AUTOMATIZAÇÃO INTELIGENTE DE DOCUMENTAÇÃO DE SOFTWARE COM LLMS

INTELLIGENT AUTOMATION OF SOFTWARE DOCUMENTATION WITH LLMS

Francis Rubi Rotilli
Graduando em Ciências da Computação. Universidade Regional Integrada do Alto Uruguai e das Missões - URI – Campus de Santo Ângelo, Brasil. E-mail: francisrrotilli@aluno.santoangelo.uri.br

Denilson Rodrigues da Silva
Doutor em Educação nas Ciências. Universidade Regional Integrada do Alto Uruguai e das Missões - URI – Campus de Santo Ângelo, Brasil. E-mail: deniro@san.uri.br

Resumo: Este trabalho apresenta a construção de um modelo experimental de chatbot com Large Language Model (LLM) destinado a automatizar a gestão de documentação técnica para equipes de desenvolvimento de software. A pesquisa aborda o problema crônico da documentação desatualizada e da falta de padronização, problemas que, embora técnicos, constituem desafios de gestão do conhecimento e de riscos operacionais. O trabalho foi concebido utilizando o framework LangChain e um modelo local (on-premise) via Ollama, uma decisão estratégica que visa garantir a privacidade e o controle sobre os dados sensíveis da organização. A metodologia evoluiu de abordagens simples de engenharia de prompt para um robusto pipeline de processamento em duas camadas, o que permitiu uma classificação precisa das intenções do usuário e a execução de tarefas complexas. Os resultados demonstram a viabilidade técnica da solução, aprimorando a precisão das respostas e a fidelidade ao conteúdo da base de conhecimento, ao mesmo tempo em que aprimora a padronização dos processos. Conclui-se que a automação da documentação técnica por meio de LLMs é uma estratégia promissora para otimizar a produtividade e mitigar a dependência de conhecimento individual, contribuindo diretamente para os objetivos de inovação, transformação e inteligência estratégica no contexto de gestão contemporânea.

Palavras-chave: Inteligência Artificial, Gestão Estratégica, Inovação Tecnológica, Documentação de Software, Large Language Models.

Abstract: This work presents the construction of an experimental chatbot model using a Large Language Model (LLM) designed to automate the management of technical documentation for software development teams. The research addresses the chronic problem of outdated documentation and lack of standardization—issues that, while technical, constitute challenges in knowledge management and operational risk. The project was developed using the LangChain framework and an on-premise model via Ollama, a strategic decision aimed at ensuring the privacy and control of an organization's sensitive data. The methodology evolved from simple prompt engineering approaches to a robust two-layer processing pipeline, which enabled accurate classification of user intent and the execution of complex tasks. The results demonstrate the technical feasibility of the solution, improving the precision of responses and fidelity to the content of the knowledge base, while also enhancing process standardization. It is concluded that the automation of technical documentation through LLMs is a promising strategy for optimizing productivity and mitigating dependency on individual knowledge, contributing directly to the objectives of innovation, transformation, and strategic intelligence within the context of contemporary management.

Keywords: Artificial Intelligence, Strategic Management, Technological Inovation, Software Documentation, Large Language Models

Revista GESTO: Revista de Gestão Estratégica de Organizações
Santo Ângelo | v. 14 | n. 1 | p. 65-76 | 2026 | DOI: http://dx.doi.org/10.31512/gesto.v14i1.2447

