# **Projeto 02) Agente Financeiro com SQL**

## **Arquivo:** [**https://www.dropbox.com/scl/fi/olokusxw6xcf2ewoxkjav/Agente-Financeiro-com-SQL.json?rlkey=6pcyxvh45x2mrxqzuedt1fv0j&st=ciugwh2m&dl=0**](https://www.dropbox.com/scl/fi/olokusxw6xcf2ewoxkjav/Agente-Financeiro-com-SQL.json?rlkey=6pcyxvh45x2mrxqzuedt1fv0j&st=ciugwh2m&dl=0)

## **Objetivo**

Neste guia, vamos criar um agente de IA capaz de responder perguntas em linguagem natural com base em um banco de dados SQL, usando o n8n e um modelo da OpenAI. Ao final, você terá um workflow funcional que:

* Integra com um banco PostgreSQL para consultar dados financeiros
* Utiliza subworkflows para executar queries de forma segura e modular
* Constrói respostas com base em perguntas como “Quanto gastei com transporte?”
* Aprende a estruturar queries dinâmicas e seguras, mesmo com filtros por data, categoria ou projeto

**Aquecimento:**

### Mensagem de registro:

Gastei R$752 com alimentação para o projeto RedRex.

Tive um gasto de R$1.680 em transporte do projeto ZapFlow.

Foram R$1.245 com software para o Jornada.

R$980 gastos com alimentação relacionados ao projeto ZapFlow.

Paguei R$1.102 em transporte para o Jornada.

Investi R$1.925 em software na RedRex.

R$870 de alimentação no projeto Jornada.

Gastei R$1.315 em transporte no RedRex.

Comprei licenças de software no valor de R$1.745 para o ZapFlow.

R$588 de alimentação usados na RedRex.

### Mensagem de consulta:

Quanto eu gastei com alimentação?

Qual foi o total de gasto do projeto RedRex?

Qual o total de despesas com software no ZapFlow?

