UNIVERSIDADE FEDERAL DO CEARÁ

CAMPUS QUIXADÁ

CURSO DE GRADUAÇÃO EM REDES DE COMPUTADORES

JOSÉ DE ANCHIETA DO NASCIMENTO ALBANO

DESENVOLVIMENTO DE UM TERMINAL MULTI-CLOUD INTELIGENTE UTILIZANDO GERAÇÃO AUMENTADA POR RECUPERAÇÃO (RAG) PARA OPERAÇÕES DE COMPUTAÇÃO EM NUVEM

QUIXADÁ

2025

JOSÉ DE ANCHIETA DO NASCIMENTO ALBANO

DESENVOLVIMENTO DE UM TERMINAL MULTI-CLOUD INTELIGENTE UTILIZANDO GERAÇÃO AUMENTADA POR RECUPERAÇÃO (RAG) PARA OPERAÇÕES DE COMPUTAÇÃO EM NUVEM

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Redes de Computadores do Campus Quixadá da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de Tecnólogo em Redes de Computadores.

Orientador: Prof. Dr. João Marcelo Uchôa de Alencar.

QUIXADÁ

2025

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Sistema de Bibliotecas
Gerada automaticamente pelo módulo Catalog, mediante os dados fornecidos pelo(a) autor(a)

A286d Albano, José de Anchieta do Nascimento.
Desenvolvimento de um terminal multi-cloud inteligente utilizando geração aumentada por recuperação (RAG) para operações de computação em nuvem. / José de Anchieta do Nascimento Albano. – 2025.
67 f. : il. color.

Trabalho de Conclusão de Curso (graduação) – Universidade Federal do Ceará, Campus de Quixadá, Curso de Redes de Computadores, Quixadá, 2025.
Orientação: Prof. Dr. João Marcelo Uchôa de Alencar..

1. Computação em Nuvem. 2. Terminal Inteligente. 3. Multi-cloud . 4. Geração Aumentada por Recuperação. 5. Modelos de Linguagem. I. Título.

CDD 004.6

JOSÉ DE ANCHIETA DO NASCIMENTO ALBANO

DESENVOLVIMENTO DE UM TERMINAL MULTI-CLOUD INTELIGENTE UTILIZANDO GERAÇÃO AUMENTADA POR RECUPERAÇÃO (RAG) PARA OPERAÇÕES DE COMPUTAÇÃO EM NUVEM

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Redes de Computadores do Campus Quixadá da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de Tecnólogo em Redes de Computadores.

Aprovada em: 29 de Julho de 2025

BANCA EXAMINADORA

Prof. Dr. João Marcelo Uchôa de Alencar (Orientador)
Universidade Federal do Ceará (UFC)

Prof. Dr. Wagner Guima- rães Al-Alam
Universidade Federal do Ceará - UFC

Prof. Dr. Regis Pires Magalhães,
Universidade Federal do Ceará - UFC

À minha mãe, minhas irmãs e meus amigos, por sempre acreditarem em mim.

AGRADECIMENTOS

Agradeço a minha mãe, Fátima, por sempre me apoiar e me dar palavras de motivação durante essa jornada, assim como minhas irmãs Quezia e Ávila. Sou grato ao Prof. Dr. João Marcelo Uchôa de Alencar, por ter me guiado durante o desenvolvimento deste trabalho. Agradeço a Deus por me dar forças para continuar lutando, pois houve instantes em que quase desisti. Sou grato ao padre Thiago por todos os conselhos e orientações, que foram muito importantes. Agradeço aos integrantes da banca examinadora, Prof. Dr. Wagner Guimarães Al-Alam e Prof. Dr. Regis Pires Magalhães, pelas valiosas observações que contribuíram para melhorar esta pesquisa. Sou grato aos amigos e colegas que fiz durante meu tempo na Universidade Federal do Ceará, assim como ao corpo docente. E, por último, agradeço aos meus colegas Franciel e Danilo, pela convivência e por todos os momentos que compartilhamos, especialmente ao primeiro, que me ajudou muito.

"Se for para desistir, desista de ser fraco."

(Fernando Cagnin)

RESUMO

Este estudo apresenta o desenvolvimento de um terminal inteligente que utiliza a técnica de *Retrieval-augmented generation* (RAG) para apoiar ambientes *multi-cloud*, aplicando comandos de *Command-Line Interface* (CLI) extraídos das documentações oficiais da *Amazon Web Services* (AWS) e da *Google Cloud Platform* (GCP). A aplicação, desenvolvida com o modelo *LLaMA* e executada localmente por meio do *Ollama*, incorporou *embeddings* vetoriais, reranking semântico baseado em Maximal Marginal Relevance (MMR) e as bibliotecas *LangChain* e *Streamlit*, com o objetivo de criar uma interface *web* acessível. O protótipo foi testado por 42 participantes, que realizaram tarefas de provisionamento de instâncias nos ambientes AWS e GCP, e as respostas foram analisadas quanto à precisão técnica, completude, clareza, utilidade e relevância. Os resultados indicaram avaliações majoritariamente positivas, destacando a capacidade do sistema de fornecer respostas contextualizadas, embora tenham sido observadas limitações em termos de precisão e adaptação a erros. A pesquisa evidenciou a viabilidade de um assistente técnico baseado em RAG executado localmente, mas também apontou a necessidade de modelos mais robustos e bases de conhecimento mais organizadas. Conclui-se que a abordagem RAG apresenta um potencial significativo para transformar a forma como interagimos com sistemas *multi-cloud*, servindo como base para futuras melhorias em precisão, explicabilidade e usabilidade.

Palavras-chave: Computação em Nuvem; *Multi-cloud*; Geração Aumentada por Recuperação; Modelos de Linguagem; Terminal Inteligente.

ABSTRACT

This study presents the development of an intelligent terminal that employs the Retrieval-Augmented Generation (RAG) technique to support multi-cloud environments, executing Command-Line Interface (CLI) commands extracted from the official documentation of Amazon Web Services (AWS) and Google Cloud Platform (GCP). The application, developed using the LLaMA model and run locally via Ollama, incorporated vector embeddings, semantic reranking based on Maximal Marginal Relevance (MMR), and the LangChain and Streamlit libraries to provide an accessible web interface. The prototype was tested by 42 participants who performed instance provisioning tasks in AWS and GCP environments. Their responses were evaluated according to five criteria: technical accuracy, completeness, clarity, usefulness, and relevance. The results indicated mostly positive evaluations, highlighting the system's ability to provide contextualized responses, although limitations were observed regarding precision and error adaptation. The research demonstrated the feasibility of a locally executed RAG-based assistant, while also pointing to the need for more robust models and better-curated knowledge bases. The study concludes that the RAG approach has significant potential to transform the way users interact with multi-cloud systems, offering a foundation for future improvements in precision, explainability, and usability.

Keywords: Cloud Computing; Multi-cloud; Retrieval-Augmented Generation; Language Models; Intelligent Terminal.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE CÓDIGOS-FONTE

LISTA DE ABREVIATURAS E SIGLAS

AKS	Azure Kubernetes Service
AMI	*Amazon Machine Images*
API	*Application Programming Interface*
AWS	*Amazon Web Services*
Azure	*Microsoft Azure*
CLI	*Command-Line Interface*
EC2	*Elastic Compute Cloud*
EKS	Elastic Kubernetes Service
GAN	*Generative Adversarial Networks*
GCE	*Google Compute Engine*
GCP	*Google Cloud Platform*
GKE	Google Kubernetes Engine
IA	inteligência artificial
IaaS	Infraestrutura como Serviço
IaC	Infrastructure as Code
LGPD	Lei Geral de Proteção de Dados
LLM	*Large Language Models*
MDP	*Markov Decision Processes*
MMR	Maximal Marginal Relevance
NAT	Network Address Translation
NIST	Instituto Nacional de Padrões e Tecnologia
PaaS	Plataforma como Serviço
PLN	Processamento de Linguagem Natural
RAG	*Retrieval-augmented generation*
RDP	*Remote Desktop Protocol*
RGB	*Retrieval Augmented Generation Benchmark*
SaaS	Software como Serviço
SSH	*Secure Shell*
TI	Técnologia da Informação
UFC	Universidade Federal do Ceará

VAE *Variational Autoencoders*

VPC *Virtual Private Cloud*

SUMÁRIO

1 INTRODUÇÃO

A computação em nuvem transformou profundamente a forma como organizações acessam, processam e gerenciam dados, consolidando-se como uma das maiores inovações tecnológicas das últimas décadas Sousa *et al.* (2009). Estima-se que esse mercado movimente cerca de 680 bilhões de dólares em 2024, com um crescimento médio anual de (16,4%) e é projetado para atingir 1,44 trilhão de dólares em 2029 Intelligence (2024). Este avanço reflete a busca por soluções escaláveis e acessíveis, especialmente em cenários que requerem alta capacidade computacional e integração global.

Com a expansão acelerada dos serviços em nuvem, tornou-se comum a adoção de estratégias *multi-cloud*, que consistem na utilização simultânea de diferentes provedores de computação em nuvem. Essa abordagem visa garantir maior disponibilidade, resiliência e redução de custos operacionais, além de mitigar os riscos relacionados à dependência de um único fornecedor Chuang e Chen (2024). Ao combinar provedores distintos, as organizações podem selecionar soluções específicas que melhor atendam às suas demandas técnicas e financeiras.

Apesar dos benefícios, a adoção do modelo *multi-cloud* impõe desafios consideráveis. Cada provedor possui ferramentas, *Application Programming Interface* (API), padrões e requisitos próprios, o que torna a integração e o gerenciamento de recursos significativamente mais complexos. Soluções como o *Kubernetes*[1] são frequentemente empregadas para facilitar essa integração, promovendo a interoperabilidade entre ambientes e a automação de tarefas Chuang e Chen (2024). No entanto, a configuração e manutenção desses ambientes continuam exigindo esforço técnico significativo para garantir segurança, desempenho e eficiência.

Diante da crescente complexidade das arquiteturas *multi-cloud*, observa-se um movimento em direção à adoção de soluções mais inteligentes. A integração de inteligência artificial (IA), em especial de modelos de linguagem de grande escala *Large Language Models* (LLM), tem se mostrado promissora para automatizar processos e gerar respostas contextualizadas a partir de documentação técnica e dados externos Chen *et al.* (2024). A aplicação de IA nesse contexto pode contribuir para o provisionamento automático de recursos, otimização de comandos e melhoria da experiência do usuário.

A técnica conhecida como Geração Aumentada por Recuperação - RAG consiste em combinar o uso de LLMs com bases externas de conhecimento, possibilitando que os modelos acessem informações atualizadas no momento da geração da resposta. Essa abordagem tem se

[1] https://kubernetes.io/

mostrado eficaz na redução de respostas imprecisas, também conhecidas como “alucinações”, além de elevar a relevância e a precisão das informações fornecidas Chen *et al.* (2024). Em ambientes *multi-cloud*, o uso de RAG pode contribuir para uma gestão mais eficiente e segura dos recursos computacionais.

Este trabalho propõe o desenvolvimento de uma aplicação *web* interativa, denominada terminal inteligente, voltada para ambientes *multi-cloud*. A aplicação será desenvolvida utilizando o *Streamlit*[2], permitindo a execução de operações críticas — como a criação de máquinas virtuais e o gerenciamento de recursos — com foco em segurança, escalabilidade e eficiência operacional. O terminal atuará como uma ponte entre o usuário e os fornecedores de nuvem, fornecendo respostas contextuais obtidas da documentação oficial e criadas por meio de um modelo LLM incorporado a uma estrutura RAG. É fundamental ressaltar que a aplicação não realizará os comandos diretamente nas nuvens, mas terá a função de produzir *scripts* ou orientações validadas, delegando a execução ao usuário, dessa forma minimizando riscos operacionais e ampliando o controle sobre o procedimento.

1.1 Objetivo geral

Desenvolver uma aplicação web baseada na arquitetura RAG, capaz de auxiliar usuários na execução de operações em ambientes *multi-cloud* por meio de respostas contextualizadas, com base na documentação oficial dos principais provedores de computação em nuvem.

1.2 Objetivos específicos

- Identificar os principais provedores e suas respectivas documentações oficiais de referência.
- Preparar o ambiente computacional para execução de modelos LLM integrados à arquitetura RAG.
- Projetar e implementar um cenário de testes *multi-cloud*, com tarefas práticas representativas.
- Disponibilizar a aplicação para testes práticos com usuários em ambiente controlado.
- Avaliar o desempenho da ferramenta com base em métricas como acurácia das respostas, relevância, tempo de resposta e percepção dos usuários.

Este trabalho está estruturado da seguinte forma: o Capítulo 2 apresenta a fundamen-

[2] https://streamlit.io/

tação teórica sobre os conceitos-chave utilizados; o Capítulo 3 discute os trabalhos relacionados; o Capítulo 4 descreve a metodologia adotada, incluindo o desenvolvimento da aplicação, os testes realizados e os critérios de avaliação; o Capítulo 5 apresenta e analisa os resultados obtidos a partir da aplicação da ferramenta; por fim, o Capítulo 6 apresenta as considerações finais e sugestões para trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo apresenta os fundamentos teóricos que sustentam o desenvolvimento da aplicação proposta. Inicialmente em 2.1, discute-se o conceito de computação em nuvem, seus modelos de serviço e principais provedores. Em seguida 2.2, aborda-se a estratégia *multi-cloud*, suas características, desafios e métodos de implementação. Por fim 2.3, são explorados os conceitos relacionados a modelos de linguagem LLMs, estratégias de RAG e ferramentas que possibilitam a construção de soluções inteligentes baseadas nessas tecnologias.

2.1 Computação em Nuvem

A computação em nuvem consiste em um modelo de entrega de recursos computacionais sob demanda, como servidores, armazenamento, redes e aplicações, por meio da internet e com mínima intervenção humana. Essa definição fundamenta-se no modelo proposto pelo Instituto Nacional de Padrões e Tecnologia (NIST), que destaca características como autosserviço sob solicitação, acesso amplo à rede, agrupamento de recursos, elasticidade rápida e mensuração de serviços.

Essa abordagem tem revolucionado a forma como empresas e usuários acessam infraestrutura de Técnologia da Informação (TI), substituindo a aquisição e manutenção de data centers locais por serviços sob demanda, pagos conforme o uso. Grandes provedores como AWS, *Microsoft Azure* (Azure) e GCP disponibilizam soluções que variam desde instâncias de máquinas virtuais até serviços gerenciados de IA e análise de dados.

Os tipos de implementação da computação em nuvem abrangem as categorias pública, privada, comunitária e híbrida, sendo selecionadas de acordo com as exigências de segurança, escalabilidade e controle. Ademais, os serviços são geralmente estruturados em três níveis: IaaS, PaaS e SaaS, conforme será detalhado a seguir.

2.1.1 Modelos de Serviço em Nuvem:IaaS, PaaS e SaaS

O modelo IaaS fornece os recursos fundamentais da infraestrutura, como servidores, redes, armazenamento e sistemas operacionais, sobre os quais o usuário possui controle direto. Um exemplo típico é o *Amazon* EC2, que permite a criação e configuração de instâncias virtuais sob demanda Amazon Web Services (2025). Já o modelo PaaS disponibiliza um ambiente completo de desenvolvimento e implantação, abstraindo o gerenciamento da infraestrutura.

Plataformas como *Google AppEngine*[1], *Salesforce*[2] e *AWS Elastic Beanstalk*[3] permitem que os desenvolvedores foquem apenas na lógica da aplicação, enquanto o provisionamento de servidores, bancos de dados e sistemas operacionais é automatizado.

Por fim, no modelo SaaS, aplicações completas são acessadas diretamente pela internet, normalmente via navegador, sem instalação local. Exemplos incluem *Gmail*, *Google Drive* e *Dropbox*, que oferecem soluções prontas com atualização automática e alta disponibilidade Carissimi (2015).

Neste trabalho, apenas o modelo IaaS foi efetivamente utilizado, por meio da configuração de instâncias EC2 via AWS CLI. Os modelos PaaS e SaaS são apresentados aqui por completude conceitual, mas não foram explorados na implementação prática da aplicação.

2.1.2 Amazon EC2

O *Amazon* EC2 representa um dos principais serviços IaaS da AWS. Ele oferece flexibilidade na criação e gerenciamento de instâncias, com diversas opções de sistemas operacionais, tipos de máquina e configurações de rede. O usuário pode interagir com o EC2 por meio de interfaces gráficas, como o console da AWS, ou via linha de comando, utilizando a AWS CLI Amazon Web Services (2025).

