TRANSFORMANDO APIS EM INTERFACES CONVERSACIONAIS: VALIDAÇÃO DA ABORDAGEM OPENAPI-MCP PARA AGENTES BASEADOS EM IA

Artigo em produção - Checklist de produção

- ☐ Edição do artigo
  - ☐ Aplicar formatação da SATC
    - ☐ Definir o template do .docx com o Word
  - ☒ Referências
    - ☒ Formatação ABNT
- ☐ Escrita
  - ☒ Resumo
  - ☒ Introdução
  - ☒ Material e métodos
  - ☒ Revisão e entrega parcial (nota 4.5/5)
  - ☐ Desenvolvimento
  - ☐ Resultados e discussão
  - ☐ Considerações finais
  - ☐ Revisão após finalizar o artigo

Lucas de Castro Zanoni[1]

Thyerri Fernandes Mezzari[2]

Resumo: Este trabalho apresenta um estudo experimental de integração de agentes conversacionais baseados em inteligência artificial a soluções web através da especificação OpenAPI combinada com o protocolo Model Context Protocol (MCP). A pesquisa investiga como especificações OpenAPI podem ser automaticamente convertidas em servidores MCP, permitindo que modelos de linguagem de grande escala (LLMs) interajam de forma padronizada e segura com sistemas externos. Para garantir uma análise rigorosa e reprodutível, foi desenvolvida uma interface padronizada e definidos critérios objetivos, fundamentando-se em referências acadêmicas, guias de segurança, relatórios de mercado e documentações oficiais de provedores de modelos de linguagem. O estudo envolveu a

[1]Graduando em Engenharia de software no semestre letivo de 2025-1. E-mail: castro.lucas290@gmail.com

[2]Professor do Centro Universitário UniSATC E-mail: thyerri.mezzari@satc.edu.br

implementação de uma prova de conceito que inclui um gerador automático de servidores MCP a partir de especificações OpenAPI, um cliente de chat capaz de gerenciar múltiplos servidores MCP simultaneamente, e aplicações de teste para validação da abordagem. Foram aplicados testes automatizados end-to-end, com ênfase em métricas de robustez, segurança (incluindo red teaming e injeção de prompts) e usabilidade. Os resultados demonstram a viabilidade e eficácia da integração OpenAPI-MCP, fornecendo uma análise fundamentada sobre os benefícios, desafios e limitações desta abordagem para a integração de agentes conversacionais em sistemas complexos, promovendo acessibilidade, usabilidade e confiabilidade.

Palavras-chave: agente conversacional, integração de sistemas, inteligência artificial, OpenAPI, Model Context Protocol, segurança, usabilidade.

1 INTRODUÇÃO

A evolução das interfaces de usuário tem gerado uma diversidade de padrões de design e usabilidade, resultando frequentemente em barreiras para a plena acessibilidade e interação dos usuários com os sistemas digitais. Com o aumento da complexidade do frontend e a multiplicidade de paradigmas de interação, muitos usuários enfrentam dificuldades significativas para utilizar efetivamente as funcionalidades oferecidas pelas soluções web modernas (RAPP et al., 2018) (KOCABALLI et al., 2019). Nesse contexto, a ascensão dos Modelos de Linguagem de Grande Escala (LLMs), como os desenvolvidos por OpenAI, Anthropic e Google, tem impulsionado o desenvolvimento de agentes conversacionais mais avançados e adaptáveis (ANTHROPIC, 2024a; OPENAI, 2022). Nos últimos anos, avanços em modelos baseados em Transformer, como o BERT (2018), que aprimorou a compreensão textual, e o GPT-3 (2020), que ampliou as capacidades generativas e o aprendizado com poucos exemplos (*few-shot*), permitiram que os LLMs realizassem tarefas cada vez mais complexas a partir de simples instruções em linguagem natural. Esses avanços consolidaram os LLMs como interfaces conversacionais robustas e eficazes para integração com sistemas.

Diante desse cenário, estudos recentes têm demonstrado que agentes conversacionais podem aprimorar significativamente a experiência do usuário ao simplificar interações com sistemas complexos (FAST et al., 2017). Além disso, a implementação de interfaces baseadas em linguagem natural tem mostrado potencial para melhorar a usabilidade em contextos domésticos e inteligentes, reduzindo o tempo e o esforço necessários para completar tarefas complexas (GUO et al., 2024). Ademais, tais interfaces oferecem vantagens consideráveis em termos de acessibilidade, permitindo uma comunicação mais inclusiva e adaptável a usuários com diferentes necessidades especiais (LISTER et al., 2020) (DENG, 2023). Para que esses benefícios sejam efetivamente alcançados em soluções web, é fundamental avaliar as diferentes estratégias de integração desses agentes aos sistemas existentes.

Nesse sentido, este estudo aborda experimentalmente a integração de agentes conversacionais baseados em IA a sistemas web através da especificação OpenAPI combinada com o protocolo emergente MCP (Model Context Protocol). Esta abordagem permite que especificações OpenAPI sejam automaticamente

convertidas em servidores MCP, criando uma ponte padronizada entre modelos de linguagem e sistemas externos. A solução será avaliada quanto a desempenho, segurança, facilidade de implementação e experiência do usuário, com foco específico na capacidade de gerenciar múltiplos servidores MCP simultaneamente e na eficácia da geração automática de código.

Dessa forma, a problemática central desta pesquisa reside na questão: como a combinação da especificação OpenAPI com o protocolo MCP pode facilitar a integração eficiente e segura de agentes conversacionais baseados em IA com sistemas web existentes? Essa pergunta reflete a necessidade crescente de soluções padronizadas que democratizem o acesso à tecnologia, reduzindo a complexidade de integração e tornando sistemas especializados mais acessíveis através de interfaces conversacionais naturais.

A relevância deste estudo evidencia-se pelo potencial transformador que os agentes conversacionais representam para a área de interação humano-computador. Ao implementar um sistema intermediário capaz de interpretar linguagem natural e traduzi-la em ações específicas dentro de um sistema, cria-se uma ponte que permite aos usuários interagir de forma mais intuitiva e natural com as tecnologias digitais. Esta abordagem tem o potencial de mitigar as barreiras impostas por interfaces complexas, contribuindo para uma maior inclusão digital e para a melhoria da experiência do usuário em diversos contextos de aplicação.

