CAPÍTULO 35

A IMPLEMENTAÇÃO DE ARQUITETURAS AGÊNTICAS SERVERLESS: INTEGRAÇÃO ENTRE APPS SCRIPT, SHEETS E COLAB SOB OS PROTOCOLOS MCP, A2A, ANP E ACP

Hélio Craveiro Pessoa Júnior[1].

[1]Universidade de Brasília, Programa de Pós-Graduação em Educação – Modalidade Profissional, Brasília, Distrito Federal.

https://lattes.cnpq.br/1415349950533370

RESUMO: Este capítulo revisa a literatura técnica e científica sobre a implementação de arquiteturas agênticas serverless apoiadas no ecossistema Google Workspace, com ênfase na integração entre Google Apps Script, Google Sheets e Google Colab. O estudo parte do problema de criar sistemas CRUD e fluxos analíticos de baixo custo, sem abrir mão de interoperabilidade, rastreabilidade e capacidade de coordenação entre agentes. A revisão articula três eixos: computação serverless e suas limitações operacionais; agentes baseados em modelos de linguagem, uso de ferramentas e sistemas multiagentes; e protocolos emergentes de interoperabilidade, especialmente Model Context Protocol (MCP), Agent2Agent (A2A), Agent Network Protocol (ANP) e Agent Communication Protocol (ACP). Conclui-se que Apps Script, Sheets e Colab podem formar uma arquitetura aplicada viável para protótipos, pesquisas e sistemas institucionais leves, desde que sejam observados limites de cota, segurança, governança de dados e padronização semântica das interfaces.

PALAVRAS-CHAVE: Google Apps Script. Google Colab. Protocolos Agênticos.

THE IMPLEMENTATION OF SERVERLESS AGENTIC ARCHITECTURES: INTEGRATION BETWEEN APPS SCRIPT, SHEETS AND COLAB UNDER MCP, A2A, ANP AND ACP PROTOCOLS

ABSTRACT: This chapter reviews technical and scientific literature on serverless agentic architectures supported by the Google Workspace ecosystem, focusing on the integration of Google Apps Script, Google Sheets and Google Colab. The study addresses the challenge of building low-cost CRUD systems and analytical workflows while preserving interoperability, traceability and coordination among agents. The review connects three axes: serverless computing and its operational constraints; language-model-based agents, tool use and multi-agent systems; and emerging interoperability protocols, especially Model Context Protocol (MCP), Agent2Agent (A2A), Agent Network Protocol (ANP) and Agent Communication

Protocol (ACP). The chapter concludes that Apps Script, Sheets and Colab can support a viable applied architecture for prototypes, research workflows and lightweight institutional systems, provided that quota limits, security, data governance and semantic standardization of interfaces are explicitly considered.

KEY-WORDS: Google Apps Script. Google Colab. Agent Protocols.

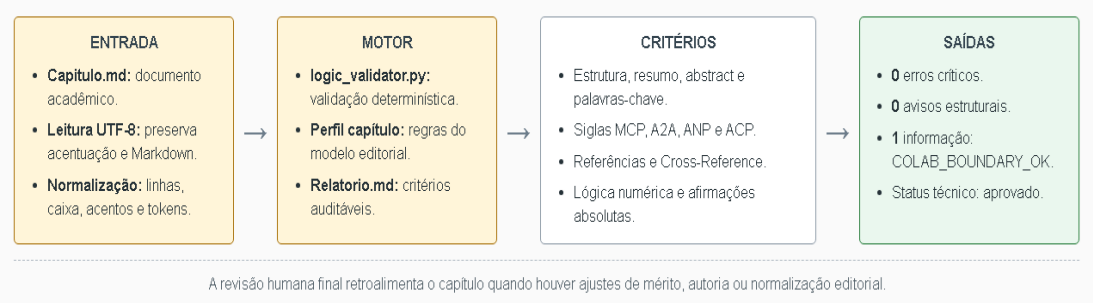
INTRODUÇÃO

A evolução recente da inteligência artificial deslocou parte importante do debate técnico dos modelos isolados para sistemas capazes de perceber contexto, acionar ferramentas, delegar tarefas e registrar resultados. Nesse cenário, o agente não é apenas uma interface conversacional; ele passa a ser uma unidade operacional que combina raciocínio, memória, planejamento, uso de APIs e comunicação com outros agentes ou serviços. Essa mudança exige arquiteturas que consigam integrar dados, automações e processamento analítico sem impor custos elevados de infraestrutura.