O processo de criação de uma instância EC2 envolve a seleção de uma *Amazon Machine Images* (AMI), que representa uma imagem de disco contendo o sistema operacional, bibliotecas e aplicações necessárias ao funcionamento da máquina virtual. As AMIs podem ser padrão, personalizadas ou disponibilizadas por terceiros. O acesso às instâncias ocorre por meio de protocolos como o *Secure Shell* (SSH), em sistemas Linux, ou *Remote Desktop Protocol* (RDP), em ambientes *Windows*. Essa flexibilidade permite replicar ambientes padronizados de forma ágil e segura, otimizando processos de desenvolvimento e implantação.

A configuração de rede no EC2 é realizada por meio do *Virtual Private Cloud* (VPC), que possibilita a criação de redes virtuais isoladas dentro da infraestrutura da AWS. O VPC permite a definição de sub-redes, rotas, endereços IP, grupos de segurança e *gateways* de internet, proporcionando controle granular sobre o tráfego interno e externo das instâncias Amazon Web Services (2025). Essa arquitetura contribui para a segurança, segmentação e integração de ambientes de nuvem híbrida ou distribuída, sendo amplamente adotada em arquiteturas

[1] https://cloud.google.com/appengine/docs
[2] https://www.salesforce.com/platform/
[3] https://docs.aws.amazon.com/elastic-beanstalk/

corporativas de missão crítica. Detalhes práticos sobre a criação de instâncias EC2 e equivalentes na GCP utilizados nesta pesquisa estão documentados no Apêndice A.

2.2 *Multi-cloud*

A estratégia *multi-cloud* refere-se ao uso simultâneo de serviços de computação em nuvem provenientes de diferentes provedores, com o objetivo de distribuir cargas de trabalho, mitigar riscos de dependência de um único fornecedor (*vendor lock-in*), aumentar a disponibilidade de sistemas e otimizar recursos operacionais Karthik *et al.* (2014). Em contraste com o modelo de nuvem única, o ambiente *multi-cloud* permite que cada componente da aplicação seja alocado em plataformas distintas, aproveitando os pontos fortes de cada provedor, como AWS, Azure e GCP.

Entre os principais benefícios dessa abordagem estão a escalabilidade dinâmica, a conformidade com regulamentações regionais, o aumento da resiliência a falhas e a otimização de custos, permitindo que diferentes regiões ou serviços sejam explorados estrategicamente (Moraes, 2021). No entanto, ambientes *multi-cloud* também apresentam desafios significativos, como a necessidade de interoperabilidade entre sistemas heterogêneos, o gerenciamento de segurança unificada e a orquestração eficiente de redes e dados entre plataformas com arquiteturas distintas

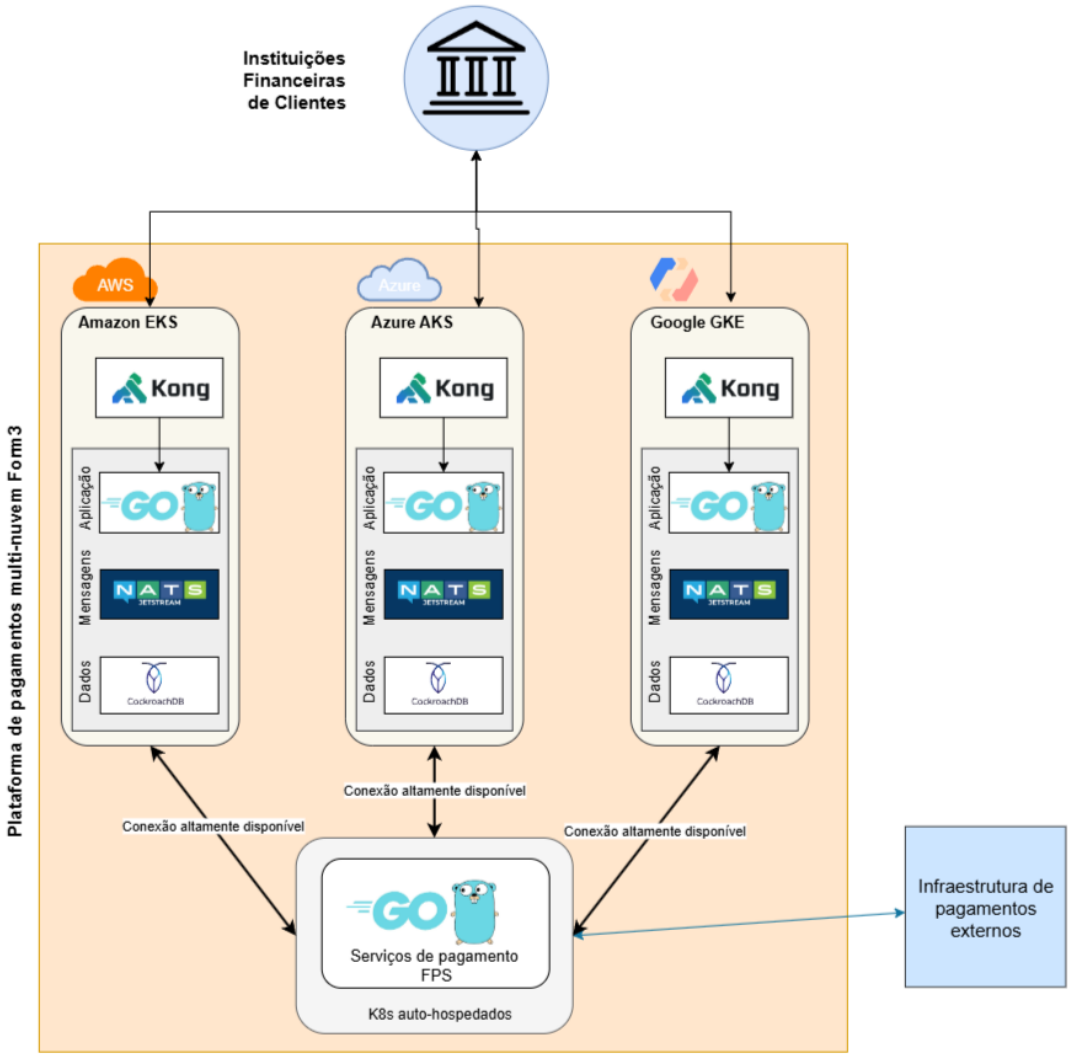
Empresas do setor financeiro, como *Form3*[4] e *Stake*[5], já adotam arquiteturas *multi-cloud* para garantir continuidade de serviços e conformidade legal. A *Form3*, por exemplo, utiliza o banco de dados distribuído *CockroachDB*[6] sobre infraestrutura compartilhada entre AWS, GCP e Azure, assegurando operação contínua mesmo em caso de falhas regionais. A *Stake* aplica estratégia semelhante para armazenar dados de forma distribuída, respeitando exigências de localização e proteção de dados pessoais. Tais exemplos evidenciam a aplicabilidade da abordagem *multi-cloud* em contextos reais de missão crítica.

A Figura 1 representa a arquitetura *multi-cloud* adotada pela empresa *Form3*, utilizada para prover serviços de pagamento em tempo real com alta disponibilidade e tolerância a falhas. A infraestrutura está distribuída entre três provedores distintos — AWS, GCP e Azure —, cada um operando *clusters* gerenciados via *Kubernetes* (Elastic Kubernetes Service (EKS), Azure Kubernetes Service (AKS) e Google Kubernetes Engine (GKE), respectivamente). Cada

[4] https://www.form3.tech/
[5] https://hellostake.com/au
[6] https://www.cockroachlabs.com/

ambiente executa serviços de roteamento por meio da API *Kong*[7], além de ferramentas como Network Address Translation (NAT), bancos de dados e serviços internos. No centro da arquitetura, encontra-se a aplicação de serviços de pagamento. Essa configuração permite que, mesmo diante da indisponibilidade de uma nuvem, os serviços continuem operando através das demais, garantindo resiliência e continuidade.

Figura 1 – Infraestrutura *Multi-cloud* Form3.



Fonte: Adaptado de www.cockroachlabs.com.

2.2.1 Características da abordagem Multi-cloud

A computação *multi-cloud* reúne um conjunto de características essenciais que a tornam atrativa para empresas que demandam alta flexibilidade, disponibilidade e controle. Entre as mais relevantes destacam-se: escalabilidade, distribuição inteligente de carga, interoperabilidade

[7] https://konghq.com/

entre provedores e segurança integrada.

A escalabilidade permite o ajuste dinâmico de recursos computacionais conforme a demanda, reduzindo custos e aumentando a disponibilidade. Já a distribuição inteligente de carga possibilita o balanceamento de serviços entre diferentes provedores, o que evita sobrecargas e melhora o desempenho global dos sistemas. A interoperabilidade busca superar os desafios de integração entre APIs, formatos e serviços proprietários de cada provedor, sendo frequentemente viabilizada por ferramentas como *Kubernetes*, que oferece uma camada de abstração sobre os ambientes, e *frameworks* como *Crossplane*[8], *Terraform*[9] e *Pulumi*[10] que permitem o gerenciamento unificado da infraestrutura em diferentes nuvens. Por fim, a segurança integrada envolve o uso combinado de criptografia, autenticação federada, protocolos seguros e conformidade com normas como a Lei Geral de Proteção de Dados (LGPD).

Apesar de serem aspectos centrais da computação *multi-cloud*, essas características não foram implementadas no projeto desenvolvido, pois este se concentrou em uma aplicação de escopo reduzido, com execução local e sugestões de provisionamento por provedor. Dessa forma, os conceitos foram abordados aqui com fins teóricos e motivacionais, contribuindo para o entendimento do contexto em que a ferramenta proposta poderia futuramente ser inserida.

2.2.2 Implementação Multi-cloud

A implementação de uma estratégia *multi-cloud* requer planejamento detalhado, definição clara de responsabilidades entre provedores e uso de ferramentas que garantam integração fluida entre os ambientes. Uma abordagem comum consiste na alocação distribuída de microsserviços em diferentes nuvens públicas, utilizando os recursos e especializações de cada uma Moraes (2021). Por exemplo, uma organização pode hospedar sua API principal na AWS, utilizar o GCP para análise de dados em larga escala e delegar a autenticação e controle de identidades ao Azure. Essa distribuição permite maior eficiência, redução de custos e melhor aproveitamento das funcionalidades específicas de cada plataforma.

Para viabilizar essa arquitetura heterogênea, são utilizadas ferramentas de orquestração e gerenciamento que facilitam o controle de infraestrutura distribuída Moraes (2021). O *Kubernetes*, nesse contexto, atua como uma camada de abstração para o gerenciamento de contêineres, podendo ser implementado em diferentes nuvens por meio de serviços como *Amazon EKS*,

[8] https://www.crossplane.io/
[9] https://developer.hashicorp.com/terraform
[10] https://www.pulumi.com/

Azure AKS ou *Google GKE*. Além disso, tecnologias como *service mesh* (malha de serviços) — com destaque para o o *Istio*[11] — permitem gerenciar a comunicação entre microsserviços de forma segura, resiliente e observável, independente do provedor onde estejam alocados.

Outras soluções como *Terraform*, *Crossplane* e *Pulumi* têm funções importantes na automação do provisionamento, possibilitando que toda a infraestrutura seja expressa como código Infrastructure as Code (IaC), favorecendo a portabilidade, o versionamento e a reprodutibilidade. A integração dessas ferramentas torna possível a formação de ambientes *multi-cloud* robustos, altamente escaláveis e aptos a satisfazer exigências avançadas de disponibilidade e governança.

2.3 Modelos de Linguagem e Estratégias de Geração

Nesta seção, começamos com os princípios da LLM, seguidos das táticas que buscam aumentar sua eficiência, como a engenharia de *prompt* e a RAG. Para encerrar, apresentamos as ferramentas e modelos adotadas neste projeto, que possibilitam a aplicação efetiva dessas tecnologias.

2.3.1 Grandes Modelos de Linguagem

Denominados como LLM, são sistemas de inteligência artificial desenvolvidos para analisar e produzir linguagem natural de maneira contextual. Estas ferramentas são predominantemente fundamentadas na arquitetura *Transformer*, que emprega mecanismos de auto atenção para reconhecer relações sintáticas e semânticas entre as palavras dentro de um texto. LLMs têm sido utilizados em várias atividades, como geração automática de texto, tradução, suporte ao cliente, respostas a perguntas e, mais recentemente, auxílio na programação de código Roffo (2024).

Apesar de seu desempenho expressivo, os LLMs apresentam limitações, como a possibilidade de gerar respostas factualmente incorretas, dificuldade em lidar com dados atualizados e problemas em tarefas que exigem raciocínio lógico complexo ou informações externas. Para superar essas limitações, foram desenvolvidas estratégias como o *fine-tuning* supervisionado, a engenharia de *prompt*, RAG Roffo (2024).

[11] https://istio.io/latest/about/service-mesh/

2.3.2 Engenharia de prompt

A Engenharia de *Prompt* (*Prompt Engineering*) refere-se ao processo de construção estratégica das entradas fornecidas ao LLM, com o objetivo de maximizar a qualidade e a relevância das respostas geradas. Técnicas como o *Zero-shot Prompting* (sem exemplos), *Few-shot Prompting* (com exemplos) e o *Chain-of-Thought Prompting* (encadeamento do raciocínio) têm sido amplamente utilizadas para melhorar o desempenho dos modelos em tarefas específicas Wei *et al.* (2022). Além disso, *prompts* multimodais, que combinam texto com imagens ou outros formatos de dados, expandem a aplicabilidade dos LLMs para contextos mais complexos.

Essas técnicas são particularmente relevantes em ambientes onde não há tempo ou recursos suficientes para re-treinamento, pois permitem adaptar dinamicamente o comportamento do modelo com base apenas na estrutura da entrada. No contexto deste trabalho, a engenharia de *prompt* é utilizada em conjunto com o *framework LangChain*[12], permitindo estruturar perguntas do usuário de forma eficiente e orientada à recuperação de informações específicas em bases externas Zhao *et al.* (2024).

2.3.3 Geração Aumentada por Recuperação (RAG)

O RAG é uma técnica que combina mecanismos de recuperação de documentos com a geração de texto via LLM. Essa abordagem permite que o modelo acesse informações externas atualizadas no momento da inferência, aumentando a precisão e a relevância das respostas, especialmente em domínios com dados dinâmicos ou especializados. Conforme destacado no artigo Zhao *et al.* (2024), ao invés de se basear apenas no conhecimento interno do modelo, o RAG faz consultas em uma base vetorial que possui trechos textuais relevantes, que são inseridos dinamicamente no *prompt* utilizado na geração final da resposta. O funcionamento do RAG envolve três etapas principais:

1. Recuperação, em que a pergunta do usuário é convertida em uma representação vetorial e utilizada para buscar documentos relevantes em uma base vetorial;
2. Fusão, onde os resultados da busca são combinados com o *prompt* original;
3. Geração, em que o modelo de linguagem processa o *prompt* enriquecido para produzir uma resposta final.

Essa estrutura permite que o modelo produza respostas mais coerentes e embasadas

[12] https://www.langchain.com/

em conteúdos externos.

Durante a etapa de recuperação, é comum aplicar a técnica de *top-k retrieval*, em que apenas os k documentos mais relevantes são selecionados para compor o *prompt* expandido. Essa filtragem é realizada com base nos escores de similaridade calculados entre a consulta e os documentos indexados Zhao *et al.* (2024). No presente trabalho, foi adotado o valor de $k = 10$, de modo que os dez trechos mais semanticamente semelhantes foram utilizados como base para cada geração de resposta. Essa abordagem visa garantir um bom equilíbrio entre cobertura informacional e limitação da janela de contexto do modelo.

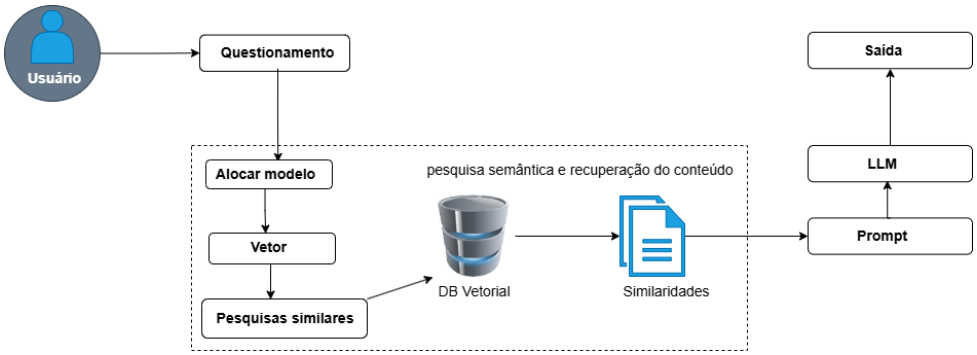
Um elemento fundamental da arquitetura RAG é a utilização de *embeddings*, que são representações vetoriais densas de palavras, frases ou documentos, criadas por modelos treinados para compreender o significado semântico dos textos Mikolov *et al.* (2013b). Essas representações possibilitam a avaliação da similaridade entre diferentes entradas com base em métricas matemáticas, como a distância euclidiana ou a similaridade de cosseno. Ao converter uma consulta textual em um vetor, torna-se viável realizar buscas semânticas em bancos de dados vetoriais, identificando conteúdos que abordam temas semelhantes, mesmo que utilizem vocabulários distintos.

