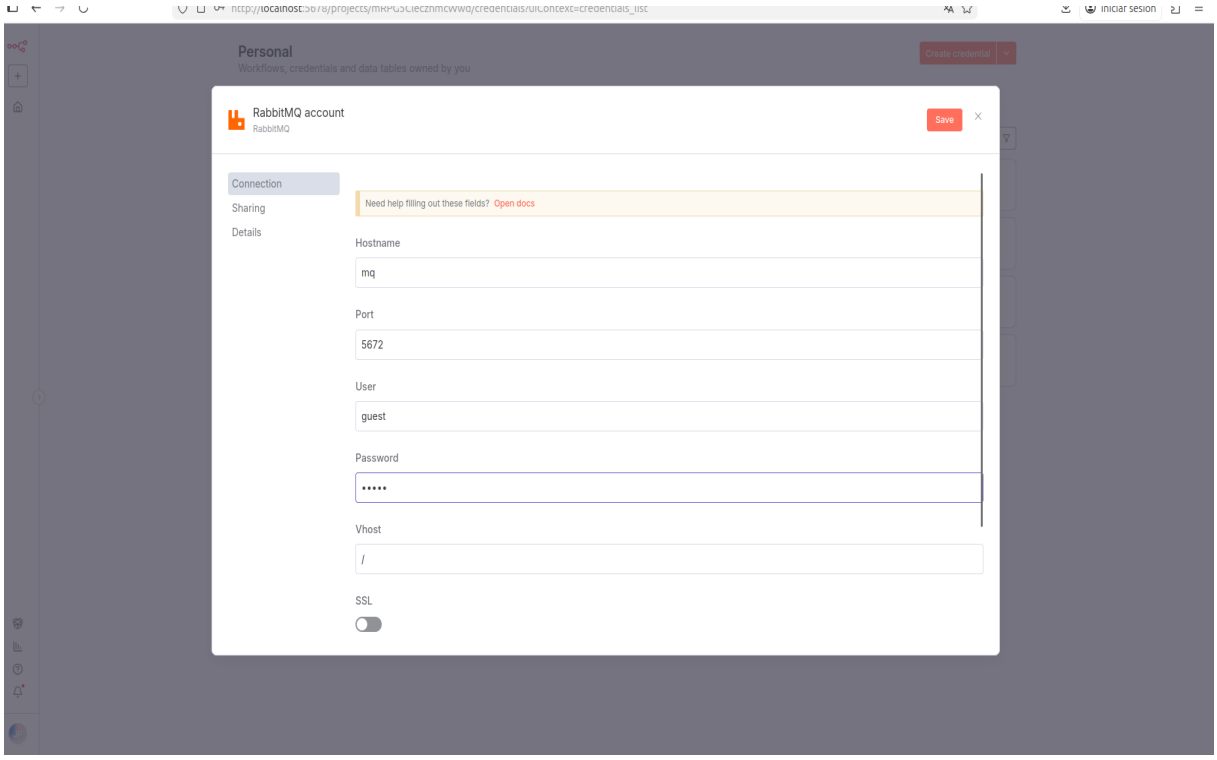
A continuación, es necesario conectar n8n a la red de microservicios. Para conectarlo se usará el siguiente comando.



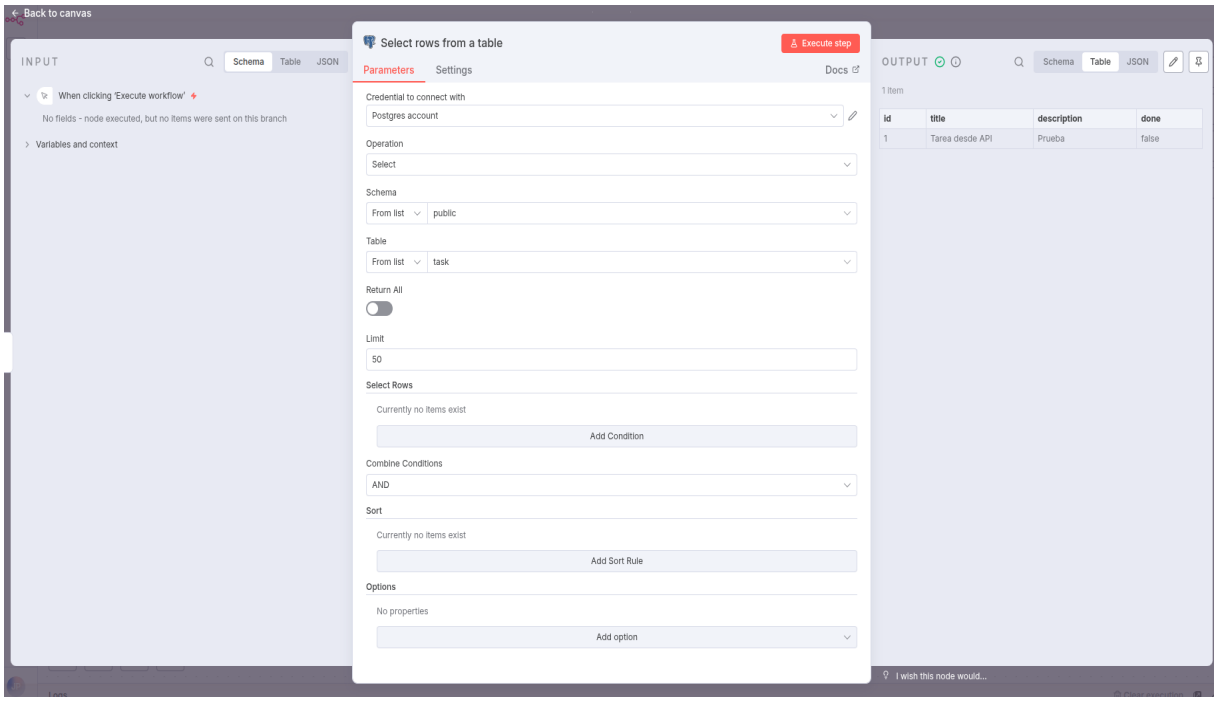
Una vez conectado n8n a la red es necesario configurar las credenciales en n8n. La primera credecial es PostgreSQL. Le asignaremos el host, database, user, password y port.



