Relatório sobre a Atividade de Busca de Produtos Relacionados em N8N

Candidato: José David Melo Dos Santos

1. Introdução

A atividade proposta consistia em desenvolver uma automação utilizando a ferramenta n8n, com o objetivo de simular um cenário onde o usuário realiza a busca de um produto por meio de uma palavra-chave, e como resposta recebe uma lista de produtos relacionados à sua pesquisa. O fluxo criado no n8n integra diferentes nós para capturar a busca do usuário, realizar a consulta em uma fonte de dados e retornar as informações de forma estruturada.

O fluxo criado na ferramenta n8n inicia com um nó do tipo Webhook, que atua como o gatilho principal da automação, aguardando uma requisição externa para iniciar o processo. Essa requisição é enviada a partir de uma chamada feita pelo Postman, que se comunica com uma API intermediária. Essa API, por sua vez, encaminha a solicitação ao webhook do n8n, ativando o fluxo. Em seguida, utiliza-se um nó do tipo HTTP Request, responsável por realizar uma requisição HTTP para a URL que contém a base de dados com os produtos fornecidos. Após isso, um nó de Código é utilizado para processar a resposta da base de dados: nesse trecho, realiza-se a extração dos campos relevantes (como código e descrição de cada produto) e aplica-se um filtro para limitar a análise aos 200 primeiros itens da lista.

Posteriormente, um nó do tipo Set é adicionado para ajustar os parâmetros e organizar os dados que serão utilizados na etapa seguinte. Em seguida, um nó de Agente de IA é empregado, onde foi realizada uma engenharia de prompt para solicitar ao modelo de inteligência artificial que retorne, em formato JSON, apenas os produtos mais relevantes relacionados à busca textual fornecida pelo usuário. Por fim, o fluxo é encerrado com um nó de resposta do Webhook, que envia o retorno da automação para a API intermediária, a qual repassa a resposta final ao Postman, completando o ciclo da requisição. Esta automação visa demonstrar a aplicabilidade da ferramenta em situações práticas de consulta e filtragem de dados.

2. Objetivos

O objetivo desta automação foi realizar a filtragem inteligente de informações provenientes de uma página da web, utilizando o apoio da Inteligência Artificial para organizar e refinar os dados. A automação processa os dados coletados e retorna uma resposta em formato JSON, estruturada de forma clara e objetiva, contendo apenas as informações mais relevantes, como a descrição e o código dos produtos. Esse processo serve como um recurso facilitador para otimizar a busca por produtos em sistemas, tornando mais eficiente a recuperação de dados específicos a partir de grandes volumes de informação.

3. Ferramentas Utilizadas

- N8N - O n8n (n-eight-n) significa "nodemation", uma combinação de “node” e “automation”. Ele é uma ferramenta de código aberto que permite criar automações personalizadas por meio de nós (nodes) conectados em um fluxo lógico. Podemos fazer diversas tarefas com o n8n, como receber dados de um formulário (via webhook), fazer uma requisição HTTP para uma API, processar os dados com JavaScript, enviar e-mails, mensagens, armazenar dados, chamar modelos de IA e diversas outras, tudo dentro de um mesmo fluxo.

- VSCode - O Visual Studio Code (VsCode) é um editor de código-fonte gratuito desenvolvido pela Microsoft. Amplamente utilizado por desenvolvedores em diversas linguagens, ele ainda conta com várias extensões, como por exemplo, a extensão do Python (suporte à linguagem, linting, Jupyter Notebook, debug). O VSCode funciona em Windows, Linux e macOS e pode ser instalado facilmente pele site oficial: http://code.visualstudio

- Postman - Ferramenta muito utilizada por desenvolvedores para testar, consumir e documentar APIs. Ele permite que você envie requisições HTTP (como GET, POST, PUT, DELETE, etc.) para servidores e veja a resposta em uma interface amigável. Ele está disponível para Windows, Linux e macOS e pode ser baixado no site oficial: https://www.postman.com/downloads/. Também possui versão web.