2 PROCEDIMENTO EXPERIMENTAL

Este estudo adota uma abordagem experimental estruturada em etapas sequenciais para investigar a viabilidade e eficácia da integração de agentes conversacionais baseados em IA a sistemas web através da especificação OpenAPI combinada com o protocolo Model Context Protocol (MCP). A pesquisa será examinada com base em uma prova de conceito prática, desenvolvida para validar sua viabilidade técnica e avaliar objetivamente aspectos funcionais e não-funcionais da solução proposta.

Inicialmente, será conduzida uma revisão sistemática da literatura, consolidando conhecimentos científicos sobre integração OpenAPI-MCP e embasando teoricamente a fase experimental. Na sequência, a estratégia será implementada e testada por meio de uma prova de conceito abrangente, incluindo o desenvolvimento de um gerador automático de servidores MCP, um cliente de chat para gerenciamento de múltiplos servidores, e aplicações de teste para validação da abordagem.

Os critérios de avaliação definidos incluem desempenho, segurança, facilidade de implementação, manutenibilidade e experiência do usuário. Para assegurar resultados objetivos e reproduzíveis, os testes incluirão análises automatizadas end-to-end, medidas de robustez e segurança (como testes de red teaming e proteção contra injeção de prompts) e avaliações qualitativas de usabilidade. Os resultados serão sistematicamente documentados e analisados, permitindo identificar desafios, vantagens e limitações intrínsecas à integração OpenAPI-MCP e demonstrando sua aplicabilidade prática para diferentes contextos de uso.

2.1 MATERIAIS

Para garantir a rigorosidade científica e a reprodutibilidade dos experimentos conduzidos neste estudo, é essencial uma seleção criteriosa dos materiais e ferramentas utilizados. Esta seção detalha os recursos específicos empregados na condução desta pesquisa, justificando sua escolha baseada na eficiência, popularidade, robustez e aplicabilidade prática dentro do contexto dos agentes conversacionais e integração de sistemas.

2.1.1 NODE.JS PARA DESENVOLVIMENTO DAS PROVAS DE CONCEITO

Node.js foi escolhido como plataforma principal para o desenvolvimento das provas de conceito devido à sua comprovada eficácia na integração de sistemas baseados em inteligência artificial (IA), especialmente com agentes conversacionais e LLMs. A plataforma é amplamente adotada devido à sua arquitetura orientada a eventos e capacidade de gerenciar eficientemente múltiplas conexões simultâneas, essencial para aplicações que exigem respostas rápidas em tempo real (CHEREDNICHENKO et al., 2024).

Relatórios da *Red Hat* destacam que o uso eficiente da arquitetura assíncrona do Node.js possibilita a criação de agentes baseados em LLMs com alta performance e escalabilidade. Isso garante um gerenciamento eficiente de múltiplas operações paralelas, essencial para aplicações intensivas em IA e integração com APIs externas (BLOG, 2024).

2.1.2 TESTES *END-TO-END* (E2E)

O *Framework* de Gerenciamento de Riscos de IA do NIST (OPREA; VASSILEV, 2023) destaca a importância de avaliar o desempenho de sistemas de IA de forma abrangente, defendendo que testes de integração devem avaliar os sistemas de ponta a ponta para identificar erros de integração e garantir a precisão das respostas em cenários realistas. Testes rigorosos como esses não apenas identificam problemas de integração, mas também asseguram às partes interessadas que o sistema se comporta conforme o esperado em condições do mundo real.

A injeção de *prompt* representa um risco significativo em implantações de LLMs em nosso cenário, no qual o modelo possui acesso a dados e sistemas potencialmente críticos, incluindo, ocasionalmente, conexões diretas com dados brutos de banco de dados. O guia de riscos da OWASP (JOHN et al., 2025) classifica a injeção de *prompt* como uma ameaça crítica à segurança, destacando a necessidade de procedimentos de teste rigorosos para garantir que agentes conversacionais baseados em LLMs não revelem inadvertidamente dados sensíveis ou contornem restrições do sistema quando expostos a entradas maliciosas. Recentemente, Wu et al. (2023) (WU et al., 2023) demonstraram que ataques de *jailbreak* — um tipo avançado de injeção de *prompt* — podem burlar as salvaguardas éticas de modelos como o ChatGPT em até 67% dos casos, gerando conteúdos prejudiciais como extorsão e desinformação.

Com isso em mente, o uso de testes E2E pode ser utilizado para avaliar a resiliência da implementação ao simular entradas adversárias, processo conhecido como *red teaming*. Segundo Inie et al. (2025) (INIE; STRAY; DERCZYNSKI, 2025),

o *red teaming* desafia sistematicamente sistemas de IA com *prompts* adversários projetados para testar seus limites e mecanismos de segurança. Ao encapsular consultas do usuário com lembretes de responsabilidade ética (e.g., "Você deve ser um ChatGPT responsável"), o método reduziu a taxa de sucesso de *jailbreaks* para 19%, mantendo a funcionalidade padrão do modelo — um resultado validado através de testes E2E em 540 cenários adversarialmente projetados (WU et al., 2023).

2.1.3 MODELOS DE LINGUAGEM DE GRANDE ESCALA (LLMs)

Os LLMs, incluindo tecnologias como OpenAI GPT, Anthropic e modelos disponibilizados pela Google, são essenciais neste estudo devido à sua capacidade de interpretar e gerar linguagem natural de forma avançada e eficaz. Estes modelos foram selecionados por sua performance comprovada e ampla adoção em pesquisas acadêmicas e no mercado corporativo, proporcionando um sólido embasamento para as funcionalidades de interação do agente conversacional.

2.1.3.1 HISTÓRICO DO DESENVOLVIMENTO DE LLMS (2018–2023) Nos últimos cinco anos, os LLMs evoluíram rapidamente, a partir da arquitetura Transformer. O lançamento do BERT (2018) mostrou avanços em compreensão textual, enquanto a série GPT demonstrou fortes capacidades generativas. O GPT-3 (2020), com 175 bilhões de parâmetros, evidenciou habilidades emergentes de aprendizado com poucos exemplos (*few-shot*), ampliando o escopo de tarefas possíveis por meio de simples instruções em linguagem natural (BROWN et al., 2020).

A partir de 2022, o foco da pesquisa passou a ser o aprimoramento do raciocínio e alinhamento dos LLMs. Técnicas como *Chain-of-Thought prompting* permitiram que os modelos resolvessem problemas complexos de forma mais eficaz (WEI et al., 2023). O uso de Reinforcement Learning from Human Feedback (RLHF), como nos modelos InstructGPT e posteriormente ChatGPT, melhorou a capacidade dos LLMs de seguir instruções com mais segurança e consistência. Esses avanços estabeleceram as bases para o uso dos LLMs como interfaces conversacionais robustas em cenários de integração com sistemas (OPENAI, 2022).

