RELATÓRIO AGENTES AUTÔNOMOS – ANÁLISE DE CSV

*Grupo Synapse 7:*

*- Conrado Gornic*

*- Fernanda Tonetti*

*- Luiz Fernando Rezende*

*- Rodrigo Mibielli*

*- Saulo Brotto*

### 1. Contextualização

O projeto fornece um agente de inteligência artificial que responde às perguntas referentes aos dados de 100 notas fiscais selecionadas aleatoriamente do arquivo de notas fiscais do mês de janeiro/2024, disponibilizado pelo Tribunal de Contas da União.

Deve-se oferecer uma *interface* de *chat* ao usuário que através de perguntas relacionadas ao contexto das notas fiscais, um agente responda em língua portuguesa e de forma clara e direta sem haver a necessidade de conhecimento técnico por parte do usuário.

O presente trabalho a partir das seções subsequentes descreve de forma detalhada como o agente foi desenvolvido partindo da escolha da arquitetura e *frameworks* utilizados, passando pela estrutura da solução criada, como instalar e acessar o sistema e finaliza com algumas considerações finais.

#### 2. Arquitetura e *Framework* escolhidos

A idéia central seria criar uma solução *open-source* visand*o* *minimizar os custos* e tornar os dados mais seguros, além de utilizar ferramentas intuitivas e fáceis de serem atualizadas. Para atender a estes requisitos, baseamos todas as decisões de projeto com a instalação do sistema em *ambiente local* utilizando apenas componentes *open-source* e usando o *Docker* como ferramenta de apoio para instalação e execução do sistema.

O Docker é uma plataforma de código aberto que automatiza a implantação, execução e gerenciamento de aplicações em contêineres. Contêineres são unidades de *software* padronizadas que contêm tudo o que uma aplicação precisa para ser executada, incluindo código, bibliotecas e dependências. O Docker permite que você crie, implante e escale aplicações de forma rápida e consistente em qualquer ambiente.

Assim, decidimos que utilizaríamos também o N8N (https://n8n.io/) rodando localmente como orquestrador dos seguintes fluxos de execução de nosso agente:

* Fluxo de extração, transformação e carga (ETF) a partir de um arquivo comprimido contendo os dois arquivos CSVs da TCU e que alimentará a base de conhecimento de nosso agente. Esta base de conhecimento estará armazenada no banco de dados *PostgreSQL* que também é *open-source;*
* Fluxo principal responsável por rodar nosso agente. O usuário faz uma pergunta sobre o contexto dos dados das notas fiscais. O agente através de LLMs *open-source* e rodando localmente, a partir da pergunta do usuáriotraduz para código SQL, realiza a consulta ao banco de dados, traduz o resultado obtido para a língua portuguesa e retorna a resposta para o usuário.

Desta forma, optamos por utilizar o N8N rodando localmente (o N8N somente é pago se rodar na nuvem), com banco de dados PostgreSQL utilizado para armazenar os dados e o *Ollama* para rodar nossas LLMs também localmente.

Foram escolhidos dois modelos LLMs, ambos de código aberto:

* O *Codellama 7B* por sua conhecida eficiência para gerar códigos a partir de perguntas via *prompt*, entre eles o SQL;
* O *Mistral* responsável por traduzir os resultados obtidos a partir de *queries* ao banco de dados para a língua portuguesa de forma clara e direta.

A *interface* de comunicação entre o agente e o usuário é através de um *chat* disponibilizado pelo próprio N8N através de um endereço na internet*.*

A instalação destas ferramentas e modelos LLMs fica a cargo do Docker que através do *script docker-compose.yml* prepara todo o ambiente local para ser executado, além de importar automaticamente as ferramentas, os modelos das LLMs e os fluxos utilizados na N8N em formato JSON. Desta forma, como pré-requisito, devemos ter o Docker instalado em nossa máquina local.