A computação serverless aparece na literatura como resposta parcial a esse desafio, pois transfere ao provedor a administração de servidores, escalonamento, disponibilidade e parte da operação da plataforma. Baldini et al. (2017) apontam que o modelo serverless se consolidou pela combinação entre execução sob demanda, orientação a eventos e redução do esforço de administração. Jonas et al. (2019), por sua vez, destacam que a promessa do serverless é simplificar a programação em nuvem, embora persistam dificuldades relacionadas a estado, latência, observabilidade e coordenação de tarefas longas.

Essa organização conceitual da arquitetura proposta pode ser visualizada na Figura 1, que sintetiza as principais camadas e fluxos entre os componentes Apps Script, Sheets e Colab.

Figura 1: Arquitetura Conceitual da Validação Lógica do Capítulo



Fonte: Elaborado pelo autor.

No contexto deste capítulo, o termo serverless é empregado em sentido aplicado: não se trata apenas de Function-as-a-Service em nuvens comerciais, mas do uso de serviços gerenciados que abstraem servidores para o usuário final. Google Apps Script, Google Sheets e Google Colab permitem compor uma arquitetura de baixo custo na qual o Sheets funciona como camada de persistência, o Apps Script como camada de automação e exposição HTTP, e o Colab como ambiente analítico baseado em notebooks Python. A relevância científica da proposta está em investigar se essa composição, normalmente vista como solução de produtividade, pode ser reinterpretada como base para sistemas agênticos interoperáveis.

A literatura recente também mostra que sistemas agênticos não escalam adequadamente quando dependem de integrações pontuais e interfaces proprietárias. Protocolos como MCP, A2A, ANP e ACP surgem justamente para organizar diferentes dimensões dessa interoperabilidade: acesso a ferramentas e dados, delegação entre agentes, descoberta em redes abertas e comunicação HTTP multimodal. Para manter aderência à literatura técnica atual, este capítulo emprega ANP como Agent Network Protocol e ACP como Agent Communication Protocol. As ideias de processos não agênticos e comunicação com recursos de nuvem, presentes na proposta-base, são tratadas como funções arquiteturais internas, e não como denominações formais das siglas.

OBJETIVO

Analisar, por meio de revisão da literatura e síntese arquitetural, como Google Apps Script, Google Sheets e Google Colab podem sustentar uma arquitetura agêntica serverless para aplicações CRUD e fluxos de análise de dados, considerando os papéis complementares dos protocolos MCP, A2A, ANP e ACP na interoperabilidade entre agentes, ferramentas, dados e serviços gerenciados.

METODOLOGIA

Este estudo caracteriza-se como pesquisa qualitativa, aplicada, exploratória e bibliográfica. A revisão foi conduzida a partir de documentos técnicos oficiais, especificações de protocolos, documentação de plataformas Google e produções científicas sobre computação serverless, uso de ferramentas por modelos de linguagem, agentes autônomos e sistemas multiagentes. Foram priorizadas fontes primárias ou de autoridade técnica reconhecida, como especificações oficiais do MCP (Model Context Protocol, 2025) e A2A (A2A Protocol Working Group, 2026), documentação do Google Apps Script e Google Sheets API, materiais institucionais da IBM Research sobre ACP e o repositório oficial do ANP.

A análise seguiu três procedimentos. Primeiro, foram identificados conceitos estruturantes da literatura: serverless, CRUD, eventos, agentes baseados em LLM, tool use, comunicação agente-agente e descoberta de capacidades. Em seguida, esses conceitos foram relacionados às camadas da arquitetura proposta: persistência, orquestração, processamento analítico e comunicação. Por fim, foi elaborada uma síntese crítica para indicar oportunidades, limites e requisitos mínimos de implementação. Não houve pesquisa com seres humanos, experimentação animal ou coleta de dados pessoais, de modo que não se aplicam procedimentos de comitê de ética.