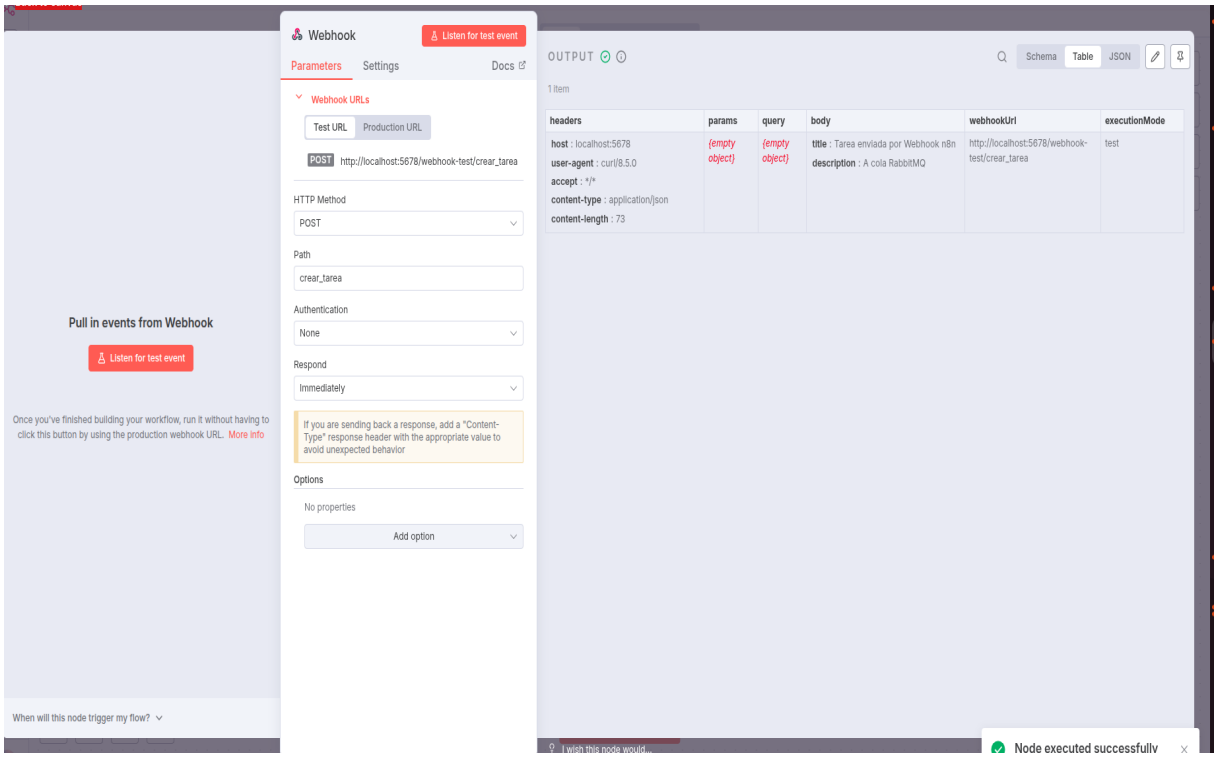
La segunda credencial es RabbitMQ donde se le asignará el host, user, password y port.



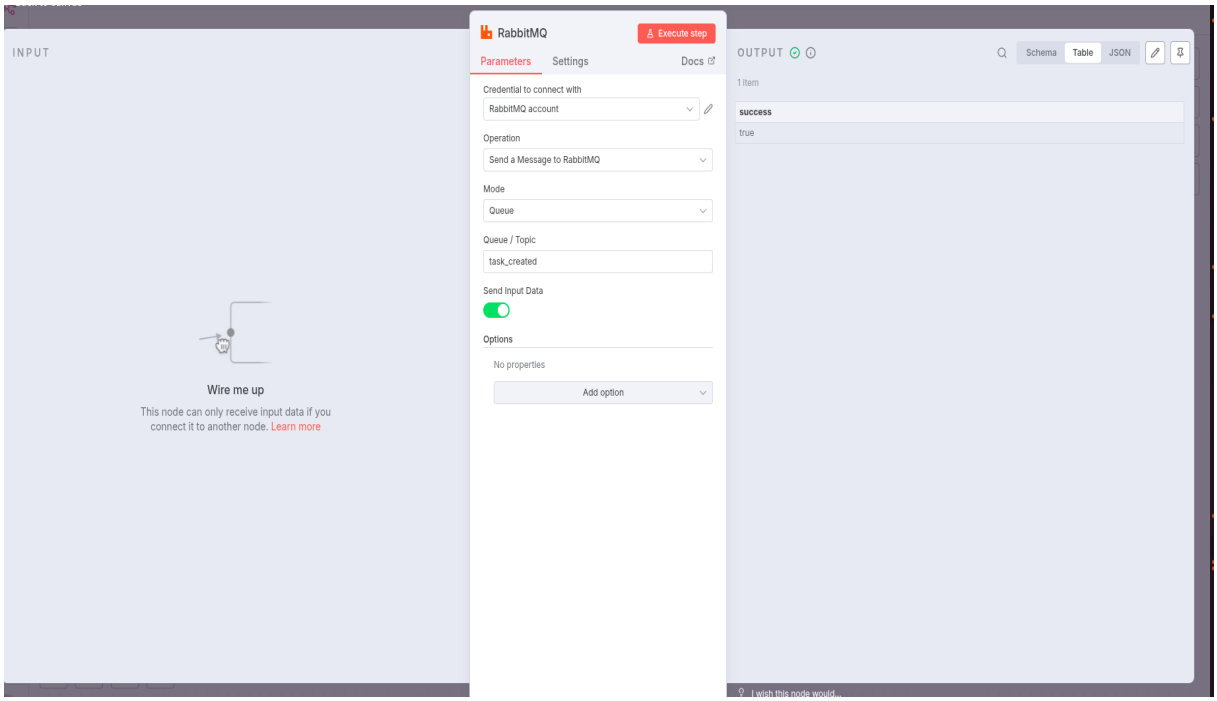
El sigueinte paso es crear un flujo que lee todas las tareas de la base de datos. Para ello, se añade un nodo Manual Trigger y a continuación un nodo PostgreSQL. Se debe selecionar la tabla task, la misma que en la práctica 5, y la operación de select.



