UNIVERSIDADE FEDERAL RURAL DO SEMI-ÁRIDO

PRÓ-REITORIA DE GRADUAÇÃO

DEPARTAMENTO DE ENGENHARIAS E TECNOLOGIA

CURSO DE ENGENHARIA DE SOFTWARE

PAULINA JULIA COSTA DE OLIVEIRA

UMA INVESTIGAÇÃO SOBRE AGENTES DE IA NA GERÊNCIA E MANUTENÇÃO DE INFRAESTRUTURA EM NUVEM DE COMPUTADORES

PAU DOS FERROS

2026

PAULINA JULIA COSTA DE OLIVEIRA

UMA INVESTIGAÇÃO SOBRE AGENTES DE IA NA GERÊNCIA E MANUTENÇÃO DE INFRAESTRUTURA EM NUVEM DE COMPUTADORES

Monografia apresentada à Universidade Federal Rural do Semi-Árido como requisito para obtenção do título de Bacharela em Engenharia de Software.

Orientador: Walber José Adriano Silva, Prof. Dr.

PAU DOS FERROS

2026

O 49i Oliveira, Paulina Julia Costa de .
Uma investigação sobre agentes de IA na gerência e manutenção de infraestrutura em nuvem de computadores / Paulina Julia Costa de Oliveira. - 2026.
34 f. : il.

Orientador: Walber José Adriano Silva.
Monografia (graduação) - Universidade Federal Rural do Semi-árido, Curso de Engenharia de Software, 2026.

1. Computação em Nuvem. 2. Inteligência Artificial. 3. Agentes de IA. 4. AIOps. 5. Self-healing. I. Silva, Walber José Adriano , orient. II. Título.

Ficha catalográfica elaborada por sistema gerador automático em conformidade com AACR2 e os dados fornecidos pelo) autor(a).
Biblioteca Campus Pau dos Ferros
Bibliotecária: Erlanda Maria Lopes da Silva
CRB: 15/966

PAULINA JULIA COSTA DE OLIVEIRA

UMA INVESTIGAÇÃO SOBRE AGENTES DE IA NA GERÊNCIA E MANUTENÇÃO DE INFRAESTRUTURA EM NUVEM DE COMPUTADORES

Monografia apresentada à Universidade Federal Rural do Semi-Árido como requisito para obtenção do título de Bacharela em Engenharia de Software.

Aprovada em: 06/07/2026

BANCA EXAMINADORA

Walber José Adriano Silva, Prof. Dr. (UFERSA)
Presidente

Reudismam Rolim de Sousa, Prof. Dr. (UFERSA)
Membro Examinador

Bruno Victor Paiva da Silva (UFRN)
Membro Examinador

AGRADECIMENTOS

Agradeço primeiramente a Deus. Em muitos momentos, quando minhas forças pareciam insuficientes, foi na fé que encontrei coragem para continuar. Nem sempre o caminho foi leve, e muitas vezes precisei acreditar antes mesmo de conseguir enxergar a linha de chegada.

À minha família, meu mais profundo agradecimento por todo amor, incentivo e compreensão, especialmente a minha mãe, Maria Elioneide Costa Campos, que sob sol, chuva e todos os desafios que enfrentou na vida, nunca deixou de priorizar o estudo para mim e para todos os meus irmãos.

Ao meu pai, que hoje não está fisicamente presente para vivenciar mais esta conquista comigo, mas permanece em mim. Concluir esta etapa também é uma forma de honrar sua memória, pois hoje sua "Julhinha" alcança mais uma graduação, agora como engenheira de software, carregando consigo todas as etapas já vencidas ao longo do caminho.

Ao meu tio, que igualmente deixou saudades, levo comigo o carinho e as lembranças dos momentos em que, no início de tudo, era quem me levava e buscava na universidade, muitas vezes tarde da noite, sempre me apoiando.

Ao meu orientador, pela paciência, disponibilidade e pelos conhecimentos compartilhados ao longo da elaboração deste trabalho e dos anos de estudo nesta instituição. Sua orientação foi fundamental para que este projeto se concretizasse.

Aos professores que contribuíram para minha formação, por cada ensinamento que ultrapassou a sala de aula e ajudou a construir a profissional e a pessoa que me tornei.

Aos amigos e colegas que caminharam ao meu lado, especialmente àqueles que, durante esses anos, compreenderam minhas ausências, celebraram as pequenas vitórias e me lembraram, tantas vezes, que eu era capaz quando eu mesma duvidava disso.Que por vezes me lembraram quem sou... Obrigada por jamais soltarem a minha mão.

Também agradeço a todas as pessoas que, direta ou indiretamente, fazem parte desta conquista.

Ela representa mais que apenas a conclusão de mais uma etapa, representa a persistência de quem se recusou a desistir.

EPÍGRAFE

"A coragem não é a ausência do medo, mas a decisão de que algo é mais importante que o medo."

(Ambrose Redmoon)

RESUMO

A rápida evolução da computação em nuvem transformou a infraestrutura de TI em sistemas altamente dinâmicos e complexos. Atualmente, a gestão dessas plataformas, como a *Amazon Web Services* (AWS), gera um grande volume de telemetria que desafia o monitoramento manual. A partir dessa lacuna, esta pesquisa investiga como os agentes de IA são capazes de atuar na gerência e manutenção de infraestrutura, compreendendo que a Inteligência Artificial (IA) surge como um elemento catalisador da automatização. A realização do estudo se dá pelo uso da ferramenta *OpenClaw* para avaliar seu comportamento em cenários de falhas em ambientes de nuvem, como a AWS. A fundamentação teórica baseia-se nos conceitos de Computação em Nuvem e AIOps, destacando a necessidade de superar as limitações do monitoramento manual. Utilizam-se agentes de IA para detecção de anomalias e manutenção preditiva, fundamentando o uso destes e do *OpenClaw* na implementação de mecanismos de *self-healing* (autocura). Metodologicamente, trata-se de um estudo exploratório e experimental, que investigou o uso de agentes de IA na gerência de infraestrutura em computação em nuvem. De acordo com os resultados, os testes demonstram que a comunicação entre o administrador e o agente ocorreu de forma satisfatória; identificou-se ainda que o agente depende de *scripts* previamente definidos para executar ações administrativas. Deste modo, compreende-se que o modelo de linguagem atua como uma interface inteligente para interpretar solicitações e apoiar a tomada de decisão, enquanto a execução das ações ocorre por procedimentos previamente autorizados, reduzindo os riscos da execução de comandos arbitrários. Os resultados indicam ainda que a arquitetura proposta apresenta potencial para identificar e recuperar falhas em serviços de infraestrutura.

Palavras-chave: computação em nuvem; inteligência artificial; agentes de IA; AIOps; autocura.

ABSTRACT

The rapid evolution of cloud computing has transformed IT infrastructure into highly dynamic and complex systems. Currently, the management of platforms such as Amazon Web Services (AWS) generates a large volume of telemetry that challenges manual monitoring. Based on this gap, this research investigates how AI agents can act in infrastructure management and maintenance, considering that Artificial Intelligence (AI) has emerged as a catalyst for automation. The study was conducted using the OpenClaw tool to evaluate its behavior in failure scenarios within cloud environments such as AWS. The theoretical foundation is based on the concepts of Cloud Computing and AIOps, highlighting the need to overcome the limitations of manual monitoring. AI agents are employed for anomaly detection and predictive maintenance, supporting the use of these agents and OpenClaw in the implementation of self-healing mechanisms. Methodologically, this is an exploratory and experimental study that investigated the use of AI agents in cloud infrastructure management. According to the results, the tests demonstrated that communication between the administrator and the agent was satisfactory. It was also found that the agent relies on predefined scripts to execute administrative actions. Thus, the language model acts as an intelligent interface for interpreting requests and supporting decision-making, while the execution of actions is carried out through previously authorized procedures, reducing the risks associated with arbitrary command execution. The results also indicate that the proposed architecture has the potential to identify and recover from failures in infrastructure services.

Keywords: cloud computing; artificial intelligence; AI agents; AIOps; self-healing.