4. Como executar

Para executar a aplicação é necessário acessar o n8n Cloud:

- Vá para: https://app.n8n.cloud
- Faça Login com sua conta

Logo após, vá para **“Create Workflow”**

Em seguida, no novo workflow em branco, clique nos três pontinhos (:) no topo da tela, selecione **“Import From File”** e use o arquivo de nome “atvd2_n8n (1)”.

Após carregado o fluxo, acesse o link: https://web.postman.co/workspace/My-Workspace~c2118c09-166d-4fc5-8bae-34db4e3a5de7/request/43694292-335aabf4-fcca-4a82-afb0-25af9f832f3b?action=share&source=copy-link&creator=43694292 Para ir direto ao Postman

Abra o Visual Studio Code e importe a pasta API_N8N e execute o arquivo app.py digitando no terminal: python app.py (É necessário ter o python instalado na máquina)

Logo após, o terminal vai mostrar algo como “Running on http://127.0.0.1:5000”
Copie o link que aparece e use no Postman para testar, como requisição POST, acrescentando **/api/reenviar**, no link, ficando algo como: http://127.0.0.1:5000/api/reenviar

Vá ao n8n novamente, clique em “**Execute Workflow**” e volte ao Postman, selecione abaixo a opção de “**raw**” e “**Text**” ao invés de JSON e escreva o nome do produto que quer procurar. Clique em “**Send**” e espere carregar a resposta.

5. Descrição do Workflow

O fluxo da solução foi composto pelos seguintes nós: Webhook, HTTP Request, Code, Edit Fields, AI Agent e Response Webhook. Pode-se observar o fluxo completo na figura 1. Cada nó foi explicado detalhadamente abaixo.

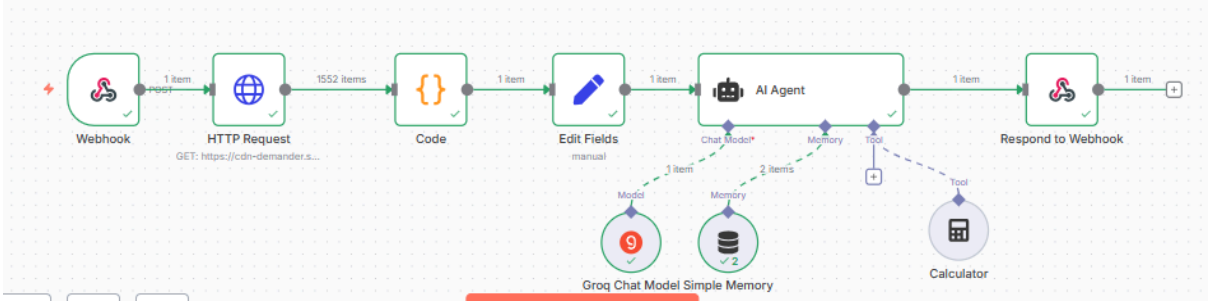

Figura 1 - Workflow da automatização

4.1 Webhook

O nó Webhook no n8n é utilizado para receber dados externos e disparar um fluxo de automação sempre que uma requisição HTTP é enviada a ele.

Neste projeto, o Webhook é acionado por uma requisição feita por uma API desenvolvida em Flask, que funciona como um gateway intermediário: ela recebe a chamada do Postman e, por sua vez, encaminha os dados para o Webhook do n8n. A implementação dessa API pode ser visualizada na Figura 2.

```python
from flask import Flask, request, jsonify
import requests

app = Flask(__name__)

URL_WEBHOOK = "https://dmelon8n.app.n8n.cloud/webhook-test/atvd2"

@app.route('/api/reenviar', methods=['POST'])

def reenviar_para_webhook():
    try:
        resposta = request.data.decode('utf-8')

        resposta = requests.post(URL_WEBHOOK, data=resposta, headers={"Content-Type": "text/plain"})

        return jsonify({
                "resposta" : resposta.json()
        }), resposta.status_code

    except Exception as e:
        return jsonify({"erro": str(e)}), 500

if __name__ == '__main__':
    app.run(debug=True)
```

Figura 2 - API para enviar requisição para o webhook

Podemos observar a rota criada na API utilizando o método POST, que recebe um texto enviado pelo Postman e o decodifica. Em seguida, a API realiza uma requisição para o Webhook do n8n, enviando o conteúdo recebido. Após o processamento do fluxo no n8n, a resposta é retornada para o Postman em formato JSON, contendo os dados filtrados.