FUNDAMENTAÇÃO TEÓRICA E REVISÃO DA LITERATURA

A revisão da literatura indica que arquiteturas agênticas serverless dependem da convergência entre quatro corpos de conhecimento: computação serverless, sistemas de dados leves, agentes baseados em modelos de linguagem e padrões de interoperabilidade.

No campo da computação serverless, Baldini et al. (2017) descrevem a emergência de plataformas que executam código em resposta a eventos e aliviam o desenvolvedor de tarefas de provisionamento, manutenção e escalonamento. Essa característica é particularmente relevante para aplicações institucionais pequenas ou médias, nas quais a manutenção de servidores, bancos de dados e filas de mensagens pode representar custo desproporcional ao valor do sistema. Entretanto, a literatura também adverte que o serverless não elimina complexidade: ele desloca parte dela para limites de execução, modelo de permissão, integração entre serviços, monitoramento e desenho de tarefas idempotentes.

Jonas et al. (2019) ampliam essa discussão ao defender que o serverless simplifica a programação em nuvem, mas ainda enfrenta limites em cargas que exigem estado compartilhado, comunicação intensiva e execução prolongada. Essa observação é central para a arquitetura aqui proposta. Apps Script é adequado para eventos, validações, endpoints e automações curtas; Colab é mais adequado para análises exploratórias, processamento Python e geração de relatórios; Sheets é prático para persistência tabular, auditoria manual e colaboração. A arquitetura torna-se mais consistente quando cada ferramenta é usada conforme sua vocação, evitando transformar uma camada leve em substituto integral de um banco transacional ou de uma plataforma de execução permanente.

No eixo dos agentes baseados em modelos de linguagem, a literatura mostra que a autonomia operacional depende da capacidade de alternar raciocínio e ação. Yao et al. (2022), ao proporem o paradigma ReAct, demonstram a importância de intercalar etapas de raciocínio com a execução de ações em ambientes externos. Schick et al. (2023), com o Toolformer, reforçam a ideia de que modelos de linguagem podem se beneficiar do uso seletivo de APIs, calculadoras, buscadores e outros recursos especializados. Esses estudos sustentam a premissa de que um agente útil em ambiente organizacional não deve apenas gerar texto: ele precisa consultar dados, executar operações, validar respostas e incorporar

resultados externos ao seu processo decisório.

Wang et al. (2024) sistematizam a evolução dos agentes autônomos baseados em LLMs em torno de componentes como perfil, memória, planejamento e ação. Essa estrutura é útil para interpretar a integração entre Apps Script, Sheets e Colab. O perfil e as permissões do agente podem ser definidos por configurações e escopos de acesso; a memória operacional pode ser representada em tabelas, logs e metadados no Sheets; o planejamento pode ser conduzido por prompts, regras e filas de tarefas; e a ação ocorre por chamadas HTTP, execução de scripts e notebooks analíticos. Assim, mesmo uma infraestrutura gratuita ou de baixo custo pode materializar elementos relevantes de uma arquitetura agêntica, desde que haja separação clara entre controle, dados e execução.

Sistemas multiagentes ampliam esse desafio porque exigem coordenação entre entidades com capacidades, estados e responsabilidades diferentes. Wu et al. (2023), ao apresentarem o AutoGen, indicam que conversas entre agentes podem organizar fluxos complexos de solução de problemas, especialmente quando agentes especializados assumem papéis complementares. Essa abordagem dialoga diretamente com o desenho proposto: um agente de interface pode receber solicitações e validar parâmetros no Apps Script; um agente de persistência pode gerenciar operações CRUD sobre Sheets; um agente analítico pode executar notebooks no Colab; e um agente supervisor pode consolidar resultados, logs e exceções.

O principal obstáculo passa a ser a interoperabilidade. Integrações ad hoc entre agentes, planilhas, scripts e notebooks tendem a funcionar em protótipos, mas se tornam frágeis quando crescem em volume, número de usuários ou criticidade. Ehtesham et al. (2025) observam que protocolos de interoperabilidade para agentes surgem para reduzir fragmentação, padronizar comunicação e tornar fluxos agênticos mais reutilizáveis. O autor compara MCP, ACP, A2A e ANP (Quadro 1) em dimensões como modo de interação, descoberta, segurança e padrão de comunicação, propondo uma adoção progressiva: primeiro acesso padronizado a ferramentas e dados, depois mensagens estruturadas, delegação de tarefas e descoberta descentralizada.

