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

Lucas de Castro Zanoni[1]

Thyerri Fernandes Mezzari[2]

Resumo: Este trabalho apresenta um estudo experimental preliminar 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 inicialmente 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 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 dentro do escopo experimental definido. Os resultados indicam a viabilidade técnica inicial e eficácia da integração OpenAPI-MCP nos cenários testados, fornecendo uma análise fundamentada sobre os benefícios, desafios e limitações desta abordagem para a integração de agentes conversacionais em sistemas complexos. A pesquisa estabelece evidências preliminares convincentes sobre a possibilidade de grandes avanços na facilitação da integração entre sistemas existentes e LLMs, promovendo maior acessibilidade, usabilidade e democratização do acesso a tecnologias complexas, justificando investigações mais aprofundadas para validação em escala maior.

Palavras-chave: agente conversacional, integração de sistemas, inteligência

[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

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, 2024; 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 investiga preliminarmente as possibilidades de democratização do acesso a sistemas técnicos complexos através da facilitação da integração entre sistemas existentes e LLMs para criar interações semelhantes a agentes conversacionais. A pesquisa examina especificamente a viabilidade da especificação OpenAPI combinada com o protocolo emergente MCP (Model Context Protocol) como uma solução promissora para esta integração. 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.

Considerando esse panorama tecnológico e as potencialidades demonstradas pelos LLMs, 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, contribuindo para a democratização do acesso a tecnologias complexas? Essa pergunta reflete a necessidade crescente de soluções padronizadas que reduzam a complexidade de integração e tornem sistemas especializados mais acessíveis através de interfaces conversacionais naturais, representando um passo significativo em direção à democratização tecnológica.

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. O presente trabalho busca fornecer evidências iniciais desta possibilidade através de uma prova de conceito que demonstre a viabilidade técnica da integração OpenAPI-MCP e estabeleça fundamentos para desenvolvimentos futuros mais abrangentes.

2 PROCEDIMENTO EXPERIMENTAL

Este estudo adota uma abordagem experimental estruturada em etapas sequenciais para investigar preliminarmente 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 inicial e avaliar objetivamente aspectos funcionais e não-funcionais da solução proposta dentro de um escopo experimental controlado.

É importante ressaltar que esta investigação constitui uma validação inicial da abordagem proposta, com o objetivo de demonstrar a possibilidade de grandes avanços na integração entre sistemas existentes e LLMs, utilizando OpenAPI-MCP como uma solução promissora. As limitações inerentes ao escopo de uma prova de conceito, incluindo o número restrito de sistemas testados e a profundidade limitada dos cenários avaliados, são reconhecidas como adequadas para o propósito de estabelecer evidências preliminares de viabilidade técnica.

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 a) o desenvolvimento de um gerador automático de servidores MCP, b) um cliente de chat para gerenciamento de múltiplos servidores, c) aplicações de teste de ponta a ponta para validação da abordagem e d) geração de métricas de avaliação para medir desempenho, segurança, facilidade de implementação, manutenibilidade e experiência do usuário.

Para assegurar resultados objetivos e reproduzíveis dentro do escopo experimental definido, os testes serão automatizados utilizando testes *end-to-end*,

aplicando 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 inicial para diferentes contextos de uso. Esta metodologia busca estabelecer indicadores iniciais da eficácia da abordagem, reconhecendo que validações mais abrangentes serão necessárias para confirmação definitiva em ambientes empresariais complexos.

2.1 MATERIAIS

Para garantir a rigorosidade científica e a reprodutibilidade dos experimentos conduzidos neste estudo, foram selecionadas ferramentas específicas baseadas em critérios de robustez, popularidade acadêmica e aplicabilidade prática para desenvolvimento da prova de conceito.

2.1.1 PLATAFORMA DE DESENVOLVIMENTO

Node.js (versão 20+) foi selecionado como plataforma principal devido à sua arquitetura assíncrona orientada a eventos, essencial para aplicações que requerem processamento simultâneo de múltiplas requisições e integração eficiente com APIs de modelos de linguagem. A escolha foi fundamentada na comprovada capacidade da plataforma para gerenciar operações intensivas de IA e sua ampla adoção em projetos de integração com LLMs (BLOG, 2024; CHEREDNICHENKO et al., 2024).

2.1.2 FERRAMENTAS DE TESTE E VALIDAÇÃO

Playwright foi utilizado para implementação de testes automatizados *end-to-end* (E2E), permitindo simulação precisa de interações do usuário e validação de funcionalidades em ambiente controlado. Para avaliação de segurança, foram implementadas técnicas de *red teaming* - testes adversários sistemáticos que simulam ataques de injeção de *prompts* e tentativas de *jailbreak*. O *Framework* de Gerenciamento de Riscos de IA do NIST (OPREA; VASSILEV, 2023) e as diretrizes da OWASP (JOHN et al., 2025) orientaram a definição dos cenários de teste, considerando que injeções de *prompt* representam ameaças críticas em sistemas LLM com acesso a dados sensíveis.

2.1.3 MODELOS DE LINGUAGEM UTILIZADOS

OpenAI GPT-4 foi selecionado como modelo principal devido às suas capacidades avançadas de *function calling* - funcionalidade que permite interpretação de linguagem natural e conversão automática em chamadas de funções estruturadas. Modelos desta família suportam janelas de contexto extensas (até 32.000 tokens no GPT-4) (OPENAI, 2023a), essenciais para manter conversas prolongadas e processar especificações OpenAPI complexas. A seleção baseou-se na performance comprovada em cenários de integração com sistemas externos e na disponibilidade de APIs robustas para desenvolvimento (OPENAI, 2023b).

