

AUTOMATIZACIÓN DE PROCESOS

Informe de Práctica 006

Diseño, implementación y documentación de flujos de trabajo avanzados utilizando n8n y microservicios.

HERRAMIENTAS

- ⚡ n8n Workflow
- ⌚ PostgreSQL
- ✉️ RabbitMQ

REPOSITORIO

🔗 github.com/aek676/itsi-2026

AUTOR

Anass El Jabiry Kaddir

FECHA

7 de diciembre de 2025

Desarrollo Guiado

01

Configuración, Credenciales y Base de Datos

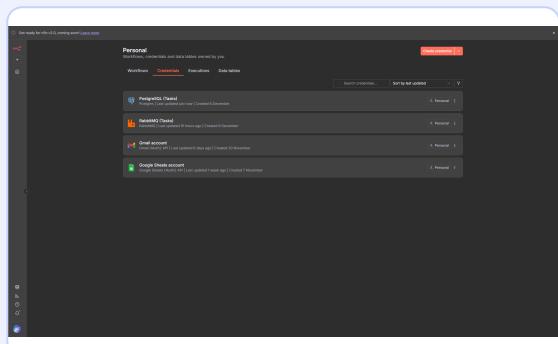
1. UNIFICACIÓN EN DOCKER COMPOSE

Se unificó todo en docker-compose.yml. n8n y microservicios comparten la red interna.

```
services:  
  n8n:  
    image: docker.n8n.io/n8nio/n8n:latest  
    ports: ["5678:5678"]  
    volumes: [n8n_data:/home/node/.n8n]  
  web:  
    build: ./task-manager-service/web  
    environment:  
      - DATABASE_URL=postgresql://user:pw@db:5432/db  
      - RABBITMQ_URL=amqp://guest:guest@mq:5672/%2F  
    depends_on: [db, mq]  
  db: # PostgreSQL  
  mq: # RabbitMQ
```

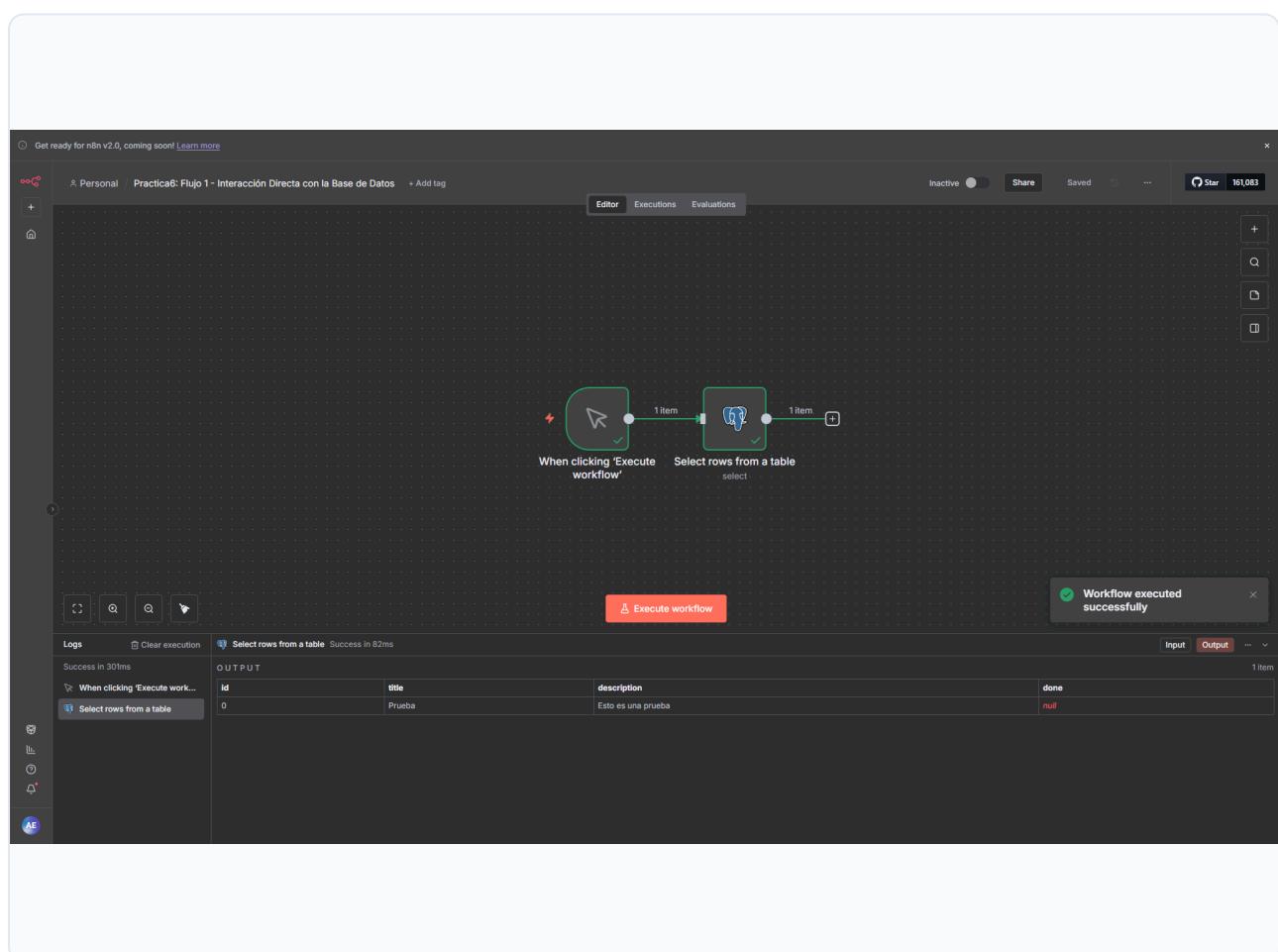
2. CONFIGURAR CREDENCIALES

Acceso para **PostgreSQL** (puerto 5432) y **RabbitMQ** (puerto 5672).