1 Introdução

O mercado de desenvolvimento de software se expande em ritmo acelerado, exigindo que as organizações aprimorem continuamente suas metodologias e processos para manterem a competitividade. Na América Latina, um relatório da IDC "*FutureScape: Worldwide IT Industry 2023 Predictions*" prevê que, até 2028, mais de 35% dos orçamentos de TI das cinco mil maiores empresas da região serão alocados para ativos de conectividade, segurança, computação e dados, com o objetivo de fornecer processos próprios como serviço e produtos inteligentes. Nesse cenário dinâmico, a gestão eficaz de projetos e equipes de desenvolvimento tornou-se uma preocupação central. Um dos desafios mais persistentes, frequentemente subestimado, reside na gestão da documentação técnica. Conforme apontam Sommerville (2010), Statish e Anand (2016) e Aghajani (2019), problemas como a documentação desatualizada, a falta de padronização e a baixa adesão dos desenvolvedores são comuns e podem persistir por todo o ciclo de vida do software. Essas falhas, aparentemente técnicas, têm implicações diretas na eficiência e na qualidade do trabalho, resultando em retrabalho, dificuldades na integração de novos membros e perda de conhecimento organizacional, o que representa um risco estratégico para as empresas.

A automação de processos, impulsionada pelo crescente uso de inteligência artificial (IA) generativa, emerge como uma solução viável para superar esses obstáculos. Estudos de Huang e Rust (2018) e Brynjolfsson e Mcaffee (2017) demonstram que a automação de tarefas padronizadas por meio de IA traz benefícios significativos em tempo e custos. Nesse contexto, a aplicação de chatbots baseados em *Large Language Models* (LLMs) para o gerenciamento de documentação de software se apresenta como uma oportunidade de inovação. A adoção de tais modelos visa não apenas facilitar a consulta e o acesso às informações, mas também apoiar a flexibilização do processo de gestão documental, tornando-o mais eficiente e resiliente. O trabalho "*Reducing outdated and inconsistent code comments during software development*" de Adam Svensson (2015) já indicava a necessidade de soluções automatizadas, embora as limitações tecnológicas da época dificultassem a descrição da "intenção do desenvolvedor" e gerassem comentários problemáticos. As LLMs, com sua capacidade de processamento e geração de linguagem natural, superam essas limitações, permitindo a criação de ferramentas mais robustas.

Este trabalho se insere na temática "Inovação, Transformação e Inteligência Estratégica" do VI Congresso Internacional em Gestão Estratégica e Controladoria de Organizações (CIGECO). O estudo se alinha a essa temática ao apresentar uma inovação tecnológica que soluciona um problema crônico de gestão, ao mesmo tempo em que promove a transformação digital, reconfigurando os processos de trabalho e aprimorando a gestão do conhecimento nas organizações. A automação da consulta e criação de documentação permite que as equipes de desenvolvimento dediquem-se a tarefas de maior valor agregado, mitigando o risco de perda de conhecimento e dependência de pessoal. Essas características, que promovem a agilidade e a resiliência operacional, refletem os princípios da inteligência estratégica, que se baseia na utilização de dados, informações e conhecimento para a tomada de decisões. O objetivo é apresentar um método capaz de reduzir custos de tempo e melhorar a fidelidade da informação para as equipes, alinhando a solução tecnológica com os imperativos de gestão e poupando o tempo de supervisores em questões mais simples, como a consulta de padrões de nomenclatura para métodos de uma classe, por exemplo.

O artigo foi sintetizado do Trabalho de Conclusão do estudante e orientador autores, para o curso de Ciência da Computação da Universidade Regional Integrada do Alto Uruguai e Missões. Este está estruturado da seguinte forma: a seção 2 apresenta a fundamentação teórica, discutindo o papel da documentação, o avanço das LLMs, a engenharia de prompt e a governança de dados; a seção 3 detalha a metodologia de construção do modelo experimental; a seção 4 discute os resultados

Revista GESTO: Revista de Gestão Estratégica de Organizações
Santo Ângelo | v. 14 | n. 1 | p. 65-76 | 2026 | DOI: http://dx.doi.org/10.31512/gesto.v14i1.2447

e as implicações gerenciais; e, por fim, a seção 5 apresenta as considerações finais e as direções para pesquisas futuras.