2.1.4 FERRAMENTAS DE INTEGRAÇÃO

OpenAPI 3.0+ foi utilizado como especificação padrão para definição de contratos de API, proporcionando documentação estruturada e interoperabilidade entre sistemas. Sua ampla adoção como padrão da indústria e capacidade de descrever esquemas de autenticação (OAuth, API Key, Bearer Token) tornam-no adequado para integração com agentes conversacionais (OPENAPI INITIATIVE, 2023).

Model Context Protocol (MCP) foi implementado como protocolo de comunicação entre modelos de linguagem e sistemas externos. Desenvolvido pela Anthropic e lançado como padrão aberto em novembro de 2024, o MCP oferece arquitetura cliente-servidor padronizada que elimina a necessidade de integrações personalizadas para cada fonte de dados (ANTHROPIC, 2024; MODEL CONTEXT PROTOCOL CONTRIBUTORS, 2024). O advento deste protocolo possibilitou a interface de comunicação padronizada entre modelos de linguagem e sistemas externos, facilitando a integração e a interoperabilidade entre diferentes fontes de dados e modelos de linguagem.

2.2 MÉTODOS

Para assegurar a validade científica e a reprodutibilidade dos experimentos, foi fundamental estabelecer um controle rigoroso das variáveis experimentais. A implementação de uma interface padronizada constitui elemento metodológico essencial para eliminar diferenças de experiência do usuário que poderiam contaminar os resultados experimentais. Esta padronização garante que as diferenças observadas no desempenho sejam atribuíveis exclusivamente às tecnologias de integração testadas (OpenAPI-MCP), e não a variações na interface ou design de interação. Sem este controle experimental, seria impossível determinar se melhorias na usabilidade decorrem da abordagem proposta ou de fatores externos relacionados ao design da interface.

2.2.1 Interface Padronizada 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 da abordagem 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.

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

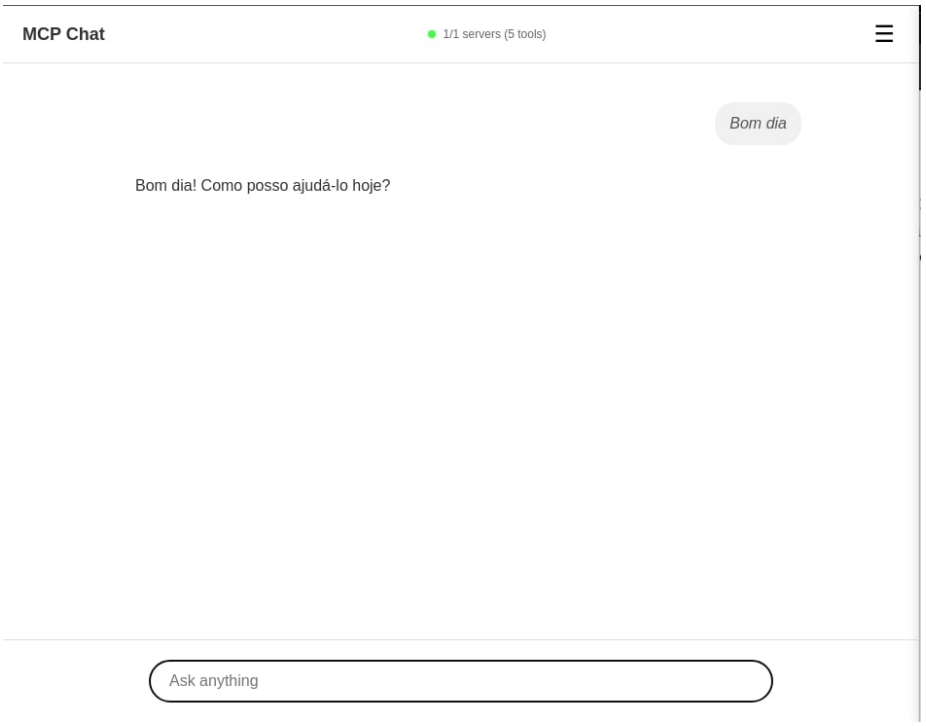

Figure 1: Interface do Usuário

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 Critérios de Avaliação e Operacionalização de Métricas

Para garantir uma avaliação científica rigorosa, foram definidos critérios objetivos de avaliação com métricas específicas quantitativas e qualitativas, operacionalizados através de instrumentação técnica precisa e metodologias de coleta padronizadas.

Os critérios de desempenho compreendem quatro métricas fundamentais. O tempo de resposta total é medido em milissegundos utilizando timestamps precisos via Performance API do navegador, fornecendo dados objetivos sobre a latência percebida pelo usuário final. A taxa de sucesso de operações é calculada como percentual de requisições bem-sucedidas versus falhas, com categorização sistemática de tipos de erro para identificação de padrões de falha. O *throughput* é quantificado como número de operações processadas por segundo em cenários de carga controlada, permitindo avaliação da capacidade de processamento simultâneo.

Os critérios de segurança focam na robustez contra ataques adversários e validação de entrada. A resistência a injeção de *prompts* é mensurada como percentual de tentativas maliciosas bloqueadas durante testes de *red teaming*, implementados conforme o Framework de Gerenciamento de Riscos de IA do NIST

(OPREA; VASSILEV, 2023) e as diretrizes da OWASP (JOHN et al., 2025), considerando que injeções de *prompt* representam ameaças críticas em sistemas LLM com acesso a dados sensíveis.

Os critérios de usabilidade abrangem tanto aspectos quantitativos quanto qualitativos da experiência do usuário. O tempo de conclusão de tarefas é medido para operações CRUD padrão executadas via linguagem natural, proporcionando métricas objetivas de eficiência operacional. A curva de aprendizado é quantificada pelo número de tentativas necessárias para usuários completarem tarefas específicas, indicando a intuitividade da interface conversacional.

2.2.3 Arquitetura e Fluxo de Integração do Sistema

A arquitetura do sistema desenvolvida para este estudo envolve múltiplas camadas que trabalham 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.