3. Flujo 1 - Interacción Directa con la Base de Datos

Workflow de lectura y escritura en PostgreSQL.



Desarrollo Guiado (Cont.)

01

Parte 2: Diseño de Patrones Asíncronos

4. DISEÑO DEL FLUJO 2 (ASÍNCRONO)

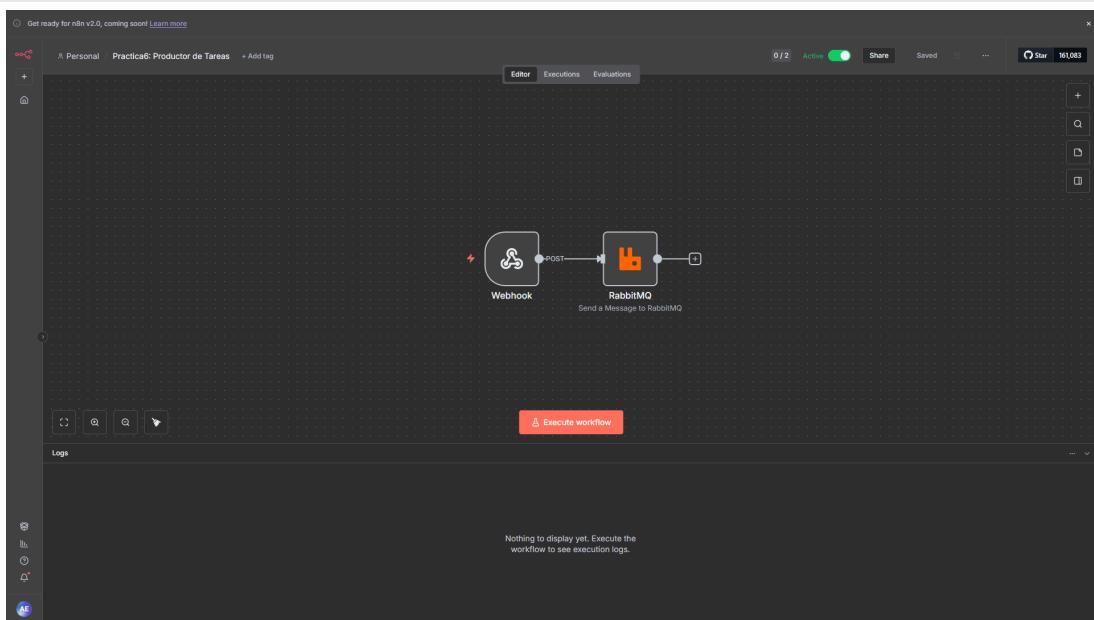
Se implementaron dos workflows interconectados mediante **RabbitMQ**.

El **Productor** recibe una petición web y encola el mensaje.

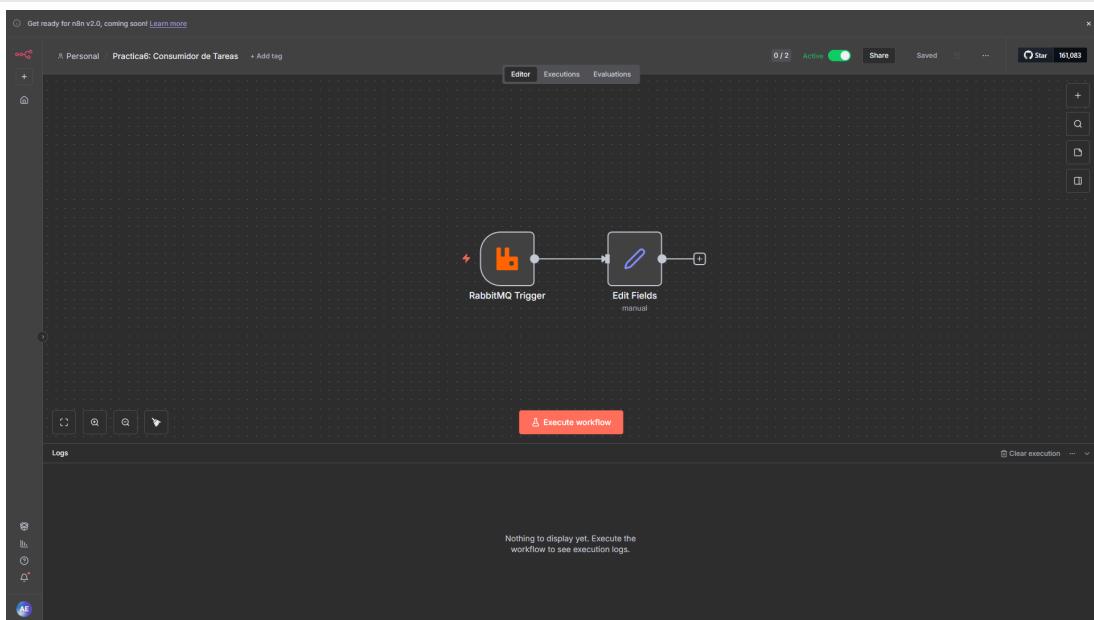
El **Consumidor** escucha la cola y procesa el mensaje.