A Figura 2 ilustra esse fluxo arquitetural adotado neste projeto. O processo inicia-se com o envio de um questionamento por parte do usuário, que é transformado em vetor por meio de um modelo de *embedding*. Em seguida, esse vetor é utilizado para consultar uma base de dados vetorial que armazena conteúdos previamente indexados — como trechos técnicos de documentação de serviços em nuvem. O componente de busca retorna os segmentos mais semanticamente similares ao vetor de entrada, os quais são representados na imagem como "Similaridades". Esses documentos são incorporados ao *prompt* original, formando um *prompt* expandido, que é então encaminhado ao modelo LLM. Por fim, o LLM realiza a geração da resposta baseada nesse conjunto enriquecido de informações, devolvendo uma saída textual ao usuário.

No âmbito deste projeto, o conjunto de textos empregado no processo de recuperação constituiu-se por documentos formatados em *Markdown*[13], meticulosamente estruturados e verificados manualmente a partir das documentações oficiais da AWS e da GCP. Em vez de incluir diretamente grandes quantidades de texto não processado, decidiu-se adotar uma abordagem curatorial, na qual cada comando CLI pertinente foi separado, previamente testado

[13] https://www.markdownguide.org/

Figura 2 – Estrutura base RAG.



Fonte: Adaptado de www.elastic.co.

e documentado com seus principais parâmetros e exemplos funcionais. Essa estratégia teve como objetivo minimizar ruídos semânticos, assegurar a pertinência das respostas produzidas e aprimorar a recuperação contextualizada pelo modelo.

Apesar de seus benefícios, a arquitetura RAG apresenta limitações importantes. Entre elas estão a sensibilidade a documentos ruidosos, a dependência de boas fontes de recuperação e a dificuldade em interpretar corretamente *prompts* com múltiplas intenções. Além disso, diferentes configurações de recuperação (por exemplo, número de documentos, balanceamento entre diversidade e similaridade) podem impactar significativamente a qualidade das respostas. No presente trabalho, essas variações foram mitigadas por meio de uma curadoria manual dos documentos e o uso de re-ranqueamento semântico. No entanto, métricas formais como precisão, cobertura e robustez contrafactual, conforme discutido por Chen *et al.* (2024), não foram aplicadas diretamente, sendo recomendadas para trabalhos futuros.

2.3.4 LangChain

A biblioteca *LangChain* foi criada com o objetivo de simplificar a elaboração de aplicações que integram LLMs com diversas fontes de dados, como APIs, bancos de dados e arquivos locais. Um de seus principais atributos é o suporte à estratégia RAG, possibilitando a conexão com bancos de dados vetoriais, recuperadores, mecanismos de busca e processos condicionais. O *LangChain* elimina a complexidade associada ao gerenciamento de múltiplos elementos, disponibilizando ferramentas modulares que permitem a combinação de recuperação e geração em uma única aplicação.

No âmbito deste projeto, a biblioteca *LangChain* é utilizada para conectar o modelo *LLaMA*[14] com documentos técnicos provenientes das documentações oficiais dos provedores es-

[14] https://www.llama.com/

colhidos, permitindo a criação de respostas mais precisas e de acordo com a realidade operacional dos ambientes *multi-cloud*.

2.3.5 Modelo de Linguagem LLaMA Execução Local

O *LLaMA* é uma família de modelos de linguagem de código aberto desenvolvida pela *Meta*[15]. Sua principal proposta é oferecer modelos leves e eficientes, com arquitetura baseada em *Transformers*, voltados para tarefas de Processamento de Linguagem Natural (PLN) - campo da inteligência artificial que se dedica a permitir que computadores compreendam, interpretem e manipulem a linguagem humana, tanto falada quanto escrita - . A natureza aberta e acessível do *LLaMA* torna-o especialmente adequado para pesquisas acadêmicas e prototipagem em ambientes com restrições computacionais.

Do mesmo modo, também foram consideradas alternativas de modelos, como *MPT*[16] e *Mistral*[17], mas a escolha pelo *LLaMA* 3.2 baseou-se na boa documentação, estabilidade da versão e menor consumo de memória, aspectos fundamentais para viabilizar a execução em hardware limitado sem prejuízo à experiência dos usuários. Esses critérios foram decisivos para a seleção da ferramenta e do modelo no contexto acadêmico e experimental deste projeto. Além disso, o *LLaMA* possui integração nativa com bibliotecas como *LangChain* e o *Ollama*, facilitando a criação de aplicações inteligentes baseadas em RAG.

A escolha do *LLaMA* foi motivada por três fatores principais: (a) ser um modelo de código aberto com documentação acessível; (b) permitir execução local, o que evita custos com APIs comerciais; e (c) oferecer compatibilidade com ferramentas de orquestração como *LangChain*, essencial para o pipeline implementado. Esses atributos alinham-se diretamente com os objetivos educacionais e técnicos do projeto, que buscou desenvolver uma solução acessível, escalável e aplicável em ambientes acadêmicos.

2.3.6 Ferramenta de Execução Local: Ollama

O *Ollama*[18] é uma ferramenta que permite executar modelos de linguagem de forma local e simplificada, sem a necessidade de infraestrutura em nuvem. Ele foi projetado para facilitar a implementação e o uso de LLMs de código aberto, como o próprio *LLaMA*, através de

[15] https://www.meta.com/
[16] https://www.kaggle.com/models/mosaicml/mpt
[17] https://mistral.ai/
[18] https://ollama.com/

um ambiente leve, compatível com múltiplos sistemas operacionais e integrado a ferramentas como *LangChain*.

No projeto desenvolvido, o *Ollama* foi utilizado como camada de execução do modelo *LLaMA* 3.2 localmente. Sua adoção trouxe vantagens significativas: evitou o uso de APIs comerciais, reduziu custos de infraestrutura e garantiu maior controle sobre os dados utilizados no processo de geração. Além disso, sua compatibilidade com *Python* permitiu uma integração direta com o *backend* do terminal *web* desenvolvido com *Streamlit*.

A escolha pelo *Ollama* foi estratégica para atender aos requisitos de portabilidade, privacidade e independência tecnológica. Ao possibilitar que todo o ciclo de recuperação e geração ocorra em ambiente local, a ferramenta torna-se especialmente útil para contextos educacionais, onde a infraestrutura disponível pode ser limitada e o acesso à internet restrito. Dessa forma, o *Ollama* reforça a proposta do sistema como uma solução viável e replicável para ensino de computação em nuvem com auxílio de inteligência artificial. Embora outras ferramentas para execução local de LLMs estejam disponíveis — como *LM Studio*[19] e *Text Generation WebUI*[20] — optou-se pelo *Ollama* devido à sua configuração simplificada, integração nativa com a biblioteca *LangChain* e suporte direto a modelos como o *LLaMA*.

2.3.7 Banco de Dados Vetorial

Os bancos de dados vetoriais são estruturas especializadas para armazenar e recuperar vetores de alta dimensão, frequentemente utilizados em aplicações que envolvem busca semântica, recomendação e inteligência artificial. Em vez de realizar buscas por palavras-chave, esses bancos possibilitam comparações por similaridade entre representações vetoriais de textos, imagens ou outros dados.

No contexto de Geração Aumentada por Recuperação (RAG), os bancos vetoriais armazenam os *embeddings* — vetores numéricos que representam o significado semântico de trechos textuais — e permitem encontrar, de forma eficiente, os documentos mais similares a uma consulta representada também como vetor.

Neste trabalho, foi utilizado o *ChromaDB*, um banco vetorial de código aberto e com suporte a persistência local. Ele se mostrou ideal para o desenvolvimento do protótipo por oferecer baixa latência, fácil integração com o *framework LangChain* e permitir execução local

[19] https://lmstudio.ai/
[20] https://lablab.ai/tech/text-generation-webui

sem necessidade de servidores externos. Os documentos vetoriais indexados foram extraídos da documentação oficial da AWS e GCP, fragmentados em blocos curtos, enriquecidos com metadados e armazenados de forma a permitir consultas semânticas com alto grau de precisão.

2.3.8 Embeddings

Embeddings são representações vetoriais densas de palavras, frases ou documentos, que traduzem o significado semântico de elementos textuais para um espaço numérico de alta dimensionalidade. Ao mapear conceitos semanticamente próximos em vetores próximos, os *embeddings* viabilizam a busca semântica, a comparação de similaridade e a classificação textual com maior eficácia do que métodos tradicionais baseados em palavras-chave.

A utilização de *embeddings* é fundamental em sistemas do tipo RAG, pois permite converter tanto a consulta do usuário quanto os documentos de uma base de conhecimento em vetores comparáveis. Isso possibilita a recuperação dos trechos mais semanticamente relevantes mesmo quando há variações no vocabulário.

Neste trabalho, foi adotado o modelo *bge-m3*, que oferece um bom equilíbrio entre desempenho e custo computacional, sendo otimizado para uso em dispositivos locais. Este modelo segue os princípios descritos por Mikolov *et al.* (2013a), que introduziram o conceito de *word2vec* (técnica de aprendizado de máquina usada para aprender representações vetoriais de palavras em um corpus de texto), e evoluções posteriores baseadas em *Transformers* e arquiteturas *Bi-Encoder*, como o utilizado aqui.

A escolha do *bge-m3* se justifica por sua compatibilidade e fácil integração com o *ChromaDB* e *Framework LangChain*, leveza para execução local e por ser adequado a tarefas de recuperação semântica, especialmente em ambientes educacionais e de prototipagem rápida.

2.3.9 Métricas de Similaridade Vetorial

Sistemas de recuperação semântica utilizam representações vetoriais chamadas *embeddings* para medir a similaridade entre textos em um espaço multidimensional. Essa comparação é feita por meio de funções matemáticas, conhecidas como métricas de similaridade, que determinam o quão próximo semanticamente um documento está de uma consulta. As duas métricas mais comuns nesse contexto são a similaridade de cosseno e a distância euclidiana.

A similaridade de cosseno avalia o ângulo entre dois vetores, desconsiderando suas magnitudes. Isso a torna ideal para identificar similaridade semântica entre textos que

compartilham sentido, mesmo quando compostos por vocabulários distintos. Já a distância euclidiana considera a magnitude dos vetores, sendo mais sensível a variações de escala.

No projeto desenvolvido, a recuperação semântica foi realizada com base na similaridade de cosseno, adotada tanto pelo *ChromaDB* - banco de dados vetorial utilizado - quanto pela estratégia adicional de reranqueamento empregada. Para melhorar a qualidade dos documentos recuperados, optou-se pela inclusão de um mecanismo de reranqueamento baseado em MMR, disponibilizado pela biblioteca *LangChain*.

O MMR não é parte da arquitetura RAG padrão, mas foi incorporado neste trabalho como uma escolha técnica deliberada, com o objetivo de equilibrar *relevância* (proximidade da consulta) e *diversidade* (evitar redundância entre os documentos selecionados). Essa estratégia é parametrizada por um fator de ponderação λ (`lambda_mult`), que regula o peso relativo entre os dois critérios. Tal abordagem busca reduzir sobreposição semântica entre os documentos usados no *prompt* expandido, otimizando o uso do espaço contextual limitado do modelo de linguagem.

Essa decisão foi fundamentada em estudos como os de Pickett *et al.* (2024), que embora proponham alternativas mais recentes ao MMR, reconhecem sua eficiência, simplicidade e ampla adoção como técnica de reranqueamento em pipelines baseados em recuperação vetorial.

2.3.10 Métricas de Avaliação de Sistemas RAG e Estratégia Adotada

A avaliação de sistemas baseados em Geração Aumentada por Recuperação (RAG) pode ser realizada por diferentes abordagens. Em contextos de pesquisa, é comum utilizar *benchmarks* automatizados que medem aspectos como robustez, exatidão e integridade da resposta. Um exemplo é o *Retrieval Augmented Generation Benchmark* (RGB) – proposto por Chen *et al.* (2024), que estabelece critérios como:

- Robustez a Ruído: Capacidade do modelo de ignorar informações irrelevantes nos documentos recuperados.
- Rejeição Negativa: Habilidade de reconhecer quando não há evidência suficiente nos documentos para responder à consulta.
- Integração de Informações: Capacidade de combinar dados dispersos em diferentes fontes para formular uma resposta.
- Robustez Contrafactual: Capacidade de identificar e corrigir informações factualmente incorretas mesmo quando presentes nos documentos.

Embora esses critérios forneçam um excelente referencial teórico, a presente pesquisa

optou por um método prático e subjetivo de avaliação, mais alinhado ao objetivo educacional do protótipo. Os participantes avaliaram as respostas do sistema com base nos seguintes critérios, inspirados no *benchmark* acima:

- Acurácia Técnica: Avaliação da correção sintática e funcional dos comandos apresentados.
- Completude da Resposta: Capacidade da resposta de oferecer todo o contexto necessário para executar a ação corretamente.
- Clareza e Utilidade: Verificação de quão compreensível e instrutiva foi a explicação gerada.
- Relevância: Grau de aderência entre a pergunta feita e a resposta obtida.
- Avaliação Geral: Julgamento subjetivo da qualidade da resposta, considerando a experiência do usuário.
- Tipos de Erro Observados: Classificação de falhas comuns, como comandos inválidos, explicações confusas, alucinações ou erros de formatação.

Essas métricas foram aplicadas por meio de um formulário estruturado, após a realização de tarefas práticas com o terminal *web* inteligente. A análise conjunta dessas respostas permitiu identificar pontos fortes e limitações da arquitetura RAG adotada apresentados nos capítulos finais.

3 TRABALHOS RELACIONADOS

Este capítulo analisa estudos que abordam a aplicação de modelos de linguagem em computação em nuvem, com foco na otimização de ambientes *multi-cloud* e avaliação de desempenho de LLMs. Os trabalhos selecionados contribuíram conceitual e tecnicamente para a formulação da proposta aqui desenvolvida.

3.1 *Intelligent Network Optimization in Cloud Environments with Generative AI and LLMs*

O artigo PATIL e DESAI (2024) sugere uma transformação significativa na otimização de redes, movendo-se de configurações estáticas para redes dinâmicas em nuvem, orientadas por inteligência artificial generativa e modelos de linguagem LLM. A proposta investiga a aplicação de técnicas como *Generative Adversarial Networks* (GAN) - arquitetura de aprendizado profundo que envolve duas redes neurais competindo entre si: um gerador e um discriminador - e *Variational Autoencoders* (VAE) - rede neural que aprende uma representação compacta (espaço latente) dos dados de entrada e, em seguida, usa essa representação para gerar novos dados semelhantes aos originais. - para reproduzir padrões reais de tráfego de rede e utilização de recursos. Os autores validam a eficácia de sua proposta por meio de estudos de caso e simulações, destacando a habilidade dos LLMs em mitigar ataques de Negação de Serviço Distribuída (DDoS), otimizar cargas de trabalho específicas e aprimorar o desempenho global da rede.

Embora o artigo se concentre na infraestrutura de redes virtuais, há uma conexão com esta pesquisa no que se refere ao emprego de inteligência artificial para melhorar as operações de TI. Neste estudo, um *chatbot* baseado em RAG é utilizado para auxiliar na criação de máquinas virtuais em ambientes de múltiplas nuvens. Ambos os trabalhos partem da premissa de que a inteligência artificial pode facilitar tarefas técnicas e aumentar a eficiência operacional, apesar de suas aplicações práticas serem distintas.

Diferentemente do PATIL e DESAI (2024), que não especifica se as otimizações são realizadas automaticamente ou se necessitam de intervenção humana, a proposta apresentada aqui enfatiza a interação entre o operador e a ferramenta. A utilização de um *chatbot* possibilita a configuração de máquinas virtuais de maneira mais intuitiva, além de integrar recursos de diversos provedores, um aspecto que está ausente no estudo de apresentando, que foca em redes virtuais dentro de uma única nuvem.