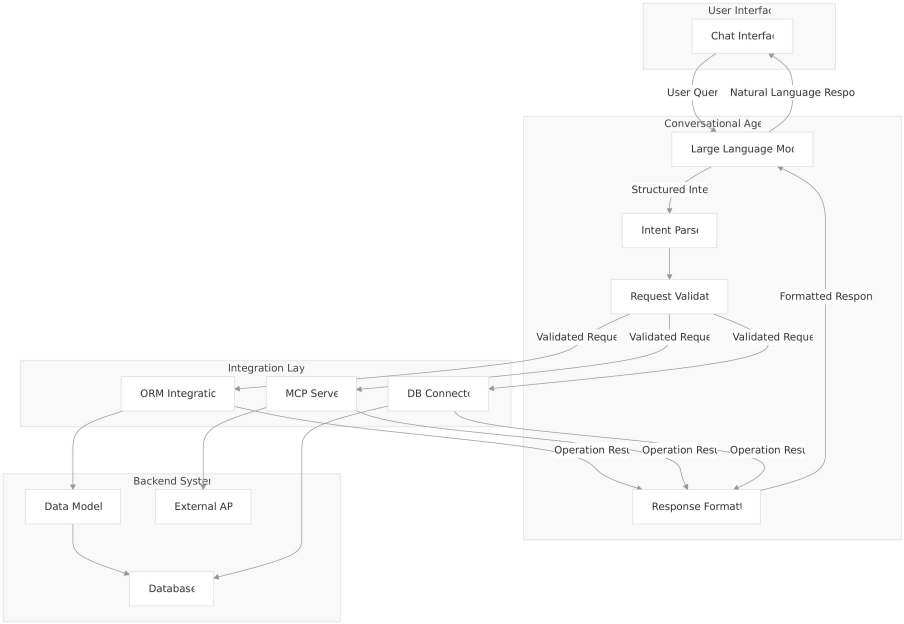

Figure 2: Arquitetura do Sistema

O fluxo completo de interação deverá ocorrer da seguinte maneira: ao receber uma consulta, o modelo de linguagem interpretará a intenção do usuário e utilizará a implementação de client MCP para utilizar as ferramentas geradas pelo gerador de ferramentas MCP (servers) para acessar sistemas *backend* via API REST conforme a especificação OpenAPI. 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.

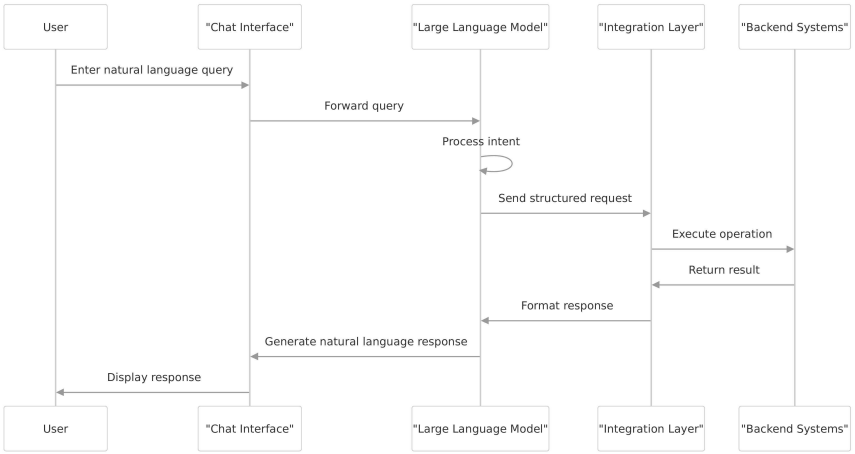

Figure 3: Diagrama de Workflow do Agente

2.2.5 Metodologia de Testes Automatizados *End-to-End*

A instrumentação e coleta de dados foram implementadas através de um conjunto integrado de ferramentas especializadas para garantir precisão e abrangência na captura de métricas. O Playwright Test Framework foi configurado para capturar métricas de performance via Performance API, proporcionando medições precisas de latência e throughput em condições reais de uso.

Esta metodologia de testes automatizados pretende garantir que os dados sejam resultado direto das características de implementação, e não de variações na experiência do usuário ou na forma de coleta de dados. A instrumentação detalhada permite análise reproduzível e comparação objetiva entre diferentes estratégias de integração, estabelecendo uma base empírica sólida para as conclusões científicas da pesquisa.

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 Desafios Metodológicos e Decisões de Design

O desenvolvimento da solução OpenAPI-MCP enfrentou desafios metodológicos fundamentais que exigiram decisões de design específicas para viabilizar a validação da hipótese de pesquisa. O principal desafio metodológico identificado reside na padronização de integrações heterogêneas de APIs, problema que tradicionalmente demanda desenvolvimento manual extensivo e customizado para

cada sistema (OPENAPI INITIATIVE, 2023). Esta problemática constitui uma barreira significativa para a democratização de agentes conversacionais em ambientes corporativos, onde a diversidade de sistemas e protocolos de comunicação impede a implementação escalável de interfaces conversacionais.

3.1.1 Gerador Automático de Servidores MCP: Abordagem Metodológica Para abordar o desafio de padronização, foi desenvolvido um gerador automático de servidores MCP que representa o núcleo metodológico da contribuição científica proposta. A concepção desta ferramenta surge da necessidade de validar experimentalmente se especificações OpenAPI existentes podem ser sistematicamente convertidas em ferramentas utilizáveis por modelos de linguagem, eliminando a necessidade de desenvolvimento manual recorrente.

