Requerimientos, necesidades y procesos.

1. Documento de Requisitos y Necesidades

Este proyecto resuelve la necesidad de auditar las preguntas que los usuarios envían al chatbot, vinculándolas directamente al usuario para una posterior consulta administrativa.

1.1. Necesidades Cubiertas

Categoría	Descripción
Auditoría	Registrar cada pregunta del usuario y la respuesta final del bot en una base de datos centralizada.
Trazabilidad	Organizar todos los registros de interacciones por un **ID de Usuario único**, independientemente del número de veces que interactúe.
Persistencia	Mantener un historial de usuarios (`usuarios`) y sus interacciones (`consultas`) de forma permanente en PostgreSQL.
Automatización	Utilizar una herramienta (n8n) para gestionar la lógica de identificación de usuario, ejecución de IA, y persistencia de datos sin intervención manual.

1.2. Requisitos Previos (Stack Tecnológico)

1. **Plataforma de Automatización:** Instalación de **n8n** (Cloud o *Self-hosted*).
2. **Mensajería:** Credenciales de **Telegram Bot** (Token API).
3. **Inteligencia Artificial:** Credenciales de **Google Gemini / Gemini Chat Model** (API Key).
4. **Base de Datos:** Servidor **PostgreSQL** accesible y credenciales configuradas en n8n.

---

2. Definición del Esquema de la Base de Datos (PostgreSQL)

Se requieren dos tablas con las siguientes estructuras. Es **crítico** que la relación `id_user` sea una clave foránea (FK) para asegurar la integridad de los datos.

2.1. Tabla `users` (Usuarios)

Esta tabla mantiene la clave de vinculación con el bot de Telegram para auditoría.

Campo	Tipo de Dato	Traducción	Justificación
`id_user`	`SERIAL PRIMARY KEY`	ID de Usuario	Se usa **`id_user`** para mantener la **nomenclatura estándar** de la clave primaria.
`telegram_id`	`BIGINT UNIQUE NOT NULL`	ID de Telegram	Se usa **`BIGINT`** porque los IDs de Telegram son números de 64 bits y **`UNIQUE`** asegura que cada ID corresponda a un solo registro.
`name`	`VARCHAR(100)`	Nombre	Se usa **`VARCHAR(100)`** porque es suficiente para capturar el nombre completo o alias del usuario de Telegram.
`registration_date`	`TIMESTAMP WITH TIME ZONE`	Fecha de Registro	Registra el evento de negocio: la primera interacción del usuario.

2.2. Tabla `interactions` (Auditoría)

Esta tabla almacena las preguntas en lenguaje natural y las respuestas del bot, vinculadas a la tabla `users`.

Campo	Tipo de Dato	Traducción	Justificación
`id_interaction`	`SERIAL PRIMARY KEY`	ID de Interacción	Identificador único de cada pregunta.
`id_user`	`INTEGER (FK)`	ID de Usuario	Clave foránea a la tabla `users` para organizar la auditoría.

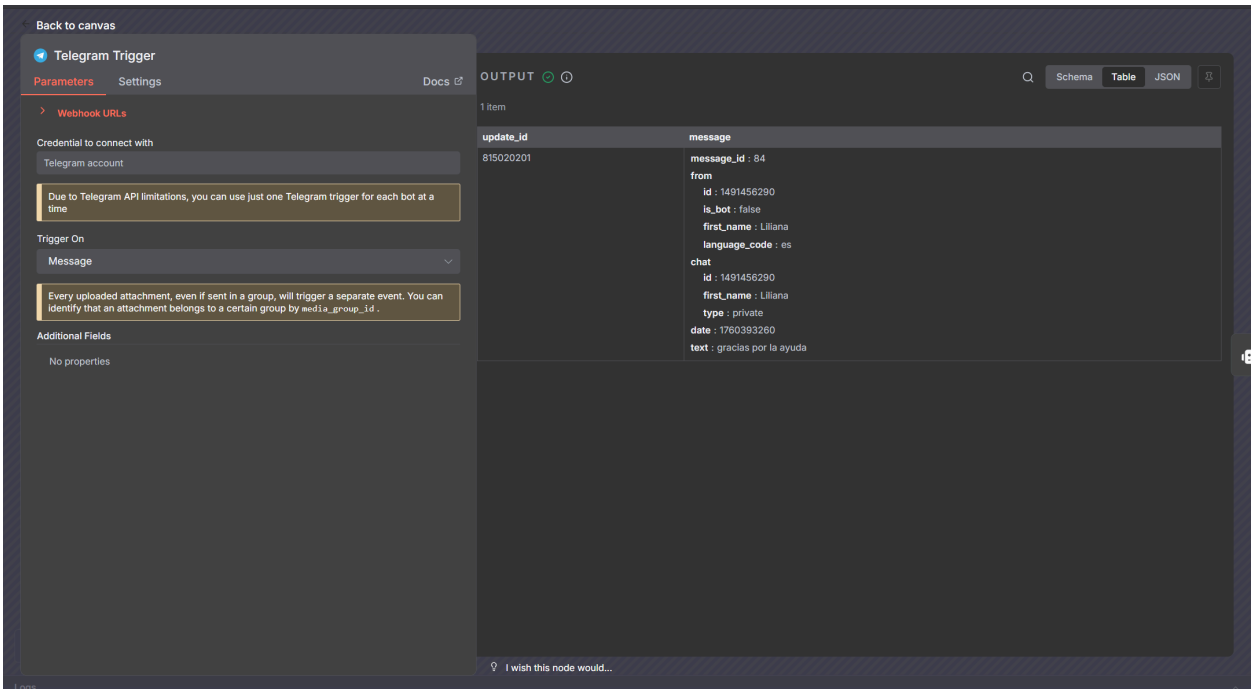
`user_question`	`TEXT NOT NULL`	Pregunta del Usuario	La pregunta en lenguaje natural. `TEXT` es flexible.
`question_date`	`TIMESTAMP WITH TIME ZONE`	Fecha de Pregunta	Momento exacto de la pregunta.
`bot_response`	`TEXT`	Respuesta del Bot	La respuesta final generada por el AI Agent.