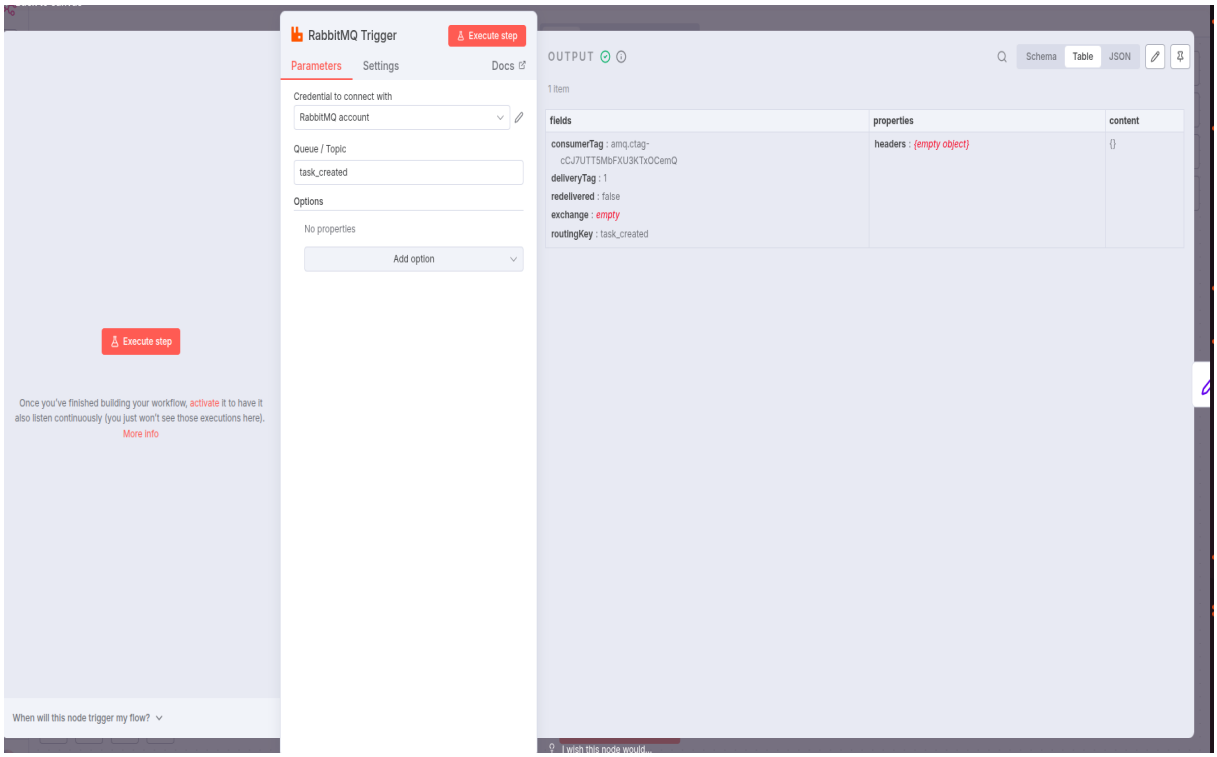
Además, también se crearán dos flujos para demostrar el patrón asíncrono. El primer flujo será productor, que recibirá una petición web y enviará la tarea a RabbitMQ. El nombre del flujo es Productor de Tareas y comenzará con un nodo Webhook Trigger.



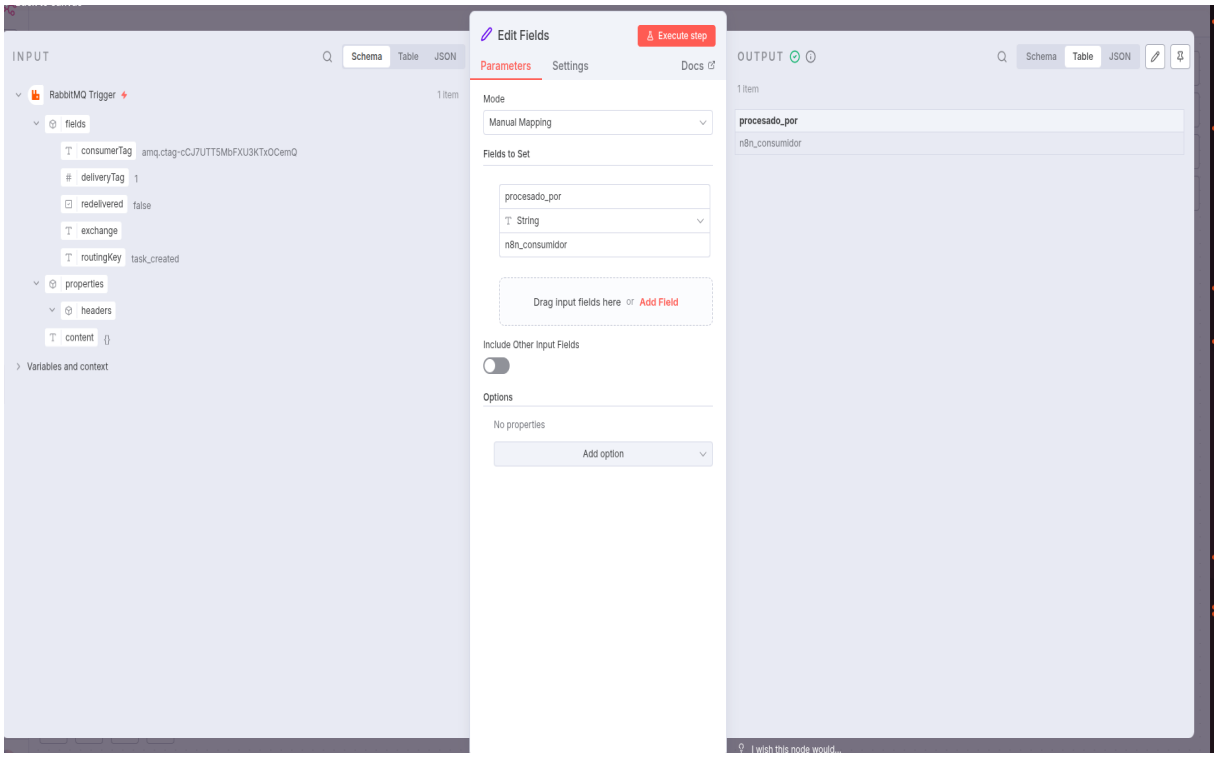
A continuación se añade un nodo RabbitMQ para que use la cola task-created y guardar los datos en esta cola.