As configurações para o webhook foram definidas como mostrado na figura 3 abaixo.

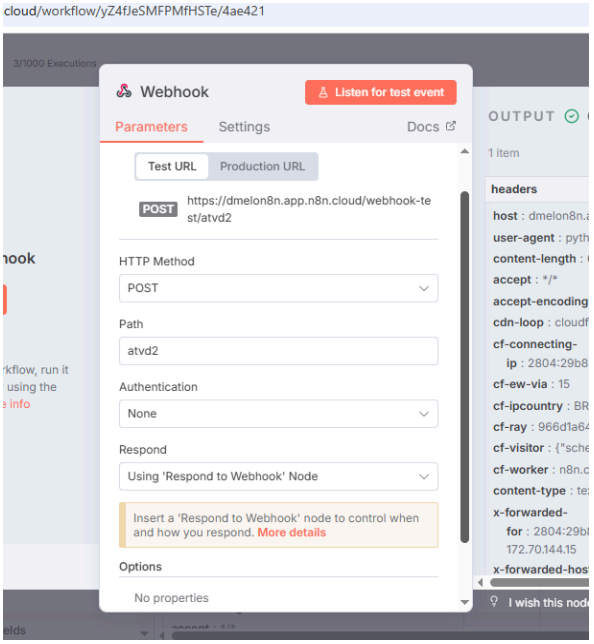

Figura 3 - Configurações do Webhook

Inicialmente, foi definido o método POST para o recebimento de dados no nó Webhook. O campo “Path” foi configurado como “atvd2”, com o objetivo de tornar o link de requisição mais legível e amigável. Já o campo “Respond” foi ajustado para “Using ‘Respond to Webhook’ Node”, permitindo que a resposta à requisição seja enviada por meio de um nó específico de resposta ao final do fluxo.

4.2 HTTP Request

O nó do HTTP Request serve para fazer requisições HTTP para APIs externas ou qualquer URL. Ou seja, ele permite que seu fluxo envie dados, busque informações ou interaja com serviços web externos. Para esta atividade, o nó de HTTP Request serviu para fazer uma requisição HTTP para o link da página na web com a base de produtos que foi entregue ao candidato, anteriormente. O retorno do nó de HTTP Request são os 1552 itens.