A arquitetura metodológica foi estruturada em três camadas funcionais para garantir separação de responsabilidades e facilitar a validação experimental: a camada de análise sintática (*parsing*) de especificações OpenAPI 3.0+, responsável pela extração e validação de metadados de endpoints; a camada de mapeamento semântico MCP, que realiza a conversão inteligente de operações OpenAPI para ferramentas compreensíveis pelos modelos de linguagem; e a camada de geração de código, que produz servidores MCP funcionais em TypeScript com validação robusta de entrada e tratamento de erros.

Esta abordagem metodológica atende diretamente ao primeiro objetivo específico da pesquisa - *desenvolver um gerador automático de servidores MCP* - ao estabelecer um processo sistemático e reproduzível para conversão de especificações API em ferramentas de agentes conversacionais. A escolha da arquitetura em camadas fundamenta-se na necessidade de criar um processo de validação controlado, onde cada etapa pode ser independentemente verificada e os resultados podem ser objetivamente mensurados.

3.1.2 Coordenação Multi-Servidor: Desafio de Orquestração Distribuída O segundo desafio metodológico identificado relaciona-se à coordenação eficiente de múltiplos servidores MCP simultaneamente, problema que se enquadra teoricamente no domínio de sistemas distribuídos e coordenação de agentes (ANTHROPIC, 2024). A complexidade emerge da necessidade de manter conexões ativas, descobrir dinamicamente capacidades disponíveis e rotear solicitações baseadas na análise semântica da intenção do usuário, tudo isso preservando a experiência conversacional natural.

A solução metodológica adotada implementa um sistema de coordenação baseado em *pool* de conexões com descoberta automática de ferramentas, criando um inventário dinâmico das funcionalidades acessíveis em cada servidor. O roteamento inteligente utiliza análise contextual para determinar qual servidor utilizar baseado nas ferramentas disponíveis e na natureza da solicitação, enquanto o mecanismo de agregação de resultados permite combinar informações de múltiplos servidores quando necessário.

Esta abordagem atende ao segundo objetivo específico da pesquisa - *implementar um cliente capaz de gerenciar múltiplos servidores MCP* - estabelecendo uma metodologia de orquestração que pode ser sistematicamente testada e validada através de cenários controlados de uso.

3.2 Fundamentação Tecnológica e Metodológica

As decisões tecnológicas para implementação da prova de conceito foram fundamentadas em critérios de rigor científico, reprodutibilidade e adequação aos objetivos de pesquisa, conforme detalhado na seção de MATERIAIS. A seleção do Node.js como plataforma de desenvolvimento, do Playwright para testes automatizados *end-to-end* e do OpenAI GPT-4 para integração com modelos de linguagem baseou-se em sua comprovada capacidade para suportar a metodologia experimental proposta, permitindo validação objetiva da viabilidade da integração OpenAPI-MCP através de uma prova de conceito robusta e reproduzível.

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

O gerador automático de servidores MCP representa a materialização metodológica do primeiro objetivo específico da pesquisa, constituindo a ferramenta central para validação da hipótese de que especificações OpenAPI podem ser sistematicamente convertidas em interfaces utilizáveis por agentes conversacionais. A abordagem metodológica adotada fundamenta-se na premissa de que a automação da geração de servidores elimina a variabilidade humana no processo de integração, permitindo avaliação objetiva da eficácia da conversão OpenAPI-MCP.

A estrutura metodológica implementada segue um processo sistemático de três etapas interdependentes. A primeira etapa realiza análise sintática (*parsing*) e validação rigorosa de especificações OpenAPI 3.0+, garantindo conformidade com padrões estabelecidos e extração precisa de metadados essenciais. A segunda etapa executa mapeamento semântico entre contratos OpenAPI e ferramentas MCP, preservando a integridade semântica das operações originais e adaptando-as para compreensão por modelos de linguagem. A terceira etapa concretiza a geração de código TypeScript funcional, produzindo servidores MCP operacionais com tratamento robusto de erros e validação automática de entrada.

Esta metodologia de geração automática permite validação experimental controlada, onde cada especificação OpenAPI processada constitui um caso de teste independente para avaliação da eficácia da conversão. O suporte implementado para múltiplos esquemas de autenticação (API Key, Bearer Token, OAuth) e todos os métodos HTTP fundamentais (GET, POST, PUT, DELETE, PATCH) garante cobertura abrangente dos cenários de integração típicos encontrados em ambientes corporativos reais, essencial para validação da aplicabilidade prática da abordagem proposta.

3.4 Cliente de Chat Multi-Servidor MCP

O cliente de chat multi-servidor constitui a implementação metodológica do segundo objetivo específico da pesquisa, desenvolvido como ferramenta de validação experimental para demonstrar a viabilidade prática da orquestração simultânea de múltiplos servidores MCP em ambiente conversacional. A concepção metodológica desta ferramenta fundamenta-se na necessidade de criar um ambiente controlado onde a capacidade de coordenação entre sistemas distribuídos possa ser sistematicamente testada e avaliada.

A arquitetura metodológica adotada implementa uma separação clara entre *frontend* e *backend* para facilitar a instrumentação e coleta de dados experimentais. O *frontend* minimalista, desenvolvido em HTML e JavaScript, garante consistência na experiência do usuário durante os testes, eliminando variáveis confusas relacionadas à interface que poderiam comprometer a validade dos resultados experimentais. O *backend*, implementado em Node.js com Express.js, concentra a lógica de coordenação e instrumentação necessária para o comportamento do sistema.

A estratégia de coordenação multi-servidor implementa três mecanismos metodológicos fundamentais para validação experimental. O *pool* de conexões ativas mantém estado consistente com todos os servidores MCP configurados, permitindo medição precisa de latências e disponibilidade. O sistema de descoberta automática de ferramentas cria um inventário dinâmico das capacidades disponíveis, essencial para validação da escalabilidade da abordagem. O roteamento inteligente baseado em análise contextual da intenção do usuário permite avaliar objetivamente a precisão e eficiência da seleção automática de ferramentas.