LISTA DE FIGURAS

LISTA DE SÍMBOLOS

AIOps	*Artificial Intelligence for IT Operations*
API	*Application Programming Interface*
AWS	*Amazon Web Services*
CPU	Central Processing Unit
EC2	*Elastic Compute Cloud*
HPC	*High Performance Computing*
IA	Inteligência Artificial
ISO	*International Organization for Standardization*
LGPD	Lei Geral de Proteção de Dados
LLMs	*Large Language Models*
MCN	Modelos de Comportamento Normal
ML	*Machine Learning*
MTTR	*Mean Time to Repair*
NIST	*National Institute of Standards and Technology*
RAM	*Random Access Memory*
RDS	*Relational Database Service*
S3	*Simple Storage Service*
SGSI	Sistema de Gestão da Segurança da Informação
SSH	*Secure Shell*
TI	Tecnologia da Informação

SUMÁRIO

1 INTRODUÇÃO

A rápida evolução da computação em nuvem transformou a infraestrutura de TI em sistemas altamente dinâmicos e complexos. Atualmente, a gestão dessas plataformas, como por exemplo a *Amazon Web Services* (AWS), gera um volume massivo de telemetria que desafia a capacidade de monitoramento manual. Diante desse cenário, a Inteligência Artificial (IA) surge como um elemento catalisador de automatização, permitindo a transição de uma manutenção reativa para uma gestão autônoma e preditiva (Dang; Lin; Huang, 2019).

Especificamente, o uso de Inteligência Artificial Generativa (Gen AI) tem permitido explorar novas formas de realizar gestão de infraestrutura, onde se delega responsabilidades a agentes de IA, que são softwares capazes de executar ações de forma autônoma e utilizar ferramentas para interagir no ambiente que se encontra. Por isso, é relevante a necessidade de se analisar como agente de IA procede em cenário de gerenciamento infraestrutura.

Contudo, trazer novas formas de abordar a administração de sistemas é fundamental, pois tem a possibilidade de expandir o entendimento da dinâmica da adoção de novas tecnologias modos tradicionais. Por exemplo, para se estabelecer um Sistema Gestor de Segurança da Informação (SGSI), descrito na normal ABNT NBR ISO/IEC 27001, é necessário múltiplas entidades envolvidas em uma organização para conseguir a efetividade da segurança. Ao invés de existir pessoas alocadas para realizar determinadas ações no SGSI, a adoção de agentes de IA para assumir funções de operação, avaliação, melhorias, entre outros, tem o potencial de mitigar a demanda operacional sobre agentes humanos e reduzir riscos críticos, como os de segurança da informação.

Por isso, este trabalho tem o objetivo de realizar uma investigação de agentes de IA na gerência e manutenção de infraestrutura em nuvem de computadores. Diferente de automações tradicionais baseadas em regras fixas, os agentes de IA podem identificar anomalias e executar ações de autorreparação, o chamado *self-healing* (Kim et al., 2025). O estudo pretende explorar ferramentas como o OpenClaw[1], que irá implementar agente de IA, para avaliar como esse agente se comporta em cenário de falha provocado em ambientes de nuvem como a AWS. Ao final, espera-se compreender os limites técnicos e a viabilidade dessa tecnologia na preservação da infraestrutura em ambientes computacionais reais.

[1] Disponível em: https://openclaw.ai/

2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo traz um embasamento teórico sobre os principais conceitos de infraestrutura de Tecnologia da Informação (TI) e IA.

2.1 Infraestrutura de TI

A gestão de TI é caracterizada pela área organizacional que é responsável pelo planejamento, coordenação e controle dos recursos tecnológicos. Ela tem seu fator estratégico em diversas organizações com base tecnológica como, por exemplo, em plataformas de *streaming* que necessitam de um grande volume de acessos, setor financeiro que está relacionado à segurança (cultura de *Financial Operations* - FinOps) e o mercado do varejo digital, que necessita de uma alta elasticidade (Laudon; Laudon, 2022). Organizações assim precisam de infraestrutura de TI, que fornece a base tecnológica, e é composta por software, rede de computadores, hardware e serviços que garantem o funcionamento dos sistemas adotados nas organizações.

Nesse contexto, uma gestão de infraestrutura eficaz se torna necessária para alinhar as capacidades tecnológicas aos objetivos da organização, garantindo agilidade e vantagem competitiva em ambientes de constante mudança (Lannacci, 2014). Para atender a essa demanda por velocidade, eficiência e alta disponibilidade dos serviços, abordagens como o *DevOps* surgiram com o intuito de integrar operações, promovendo automação e melhoria contínua na entrega de serviços (Bass; Weber; Zhu, 2015).

A necessidade de provisionar recursos rapidamente e de escalar aplicações em tempo real exigem uma mudança no paradigma de como a infraestrutura é consumida e gerenciada pelas equipes de operações (Morris, 2016). Em um ambiente organizacional tradicional, a TI é responsável por todas as mudanças no ambiente, desde os elementos físicos como *hardware* de servidores até atualização de *software*. Em um cenário de infraestrutura de TI em nuvem de computadores, esta sobrecarga gerencial é mitigada pelo provisionamento de serviços pelo próprio provedor do serviço de nuvem. Isso tem estimulado a adoção de ambientes distribuídos e virtualizados que emergem como soluções a abordagens tradicionais de sistemas (Tanenbaum et al., 2008; Erl; Monroy, 2023).

2.1.1 Servidores Web

Alguns exemplos de servidores web responsáveis por receber requisições de clientes por meio dos protocolos *HyperText Transfer Protocol* (HTTP) e *HyperText Transfer Protocol Secure* (HTTPS), para processá-las e disponibilizar páginas, arquivos e serviços aos usuários. Além da entrega de conteúdo, esses servidores podem atuar como balanceadores de cargas e intermediários entre clientes e aplicações, desempenhando papel importante na disponibilidade e no desempenho dos sistemas.

Entre os servidores web mais utilizados destacam-se o Apache HTTP Server[1] e o Nginx[2]. O Apache HTTP *Server* é um projeto de código aberto adotado devido à sua arquitetura modular, que permite a inclusão de funcionalidades por meio de módulos específicos.

Um dos métodos de acesso e controle de servidores remotamente com segurança é o protocolo de rede, *Secure Shell (SSH)*[3]. O SSH é um método para enviar comandos com segurança a um computador em uma rede não segura, que usa criptografia para autenticar e criptografar conexões entre dispositivos.

2.1.2 Computação em Nuvem

Para suprir a necessidade de provisionamento dinâmico e contornar a rigidez estrutural dos *datas centers* locais, a Computação em Nuvem (*Cloud Computing*) consolidou-se como o paradigma tecnológico que transforma infraestrutura em serviço, atendendo a essa demanda por flexibilidade e escalonamento. Conforme a definição clássica do *National Institute of Standards and Technology* (NIST), a nuvem é um modelo que permite acesso ubíquo, conveniente e sob demanda a um *pool* compartilhado de recursos computacionais configuráveis, tais como redes, servidores, armazenamento e aplicações que podem ser rapidamente provisionados e liberados com mínimo esforço de gerenciamento (Mell; Grance et al., 2011).

No contexto da transição de arquiteturas complexas para ambientes virtuais, a literatura aponta a Computação em Nuvem como um dos principais fatores para a modernização da infraestrutura de TI. Nesse cenário, (Borin et al., 2023) destacam que a Computação em Nuvem representa uma mudança de paradigma ao proporcionar recursos computacionais escaláveis sob demanda, substituindo a aquisição de infraestrutura física por um modelo de provisionamento

[1] Disponível em: https://apache2.org/
[2] Disponível em: https://nginx.org/en/
[3] Disponível em: https://www.openssh.org/manual.html

dinâmico e tarifação baseada exclusivamente no uso *(pay-per-use)*. Segundo os autores, essas características favorecem maior flexibilidade, escalabilidade e otimização de custos, aspectos que justificam a ampla adoção da nuvem em aplicações distribuídas e em soluções baseadas em IA, como a arquitetura empregada neste trabalho.

No entanto, a transição para ambientes em nuvem altamente elásticos e distribuídos trouxe um novo desafio operacional: o aumento exponencial da complexidade de gerenciamento (Srivastava; Agrawal, 2026). À medida que as arquiteturas modernas adotam dezenas ou centenas de serviços escalando dinamicamente, a interdependência dos recursos e o volume de dados de telemetria, como *logs*, métricas e rastreamentos dispersos, criam uma teia arquitetônica de difícil compreensão sistêmica (Newman, 2022).