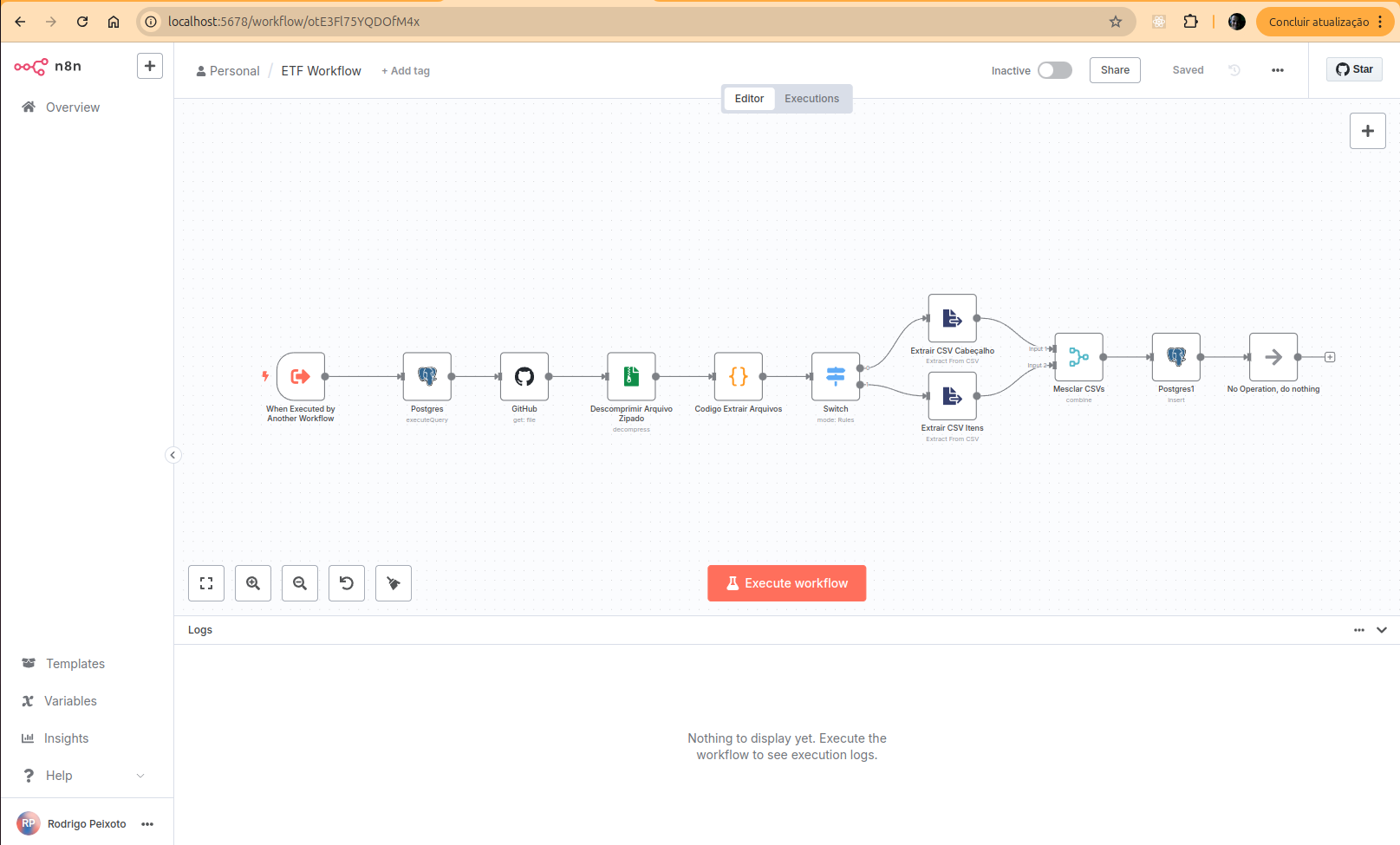
A execução do agente é facilitada através de um arquivo tipo *shell script* para máquinas Linux e *batch script* para máquinas ambientadas no Windows.