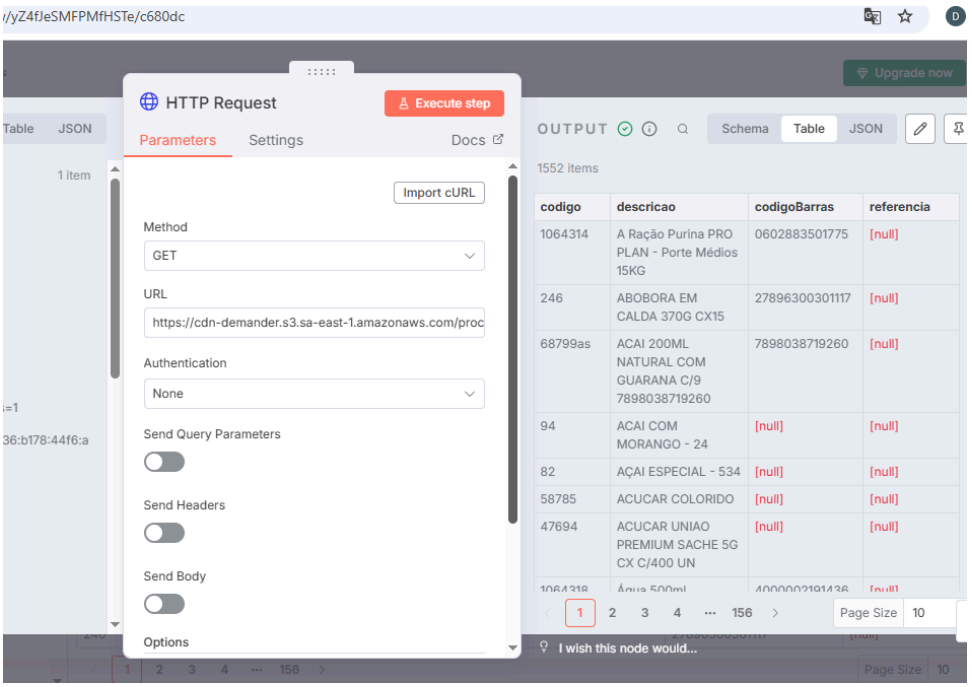

Figura 4 - Configurações do HTTP Request

Nas configurações do nó de HTTP Request foi definido o método de requisição como método de GET, pois queremos somente extrair o conteúdo da página da web e o campo de URL com o link da url da página com a base de produtos. Pode-se observar ao lado, a saída do nó com os itens extraídos em formato de tabela.

4.3 Code

O nó de Code no n8n serve para escrever código personalizado em JavaScript dentro do fluxo. Ele é bastante útil quando se precisa fazer algo mais específico que os outros nós não conseguem resolver sozinhos. Neste trabalho, o nó de Code serviu para integrar um código em JavaScript que extrai as informações de “Descrição” e “Código” de cada produto na lista para deixar a lista mais limpa. Além disso, o código foi configurado para retornar somente os 200 primeiros produtos, como forma de contornar a limitação de tokens no prompt ao usar a inteligência artificial, uma vez que o agente de IA utilizado é gratuito e possui restrições quanto ao volume de dados processados. O código utilizado pode ser visto na figura 5.

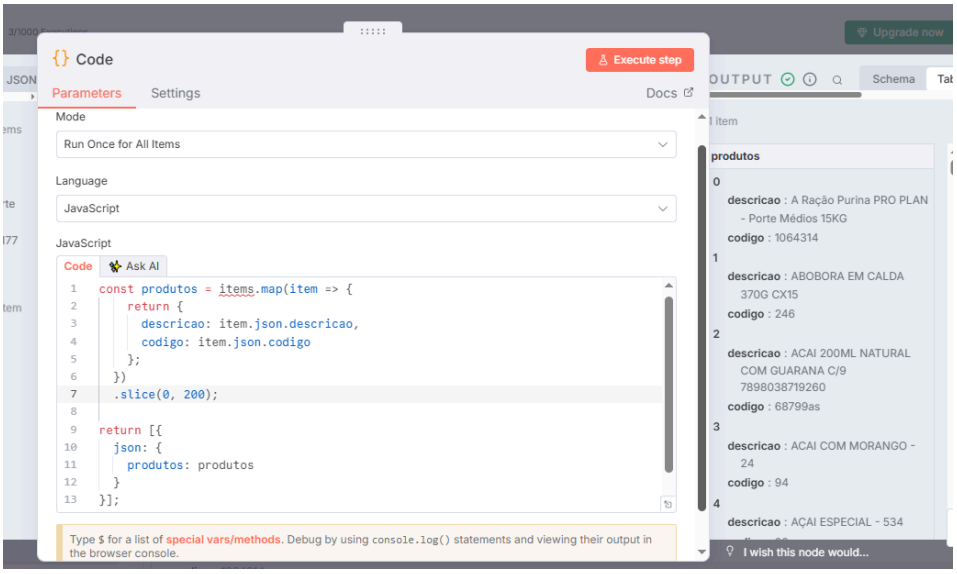

Figura 5 - Código para extrair informações da resposta do HTTP Request

Primeiramente, foi criada uma variável chamada "produtos", que utiliza o método **`map()`** para percorrer todos os itens da lista e retornar apenas os campos "Descrição" e "Código" de cada produto. Em seguida, foi aplicado o método **`slice()`** para extrair apenas os 200 primeiros produtos da lista original. O resultado final é uma lista mais enxuta, armazenada na variável "produtos", contendo somente as informações essenciais dos primeiros 200 itens.

4.4 Edit Fields

Já o nó de Edit Fields serve para alterar, remover ou reorganizar os campos dos dados JSON que estão passando pelo seu fluxo. Este nó é muito útil quando se quer limpar, renomear ou transformar a estrutura dos dados, sem precisar escrever código. Na figura 6 é possível analisar as ações feitas neste nó.