É nesse cenário de sobrecarga informacional que as abordagens tradicionais de gestão se mostram severamente insuficientes. Estratégias convencionais, como o monitoramento visual de painéis *dashboards*, a configuração de alerta baseados em limites estáticos, que frequentemente geram falsos positivos e fadiga de alerta, e a análise reativa por meio de chamados *tickets*, não acompanham a velocidade exigida pela nuvem. A necessidade de respostas em tempo real tornou a detecção e resolução de falhas a olho humano praticamente inviável, exigindo a adoção de operações automatizadas e inteligentes (Murphy et al., 2016).

2.1.3 Amazon Web Service - AWS

Dentre as plataformas mais conhecidas que oferecem serviços em computação em nuvem está a AWS. A AWS fornece um conjunto de serviços para infraestrutura, armazenamento, processamento, redes, segurança,IA, banco de dados e análise de dados. A plataforma, na qual, seus serviços são utilizados por empresas, instituições de pesquisa e organizações de diferentes tamanhos para hospedar aplicações e gerenciar recursos computacionais.

O modelo de computação em nuvem adotado pela AWS permite que recursos computacionais sejam disponibilizados sob demanda, eliminando a responsabilidade da necessidade de aquisição e manutenção de infraestrutura física própria da organização que usa o serviço. Nesse modelo, as organizações podem provisionar servidores, armazenar dados e utilizar diversos serviços especializados por meio da internet, pagando apenas pelos recursos efetivamente consumidos. Essa abordagem proporciona flexibilidade, escalabilidade e redução de custos operacionais em aplicações com demandas variáveis.

A AWS disponibiliza centenas de serviços organizados em diferentes categorias. Entre

os mais utilizados destacam-se o *Amazon EC2 (Elastic Compute Cloud)*, responsável pelo provisionamento de instâncias, em que é possível realizar diferentes configurações de acordo com os recursos que são necessários para a aplicação, como o número de vCPUs, GPUs e memória RAM. As instâncias são classificadas por famílias, como as do tipo t3-large, onde t3 representa instâncias que possuem uma performance mais robusta e *large* representa o tamanho de recursos computacionais presentes.

Outros serviços de destaque é o Amazon S3 *(Simple Storage Service)*, destinado ao armazenamento de objetos; o Amazon RDS *(Relational Database Service)*, que facilita o gerenciamento de bancos de dados relacionais; e o AWS *Lambda*, que permite a execução de código no modelo *serverless* - fornece o serviço sem a necessidade de administrar servidores.

2.2 Inteligência Artificial

A IA tem desempenhado um papel cada vez mais relevante na automação e otimização de sistemas computacionais, especialmente em ambientes caracterizados por grande volume de dados e alta complexidade operacional. Nesse contexto, a literatura apresenta diferentes abordagens que empregam técnicas de IA para apoiar o monitoramento, a análise e a tomada de decisão em ambientes de infraestruturas. Esta seção aborda essas aplicações, com ênfase no conceito de *Artificial Intelligence for IT Operations (AIOps)*, que fundamenta a proposta desenvolvida neste trabalho.

2.2.1 AIOps - Inteligência Artificial para Operações de TI

Para lidar com a complexidade gerada pelos ambientes de nuvem e garantir a resiliência contínua dos serviços, emerge o conceito de AIOps (*Artificial Intelligence for IT Operations*). Conforme destacado por (Dang; Lin; Huang, 2019), o AIOps surge da necessidade de capacitar as equipes de operações com algoritmos avançados para lidar eficientemente com a escala, o volume massivo de telemetria e a dinâmica imprevisível das infraestruturas modernas em nuvem. O termo refere-se à aplicação de técnicas de IA, especialmente o *Machine Learning* (Aprendizado de Máquina), para automatizar, aprimorar e otimizar as operações de TI.

Na prática, o AIOps atua ingerindo e analisando o volume de dados operacionais gerados pela infraestrutura como: métricas de consumo de recursos, *logs* de eventos e tráfego de rede para identificar padrões que passariam despercebidos pela análise humana. O grande diferencial

dessas plataformas em relação à automação tradicional baseada em *scripts* de regras estáticas é a sua capacidade preditiva e de adaptação contínua.

Ao invés de apenas alertar a equipe responsavel pelo sistema sobre um incidente que já ocorreu, sistemas munidos de AIOps buscam prever anomalias e gargalos de desempenho antes que causem impacto sistêmico. Mais do que isso, ao integrar capacidades de *Self-healing* (auto-remediação), a infraestrutura torna-se capaz de executar ações corretivas autônomas como: o reinício de contêineres falhos ou o redirecionamento de rotas de rede, reduzindo drasticamente o Tempo Médio de Reparo (MTTR) e preservando a disponibilidade do sistema (Dang; Lin; Huang, 2019).

2.2.2 Machine Learning

O núcleo analítico das plataformas de AIOps é fundamentado em técnicas de ML. Essa subárea da IA concentra-se no desenvolvimento de algoritmos capazes de aprender padrões e extrair conhecimento a partir de grandes volumes de dados, sem a necessidade de programação explícita para cada cenário (Boutaba et al.,). No contexto da gestão de infraestrutura de TI, o principal caso de uso do ML é a detecção de anomalias.

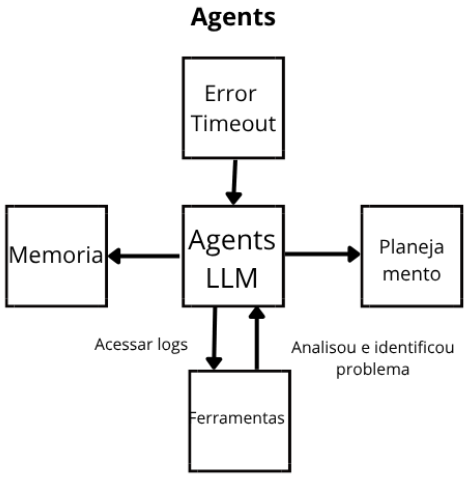
Em ambientes dinâmicos de nuvem, o comportamento do sistema (o *baseline*) flutua constantemente de acordo com a demanda de usuários e períodos de maior utilização. Modelos preditivos de ML são treinados utilizando dados históricos de telemetria para modelar esse comportamento padrão. Uma vez em operação, esses algoritmos monitoram o fluxo contínuo de dados em tempo real e identificam desvios matemáticos significativos como: um vazamento de memória silencioso ou um pico anômalo de processamento.

Contudo, embora o ML tradicional seja eficaz na detecção de anomalias, ele carece de interpretação semântica para resolver problemas complexos (Yang et al., 2024). Para superar essa limitação, plataformas de AIOps têm integrado a IA Generativa (GenAI), impulsionada pelos *Large Language Models* (LLMs), modelos de linguagem de grande escala capazes de processar e gerar textos complexos. Que diferente dos modelos preditivos, as LLMs atuam como um motor de inferência para interpretar documentações e gerar planos de ação (Hou et al., 2024), provendo a base lógica para a operação dos agentes autônomos (Zhao et al., 2023).

2.2.3 Agentes de IA

Enquanto a ML atua como o componente de percepção e diagnóstico, a execução das correções recai sobre os Agentes Autônomos, também conhecidos como agentes de IA. Um agente de IA pode ser definido como uma entidade de software situada em uma máquina virtual da infraestrutura AWS, neste caso, a infraestrutura virtualizada que o percebe continuamente por meio de sensores, como: ferramentas de monitoramento, e atua sobre ele de forma autônoma visando atingir objetivos específicos, como a manutenção da estabilidade do sistema, assim como na Figura 7.

Figura 1 – Descrição do funcionamento de um agente e seus recursos, como memória e planejamento diante de um cenário de erro de *timeout*.



Fonte: Autoria própria (2026).

Na Figura 7, podemos observar o agente se comunicando com suas respectivas ferramentas, com o objetivo de acessar os logs de uma aplicação e em seguida analisar e identificar possiveis problemas. Essa abordagem baseia-se nos princípios da Computação Autonômica *(Autonomic Computing)*, introduzidos por (Kephart; Chess, 2003). Todavia, para lidar com a escala e a heterogeneidade modernas, o paradigma evoluiu para a integração de agentes baseados em LLMs. Pesquisas recentes, como as de Zhao *et al.* (2023) e Microsoft Research (2024), demonstram que esses agentes superam as limitações de sistemas baseados em regras, permitindo que a infraestrutura não apenas execute tarefas pré-programadas, mas exiba capacidades avançadas de raciocínio para o diagnóstico de falhas complexas e a execução de processos de

self-healing de forma dinâmica e adaptativa Apuri, Smith e Doe (2024).

