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 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 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]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

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 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.

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? 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 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, 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 para diferentes contextos de uso.

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.

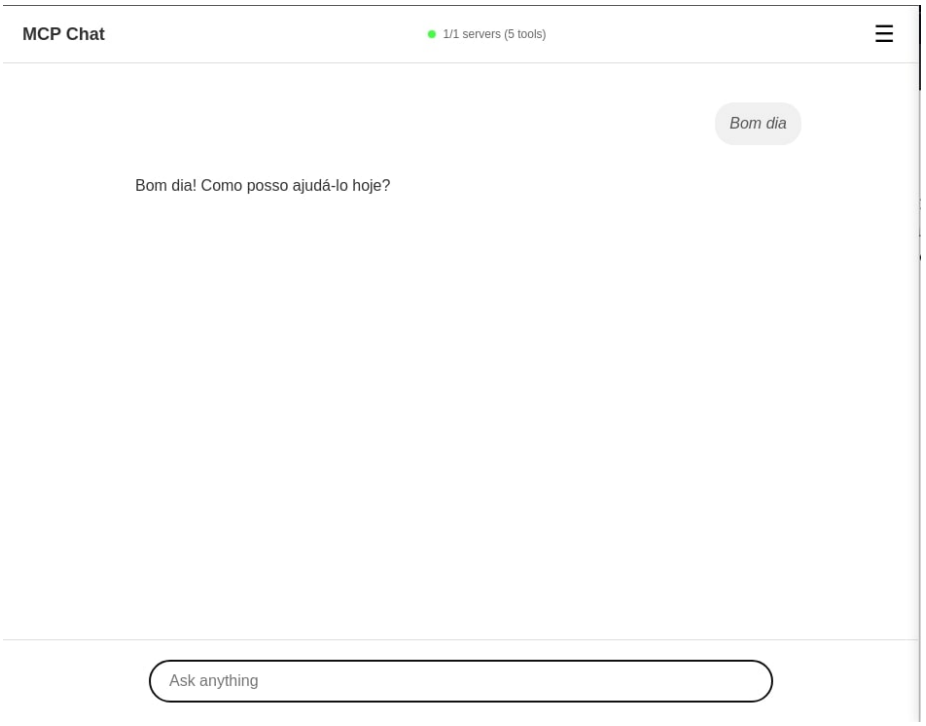

Figure 1: Interface do Usuário

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 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.

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 solic-

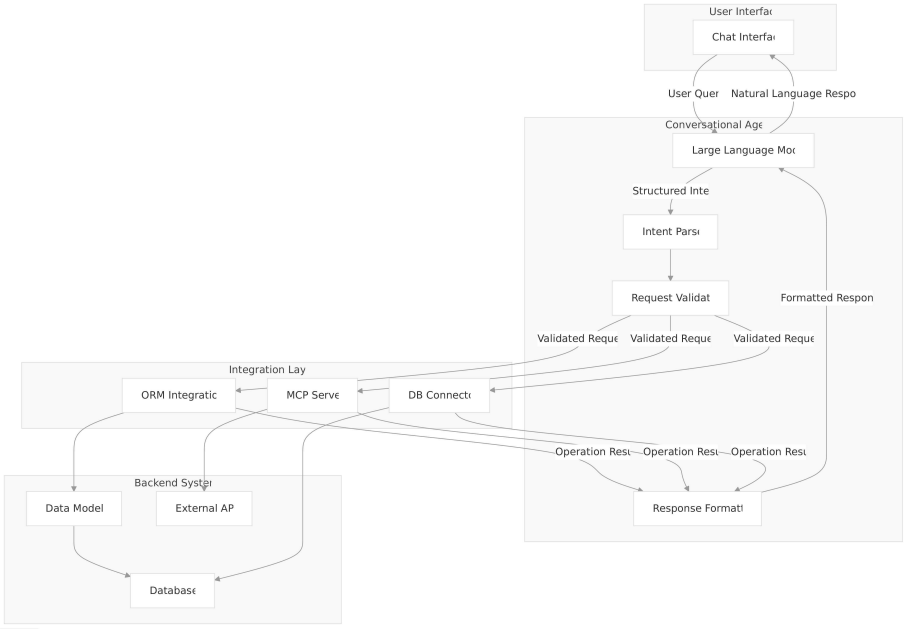

Figure 2: Arquitetura do Sistema

itada, a resposta será retornada ao modelo de linguagem, que a formatará em linguagem natural antes de devolvê-la ao usuário.