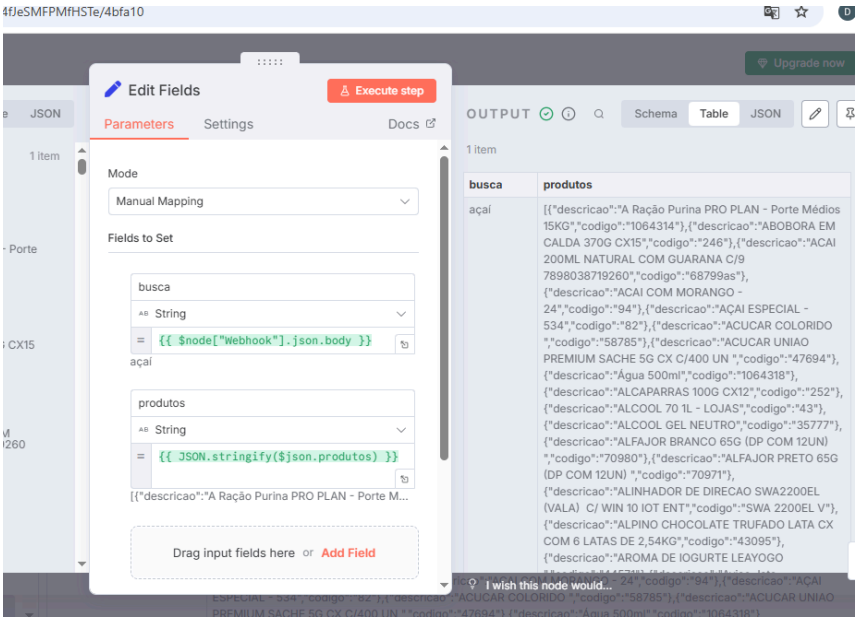

Figura 6 - Configurações Edit Fields

Neste nó, o campo de texto recebido pelo webhook foi renomeado para “busca”, facilitando sua identificação e uso nos próximos nós. Além disso, a lista de produtos gerada na saída do nó Code foi renomeada como “Produtos”. Com essa configuração, sempre que for necessário referenciar a lista de produtos ao longo do fluxo, basta utilizar o nome “Produtos” como identificador, o que torna a automação mais legível e organizada.

4.5 AI Agent

O nó de AI Agent é utilizado para interagir com modelos de inteligência artificial, como o ChatGPT, Cohere, Anthropic, entre outros, diretamente dentro do fluxo de automação. Para esta atividade, optou-se por utilizar o Groq Chat Model, que utiliza o modelo llama3-8b-8192, por ser gratuito e não necessitar de pagamentos para utilizar sua API.

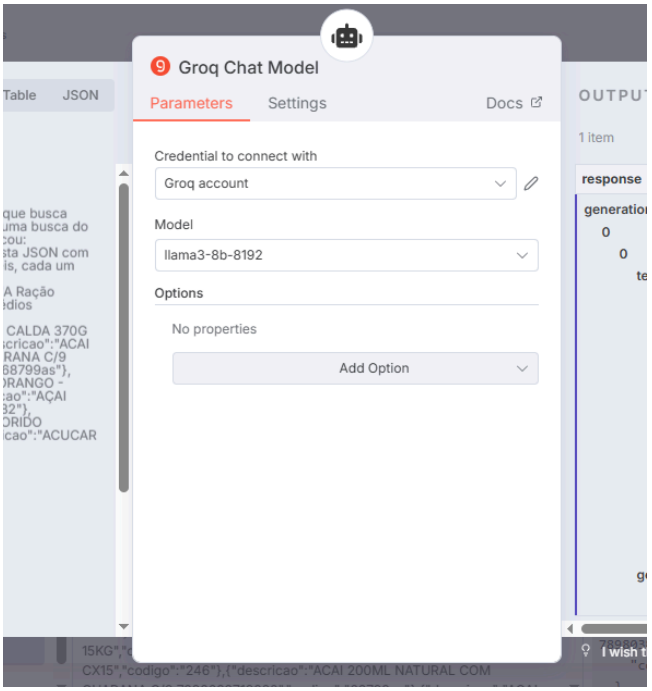

Figura 7 - Modelo de Agente de IA para Utilização

Além disso, foi necessário configurar cuidadosamente o prompt de comando utilizado pelo agente de Inteligência Artificial. Essa etapa é fundamental, pois o comportamento da IA depende diretamente das instruções definidas no prompt.