💡 Nota: Interruptor "Active" activado.

DISEÑO PRODUCTOR (WEBHOOK → RABBITMQ)



DISEÑO CONSUMIDOR (RABBITMQ TRIGGER)



Verificación del Sistema

Paso 5: Pruebas de Ejecución (Parte 1)

01

Validación del ciclo completo: Petición externa (Postman) → Ejecución Productor (n8n).

1. PETICIÓN POSTMAN

The screenshot shows the Postman interface with the following details:

- Request Method:** POST
- URL:** <http://localhost:5678/webhook/e28ce6ba-1af9-44fb-aaf9-22f9af0a3be5>
- Body (JSON):**

```
1 [{}]
2   "message": "Workflow was started"
3 ]
```
- Status:** 200 OK
- Time:** 71 ms
- Size:** 534 B

2. EJECUCIÓN PRODUCTOR

The screenshot shows the n8n interface with the following details:

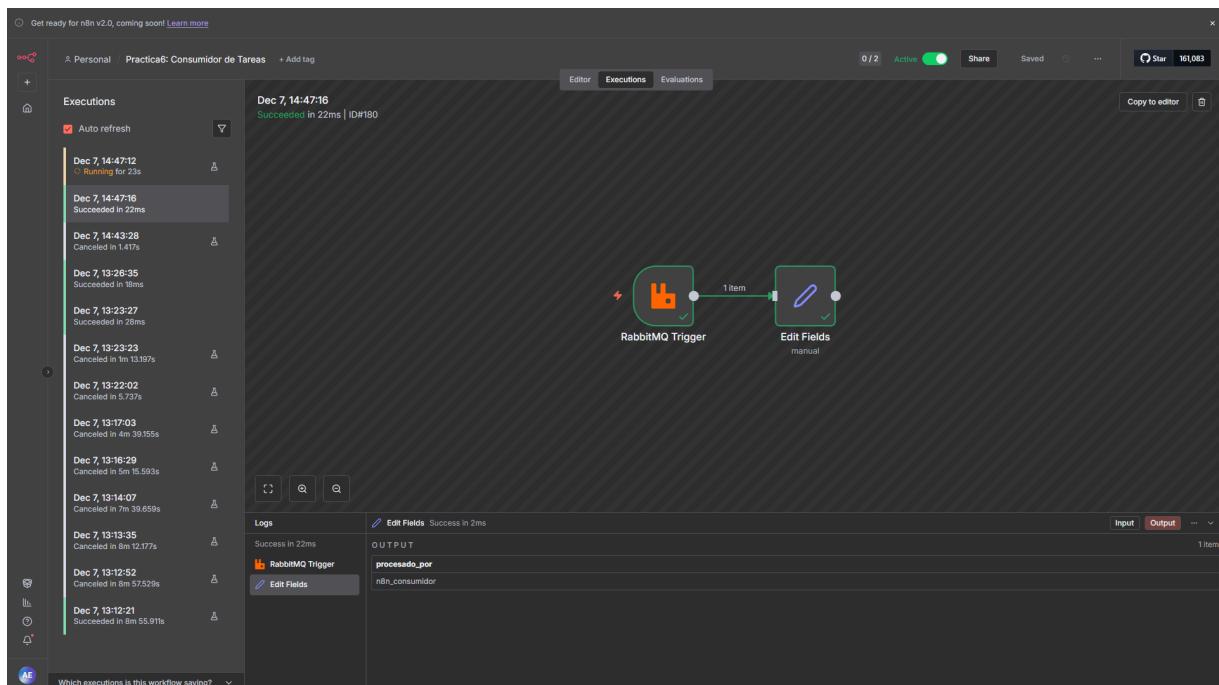
- Workflow:** A sequence of nodes: Webhook → POST → RabbitMQ.
- Logs:** Success in 133ms
- Output:** success: true

Verificación del Sistema

Paso 5: Pruebas de Ejecución (Parte 2)