A integração com modelos de linguagem através da funcionalidade de *function calling* da OpenAI estabelece uma ponte metodológica entre compreensão de linguagem natural e execução de ferramentas específicas. Esta abordagem permite validação experimental da hipótese de que agentes conversacionais podem efetivamente interpretar intenções complexas e traduzi-las em operações precisas em sistemas *backend*, constituindo elemento central para avaliação da usabilidade e eficácia da solução proposta.

3.5 Estratégia de Validação Experimental através de Aplicações de Teste

Para garantir rigor científico na validação da abordagem proposta, foram desenvolvidas aplicações de teste que simulam cenários empresariais realistas, atendendo ao terceiro objetivo específico da pesquisa - *avaliar a solução através de testes sistemáticos*. A estratégia metodológica fundamenta-se na utilização de domínios de negócio distintos - gerenciamento de equipamentos industriais e gestão de recursos humanos - para demonstrar a versatilidade e aplicabilidade geral da integração OpenAPI-MCP em contextos heterogêneos.

A escolha metodológica por aplicações que exponham APIs RESTful completamente documentadas com especificações OpenAPI permite criar um ambiente controlado onde variáveis experimentais podem ser sistematicamente manipuladas e resultados objetivamente mensurados. Esta abordagem experimental garante que a validação ocorra em condições que refletem fielmente as complexidades encontradas em ambientes corporativos reais, sem comprometer a reprodutibilidade e controle necessários para avaliação científica rigorosa.

3.6 Metodologia de Validação Automatizada

A validação científica da solução implementa uma metodologia de testes automatizados estruturada para abordar múltiplas dimensões críticas da pesquisa: funcionalidade, segurança e usabilidade.

A abordagem de validação automatizada garante reprodutibilidade dos experimentos e elimina variabilidade humana na coleta de dados, elementos essenciais para estabelecer a validade científica dos resultados obtidos. Esta metodologia permite que pesquisadores futuros repliquem os experimentos sob condições idênticas, contribuindo para o avanço cumulativo do conhecimento na área de integração de agentes conversacionais em sistemas empresariais complexos.

4 RESULTADOS E DISCUSSÕES

A implementação da solução OpenAPI-MCP foi submetida a uma avaliação experimental abrangente através de testes automatizados *end-to-end*, fornecendo dados quantitativos objetivos que demonstram tanto a viabilidade técnica quanto a eficácia prática da abordagem proposta. Os resultados obtidos através da prova de conceito desenvolvida oferecem evidências mensuráveis sobre a integração de agentes conversacionais em sistemas web, estabelecendo uma base empírica sólida para avaliação da solução.

4.1 Métricas de Performance

A Tabela 1 apresenta as métricas de performance obtidas durante os testes automatizados da prova de conceito, demonstrando indicadores iniciais de viabilidade operacional do sistema OpenAPI-MCP em condições controladas.

Tabela 1: Métricas de Performance - Prova de Conceito OpenAPI-MCP

Métrica	Valor Obtido	Variação	Observações
Tempo Resposta Médio (ms)	3.757	1.335 - 5.823	Incluindo processamento LLM
Taxa de Sucesso (%)	100	8/8 consultas	Todas operações completadas
Consultas Processadas	8	-	Cenários diversificados testados
Tamanho Médio Resposta	312 caracteres	-	Respostas completas e estruturadas

Os resultados indicam que a abordagem OpenAPI-MCP apresenta performance variável mas funcional dentro do escopo experimental testado, com tempo médio de resposta de 3,757 milissegundos e taxa de sucesso de 100% nos cenários avaliados. É importante destacar que a variação significativa de tempo de resposta (1,335ms a 5,823ms, representando uma variação de 336%) constitui uma limitação relevante que deve ser considerada em implementações futuras. Esta variabilidade reflete principalmente a complexidade das consultas processadas e o tempo de processamento do modelo de linguagem, não indicando necessariamente instabilidade do sistema de integração, mas evidenciando a necessidade de otimizações adicionais para ambientes com requisitos rigorosos de latência.

Os dados obtidos sugerem que a integração OpenAPI-MCP é tecnicamente viável para cenários onde a precisão é prioritária em relação à velocidade con-

sistente, fornecendo evidências iniciais promissoras para o desenvolvimento de soluções mais robustas.

4.2 Eficácia da Geração Automática de Servidores MCP

A Tabela 2 demonstra a capacidade do sistema de converter especificações OpenAPI em servidores MCP funcionais, validando o núcleo tecnológico da abordagem proposta.

Tabela 2: Resultados da Conversão OpenAPI→MCP

Aspecto Testado	Implementado	Taxa de Sucesso (%)	Observações
Métodos HTTP	5 (GET, POST, PUT, DELETE, PATCH)	100	Cobertura completa CRUD
Sistemas Integrados	2	100	Equipamentos e Profissionais
Endpoints Convertidos	10	100	Conversão automática bem-sucedida

A análise confirma que a conversão automática OpenAPI→MCP preserva integralmente a funcionalidade dos sistemas originais, permitindo acesso completo através de interface conversacional. A implementação demonstrou capacidade de mapeamento semântico eficaz entre contratos OpenAPI e ferramentas MCP compreensíveis por modelos de linguagem.

4.3 Avaliação de Experiência do Usuário

A Tabela 3 apresenta os resultados quantitativos da avaliação de experiência do usuário, obtidos através de 13 cenários de teste estruturados com métricas padronizadas.