2.2.4 Openclaw

O *Openclaw* é uma plataforma voltada para configuração e uso de agentes de IA, no qual, possuí código aberto voltado ao desenvolvimento e execução de agentes inteligentes baseados em LLMs. Sua principal funcionalidade é permitir que esses agentes não apenas respondem a consultas em linguagem natural, mas que possam executar ações concretas em diferentes sistemas, como manipular arquivos, acessar serviços web, utilizar ferramentas externas e automatizar diferentes fluxos de trabalho.

Diferentemente de assistentes conversacionais tradicionais, o *Openclaw* utiliza uma abordagem orientada a agentes, como, por exemplo, os modelos de linguagem Gemini[4], ChatGPT[5], Claude 3[6] e Codex[7] são integrados a um conjunto de ferramentas capazes de realizar operações no ambiente computacional. Um dos métodos de se comunicar com o agente, é por meio de um canal de comunicação com o usuário, que é o meio pelo qual ocorre a troca de informações entre o usuário e um sistema computacional realizado por meio da plataformas, como o Telegram, Discord e entre outros.

Dessa forma, o agente pode interpretar solicitações do usuário, planejar uma sequência de ações e utilizar ferramentas externas ao *Openclaw* para concluir tarefas de forma autônoma ou semi autônoma. Entre alguns exemplos das tarefas que podem ser executadas pelo agente estão o gerenciamento de e-mails, a navegação na web, a execução de comandos no sistema operacional, o acesso a APIs e a automação de processos administrativos.

[4] Disponível em: https://gemini.google.com
[5] Disponível em: https://chatgpt.com
[6] Disponível em: https://claude.com
[7] Disponível em: https://openai.com/codex.

3 TRABALHOS RELACIONADOS

Trabalhos na literatura utilizam do uso de agentes de IA para realizar o monitoramento de recursos de servidores em infraestruturas. Um exemplo é o trabalho desenvolvido por Weidener *et al.* (2026), no qual a partir do ecossistema *OpenClaw* e da plataforma *Moltbook*[1], que propõe arquiteturas para pesquisa científica autônoma baseadas em colaboração entre agentes de inteligência artificial. Os autores apresentam plataformas como*ClawdLab*[2] e *Beach.science*[3], destinadas à coordenação de agentes especializados, governança de processos científicos, validação baseada em ferramentas externas e execução distribuída de atividades de pesquisa. Além disso, o estudo propõe uma classificação para sistemas multiagentes, classificando-os em *pipelines* de agente único, fluxos de trabalho multiagentes predeterminados e sistemas descentralizados. Embora o objetivo do trabalho citado esteja voltado para a automação da pesquisa científica e não para a gestão de infraestrutura computacional, suas contribuições demonstram aplicações de arquiteturas multiagentes para coordenação, verificação, monitoramento e tomada de decisão em ambientes distribuídos.

Outro trabalho é o *framework* de automação para química computacional baseado no *Openclaw* Ding *et al.* (2026), no qual o *framework* atua como um agente responsável pela coordenação e supervisão de tarefas distribuídas. A arquitetura proposta desacopla o planejamento, a execução e a interação com ambientes computacionais heterogêneos, permitindo a orquestração de *workflows* complexos, recuperação controlada de falhas e integração com ferramentas externas na aplicação química executando em ambientes de HPC - *High Performance Computing*.

Apesar dos avanços recentes na utilização do *OpenClaw* para coordenação de agentes autônomos em ambientes de pesquisa científica e computação de alto desempenho, observa-se uma lacuna na literatura quanto à aplicação desse *framework* em cenários de computação em nuvem voltados à implantação, gerenciamento e monitoramento de aplicações. Os trabalhos existentes concentram-se principalmente na orquestração de *workflows* científicos, na coordenação entre agentes especializados e na execução de tarefas computacionais distribuídas, sem investigar aspectos relacionados à observabilidade operacional, monitoramento de recursos computacionais, gerenciamento de infraestrutura em nuvem e integração com serviços de provedores *Cloud*.

Diante dessa lacuna, este trabalho propõe a utilização do *framework OpenClaw* em um ambiente de computação em nuvem baseado na AWS, para avaliar sua capacidade de coordenar

[1] Dísponivelem:https://www.moltbook.com/
[2] Dísponivel em: https://www.clawdlab.xyz/
[3] Dísponivel em: https://beach.science/

agentes autônomos para implantação, monitoramento e supervisão de aplicações em ambientes de memória distribuídas.

4 DESENVOLVIMENTO

Este capítulo apresenta o desenvolvimento do ambiente utilizado para a investigação do agente de IA aplicado nesta presente pesquisa. Neles são descritos os procedimentos metodológicos adotados, a arquitetura do sistema implementado, bem como a configuração do agente de IA e das ferramentas utilizadas para permitir o monitoramento, diagnóstico e execução de ações sobre o ambiente.

A Seção 4.1 apresenta a metodologia adotada na presente pesquisa, descrevendo a abordagem experimental, o ambiente utilizado para os experimentos e critérios manuseados na avaliação do agente de IA. A Seção 4.2 descreve a arquitetura do sistema, demonstrando os componentes que compõem o ambiente, assim como sua organização e funcionamento. Por fim, a Seção 4.3 demonstra a arquitetura do agente, assim como o processo de implantação do Openclaw, a configuração do ambiente com o modelos, as ferramentas utilizadas e fluxo da comunicação entre usuário da aplicação e o agente e a infraestrutura.

4.1 Metodologia

A presente pesquisa ocorre de modo exploratório e experimental, no qual será estudado sobre o agente de IA e como ele pode ser utilizados na gerência de infraestrutura em computação em nuvem. A investigação terá por objetivo analisar como esse agente funciona de forma autônoma para monitorar e manter um ambiente computacional, explorando os limites e desafios dessa automação em comparação à gerência humana tradicional.

Durante a pesquisa, foram realizadas leituras exploratórias de artigos científicos nas áreas de arquiteturas de IA e Modelos de Comportamento Normal (MCN), a fim de entender como identificar anomalias em serviços, recursos e ativos de rede, em geral, como a utilização excessiva de CPU, consumo exagerado de memória RAM, indisponibilidade de serviços web, falhas e comportamentos incomuns relacionados à conectividade de rede. Para os experimentos, utilizamos a plataforma AWS como ambiente de computação em nuvem, na qual será implementada e explorada a ferramenta Openclaw para atuar como um *gateway* de IA com acesso direto ao terminal do sistema operacional.

Diante disto, realizamos a criação de um cenário de estresse, na qual será provocada falha proposital na infraestrutura, como a interrupção de serviços web. O agente de IA deverá realizar o diagnóstico da causa raiz e ajustar os serviços de forma autônoma.

Ao final, analisamos métricas de desempenho, incluindo o uso e consumo da CPU, memória, uso do disco e estatistica de rede. Além disso, a eficácia das ações do agente será avaliada por meio da restauração automática sem a necessidade de intervenção humana.

4.2 Arquitetura do Sistema

Como primeira etapa experimental, foi configurado um ambiente de testes composto por uma instância, que corresponde a uma máquina virtual em um ambiente de computação em nuvem, comporta de recursos computacionais, como vCPU, memória, armazenamento e interfaces de rede.

No ambiente de nuvem, a instância contém o agente de IA e um servidor web Apache. Esse primeiro cenário permitiu validar a comunicação entre o agente de IA e os serviços presentes no ambiente operacional, possibilitando a execução de comandos, a realização de diagnósticos e a verificação da capacidade do agente em interagir com os componentes da infraestrutura. A arquitetura do sistema foi composta pelos seguintes elementos: o servidor *web Apache*, responsável por disponibilizar a aplicação ou serviço acessível externamente a outra instância na qual encontra-se o agente.

O agente, responsável por interpretar comandos, acessar informações do sistema operacional e executar ações administrativas; e o canal de comunicação com o usuário, realizado por meio da plataforma Telegram, permitindo o envio de solicitações e o acompanhamento das respostas fornecidas pelo agente.