#### 3. Estrutura da Solução

Como dito na seção anterior, o projeto utiliza o N8N como espinha dorsal e utiliza dois fluxos que serão detalhados a seguir.

3.1 – *Workflow “ETF Workflow”*

Este *workflow* é invocado a partir do fluxoprincipal sempre após o usuário enviar a sua pergunta e que alimentará a base de conhecimento de nosso agente. Ele é responsável pela funcionalidade de ETF (*Extract, Transform and Load*) dos dados das notas fiscais que estão armazenados inicialmente em dois arquivos em formato CSV dentro de um arquivo zipado e que alimentarão o banco de dados Postgre SQL (<https://www.postgresql.org/>) (veja a Figura 3.1).

Figura 3.1: Subfluxo responsável por criar toda a estrutura e preenchimento da base de conhecimento.

A partir da figura acima, vemos que o *workflow* é disparado a partir da chamada de outro *workflow,* como uma *sub-rotina* (neste caso, o TCU Workflow o invoca).

Este fluxo sempre será disparado logo após o usuário enviar a sua pergunta no *chat* e antes do agente ser executado e por uma simples razão: para que o agente possa responder às perguntas do usuário ele precisa de uma base de conhecimento pré preenchida com os dados referentes às notas fiscais; caso o contrário, ele não saberá responder a nenhuma pergunta do usuário (pelo menos referentes às notas fiscais do problema).

A seguir o nó “Postgres” é responsável por criar ou recriar a tabela de nome “notas\_fiscais” que armazenará os dados dos CSVs extraídos e mesclados no banco de dados de *schema public* do banco de dados PostgreSQL. A declaração da tabela em linguagem SQL é a seguinte:

DROP TABLE IF EXISTS "notas\_fiscais";

CREATE TABLE IF NOT EXISTS "notas\_fiscais" (

"id" serial primary key,

"chave\_de\_acesso" TEXT NOT NULL,

"modelo" TEXT NOT NULL,

"serie" TEXT NOT NULL,

"numero" TEXT NOT NULL,

"natureza" TEXT NOT NULL,

"data\_emissao" TIMESTAMP WITHOUT TIME ZONE,

"evento\_mais\_recente" TEXT NOT NULL,

"timestamp\_mais\_recente" TIMESTAMP WITHOUT TIME ZONE,

"cpfcnpj\_emitente" TEXT NOT NULL,

"razao\_social\_emitente" TEXT NOT NULL,

"inscricao\_estadual\_emitente" TEXT NOT NULL,

"uf\_emitente" TEXT NOT NULL,

"municipio\_emitente" TEXT NOT NULL,

"cnpj\_destinatario" TEXT NOT NULL,

"nome\_destinatario" TEXT NOT NULL,

"uf\_destinatario" TEXT NOT NULL,

"indicador\_ie\_destinatario" TEXT NOT NULL,

"destino\_operacao" TEXT NOT NULL,

"consumidor\_final" TEXT NOT NULL,

"presenca\_comprador" TEXT NOT NULL,

"valor\_nota\_fiscal" DECIMAL(10,2) NOT NULL,

"numero\_produto" TEXT NOT NULL,

"descricao\_produto\_servico" TEXT NOT NULL,

"codigo\_ncm\_sh" TEXT NOT NULL,

"ncm\_sh\_tipo\_produto" TEXT,

"cfop" TEXT NOT NULL,

"quantidade" DECIMAL(10,2) NOT NULL,

"unidade" TEXT NOT NULL,

"valor\_unitario" DECIMAL(10,2) NOT NULL,

"valor\_total" DECIMAL(10,2) NOT NULL

);