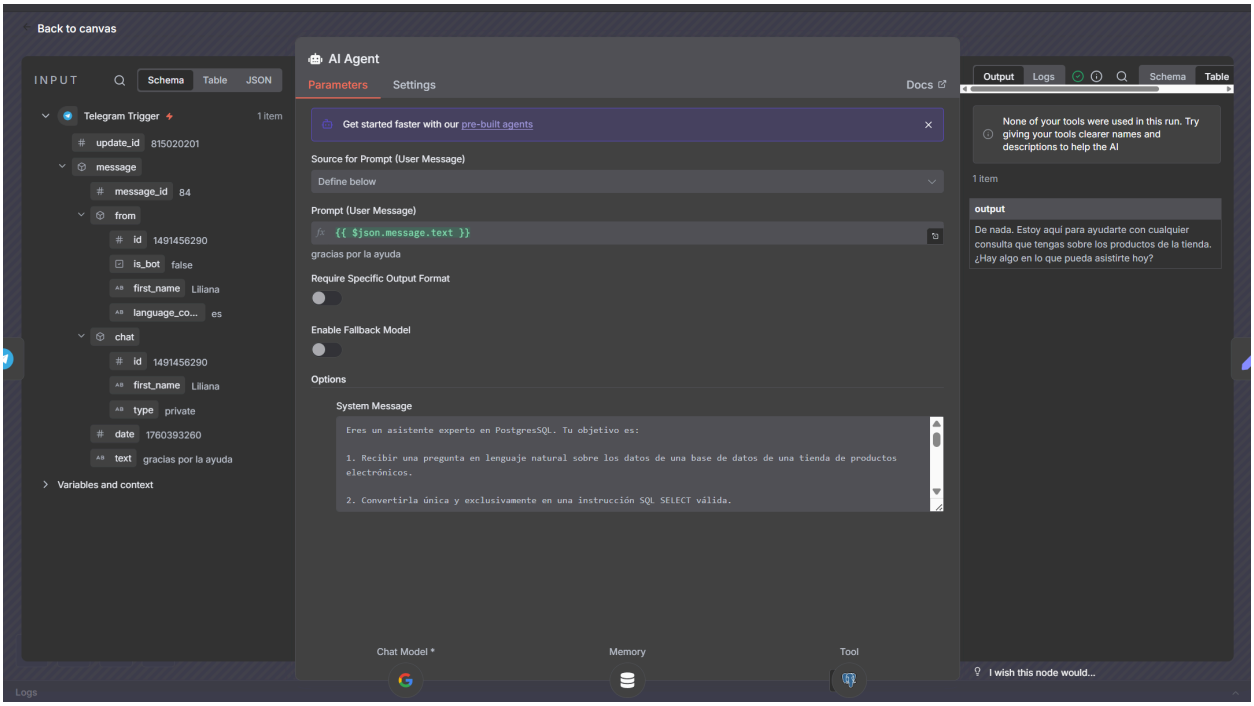
3. Proceso del Workflow

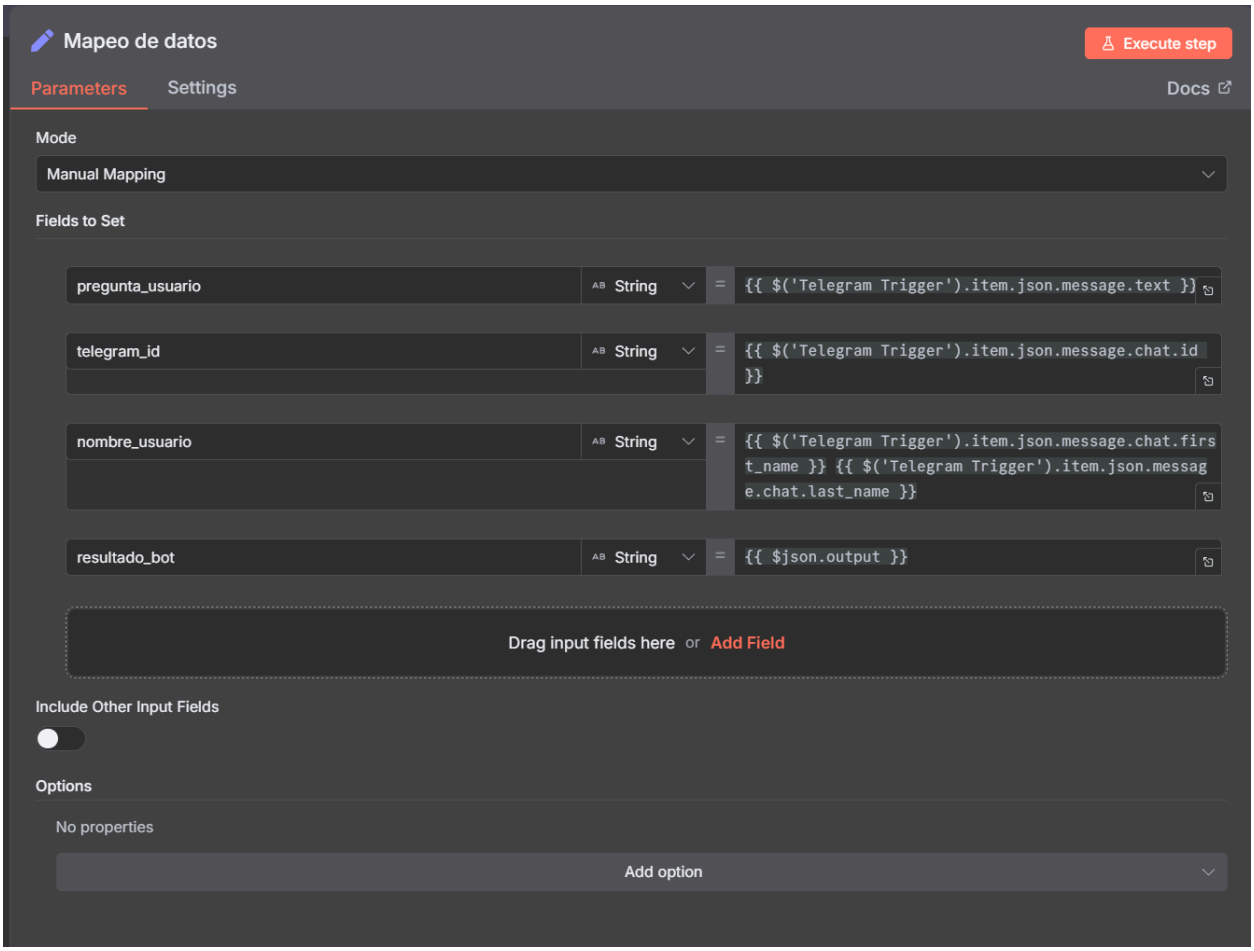
El flujo se divide en tres fases principales: **Preparación**, **Identificación y Bifurcación**, **Auditoría Final y Respuesta**.

Fase A: Preparación y Ejecución de IA

#	Nodo	Propósito	Configuración Crítica
1	**Telegram Trigger**	Iniciar el *workflow* con un mensaje entrante.	**Updates:** `message`.
2	**AI Agent**	Procesar la pregunta y generar una respuesta basada en la Tool (Postgres).	**Input:** `{{ $json["message"]["text"] }}`. **Output:** `Response`.
3	**Mapeo de datos**	Centralizar datos clave del Trigger y del Agente en un solo ítem.	Crear campos clave (usando el nombre del nodo `Telegram Trigger` y el `$json` del Agente): • `telegram_id` • `nombre_usuario` • `pregunta_usuario` • `resultado_bot`


Back to canvas
Telegram Trigger
Parameters
Settings
Docs
Webhook URLs
Credential to connect with
Telegram account
Due to Telegram API limitations, you can use just one Telegram trigger for each bot at a time
Trigger On
Message
Every uploaded attachment, even if sent in a group, will trigger a separate event. You can identify that an attachment belongs to a certain group by media_group_id.
Additional Fields
No properties
OUTPUT
Schema
Table
JSON
1 item
update_id
message
815020201
message_id : 84
from
id : 1491456290