Quadro 1: Síntese da literatura aplicada à arquitetura proposta.

Eixo	Contribuição da literatura	Aplicação na arquitetura
Serverless	Reduz administração de infraestrutura, mas exige atenção a estado, cotas e observabilidade.	Apps Script executa eventos e endpoints curtos; Colab processa tarefas analíticas delimitadas.
Agentes LLM	Agentes tornam-se mais úteis quando combinam raciocínio, memória, ferramentas e ações externas.	Sheets registra memória operacional; Apps Script executa ações; Colab produz análises.
MCP	Padroniza acesso a ferramentas, recursos e prompts por interface estruturada.	Expõe tabelas, esquemas e operações CRUD como recursos ou ferramentas controladas.
A2A	Padroniza colaboração, descoberta e delegação entre agentes independentes.	Coordena agente de interface, agente de persistência e agente analítico.
ACP	Valoriza comunicação HTTP simples, multimodal e assíncrona entre agentes heterogêneos.	Facilita mensagens entre Apps Script e agentes externos sem acoplamento forte a SDKs.
ANP	Enfatiza identidade, descoberta e comunicação segura em redes abertas de agentes.	Orienta evolução para identificação de agentes, catálogo de capacidades e governança.

Fonte: Elaboração própria a partir da literatura revisada.

O Model Context Protocol (MCP) ocupa papel fundamental nesse percurso. A especificação do MCP define uma arquitetura composta por hosts, clients e servers, comunicando-se por mensagens JSON-RPC 2.0 e expondo recursos, prompts e ferramentas (Model Context Protocol, 2025). Para o caso estudado, isso significa que uma planilha não precisa ser vista apenas como arquivo: ela pode ser descrita como recurso estruturado, com esquemas, operações permitidas e ferramentas controladas para leitura, inserção, atualização e exclusão. Em termos práticos, o Apps Script pode funcionar como adaptador entre a planilha e um servidor MCP, traduzindo operações de alto nível em chamadas seguras ao Sheets.

O Agent2Agent Protocol (A2A) complementa o MCP por tratar da comunicação entre agentes independentes. A própria especificação do A2A afirma que MCP se concentra no acesso de agentes a ferramentas, APIs e fontes de dados, enquanto A2A se concentra na colaboração entre agentes pares, incluindo descoberta, delegação, tarefas e troca de resultados (A2A Protocol Working Group, 2026). Essa distinção é decisiva para evitar sobreposição conceitual. Em uma arquitetura com Apps Script, Sheets e Colab, o MCP pode padronizar o acesso ao cadastro, aos logs e aos metadados; o A2A pode padronizar a solicitação de uma análise ao agente Colab, o acompanhamento de seu status e o retorno de artefatos, como tabelas, gráficos ou relatórios.

O ACP, introduzido no ecossistema BeeAI/IBM Research e posteriormente incorporado ao movimento de convergência com A2A sob governança aberta, reforça a importância de interfaces HTTP simples, mensagens multimodais e comunicação assíncrona (IBM Research, 2025). Sua contribuição para este capítulo está menos na competição com A2A e mais na ênfase pragmática: agentes de frameworks diferentes devem conseguir trocar mensagens sem dependência obrigatória de SDKs específicos. Essa característica é coerente com o uso de Apps Script, que opera de forma natural por endpoints HTTP doGet(e) e doPost(e), e com Colab, que pode consumir arquivos, APIs e mensagens intermediadas por Drive ou HTTP.

O ANP, por sua vez, desloca a discussão para redes abertas de agentes. O projeto Agent Network Protocol apresenta uma arquitetura em três camadas: identidade e comunicação segura baseada em Decentralized Identifiers (DIDs), camada de metaprotocolo para negociação e camada de aplicação para descrição e descoberta de capacidades (Agent Network Protocol, 2025). Embora uma implementação completa de ANP seja mais ampla do que o escopo de uma solução Apps Script-Sheets-Colab, seus princípios ajudam a formular requisitos de evolução: cada agente deve ser identificável, suas capacidades devem ser publicadas de modo legível por máquina, e a comunicação deve preservar autenticidade, autorização e rastreabilidade.