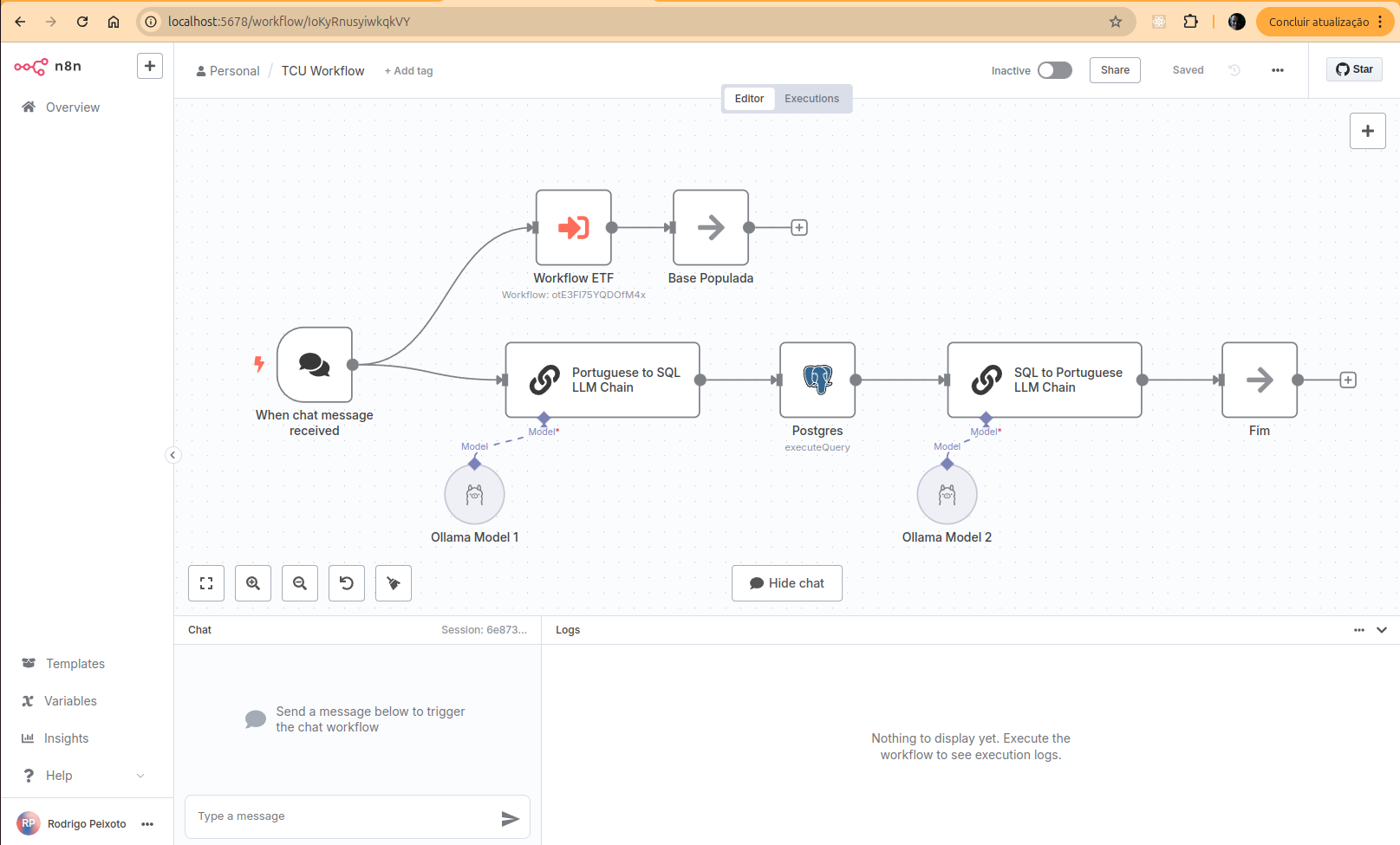
A seguir, o nó “Github” é o responsável por localizar o arquivo zipado de nome 202401\_Nfs.zip que está armazenado no próprio Github do projeto e que contém os dois arquivos CSVs.