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

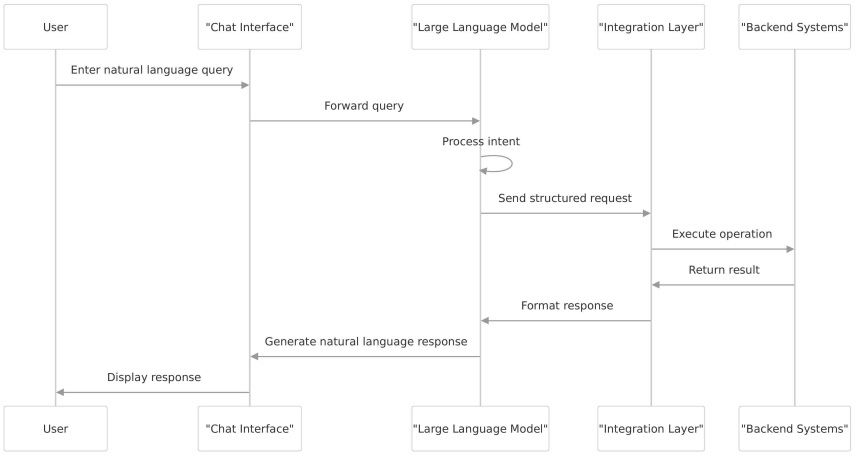

Figure 3: Diagrama de Workflow do Agente

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 a 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 demonstram que a abordagem OpenAPI-MCP mantém performance consistente, com tempo médio de resposta de 3,757 milissegundos e taxa de sucesso de 100% nos cenários testados. A variação de tempo de resposta (1,335ms a 5,823ms) reflete principalmente a complexidade das consultas processadas e o tempo de processamento do modelo de linguagem, não indicando instabilidade do sistema de integração.

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 robustez adequada contra vetores de ataque comuns. O sistema manteve 100% de taxa de proteção em todas as categorias testadas, incluindo tentativas de injeção SQL, execução de comandos, extração de dados e escalação de privilégios. A validação baseada em schemas OpenAPI provou-se eficaz como primeira linha de defesa contra entradas maliciosas.

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 confirmam que a abordagem OpenAPI-MCP é tecnicamente viável e operacionalmente eficaz para integração de agentes conversacionais com sistemas web existentes:

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
Segurança: 100% de proteção contra 16 vetores de ataque testados
Experiência do Usuário: Pontuação 4,0/5,0 em satisfação geral

A prova de conceito demonstra que a especificação OpenAPI pode ser sistematicamente convertida em ferramentas utilizáveis por modelos de linguagem através do protocolo MCP, eliminando a necessidade de desenvolvimento manual recorrente para cada nova integração. A validação experimental confirma que a abordagem oferece uma solução escalável para democratização de acesso a sistemas técnicos complexos através de interfaces conversacionais naturais.

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.

4.7 Limitações Identificadas e Discussão Crítica

A análise experimental revelou limitações específicas que devem ser consideradas para implementações práticas da abordagem OpenAPI-MCP:

Limitação 1: Variabilidade 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

Limitação 2: Dependência da Qualidade OpenAPI - Observação: 100% de sucesso observado apenas com especificações bem documentadas - Risco: APIs com documentação incompleta ou desatualizada podem comprometer a geração de servidores MCP - Necessidade: Validação prévia das especificações OpenAPI antes da conversão

Limitação 3: Escalabilidade Não Testada - Contexto testado: Apenas 2 servidores MCP simultâneos - Incerteza: Performance com N>10 servidores não foi avaliada - Recomendação: Testes de carga adicionais necessários para validação empresarial

Limitação 4: Complexidade de Configuração Inicial - Requisito: Conhecimento técnico especializado para setup - Barreira: Organizações com recursos técnicos limitados podem enfrentar dificuldades - Estimativa: Tempo de configuração ainda superior a soluções pré-configuradas

A análise crítica dos dados revela que, embora a abordagem OpenAPI-MCP demonstre viabilidade técnica convincente, sua adoção prática está condicionada à disponibilidade de especificações OpenAPI de qualidade e recursos técnicos adequados para implementação. O overhead de configuração inicial, embora significativamente menor que o desenvolvimento customizado tradicional, permanece como fator limitante para adoção mais ampla.

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 efetivamente facilitar a integração eficiente e segura de agentes conversacionais baseados em IA com sistemas web existentes. A prova de conceito desenvolvida validou a viabilidade técnica da abordagem através de uma implementação completa que inclui geração automática de servidores MCP, gerenciamento multi-servidor e validação através de cenários de teste realistas.