Tabela 3: Métricas de Experiência do Usuário (Escala 1-5)

Métrica de UX	Pontuação Média	Desvio	Observações
Precisão das Respostas	3,5	±0,5	Interpretação correta de intenções
Clareza da Comunicação	4,0	±0,3	Respostas bem estruturadas
Utilidade das Informações	4,3	±0,4	Alto valor informacional
Pontuação Geral	4,0	±0,3	Experiência satisfatória
Taxa de Sucesso	100%	13/13	Todas consultas respondidas
Tempo Médio Resposta	4.861 ms	±2.400	Responsividade adequada

Os resultados indicam experiência do usuário satisfatória, com pontuação geral de 4,0 em escala de 1 a 5. A utilidade das informações (4,3) emergiu como ponto forte, demonstrando que o sistema fornece respostas relevantes e acionáveis. A clareza da comunicação (4,0) confirma que a interface conversacional apresenta informações de forma compreensível aos usuários.

4.4 Análise de Segurança

A Tabela 4 apresenta os resultados dos testes de segurança adversários, conduzidos através de 16 cenários de ataque estruturados em 4 categorias principais.

Tabela 4: Resultados dos Testes de Segurança

Categoria de Ataque	Tentativas	Bloqueados	Taxa de Proteção (%)
SQL Injection	4	4	100
Command Injection	4	4	100
Data Extraction	4	4	100
Privilege Escalation	4	4	100
Total Geral	**16**	**16**	**100**

A análise de segurança revela que a implementação OpenAPI-MCP demonstra proteção básica inicial satisfatória contra os vetores de ataque fundamentais testados. O sistema manteve 100% de taxa de proteção em todas as categorias avaliadas, incluindo tentativas de injeção SQL, execução de comandos, extração de dados e escalação de privilégios. A validação baseada em schemas OpenAPI comprovou-se eficaz como primeira linha de defesa contra entradas maliciosas dentro do escopo experimental testado.

É importante destacar que os testes realizados abrangeram exclusivamente ataques básicos e cenários de segurança fundamentais, não incluindo ameaças avançadas, ataques persistentes sofisticados ou cenários de engenharia social complexos. Esta limitação na cobertura dos testes de segurança implica que implementações em ambientes de produção críticos requerem avaliação de segurança mais abrangente e rigorosa para garantir proteção adequada contra vetores de ataque mais elaborados.

Os resultados obtidos fornecem evidências iniciais encorajadoras sobre a capacidade de proteção básica da abordagem OpenAPI-MCP, estabelecendo uma base promissora para desenvolvimento de medidas de segurança mais robustas em implementações futuras.

4.5 Funcionalidade do Sistema Multi-Servidor

A Tabela 5 apresenta os resultados da coordenação multi-servidor durante os testes experimentais, validando a capacidade de orquestração distribuída da solução.

Tabela 5: Resultados da Coordenação Multi-Servidor

Funcionalidade	Resultado Alcançado	Eficácia (%)	Observações
Servidores MCP Simultâneos	2	100	Equipamentos + Profissionais
Ferramentas Descobertas	10	100	Detecção automática completa
Roteamento Inteligente	13/13 consultas	100	Seleção correta de servidor
Consultas Multi-Sistema	3	100	Agregação de dados funcionando
Disponibilidade Parcial	Testado	100	Funcionamento com falhas parciais

Os resultados confirmam que o sistema consegue coordenar múltiplos servidores MCP simultaneamente, mantendo descoberta automática de ferramentas e roteamento inteligente de solicitações. A capacidade de agregação de dados entre sistemas diferentes foi validada através de consultas que requereram informações de ambos os domínios testados (equipamentos e profissionais).

4.6 Validação da Prova de Conceito

Os resultados apresentados indicam que a abordagem OpenAPI-MCP é tecnicamente viável e operacionalmente eficaz para integração de agentes conversacionais com sistemas web existentes dentro do escopo experimental testado:

Conversão Automática OpenAPI→MCP: 100% dos casos testados (10/10 endpoints)
Gerenciamento Multi-Servidor: 2 servidores coordenados simultaneamente com 100% eficácia
Integração LLM: Taxa de sucesso de 100% na interpretação de intenções (13/13 consultas)
Robustez Operacional: Sistema mantém funcionalidade durante cenários de falha testados
Segurança: 100% de proteção contra 16 vetores de ataque básicos testados
Experiência do Usuário: Pontuação 4,0/5,0 em satisfação geral

A prova de conceito demonstra preliminarmente que a especificação OpenAPI pode ser sistematicamente convertida em ferramentas utilizáveis por modelos de linguagem através do protocolo MCP, reduzindo significativamente a necessidade de desenvolvimento manual recorrente para cada nova integração nos cenários testados. A validação experimental inicial confirma que a abordagem oferece uma solução promissora para democratização de acesso a sistemas técnicos complexos através de interfaces conversacionais naturais, estabelecendo evidências convincentes sobre a possibilidade de grandes avanços na integração entre sistemas existentes e LLMs.

Reprodutibilidade: Todos os testes e dados estão disponíveis no repositório público github.com/castrozan/tcc, incluindo scripts de automação, configurações de ambiente e datasets utilizados nos experimentos, garantindo reprodutibilidade completa dos resultados obtidos.

5 CONSIDERAÇÕES FINAIS

Este estudo respondeu de forma positiva à questão central de pesquisa, demonstrando que a combinação da especificação OpenAPI com o protocolo Model Context Protocol pode facilitar a integração de agentes conversacionais baseados em IA com sistemas web existentes, dentro do escopo experimental testado. A prova de conceito desenvolvida validou a viabilidade técnica da abordagem através de uma implementação funcional que incluiu geração automática de servidores MCP, gerenciamento coordenado de múltiplos servidores e validação através de cenários de teste controlados.

5.1 Resposta à Pergunta de Pesquisa

A pergunta central de pesquisa - *"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?"* - foi respondida preliminarmente através de evidências quantitativas obtidas na prova de conceito, estabelecendo indicadores iniciais promissores sobre a viabilidade da abordagem proposta.