No eixo das plataformas Google, a documentação oficial confirma a viabilidade técnica das camadas propostas. O Google Apps Script permite publicar scripts como web apps desde que contenham funções doGet(e) ou doPost(e) que retornem saídas HTML ou texto (Google Developers, 2026c). Também oferece gatilhos simples, como onOpen(e) e onEdit(e), e gatilhos instaláveis para cenários que exigem autorização ou maior controle (Google Developers, 2026b). Essa combinação favorece fluxos orientados a eventos, como validação de novos registros, atualização de status, envio de requisições HTTP e acionamento de rotinas periódicas.

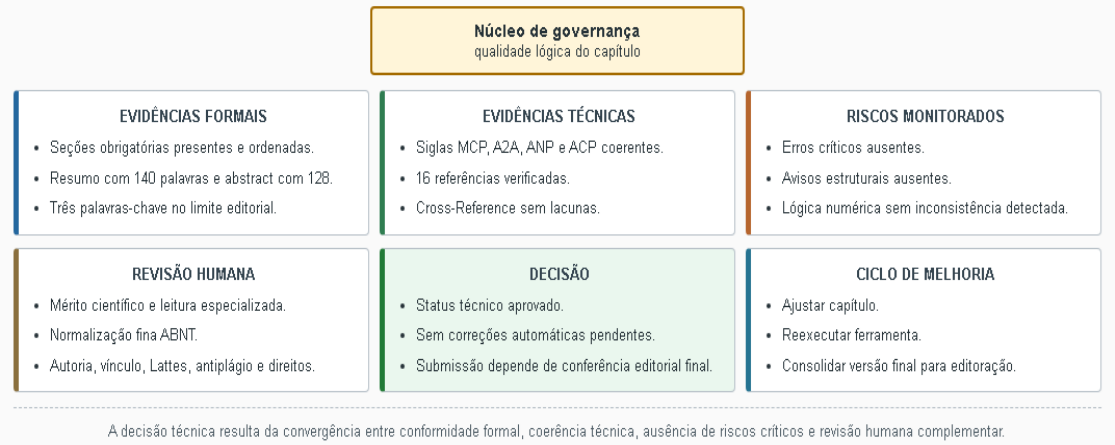
O Google Sheets, por sua vez, oferece uma representação tabular que é compreensível para usuários não técnicos e acessível por API. A documentação da Sheets API descreve recursos para leitura e escrita de valores em intervalos, organizados por linhas ou colunas (Google Developers, 2025). Isso torna o Sheets adequado como repositório leve de configurações, filas, histórico de tarefas, registros CRUD e indicadores operacionais. Sua principal vantagem não é competir com bancos relacionais, mas servir como camada colaborativa, auditável e de baixo atrito para projetos que precisam de transparência e edição manual controlada.

O Google Colab completa a arquitetura como camada analítica. Segundo a documentação do Colab, notebooks são armazenados no Google Drive ou carregados do GitHub, e o código é executado em máquinas virtuais privadas do usuário, com estado sujeito a interrupções e limites do serviço (Google Colab, 2026). Essa natureza torna o Colab adequado para análises exploratórias, scripts Python, geração de gráficos, validação

estatística e relatórios sob demanda, mas inadequado como servidor permanente. A literatura serverless ajuda a interpretar esse limite: o Colab deve ser acionado por tarefas bem delimitadas, com entradas e saídas persistidas fora da máquina virtual, preferencialmente no Drive, Sheets ou outro repositório controlado.

A integração entre validação automatizada, evidências do relatório, revisão humana e decisão de prontidão para submissão é detalhada na Figura 2, que apresenta a matriz de governança da qualidade acadêmica adotada neste capítulo.

Figura 2: Matriz de Governança da Qualidade Acadêmica



Fonte: Elaborado pelo autor.