is_bot : false
first_name : Liliana
language_code : es
chat
id : 1491456290
first_name : Liliana
type : private
date : 1760393260
text : gracias por la ayuda
I wish this node would...


Back to canvas
INPUT
Schema
Table
JSON
Telegram Trigger
1 item
update_id 815020201
message
message_id 84
from
id 1491456290
is_bot false
first_name Liliana
language_co... es
chat
id 1491456290
first_name Liliana
type private
date 1760393260
text gracias por la ayuda
Variables and context
AI Agent
Parameters
Settings
Docs
Get started faster with our pre-built agents
Source for Prompt (User Message)
Define below
Prompt (User Message)
{{ $json.message.text }}
gracias por la ayuda
Require Specific Output Format
Enable Fallback Model
Options
System Message
Eres un asistente experto en PostgresSQL. Tu objetivo es:
1. Recibir una pregunta en lenguaje natural sobre los datos de una base de datos de una tienda de productos electrónicos.
2. Convertirla única y exclusivamente en una instrucción SQL SELECT válida.
Chat Model *
Memory
Tool
Output
Logs
Schema
Table
None of your tools were used in this run. Try giving your tools clearer names and descriptions to help the AI
1 item
output
De nada. Estoy aquí para ayudarte con cualquier consulta que tengas sobre los productos de la tienda. ¿Hay algo en lo que pueda asistirte hoy?
I wish this node would...
Logs