No caso desta atividade, o prompt foi elaborado com orientações claras para garantir que o modelo filtrasse apenas os produtos que realmente existem na lista fornecida, evitando que ele criasse, inferisse ou sugerisse produtos que não estão presentes nos dados originais.

Outro ponto importante incluído no prompt foi a forma como a IA deve realizar a busca, o modelo deve utilizar o campo de descrição do produto como base principal para comparar e identificar produtos que tenham relação com a entrada de texto fornecida pelo usuário. Essa abordagem garante respostas mais precisas, evitando desvios e mantendo a integridade dos dados utilizados no sistema.

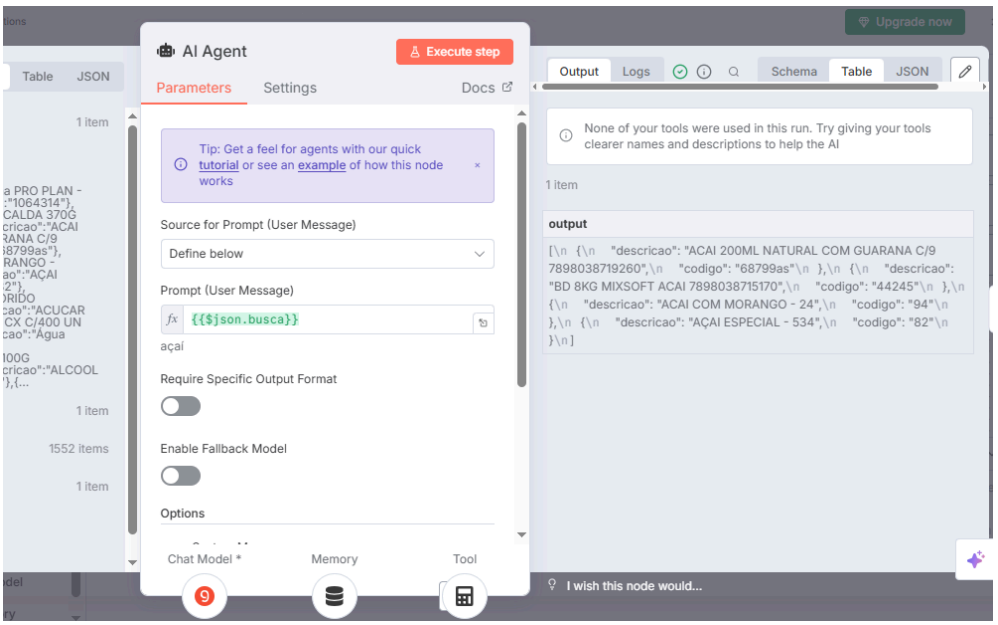

Figura 8 - Configurações de Prompt de Comando da IA

No prompt da IA temos que configurar dois campos, o prompt de usuário, que vai ser a entrada do usuário para o modelo e o prompt de sistema que é o comando que deve ser dado para o modelo, neste caso, o prompt de usuário é, o parâmetro de “busca”. Já o prompt de sistema pode ser visto na tabela abaixo.

Prompt de Sistema
Você é um sistema que busca produtos relacionados com uma busca do usuário. O usuário buscou: "{{ $node["Webhook"].json['body'] }}" Aqui está uma lista JSON com todos os produtos disponíveis, cada um com código e descrição: {{ $json.produtos }} Sua tarefa é: 1. Filtrar APENAS os produtos exatamente existentes na lista. 2. Procurar quais produtos da lista mais se relacionam com a busca do usuário, utilize a descrição de cada produto na lista como critério de busca. 3. Responder SOMENTE com uma lista JSON, contendo apenas os produtos encontrados que são relevantes para o texto do usuário. 4. NÃO CRIE OU INVENTE PRODUTOS. Se não houver nenhum produto correspondente, retorne uma lista vazia (`[]`). Exemplo de saída esperada: {   [

```
    {
      "descricao": "Moletom Cinza Básico",
      "codigo": "MTL-002"
    }
  ]
}
```

Tabela 1 - Prompt de Sistema do Modelo

4.6 Respond to Webhook

Por fim, este nó serve para enviar uma resposta personalizada de volta para quem fez uma requisição ao nó do webhook. Na figura 9 é possível ver sua configuração. A resposta foi configurada para ser em JSON.