O *Openclaw* foi implantado em uma instância do serviço Amazon EC2, pertencente à família T3, com configuração *t3.large*, executando o sistema operacional *Ubuntu Server*. A configuração *t3.large* disponibiliza 2 vCPUs e 8 GiB de memória RAM, recursos considerados suficientes para a execução do *Openclaw* e dos serviços utilizados neste trabalho.

4.3 Arquitetura do Agente

A configuração deste ambiente aconteceu por meio do protocolo de comunicação SSH na instância e a instalação do *openclaw* foi realizada seguindo as orientações disponíveis em sua documentação oficial[1], sendo instalados também o gerenciador de pacotes *Node.js* na versão *latest* (v24.15) para execução do *gateway* que realiza a comunicação entre o modelo de

[1] Disponível em: https://openclaw.ia.br/docs/

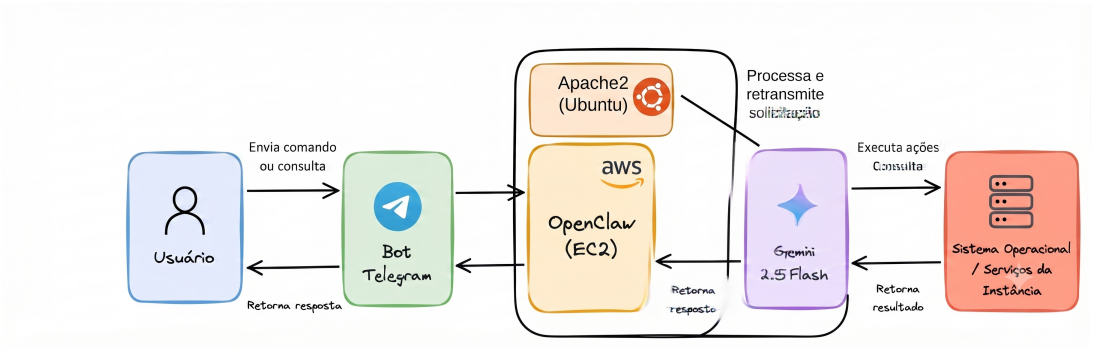
linguagem, as ferramentas disponíveis para o agente e o sistema operacional. Para disponibilizar os serviços na instância.

Como LLM, foi configurado para o *Google/gemini-2.5-flash* durante a instalação do *openclaw*, essa escolha motivou-se devido à disponibilidade de uma camada gratuita de utilização e por apresentar desempenho adequado para a execução de tarefas de interpretação de comandos e geração de respostas.

A comunicação entre o usuário e o agente foi realizada através da integração com um *bot* da plataforma Telegram, escolhido este, por sua facilidade de configuração, ampla disponibilidade e suporte a comunicação em tempo real, permitindo o envio de comandos e consultas de maneira simples e acessível.

A arquitetura final desse sistema, ficou composta pela interação entre usuário que gerência o agente, que faz a solicitação através da plataforma Telegram, o agente executando na instância, é o responsável pelo processamento das requisições e comunicação com o modelo Gemini 2.5 *flash*, e os recursos do ambiente operacional que conseguem ser manipulados pelo agente conforme as permissões configuradas no ambiente, demonstradas na Figura 2.

Figura 2 – Arquitetura do sistema com integração Apache2.



Fonte: Autoria própria (2026).[2]

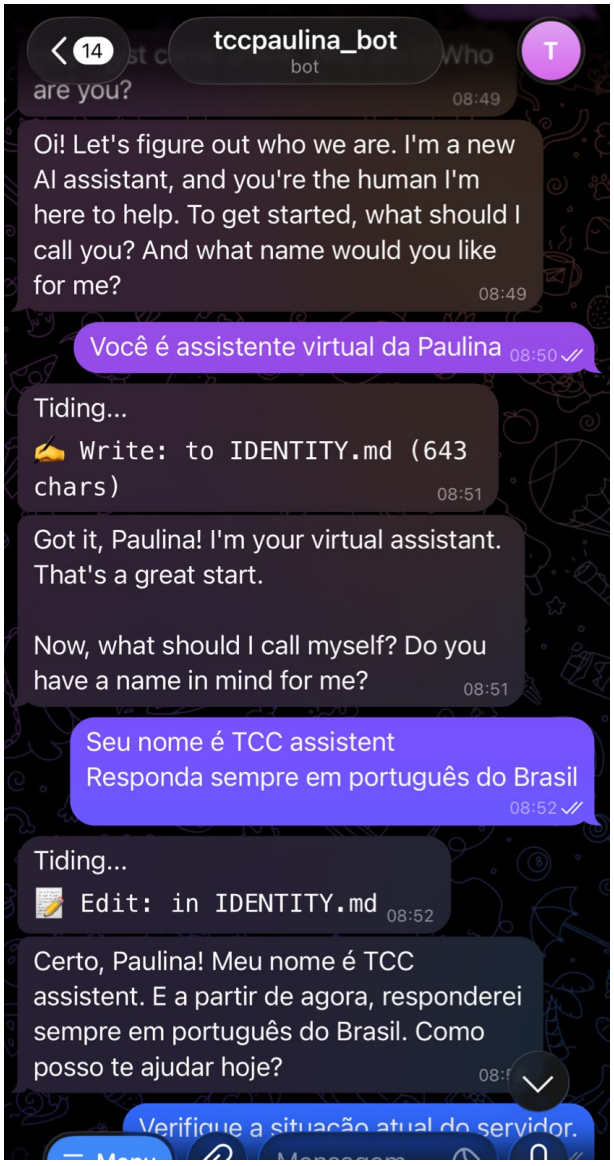
Na Figura 2 é demonstrado o fluxo de comunicação entre o usuário, o *bot* no Telegram e a infraestrutura na AWS, em que o Apache2 atua no processamento das requisições junto ao modelo de IA. Foi realizado uma configuração de validação de identidade para o agente via Telegram.

[2] O diagrama de arquitetura foi elaborado pela autora utilizando ferramenta de inteligência artificial generativa.

5 RESULTADOS E DISCUSSÕES

Neste capítulo, apresentamos os resultados obtidos a partir dos testes práticos realizados com o agente do *Openclaw* usando o modelo *Gemini 2.5 Flash* na infraestrutura AWS. Inicialmente avaliamos a capacidade de interação entre o *bot* do Telegram e o agente. Em seguida, a sua forma de coletar as métricas do servidor e posteriormente experimentação de um cenário de falha no servidor.

Figura 3 – Configuração de identidade do agente.



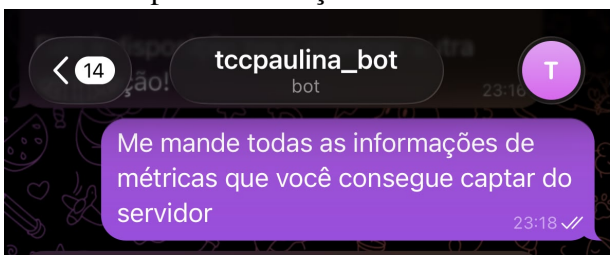
Fonte: Autoria própria (2026).

Com isso, para validar a comunicação em linguagem natural e a capacidade de interpretação do agente, foi enviado via plataforma Telegram, através do *bot*, um comando inicial,

para realização de uma etapa de configuração do agente e estabelecer sua identidade. Conforme demonstrado na Figura 3, a comunicação inicial permitiu definir o papel do agente como assistente virtual, estabelecendo parâmetros operacionais, como a preferência pela língua portuguesa. Esse comportamento demonstra que o sistema é capaz de manter um contexto de interação entre diferentes consultas, facilitando a administração do servidor por meio de linguagem natural.