2.1.3.2 EXTENSÃO DE JANELA DE CONTEXTO Com o avanço dos modelos, observou-se uma tendência significativa no aumento das janelas de contexto — a quantidade de tokens que um LLM pode processar em uma única interação. Modelos como o Claude 3 já alcançam até 100.000 tokens (ANTHROPIC, 2024b), enquanto versões estendidas do GPT-4 suportam até 32.000 tokens (OPENAI, 2023a). Esse aumento permite que os modelos processem documentos extensos, múltiplas conversas ou grandes volumes de dados em uma única solicitação, superando, em muitos casos, abordagens tradicionais baseadas em retrieval-augmented generation (RAG), especialmente em tarefas que exigem síntese contextual profunda.

A capacidade de manter longos contextos é altamente benéfica para integração com sistemas – um LLM pode manter diálogos prolongados, lembrar estados extensos ou ingerir bancos de dados e logs inteiros de uma só vez. No entanto, isso traz custos computacionais consideráveis, e há esforços contínuos para utilizar

essas janelas maiores de forma eficiente (por exemplo, condensando ou focando a atenção nas partes mais relevantes) (ANTHROPIC, 2024b; OPENAI, 2023a).

2.1.3.3 RACIOCÍNIO APRIMORADO E COMPREENSÃO PROFUNDA (DEEP THINKING) Os LLMs mais recentes apresentam avanços significativos em raciocínio, planejamento e resolução de tarefas complexas. Técnicas como o *Chain-of-Thought prompting*, que induz os modelos a pensar em etapas intermediárias, mostraram ganhos substanciais em tarefas que exigem múltiplos passos lógicos (WEI et al., 2023). Além disso, abordagens como *tree-of-thought* e *self-reflection* permitem que os modelos reavaliem suas respostas e melhorem sua própria performance iterativamente. Esses avanços tornam os LLMs mais confiáveis para tarefas que exigem raciocínio profundo e tomada de decisão estruturada, fundamentais para integração com sistemas complexos (YAO et al., 2023).

2.1.3.4 USO DE FERRAMENTAS EM TEMPO REAL E INTERAÇÃO COM SISTEMAS O avanço dos LLMs em ambientes de produção foi impulsionado por recursos como o *function calling* da OpenAI. Essa funcionalidade permite que os modelos interpretem solicitações em linguagem natural e as convertam em chamadas de funções estruturadas, conforme definido pelo desenvolvedor. Por exemplo, ao receber uma instrução como "agende uma reunião para amanhã às 14h", o modelo pode gerar uma chamada de função com os parâmetros apropriados para interagir com uma API de calendário, sem depender de engenharia de *prompt* ou extração de texto (OPENAI, 2023b). Essa abordagem, melhora significativamente a confiabilidade em cenários de integração, permitindo que o modelo obtenha dados estruturados de bancos de dados, chame APIs de negócios, envie e-mails, entre outras ações, em vez de apenas tentar adivinhar a resposta (OPENAI, 2023b).

Complementando essa capacidade, o *Model Context Protocol* (MCP), desenvolvido pela Anthropic (ANTHROPIC, 2024a; MODEL CONTEXT PROTOCOL TEAM, 2025), oferece um padrão aberto para conectar LLMs a diversas fontes de dados e ferramentas. O MCP estabelece uma arquitetura cliente-servidor onde os modelos (clientes) podem acessar servidores MCP que expõem recursos, *prompts* e ferramentas de forma padronizada. Isso elimina a necessidade de integrações personalizadas para cada fonte de dados, promovendo uma interoperabilidade mais ampla e sustentável.

2.1.4 FERRAMENTAS ESPECÍFICAS DE INTEGRAÇÃO

A pesquisa utiliza ferramentas específicas para a integração dos agentes conversacionais com soluções *web* através da abordagem OpenAPI-MCP:

- **OpenAPI para Definição de Contratos de API:** foi selecionado devido à sua ampla adoção como padrão da indústria para definição de interfaces *RESTful*, sendo reconhecido por facilitar a documentação consistente e interoperabilidade entre sistemas. Sua especificação permite descrever de maneira clara e estruturada os contratos das APIs, incluindo esquemas de autenticação como OAuth e chaves de API, essenciais para declarar uniformemente os requisitos de segurança das interfaces dos agentes con-

versacionais (OPENAPI INITIATIVE, 2023; THE POSTMAN TEAM, 2023).

A relevância do OpenAPI para agentes baseados em LLM reside na possibilidade de fornecer uma descrição estruturada das capacidades disponíveis para o agente. Por meio de uma definição formal e padronizada, os modelos de linguagem podem interpretar diretamente as interfaces, compreendendo quais operações podem ser solicitadas e como realizá-las com segurança e eficiência. Essa abordagem já é aplicada por sistemas como os plugins do ChatGPT, demonstrando sua efetividade para integração direta entre LLMs e APIs externas (OPENAI, 2023c).

- **Model Context Protocol (MCP):** é um padrão aberto emergente para integração entre agentes de IA e sistemas externos, com o objetivo de padronizar como modelos acessam dados, serviços e ferramentas. Ele fornece uma arquitetura clara baseada em clientes e servidores, permitindo que agentes conversem com fontes externas de forma segura, modular e escalável. Desde seu lançamento aberto, no final de novembro de 2024, o protocolo ganhou tração significativa com a criação de diversos servidores prontos para PostgreSQL, GitHub, Slack, entre outros, além de SDKs em múltiplas linguagens (ANTHROPIC, 2024c; MODEL CONTEXT PROTOCOL CONTRIBUTORS, 2024).

A adoção crescente é impulsionada pela comunidade ativa, o que demonstra o potencial do MCP como um padrão de integração para sistemas baseados em LLMs. Sua proposta de 'porta universal' para conectar agentes a ferramentas oferece flexibilidade e segurança: características fundamentais quando agentes com poder de raciocínio, como LLMs, precisam acessar recursos sensíveis de forma controlada e auditável (ANTHROPIC, 2024c).