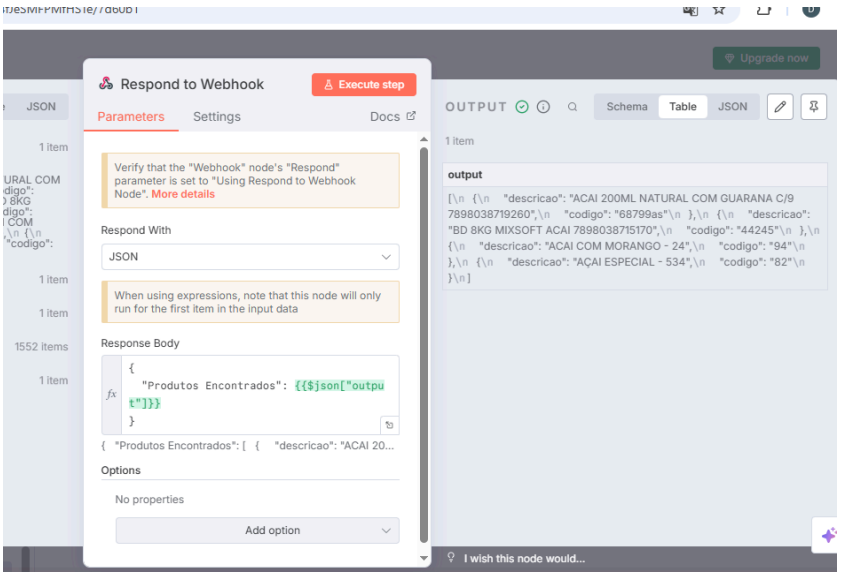

Figura 9 - Respond to Webhook

6. Resultados

A seguir, são apresentados os resultados obtidos a partir da execução do fluxo desenvolvido no n8n.

Figura 10 - Teste 1

No teste 1 foi possível observar que ao pesquisar pelo produto “moletom” o modelo não responde bem, retornando produtos que nada se relacionam com a palavra moletom