01

3. EJECUCIÓN CONSUMIDOR (CONFIRMACIÓN DE RECEPCIÓN)

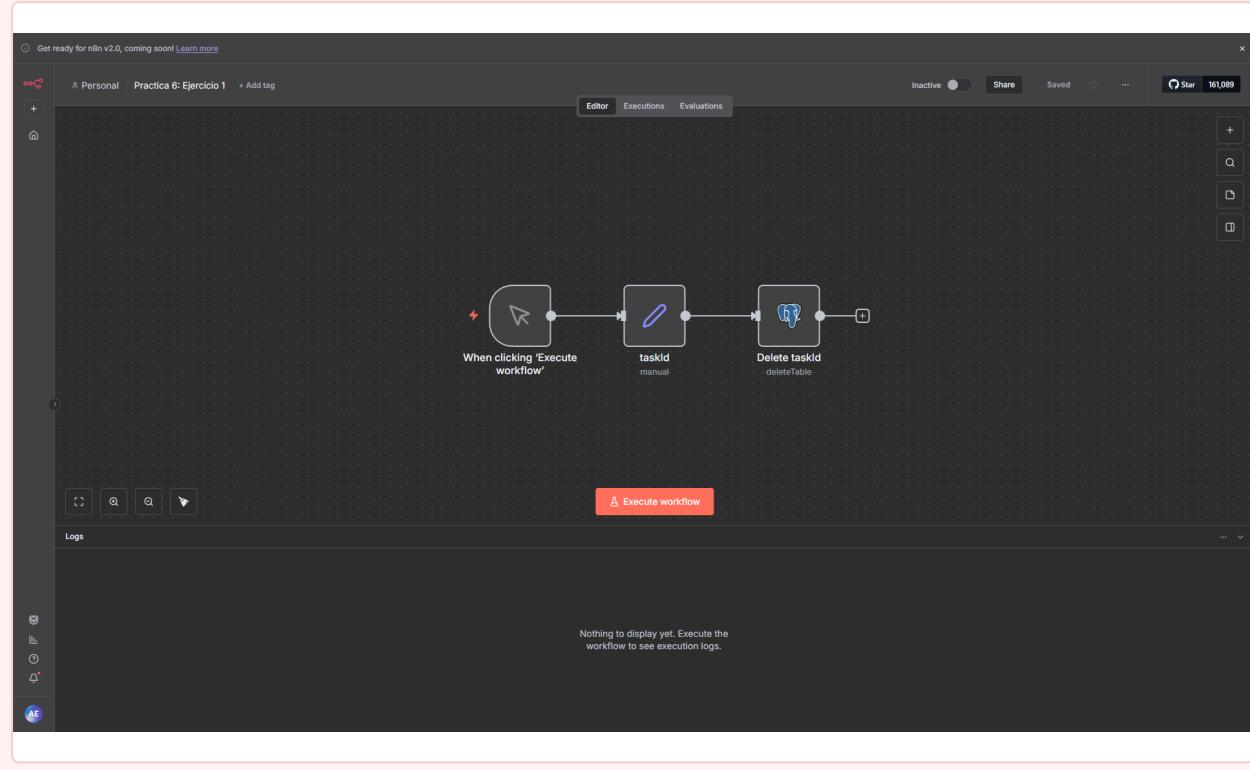


Ejercicio 1: Eliminación

02

Diseño y Configuración

1. DISEÑO DEL FLUJO



2. CONFIGURACIÓN NODO DELETE

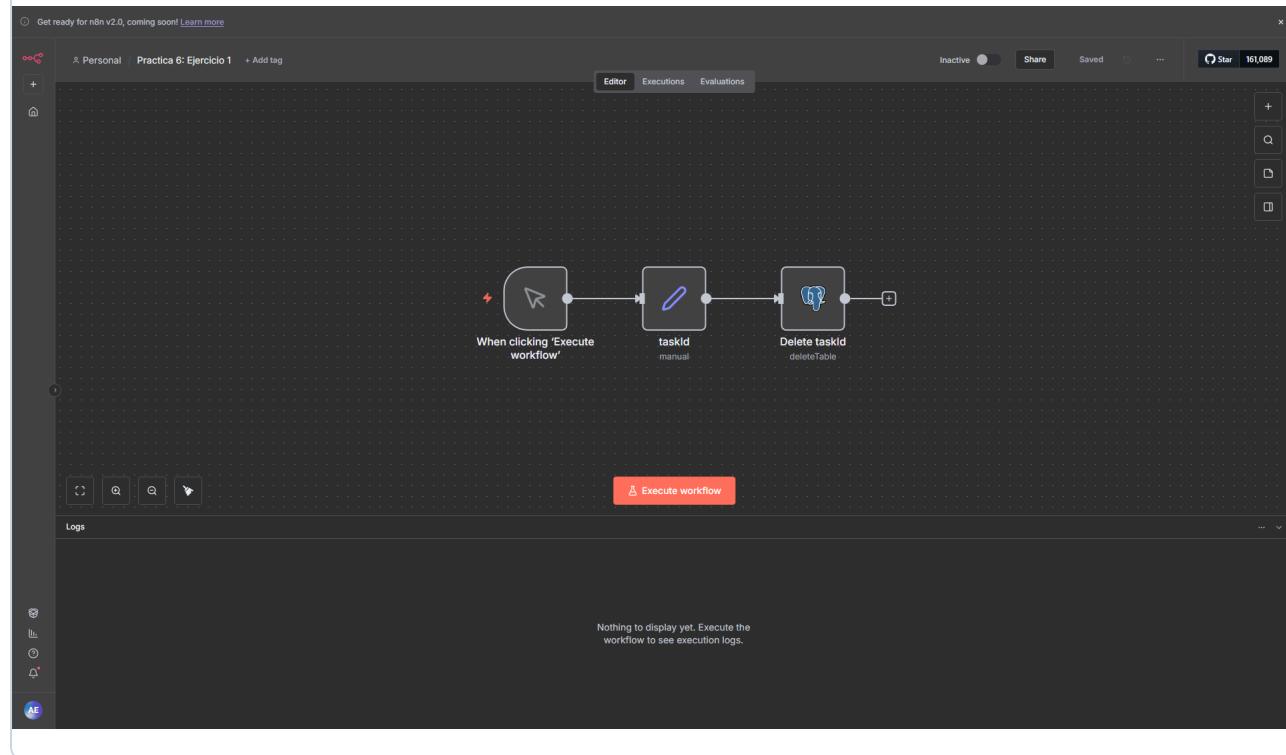
The screenshot shows the configuration dialog for the 'Delete taskId' node. The dialog is divided into two main sections: 'INPUT' on the left and 'OUTPUT' on the right. In the 'INPUT' section, there's a dropdown menu set to 'taskId'. Below it, a message says 'No input data yet' and 'Execute previous nodes (From the earliest node that needs it)'. In the 'OUTPUT' section, there's a message 'Execute this node to view data or set mock data'. The central part of the dialog contains settings for the 'Delete' operation, including 'Schema' (From list: public), 'Table' (From list: task), and 'Command' (Delete). Under 'Select Rows', there's a condition: 'Column: id', 'Operator: Equal', and 'Value: {{ \$json.taskId }}'. There are also sections for 'Combine Conditions' (set to AND) and 'Options' (No properties).

