Assistente Inteligente para Integração de Dados com Protocolo MCP Aplicado à Gestão de Colaboradores

Luca Aguiar Costa Carvalho[1], **Marlon Marcon**[1]

[1]Universidade Tecnológica Federal do Paraná (UTFPR)
Dois Vizinhos – PR – Brasil

lucaaguiar@alunos.utfpr.edu.br, marlonmarcon@utfpr.edu.br

Abstract. *This work presents the development of an intelligent system that integrates and queries multiple data sources using natural language. The solution, built in Python, demonstrates the practical use of the Model Context Protocol (MCP), enabling agents to automatically select tools based on descriptions. The system connects an Excel spreadsheet, a PostgreSQL database, a MongoDB database, and a public weather API. It allows operations like checking birthdays, retrieving weather forecasts, and finding vacation records, even handling approximate names and typos.*

Resumo. *Este trabalho apresenta o desenvolvimento de um sistema inteligente voltado à integração e consulta de diferentes fontes de dados por meio de linguagem natural. A solução construída em Python demonstra o uso prático do Model Context Protocol (MCP), permitindo a criação de agentes que entendem descrições de ferramentas e as selecionam automaticamente. O sistema conecta simultaneamente uma planilha Excel, um banco PostgreSQL, um banco MongoDB e uma API pública de clima. O assistente lida com nomes aproximados e erros de digitação, permitindo descobrir aniversários, prever o tempo e consultar férias.*

1. Introdução

A crescente fragmentação de dados em ambientes corporativos, dispersos entre sistemas relacionais, não relacionais e APIs, impõe desafios significativos à interoperabilidade e à descoberta de conhecimento [Patel 2019]. Para favorecer a integração de fontes e serviços de dados, o padrão MCP (Model Context Protocol)[Anthropic 2025b], foi criado para permitir que um LLM (*Large Language Models*) tome decisões autônomas sobre qual ferramenta utilizar com base em descrições semânticas dessas ferramentas. Este trabalho apresenta uma aplicação fundamentada no MCP, para criação de agentes inteligentes capazes de interagir com múltiplas fontes de dados via linguagem natural.

A arquitetura adota o padrão *LLM Agents*, no qual o modelo de linguagem atua como um orquestrador central que seleciona e executa ferramentas dinamicamente para gerar respostas [Wang et al. 2024]. A viabilidade desta abordagem é demonstrada através de uma ferramenta desenvolvida em Streamlit [Streamlit 2025], que integra dados de planilhas Excel, bancos PostgreSQL [PostgreSQL 2025] e MongoDB [MongoDB 2025], além da API Open-Meteo. O sistema utiliza o modelo Claude, da Anthropic, aproveitando sua capacidade avançada de raciocínio e vasta janela de contexto para a orquestração eficiente das ferramentas [Anthropic 2025a].

2. Metodologia

A construção da aplicação baseou-se em ciclos de prototipação incremental, tendo como princípio a criação de um sistema capaz de realizar consultas a dados dispersos por meio de linguagem natural. O desenvolvimento foi conduzido em Python e estruturado nas seguintes etapas:

- **Implementação das ferramentas:** Cada consulta (por nome, férias, cursos, clima) foi desenvolvida como uma função independente com entrada padronizada.
- **Documentação semântica e MCP:** As ferramentas foram descritas com *docstrings* claras (nome, parâmetros, descrição e tipo de retorno), seguindo o padrão do Model Context Protocol (MCP). Isso fornece ao LLM contexto para tomar decisões autônomas.
- **Orquestração com LangChain:** A biblioteca LangChain foi utilizada para conectar o modelo de linguagem (Claude) às funções Python. Adotou-se o agente do tipo *zero-shot-react-description*, que interpreta a pergunta e escolhe a ferramenta adequada baseada na sua descrição.
- **Integração de fontes de dados:** Foram conectadas quatro fontes distintas para demonstrar a interoperabilidade: uma planilha Excel (dados básicos, via *pandas*), um banco PostgreSQL (cursos, via *psycopg2*), um banco MongoDB (férias, via *pymongo*) e a API Open-Meteo (clima, via *requests*).
- **Interface do usuário:** A biblioteca Streamlit foi empregada para construir uma interface onde o usuário insere perguntas em linguagem natural e recebe as respostas formatadas.

2.1. Arquitetura da Aplicação

A arquitetura da aplicação pode ser visualizada na Figura 1, que representa a interação entre os componentes principais: a interface, o agente (LLM + MCP), o servidor MCP e as fontes de dados (estruturadas e externas). O agente recebe uma pergunta em linguagem natural, analisa a solicitação, interpreta as ferramentas disponíveis registradas no servidor via MCP e executa a função apropriada. Os dados podem vir de diferentes fontes (planilhas, bancos de dados e APIs), e o retorno é apresentado ao usuário de forma estruturada.

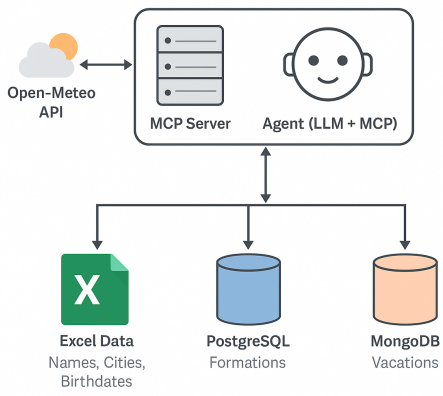

Figura 1. Arquitetura geral da aplicação com uso do MCP

2.2. Metodologia de Desenvolvimento

O sistema foi desenvolvido através de ciclos de prototipação incremental, priorizando a modularidade e a semântica das operações. Inicialmente, as funcionalidades foram implementadas como funções independentes e detalhadas via documentação semântica (*docstrings*), permitindo que o framework LangChain as registrasse como *tools* acionáveis pelo agente.