Em seguida, através de outro nó, ele descomprime o arquivo zipado e através de um código escrito em *Javascript* no próximo nó, separa os dois conteúdos em memória. Depois a partir de um nó *switch* ele trata separadamente cada arquivo CSV em função de seu nome e finalmente através de um nó *merge*, mescla os dados de ambos através de uma chave única de “chave\_de\_acesso” inserindo por fim estes dados na tabela de nome “notas\_fiscais” no PostgreSQL.

Após popular a nossa base de dados, o processo de ETF é finalizado e a base de conhecimento do agente está alimentada com os dados oriundos dos arquivos CSVs.

3.2 – *Workflow “TCU Workflow”*

A idéia por trás é que, uma vez o banco de dados ou a base de conhecimento de nosso agente esteja preenchido com os dados das Notas Fiscais, seja possível executar consultas SQL a partir da tradução de uma pergunta de interesse do usuário via *prompt,* obter o resultado da consulta, traduzir novamente para língua portuguesa e exibir a resposta para o usuário. Este é o *core* da aplicação e o papel deste fluxo principal (Veja a figura 3.2).

Figura 3.2: O Workflow Principal responsável por responder às perguntas do usuário sobre as notas fiscais.

O evento que dá início ao fluxo é uma mensagem oriunda do *chat* de perguntas e respostas. Em seguida, o *Workflow ETF* é chamado afim de popular a base de conhecimento do agente explanado no fluxo acima.

Após popular os dados na tabela única “notas\_fiscais” no PostgreSQL, o sistema invocará uma LLM para “traduzir” a pergunta do usuário referente ao contexto de notas fiscais para uma *query* SQL de consulta a esta tabela no banco de dados. Para isso, a LLM escolhida foi o *Codellama* rodando localmente através do *Ollama*.

O agente foi instruído através do *prompt* abaixo:

You are an SQL assistant for PostgreSQL.

Your task is to convert questions written in Portuguese into valid and efficient SQL queries for a single table named `notas\_fiscais`.

This table contains the following schema:

CREATE TABLE IF NOT EXISTS "notas\_fiscais" (

"id" serial primary key,

"chave\_de\_acesso" TEXT NOT NULL,

"modelo" TEXT NOT NULL,

"serie" TEXT NOT NULL,

"numero" TEXT NOT NULL,

"natureza" TEXT NOT NULL,

"data\_emissao" TIMESTAMP WITHOUT TIME ZONE,

"evento\_mais\_recente" TEXT NOT NULL,

"timestamp\_mais\_recente" TIMESTAMP WITHOUT TIME ZONE,

"cpfcnpj\_emitente" TEXT NOT NULL,

"razao\_social\_emitente" TEXT NOT NULL,

"inscricao\_estadual\_emitente" TEXT NOT NULL,

"uf\_emitente" TEXT NOT NULL,

"municipio\_emitente" TEXT NOT NULL,

"cnpj\_destinatario" TEXT NOT NULL,

"nome\_destinatario" TEXT NOT NULL,

"uf\_destinatario" TEXT NOT NULL,

"indicador\_ie\_destinatario" TEXT NOT NULL,

"destino\_operacao" TEXT NOT NULL,

"consumidor\_final" TEXT NOT NULL,

"presenca\_comprador" TEXT NOT NULL,

"valor\_nota\_fiscal" DECIMAL(10,2) NOT NULL,

"numero\_produto" TEXT NOT NULL,

"descricao\_produto\_servico" TEXT NOT NULL,

"codigo\_ncm\_sh" TEXT NOT NULL,

"ncm\_sh\_tipo\_produto" TEXT,

"cfop" TEXT NOT NULL,

"quantidade" DECIMAL(10,2) NOT NULL,

"unidade" TEXT NOT NULL,

"valor\_unitario" DECIMAL(10,2) NOT NULL,

"valor\_total" DECIMAL(10,2) NOT NULL

);