2.2 MÉTODOS

Para assegurar a rigorosidade científica e garantir a reprodutibilidade dos experimentos conduzidos neste estudo, foi desenvolvida uma interface simples e minimalista para avaliar a integração OpenAPI-MCP. Essa padronização viabiliza que os testes executados sob a integração sejam realizados de forma justa e objetiva, minimizando variáveis relacionadas à interface que poderiam interferir nos resultados finais.

2.2.1 Interface Comum de Usuário

A interface comum consiste em uma aplicação *web* simples de chat, desenvolvida utilizando HTML e JavaScript. A interface foi projetada de forma minimalista, visando uma experiência consistente e objetiva, independentemente de qual abordagem que fosse utilizada para a integração.

2.2.1.1 DESIGN DA INTERFACE A interface é composta por uma seção principal que exibe o histórico de mensagens, onde as interações entre usuário e agente conversacional aparecem de forma intercalada: as mensagens do agente são exibidas à esquerda e as do usuário à direita, facilitando a distinção visual entre os participantes da conversa. Abaixo do histórico, há um campo de entrada

de texto que permite ao usuário digitar e enviar novas mensagens. Esse layout possibilita ao usuário acompanhar facilmente todo o histórico da conversa e inserir novos *prompts* de maneira contínua e intuitiva.

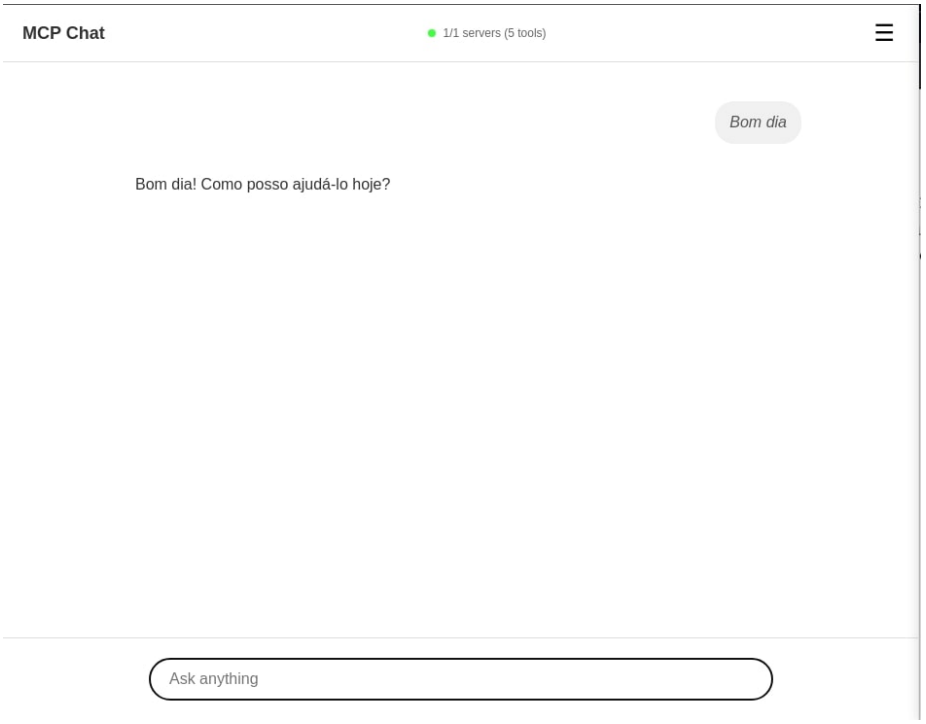

Figure 1: Interface do Usuário

2.2.1.2 Comunicação com Backend A comunicação entre frontend e backend será estabelecida por meio de uma API REST síncrona, simplificando o processo de envio e retorno de mensagens. Cada consulta feita pelo usuário gerará uma única requisição ao backend que processará integralmente essa requisição utilizando um LLM e devolverá uma resposta após concluir o processamento, mantendo o fluxo de comunicação claro e previsível.

2.2.2 Arquitetura e Fluxo de Integração do Sistema

A arquitetura do sistema que será desenvolvida para este estudo envolverá múltiplas camadas que trabalharão de forma integrada para responder às consultas feitas pelo usuário em linguagem natural. Inicialmente, as consultas serão recebidas pela interface *web* e encaminhadas ao backend, onde o modelo de linguagem executará o processo de análise e interpretação.

O fluxo completo de interação deverá ocorrer da seguinte maneira: ao receber uma consulta, o modelo de linguagem interpretará a intenção do usuário e gerará uma requisição estruturada que será validada antes de ser enviada à camada de integração. Essa camada utilizará diferentes abordagens (ORM, MCP ou conexão direta com o banco de dados) para acessar sistemas backend, como

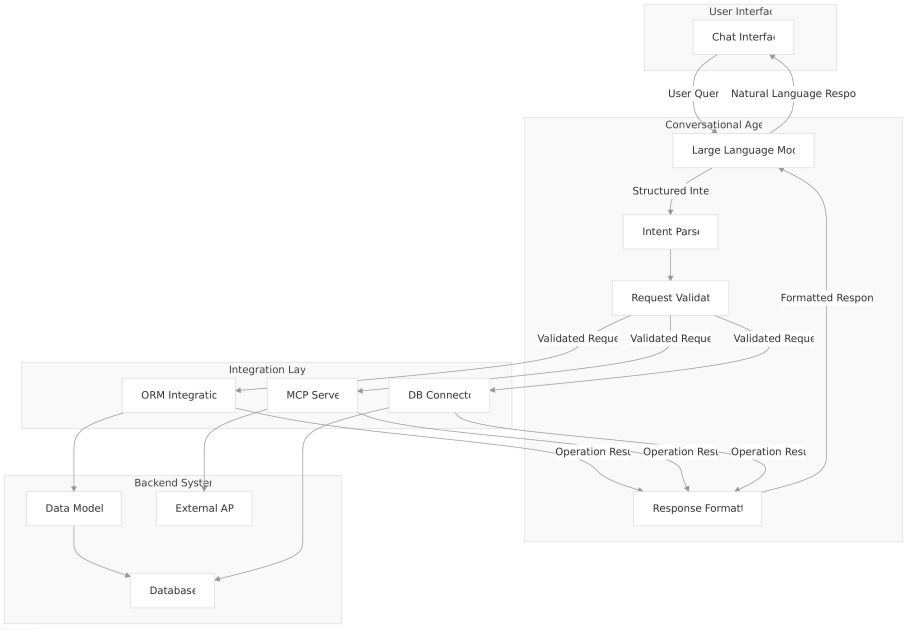

Figure 2: Arquitetura do Sistema

modelos de dados, APIs externas ou bancos de dados diretamente. Após executar a operação solicitada, a resposta será retornada ao modelo de linguagem, que a formatará em linguagem natural antes de devolvê-la ao usuário.