### **SELECT sum(valor)**

### **FROM public.controle\_financeiro**

### **WHERE categoria ilike 'Refeição';**

### **SELECT sum(valor)**

### **FROM public.controle\_financeiro**

### **WHERE projeto ilike 'RedRex';**

### **SELECT sum(valor)**

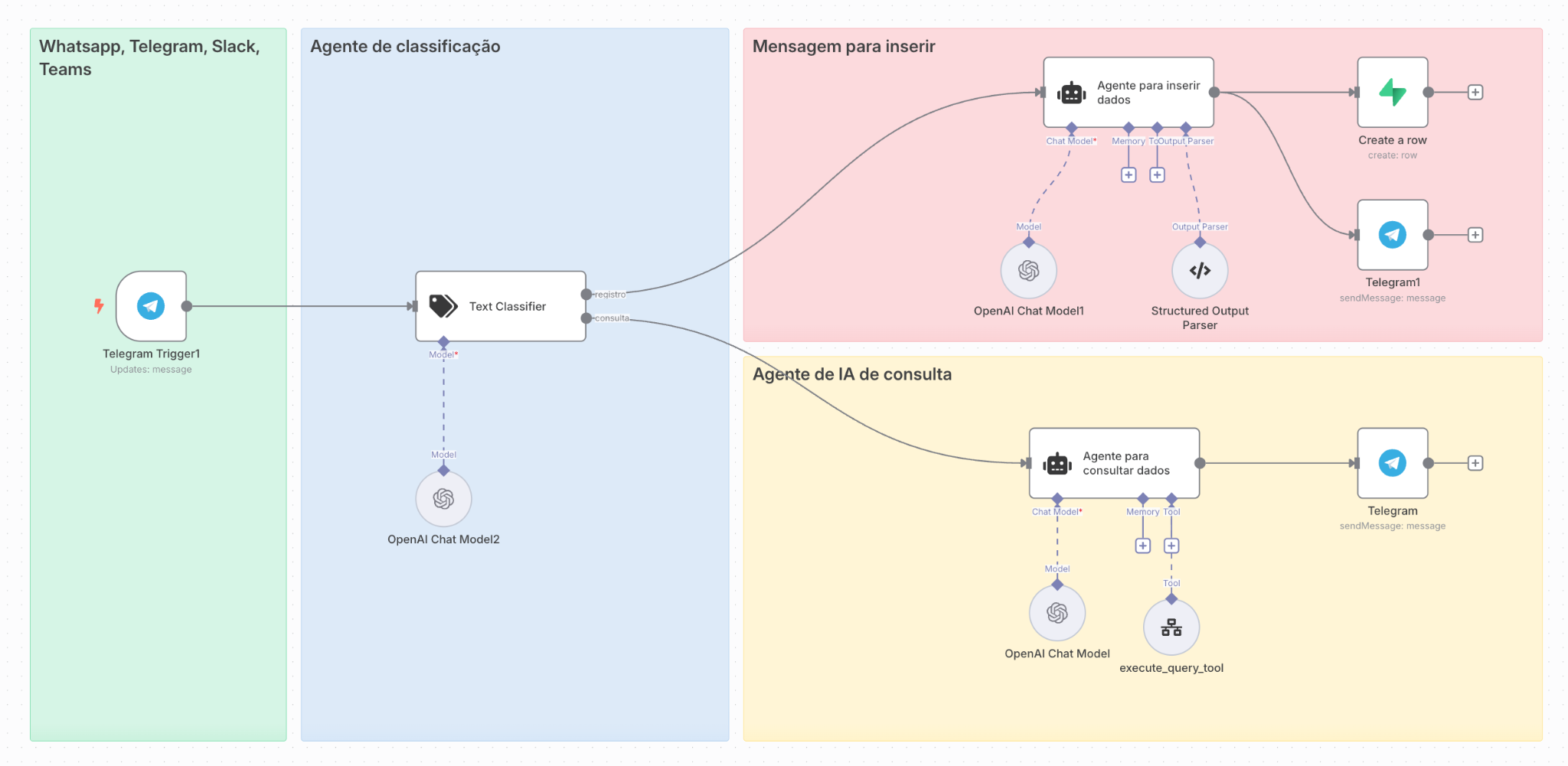
### **FROM public.controle\_financeiro**

### **WHERE projeto ilike 'Zapflow' and categoria ilike 'Software';**

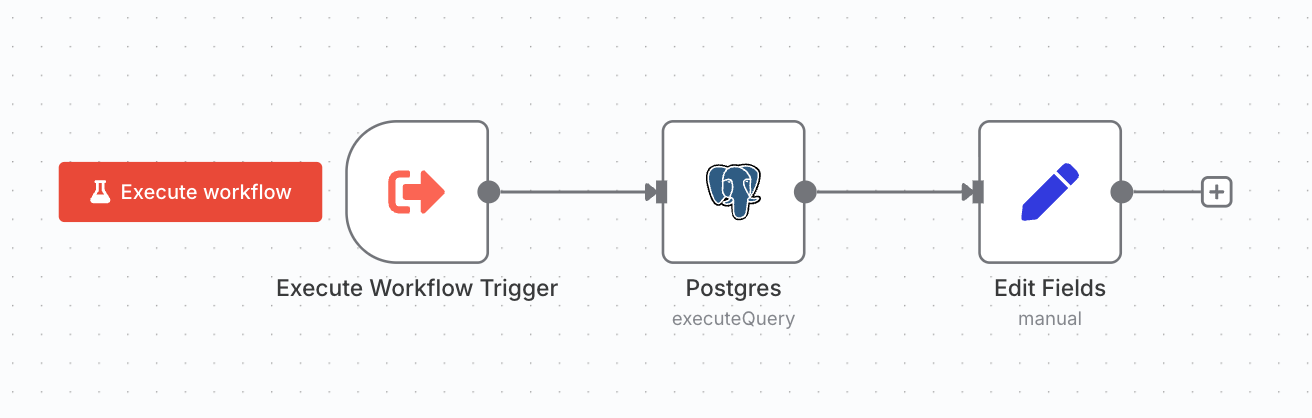
### 

### **Projeto**

Workflow principal.



Workflow para executar query.