RESULTADOS E DISCUSSÃO

A revisão permite sustentar que a arquitetura proposta é conceitualmente coerente, desde que seja apresentada como solução serverless leve e não como substituto universal de plataformas corporativas completas. O Apps Script deve assumir a camada de orquestração e exposição de endpoints, pois sua documentação oficial demonstra suporte a web apps, eventos e execução integrada ao Google Workspace. O Sheets deve assumir a camada de persistência tabular e governança operacional, especialmente para registros CRUD, filas de tarefas e logs. O Colab deve assumir a camada de processamento analítico especializado, desde que os notebooks sejam acionados de modo controlado e não dependam de estado efêmero.

O primeiro resultado teórico é a separação entre acesso a ferramentas e comunicação entre agentes. MCP é mais apropriado para descrever recursos, ferramentas e operações sobre dados; A2A é mais apropriado para delegar tarefas entre agentes. Essa separação evita que a arquitetura trate uma planilha, um script e um notebook como componentes equivalentes. Cada um possui função distinta: a planilha armazena estado, o script executa e controla eventos, o notebook processa, e o protocolo define como esses componentes se tornam acessíveis a agentes.

O segundo resultado é a necessidade de desenhar operações como tarefas rastreáveis. Em vez de acionar diretamente uma análise pesada a partir de uma edição em planilha, a literatura recomenda desacoplamento: o evento cria uma tarefa no Sheets; o Apps Script valida permissões, registra status e envia mensagem; o agente analítico consome a tarefa, processa no Colab e grava o resultado. Esse padrão reduz perda de informação, melhora auditoria e permite reprocessamento em caso de falha.

O terceiro resultado envolve segurança e governança. A documentação do Apps Script registra cotas, limites de tempo de execução e exceções quando limites são ultrapassados (Google Developers, 2026a). Logo, uma arquitetura robusta precisa prever controle de volume, idempotência, tratamento de erros e monitoramento. Do ponto de vista dos protocolos, A2A e ANP reforçam autenticação, autorização, descoberta controlada e descrição de capacidades; MCP reforça a delimitação de ferramentas e recursos acessíveis ao agente. A aplicação desses princípios evita que um agente tenha acesso irrestrito a uma planilha inteira quando precisa apenas de um subconjunto de campos ou operações.

O quarto resultado é a adequação da proposta a cenários de pesquisa, educação, prototipagem e operações institucionais de pequeno e médio porte. A arquitetura é especialmente útil quando usuários precisam auditar dados em planilhas, executar rotinas Python sem manter servidores e construir interfaces simples para consulta, cadastro e relatório. Entretanto, a própria literatura indica limites: alta concorrência, dados sensíveis em grande escala, requisitos transacionais fortes e execução contínua exigem camadas adicionais, como bancos de dados dedicados, filas gerenciadas, controle de segredos, observabilidade centralizada e infraestrutura cloud convencional.

CONSIDERAÇÕES FINAIS

A revisão da literatura demonstra que a integração entre Google Apps Script, Google Sheets e Google Colab pode sustentar uma arquitetura agêntica serverless viável para sistemas CRUD e fluxos analíticos de baixo custo. A robustez da proposta não decorre apenas das ferramentas Google, mas do modo como elas são organizadas: Sheets como persistência e memória operacional, Apps Script como orquestrador orientado a eventos e Colab como camada analítica sob demanda.

Os protocolos MCP, A2A, ACP e ANP oferecem uma linguagem técnica para amadurecer essa arquitetura. MCP estrutura o acesso a dados e ferramentas; A2A estrutura delegação e colaboração entre agentes; ACP reforça comunicação HTTP pragmática e interoperável; ANP amplia a discussão para identidade, descoberta e redes abertas. A principal contribuição do capítulo é mostrar que esses protocolos não precisam ser tratados como modismos concorrentes, mas como camadas complementares de um mesmo problema: permitir que agentes usem ferramentas, conversem entre si, descubram capacidades e operem com governança.

Conclui-se que a proposta é adequada para protótipos avançados, automação acadêmica, sistemas departamentais e pesquisa aplicada em inteligência artificial agêntica. Para uso em ambientes críticos, recomenda-se evoluir a arquitetura com controle de acesso granular, logs imutáveis, validação de entradas, testes de carga, monitoramento de cotas e segregação entre dados sensíveis e dados operacionais. Assim, a arquitetura mantém sua vantagem de baixo custo sem perder critérios essenciais de confiabilidade, segurança e manutenção.