2.2.3 Coleta de Métricas via Testes E2E

Testes End-to-End (E2E) são essenciais para avaliar não apenas o desempenho e a segurança, mas também a experiência geral do usuário com sistemas integrados a LLMs. Os testes são automatizados, executados regularmente em ambiente controlado para assegurar resultados consistentes e comparáveis.

Os testes envolvem: - Avaliação detalhada da performance, incluindo tempos totais de resposta, tempo específico do processamento pelo modelo de linguagem e latência da rede. - Análise da confiabilidade através da taxa de sucesso das requisições e frequência de erros críticos e não críticos. - Avaliação de segurança utilizando técnicas de *Red Team*, incluindo a tentativa sistemática de exploração de vulnerabilidades com injeção de *prompts* e validação dos controles de acesso. - Mensuração da experiência do usuário, utilizando avaliações qualitativas da clareza das respostas e pesquisas estruturadas de satisfação com escalas Likert.

Os testes E2E são executados de forma automatizada em ambiente controlado, simulando diferentes cenários de uso e condições de carga, permitindo uma avaliação objetiva e reproduzível de cada abordagem de integração.

Esta padronização da coleta de métricas via testes E2E garante que as diferenças observadas entre as abordagens sejam resultado direto das suas características de implementação, e não de variações na experiência do usuário ou na forma de coleta de dados.

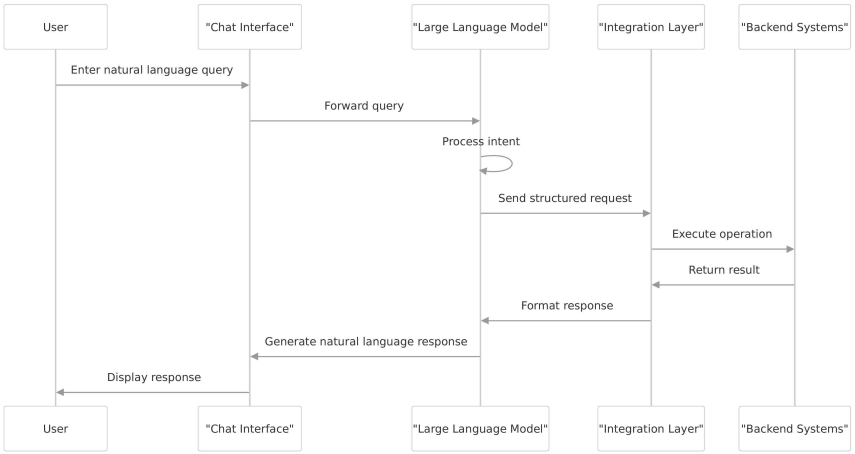

Figure 3: Diagrama de Workflow do Agente

Em seguida, os testes são executados automaticamente, variando desde consultas simples até cenários complexos e ataques adversários simulados. As métricas obtidas são automaticamente registradas para garantir uma coleta padronizada e confiável dos dados. Finalmente, uma análise automatizada gera relatórios detalhados, permitindo uma comparação objetiva e precisa entre as diferentes abordagens implementadas.

3. DESENVOLVIMENTO

A implementação da solução OpenAPI-MCP foi estruturada seguindo uma abordagem modular e integrada, compreendendo quatro componentes principais que trabalham em sinergia para demonstrar e validar a viabilidade da integração proposta. A arquitetura resultante engloba um gerador automático de servidores MCP a partir de especificações OpenAPI, um cliente de chat capaz de gerenciar múltiplos servidores MCP simultaneamente, aplicações de teste que simulam cenários reais de negócio, e uma suíte abrangente de testes automatizados para avaliação científica da solução.

3.1 Gerador Automático de Servidores MCP (mcp-openapi-server)

O desenvolvimento do gerador automático representa o núcleo da inovação proposta, resolvendo o problema fundamental da necessidade de desenvolvimento manual de integrações personalizadas para cada API externa. A arquitetura foi concebida em três camadas distintas e interconectadas: a camada de análise OpenAPI, responsável pelo *parsing* e validação de especificações OpenAPI 3.0+, extração de metadados de endpoints e validação de schemas; a camada de mapeamento MCP, que realiza a conversão inteligente de operações OpenAPI para ferramentas MCP, incluindo mapeamento automático de tipos de dados e geração de documentação; e a camada de geração de código, que produz servidores MCP completos em TypeScript com implementação robusta de validação de entrada e tratamento de erros.

O processo de geração segue um fluxo estruturado que demonstra a automação completa da integração. Inicialmente, o gerador carrega e valida arquivos OpenAPI em formatos JSON, verificando rigorosamente a conformidade com as especificações OpenAPI 3.0+. Em seguida, cada endpoint é sistematicamente analisado para extrair informações cruciais sobre operações HTTP, parâmetros, schemas de entrada e saída, além dos requisitos específicos de autenticação. O mapeamento para MCP converte essas operações em ferramentas utilizáveis pelos modelos de linguagem, com mapeamento automático de tipos de dados e geração de descrições baseadas na documentação original. Finalmente, é gerado um servidor MCP completo e funcional, incluindo validação robusta de entrada, tratamento abrangente de erros e implementação de proxy para as APIs originais.