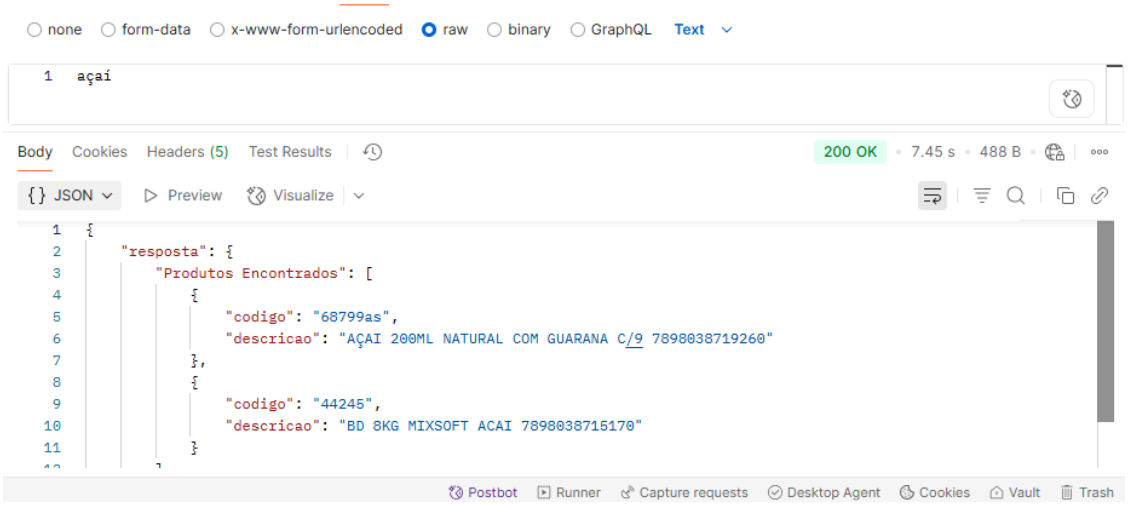

Figura 11 - Teste 2

Já no teste 2, pesquisando pelo produto “açaí” o modelo teve um rendimento melhor, retornando produtos que se relacionam com a palavra e retornando também o código do produto corretamente, porém, não retornou todos os produtos relacionados, talvez por conta da limitação no número de produtos da lista com os 200 produtos ou por uma falha no prompt de comando.

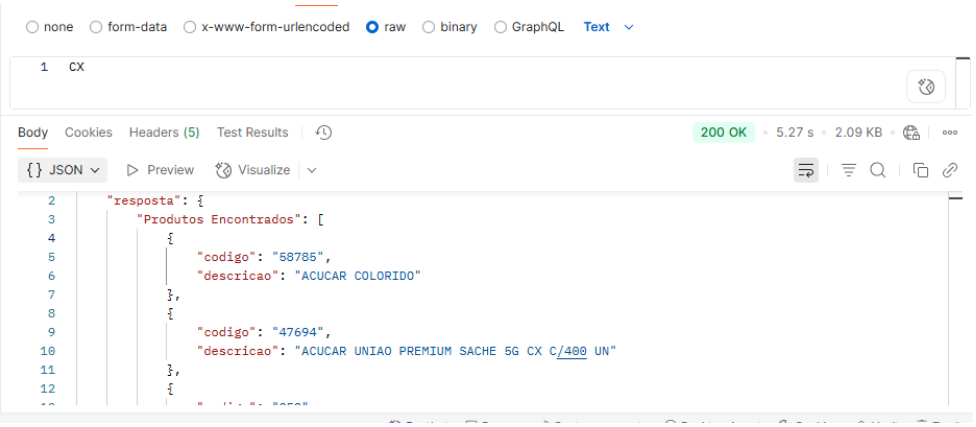

Figura 12 - Teste 3

No teste 3, ao pesquisar a palavra “CX” o modelo retorna alguns produtos que têm a palavra “CX” na descrição, porém, também erra muitos produtos, o que mostra que o modelo, aprendeu, porém com muitas falhas e limitações.

7. Conclusão

Dessa forma, pode-se concluir que o principal objetivo deste trabalho foi alcançado, ao demonstrar na prática a viabilidade e a utilidade da automatização por meio do n8n, quando integrada com Inteligência Artificial. Essa integração mostrou-se eficaz na tarefa de realizar buscas inteligentes em uma lista extensa de produtos, facilitando significativamente a identificação dos itens desejados a partir de descrições textuais. Em cenários onde a busca manual seria exaustiva e pouco prática, a solução desenvolvida se destacou como uma alternativa eficiente, precisa e escalável.

O modelo apresentou um desempenho razoável no processo de busca, demonstrando certo nível de aprendizado. No entanto, foram observadas algumas limitações. Em determinados testes, o modelo acabou gerando respostas com produtos inexistentes, o que contraria diretamente as instruções fornecidas no prompt de comando, que especificava claramente a necessidade de limitar os resultados apenas aos produtos presentes na lista. Esse comportamento pode indicar a necessidade de ajustes no prompt, com comandos mais restritivos ou mais bem elaborados, ou ainda a possibilidade de testar com outro modelo de Inteligência Artificial, que possa lidar melhor com essas restrições e, assim, minimizar os erros identificados.

Por fim, destaca-se que a integração entre o n8n e modelos de Inteligência Artificial se mostrou promissora para automatizar tarefas repetitivas e potencializar a busca por informações em grandes volumes de dados. Ainda que haja espaço para melhorias, como o aprimoramento do prompt e a escolha de modelos mais adequados à tarefa, os resultados obtidos reforçam o potencial dessas tecnologias no apoio a processos que antes exigiriam esforço manual e tempo considerável. Espera-se que, com ajustes futuros, a aplicação se torne ainda mais precisa e eficiente.