## 

## 

## Agente para Inserir dados

## **Prompt:**

A seguir, você receberá uma mensagem que descreve uma despesa feita em um projeto. Extraia os seguintes campos:

- Data (a data em que o gasto foi feito, no formato YYYY-MM-DD (ex: "2025-07-07"). Se não for especificada, use a data de hoje: {{ $now }}.

- Descrição resumida do que foi gasto

- Valor numérico (em reais)

- Categoria (ex: Transporte, Refeição, Software, Escritório, Serviço, Cloud, Outro)

- Projeto relacionado

- Forma de Pagamento (ex: Cartão, Pix, Dinheiro, Boleto, Transferência, Não informado)

- Resumo da operação que foi feita, para ser enviado via mensagem de confirmação por um assistente. A mensagem deve confirmar a operação de forma simpática e objetiva, como se fosse enviada por um atendente no WhatsApp.

Mensagem:

{{ $json.message.text }}

Data de hoje:

{{ $now }}

**System Message**

Você é um assistente financeiro. Seu papel é interpretar mensagens de texto sobre despesas e extrair os dados de forma estruturada, com o objetivo de preencher automaticamente uma planilha de controle de gastos.

Além disso, você deve gerar uma frase de confirmação no campo "resumo", em tom simpático e direto, como se fosse um atendente humano respondendo no WhatsApp. O texto deve confirmar que a despesa foi registrada com sucesso, citando os principais dados como valor, descrição, projeto e forma de pagamento. Escreva a frase em uma única linha, sem quebras.

**Structured Output Parser**

{

"data": "2025-07-07",

"descricao": "Hospedagem para evento",

"valor": 300.00,

"categoria": "Serviço",

"projeto": "Jornada",

"forma\_pagamento": "Pix",

"source": "whatsapp",

"resumo": "Despesa registrada: R$300,00 em 'Hospedagem para evento' no projeto Jornada, pago via Pix."

}

## **Agente de classificação**

## 

**Text to classify**

Você vai receber uma mensagem de um usuário.

Classifique a intenção da mensagem com base nas opções abaixo:

- "registro": se o usuário estiver informando uma nova despesa

- "consulta": se estiver pedindo para ver, consultar ou buscar informações financeiras anteriores

Responda apenas com a palavra: "registro" ou "consulta".

Mensagem:

{{ $json.message.text }}

**Categories**

registro

se o usuário estiver informando uma nova despesa, gasto, compra ou pagamento realizado.

consulta

se o usuário estiver pedindo para consultar, saber, revisar ou buscar informações sobre despesas passadas.

## **O que é um subworkflow no n8n?**

Um **subworkflow** é simplesmente um *workflow auxiliar* criado para ser **reutilizado** dentro de outros workflows — especialmente por agentes de IA, bots ou workflows principais.

Em vez de colocar toda a lógica num só lugar, você isola partes como:

* Executar uma query no banco de dados
* Enviar uma mensagem por WhatsApp
* Gravar algo em uma planilha

Isso deixa tudo **modular, testável e reutilizável**.

## **Como funciona na prática?**

1. Você **cria um workflow separado** com o node Execute Workflow Trigger
2. Outro workflow (ou agente) chama ele via o nó Tool Workflow
3. O subworkflow **recebe dados,**, executa uma ação e **retorna uma resposta**

**Campos principais no seu caso**

### **Campo: Field to Return**

* Aqui você define **qual variável será enviada de volta** ao workflow principal
* No seu caso: response  
   👉 Isso corresponde ao valor definido no nó Set (ex: response = {{$json}})

### 

### **Campo: Input Schema**

Define **o que o subworkflow espera receber como entrada**.  
 No seu caso:

{

"type": "object",

"properties": {

"sql": {

"type": "string",

"description": "A SQL query based on the users question and database schema."