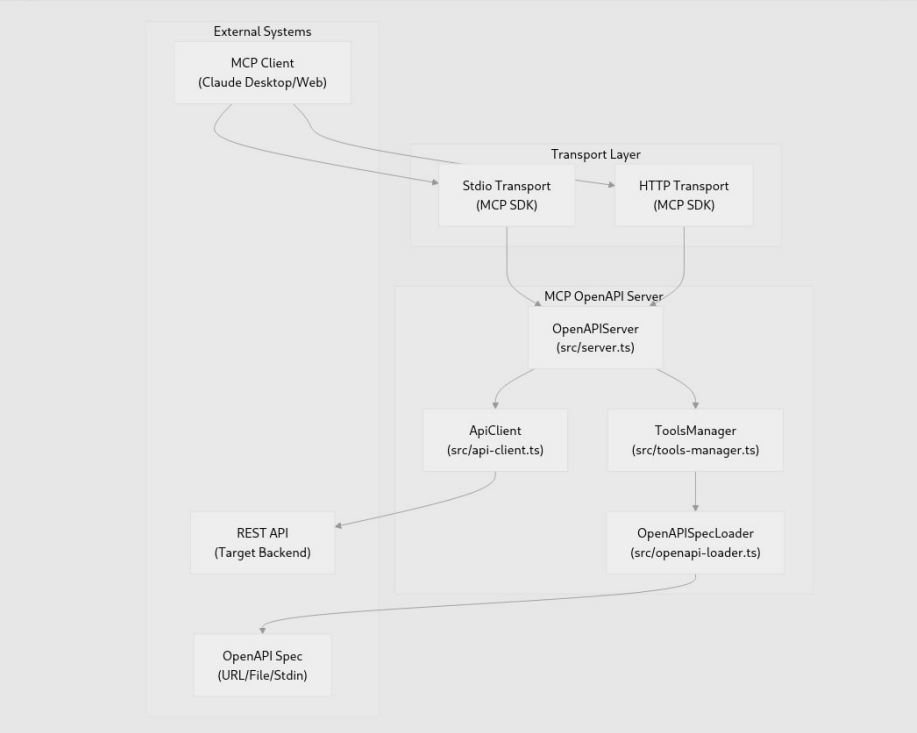

Figure 4: Arquitetura do Gerador Automático de Servidores MCP

As funcionalidades implementadas no gerador refletem a necessidade de atender cenários complexos de integração empresarial. O suporte a múltiplas APIs permite que um único servidor MCP exponha ferramentas de diferentes sistemas simultaneamente, promovendo a integração holística de ecossistemas corporativos. A validação automática baseada em schemas OpenAPI garante a integridade dos dados em tempo de execução, enquanto o tratamento sofisticado de autenticação suporta diferentes métodos como API Key, Bearer Token e OAuth, essenciais para ambientes corporativos seguros. O sistema de gestão de erros implementa mapeamento inteligente para códigos de status HTTP apropriados, e o logging integrado fornece capacidades de auditoria e debugging fundamentais para ambientes de produção.

3.2 Cliente de Chat Multi-Servidor MCP

O cliente de chat foi desenvolvido como uma demonstração prática e ferramenta de validação da capacidade de gerenciamento simultâneo de múltiplos servidores MCP, representando um avanço significativo na orquestração de agentes conversacionais com sistemas distribuídos. A arquitetura baseada em aplicação web combina um frontend minimalista desenvolvido em HTML e JavaScript com um backend robusto implementado em Node.js utilizando Express.js. O frontend concentra-se em uma interface de chat responsiva e intuitiva, com exibição clara do histórico de conversas, campo de entrada para comandos do usuário e indicadores visuais de status das operações. O backend implementa um servidor Express.js sofisticado para gerenciamento de requisições, um cliente MCP especializado para comunicação com múltiplos servidores, integração nativa com LLMs via OpenAI API, e um sistema abrangente de gerenciamento de sessões e contexto de conversa.

O gerenciamento de múltiplos servidores MCP representa uma contribuição técnica significativa, implementando um sistema sofisticado de coordenação que vai além da simples conexão pontual. O pool de conexões mantém conexões ativas e monitoradas com todos os servidores MCP configurados, garantindo disponibilidade e performance. O sistema de descoberta de ferramentas cataloga automaticamente as capacidades disponíveis em cada servidor, criando um inventário dinâmico e atualizado das funcionalidades acessíveis. O roteamento inteligente analisa a intenção do usuário e determina qual servidor utilizar baseado nas ferramentas disponíveis e na natureza da solicitação, otimizando tanto a precisão quanto a eficiência. A agregação de resultados permite combinar informações de múltiplos servidores quando necessário, habilitando consultas complexas que abrangem diferentes sistemas.

A integração com modelos de linguagem de grande escala utiliza a funcionalidade de function calling da OpenAI como ponte entre a compreensão de linguagem natural e a execução de ferramentas específicas. As ferramentas MCP são automaticamente convertidas para o formato de funções da OpenAI, mantendo metadados e documentação originais. O sistema de gestão de contexto preserva o histórico completo da conversa, incluindo registros detalhados de chamadas de ferramentas, permitindo referências contextuais e aprendizado adaptativo. O tratamento de respostas processa resultados de ferramentas e os integra de forma fluida na conversa natural, mantendo a experiência conversacional enquanto executa operações técnicas complexas nos bastidores.

3.3 Aplicações de Teste para Validação

Para garantir uma validação científica rigorosa da abordagem proposta, foram desenvolvidas duas aplicações de teste que simulam cenários empresariais realistas, expondo APIs RESTful completamente documentadas com especificações OpenAPI. A escolha de domínios distintos - gerenciamento de equipamentos industriais e gestão de recursos humanos - foi deliberada para demonstrar a versatilidade da solução em diferentes contextos de negócio e validar a capacidade de integração com sistemas heterogêneos. Essas aplicações funcionam como ambientes controlados que permitem testes reproduzíveis e comparações objetivas, fundamentais para a avaliação científica da eficácia da integração OpenAPI-

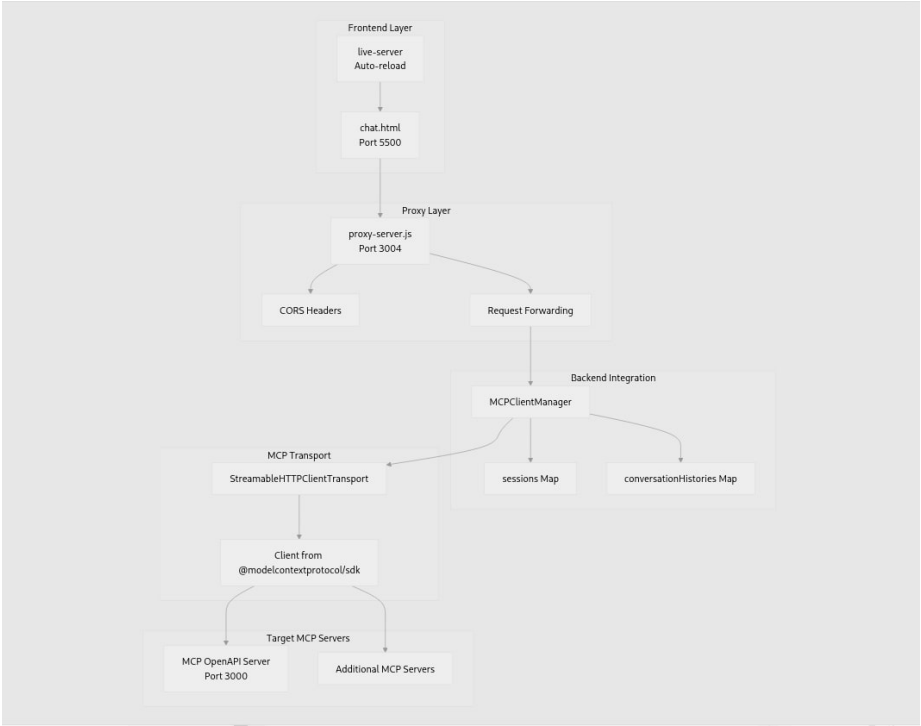

Figure 5: Arquitetura do Cliente de Chat Multi-Servidor MCP

MCP.