Ejercicio 1: Eliminación

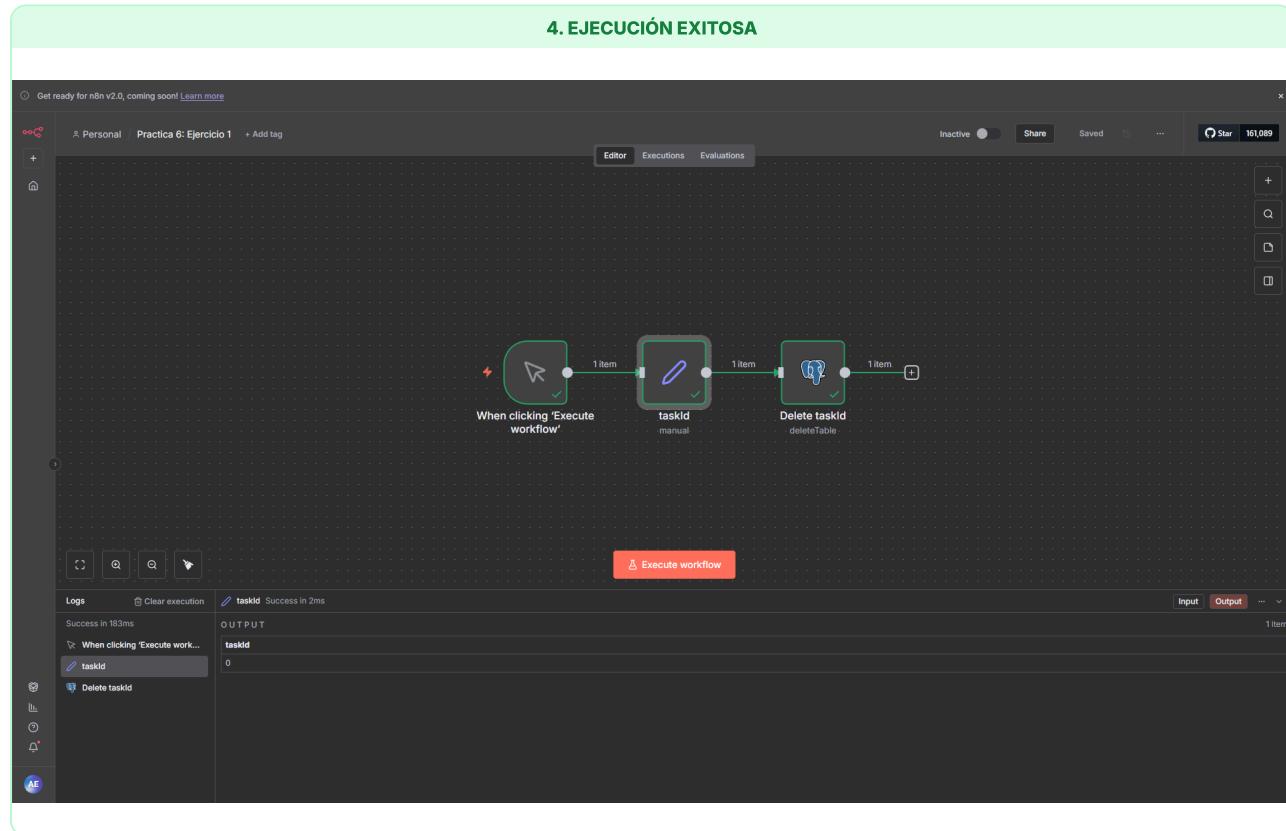
02

Datos y Ejecución

3. DATOS INICIALES (BD)



4. EJECUCIÓN EXITOSA

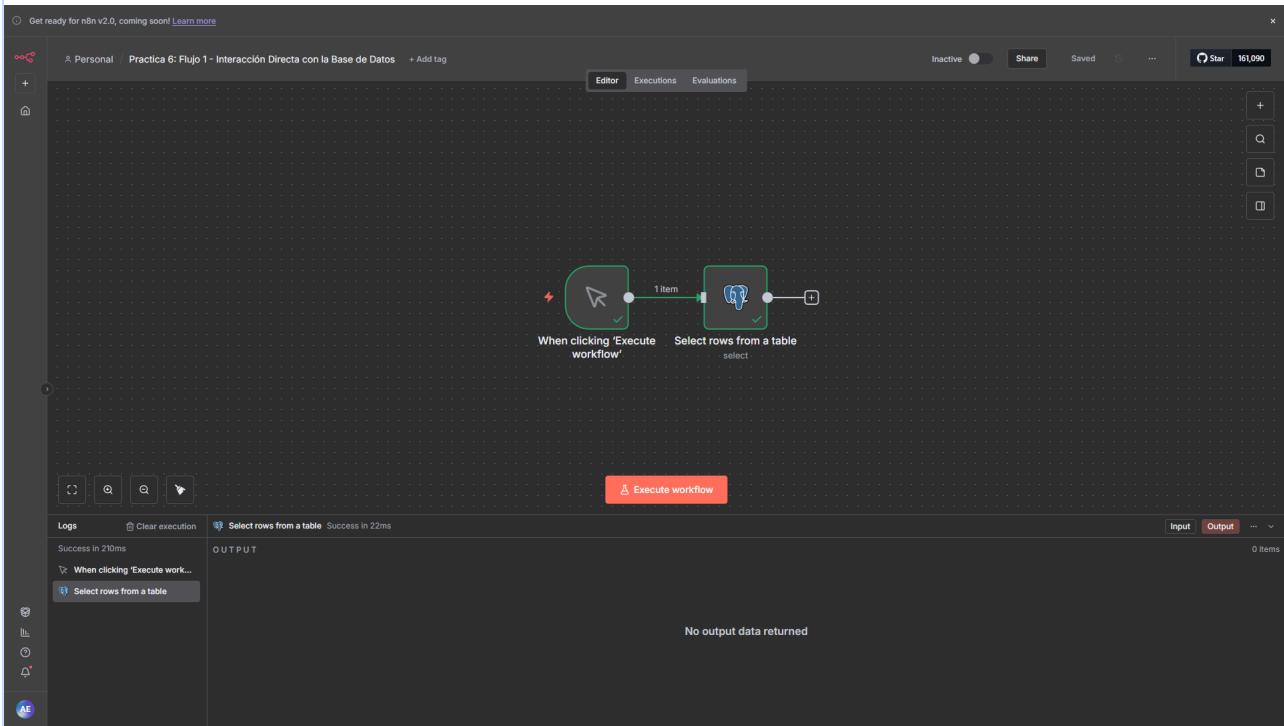


Ejercicio 1: Eliminación

02

Verificación Final

5. VERIFICACIÓN DE BORRADO



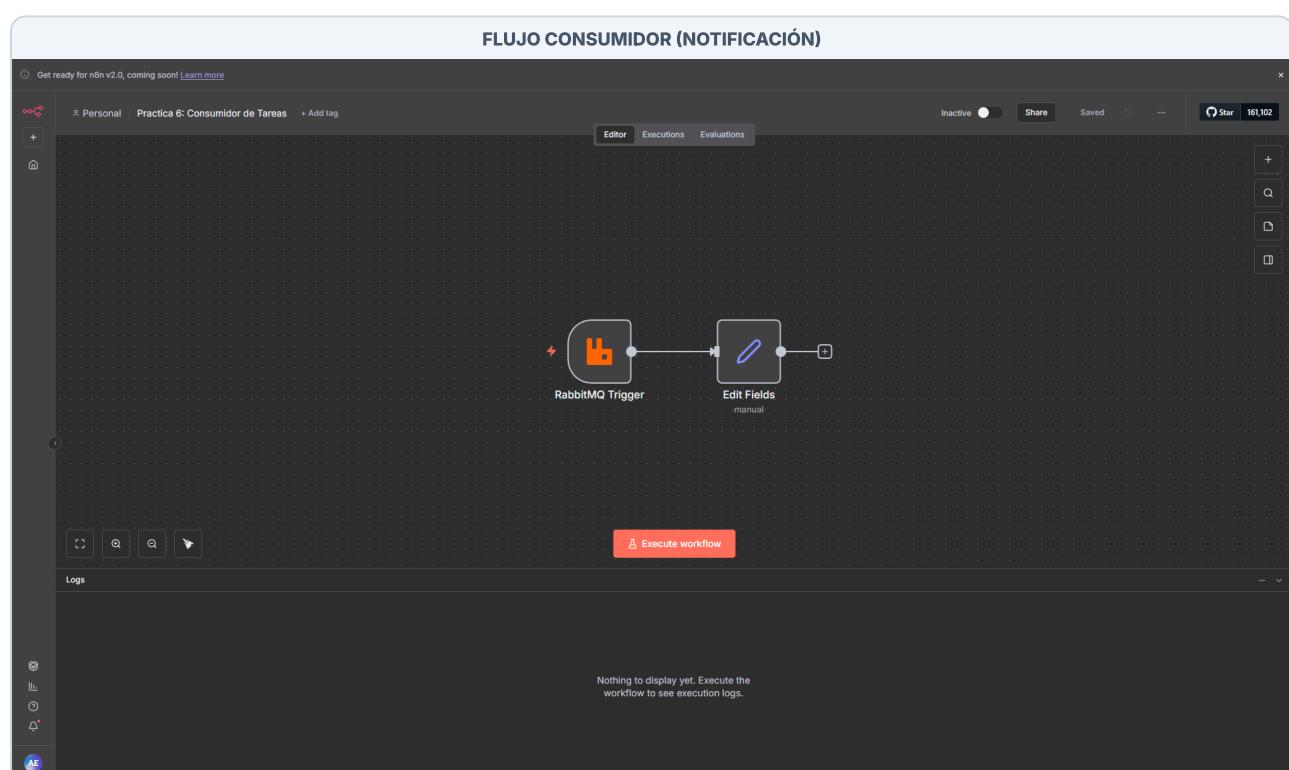
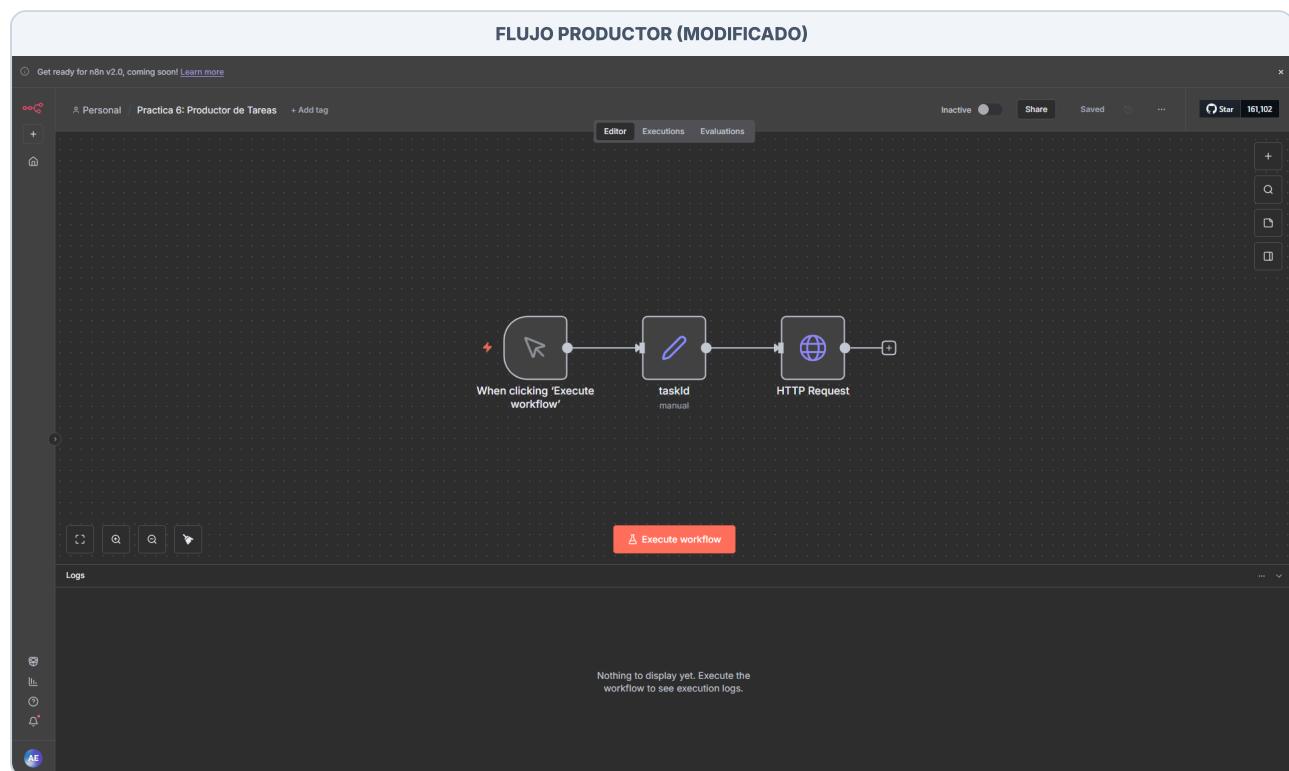
Ejercicio 2: Notificaciones

03

Diseño de Flujos

OBJETIVO

Reemplazar consumidor Python por n8n (Queue: task_completed).



Ejercicio 2: Notificaciones

03

Verificación de Ejecución

EJECUCIÓN PRODUCTOR