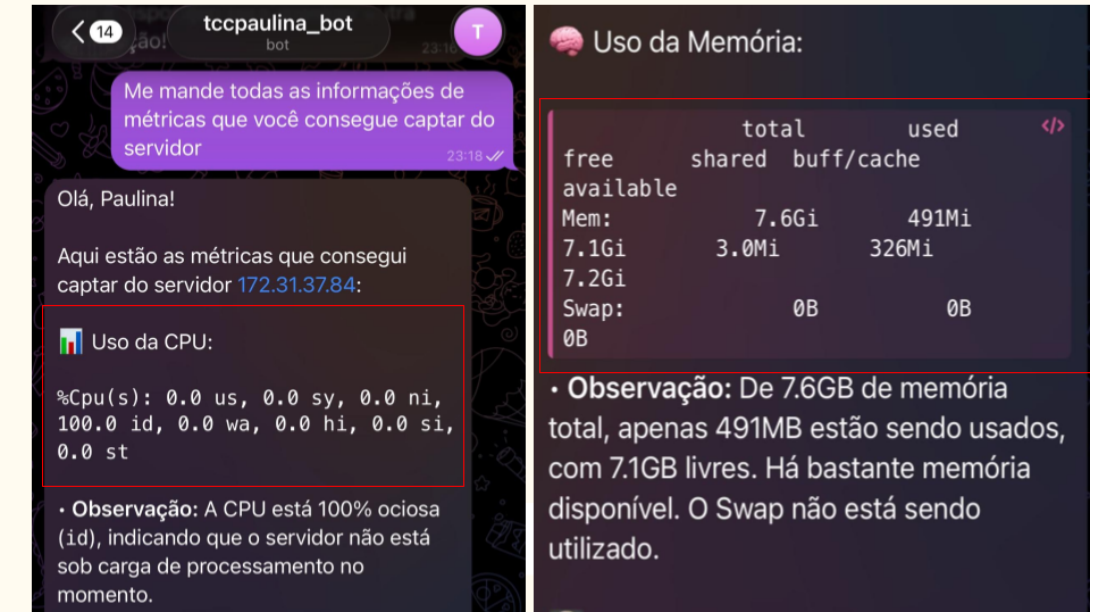
Após a primeira interação com o *bot*, solicitamos ao agente um diagnóstico do servidor com as métricas que configuramos para monitorar dentro das configurações do servidor na instância. A requisição foi interpretada pelo modelo de linguagem, que identificou a solicitação do usuário e executou o *script* responsável pela coleta das informações do sistema operacional, demonstrado na Figura 4.

Figura 4 – Prompt de solicitação de métricas do servidor.



Fonte: Autoria própria (2026).

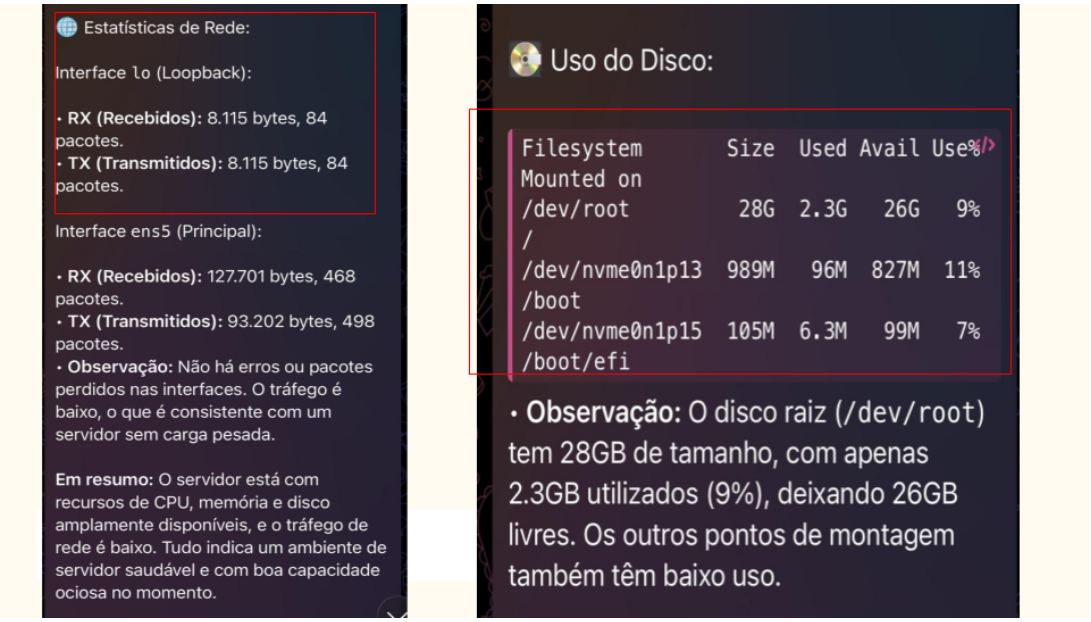
Figura 5 – Resposta do agente referente a coleta das métricas da CPU e Memória.



Fonte: Autoria própria (2026).

Os resultados apresentados nas Figuras 5 e 6 demonstram que o agente foi capaz de

Figura 6 – Resposta do agente referente a coleta das Métricas de estatística de rede e uso do disco.



Fonte: Autoria própria (2026).

apresentar diferentes métricas do sistema em uma única resposta estruturada. Foram disponibilizadas informações referentes ao consumo de CPU, utilização de memória, uso do disco e estatísticas das interfaces de rede. Esse comportamento evidencia que a arquitetura permite consultar informações operacionais sem necessidade de acesso direto ao terminal do servidor.

5.1 Experimento de falha do servidor Apache

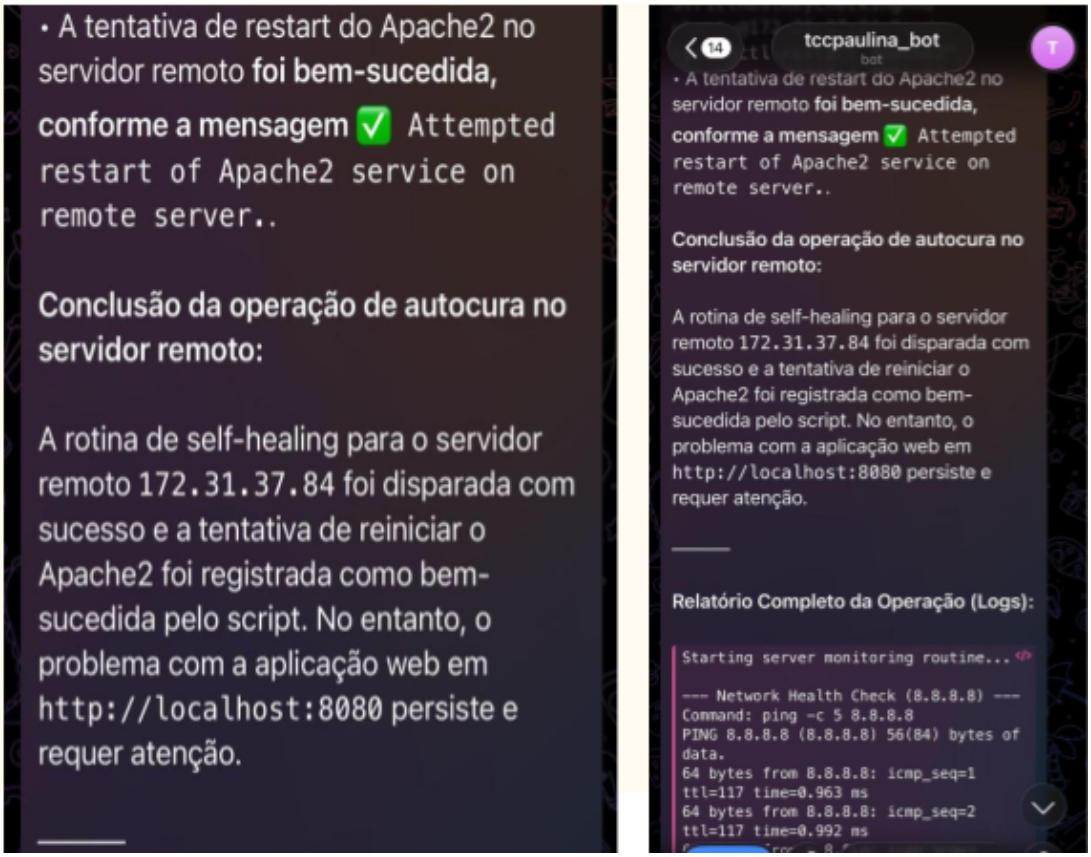
Após a validação da comunicação entre o agente de IA e o ambiente monitorado, foi realizado um experimento com o objetivo de avaliar a capacidade do agente em identificar e auxiliar na restauração de um serviço *Apache*, que foi desativado via comando de terminal, sua restauração ocorreu por meio de um *script* pré configurado no *Openclaw*, ilustrado no Apêndice A.