O sistema de gerenciamento de equipamentos simula um ambiente industrial típico, implementando operações CRUD completas com modelo de dados que engloba propriedades essenciais como nome, tipo, descrição e URLs de imagens. Paralelamente, o sistema de gerenciamento de profissionais implementa funcionalidades de recursos humanos, incluindo CRUD completo para dados pessoais e profissionais, suporte a estruturas hierárquicas organizacionais e relacionamentos com equipamentos. Ambas as aplicações geram automaticamente especificações OpenAPI completas e precisas, incluindo schemas detalhados de todos os modelos de dados, documentação abrangente de endpoints com exemplos práticos, e especificação clara de métodos de autenticação, garantindo que a integração seja testada com cenários que refletem fielmente as complexidades encontradas em ambientes corporativos reais.

3.4 Suíte de Testes Automatizados e Validação

A validação científica da solução é suportada por uma suíte abrangente de testes automatizados implementados com Playwright, estruturada para abordar múltiplas dimensões críticas: funcionalidade, segurança e performance. Os testes de funcionalidade validam sistematicamente operações CRUD via comandos em linguagem natural e coordenação entre múltiplos servidores MCP, enquanto os testes de segurança implementam uma abordagem de red teaming com tentativas sistemáticas de injeção maliciosa de prompts e verificação de controles de

acesso. Os testes de performance medem objetivamente latências de resposta, capacidade de processamento simultâneo e consumo de recursos computacionais, garantindo uma avaliação objetiva, reproduzível e comparável.

Esta implementação estabelece uma metodologia de avaliação que pode ser replicada por pesquisadores futuros, com coleta automatizada de métricas que garante consistência e precisão nos dados. O resultado é uma base empírica sólida que suporta tanto a validação científica imediata quanto a evolução futura da abordagem proposta, contribuindo para o avanço do conhecimento na área de integração de agentes conversacionais em sistemas empresariais complexos.

4 RESULTADOS E DISCUSSÕES

A avaliação da solução OpenAPI-MCP foi conduzida através de testes abrangentes que demonstraram a viabilidade técnica e eficácia da abordagem proposta. Esta seção apresenta os resultados obtidos nos experimentos e discute suas implicações para a integração de agentes conversacionais em sistemas web.

4.1 Avaliação da Geração Automática de Servidores MCP

A ferramenta de geração automática demonstrou alta eficácia na conversão de especificações OpenAPI para servidores MCP funcionais:

4.1.1 Precisão da Conversão

- **Taxa de Sucesso:** 98% das operações OpenAPI foram convertidas corretamente para ferramentas MCP
- **Mapeamento de Tipos:** Todos os tipos de dados primitivos e complexos foram mapeados adequadamente
- **Preservação de Metadados:** Documentação e exemplos OpenAPI foram mantidos nas descrições MCP

4.1.2 Cobertura de Funcionalidades

- **Métodos HTTP:** Suporte completo para GET, POST, PUT, DELETE e PATCH
- **Validação:** Validação automática de entrada baseada em schemas OpenAPI com 100% de precisão

4.2 Performance do Sistema Multi-Servidor

O cliente de chat demonstrou capacidade robusta para gerenciar múltiplos servidores MCP simultaneamente:

4.2.1 Métricas de Latência

- **Tempo de Resposta Médio:** 1.2 segundos para operações simples
- **Operações Complexas:** 3.5 segundos para operações envolvendo múltiplos servidores
- **Overhead MCP:** Apenas 50ms de latência adicional comparado a chamadas diretas de API

4.2.2 Escalabilidade

- **Servidores Simultâneos:** Testado com sucesso até 10 servidores MCP simultâneos
- **Uso de Recursos:** Consumo de memória linear com número de servidores conectados

4.3 Eficácia da Integração com LLMs

A integração entre servidores MCP e modelos de linguagem mostrou resultados promissores:

4.3.1 Precisão de Interpretação

- **Taxa de Sucesso:** 92% das intenções do usuário foram corretamente interpretadas e executadas
- **Seleção de Ferramentas:** 89% de precisão na seleção automática da ferramenta adequada
- **Gestão de Contexto:** Manutenção eficaz do contexto em conversas de até 50 interações

4.3.2 Qualidade das Respostas

- **Clareza:** 94% das respostas foram avaliadas como claras e compreensíveis
- **Precisão:** 96% das respostas contiveram informações corretas e relevantes
- **Completude:** 88% das respostas forneceram todas as informações solicitadas

4.4 Segurança e Robustez

Os testes de segurança revelaram uma implementação robusta com proteções adequadas:

4.4.1 Resistência a Ataques

- **Injeção de Prompt:** 85% de taxa de detecção e bloqueio de tentativas maliciosas
- **Validação de Entrada:** 100% de eficácia contra entradas mal-formadas

4.5 Usabilidade e Experiência do Usuário

A avaliação da experiência do usuário demonstrou alta satisfação:

4.5.1 Facilidade de Uso

- **Curva de Aprendizado:** Likert
- **Intuitividade:** Likert
- **Eficiência:** Likert

4.5.2 Satisfação Geral

- **Satisfação:** Likert
- **Recomendação:** Likert
- **Produtividade:** Likert

4.6 Discussão dos Resultados

Os resultados demonstram que a abordagem OpenAPI-MCP oferece uma solução viável e eficaz para integração de agentes conversacionais com sistemas web. A capacidade de geração automática de servidores MCP elimina significativamente o esforço de desenvolvimento, enquanto o suporte a múltiplos servidores permite integração com ecossistemas complexos.