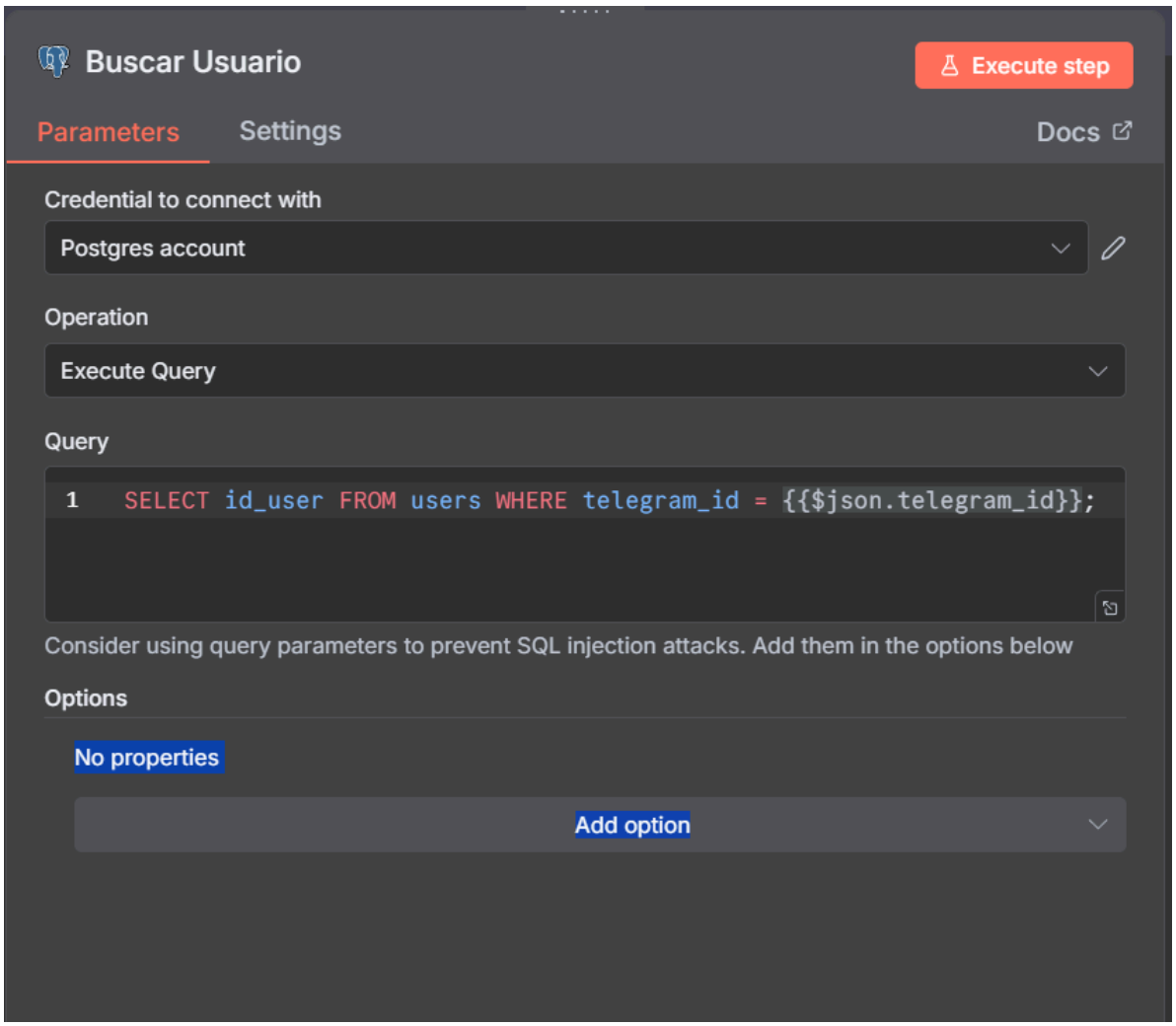


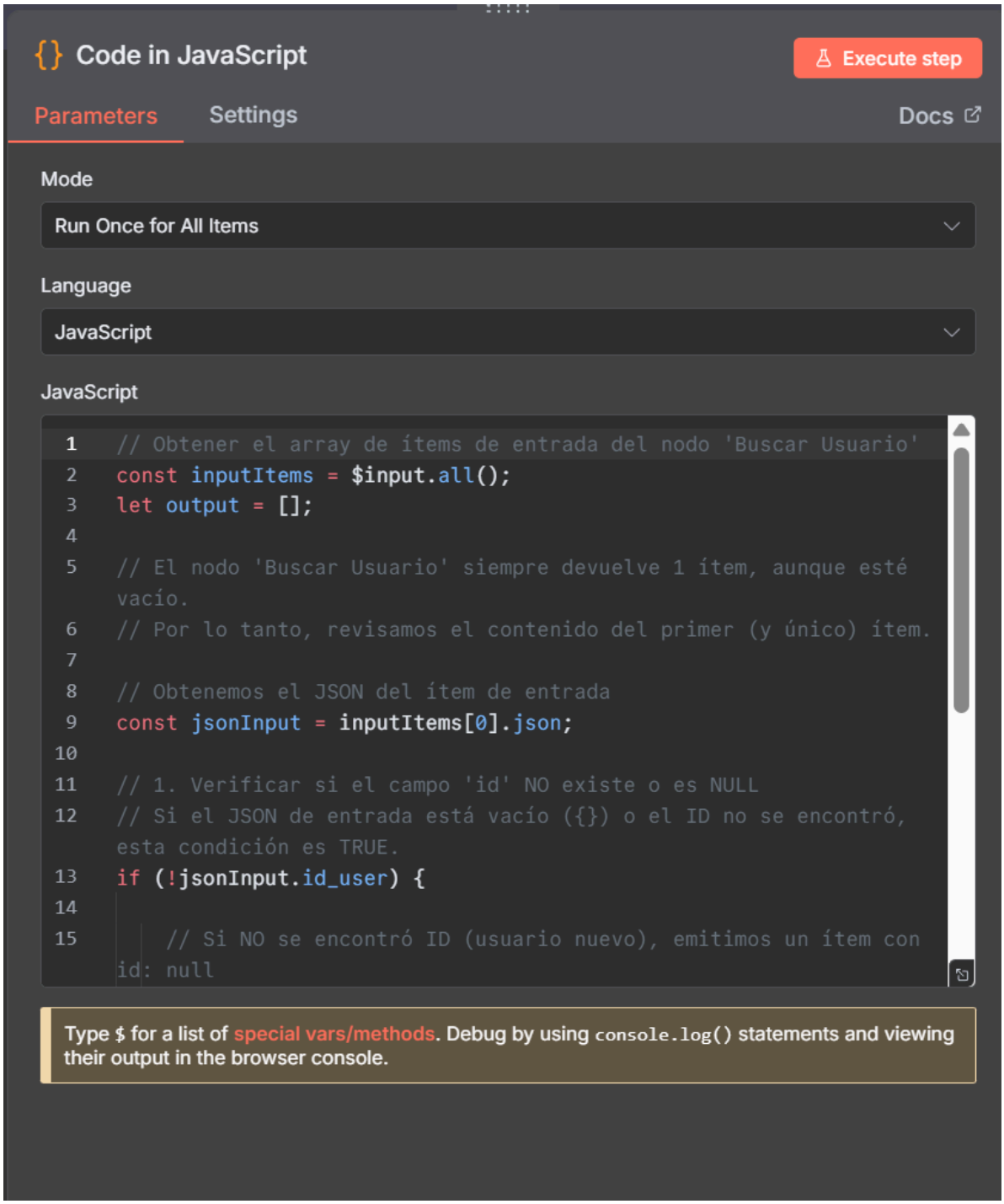
Fase B: Identificación y Lógica de Bifurcación