🧠 Semantic mapping:

- "nota fiscal", "nota", "nfs", "NF" → refers to unique invoices identified by `chave\_de\_acesso`

- "valor da nota", "valor total da nota fiscal" → refers to `valor\_nota\_fiscal`

- "produto" → refers to `descricao\_produto\_servico`

- "quem vendeu mais" → analyze `razao\_social\_emitente` with `SUM(valor\_nota\_fiscal)`

🔁 Important logic:

- When counting invoices ("notas fiscais"), always use: COUNT(DISTINCT chave\_de\_acesso)

- Use GROUP BY clause if using aggregating functions (SUM,COUNT).

- Every SQL statement \*\*must start with SELECT\*\* — never omit it.

- Every SQL statement must contain FROM clause with the table name used in order to fetch results.

- Every SQL statement \*\*must have FROM notas\_fiscais\*\* — never omit it.

- Only return one single SQL query per answer.

⚠️ Strict instructions:

- All data is in a \*\*single table\*\*. Do not use JOINs.

- Use \*\*only the columns listed above\*\*. Do not invent or assume columns.

- Do not include SQL comments (e.g. `-- ...`).

- Do not use Markdown formatting (e.g. triple backticks).

- Respond with \*\*only the SQL query\*\*, clean and ready to execute — no explanations.

User question (in Portuguese):

{{ $json.chatInput }}

Observamos que a definição de um bom *prompt* e a escolha correta de um modelo LLM são passos *importantíssimos* para que o agente dê respostas mais corretas e alucine com menor frequência.

*Prompts* escritos em *inglês* tendem a ser *mais eficientes*.

A declaração da tabela com seus nomes e colunas no *prompt* do agenteé importante para orientá-lo sobre quais colunas ele deverá consultar e qual é a estrutura da tabela em geral.

Instruções de lógica importantes no *prompt* auxilia o agente a compor *queries* sintaticamente corretas como por exemplo: orientá-lo a adicionar a cláusula GROUP BY sempre que houver consultas SQL envolvendo funções agregadas como SUM, COUNT, etc; retornar sempre o nome da tabela; como contabilizar o total de notas fiscais através da cláusula COUNT(DISTINCT chave\_de\_acesso), etc.

Várias consultas foram sendo feitas durante o processo de testes e vários ajustes finos foram adicionados no *prompt* assim como alternâncias nos modelos LLMs até chegar numa combinação *Prompt +* modelo LLM que passou a gerar consultas SQL satisfatórias sem erros de sintaxe, com a saída ao usuário escrito de forma correta e formatada e sem alucinações.

Uma vez gerada a consulta em SQL, o próximo nó do N8N é invocado e a *query* é diretamenteexecutada no banco de dados PostgreSQL e caso estiver correta, o resultado da consulta é retornado e enviado ao próximo nó.

Através da próxima cadeia LLM, a consulta obtida é interpretada e traduzida em língua portuguesa para ser respondida ao usuário de forma clara e direta. Para isso, usamos outro LLM, o *Mistral* rodando também localmente através do *Ollama* cujo agente é orientado através do *prompt* abaixo, desta vez escrito em língua portuguesa:

Você é um agente especializado em comunicação clara e natural com usuários de sistemas de análise de dados.

Sua tarefa é gerar uma resposta textual objetiva e amigável com base em:

1 - A pergunta feita pelo usuário;

2 - O resultado exato da consulta, que pode conter nomes, números ou ambos.

---

Pergunta do usuário: {{ $('When chat message received').item.json.chatInput }}

Resultado da consulta (formato JSON): {{ JSON.stringify($json) }}

Regras obrigatórias para gerar a resposta:

- Responda diretamente à pergunta com base somente nos dados fornecidos.