Para crear el flujo consumidor, llamado Consumidor de Tareas, se pondrá un RabbitMQ Trigger como primer nodo para que escuche la cola task-created.



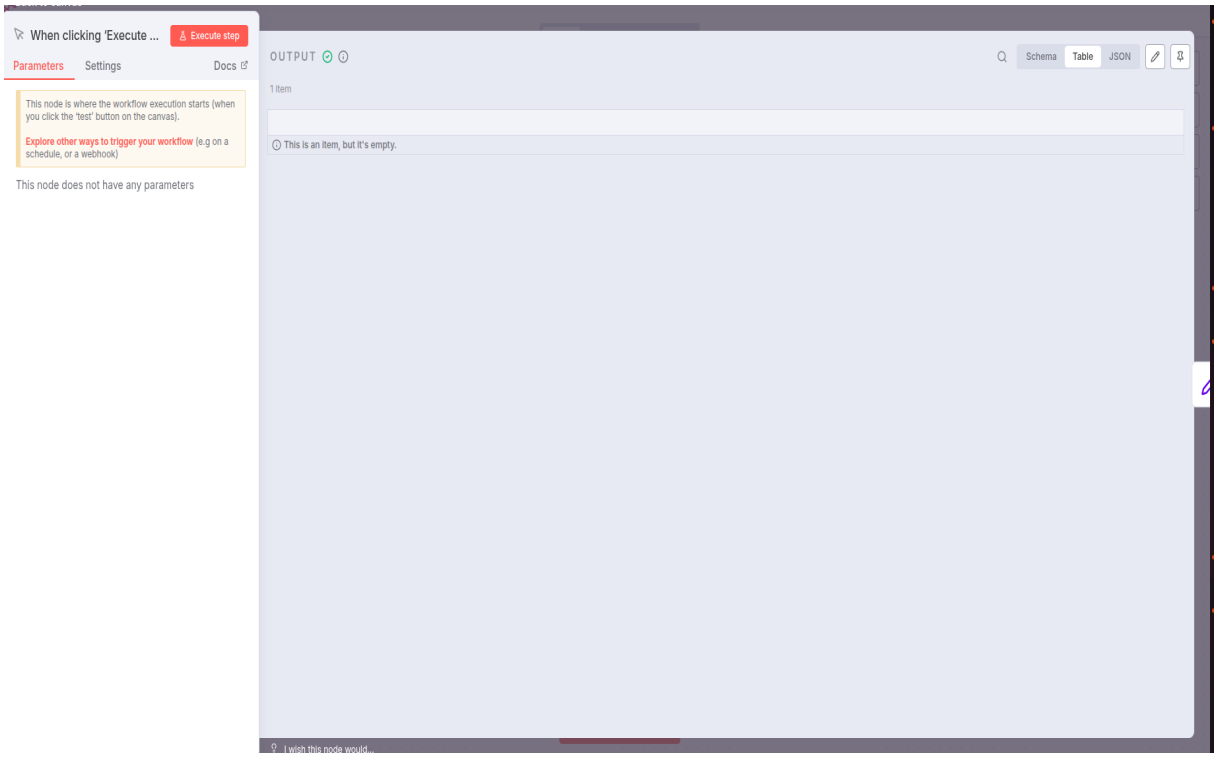
Por último, se añadirá un nodo Edit Fields(Set) para simular un procesamiento. Se añadirá un campo con el nombre de procesado-por y el valor de n8n-consumidor.