#	Nodo	Propósito	Configuración Crítica
4	**Buscar Usuario**	Comprobar si el usuario ya existe en la tabla `users`.	**Query:** `SELECT id_user FROM users WHERE telegram_id = {{$json.telegram_id}};`
5	**Code in JavaScript**	**Garantizar la salida.** Obliga al flujo a emitir un ítem con el `id_user` real o **`{id_user: null}`** si el usuario no existe.	**Script:** Usar la lógica de `if (!jsonInput.id_user)` para retornar `{id_user: null}` o `{id_user: [ID_REAL]}`.
6	**IF**	Decidir si el usuario es nuevo o existente.	**Condition:** `={{ $json.id_user }}` **is empty**. (Si es `null`, es TRUE, usuario nuevo).

7	**Crear usuario** (Rama TRUE)	Inserta un nuevo usuario y obtiene su ID interno.	**Query:** `INSERT INTO users (telegram_id, name) VALUES ({{ $('Mapeo de datos').first().json.telegram_id }}, '{{ $('Mapeo de datos').first().json.nombre_usuario }}') RETURNING id_user;`

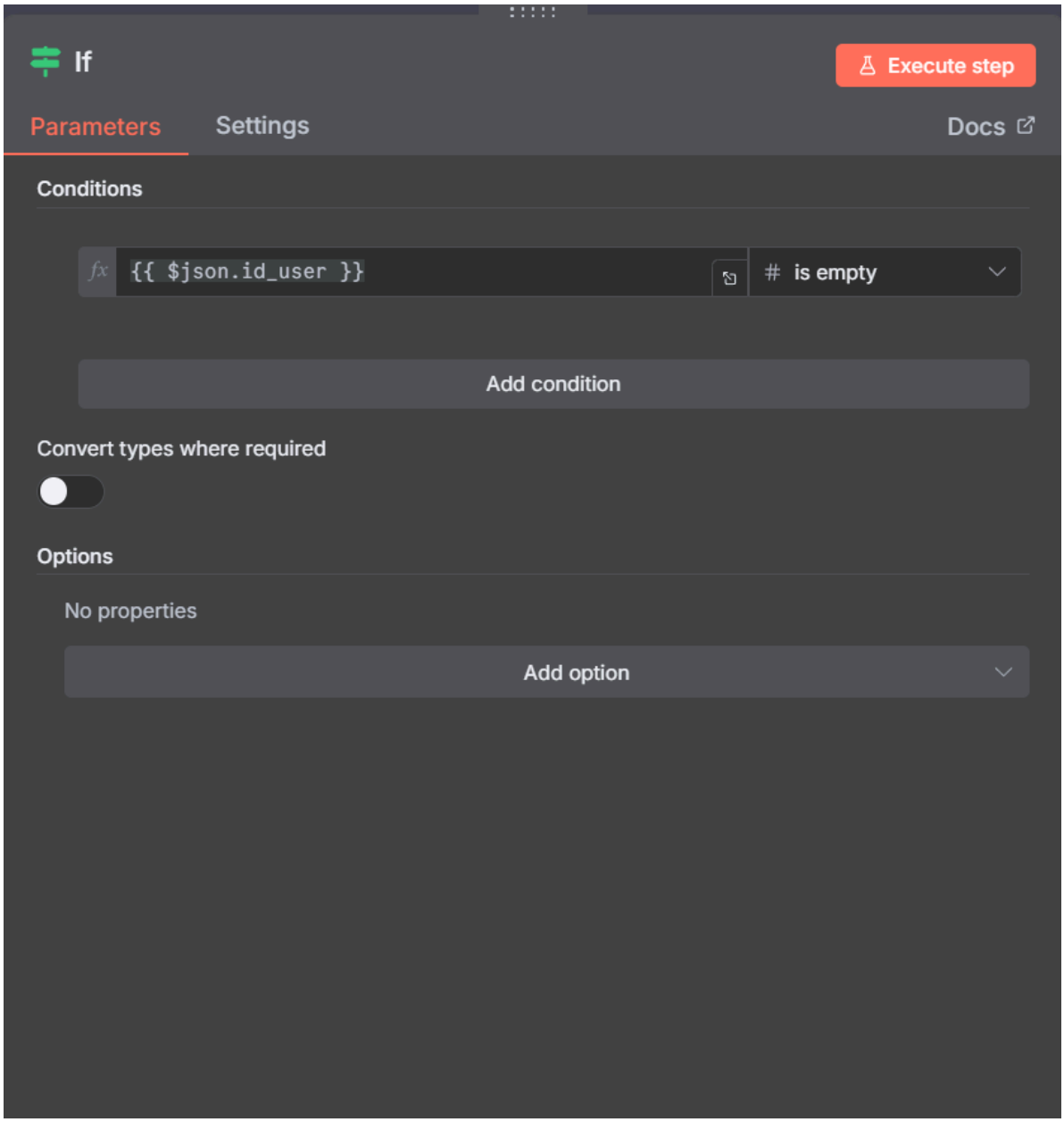
Si el usuario existe la consulta devuelve el ID de la base de datos al que esta asociado el “telegram_id”, de lo contrario el nodo “Code” devuelve “null”