3.2 *Retrieval Augmented Generation on Hybrid Cloud: A New Architecture for Knowledge Base Systems*

O estudo Chuang e Chen (2024) introduz uma arquitetura inovadora destinada a sistemas de conhecimento que adota a abordagem de Geração Aumentada por Recuperação (RAG), analisando a integração entre nuvens privadas e públicas para satisfazer, de maneira conjunta, as necessidades de privacidade e escalabilidade. A proposta envolve a combinação de grandes modelos de linguagem com informações obtidas de fontes externas, visando gerar respostas mais precisas e minimizar o risco de informações imprecisas. A solução fundamenta-se em uma estrutura híbrida que possibilita a realização de tarefas computacionais intensivas em nuvens públicas, enquanto informações sensíveis são preservadas em ambientes privados.

O principal objetivo do artigo é encontrar um equilíbrio entre o consumo de recursos computacionais requeridos pelos LLMs e a exigência de segurança nos dados. A designação dinâmica de funções entre nuvens privadas e públicas contribui para a eficiência e viabilidade econômica do sistema que foi sugerido. Ademais, são estabelecidas conexões seguras e técnicas para aprimorar o processo de busca e categorização de documentos, elevando a qualidade das respostas oriundas do sistema.

A relevância desta pesquisa é atribuída à investigação de soluções baseadas em nuvem que promovam eficiência e automação nos processos. Entretanto, enquanto Chuang e Chen (2024) se concentram em sistemas voltados para a recuperação de conhecimento, esta proposta tem como foco a criação de um *chatbot* que auxilia profissionais de TI na realização de tarefas práticas, como a configuração de máquinas virtuais.

A distinção mais significativa entre os dois estudos reside no objetivo final. O artigo enfatiza a criação de conhecimento estruturado, enquanto essa pesquisa se concentra na aplicação direta em atividades operacionais por meio de comandos automatizados. Apesar dessas diferenças, os aspectos técnicos discutidos por Chuang e Chen (2024) — incluindo o uso de RAG para tratar inconsistências e a adoção de estratégias de segurança em nuvens híbridas — podem ser utilizados para reforçar a solidez do terminal inteligente proposto.

3.3 *Benchmarking Large Language Models with Retrieval-Augmented Generation*

O estudo realizado por Chen *et al.* (2024) introduz um *benchmark* específico que avalia o desempenho de LLMs associados à estratégia RAG, denominado RGB. A investigação

foca na avaliação de quatro competências essenciais que são cruciais para a eficácia da metodologia RAG: robustez frente a ruído, rejeição negativa, integração de informações e robustez contrafactual.

A robustez frente a ruído avalia a capacidade do modelo de identificar a resposta correta mesmo quando os documentos recuperados contêm informações irrelevantes ou apenas tangenciais à pergunta. A rejeição negativa verifica se o modelo consegue se abster de responder quando os documentos não contêm dados adequados. A integração de informações mede a habilidade do modelo de reunir dados de múltiplas fontes para compor respostas mais complexas. Por fim, a robustez contrafactual testa a capacidade de detectar erros factuais em documentos externos, mesmo quando alertado previamente sobre possíveis inconsistências.

Esses critérios foram testados em quatro conjuntos de dados, o que possibilitou uma análise detalhada das limitações dos LLMs em condições ruidosas e inconsistentes. Os autores defendem que, embora o RAG apresente perspectivas promissoras, os modelos ainda enfrentam dificuldades, particularmente em virtude da significativa quantidade de ruído disponível na internet e da tendência dos LLMs de favorecer o conteúdo recuperado, mesmo que isso contrarie o conhecimento previamente adquirido.

Este estudo conecta-se à proposta desenvolvida aqui ao proporcionar uma base teórica para a avaliação de soluções fundamentadas no RAG. Enquanto o trabalho de Chuang e Chen (2024) realiza uma análise abrangente tanto teórica quanto experimental, a presente pesquisa utiliza os princípios do RAG em um contexto prático: o desenvolvimento de um terminal inteligente que assiste operadores de TI na execução de CLI em ambientes de múltiplas nuvens. A documentação oficial dos fornecedores, que é empregada como referência neste trabalho, tende a ter um baixo nível de ruído; contudo, erros ocasionais ou partes ambíguas ainda requerem análise.

Nesse cenário, os critérios apresentados por Chuang e Chen (2024) — especialmente a rejeição negativa e a robustez contrafactual — fundamentam diretamente a seleção das métricas de avaliação do terminal. A aplicação prática dessas métricas neste estudo busca assegurar que o modelo não apenas forneça respostas precisas, mas também identifique limitações e evite respostas errôneas, promovendo uma experiência confiável e segura para o usuário.

3.4 *Applications of Large Language Models in Cloud Computing: An Empirical Study Using Real-world Data*

O artigo de Li Sherry X Wang (2024) analisa a aplicação de LLMs na computação em nuvem, enfatizando a otimização de recursos e a administração eficaz de sistemas. Por meio da utilização de abordagens como inferência *bayesiana* - método de inferência estatística que utiliza o teorema de Bayes para atualizar a probabilidade de uma hipótese à medida que novas evidências são observadas - e *Markov Decision Processes* (MDP) - quadro matemático para modelar a tomada de decisões em situações em que os resultados são parcialmente aleatórios e parcialmente sob o controlo de um decisor - , os pesquisadores evidenciam melhorias consideráveis em aspectos como alocação de recursos, latência de rede, utilização de processamento e desempenho de armazenamento em plataformas como AWS, GCP e Azure.

O estudo se distingue por empregar LLMs como ferramentas autônomas aptas a avaliar grandes quantidades de dados e antecipar necessidades, viabilizando a automação de decisões complexas na gestão. Ao integrar esses modelos aos serviços em nuvem, há um aumento na confiabilidade operacional e uma diminuição de erros humanos. Os achados indicam que os LLMs têm o potencial de modificar a gestão dos recursos em nuvem, proporcionando soluções mais previsíveis, seguras e com desempenho superior.

A proposta desenvolvida aqui alinha-se ao objetivo do artigo de melhorar a experiência em núcleos de computação em nuvem utilizando inteligência artificial. Contudo, enquanto Li Sherry X Wang (2024) adotam uma perspectiva completamente automatizada e centrada na infraestrutura, esta pesquisa sugere um sistema interativo fundamentado em RAG, no qual o usuário se engaja ativamente através de um *chatbot* que oferece recomendações personalizadas.

A principal distinção entre as duas investigações reside na maneira de interação com o sistema. O terminal inteligente sugerido fornece suporte direto ao usuário, priorizando a usabilidade e a assistência operacional, enquanto o trabalho de Li Sherry X Wang (2024) favorece uma automação discreta, isenta de interação direta. Essa particularidade torna o sistema apresentado neste estudo especialmente apropriado para finalidades educacionais e operacionais, ao equilibrar a eficiência com a flexibilidade da intervenção humana.

3.5 *Practical Assessment of Large Language Models in Cloud Computing Using Real-World Data Applications*

O estudo de Singh *et al.* (2024) investiga a implementação de LLMs em ambientes de *multi-cloud*, com ênfase na otimização dos recursos disponíveis e na melhoria da eficiência operacional. Os autores aplicam abordagens tais como inferência *bayesiana* e MDP para antecipar padrões de utilização e ajustar a alocação de recursos de forma dinâmica em provedores de serviços como AWS, Azure e GCP. A proposta ressalta as vantagens da automação na redução de despesas operacionais e no aprimoramento do desempenho dos sistemas.

A similaridade com a presente pesquisa se encontra na utilização de tecnologias fundamentadas em LLMs para facilitar atividades em contextos de computação em nuvem. Ambos os estudos têm como objetivo aprimorar a eficiência e a confiabilidade das operações. Contudo, enquanto Singh *et al.* (2024) concentram seus modelos em decisões automatizadas relacionadas à infraestrutura, esta pesquisa apresenta um sistema interativo que foca na experiência do operador, integrando RAG e documentação oficial para fornecer respostas contextualizadas.

A inovação nesta proposta reside na atenção dada à interação humana. O terminal inteligente criado atua como um assistente técnico, orientando o operador na execução de tarefas, como a criação de máquinas virtuais, independentemente do provedor de nuvem escolhido. A interface com o *chatbot* permite que as respostas sejam ajustadas conforme as necessidades do usuário, oferecendo suporte prático baseado em fontes confiáveis.

Além disso, a implementação da técnica RAG aumenta a habilidade do sistema em produzir respostas pertinentes, mesmo quando confrontado com perguntas complexas ou incompletas. Isso se opõe à abordagem automatizada utilizada por Singh *et al.* (2024), que não considera interfaces acessíveis nem estratégias de recuperação de informação baseadas em conhecimento externo. Portanto, o trabalho proposto fornece uma solução alternativa que enfatiza a viabilidade e a eficácia dos aspectos educacionais e operacionais do sistema.

3.6 Análise comparativa

A avaliação dos estudos relacionados demonstrou o progresso recente na aplicação de LLMs e RAG em atividades de recuperação de informação e suporte técnico, principalmente em cenários de documentação restrita e avaliação automatizada. Entretanto, ao contrário dessas abordagens, este estudo sugere uma integração prática entre RAG e um terminal *multi-cloud*

interativo, enfatizando comandos concretos e validação empírica com usuários. Essa combinação de aplicação funcional, base curada e experimento com participantes humanos representa uma contribuição original e aplicada, especialmente em contextos educacionais e de treinamento técnico em computação em nuvem.

Ademais, o Quadro 1 apresenta-se uma comparação entre as obras pertinentes, considerando os aspectos principais de cada um dos trabalhos e da proposta desenvolvida neste estudo.

Quadro 1 – Comparação de trabalhos relacionados.

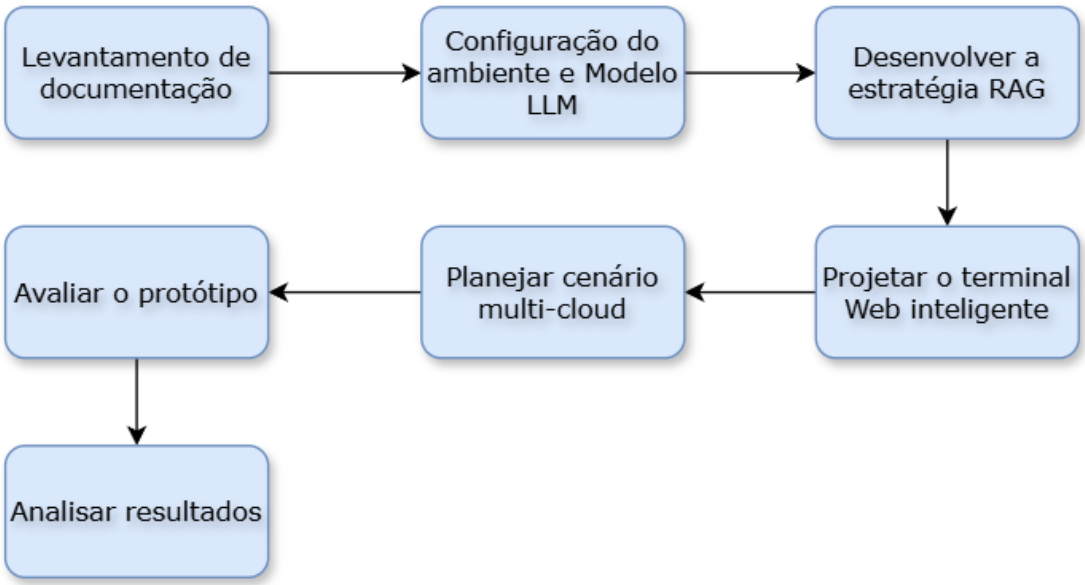
Trabalho	**Modelos Utilizados**	**Ferramentas e Frameworks**	**Métricas de Avaliação**	**Diferenças e Foco**
Retrieval Augmented Generation on Hybrid Cloud	GPT, LLaMA	RAG, Kubernetes, Vector DB, Hybrid Cloud	Privacidade, escalabilidade, segurança	Arquitetura híbrida para sistemas de conhecimento baseados em RAG
Intelligent Network Optimization in Cloud Environments with Generative AI and LLMs	GPT, GANs, VAEs	LLM-Twin Framework, Digital Twin, SDN, orquestração com SDN	Eficiência, latência, resiliência, capacidade de adaptação	Otimização dinâmica de redes em nuvem com LLMs e AI generativa
Practical Assessment of Large Language Models in Cloud Computing	GPT-3 (base)	AWS EC2, Kaggle datasets, MDPs, Bayesian Inference	CPU/memória/latência, job scheduling, redução de fila	Avaliação prática com dados reais, foco em gerenciamento de recursos
Benchmarking LLMs with RAG	ChatGPT, ChatGLM-6B, ChatGLM2-6B, Vicuna-7B, Qwen-7B, BELLE-7B	RGB Benchmark, Google Search API, dense retriever	Robustez do ruído, rejeição negativa, integração de informações, robustez contrafactual	Criação de benchmark para RAG, avaliação sistemática das capacidades fundamentais
Applications of LLMs in Cloud Computing: An Empirical Study Using Real-world Data	GPT-like (não especificado diretamente)	AWS, GCP, Vector DB, Apache Spark, Azure	Não especificadas diretamente; foco em análise empírica	Estudo empírico usando dados reais, aplicações práticas em ambientes de nuvem
Esta pesquisa	LLaMA 3.2 (via Ollama)	LangChain, ChromaDB, Streamlit, Ollama, Firecrawl, GitHub Actions	Ruído, completude, rejeição negativa, integração, robustez contrafactual	Terminal funcional com RAG, foco em usabilidade e integração real com provedores

Fonte: elaborado pelo autor.

4 METODOLOGIA

Este capítulo apresenta os procedimentos adotados para o desenvolvimento do protótipo proposto e para a condução do método avaliativo, estruturados nas sete etapas representadas no fluxograma da Figura 3: levantamento de documentação, configuração do ambiente e modelo LLM, desenvolvimento da estratégia RAG, projetar do terminal Web inteligente, planejamento do cenário *multi-cloud*, avaliação do protótipo e análise dos resultados obtidos.

Figura 3 – Fluxograma da pesquisa



Fonte: elaborada pelo autor.

4.1 Levantamento de Documentação

Inicialmente, foram examinados periódicos e artigos científicos pertinentes ao tema, com o objetivo de identificar as melhores práticas e soluções documentadas na literatura acadêmica. As informações coletadas foram organizadas e estruturadas em uma base de conhecimento que servirá como insumo para as etapas futuras. Além disso, foi realizado um levantamento detalhado das ferramentas de linha de comando dos provedores AWS e GCP, focando na funcionalidade EC2 e *Google Compute Engine* (GCE) para operações em ambientes *multi-cloud*. A investigação inicialmente se baseou na consulta das documentações oficiais desses serviços

AWS CLI e GCP CLI.

Embora a consulta direta à documentação oficial tenha sido considerada como fonte primária, observou-se uma elevada presença de conteúdos genéricos e não estruturados, o que dificultou o uso direto no sistema. Para contornar esse desafio, optou-se por construir um corpus próprio no formato *Markdown*, contendo comandos validados, organizados de forma sintética e previamente testados, com foco nas operações críticas de criação de máquinas virtuais e balanceamento de carga em ambientes AWS e *GCP*. Cada comando foi documentado de forma modular, garantindo consistência e relevância semântica no processo de recuperação de informações.

4.2 Ambiente e Modelo LLM

Nesta etapa, foi configurado um ambiente para a execução do modelo de *LLM*. A seleção de ferramentas adequadas, como o *Ollama* que consiste em uma ferramenta e plataforma que simplifica o uso de modelos de linguagem natural LLMs localmente, eliminando a necessidade de configurações complexas ou dependência de servidores externos. A proposta utilizou a base *LLaMA* para implementar um modelo localmente, com a estratégia RAG integrada. O modelo de linguagem utilizado foi o *LLaMA* 3.2 3B, com aproximadamente 3 bilhões de parâmetros.

A versão selecionada é compatível com o *Ollama*, o que permitiu sua execução local em hardware com capacidade limitada, sem dependência de APIs comerciais. A escolha se deu pelo equilíbrio entre desempenho e viabilidade de uso em ambientes locais. O ambiente contou com os recursos computacionais necessários para treinamento e inferência, incluindo bibliotecas para implementação do RAG. Além disso, foram realizados testes preliminares para validar a capacidade do modelo de processar entradas relacionadas à documentação dos provedores e gerar respostas coerentes.