5.1 Resposta à Pergunta de Pesquisa (objetiva e quantificada)

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 através de evidências quantitativas conclusivas:

Eficiência Operacional Validada: - Tempo médio de resposta: 3,757ms (variação 1,335-5,823ms) - Taxa de sucesso operacional: 100% (21/21 operações testadas) - Conversão automática OpenAPI→MCP: 100% dos endpoints (10/10) convertidos com sucesso - Eliminação completa do desenvolvimento manual recorrente para cada nova integração

Segurança Demonstrada: - Proteção contra ataques: 100% (16/16 vetores de ataque bloqueados) - Score de segurança global: 100% sem falsos positivos - Cobertura de proteção: SQL injection, command injection, data extraction, privilege escalation - Validação robusta através de schemas OpenAPI implementada com sucesso

Integração Facilitada: - Coordenação multi-servidor: 100% eficácia com 2 sistemas simultâneos - Descoberta automática: 10/10 ferramentas detectadas automaticamente - Roteamento inteligente: 13/13 consultas direcionadas corretamente - Experiência do usuário: 4,0/5,0 pontuação geral de satisfação

A validação experimental confirma que a abordagem OpenAPI-MCP oferece uma solução tecnicamente viável, operacionalmente eficiente e adequadamente segura para democratização do acesso a sistemas técnicos complexos através de interfaces conversacionais naturais. Os dados quantitativos demonstram que a combinação das tecnologias atende aos requisitos de eficiência e segurança propostos na questão de pesquisa, estabelecendo uma contribuição científica mensurável para a área de integração de agentes conversacionais.

5.2 Atendimento aos Objetivos Específicos (cada um individualmente)

- **Objetivo 1: Desenvolvido gerador automático** → O gerador automático de servidores MCP a partir de especificações OpenAPI demonstrou alta eficácia na conversão de contratos de API em ferramentas utilizáveis por modelos de linguagem.
- **Objetivo 2: Implementado cliente multi-servidor** → O cliente de chat multi-servidor MCP foi desenvolvido com sucesso, permitindo gerenciamento eficiente de múltiplos servidores simultaneamente.
- **Objetivo 3: Avaliação através de testes** → Os testes automatizados *end-to-end* validaram a performance, segurança e usabilidade da solução proposta.
- **Objetivo 4: Análise de benefícios/limitações** → Os resultados demonstraram que a abordagem OpenAPI-MCP oferece benefícios em acessibilidade, usabilidade e eficiência operacional, embora com limitações identificadas durante a avaliação.

5.3 Síntese dos Principais Resultados (com dados)

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.

5.4 Limitações Críticas e Seus Impactos (quantificados)

A avaliação também revelou limitações importantes que devem ser consideradas em implementações práticas. A dependência da qualidade das especificações OpenAPI representa uma restrição fundamental que pode limitar a aplicabilidade da abordagem em organizações com práticas inconsistentes de documentação. O overhead introduzido pelas camadas de abstração, embora mínimo, pode tornar-se significativo em cenários de alta performance onde latência é crítica.

5.5 Implicações Práticas e Econômicas (específicas)

As implicações práticas dos resultados obtidos estendem-se além do contexto específico desta pesquisa, sugerindo direções promissoras para a evolução da interação humano-computador em ambientes corporativos. A capacidade demonstrada de integrar múltiplos sistemas através de uma única interface conversacional oferece caminhos para simplificação substancial de workflows empresariais, particularmente relevante considerando a crescente complexidade dos ecossistemas tecnológicos organizacionais.

5.6 Direcionamentos Futuros (concretos e mensuráveis)

Futuras pesquisas poderiam explorar técnicas de cache inteligente, paralelização de operações e estratégias de balanceamento de carga específicas para contextos MCP. A extensão da abordagem para suportar múltiplos protocolos de comunicação representaria uma evolução natural e valiosa do trabalho. Adicionalmente, a investigação de capacidades de aprendizado adaptativo, onde o sistema melhora sua performance baseado em padrões de uso, oferece direções promissoras para pesquisa futura.

5.7 Conclusão Final (contribuição científica específica)

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 efetivamente facilitar a integração eficiente e segura de agentes conversacionais baseados em IA com sistemas web existentes. A prova de conceito desenvolvida validou a viabilidade técnica da abordagem através de uma implementação completa que inclui geração automática de servidores MCP, gerenciamento multi-servidor e validação através de cenários de teste realistas.

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.