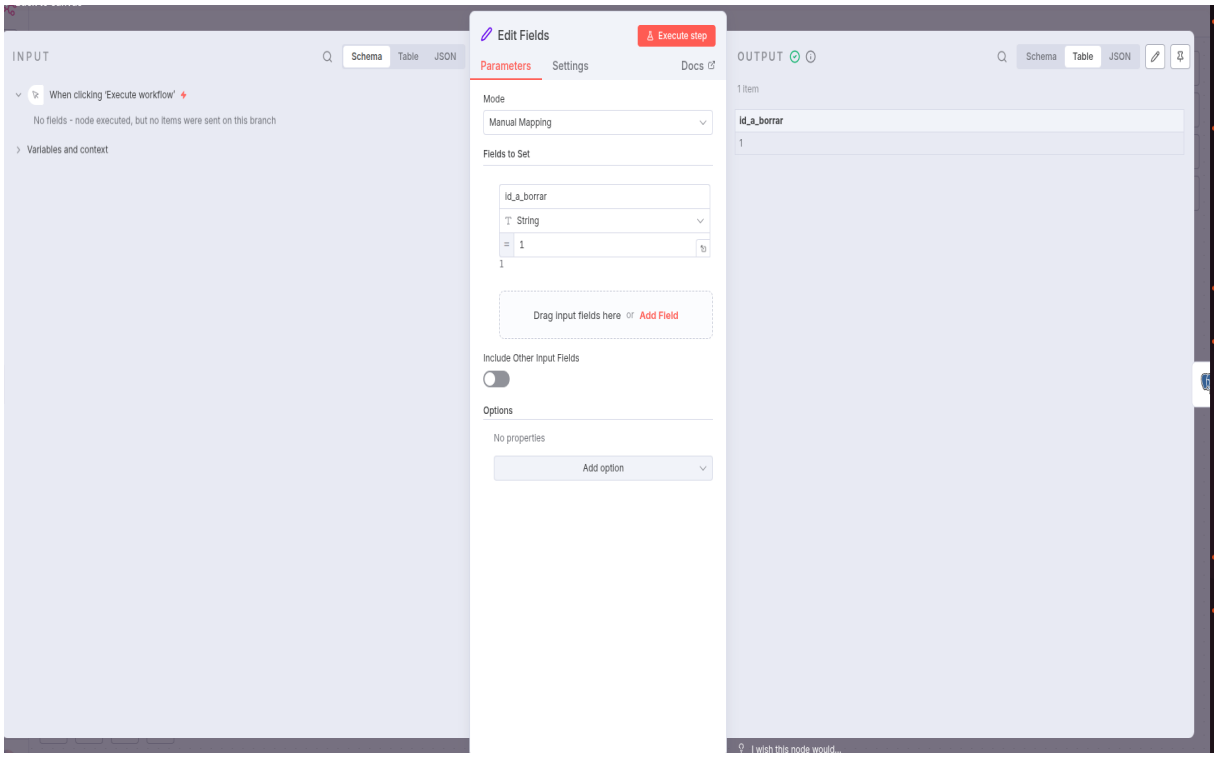
Capítulo 2

Ejercicio 1

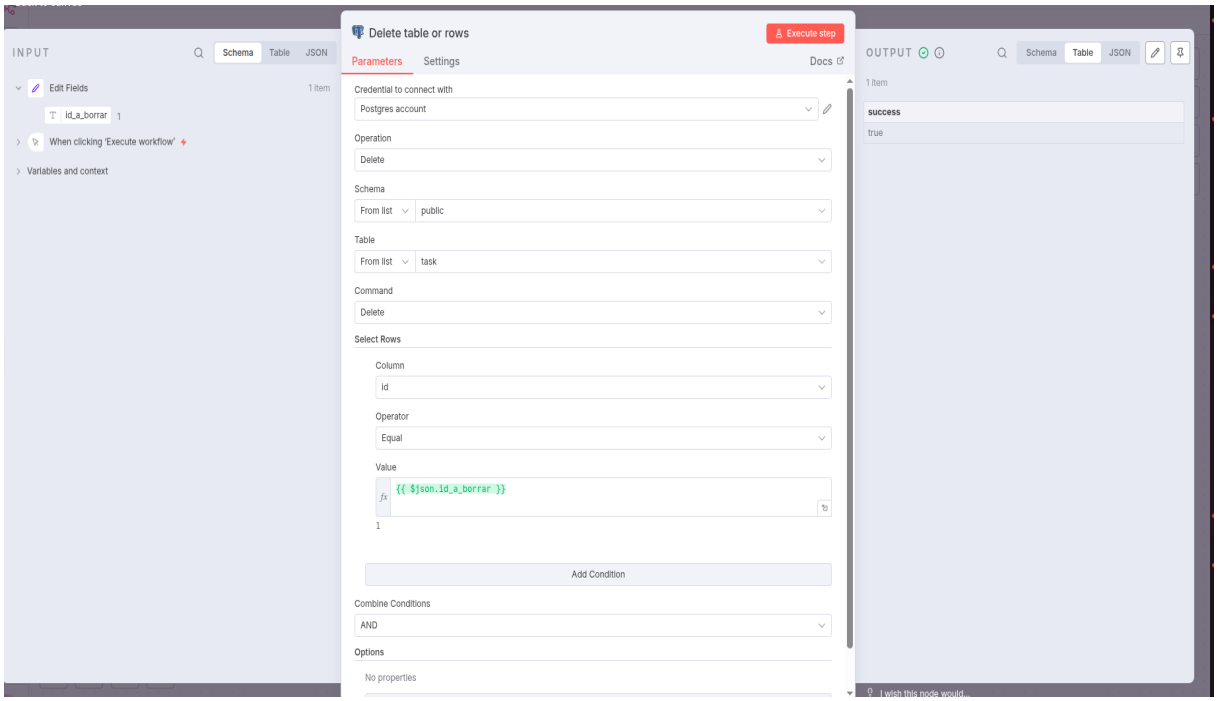
El objetivo del ejercicio 1 es crear un flujo de trabajo que elimine una tarea de la base de datos PostgreSQL. Para conseguir el objetivo lo primero que se debe hacer es añadir un Manual Trigger para dar comienzo al flujo de trabajo.



A continuación, se añade un Edit Fields(Set) para simular que tarea eliminar. Para definir la tarea se usará su ID.



El último paso será añadir un no PostgreSQL con la operación delete. Para eliminar la tarea se configura la condición WHERE para que eliminar la fila donde la id sea igual a la id pasada.