The screenshot shows the n8n interface for the 'Práctica 6: Productor de Tareas' workflow. The left sidebar lists various executions, with the most recent one, 'Dec 7, 15:48:28', highlighted. The main panel displays the workflow diagram and its logs.

Workflow Diagram:

```
graph LR; Start(( )) --> Click[When clicking 'Execute workflow']; Click --> TaskId[taskID manual]; TaskId --> HttpRequest[HTTP Request]
```

Logs:

- Success in 59ms
- Succeeded in 61ms
- Succeeded in 15ms
- Succeeded in 11ms
- Succeeded in 18ms
- Error in 11ms
- Error in 19ms
- Error in 28ms
- Error in 20ms
- Error in 37ms
- Error in 212ms
- Succeeded in 16ms
- Succeeded in 133ms
- Succeeded in 117ms

HTTP Request Log:

```
HTTP Request Success in 38ms
C O U T P U T
task
description : Esta es una prueba para completar una tarea con n8n
done : true
id : 2
title : Tarea a completar
```

EJECUCIÓN CONSUMIDOR

The screenshot shows the n8n interface for the 'Práctica 6: Ejercicio 2' workflow. The left sidebar lists executions, with the most recent one, 'Dec 7, 15:48:28', highlighted. The main panel displays the workflow diagram and its logs.