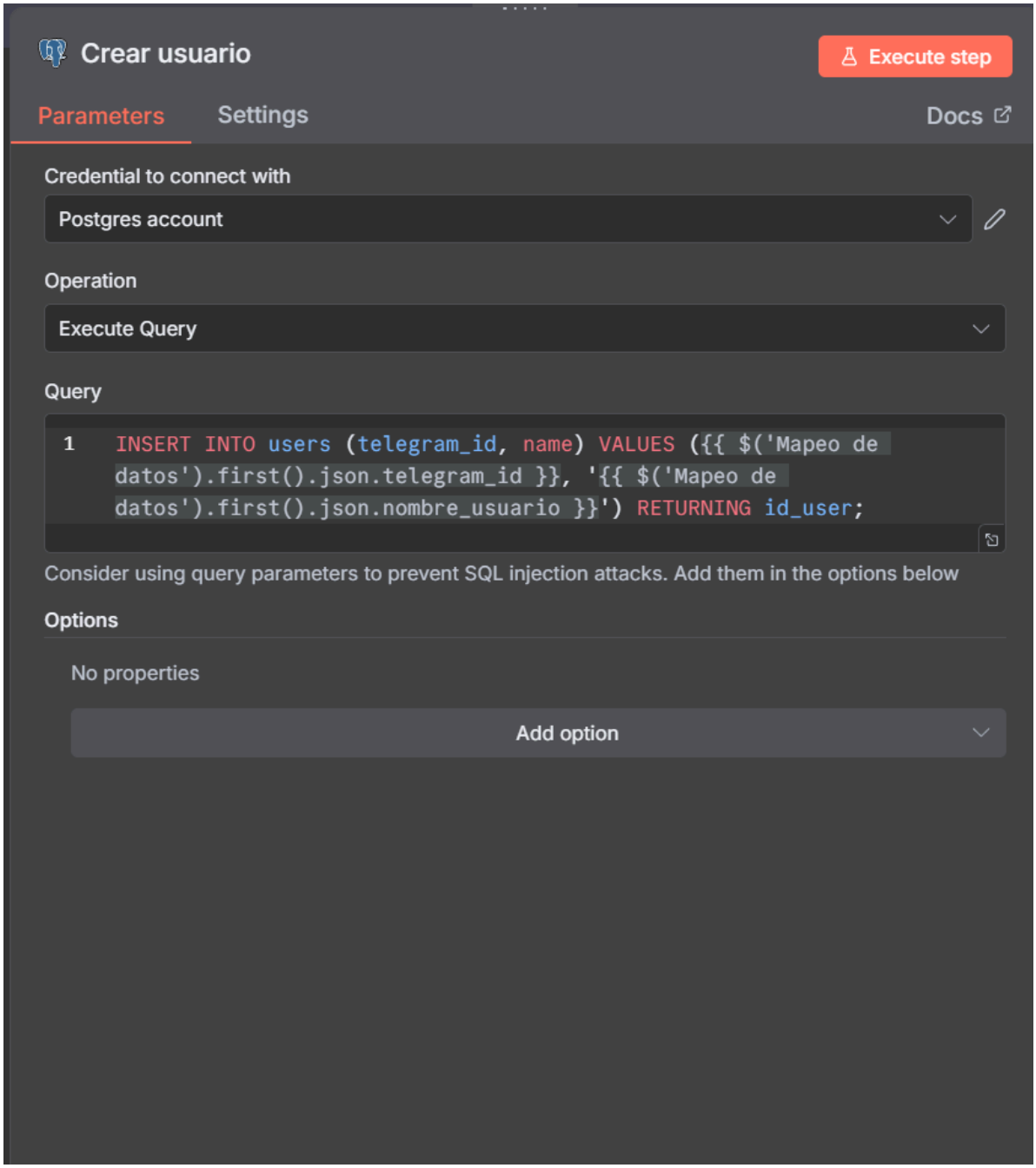



Code in JavaScript
Execute step
Parameters
Settings
Docs
Mode
Run Once for All Items
Language
JavaScript
JavaScript
1 // Obtener el array de ítems de entrada del nodo 'Buscar Usuario'
2 const inputItems = $input.all();
3 let output = [];
4
5 // El nodo 'Buscar Usuario' siempre devuelve 1 ítem, aunque esté vacío.
6 // Por lo tanto, revisamos el contenido del primer (y único) ítem.
7
8 // Obtenemos el JSON del ítem de entrada
9 const jsonInput = inputItems[0].json;
10
11 // 1. Verificar si el campo 'id' NO existe o es NULL
12 // Si el JSON de entrada está vacío ({}) o el ID no se encontró, esta condición es TRUE.
13 if (!jsonInput.id_user) {
14
15 // Si NO se encontró ID (usuario nuevo), emitimos un ítem con id: null
Type $ for a list of special vars/methods. Debug by using console.log() statements and viewing their output in the browser console.