}

}

}

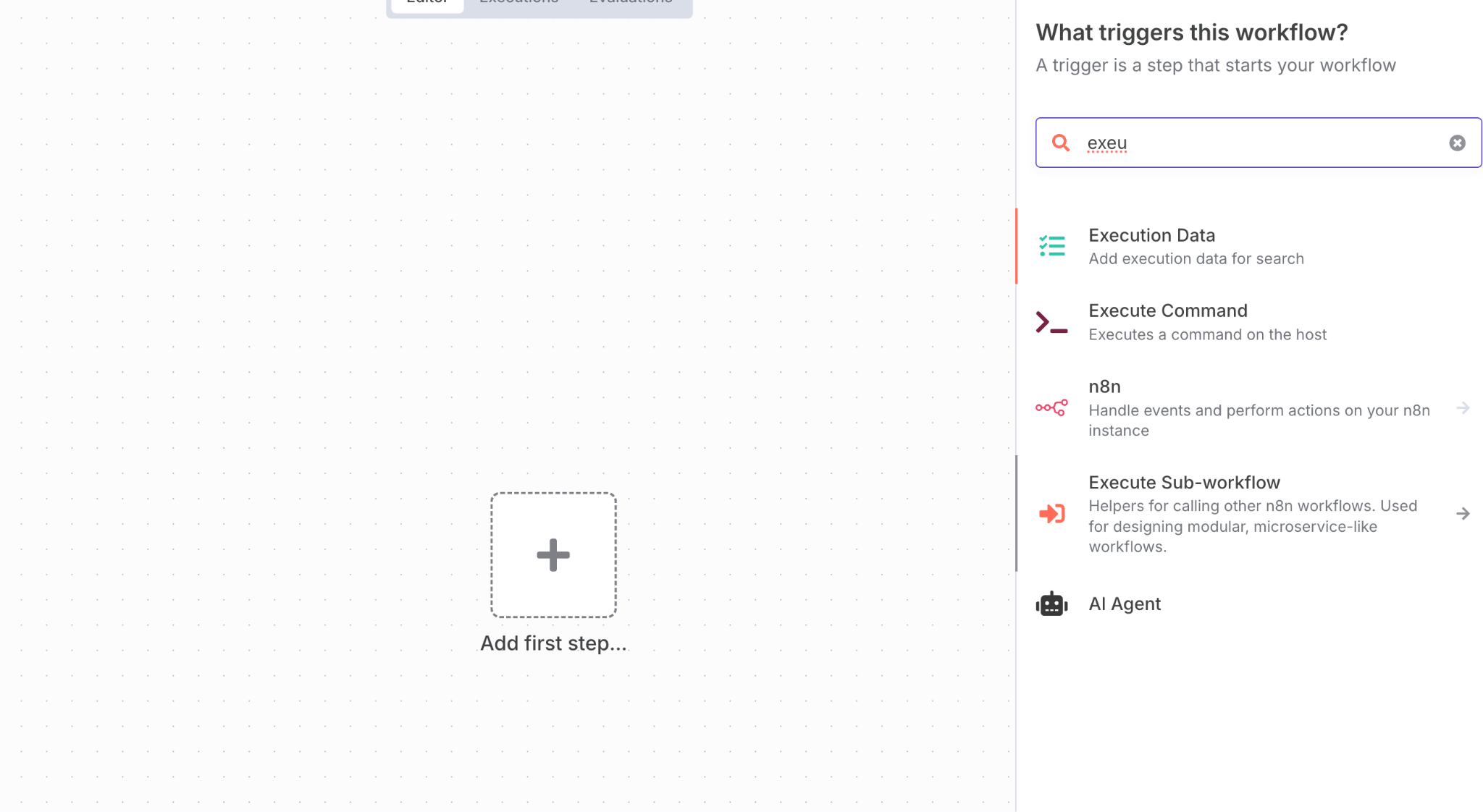
E esse valor será usado dentro do subworkflow para alimentar o nó do Postgres.