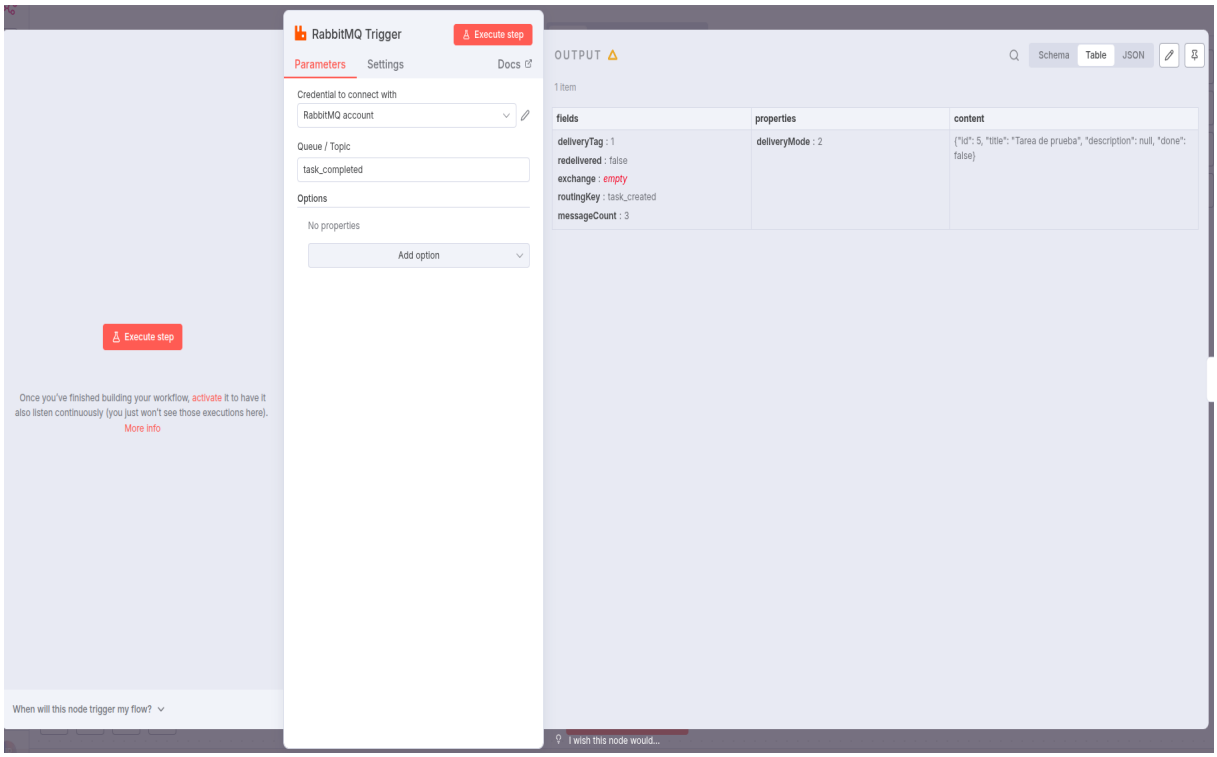
Capítulo 3

Ejercicio 2

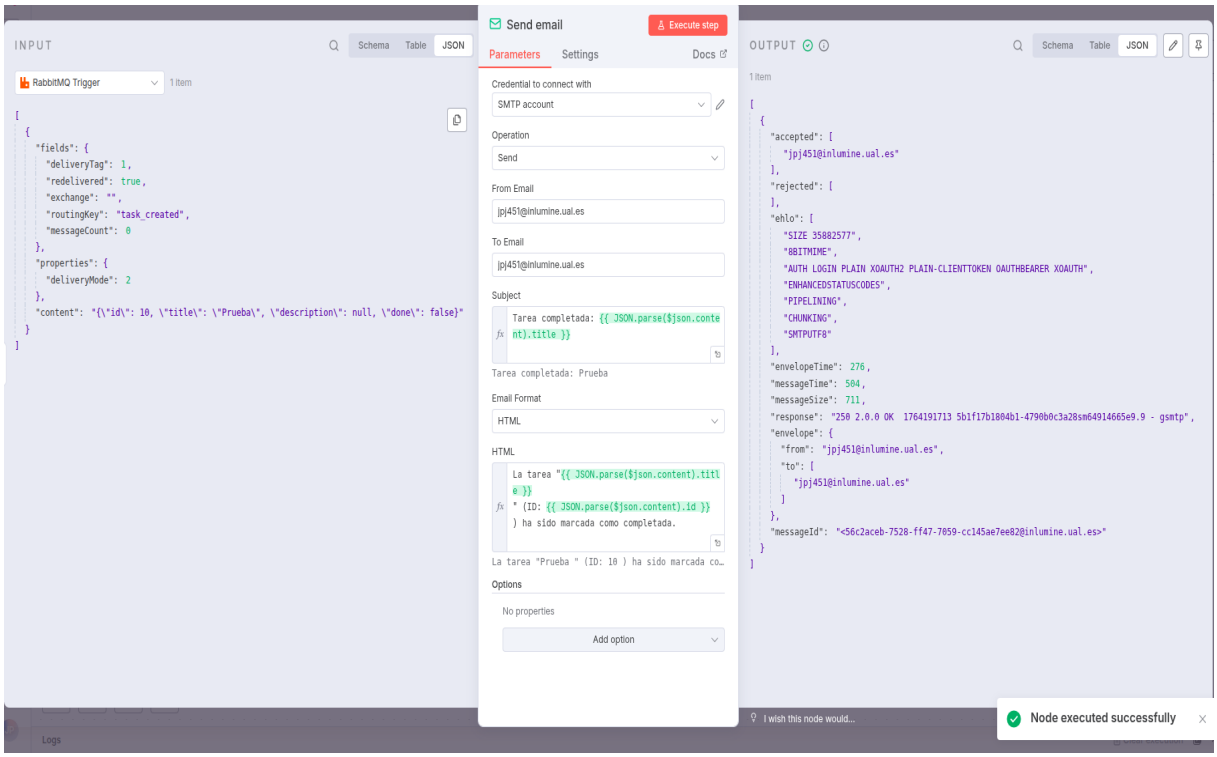
El objetivo del ejercicio 2 es crear un endpoint para actualizar un tarea para que pase de la cola creada a completada. Para conseguir el objetivo debemos añadir el endpoint de PUT en el archivo app.py.

```python
@app.route("/tasks/<int:task_id>/complete", methods=["PUT"])
def complete_task(task_id):
    task = Task.query.get(task_id)

    if not task:
        return jsonify({"error": "task not found"}), 404

    task.done = True
    db.session.commit()

    event = {
        "id": task.id,
        "title": task.title,
        "status": "completed"
    }

    publish_message("task_completed", event)

    return jsonify(task.to_dict()), 200
```

El siguiente paso será crear un flujo de trabajo que comience con un nodo RabbitMQ Trigger para que escuche la cola task-completed.