Workflow Diagram:

```
graph LR; RabbitMQTrigger[RabbitMQ Trigger] --> EditFields[Edit Fields manual]
```

Logs:

- Success in 19ms
- RabbitMQ Trigger
- Edit Fields Success in 3ms

Output:

```
C O U T P U T
This is an item, but it's empty.
```

Ejercicio 3: API con Webhook

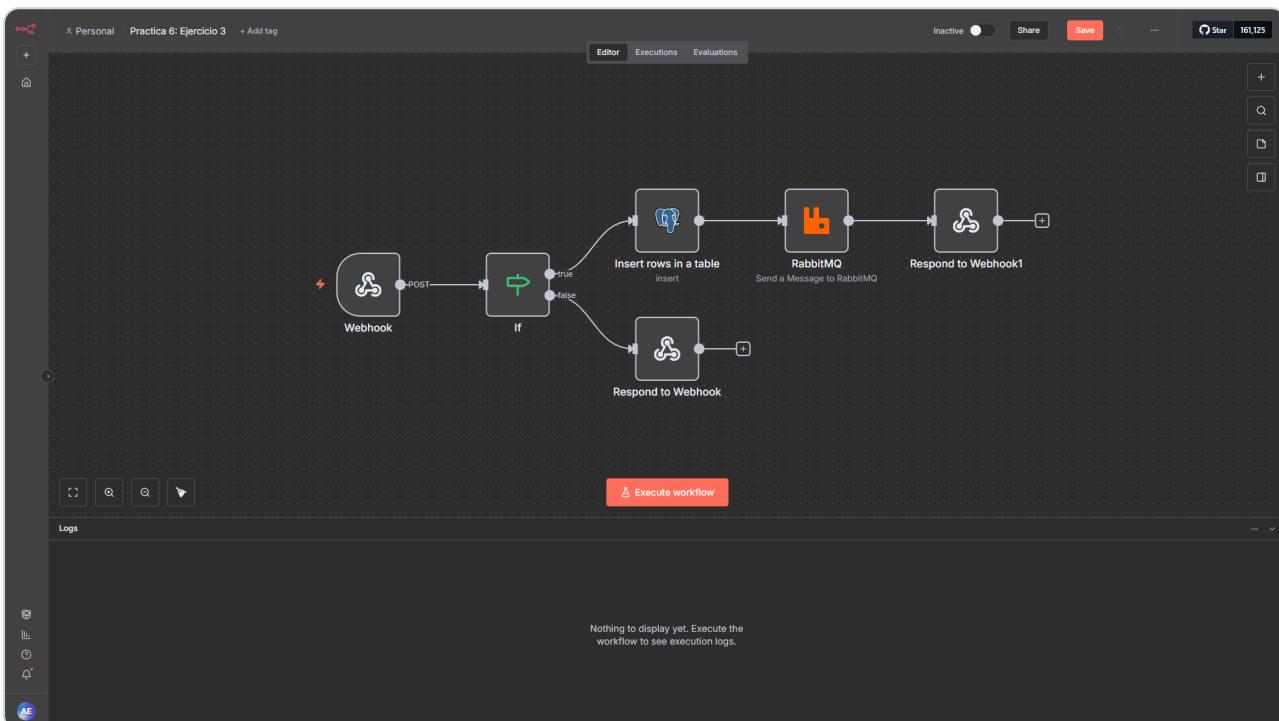
04

Diseño y Caso de Éxito

OBJETIVO

Reemplazo de API Flask con n8n (Webhook + DB + RabbitMQ + Response).

1. DISEÑO DEL WORKFLOW



2. POSTMAN: 201 CREATED

The screenshot shows the Postman application interface with the following details:

- The title bar says 'HTTP POST: Tasks'.
- The URL field contains 'http://localhost:5678/webhook-test/519da9a5-3277-4835-b62f-58979acf9cb9'.
- The 'Body' tab is selected, showing raw JSON data:

```
1 [  
2   {"title": "Ejercicio 3",  
3   "description": "Prueba ejercicio 3"}]
```
- The status bar at the bottom indicates 'Status: 201 Created Time: 258 ms Size: 521 B'.
- The response body is displayed in the 'Pretty' tab:

```
1 [  
2   "success": true  
3 ]
```

Ejercicio 3: API con Webhook

04

Manejo de Errores (Bonus)

3. POSTMAN: 400 BAD REQUEST

The screenshot shows the Postman interface with a red header bar indicating a '400 BAD REQUEST'. The main window displays a POST request to 'http://localhost:5678/webhook-test/519da9a5-3277-4835-b62f-58979afc9cb9'. The 'Body' tab is selected, showing raw JSON input:

```
1 [{}  
2 .... "description": "Prueba ejercicio 3"  
3 ]
```

The 'Test Results' tab shows the response status: Status: 400 Bad Request Time: 90 ms Size: 564 B. The response body is:

```
1 [{}  
2     "error": "Falta el campo title",  
3     "status": "Bad Request"  
4 ]
```

At the bottom, it shows 'WSL: Debian' and 'Connected to Discord'.