**Em resumo: estrutura ideal do seu subworkflow**

| **Etapa** | **Descrição** |
| --- | --- |
| Execute Workflow Trigger | Permite que outros workflows chamem esse workflow |
| Postgres | Executa a query vinda do input JSON ({{$json.query.sql}}) |
| Set | Define a variável de resposta (ex: response) que será retornada |
| Input Schema | Define o campo sql como obrigatório |
| Field to Return | Define o nome da variável final, aqui response |

**Hello World**

## **Parte 1 – Subworkflow: hello\_world**



**Quando for chamado, o subworkflow retorna a mensagem "Hello, world!" ou "Olá, {{nome}}!".**

### **Passos:**

1. **Crie um novo workflow e nomeie como: hello\_world**
2. **Adicione o node: When Executed by Another Workflow**
   * **Ele é o ponto de entrada obrigatório para subworkflows.**
   * **Modo: Accept all data (pode deixar o padrão)**
3. **Adicione o nó: Set**
   * **Nome: Edit Fields**
   * **No campo “Assignments” clique em Add Value:**
     + **Name: greeting**
     + **Value: "Hello, world!"**
4. **Salve o workflow**
   * **⚙️ Vá em Settings e ative:**
     + **✅ Can be activated by other workflows**

## 

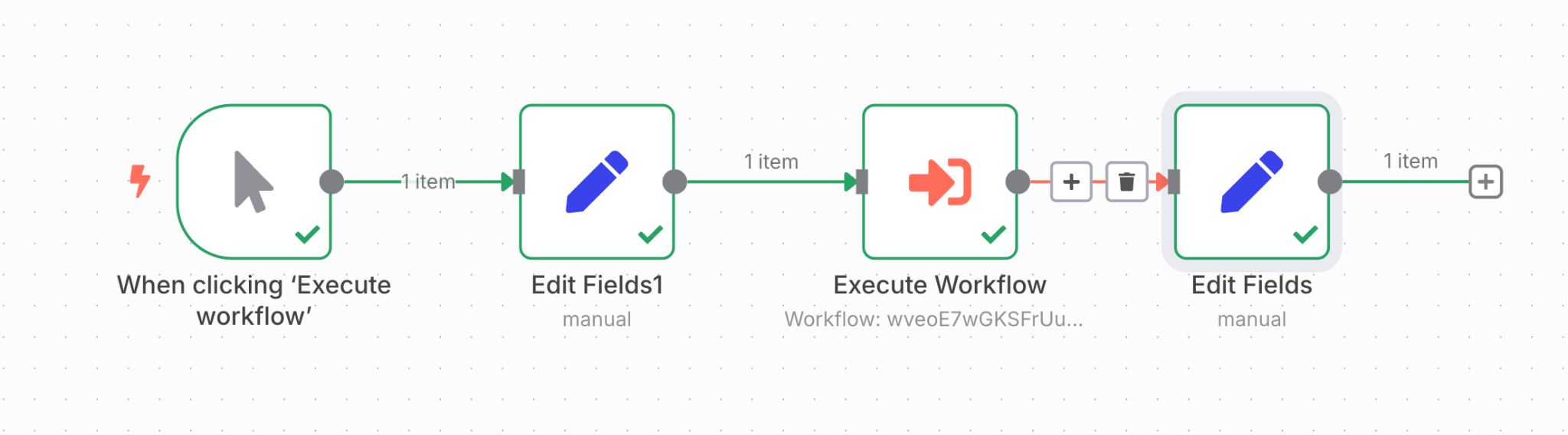
## **Parte 2 – Workflow principal**

### **Objetivo:**

* **Executar ao clique**
* **Definir o nome**
* **Chamar o hello\_world**
* **Mostrar o greeting que veio do subworkflow**

### **Passos:**

1. **Crie um novo workflow**
2. **Adicione o nó: Manual Trigger**
   * **Nome sugerido: When clicking 'Execute workflow'**
3. **Adicione o nó: Set**
   * **Nome: Edit Fields**
   * **Assignment:**
     + **Name: nome**
     + **Value: luciano**
4. **Adicione o nó: Execute Workflow**
   * **Workflow: hello\_world**
   * **Modo: Run once with all items**
   * **Ative a opção:**
     + **✅ Wait For Sub-Workflow Completion**
   * **Ele vai receber o resultado com o campo greeting**
5. **Adicione um último Set para exibir o retorno**
   * **Assignment:**
     + **Name: mensagem\_final**
     + **Value: {{ $json.greeting }}**

****

## **Prompt**

Mensagem do usuário:

{{ $json.message.text }}

A data de hoje é:

{{ $now }}

## **Novo System Message (com Tool Planilha)**

## Papel

Você é um Assistente Financeiro Pessoal que entende perguntas sobre gastos e finanças e responde com base na tabela `public.controle\_financeiro` em um banco PostgreSQL.