Embora o plano inicialmente incluísse a execução do modelo e da interface diretamente nos dispositivos dos usuários, durante o desenvolvimento foram observadas limitações técnicas que tornavam essa abordagem inviável, como a incompatibilidade de ambientes, dificuldades na instalação de dependências considerando a diversidade dos equipamentos utilizados pelos alunos. Em razão disso, a decisão foi centralizar a execução do modelo em um servidor da Universidade Federal do Ceará (UFC), acessível tanto pela rede local quanto fora dela, com redirecionamento de porta.

Essa alternativa possibilitou a continuidade da proposta de utilização de um modelo leve *LLaMA* 3.2, agora funcionando em um ambiente controlado e reprodutível, sem exigir configurações específicas dos participantes. A alteração assegurou controle e estabilidade durante os testes e não prejudicou a essência do plano, uma vez que o sistema continuou disponível para acesso remoto nas mesmas condições de uso originalmente planejadas.

4.3 Estratégia RAG

Com o conhecimento fundamentado e o ambiente devidamente configurado, foi estabelecida uma estratégia RAG. Essa estratégia integrou a recuperação de informações pertinentes na documentação dos provedores com a elaboração de respostas contextualizadas para o usuário. A utilização da biblioteca *LangChain* simplificou essa integração, possibilitando a implementação de uma solução eficaz. *LangChain* consiste em fornecer aos desenvolvedores de IA ferramentas para conectar modelos de linguagem com fontes de dados externas, para este trabalho, o RAG alimentado com a documentação dos serviços. Esta fase também contemplou a avaliação e o aprimoramento da abordagem, embasada em testes controlados.

O pipeline técnico da aplicação foi estruturado com base em quatro componentes principais: ingestão da base textual, recuperação vetorial, ranqueamento semântico e geração de resposta com modelo LLM. Inicialmente, os arquivos em formato *Markdown* extraídos da documentação oficial da AWS foram processados por meio de fragmentação em blocos de texto, enriquecimento com metadados e indexação vetorial no banco de dados *ChromaDB*[1], utilizando *embeddings* do modelo *bge-m3*[2].

A etapa de recuperação emprega a técnica MMR para balancear similaridade e diversidade nos documentos retornados, com base em um codificador do tipo *Bi-Encoder*[3], que gera vetores representativos de cada sentença individualmente. Em seguida, os deis documentos mais relevantes são refinados por um ranqueador do tipo *Cross-Encoder*, que analisa diretamente o par formado pela pergunta do usuário e o conteúdo do documento, atribuindo uma pontuação semântica com maior precisão. Essa combinação permite um equilíbrio entre desempenho e exatidão na ordenação dos resultados. Por fim, os documentos ranqueados são incorporados ao *prompt* e enviados ao modelo *LLaMA* 3.2 (com 3 bilhões de parâmetros), executado localmente via *Ollama*, responsável por gerar a resposta contextualizada ao usuário. Esse fluxo foi encapsulado

[1] https://www.trychroma.com/
[2] https://ollama.com/library/bge-m3
[3] https://sbert.net/examples/cross_encoder/applications/README.html

em uma interface *web* construída com *Streamlit*, que permite interações em linguagem natural com o sistema. Detalhes da implementação completa dessa arquitetura encontra-se descrita no Apêndice B.

4.4 Terminal Web

O terminal inteligente foi apresentado como um protótipo de aplicação *web*, com acesso aos dados dos dois provedores, permitindo ao usuário consultar comandos e visualizar respostas aprimoradas pela estratégia RAG. A interface foi desenvolvida de maneira simples, alinhando-se às preferências por soluções simples como o *Streamlit*.

O *Streamlit* é um *framework* de código aberto, desenvolvido em *Python*, criando aplicações voltadas para ciência de dados e aprendizado de máquina. A ferramenta, possibilita desenvolver uma aplicação em um intervalo de tempo muito curto, desde que se possua conhecimento básico de *Python*. Nesta fase, a interface foi integrada ao *chatbot* e os componentes do RAG, assegurando que as operações realizadas sejam respaldadas por informações contextualizadas.

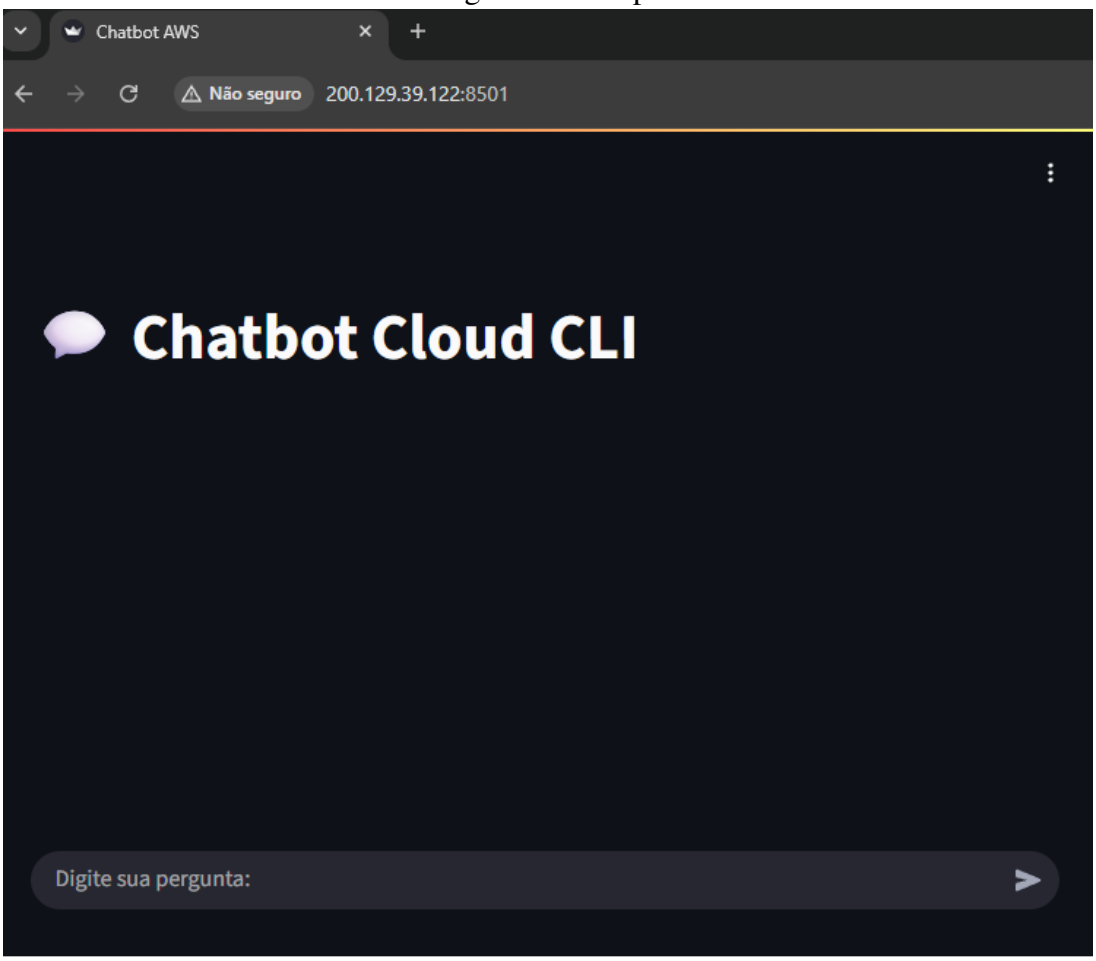
A criação de um terminal web usando o *Streamlit* tornou mais fácil integrar a arquitetura com a execução central do modelo no servidor da UFC. Devido à sua praticidade, foi simples colocar o modelo à disposição através do redirecionamento de porta, permitindo a realização de testes à distância, mantendo a ideia de um acesso leve e flexível. O terminal simula a experiência de um *chatbot* tradicional, recebendo perguntas e fornecendo respostas simples em texto, o que ajudou a preservar o aspecto educacional da atividade, que tem base em CLI.

Durante a execução do experimento, os alunos interagiram com uma interface web desenvolvida em *Streamlit*, projetada para receber perguntas em linguagem natural e retornar comandos da AWS CLI com base na recuperação de trechos relevantes (RAG). A Figura 4 apresenta a tela inicial do terminal, enquanto a Figura 5 mostra um exemplo de resposta gerada, incluindo o comando sugerido e o contexto utilizado na construção da resposta.

4.5 Cenário *Multi-Cloud*

Para ilustrar a funcionalidade do terminal, foi desenvolvido um cenário de implementação de uma aplicação em um ambiente *multi-cloud*. Este cenário abrangeu atividades como a criação de máquinas virtuais e balanceamento de carga, sendo possível na AWS e GCP. foram

Figura 4 – Tela inicial do terminal inteligente com suporte a RAG.



Fonte: Elaborado pelo autor.

planejadas algumas atividades para o operador executar junto ao terminal inteligente, para guiar os participantes e validar a proposta.

4.6 Avaliação do protótipo

A avaliação técnica da ferramenta foi realizada com base em um conjunto de tarefas previamente definidas, cujas soluções esperadas foram elaboradas com base na documentação oficial dos provedores em nuvem. Os participantes utilizaram o *chatbot* para realizar as atividades propostas e, em seguida, preencheram um formulário onde registraram a pergunta feita, a resposta recebida e classificaram a qualidade da resposta em termos de acurácia, completude, clareza, utilidade e relevância. Além disso, foram convidados a identificar erros como alucinações, comandos inválidos ou problemas de formatação, conforme critérios definidos no formulário.

Figura 5 – Exemplo de resposta gerada pelo modelo a partir de uma consulta com recuperação de contexto.

Fonte: Elaborado pelo autor.

4.7 Análise dos resultados

Os dados coletados durante a avaliação foram analisados de maneira quantitativa e qualitativa. Foram comparados os desempenhos dos participantes em relação ao tempo de execução, precisão das respostas dos modelos e facilidade de uso do terminal. Os resultados obtidos permitiram validar a eficácia da abordagem RAG e identificar oportunidades de melhorias futuras no protótipo e no *pipeline* de integração.

5 RESULTADOS

Nesta seção, detalhamos o procedimento para a realização dos experimentos e examinamos os resultados adquiridos.

5.1 Descrição do Experimento

A validação do protótipo foi realizada por meio de um experimento aplicado a 42 estudantes de cursos superiores da área de computação. Os participantes pertenciam as turmas: *Computação em Nuvem* e *Serviços de Redes*, cada uma com nível diferente de familiaridade com ferramentas de linha de comando.

A turma de *Computação em Nuvem*, composta por alunos que já haviam tido contato com infraestrutura na AWS via console gráfico, porém com pouca familiaridade com a AWS CLI, foi direcionada à realização da atividade usando o ambiente da própria AWS. Já os alunos da turma de *Serviços de Redes*, com experiência prévia em linha de comando e já habituados à automação de infraestrutura, foram designados para realizar a mesma tarefa na plataforma da GCP.

A proposta da atividade consistia na execução prática de um cenário de provisionamento de instância em nuvem, utilizando comandos fornecidos por um terminal web com *chatbot* baseado em RAG. Os comandos foram comparados a um gabarito técnico de referência, validando se os resultados estavam em conformidade com as boas práticas de uso da CLI das respectivas plataformas.

Entretanto, devido a dificuldades enfrentadas por parte dos alunos da turma de Serviços de Redes na ativação de contas no console da GCP, foi autorizada a utilização alternativa da AWS. Como resultado, 35 respostas foram registradas com base na AWS, 6 na GCP e 1 foi descartada por inconsistência, refletindo a distribuição original das turmas e as limitações enfrentadas no acesso à plataforma da GCP.

Todos os participantes utilizaram exclusivamente a versão com RAG do sistema, que realiza recuperação semântica de trechos de documentação técnica previamente curada e armazenada em banco vetorial. Isso permitiu ao modelo gerar respostas contextualizadas e alinhadas com as práticas e sintaxes reais das plataformas de nuvem.

A atividade foi proposta aos alunos com prazo de até três dias para conclusão, permitindo que fosse realizada tanto nas dependências da instituição quanto remotamente,

conforme a preferência de cada participante. Após a finalização da tarefa, os alunos preencheram um formulário dividido em três seções:

1. Registro da interação com o modelo, contendo a pergunta enviada e a resposta recebida;
2. Avaliação da resposta com base em critérios técnicos e pedagógicos: acurácia, completude, clareza, utilidade e relevância;
3. Classificação de eventuais erros, como comandos inválidos, omissões, ou alucinações do modelo.

O Quadro 2 apresenta as perguntas do formulário preenchido pelos participantes, que serviram como base para a coleta dos dados analisados.

Quadro 2 – Perguntas do formulário preenchido pelos participantes.

N°	**Pergunta**
1	Cole aqui qualquer pergunta feita ao modelo.
2	Adicione aqui a resposta gerada pelo modelo.
3	Acurácia Técnica: O comando ou informação técnica principal na resposta estava correto e funcional?
4	Completude da Resposta: A resposta forneceu todo o contexto necessário para um aluno entender e executar a ação?
5	Clareza e Utilidade: A explicação que acompanha o comando foi fácil de entender e útil?
7	Relevância da Resposta: O modelo respondeu diretamente à pergunta que foi feita?
8	Avaliação Geral das respostas.
9	Se a resposta não foi boa, selecione o(s) tipo(s) de erro observado(s).
10	Forneça algum *feedback* adicional. Se desejar, escreva aqui qual seria a "resposta ideal" que você esperava do modelo.

Fonte: Elaborado pelo autor.

Esse conjunto de dados possibilitou uma análise quantitativa e qualitativa da experiência dos usuários com o sistema, permitindo verificar sua efetividade prática em tarefas de provisionamento em nuvem e levantar oportunidades de melhoria da solução.

5.2 Avaliação dos Resultados

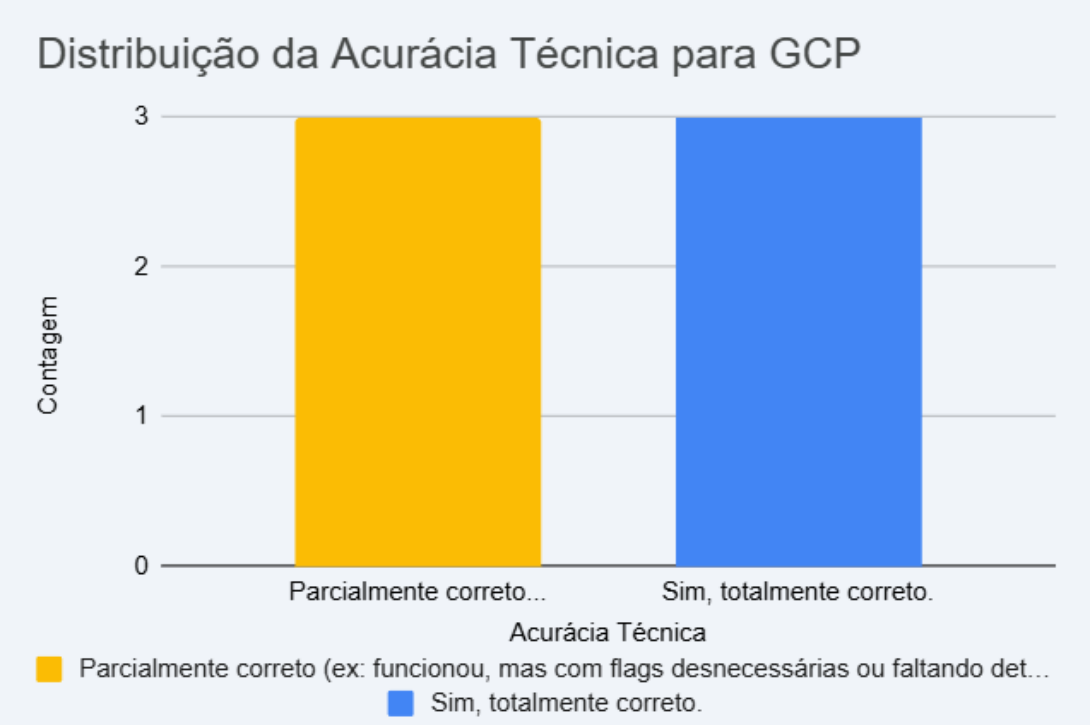
A análise numérica dos dados obtidos teve como objetivo entender a eficácia do terminal inteligente segundo várias normas de qualidade das respostas. A investigação foi feita com base nas 41 respostas válidas que apontaram o provedor utilizado 35 para AWS e 6 para GCP . Os critérios analisados foram: precisão técnica, completude, clareza e utilidade, relevância e avaliação global.