A integração de dados abrangeu quatro frentes tecnológicas:

- **Excel (*pandas*):** para armazenamento de dados cadastrais básicos;
- **PostgreSQL (*psycopg2*):** para gestão de registros estruturados de formações;
- **MongoDB (*pymongo*):** para dados não relacionais de férias;
- **API Open-Meteo (*requests*):** para consumo de dados climáticos externos.

A interface foi construída com Streamlit, viabilizando a interação em linguagem natural. Para garantir a robustez, realizaram-se testes de validação com entradas ambíguas e erros ortográficos. A arquitetura resultante, provou ser uma solução eficiente e extensível, facilitando a inclusão de novas ferramentas e a adaptação do sistema para outros domínios corporativos, como chatbots e repositórios internos.

3. Resultados e Funcionalidades

A aplicação resultante demonstra a capacidade do protocolo MCP de intermediar o acesso a dados heterogêneos, apresentando um sistema que não apenas integra fontes distintas, mas exibe resiliência a imprecisões e alta capacidade de composição lógica.

3.1. Consultas Resilientes e Integração de Dados

Um dos resultados mais significativos é a superação da necessidade de *queries* exatas. O agente inteligente, atuando como orquestrador [Wang et al. 2024], utiliza as descrições semânticas das ferramentas para lidar com a fragmentação das informações [Patel 2019]:

- **Resiliência a Erros:** Na busca de aniversários (Excel) e férias (MongoDB [MongoDB 2025]), o sistema realiza buscas aproximadas (*fuzzy matching*). Ao receber entradas como "Charlin" ou "Carlinhos", o modelo Claude [Anthropic 2025a] interpreta as variações e testa correspondências como "Charles" ou "Carlos E.", superando as limitações de buscas tradicionais.
- **Dados Estruturados e Externos:** O sistema mantém a consistência transacional ao consultar formações no PostgreSQL [PostgreSQL 2025] e integra dados em tempo real via API Open-Meteo para previsões climáticas, demonstrando flexibilidade entre o ecossistema interno e a web.

3.2. Orquestração e Consultas Compostas

O avanço mais expressivo desta implementação é a capacidade de realizar consultas compostas sem codificação prévia de fluxos (*hardcoded*). Graças ao padrão *LLM Agents*, o sistema decompõe perguntas complexas que exigem a junção de múltiplas fontes.

Por exemplo, ao ser questionado: "Qual era a temperatura no aniversário do Charlin Chaplin?", o agente executa um plano de ação autônomo: primeiro localiza a data e cidade na planilha Excel para, em seguida, alimentar a API climática com esses parâmetros. Esse raciocínio contextual é evidenciado na Figura 2.

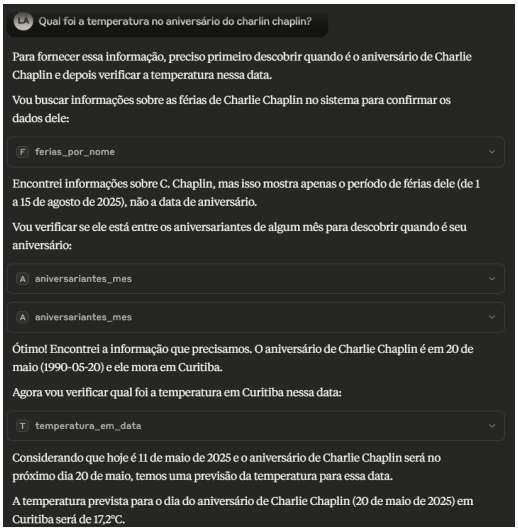

Figura 2. Teste realizado na interface Claude Desktop com execução multi-etapas

4. Conclusão

A abordagem apresentada mostrou-se eficiente para criar um sistema inteligente e extensível de consulta a dados heterogêneos. A combinação do LangChain com o protocolo MCP demonstrou ser uma solução prática, transformando funções comuns em operações acionáveis por linguagem natural.

Mais do que uma aplicação específica, o trabalho mostra como é possível unificar fontes em um único ponto de acesso flexível. Essa arquitetura permite que novas ferramentas sejam facilmente adicionadas, ampliando o escopo para outros domínios corporativos e facilitando a descoberta de conhecimento a partir de dados brutos, em total alinhamento com os desafios atuais da área de bancos de dados.

Como trabalhos futuros, pretende-se explorar aspectos de custo computacional, latência de resposta, segurança de dados, privacidade de informações sensíveis e escalabilidade do sistema para ambientes corporativos. Além disso, será investigada a comparação com soluções de integração de dados heterogêneos similares, a fim de validar a efetividade da abordagem MCP em relação a outras metodologias consolidadas na literatura.

Referências

Anthropic (2025a). Claude api documentation. Acesso em 22 de junho de 2025.

Anthropic (2025b). Model context protocol (mcp). Acesso em 22 de junho de 2025.

MongoDB (2025). Mongodb manual. Acesso em 22 de junho de 2025.

Patel, J. (2019). Bridging data silos using big data integration. International Journal of Database Management Systems, 11(3):01–06.

PostgreSQL (2025). Postgresql documentation. Acesso em 22 de junho de 2025.

Streamlit (2025). Streamlit documentation. Acesso em 22 de junho de 2025.

Wang, L., Ma, C., Feng, X., Zhang, Z., Yang, H., Zhang, J., Chen, Z., Tang, J., Chen, X., Lin, Y., et al. (2024). A survey on large language model based autonomous agents. Frontiers of Computer Science, 18(6):186345.