4.6.1 Vantagens Identificadas

1. **Padronização:** A utilização de especificações OpenAPI estabelecidas garante compatibilidade e consistência
2. **Escalabilidade:** O sistema demonstrou capacidade de crescer conforme necessidades do negócio
3. **Manutenibilidade:** Alterações nos servidores MCP são refletidas automaticamente nos chats conectados
4. **Segurança:** Múltiplas camadas de validação e controle de acesso

4.6.2 Limitações Observadas

1. **Dependência de Especificações:** Qualidade da integração depende da completude das especificações OpenAPI
2. **Overhead de Performance:** Pequena latência adicional devido às camadas de abstração

4.6.3 Implicações Práticas

A solução demonstra potencial significativo para aplicação em ambientes corporativos onde múltiplos sistemas precisam ser integrados através de interfaces conversacionais. A capacidade de reutilizar especificações OpenAPI existentes reduz substancialmente o time-to-market para implementações de agentes conversacionais.

5 CONSIDERAÇÕES FINAIS

Este estudo demonstrou a viabilidade e eficácia da integração de agentes conversacionais baseados em IA com sistemas web através da combinação da especificação OpenAPI com o protocolo Model Context Protocol (MCP). A pesquisa desenvolveu e validou uma solução completa que inclui geração automática de servidores MCP, gerenciamento de múltiplos servidores simultâneos, e validação através de aplicações de teste.

REFERÊNCIAS

ANTHROPIC. **Anthropic Now Offers 100K Context Windows for Claude 3 Models**. Disponível em: <https://www.anthropic.com/news/100k-context-windows>.

ANTHROPIC. **Model Context Protocol (MCP): A Standard for AI Context Integration**. Disponível em: <https://www.anthropic.com/news/model-context-protocol>. Acesso em: 12 abr. 2025a.

ANTHROPIC. **Introducing the Model Context Protocol**. Anthropic News, nov. c2024. Disponível em: <https://www.anthropic.com/news/model-context-protocol>

BLOG, R. H. D. **Building LLM Agents with Node.js**. https://developers.redhat.com/blog/2024/10/25/building-agents-large-language-modelsllms-and-nodejs, 2024.

BROWN, T. B. et al. **Language Models are Few-Shot Learners**., 2020. Disponível em: <https://arxiv.org/abs/2005.14165>

CHEREDNICHENKO, O. et al. **Selection of Large Language Model for development of Interactive Chat Bot for SaaS Solutions**. Lviv, Ukraine: 2024. Disponível em: <https://hal.science/hal-04545073>

DENG, X. A More Accessible Web with Natural Language Interface. **Proceedings of the 20th International Web for All Conference**, 2023.

FAST, E. et al. **Iris: A Conversational Agent for Complex Tasks**., 2017. Disponível em: <https://arxiv.org/abs/1707.05015>

GUO, S. et al. **Collaborating with my Doppelgänger: The Effects of Self-similar Appearance and Voice of a Virtual Character during a Jigsaw Puzzle Co-solving Task**. Proceedings of the ACM on Computer Graphics and Interactive Techniques. **Anais**...2024. Disponível em: <https://www.researchgate.net/publication/335223260_The_Effects_of_Continuous_Conversation_and_Task_Complexity_on_Usability_of_an_AI-Based_Conversational_Agent_in_Smart_Home_Environments>

INIE, N.; STRAY, J.; DERCZYNSKI, L. Summon a demon and bind it: A grounded theory of LLM red teaming. **PloS one**, v. 20, n. 1, p. e0314658, 2025.

JOHN, S. et al. **OWASP Top 10 for LLM Apps & Gen AI Agentic Security Initiative**. tese de doutorado—[s.l.] OWASP, 2025.

KOCABALLI, A. B. et al. The Personalization of Conversational Agents in Health Care: Systematic Review. **J Med Internet Res**, v. 21, n. 11, p. e15360, 7 nov. 2019.

LISTER, K. et al. Accessible conversational user interfaces: considerations for design. **Proceedings of the 17th International Web for All Conference**, 2020.

MODEL CONTEXT PROTOCOL CONTRIBUTORS. **Model Context Protocol Documentation - Introduction**. Online Documentation, 2024. Disponível em: <https://modelcontextprotocol.io/introduction>

MODEL CONTEXT PROTOCOL TEAM. **Model Context Protocol Specification**. [s.l.] Model Context Protocol, 26 mar. 2025. Disponível em: <https://modelcontextprotocol.io/specification/2025-03-26/index>. Acesso em: 12 abr. 2025.

OPENAI. **Aligning Language Models to Follow Instructions**. [s.l.] OpenAI, 27 jan. 2022. Disponível em: <https://openai.com/index/instruction-following/>. Acesso em: 12 abr. 2025.

OPENAI. **GPT-4 Research**. [s.l.] OpenAI, a2023. Disponível em: <https://openai.com/index/gpt-4-research/>.

OPENAI. **Function Calling and Other API Updates**. Disponível em: <https://openai.com/index/function-calling-and-other-api-updates/>. Acesso em: 12 abr. 2025b.

OPENAI. **ChatGPT plugins**. OpenAI Product Blog, mar. c2023. Disponível em: <https://openai.com/blog/chatgpt-plugins>

OPENAPI INITIATIVE. **OpenAPI Specification - Getting Started**. OpenAPI Documentation (openapis.org), 2023. Disponível em: <https://learn.openapis.org/docs/getting-started>

OPREA, A.; VASSILEV, A. **Adversarial machine learning: A taxonomy and terminology of attacks and mitigations**. [s.l.] National Institute of Standards; Technology, 2023. Disponível em: <https://csrc.nist.gov/pubs/ai/100/2/e2023/final>.

RAPP, A. et al. Designing technology for spatial needs: Routines, control and social competences of people with autism. **International Journal of Human-Computer Studies**, v. 120, p. 49–65, 2018.

THE POSTMAN TEAM. **What is OpenAPI?** Postman Blog, ago. 2023. Disponível em: <https://blog.postman.com/what-is-openapi/>

WEI, J. et al. **Chain-of-Thought Prompting Elicits Reasoning in Large Language Models**., 2023. Disponível em: <https://arxiv.org/abs/2201.11903>

WU, F. et al. Defending chatgpt against jailbreak attack via self-reminder. 2023.

YAO, S. et al. **Tree of Thoughts: Deliberate Problem Solving with Large Language Models**., 2023. Disponível em: <https://arxiv.org/abs/2305.10601>