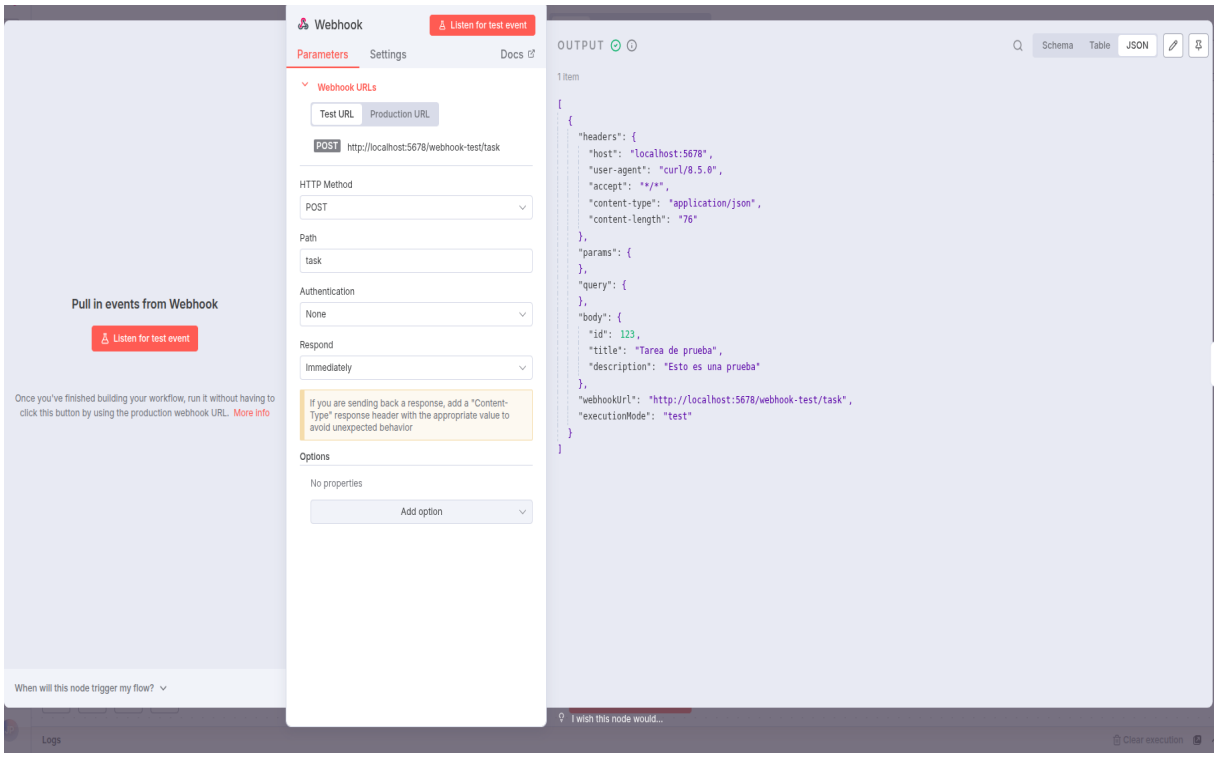
Cuando reciba un mensaje, el flujo debe usar un nodo Send Email para notificar que la tarea ha sido completada. En el email se pondrá el título y ID de la tarea.