O experimento consistiu na interrupção proposital do serviço *Apache*, simulando uma indisponibilidade do servidor Web. Após a interrupção, o agente foi consultado por meio do Telegram sobre o estado do servidor. Com base nas informações coletadas do ambiente, o agente identificou que o serviço estava inativo e executou o procedimento de autocura.

Como métricas de avaliação foram consideradas:

- capacidade de identificar corretamente a falha;
- capacidade de executar o procedimento de recuperação;

Figura 7 – Registro do Servidor Apache funcionando após autocura.



Fonte: Autoria própria (2026).

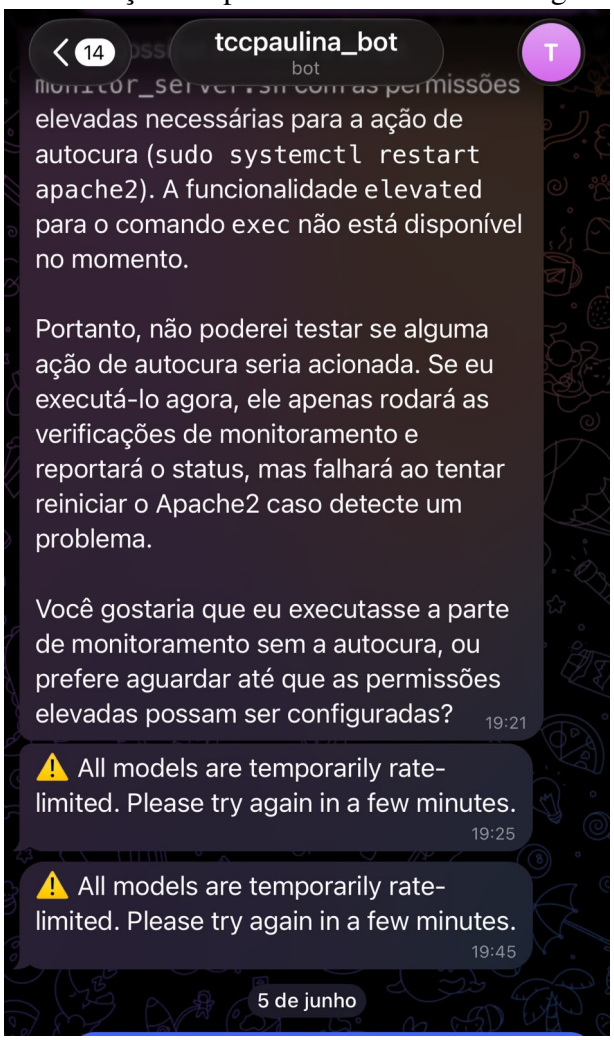
- funcionamento da comunicação entre usuário, Telegram e agente de IA.

Essas métricas foram analisadas conforme a ação era executada pelo *bot* e o serviço voltava, assim como a velocidade ao qual obteve-se retorno. A Figura 7 apresentam quando o servidor foi restaurado. Ao final da execução verificamos que o servidor retornou ao estado operacional, validando a integração entre o modelo de linguagem e a plataforma *Openclaw*.

5.2 Discussões com a literaratura revisada

Os testes realizados demonstraram que a comunicação entre o usuário e o agente de IA ocorreu de maneira que conseguimos atingir as métricas de avaliação propostas. Embora os resultados tenham atendido aos objetivos definidos para o trabalho, uma limitação observada está relacionada ao uso da versão gratuita da API do modelo *Gemini*. Durante a realização dos experimentos, o serviço apresentou restrições na quantidade de requisições permitidas em determinados intervalos de tempo, ocasionando mensagens de limite de uso. demonstrado na Figura 8. Essa limitação não comprometeu a validação da arquitetura proposta, porém restringiu a realização de um número maior de experimentos consecutivos.

Figura 8 – Limitação temporaria do modelo ao atingir os tokens



Fonte: Autoria própria (2026).

Outro aspecto importante observado foi que o agente depende de *scripts* previamente definidos para executar ações administrativas. Dessa forma, o modelo de linguagem atua como uma interface inteligente para interpretação das solicitações e tomada de decisão, enquanto a execução das ações ocorre por meio de procedimentos previamente autorizados, reduzindo riscos associados à execução de comandos arbitrários.

6 CONCLUSÃO

Os resultados obtidos indicam que a arquitetura proposta apresenta potencial para realizar a identificação e recuperação de falhas em serviços de infraestrutura. A integração entre o agente, o Telegram e os mecanismos de automação se mostrou funcional e adequada para o cenário proposto. Sendo possível captar métricas com CPU, memória, uso de disco e estatística de rede, além do agente conseguir monitorar a saúde e o estado do servidor na instância externa. Verificação essa para saber a saúde do sistema e identificar possíveis causas das falhas do servidor e pela necessidade de identificar se é necessário alocar recursos computacionais. Por fim, o agente conseguiu restaurar o estado do servidor.

6.1 Trabalhos futuros

Como trabalhos futuros, pretende-se ampliar as capacidades do agente por meio da implementação de mecanismos de monitoramento contínuo, permitindo a detecção automática de falhas e a execução de ações corretivas sem intervenção humana. Também pretende-se incorporar sistemas de notificação proativa para alertar administradores sobre eventos críticos, expandir o conjunto de procedimentos automatizados para diferentes serviços de infraestrutura e avaliar o desempenho da solução em ambientes distribuídos e de maior escala. Outra possibilidade consiste na utilização de modelos de linguagem com menos restrições de uso ou hospedados localmente.

Declaração de Uso de Inteligência Artificial

Em conformidade com as diretrizes de integridade científica e boas práticas acadêmicas, a autora declara que utilizou ferramentas de IA, incluindo o ChatGPT (OpenAI, modelo GPT-5.5), exclusivamente para auxiliar na melhoria da linguagem, legibilidade e organização deste manuscrito. O uso dessas ferramentas foi realizado sob supervisão humana, e a autora revisou e editou cuidadosamente o conteúdo, assumindo total responsabilidade pelo texto final apresentado.

REFERÊNCIAS

APURI, S.; SMITH, J.; DOE, J. Self-healing infrastructure: Autonomous llm agents for real-time remediation. **International Journal of Intelligent Systems and Applications in Engineering**, v. 12, n. 4, 2024.

DING, M. *et al.* Automating computational chemistry workflows via openclaw and domain-specific skills. **Journal of Chemical Theory and Computation**, ACS Publications, 2026.

Microsoft Research. Aiopslab: A holistic framework to evaluate ai agents for enabling autonomous clouds. In: **Proceedings of International Conference on Autonomic Computing and Self-Organizing Systems (ACSOS)**. [S.l.: s.n.], 2024.

WEIDENER, L. *et al.* From agent-only social networks to autonomous scientific research: Lessons from openclaw and moltbook, and the architecture of clawdlab and beach. science. **arXiv preprint arXiv:2602.19810**, 2026.

ZHAO, W. X. *et al.* A survey on large language model based agents. **arXiv preprint arXiv:2308.11432**, 2023.

APÊNDICE A – *SCRIPT* DE MONITORAMENTO E RECUPERAÇÃO AUTOMÁTICA DA INFRAESTRUTURA

O Apêndice A apresenta o *script Bash* desenvolvido para o monitoramento da infraestrutura utilizada nos experimentos deste trabalho. O *script* realiza verificações periódicas de conectividade da rede, disponibilidade dos serviços locais e remotos e funcionamento do *OpenClaw*. Quando uma anomalia é identificada, são executadas ações automáticas de recuperação, como a reinicialização do serviço Apache, além da coleta de informações de diagnóstico por meio de comandos de análise da rede, das portas em uso e dos registros de erros do sistema operacional.