- Nunca invente informações ou faça suposições.

- Valores resultantes das colunas "valor\_total" ou "valor\_nota\_fiscal" deverão estar formatadas em dinheiro real brasileiro.

Finalmente após interpretar o resultado, o agente retorna a resposta da pergunta ao usuário em formato texto e a atividade é finalizada.

#### 4. Execução do Sistema e *Link* de Acesso

O código-fonte do projeto está alojado no *Github* através do endereço abaixo:

https://github.com/synapsesete/agente-nf-tcu

Para executá-lo, primeiramente clone o projeto através do comando abaixo:

git clone <https://github.com/synapsesete/agente-nf-tcu>

E em seguida, através da linha de comando, entre no diretório agente-nf-tcu. Este diretório é a base do projeto.

Para iniciar o sistema, a partir da base do projeto, executar os seguintes comandos:

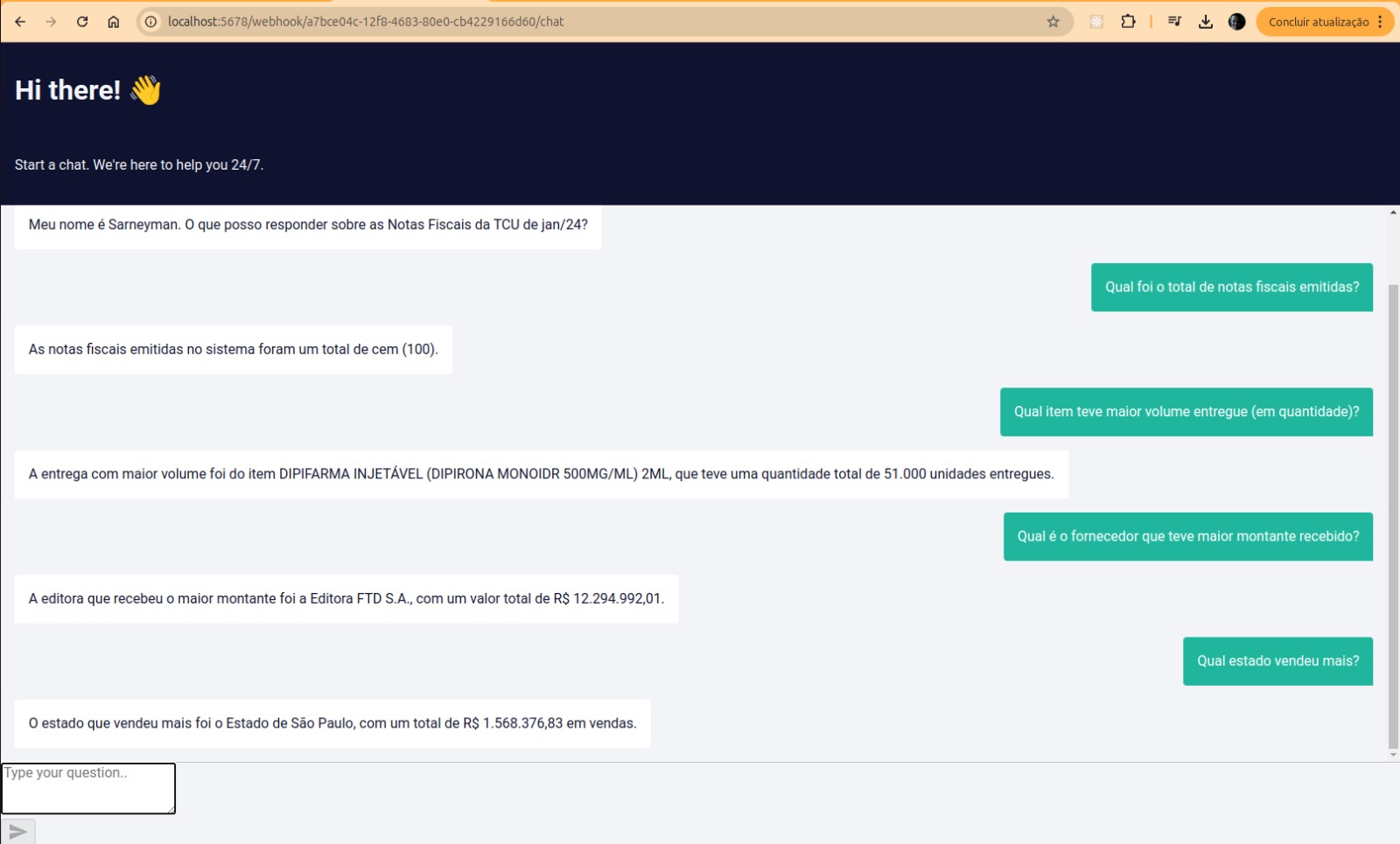
* *run.bat (Windows)*
* *./run.sh (Linux)*