Edit JavaScript

```javascript
// Obtener el array de ítems de entrada del nodo 'Buscar Usuario'
const inputItems = $input.all();
let output = [];

// El nodo 'Buscar Usuario' siempre devuelve 1 ítem, aunque esté vacío.
// Por lo tanto, revisamos el contenido del primer (y único) ítem.

// Obtenemos el JSON del ítem de entrada
const jsonInput = inputItems[0].json;

// 1. Verificar si el campo 'id' NO existe o es NULL
// Si el JSON de entrada está vacío ({}) o el ID no se encontró, esta condición es TRUE.
if (!jsonInput.id_user) {

    // Si NO se encontró ID (usuario nuevo), emitimos un ítem con id: null
    output.push({
        json: {
            id_user: null
        }
    });
} else {
    // 2. Si el campo 'id' SÍ existe y tiene valor (usuario encontrado)
    // Devolvemos el JSON de entrada sin cambios.
    output.push({
        json: jsonInput
    });
}

// Devolvemos el array de salida (1 ítem garantizado).
return output;
```


If
Execute step
Parameters
Settings
Docs
Conditions
fx
{{ $json.id_user }}
is empty
Add condition
Convert types where required
Options
No properties
Add option

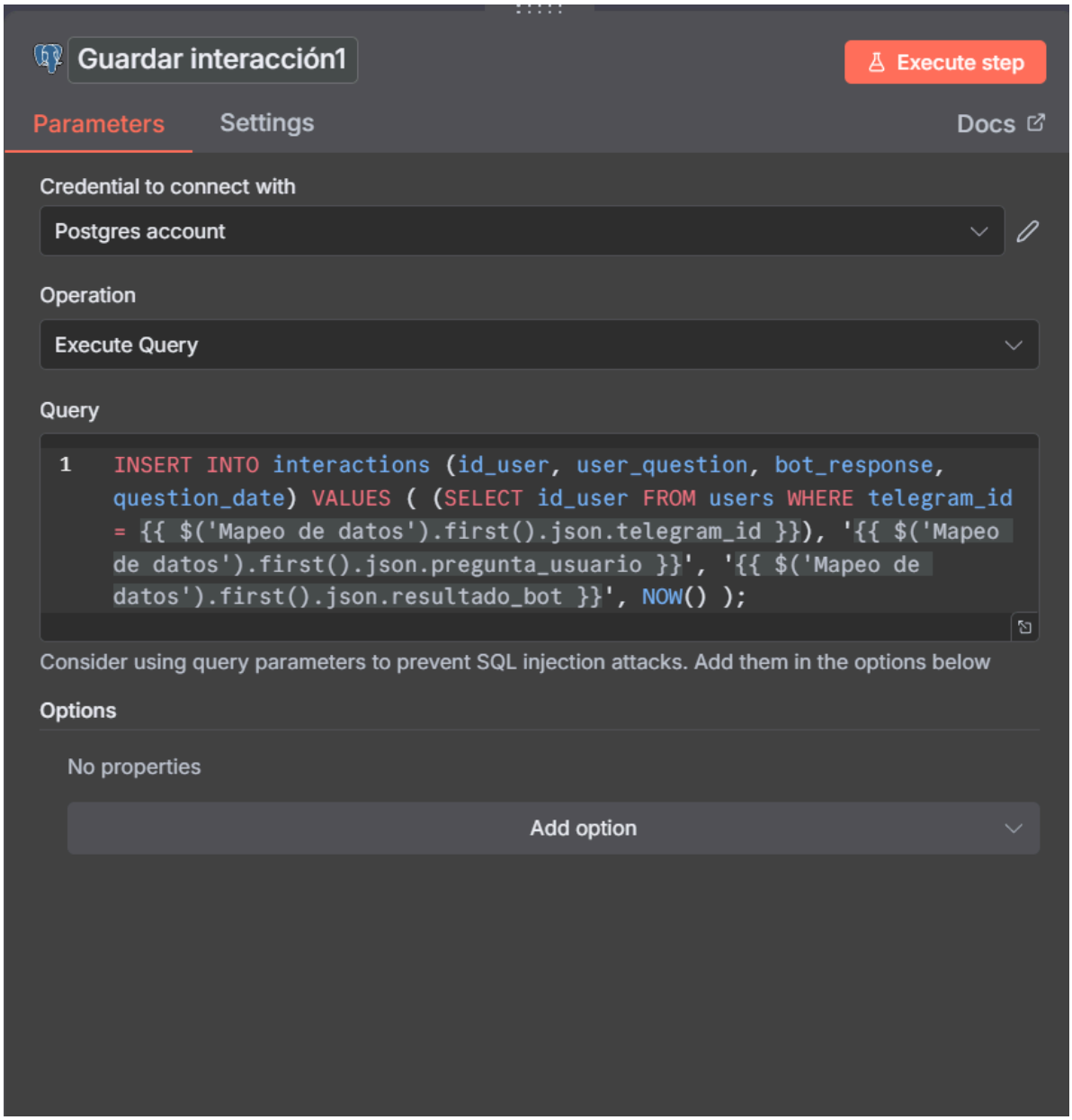


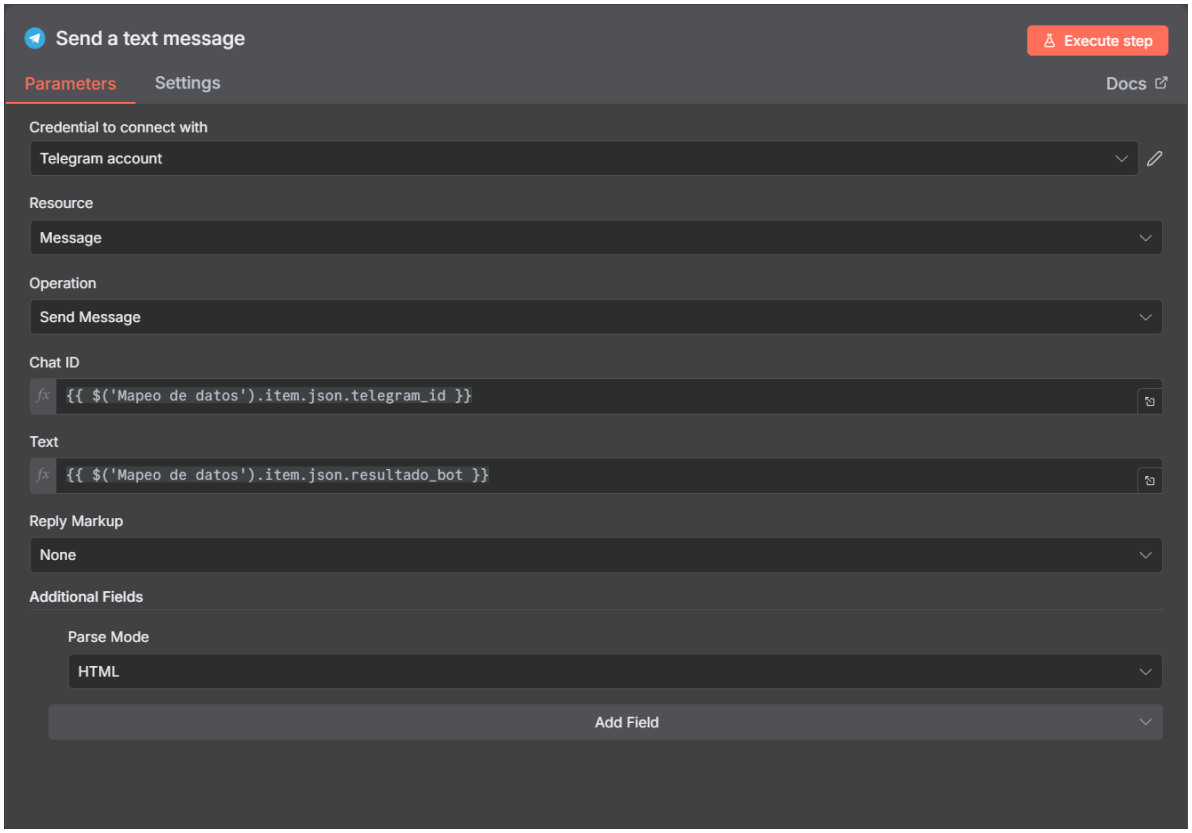
Fase C: Auditoría Final y Respuesta (El Proceso Central)

Independientemente de la rama que se tome (usuario nuevo o existente), el flujo converge en la auditoría y la respuesta.

#	Nodo	Propósito	Conexiones Críticas
8	**Guardar interacción1**	**Auditoría de la Rama TRUE.**	**Query:** `INSERT INTO interactions (id_user, user_question, bot_response,`