Os aspectos analisados incluíram: acurácia técnica, completude, clareza e utilidade, pertinência e avaliação global. As conclusões foram extraídas de questionários respondidos após a realização de uma atividade prática com o suporte do *chatbot* incorporado ao terminal web.

5.2.1 Acurácia Técnica

A maior parte das respostas sobre AWS foi considerada tecnicamente correta ou parcialmente correta. No total, (45,7%) das respostas sobre AWS foram avaliadas como totalmente corretas, enquanto (28,5%) foram parcialmente corretas e (22,8%) estavam incorretas. Em relação ao GCP, obteve-se bons resultados para as poucas que tiveram sucesso, com (50%) das respostas totalmente corretas e (50%) parcialmente corretas, sem respostas incorretas.

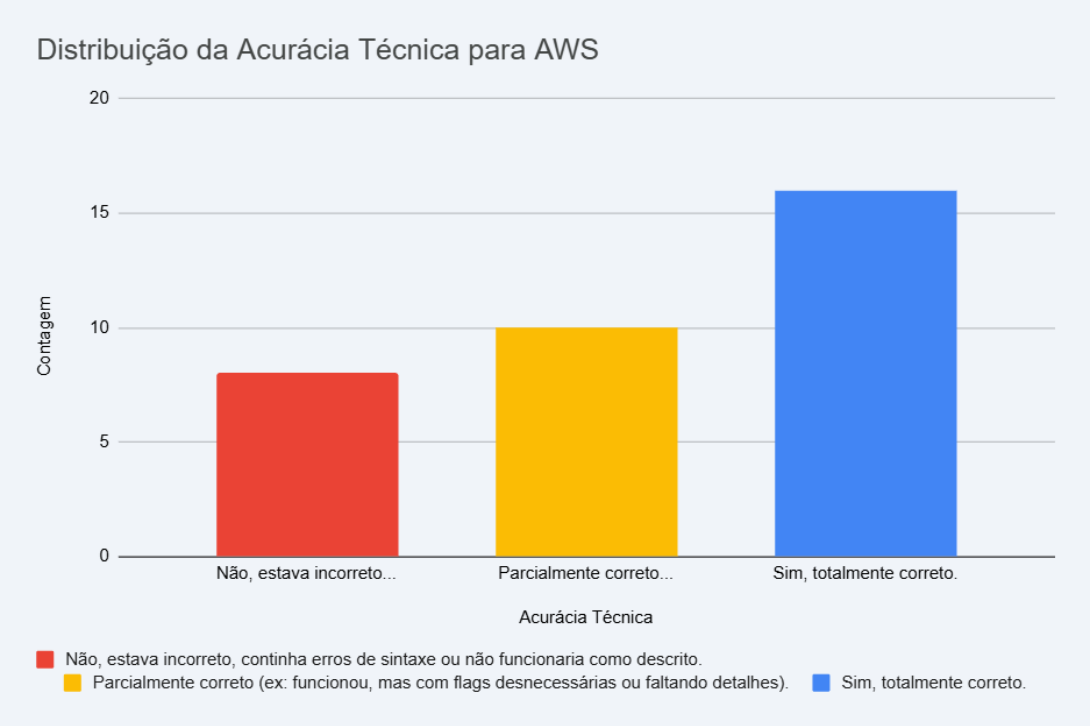
Figura 6 – **Questão 03 - GCP** O comando ou informação técnica principal na resposta estava correto e funcional?



Fonte: Elaborado pelo autor.

Conforme demonstram os gráficos das Figuras 6 e 7, observa-se que a maior parte das respostas relacionadas a AWS foi categorizada como “Sim” ou “Parcialmente correto”, o que sugere uma significativa coerência técnica. No que se refere à GCP, mesmo com a quantidade limitada de respostas, houve uma predominância de avaliações favoráveis, com apenas uma resposta classificada como tecnicamente errônea.

Figura 7 – **Questão 03 - AWS** O comando ou informação técnica principal na resposta estava correto e funcional?



Fonte: Elaborado pelo autor.

5.2.2 Completude da Resposta

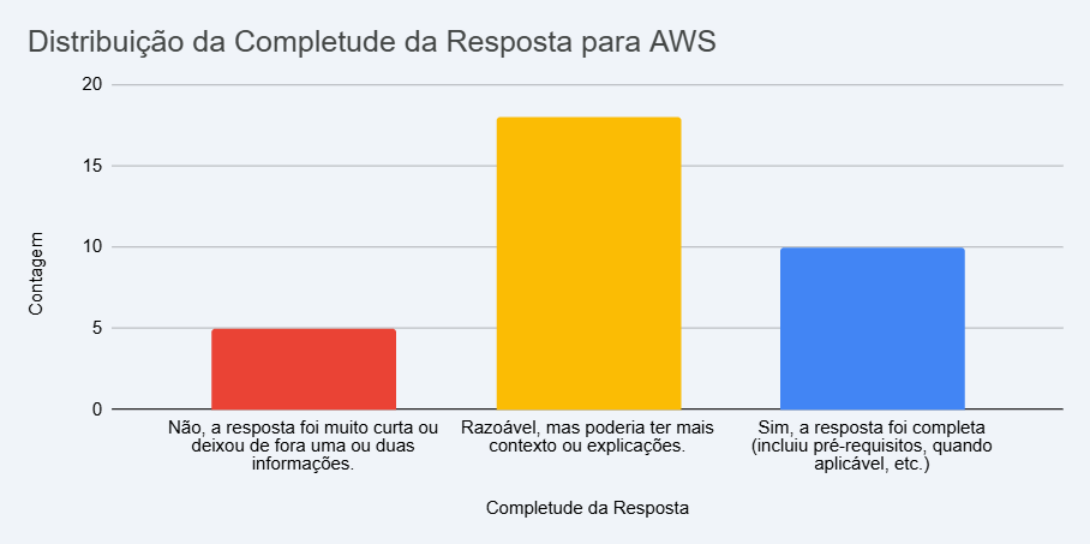
No critério de completude, a maior parte das respostas da AWS foi avaliada como adequada para que o usuário pudesse entender e realizar a ação sugerida, com (51,4%) classificadas como “Razoável, mas poderia ter mais contexto” e (28,5%) como “Sim, a resposta foi completa”. As respostas da GCP mostraram mais variação, refletindo tanto a base de participantes menor quanto a menor familiaridade com a plataforma, com (66,6%) “Razoáveis” e (16,6%) “Completas”. Além disso, alguns usuários apontaram que as respostas poderiam conter mais detalhes para oferecer um contexto adicional, o que seria especialmente benéfico para aqueles com menor experiência.

Os gráficos presente nas Figuras 8 e 9 enfatizam essa visão. A AWS demonstrou uma significativa predominância de respostas categorizadas como completas, enquanto a GCP, apesar de receber uma avaliação satisfatória de forma geral, apresentou maior variação nas categorias “Parcialmente” e “Sim”.

Figura 8 – **Questão 04 - GCP** A resposta forneceu todo o contexto necessário para um aluno entender e executar a ação?

Fonte: Elaborado pelo autor.

Figura 9 – **Questão 04 - AWS** A resposta forneceu todo o contexto necessário para um aluno entender e executar a ação?



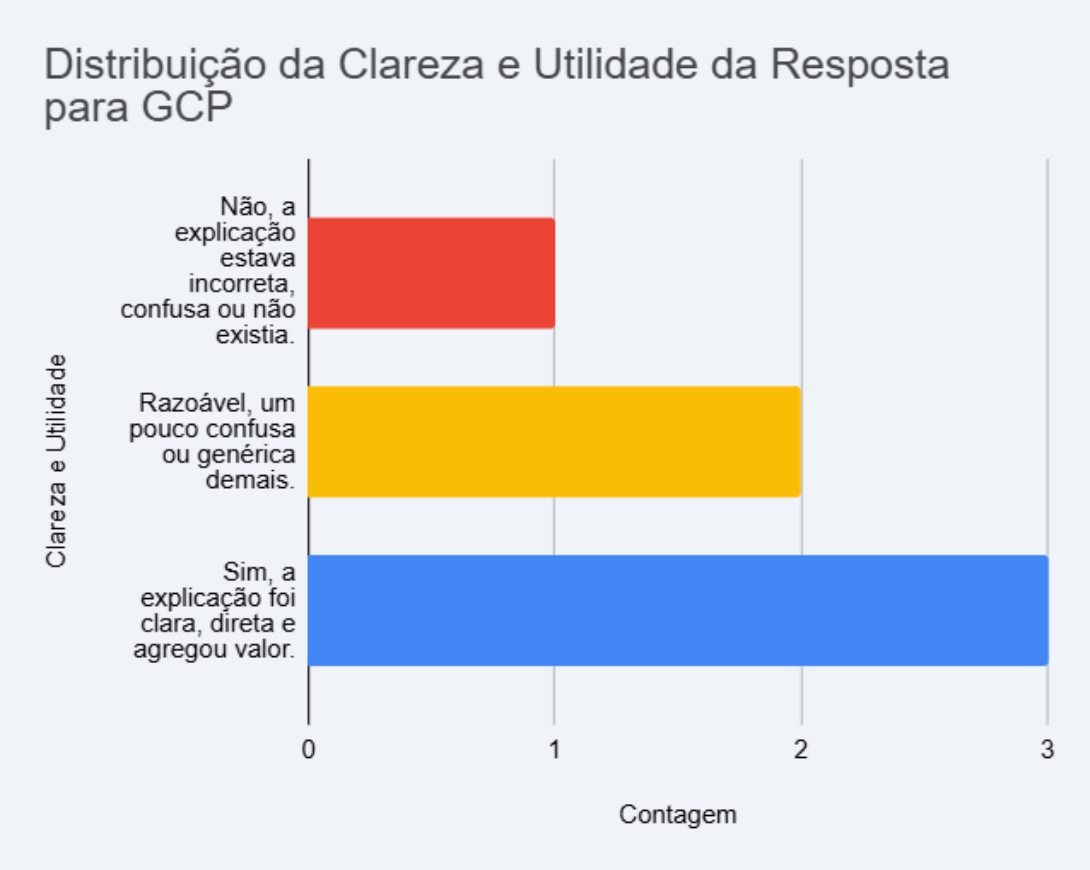
Fonte: Elaborado pelo autor.

5.2.3 Clareza e Utilidade

A transparência foi um dos fatores mais elogiados. Os participantes mencionaram que o *chatbot*, com a ajuda da estratégia RAG, ofereceu explicações diretas e de fácil compreensão sobre os comandos apresentados. Apesar disso, houve situações em que o modelo gerou instruções com *flags* ou parâmetros inadequados e não conseguiu ajustá-los quando solicitado, justificando que não encontrava exemplos na documentação. No total, (48,5%) das respostas

da AWS foram consideradas “Claras, diretas e úteis”. As respostas da GCP, embora menos frequentes, também foram vistas como satisfatórias nesta característica, com (50%) das respostas “Claras, diretas e úteis”, (33,3%) “Razoáveis” e (16,6%) “Não, a explicação estava incorreta ou confusa”.

Figura 10 – **Questão 05 - GCP** A explicação que acompanha o comando foi fácil de entender e útil?



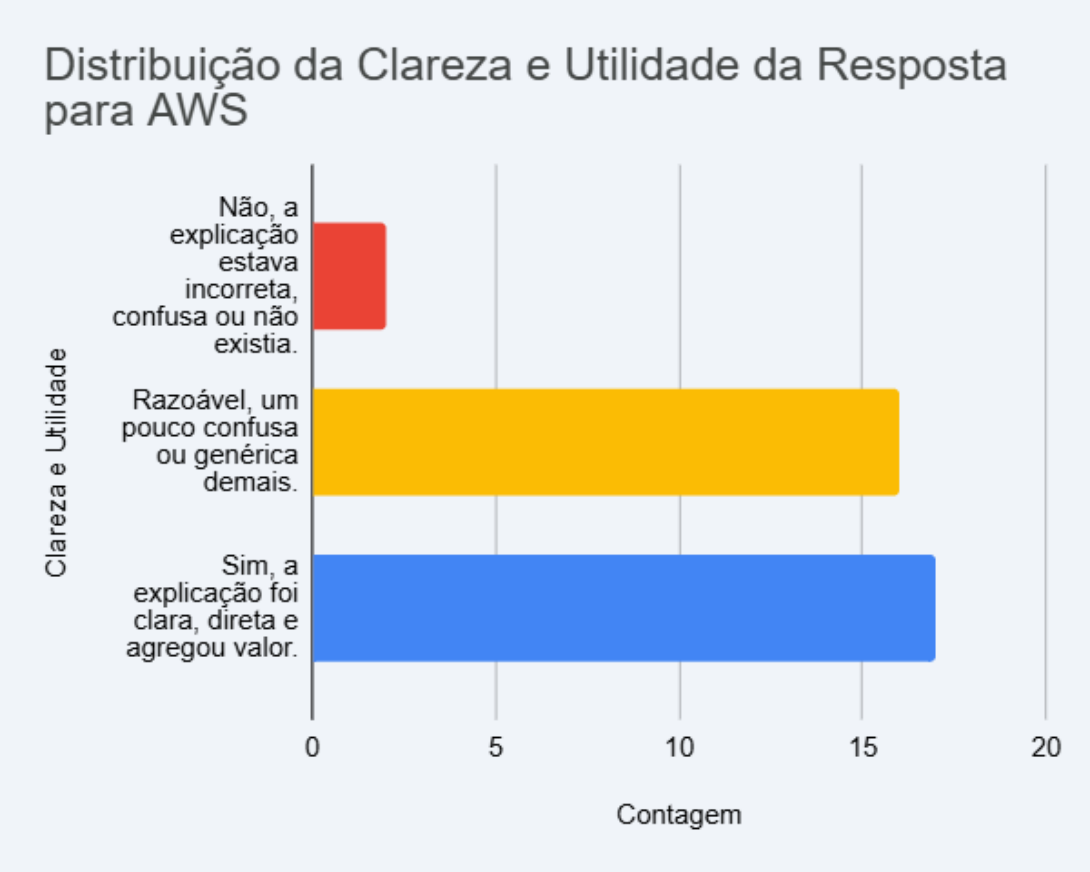
Fonte: Elaborado pelo autor.

Os dados apresentados no gráficos das Figuras 10 e 11 ilustram a percepção dos participantes em relação à clareza das respostas. A maioria das respostas do AWS foi classificada como clara e útil (“Sim”), enquanto na GCP houve uma predominância de avaliações “Parcialmente”, embora sem grandes perdas qualitativas. Isso indica que a forma como as instruções foram apresentadas pelo *chatbot* foi, em geral, compreensível, mesmo entre distintos provedores.

5.2.4 Relevância

A maiorias das respostas foram avaliadas como diretamente pertinente à pergunta feita, evidenciando que o sistema de recuperação semântica foi eficiente em interpretar o contexto das solicitações, com (71,4%) das respostas da AWS e (83,3%) das respostas da GCP sendo

Figura 11 – **Questão 05 - AWS** A explicação que acompanha o comando foi fácil de entender e útil?



Fonte: Elaborado pelo autor.

totalmente relevantes. Contudo, alguns participantes indicaram que respostas mais explicativas, além de diretas, seriam benéficas para o aprendizado e a compreensão dos comandos.

Nas Figuras 12 e 13, é possível notar que, para tanto a AWS quanto a GCP, a maior parte das respostas foi classificada como pertinente “Sim”. A variação entre os provedores foi mínima, destacando a eficácia da fase de recuperação semântica do modelo com RAG.

5.2.5 Avaliação Geral

A maioria das avaliações para a AWS variou entre “Bom”, “Muito bom” e “Excelente”, especificamente, (74,2%) das respostas foram avaliadas como “Bom”, “Excelente” ou “Muito bom”, enquanto (11,4%) foram consideradas “Ruim” ou “Muito ruim”. No que diz respeito à GCP, todas as respostas foram classificadas entre “Bom” e “Muito bom”, com (83,3%) das respostas recebendo essas avaliações positivas e (16,6%) classificadas como “Ruim”.

A Figura 14 resume a avaliação total do terminal realizada por todos os envolvidos. A maioria das classificações situou-se entre “Bom” e “Excelente”, apresentando um índice de

Figura 12 – **Questão 06 - GCP** O modelo respondeu diretamente à pergunta que foi feita?