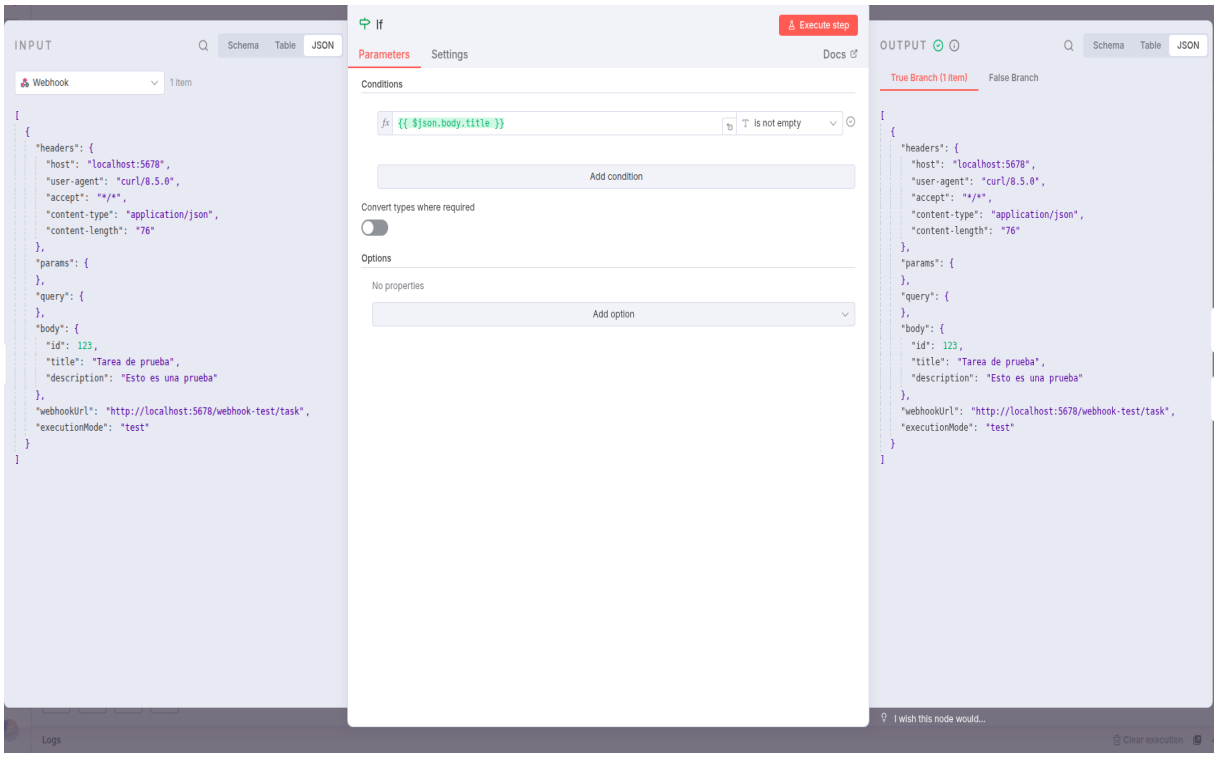
Capítulo 4

Ejercicio 3

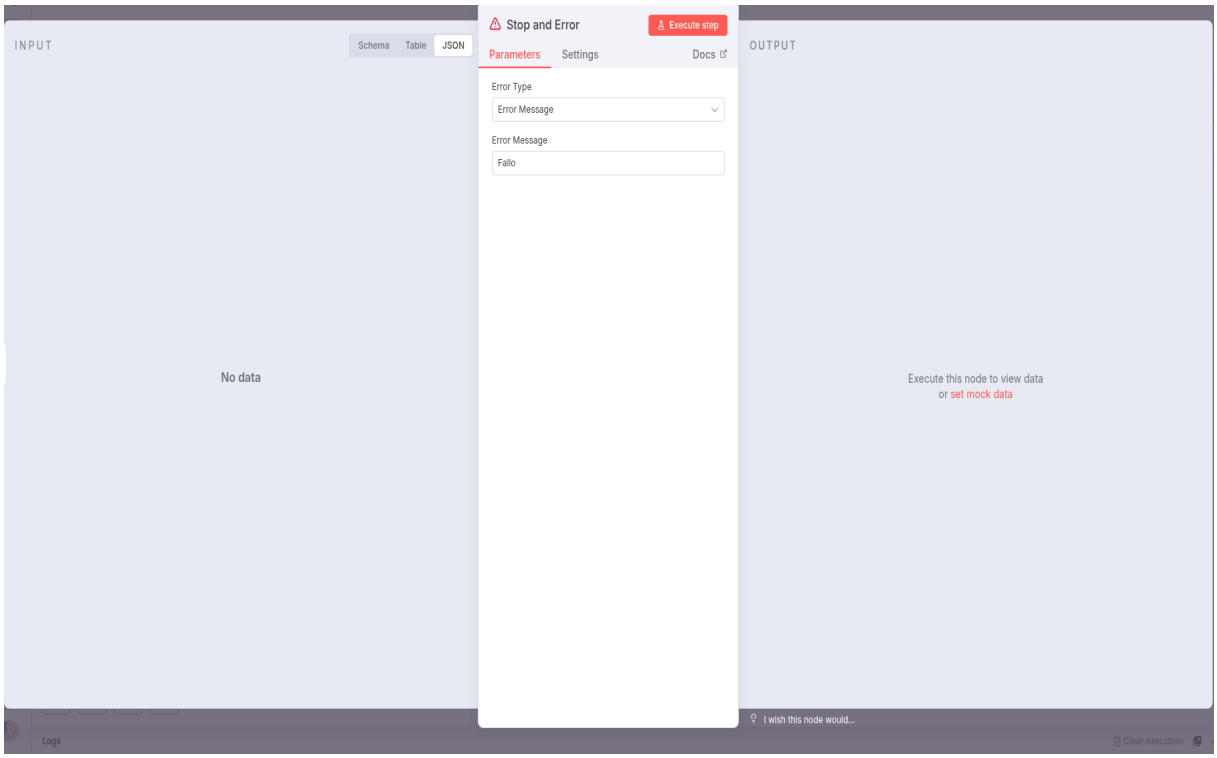
El objetivo del ejercicio 3 es reemplazar completamente la lógica del endpoint POST /tasks de la API de Flask con un flujo de trabajo de n8n. Para ello es necesario crear un nuevo flujo de trabajo que comience con un Webhook Trigger.



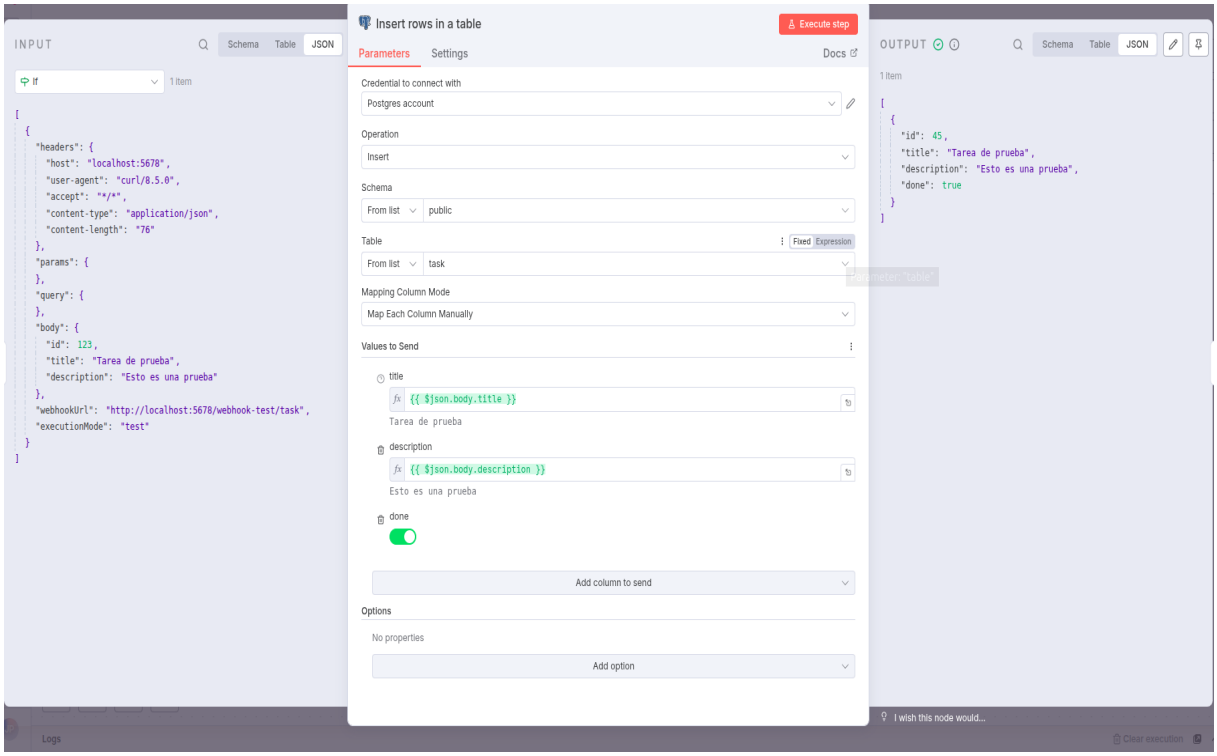
A continuación, se añadirá un nodo IF para validar que el campo de title existe y no esta vacio.



Si el campo esta vacio, se usará un nodo Stop and Error para comunicar el error al usuario.



Si la entrada de datos es correcta se dirigirá a un nodo PostgreSQL para insertar una nueva tarea a la tabla task.



El último paso seía publicar los datos de la tarea en la cola de task-created de Rabbit. Para ello, se usará un nodo RabbitMQ.