	(Después de Crear usuario)	Guarda la pregunta y la respuesta.	`question_date) VALUES ( (SELECT id_user FROM users WHERE telegram_id = {{ $('Mapeo de datos').first().json.telegram_id }}), '{{ $('Mapeo de datos').first().json.pregunta_usuario }}', '{{ $('Mapeo de datos').first().json.resultado_bot }}', NOW() );`
9	**Guardar Interacción** (Rama FALSE)	**Auditoría de la Rama FALSE.** Mismo *query* que el anterior.	Ambas ramas (8 y 9) se conectan al nodo final.
10	**Send a text message**	Enviar la respuesta final de la IA al usuario.	**Chat ID:** `{{ $('Mapeo de datos').item.json.telegram_id }}` . **Text:** `{{ $('Mapeo de datos').item.json.resultado_bot }}` .


Guardar interacción1
Execute step
Parameters
Settings
Docs
Credential to connect with
Postgres account
Operation
Execute Query
Query
1 INSERT INTO interactions (id_user, user_question, bot_response, question_date) VALUES ((SELECT id_user FROM users WHERE telegram_id = {{ $('Mapeo de datos').first().json.telegram_id }}), '{{ $('Mapeo de datos').first().json.pregunta_usuario }}', '{{ $('Mapeo de datos').first().json.resultado_bot }}', NOW());
Consider using query parameters to prevent SQL injection attacks. Add them in the options below
Options
No properties
Add option


Send a text message
Execute step
Parameters
Settings
Docs
Credential to connect with
Telegram account
Resource
Message
Operation
Send Message
Chat ID
{{ $('Mapeo de datos').item.json.telegram_id }}
Text
{{ $('Mapeo de datos').item.json.resultado_bot }}
Reply Markup
None
Additional Fields
Parse Mode
HTML
Add Field