Código-fonte 1 – *Script* de monitoramento da infraestrutura com Openclaw

```
ALERT_MESSAGE=""
ISSUE_DETECTED=0
FULL_LOG=""

append_to_log() {
    local description="$1"
    local content="$2"
    FULL_LOG+="\n--- $description ---\n"
    FULL_LOG+="$content\n"
    echo "$content"
}

echo "Starting server monitoring routine..."

ALERT_MESSAGE=""
ISSUE_DETECTED=0
FULL_LOG=""

append_to_log() {
    local description="$1"
    local content="$2"
    FULL_LOG+="\n--- $description ---\n"
    FULL_LOG+="$content\n"
    echo "$content"
}
```

```
26
27 echo "Starting server monitoring routine..."
28 FULL_LOG+="Starting server monitoring routine...\n"
29
30 echo "\n--- Network Health Check (8.8.8.8) ---"
31 PING_CMD_OUTPUT_8888=$(ping -c 5 8.8.8.8 2>&1)
32 append_to_log "Command: ping -c 5 8.8.8.8" "$PING_CMD_OUTPUT_8888"
33 LATENCY_8888=$(echo "$PING_CMD_OUTPUT_8888" | grep 'avg/max' | awk -F'/' '{print $5}
       ' | cut -d'.' -f1 || echo "0")
34 PACKET_LOSS_8888=$(echo "$PING_CMD_OUTPUT_8888" | grep 'packet loss' | awk '{print
       $6}' | sed 's/%//' || echo "101")
35
36 if (( $(echo "$LATENCY_8888 > 100" | bc -l) )) || (( $(echo "$PACKET_LOSS_8888 > 5"
        | bc -l) )); then
37     ALERT_MESSAGE+="ERROR: High latency (${LATENCY_8888}ms) or packet loss (${
           PACKET_LOSS_8888}%) to 8.8.8.8 detected.\n"
38     ISSUE_DETECTED=1
39 fi
40
41 echo "\n--- Network Health Check (1.1.1.1) ---"
42 PING_CMD_OUTPUT_1111=$(ping -c 5 1.1.1.1 2>&1)
43 append_to_log "Command: ping -c 5 1.1.1.1" "$PING_CMD_OUTPUT_1111"
44 LATENCY_1111=$(echo "$PING_CMD_OUTPUT_1111" | grep 'avg/max' | awk -F'/' '{print $5}
       ' | cut -d'.' -f1 || echo "0")
45 PACKET_LOSS_1111=$(echo "$PING_CMD_OUTPUT_1111" | grep 'packet loss' | awk '{print
       $6}' | sed 's/%//' || echo "101")
46
47 if (( $(echo "$LATENCY_1111 > 100" | bc -l) )) || (( $(echo "$PACKET_LOSS_1111 > 5"
        | bc -l) )); then
48     ALERT_MESSAGE+="ERROR: High latency (${LATENCY_1111}ms) or packet loss (${
           PACKET_LOSS_1111}%) to 1.1.1.1 detected.\n"
49     ISSUE_DETECTED=1
50 fi
51
52 echo "\n--- OpenClaw Gateway Status (http://localhost:18789) ---"
53 CURL_CMD_OUTPUT_18789=$(curl -Is http://localhost:18789 --connect-timeout 5 2>&1)
54 append_to_log "Command: curl -Is http://localhost:18789 --connect-timeout 5" "
       $CURL_CMD_OUTPUT_18789"
55 HTTP_STATUS_18789=$(echo "$CURL_CMD_OUTPUT_18789" | grep "HTTP" | awk '{print $2}'
```

```
    | head -n 1 || echo "000")
56
57  if [[ "$HTTP_STATUS_18789" != "200" ]]; then
58      ALERT_MESSAGE+="ERROR: OpenClaw Gateway (http://localhost:18789) is not healthy.
            HTTP Status: ${HTTP_STATUS_18789}\n"
59      ISSUE_DETECTED=1
60  fi
61
62  echo "\n--- New Web App Status (http://localhost:8080) ---"
63  CURL_CMD_OUTPUT_8080=$(curl -Is http://localhost:8080 --connect-timeout 5 2>&1)
64  append_to_log "Command: curl -Is http://localhost:8080 --connect-timeout 5" "
        $CURL_CMD_OUTPUT_8080"
65  HTTP_STATUS_8080=$(echo "$CURL_CMD_OUTPUT_8080" | grep "HTTP" | awk '{print $2}' |
        head -n 1 || echo "000")
66
67  if [[ "$HTTP_STATUS_8080" != "200" ]]; then
68      ALERT_MESSAGE+="ERROR: New Web Application (http://localhost:8080) is not
            healthy. HTTP Status: ${HTTP_STATUS_8080}\n"
69      ISSUE_DETECTED=1
70  fi
71
72  echo "\n--- Apache Status (http://localhost:80) ---"
73  CURL_CMD_OUTPUT_80=$(curl -Is http://localhost:80 --connect-timeout 5 2>&1)
74  append_to_log "Command: curl -Is http://localhost:80 --connect-timeout 5" "
        $CURL_CMD_OUTPUT_80"
75  HTTP_STATUS_80=$(echo "$CURL_CMD_OUTPUT_80" | grep "HTTP" | awk '{print $2}' | head
        -n 1 || echo "000")
76
77  if [[ "$HTTP_STATUS_80" != "200" ]]; then
78      ALERT_MESSAGE+="ERROR: Apache (http://localhost:80) is not healthy. HTTP Status:
            ${HTTP_STATUS_80}\n"
79      ISSUE_DETECTED=1
80
81      echo "Attempting to restart Apache2 service..."
82      RESTART_OUTPUT=$(sudo systemctl restart apache2 2>&1)
83      append_to_log "Command: sudo systemctl restart apache2" "$RESTART_OUTPUT"
84
85      if systemctl is-active --quiet apache2; then
86          ALERT_MESSAGE+="INFO: Apache2 service restarted successfully.\n"
```

```
87      else
88          ALERT_MESSAGE+="ERROR: Failed to restart Apache2 service.\n"
89      fi
90  fi
91
92  echo "\n--- Remote Server Status (http://172.31.37.84:80) ---"
93  CURL_CMD_OUTPUT_REMOTE=$(curl -Is http://172.31.37.84:80 --connect-timeout 5 2>&1)
94  append_to_log "Command: curl -Is http://172.31.37.84:80 --connect-timeout 5" "
        $CURL_CMD_OUTPUT_REMOTE"
95  HTTP_STATUS_REMOTE=$(echo "$CURL_CMD_OUTPUT_REMOTE" | grep "HTTP" | awk '{print $2}
        ' | head -n 1 || echo "000")
96
97  if [[ "$HTTP_STATUS_REMOTE" != "200" ]]; then
98      ALERT_MESSAGE+="ERROR: Remote Server (http://172.31.37.84:80) is not healthy.
            HTTP Status: ${HTTP_STATUS_REMOTE}\n"
99      ISSUE_DETECTED=1
100
101     echo "Attempting to restart Apache2 service on remote server..."
102
103     RESTART_OUTPUT_REMOTE=$(ssh -i ~/.ssh/id_agente -o StrictHostKeyChecking=no
            ubuntu@172.31.37.84 "sudo systemctl restart apache2" 2>&1)
104
105     append_to_log "Command: ssh -i ~/.ssh/id_agente -o StrictHostKeyChecking=no
            ubuntu@172.31.37.84 \"sudo systemctl restart apache2\"" "
            $RESTART_OUTPUT_REMOTE"
106
107     if echo "$RESTART_OUTPUT_REMOTE" | grep -q "System has not been booted with
            systemd as init system"; then
108         ALERT_MESSAGE+="WARNING: Could not restart Apache2 on remote server. Systemd
                not found on remote host. Output: $RESTART_OUTPUT_REMOTE\n"
109     elif echo "$RESTART_OUTPUT_REMOTE" | grep -qi "failed"; then
110         ALERT_MESSAGE+="ERROR: Failed to restart Apache2 service on remote server.
                Output: $RESTART_OUTPUT_REMOTE\n"
111     else
112         ALERT_MESSAGE+="INFO: Attempted restart of Apache2 service on remote server.
                Output: $RESTART_OUTPUT_REMOTE\n"
113     fi
114 fi
115
```

```
116 if [ "$ISSUE_DETECTED" -eq 1 ]; then
117     ALERT_MESSAGE+="\n--- Triggering Troubleshooting Routine ---\n"
118 
119     append_to_log "Command: mtr -r -c 5 8.8.8.8" "$(mtr -r -c 5 8.8.8.8 2>&1)"
120     append_to_log "Command: ss -tulpen" "$(ss -tulpen 2>&1)"
121     append_to_log "Command: journalctl -n 20 -p err" "$(journalctl -n 20 -p err
        2>&1)"
122 
123     echo -e "SERVER MONITORING ALERT\n\n$ALERT_MESSAGE\n\n--- Full Logs ---\n\`\`\`\
        n$FULL_LOG\n\`\`\`"
124 else
125     echo "Server health checks passed. No anomalies detected."
126 fi
```