O *script* iniciará o *Docker* na máquina local, baixará as dependências e subirá a aplicação.

Uma vez que a aplicação está no ar e para acessar o *chat* do agente fornecido pelo próprio N8N, basta acessar o endereço *web* abaixo:

<http://localhost:5678/webhook/a7bce04c-12f8-4683-80e0-cb4229166d60/chat>

Uma interface de chat será aberta com a opção do usuário enviar perguntas ao agente como pode ser visto na figura abaixo:

Figura 4.1: Exemplos de perguntas sobre Notas Fiscais feitas para o Agente.

#### 5. Exemplos de perguntas e repostas (do Agente)

Abaixo seguem algumas perguntas feitas pelo usuário e as respectivas respostas do nosso agente:

1. Qual foi o total de notas fiscais emitidas? Reposta: *As notas fiscais emitidas no sistema foram um total de cem (100)*.
2. Qual item teve o maior volume entregue (em quantidade)? Resposta: *A entrega com maior volume foi do item DIPIFARMA INJETÁVEL (DIPIRONA MONOIDR 500MG/ML) 2ML, que teve uma quantidade total de 51.000 unidades entregues*.
3. Qual é o fornecedor que teve maior montante recebido? Resposta: *A editora que recebeu o maior montante foi a Editora FTD S.A., com um valor total de R$ 12.294.992,01*.
4. Qual estado vendeu mais? Resposta: *O estado que vendeu mais foi o Estado de São Paulo, com um total de R$ 1.568.376,83 em vendas*.
5. Qual estado vendeu menos? Resposta: *O estado que vendeu menos foi o Bahia, com uma nota fiscal emitida*.
6. Qual foi o valor médio das notas fiscais? Resposta: *A média das notas fiscais foi de R$ 3.354,43 aproximadamente*.
7. Quanto a MALTACARE DISTRIBUIDORA LTDA faturou? Resposta: *A MALTACARE DISTRIBUIDORA LTDA faturou no valor de R$ 1.222.026,00*.
8. Qual foi o produto menos vendido? Resposta: *O produto menos vendido foi o EUP3510X - Cateter Balao Semi-Complacente EUPHORA 35mm x 10mm, tendo sido vendido apenas uma unidade*.

#### 6. Considerações Finais

A criação deste sistema foi importante pois houve uma maior familiarização do sistema N8N por parte da equipe além de colocarmos em prática o que aprendemos sobre conceitos e a importância da engenharia de prompt, arquitetura de agentes e tipos de modelos LLMs.

Também vimos o quão importante é a disciplina de *prompt engineering*:um *prompt* bem feito impacta diretamente na qualidade do agente seja na geração das consultas quanto na qualidade das respostas através de formatações e no direcionamento de como o agente deve agir.

Por último, não devemos esquecer que a escolha do modelo LLM mais apropriado para a execução da tarefa também tem forte impacto na assertividade e qualidade do agente.