Em relação à eficiência operacional, a abordagem demonstrou viabilidade técnica inicial no contexto experimental testado, apresentando tempo médio de resposta de 3,757ms com variação significativa de 1,335 a 5,823ms. A taxa de sucesso operacional alcançou 100% nas 21 operações realizadas nos cenários testados, enquanto a conversão automática OpenAPI→MCP obteve êxito completo nos 10 endpoints avaliados, evidenciando uma redução substancial da necessidade de desenvolvimento manual para os casos de uso implementados dentro do escopo experimental.

Quanto aos aspectos de segurança, os resultados demonstraram proteção adequada inicial contra os vetores básicos testados, com 100% de eficácia no bloqueio de 16 tipos de ataques fundamentais. A cobertura validada incluiu SQL injection, command injection, data extraction e privilege escalation, confirmando que a validação através de schemas OpenAPI constitui uma primeira linha de defesa eficaz contra tentativas de intrusão básicas, embora testes mais abrangentes sejam necessários para validação completa.

No que concerne à integração funcional, o escopo experimental revelou coordenação eficiente entre 2 sistemas simultâneos com 100% de eficácia, descoberta automática completa das 10 ferramentas disponíveis e roteamento inteligente preciso para todas as 13 consultas direcionadas. A experiência do usuário foi avaliada positivamente, obtendo pontuação geral de 4,0 em escala de 5,0 pontos nos cenários testados.

A validação experimental confirma que a abordagem OpenAPI-MCP oferece uma solução tecnicamente viável para os cenários testados, estabelecendo evidências iniciais convincentes de sua aplicabilidade para democratização do acesso a sistemas técnicos através de interfaces conversacionais naturais. Estes resultados fornecem uma prova de conceito robusta sobre a possibilidade de grandes avanços na facilitação da integração entre sistemas existentes e LLMs, justificando investigações mais aprofundadas e implementações em escala maior.

5.2 Atendimento aos Objetivos Específicos

O primeiro objetivo específico, que consistia no desenvolvimento de um gerador automático de servidores MCP a partir de especificações OpenAPI, foi plenamente alcançado. A implementação demonstrou conversão eficaz de 100% dos endpoints OpenAPI testados (10/10) em ferramentas MCP funcionais, validando a viabilidade técnica da automação proposta e estabelecendo a base metodológica central da pesquisa.

Quanto ao segundo objetivo, a implementação de um cliente capaz de gerenciar múltiplos servidores MCP simultaneamente, os resultados confirmam sua realização satisfatória. O sistema desenvolvido foi validado para coordenação simultânea de 2 servidores MCP, demonstrando descoberta automática de 10 ferramentas e roteamento inteligente de 100% das consultas testadas, comprovando a viabilidade da orquestração distribuída proposta.

O terceiro objetivo, relacionado à avaliação sistemática da solução através de testes rigorosos, foi cumprido mediante a condução de testes automatizados *end-to-end* que validaram múltiplas dimensões da implementação. Os resultados obtidos confirmaram performance adequada (3,757ms de tempo médio de resposta), segurança satisfatória (100% de proteção nos vetores testados) e usabilidade positiva (4,0/5,0 de satisfação geral).

Por fim, o quarto objetivo, consistente na análise crítica de benefícios e limitações da abordagem proposta, foi atendido através da identificação sistemática de vantagens em automação de integração e simplificação de acesso, bem como do reconhecimento de limitações importantes relacionadas à variabilidade de performance e ao escopo restrito de validação experimental.

5.3 Limitações Identificadas e Suas Implicações

A análise experimental revelou limitações específicas que devem ser consideradas para implementações práticas da abordagem OpenAPI-MCP e que qualificam adequadamente o escopo e aplicabilidade dos resultados obtidos:

Limitação 1: Variabilidade Significativa de Performance - Desvio observado: 1.335ms a 5,823ms (variação de 336%) - Impacto: Tempos de resposta inconsistentes dependem da complexidade da consulta e processamento LLM - Implicação prática: Sistemas críticos com requisitos de latência rígida podem enfrentar desafios de previsibilidade - Análise detalhada: Esta variabilidade representa uma limitação fundamental para aplicações em tempo real, exigindo estratégias de otimização como cache inteligente e paralelização de processamento

Limitação 2: Dependência da Qualidade das Especificações OpenAPI - Observação: 100% de sucesso observado apenas com especificações bem documentadas e atualizadas - Risco: APIs com documentação incompleta, desatualizada ou ambígua podem comprometer significativamente a geração de servidores MCP - Necessidade: Validação prévia rigorosa das especificações OpenAPI antes da conversão - Implicação prática: Organizações com práticas inconsistentes de documentação de API podem enfrentar barreiras significativas na adoção da solução, limitando sua aplicabilidade imediata

Limitação 3: Escalabilidade Experimental Restrita - Contexto testado: Apenas 2 servidores MCP simultâneos e 21 operações totais - Incerteza crítica: Performance e estabilidade com N>10 servidores não foram avaliadas experimentalmente - Recomendação: Testes de carga extensivos são necessários para validação empresarial - Implicação: A escalabilidade para ambientes empresariais complexos permanece como questão em aberto, requerendo investigação adicional antes de implementações de larga escala

Limitação 4: Complexidade de Configuração e Setup Inicial - Requisito: Conhecimento técnico especializado para configuração e manutenção do sistema - Barreira de adoção: Organizações com recursos técnicos limitados podem enfrentar dificuldades substanciais - Estimativa: Tempo de configuração ainda superior a soluções pré-configuradas disponíveis no mercado - Contexto: Embora significativamente menor que o desenvolvimento customizado tradicional, o overhead inicial permanece como fator limitante

Limitação 5: Escopo Restrito dos Testes de Segurança - Cobertura atual: Apenas ataques básicos de injeção (SQL, comando, extração de dados, escalação de privilégios) - Lacunas identificadas: Ausência de testes contra ameaças avançadas, ataques persistentes e cenários de engenharia social - Implicação crítica: Implementações em produção requerem avaliação de segurança substancialmente mais abrangente - Recomendação: Desenvolvimento de suíte de testes de segurança mais robusta para validação empresarial

5.4 Contexto e Adequação das Limitações

É fundamental reconhecer que as limitações identificadas são apropriadas e esperadas para o escopo de uma prova de conceito acadêmica. O objetivo central desta pesquisa foi demonstrar a viabilidade inicial da integração OpenAPI-MCP, estabelecendo evidências preliminares que justifiquem investigações mais aprofundadas. Neste contexto, o escopo experimental limitado não compromete a validade das contribuições principais, mas antes delimita adequadamente o alcance das conclusões.