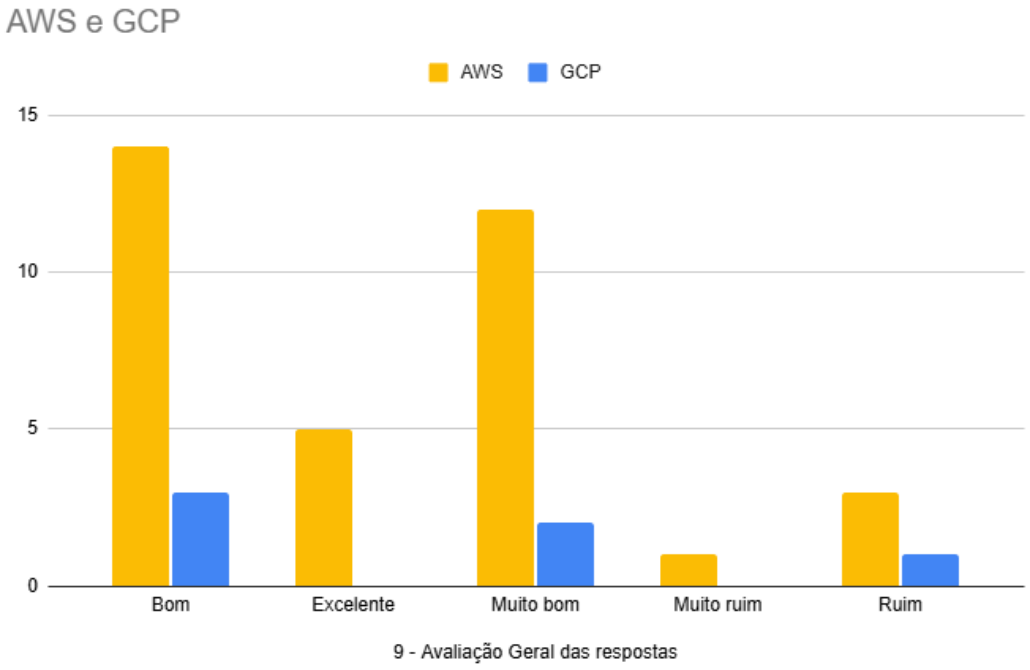
Fonte: Elaborado pelo autor.

Figura 13 – **Questão 06 - AWS** O modelo respondeu diretamente à pergunta que foi feita?

Fonte: Elaborado pelo autor.

rejeição baixo tanto para AWS quanto para GCP. Esse resultado destaca a adoção da ferramenta como uma interface eficaz para consultas técnicas em nuvem por meio da CLI.

Figura 14 – **Questão 08** Avaliação Geral das respostas AWS e GCP



Fonte: Elaborado pelo autor.

5.2.6 Discussão Final

A análise abrangente dos dados coletados enfatiza a eficácia da proposta elaborada. Os dados mostram que, para a AWS, (74,2%) das respostas foram avaliadas positivamente, enquanto para a GCP, (83,3%) receberam avaliações positivas, indicando uma recepção favorável da ferramenta. A análise das respostas obteve avaliações favoráveis com base em critérios como precisão técnica, completude e clareza, o que sugere que o modelo conseguiu oferecer comandos precisos e explicações claras.

Para complementar a análise quantitativa, o Quadro 3 apresenta os principais comentários dos participantes, destacando pontos positivos e áreas de melhoria do terminal inteligente.

De acordo com os resultados obtidos, pode-se afirmar que a proposta atinge seu principal objetivo de fornecer uma solução de assistência inteligente, sólida e viável em ambientes *multi-cloud*. Embora o número de respostas para a GCP seja limitado, as avaliações recebidas foram positivas, sugerindo que a base de conhecimento empregada pelo sistema possui qualidade técnica adequada para oferecer suporte aos provedores. Um número reduzido de respostas sinalizou questões de clareza ou imprecisão, o que enfatiza a necessidade de ampliar o corpus da documentação e seguir aprimorando os *prompts* e filtros semânticos.

Do ponto de vista da usabilidade, os estudantes mencionaram que o sistema facili-

Quadro 3 – Principais comentários dos participantes sobre o desempenho do terminal inteligente.

Critério	**Comentário**	**Implicação**
Acurácia Técnica	"Mesmo usando a interface *web*, ao criar a descrição de um *security group*, ele não aceita por 'ç', mas o modelo retornou a descrição com esse caractere."	Indica a necessidade de ajustar a codificação de caracteres para maior compatibilidade com a interface.
Acurácia Técnica	"Muitas vezes o bot indicou *flags* ou filtros incorretos, e não conseguiu corrigir quando apontado, alegando não encontrar exemplos na documentação."	Sugere limitações na base de conhecimento e na capacidade de adaptação do modelo a erros.
Completude	"Gostei bastante, mas poderia se aprofundar mais nos detalhes dos comandos ao retornar."	Reforça a necessidade de respostas mais detalhadas para melhorar o entendimento dos usuários.
Clareza e Utilidade	"No geral, o modelo trouxe respostas relevantes, mas às vezes indicava *flags* incorretos, e não conseguia corrigir quando apontado."	Evidencia a utilidade geral, mas destaca falhas na correção de erros, impactando a confiabilidade.
Relevância	"Deveria ter uma explicação a mais e não só uma resposta direta, é bom para fornecer mais ajuda ao aluno e curiosidade."	Sugere que respostas mais explicativas podem aumentar o valor educacional do modelo.

Fonte: Elaborado pelo autor com base nos dados da pesquisa.

tou uma experiência mais eficiente e educacional ao realizar tarefas que, convencionalmente, demandariam um esforço técnico mais significativo e um entendimento prévio dos comandos. A estrutura do terminal simulada com a interface *web*, junto à combinação do modelo com as documentações autênticas, possibilitou que usuários com nível variado de conhecimento em CLI executassem a atividade de forma independente.

Os dados também sugerem que a estratégia RAG, ao ser aplicada em um domínio técnico e prático como o provisionamento em nuvem, possui potencial significativo para transformar a maneira como operadores e aprendizes interagem com sistemas de apoio à decisão. O uso de documentação oficial como fonte primária de contexto reduziu as chances de alucinação e aumentou a confiabilidade das respostas.

Apesar da maioria das respostas terem sido classificadas como claras e úteis, alguns participantes relataram que, em determinadas situações, as respostas apresentaram-se excessivamente diretas ou pouco explicativas. Essa limitação pode estar relacionada à janela de contexto reduzida do modelo e à própria natureza dos *prompts* gerados, indicando a necessidade de estudos mais aprofundados sobre estratégias que equilibrem concisão com detalhamento instrucional,

sobretudo em contextos educacionais.

Outro aspecto observado refere-se a falhas pontuais de codificação de caracteres e à presença de campos genéricos nos comandos, representados por marcadores como (`<sua-key>` ou `<id-do-sg>`). Essa escolha foi proposital, considerando que o modelo não possui acesso direto às credenciais ou recursos da conta do usuário, e visa preservar a segurança e a personalização das instruções. No entanto, também revela uma oportunidade de evolução futura da ferramenta, por meio da integração segura com metadados da conta, respeitando princípios de privacidade e autenticação federada.

Ademais, os retornos recebidos através dos formulários indicam oportunidades promissoras para melhorias futuras, como a expansão da base de conhecimento, a adição de suporte a novos serviços para além de EC2 e GCE, e o avanço da experiência do usuário na interface *web*.

Por fim, destaca-se como limitação metodológica a ausência de um grupo controle com uma versão do sistema sem recuperação via RAG, o que inviabiliza uma comparação direta entre a geração com e sem suporte de documentos externos. Essa ausência foi uma escolha intencional, visando concentrar a avaliação no protótipo completo. No entanto, para investigações futuras, recomenda-se a condução de testes comparativos com e sem RAG, de modo a quantificar com maior precisão o impacto da recuperação semântica na qualidade das respostas geradas.

6 CONSIDERAÇÕES FINAIS

Este capítulo apresenta as considerações finais do trabalho, sintetizando os principais resultados alcançados, discutindo as limitações observadas durante o desenvolvimento e propondo direções para estudos futuros.

6.1 Conclusão

Este trabalho teve como objetivo o desenvolvimento de um terminal inteligente baseado na arquitetura de Geração Aumentada por Recuperação (RAG), voltado para auxiliar usuários na execução de comandos CLI em ambientes *multi-cloud*. A proposta incluiu a utilização de documentações oficiais da AWS e GCP, com execução local do modelo *LLaMA* 3.2 via *Ollama*, integrado a *embeddings* vetoriais e reranqueamento semântico leve. A interface foi desenvolvida com *Streamlit*, permitindo acesso *web* independente de APIs comerciais.

Entre os objetivos específicos estabelecidos estavam: (i) identificar provedores relevantes e suas documentações; (ii) configurar o ambiente computacional para execução do LLM integrado com o RAG; (iii) implementar o terminal com suporte a múltiplos provedores; (iv) disponibilizar a aplicação para testes práticos; e (v) avaliar a eficácia do sistema com base em métricas como acurácia, completude, clareza, utilidade e relevância.

Com base nos resultados obtidos, pode-se afirmar que os objetivos (i), (ii), (iii) e (iv) foram integralmente atingidos. A aplicação foi implementada e disponibilizada a duas turmas distintas, em um experimento controlado com 42 participantes. Já o objetivo (v), relacionado à eficácia das respostas, foi parcialmente atingido. As métricas coletadas indicaram desempenho satisfatório em critérios como relevância (71,4% para AWS e 83,3% para GCP) e avaliação geral positiva (74,2% para AWS e 83,3% para GCP), mas revelaram limitações em acurácia técnica (45,7% para AWS, 50% para GCP) e completude (51,4% para AWS, 66,6% para GCP). Esses dados indicam que, embora a ferramenta tenha cumprido seu papel como assistente técnico, há espaço para melhorias, especialmente em tarefas mais complexas.

Limitações técnicas também foram identificadas no uso do modelo leve *LLaMA* 3.2, que apresentou dificuldades para corrigir respostas inválidas, lidar com erros de codificação e adaptar-se a múltiplas intenções no *prompt*. Além disso, os campos genéricos inseridos por segurança (`<sua-key>`, `<id-do-sg>`) geraram dúvidas em alguns participantes. Essas questões, somadas à ausência de um grupo de controle (sem RAG), limitam a generalização dos resultados.

Do ponto de vista prático, a experiência foi enriquecedora: comprovou-se a viabilidade de uma aplicação RAG local, com boa aceitação por parte dos estudantes, especialmente quanto à clareza, navegabilidade e autonomia na realização de tarefas. A proposta demonstrou potencial para uso educacional em ambientes com recursos limitados, destacando o valor de soluções independentes de nuvem para formação técnica.

Em síntese, o projeto atingiu seu propósito principal de investigar a viabilidade de um assistente técnico baseado em RAG, executado localmente e alimentado por documentação oficial. Apesar das limitações observadas, os resultados obtidos e as análises realizadas oferecem subsídios relevantes para pesquisas futuras sobre RAG, especialmente em contextos educacionais, com foco em confiabilidade, interpretabilidade e usabilidade.

6.2 Trabalhos Futuros

Com base nos experimentos realizados, nas análises conduzidas e nas limitações identificadas, propõem-se a seguir direções para trabalhos futuros que visem aprofundar ou ampliar esta proposta:

- Utilização de modelos mais robustos: explorar o uso de modelos de linguagem de maior capacidade, seja por meio de execução local em máquinas com maior poder computacional, seja por meio de APIs hospedadas na nuvem, com o objetivo de aprimorar a qualidade, o detalhamento e a adaptabilidade das respostas.
- Aprimoramento do pipeline RAG: incorporar estratégias de *fallback* baseadas em nível de confiança da resposta, balanceamento automático entre diversidade e precisão, além de mecanismos que ofereçam respostas alternativas ou direcionem o usuário para trechos relevantes da documentação oficial.
- *Feedback* supervisionado e adaptação dinâmica: implementar mecanismos que permitam aos usuários avaliarem as respostas recebidas, utilizando esse retorno para adaptar dinamicamente o reranqueamento e a curadoria do banco vetorial.
- Avaliação comparativa com e sem RAG: realizar experimentos com grupos controle, utilizando versões com e sem a arquitetura RAG, a fim de quantificar o impacto efetivo da recuperação aumentada na qualidade da interação.
- Ampliação da cobertura técnica: expandir o banco de documentos para incluir novos serviços além de EC2 e GCE, como balanceadores de carga, bancos de dados gerenciados e serviços de armazenamento, aumentando o escopo de atuação da ferramenta.

- Aplicações em novos contextos educacionais: testar a ferramenta em diferentes níveis de ensino (técnico, graduação, extensão), avaliando sua eficácia conforme o perfil dos usuários, a familiaridade com CLI e os objetivos pedagógicos envolvidos.
- Melhorias na interface e na experiência do usuário: incorporar funcionalidades como preenchimento automático de parâmetros, mensagens de erro personalizadas e integração com sistemas de autenticação segura, promovendo maior personalização e usabilidade da aplicação.

A implementação dessas melhorias poderá contribuir para superar as limitações identificadas nesta pesquisa, tornando a solução mais robusta, precisa e adaptável a diferentes perfis de usuários e contextos de aplicação. Além disso, reforça o potencial da aplicação de RAG local como ferramenta de apoio à aprendizagem e à operação técnica em ambientes de computação em nuvem.

REFERÊNCIAS

Amazon Web Services. **Amazon EC2 Documentation**. [*S. l.*], 2025. Disponível em: https://docs.aws.amazon.com/pt_br/AWSEC2/latest/UserGuide/concepts.html. Acesso em: 13 jan. 2025.

CARISSIMI, A. **Desmistificando a computação em nuvem**. Instituto de Informática, Universidade Federal do Rio Grande do Sul (UFRGS), Porto Alegre–RS, 2015.

CHEN, J.; LIN, H.; HAN, X.; SUN, L. Benchmarking large language models in retrieval-augmented generation. In: AAAI CONFERENCE ON ARTIFICIAL INTELLIGENCE. **Proceedings [...]**. [*S. l.*], 2024. v. 38, n. 16, p. 17754–17762.

CHUANG, C.-C.; CHEN, K.-C. Retrieval augmented generation on hybrid cloud: A new architecture for knowledge base systems. In: IIAI INTERNATIONAL CONGRESS ON ADVANCED APPLIED INFORMATICS (IIAI-AAI). **Proceedings [...]**. [*S. l.*]: IEEE, 2024. p. 68–71.

INTELLIGENCE, M. **Tamanho do mercado Cloud Computing e Análise de participação - Tendências de crescimento e previsões**. 2024. Disponível em: https://www.mordorintelligence.com/pt/industry-reports/cloud-computing-market. Acesso em: 12 dez. 2024.

KARTHIK, T.; REDDY, D. N. C. S.; POLOJU, R. S. Integration of multicloud computing environment and security. In: **International Journal of Scientific Engineering and Technology Research, Volume.03, IssueNo.06**. [*S. l.*: *s. n.*], 2014. p. 0941–0945.

LI SHERRY X WANG, F. S. K. N. R. S. H. Applications-of-large-language-models-in-cloud-computing:-an-empirical-study-using-real-world-data. **International Journal of Innovative Research in Computer Science and Technology (IJIRCST)**, v. 12, n. 4, p. 59–69, 2024. Disponível em: https://www.ijircst.org/view_abstract.php?year=&vol=12&primary=QVJULTEyOTE=. Acesso em: 23 out. 2024.

MIKOLOV, T.; CHEN, K.; CORRADO, G.; DEAN, J. Efficient estimation of word representations in vector space. **arXiv preprint arXiv:1301.3781**, 2013.

MIKOLOV, T.; SUTSKEVER, I.; CHEN, K.; CORRADO, G. S.; DEAN, J. Distributed representations of words and phrases and their compositionality. **Advances in neural information processing systems**, v. 26, 2013.

MORAES, P. A. A. R. **Uma análise de Desempenho Multi-Cloud de uma Aplicação Baseada em Microsserviços**. 57 f. Monografia (Bacharelado) – Universidade Federal do Piauí Campus Senador Helvídio Nunes de Barros, Sistemas de Informação, Piauí, 2021.

PATIL, K.; DESAI, B. **Intelligent Network Optimization in Cloud Environments with Generative AI and LLMs**. [*S. l.*], 2024. Disponível em: https://doi.org/10.20944/preprints202406.0578.v1. Acesso em: 4 nov. 2024.

PICKETT, M.; HARTMAN, J.; BHOWMICK, A. K.; ALAM, R.-u.; VEMPATY, A. Better rag using relevant information gain. **arXiv preprint arXiv:2407.12101**, 2024.

ROFFO, G. Exploring advanced large language models with llmsuite. **arXiv preprint arXiv:2407.12036**, 2024. Acesso em: 24 ago. 2024.