REFERÊNCIAS

A2A PROTOCOL WORKING GROUP. **Agent2Agent (A2A) Protocol Specification**. A2A Protocol, 2026. Disponível em: https://a2a-protocol.org/latest/specification/. Acesso em: 23 maio 2026.

AGENT NETWORK PROTOCOL. **AgentNetworkProtocol: open source protocol for agent communication**. GitHub, 2025. Disponível em: https://github.com/agent-network-protocol/AgentNetworkProtocol. Acesso em: 23 maio 2026.

BALDINI, Ioana et al. **Serverless computing: current trends and open problems**. arXiv, 2017. Disponível em: https://arxiv.org/abs/1706.03178. Acesso em: 23 maio 2026.

EHTESHAM, Abul et al. **A survey of agent interoperability protocols: Model Context Protocol (MCP), Agent Communication Protocol (ACP), Agent-to-Agent Protocol (A2A), and Agent Network Protocol (ANP)**. arXiv, 2025. DOI: 10.48550/arXiv.2505.02279. Disponível em: https://arxiv.org/abs/2505.02279. Acesso em: 23 maio 2026.

GOOGLE COLAB. **Frequently Asked Questions**. Google Research, 2026. Disponível em: https://research.google.com/colaboratory/faq.html. Acesso em: 23 maio 2026.

GOOGLE DEVELOPERS. **REST Resource: spreadsheets.values**. Google for Developers, 2025. Disponível em: https://developers.google.com/workspace/sheets/api/reference/rest/v4/spreadsheets.values. Acesso em: 23 maio 2026.

GOOGLE DEVELOPERS. **Quotas for Google Services**. Google for Developers, 2026a. Disponível em: https://developers.google.com/apps-script/guides/services/quotas. Acesso em: 23 maio 2026.

GOOGLE DEVELOPERS. **Simple Triggers**. Google for Developers, 2026b. Disponível em: https://developers.google.com/apps-script/guides/triggers. Acesso em: 23 maio 2026.

GOOGLE DEVELOPERS. **Web Apps**. Google for Developers, 2026c. Disponível em: https://developers.google.com/apps-script/guides/web. Acesso em: 23 maio 2026.

IBM RESEARCH. **Agent Communication Protocol (ACP)**. IBM Research, 2025. Disponível em: https://research.ibm.com/projects/agent-communication-protocol. Acesso em: 23 maio 2026.

JONAS, Eric et al. **Cloud programming simplified: a Berkeley view on serverless computing**. arXiv, 2019. Disponível em: https://arxiv.org/abs/1902.03383. Acesso em: 23 maio 2026.

MODEL CONTEXT PROTOCOL. **Specification**. Model Context Protocol, 2025. Disponível em: https://modelcontextprotocol.io/specification/2025-03-26/. Acesso em: 23 maio 2026.

SCHICK, Timo et al. Toolformer: language models can teach themselves to use tools. In: **ADVANCES IN NEURAL INFORMATION PROCESSING SYSTEMS**, 36., 2023. Proceedings [...]. [S. l.]: Curran Associates, 2023. Disponível em: https://proceedings.neurips.cc/paper_files/paper/2023/hash/d842425e4bf79ba039352da0f658a906-Abstract-Conference.html. Acesso em: 23 maio 2026.

WANG, Lei et al. A survey on large language model based autonomous agents. **Frontiers of Computer Science**, v. 18, n. 6, 2024. DOI: 10.1007/s11704-024-40231-1. Disponível em: https://link.springer.com/article/10.1007/s11704-024-40231-1. Acesso em: 23 maio 2026.

WU, Qingyun et al. **AutoGen: enabling next-gen LLM applications via multi-agent conversation**. arXiv, 2023. Disponível em: https://arxiv.org/abs/2308.08155. Acesso em: 23 maio 2026.

YAO, Shunyu et al. **ReAct: synergizing reasoning and acting in language models**. arXiv, 2022. Disponível em: https://arxiv.org/abs/2210.03629. Acesso em: 23 maio 2026.