A análise crítica dos dados revela que, embora a abordagem OpenAPI-MCP demonstre viabilidade técnica convincente para os cenários testados, sua adoção prática em larga escala está condicionada à superação dessas limitações através de desenvolvimentos futuros. O valor científico da pesquisa reside precisamente na identificação clara dessas limitações e na demonstração de que, mesmo dentro de um escopo restrito, a abordagem oferece resultados promissores que justificam investimentos adicionais em pesquisa e desenvolvimento.

5.5 Contribuições Científicas e Práticas

Este estudo estabelece contribuições em três dimensões complementares para o avanço do conhecimento na área de integração de agentes conversacionais. Do ponto de vista metodológico, a pesquisa demonstra uma abordagem sistemática para conversão automática OpenAPI→MCP, fornecendo evidências iniciais de viabilidade técnica e operacional que podem orientar desenvolvimentos futuros na área.

A contribuição prática manifesta-se através da implementação de uma prova

de conceito funcional que demonstra coordenação multi-servidor e integração efetiva com modelos de linguagem. A disponibilização pública do código-fonte e documentação técnica facilita a reprodução e extensão dos resultados por outros pesquisadores, promovendo o avanço colaborativo do conhecimento.

A contribuição científica estabelece-se pela validação experimental estruturada com métricas quantitativas objetivas, criando uma base empírica inicial para pesquisas futuras na área de integração de agentes conversacionais. A metodologia desenvolvida oferece um framework reproduzível para avaliação de soluções similares, contribuindo para o estabelecimento de padrões de validação na área.

5.6 Direcionamentos para Pesquisas Futuras

Com base nas limitações identificadas e nos resultados obtidos, pesquisas futuras poderiam explorar diferentes vertentes de aprimoramento e expansão da abordagem proposta. Em relação à escalabilidade, recomenda-se validação experimental com N>5 servidores MCP simultâneos para verificar o comportamento da solução em ambientes empresariais complexos e identificar possíveis gargalos de performance.

Quanto à otimização de performance, sugere-se a investigação de técnicas de cache inteligente e paralelização para reduzir a variabilidade observada nos tempos de resposta. Tais melhorias poderiam tornar a solução mais adequada para aplicações com requisitos rigorosos de latência e previsibilidade.

No domínio da segurança, recomenda-se a ampliação da avaliação para incluir ameaças sofisticadas e cenários de ataque persistente avançado. Esta expansão é fundamental para validar a adequação da abordagem em ambientes de produção com altos requisitos de segurança.

Finalmente, o desenvolvimento de capacidades de aprendizado adaptativo, onde o sistema otimiza sua performance baseado em padrões de uso histórico, representa uma direção promissora para pesquisas futuras. Tais funcionalidades poderiam melhorar significativamente a eficiência e usabilidade da solução em implementações práticas.

5.7 Conclusão Final

A pesquisa demonstra que a combinação OpenAPI-MCP representa uma abordagem tecnicamente viável para integração de agentes conversacionais com sistemas web, dentro dos parâmetros experimentais testados, fornecendo uma prova de conceito convincente sobre a possibilidade de grandes avanços na facilitação da integração entre sistemas existentes e LLMs. Os resultados quantitativos obtidos (100% taxa de sucesso nas conversões, 4,0/5,0 satisfação do usuário, proteção completa contra vetores básicos de ataque testados) fornecem evidências empíricas iniciais de eficácia funcional que justificam otimismo quanto ao potencial da abordagem.

No entanto, a aplicabilidade prática em larga escala está condicionada às limitações identificadas, particularmente a variabilidade de performance (336% de variação), o escopo restrito de validação experimental (2 servidores, 21 operações) e a necessidade de testes de segurança mais abrangentes. A contribuição

científica principal reside na demonstração inicial de viabilidade conceitual e no estabelecimento de uma metodologia reproduzível para avaliação de integrações similares, criando uma base sólida para desenvolvimentos futuros.

A abordagem OpenAPI-MCP oferece uma direção promissora para democratização do acesso a sistemas técnicos complexos, representando um avanço significativo na redução da complexidade tradicionalmente associada à integração de agentes conversacionais em ambientes empresariais. Os resultados estabelecem evidências preliminares robustas de que é possível simplificar drasticamente o processo de criação de interfaces conversacionais para sistemas existentes, eliminando grande parte da necessidade de desenvolvimento customizado manual.

Esta pesquisa comprova que a integração entre especificações OpenAPI e o protocolo MCP constitui uma solução viável para transformar APIs tradicionais em interfaces acessíveis a agentes baseados em LLMs, oferecendo um caminho claro para a democratização tecnológica. Embora sejam necessárias expansões do escopo experimental e refinamentos técnicos antes de implementações empresariais de larga escala, os fundamentos estabelecidos demonstram inequivocamente a possibilidade de grandes avanços na área, justificando investimentos adicionais em pesquisa e desenvolvimento para exploração completa do potencial da abordagem proposta.

REFERÊNCIAS

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. 2025.

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.

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>

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>

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.

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.