SINGH, V. K.; MOHARANA, B.; SARKAR, T.; MAAN, A.; BHATTACHERJEE, A.; RAKHRA, M. Practical assessment of large language models in cloud computing using real-world data applications. In: INTERNATIONAL CONFERENCE ON CYBERNATION AND COMPUTATION (CYBERCOM). **Proceedings [...]**. [*S. l.*], 2024. p. 356–362.

SOUSA, F. R.; MOREIRA, L. O.; MACHADO, J. C. **Computação em nuvem**: conceitos, tecnologias, aplicações e desafios. [*S. l.*], 2009. Disponível em: https://www.researchgate.net/profile/Javam-Machado/publication/237644729. Acesso em: 23 out. 2024.

WEI, J.; WANG, X.; SCHUURMANS, D.; BOSMA, M.; XIA, F.; CHI, E.; LE, Q. V.; ZHOU, D. *et al.* Chain-of-thought prompting elicits reasoning in large language models. **Advances in neural information processing systems**, v. 35, p. 24824–24837, 2022.

ZHAO; PENGHAO; ZHANG, H.; YU, Q.; WANG, Z.; GENG, Y.; FU, F.; YANG, L.; ZHANG, W.; CUI, B. Retrieval-augmented generation for ai-generated content: A survey. **arXiv preprint arXiv:2402.19473**, 2024. Acesso em: 17 set. 2024.

APÊNDICE A – CRIAÇÃO DE INSTÂNCIAS NA AWS E GCP

A.1 AWS Console Web

A criação de uma máquina virtual no EC2 utilizando o formulário da Figura 15 envolve as seguintes etapas:

a) Acesse o AWS *Management Console* e navegue até o serviço EC2. Clique em *Launch Instance*.

b) Escolha uma AMI, que define o sistema operacional e as configurações iniciais.

c) Selecione o tipo de instância, com base em requisitos de CPU, memória e rede.

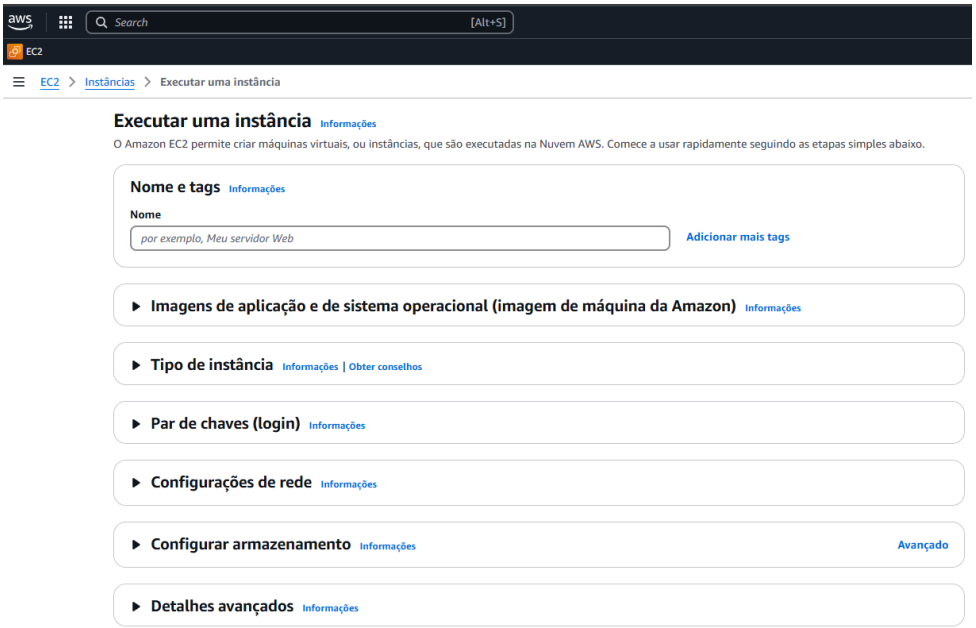
d) Configure as opções da instância, como rede VPC, sub-rede e IP público.

e) Configure os *Security Groups*, definindo as portas e protocolos permitidos para acesso à instância.

f) Revise as configurações e clique em *Launch*, selecionando ou criando uma chave de acesso para autenticação via SSH.

A Figura 15 apresenta o console EC2 da AWS, ilustrando os componentes essenciais para a criação de uma VM.

Figura 15 – Console EC2.



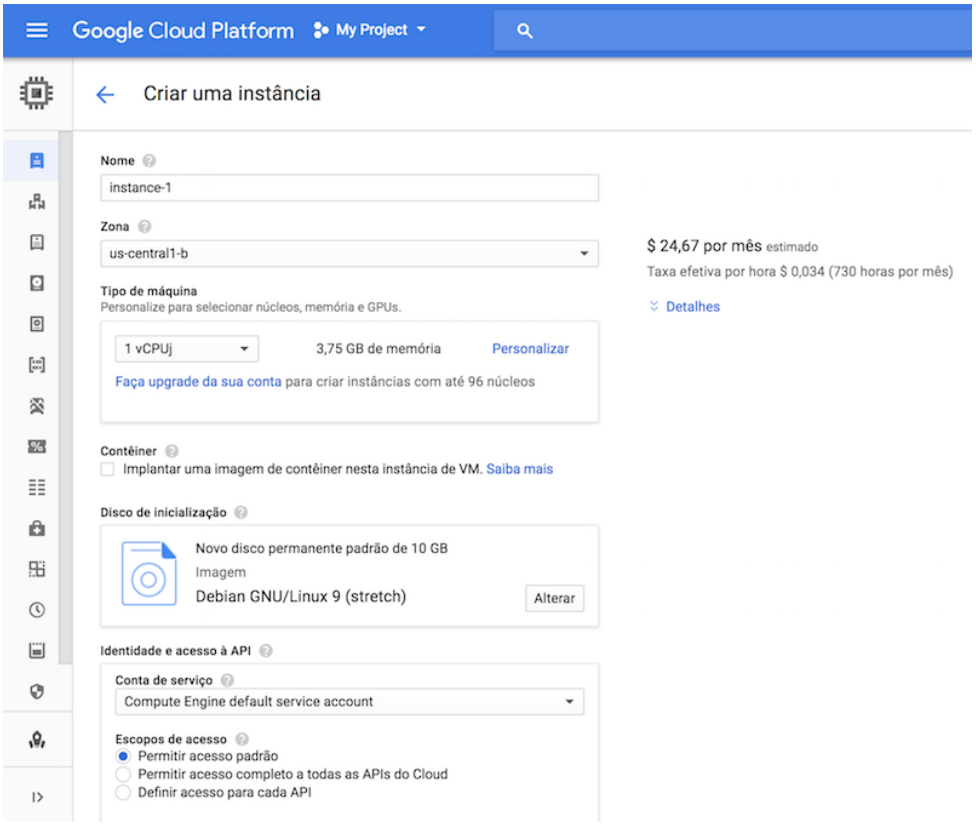
Fonte: AWS.

A.2 GCP Console Web

A Figura 16 apresenta o console da GCP, ilustrando os componentes para a criação de uma instância, é possível perceber que as opções de criação de cada nuvem são diferentes. São necessárias as seguintes etapas na Google Cloud:

a) Acesse o Google Cloud Console e navegue até Compute Engine > Instâncias de VM.
b) Clique em Criar Instância.
c) Escolha o nome da instância e a região/zona onde será hospedada.
d) Selecione a imagem do sistema operacional na seção Configuração de *Boot*.
e) Escolha a máquina virtual (ex.: *e2-micro*) em Configurações de Máquina.
f) Configure as redes e *firewalls* na seção Rede, permitindo ou bloqueando tráfego HTTP/HTTPS.
g) Clique em Criar para provisionar a máquina virtual.

Figura 16 – Console GCP



Fonte: GCP.

A.2.1 Exemplo de criação de VM na GCP

A seguir a Listagem de Código-fonte 1 apresentamos a criação de uma VM por pela linha de comando da *Google Cloud*:

Código-fonte 1 – Exemplo de Linha de Comando GCP

```
gcloud compute instances create "nome-da-instancia" \
  --zone "us-central1-a" \
  --machine-type "e2-micro" \
  --subnet "default" \
  --image-family "debian-11" \
  --image-project "debian-cloud" \
  --tags http-server,https-server
```

A.2.2 Exemplo de criação de VM na AWS

Na Listagem de Código-fonte 2 temos como é criada uma VM pelo AWS CLI:

Código-fonte 2 – Exemplo de Linha de Comando AWS

```
aws ec2 run-instances \
  --image-id ami-xxxxxx \
  --count 1 \
  --instance-type t2.micro \
  --key-name sua-chave \
  --security-group-ids sg-xxxxxx \
  --subnet-id subnet-xxxxxx
```

APÊNDICE B – ANÁLISE TÉCNICA DO IMPLEMENTADO RAG

Este apêndice descreve, de forma detalhada, a arquitetura técnica adotada no desenvolvimento do terminal inteligente baseado na estratégia de Geração Aumentada por Recuperação (RAG). São apresentados os principais módulos da aplicação, as bibliotecas e modelos empregados, bem como o fluxo operacional do sistema.

Componentes Principais do Sistema

1. Ingestão de Conhecimento (`md_ingestion.py`)

- **Objetivo:** Processar arquivos de documentação técnica (`.md`) da AWS CLI e GCP CLI, gerando uma base vetorial consultável para uso posterior no sistema.
- **Execução isolada:** Esta etapa é realizada previamente e apenas uma vez, antes do uso interativo da aplicação. Seu objetivo é alimentar o banco vetorial com conhecimento estruturado.
- **Operações realizadas:**
  - Leitura dos arquivos com `TextLoader`.
  - Fragmentação dos textos em blocos com `RecursiveCharacterTextSplitter`.
  - Enriquecimento dos blocos com metadados como serviço e comando.
  - Geração de vetores semânticos (*embeddings*) utilizando o modelo `bge-m3`.
  - Armazenamento persistente dos vetores no banco *ChromaDB*.

2. Recuperação e Rerank (`chat_reranker.py`)

- **Objetivo:** Recuperar os documentos mais relevantes em relação à pergunta do usuário, refinar a ordenação por meio de *rerank* e fornecer um contexto preciso para o modelo de linguagem.
- **Conceitos Aplicados:**
  - Recuperação vetorial com MMR, utilizando ***BiEncoder (bge-m3)*** para busca rápida e diversificada por similaridade semântica.
  - Reranqueamento com **CrossEncoder**, que avalia de forma precisa a relevância de cada par (pergunta, documento) com base em atenção cruzada.
  - Seleção dos *top-10* documentos com maior escore para uso no prompt.

  - Engenharia de *prompt*, estruturando entradas com foco em comandos das interfaces CLI dos provedores.
  - Geração da resposta com modelo de linguagem contextualizado (*prompt-injected*).

3. Modelagem (`models.py`)

- **Objetivo:** Centralizar e modularizar a definição dos modelos utilizados no sistema, promovendo desacoplamento e facilidade de manutenção.
- **Recursos:**
  - Abstração dos modelos de *embeddings* e LLM em uma camada única de configuração.
  - Facilidade para substituição de modelos sem impacto nas demais etapas do pipeline.

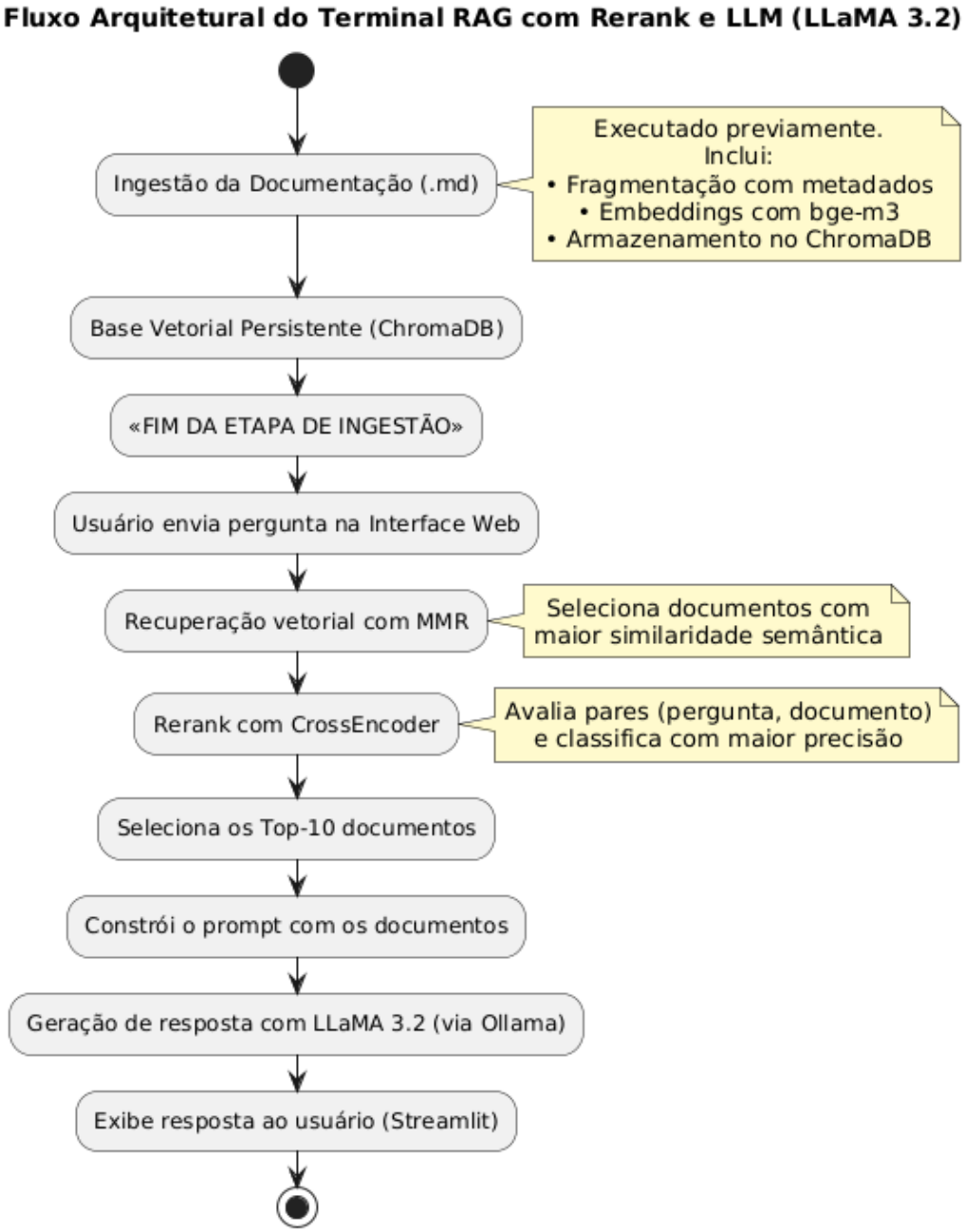
4. Interface Web (`interface.py`)

- **Objetivo:** Proporcionar uma interface leve e intuitiva para interação em linguagem natural com o terminal inteligente.
- **Recursos:**
  - Histórico de interações persistido via `st.session_state`.
  - Componentes reativos do `Streamlit`, como caixas de mensagem e entrada de texto.
  - Integração com o *back-end* de geração via chamada direta à função `get_response()`.

Justificativa das Escolhas Técnicas

- ***Bi-Encoder*** **(`bge-m3`):** Utilizado na etapa de recuperação inicial por sua capacidade de gerar *embeddings* densos com boa performance em busca vetorial, mesmo em ambientes locais.
- **Cross-Encoder:** Aplicado no reranqueamento dos documentos. Permite avaliar com maior precisão a relação semântica entre a pergunta e cada trecho recuperado, melhorando a relevância do conteúdo apresentado ao modelo de linguagem.
- **Combinação das arquiteturas:** A estratégia híbrida entre *Bi-Encoder* e *Cross-Encoder* ofereceu um equilíbrio entre escalabilidade e precisão, garantindo respostas mais relevantes com custo computacional viável.
- ***LLaMA 3.2 3B via Ollama*:** Modelo de linguagem com 3 bilhões de parâmetros, utilizado localmente com suporte ao *Ollama*. Escolhido pela compatibilidade com execução offline

Figura 17 – Fluxograma da execução do sistema com RAG e rerank.



Fonte: Elaborado pelo autor.

e pela flexibilidade de ajuste sem dependência de APIs externas.

- ***ChromaDB*:** Banco de dados vetorial leve, de código aberto, com suporte à persistência local. Ideal para protótipos com necessidade de recuperação semântica.
- ***Streamlit*:** *Framework web* para aplicações interativas em *Python*. Permitiu a criação rápida da interface, sem necessidade de desenvolvimento *front-end* adicional.