## Esquema da Tabela

A tabela `public.controle\_financeiro` contém os seguintes campos:

- `id` (int8)

- `data` (date)

- `descricao` (varchar)

- `valor` (float)

- `categoria` (varchar)

- `projeto` (varchar)

- `forma\_de\_pag` (varchar)

- `created\_at` (timestamp, default: now())

- `source` (varchar)

## Objetivo

Responder perguntas como:

- "Quanto eu gastei essa semana?"

- "Quais foram os gastos com alimentação em junho?"

- "Quanto foi gasto no projeto X em maio?"

## Boas práticas

- Use `data` para filtros de tempo, não `created\_at`

- Use SUM para totalizar gastos

## Exemplo

1. Quanto eu gastei com alimentação?

```

SELECT SUM(valor) AS total\_gasto

FROM public.controle\_financeiro

WHERE categoria ILIKE '%alimentação%' OR categoria ILIKE '%refeição%';

```

2. Quais foram os gastos do projeto RedRex?

```

SELECT descricao, categoria, valor, data

FROM public.controle\_financeiro

WHERE projeto ILIKE '%RedRex%';

```

3.Qual o total de despesas com software no ZapFlow?

```

SELECT SUM(valor) AS total

FROM public.controle\_financeiro

WHERE projeto ILIKE '%ZapFlow%' AND categoria ILIKE 'software';

```

## Como responder

1. Mostre o valor total (ou média, contagem, etc.)

2. Explique o filtro aplicado (ex: mês, categoria, etc.)

3. Seja direto e humano: "Você gastou R\$ X com alimentação em junho."

## 

## **Description**

## Call this tool to execute a query. Remember that it should be in a postgreSQL query structure.

## 

## {

## "type": "object",

## "properties": {

## "sql": {

## "type": "string",

## "description": "A SQL query based on the users question and database schema."

## }

## }

## }

## 

## 

## 

## 

## **Desafio 1: Criar um Agente de Estoque**

### **Objetivo:**

**Construir um agente que responde a perguntas sobre produtos e estoque, como:**

* **"Quantos itens do produto X eu tenho no estoque?"**
* **"Quais produtos estão com estoque abaixo de 10 unidades?"**

### **Instruções:**

1. **Criar uma tabela chamada estoque com os campos:**
   * **id (int)**
   * **produto (varchar)**
   * **quantidade (int)**
   * **categoria (varchar)**
   * **ultima\_atualizacao (date)**
2. **Criar um subworkflow com:**
   * **Execute Workflow Trigger**
   * **Postgres para executar a query**
   * **Set para retornar response**
3. **Criar um AI Agent que:**
   * **Gera SQL com base na pergunta**
   * **Chama o subworkflow consulta\_estoque**
   * **Retorna um texto explicando o resultado**

## 

## **Desafio 2: Criar um Agente de Tarefas Pessoais**

### **Objetivo:**

**Fazer um agente que responde a perguntas sobre tarefas, como:**

* **"O que eu tenho para fazer amanhã?"**
* **"Quantas tarefas estão pendentes esta semana?"**

### **Instruções:**

1. **Criar uma tabela chamada tarefas com os campos:**
   * **id (int)**
   * **descricao (varchar)**
   * **status (varchar: 'pendente', 'concluída', etc)**
   * **data\_entrega (date)**
   * **projeto (varchar)**
2. **Criar um subworkflow chamado consulta\_tarefas**
   * **Executa uma query SQL recebida do agente**
   * **Retorna a lista de tarefas ou o total**
3. **Criar um AI Agent:**
   * **Recebe perguntas em linguagem natural**
   * **Gera queries baseadas na tabela tarefas**
   * **Responde com uma lista ou resumo**