2 Fundamentação teórica

O avanço das tecnologias de Processamento de Linguagem Natural (PLN), impulsionado pela popularização dos Modelos de Linguagem de Grande Escala (LLMs), tem provocado transformações significativas nas práticas tradicionais do desenvolvimento de software. Dentre essas práticas, destaca-se a documentação de código, frequentemente negligenciada, mas essencial para garantir a qualidade, a comunicação entre equipes e a manutenção eficiente dos sistemas. À medida que os projetos se tornam mais complexos e colaborativos, cresce a necessidade por soluções que promovam uma documentação mais acessível, precisa e passível de automação. Diante desse cenário, este capítulo tem como objetivo apresentar os fundamentos teóricos e técnicos que sustentam a proposta deste trabalho, que consiste no desenvolvimento e aplicação de um modelo baseado em LLMs para apoiar a geração e manutenção de documentação de software. Para tanto, são abordados conceitos centrais como o papel da documentação no ciclo de vida do software, as capacidades e limitações dos LLMs, os princípios da engenharia de prompts, o uso de frameworks[1] especializados como o LangChain, além dos desafios relacionados à segurança e privacidade dos dados. Com isso, busca-se estabelecer uma base teórica consistente que oriente e justifique as decisões técnicas e metodológicas adotadas na condução do projeto.

2.1 A Documentação de Software como Ativo Estratégico e Ferramenta de Gestão

A documentação de software transcende o papel de um mero registro técnico, constituindo-se em um ativo estratégico fundamental para o ciclo de vida de qualquer sistema. A sua principal função, além de descrever funcionalidades, é servir como um elo de comunicação entre os membros de uma equipe, tanto no presente quanto no futuro. Conforme aponta Pressman e Maxim (2016) , a documentação é um produto consumido por outros, tornando-se essencial para a colaboração e a manutenção de longo prazo. A ausência de uma documentação eficiente impacta negativamente a produtividade e a velocidade de entrega, além de aumentar o risco de dependência de conhecimento individual, o que pode comprometer a operação em caso de rotatividade de desenvolvedores.

No contexto das metodologias ágeis, a documentação não é abandonada, mas adaptada à dinâmica do desenvolvimento iterativo e incremental. Fowler e Highsmith (2001) argumentam que o foco se desloca da documentação excessiva para um equilíbrio com a comunicação contínua, priorizando a produção de artefatos concisos e atualizados *"just in time"*. Essa abordagem exige que as organizações desenvolvam mecanismos para garantir a consistência e a utilidade da documentação. Ferramentas que automatizam a geração de documentação, como docstrings, JavaDoc e Swagger/OpenAPI, são exemplos de como a padronização e a automação se tornaram pilares para a gestão eficaz do conhecimento técnico.

2.2 O Papel das LLMs na Inovação e Transformação Organizacional

Os Modelos de Linguagem de Grande Escala (LLMs), como GPT e LLaMA, representam um avanço significativo na área de Processamento de Linguagem Natural (NLP), com a capacidade de compreender, gerar e traduzir textos com notável precisão. Embora demonstrem uma capacidade

1 Um conjunto de ferramentas e códigos reutilizáveis que fornecem uma estrutura predefinida para acelerar o desenvolvimento de software.

Revista GESTO: Revista de Gestão Estratégica de Organizações
Santo Ângelo | v. 14 | n. 1 | p. 65-76 | 2026 | DOI: http://dx.doi.org/10.31512/gesto.v14i1.2447

impressionante de generalização, eles são inerentemente limitados ao conhecimento contido em seus dados de treinamento iniciais. Para superar essa limitação e fornecer respostas específicas para um contexto empresarial, a técnica de Geração Aumentada por Recuperação (RAG - *Retrieval-Augmented Generation*) tornou-se crucial. A técnica RAG combina a capacidade generativa do modelo com a busca em bases de conhecimento externas, permitindo que as respostas sejam mais completas, flexíveis e contextuais. Lewis e Perez (2021) validam a capacidade do RAG de ir além da simples extração e gerar respostas abstratas de forma livre, o que a torna ideal para aplicações corporativas.

No contexto do desenvolvimento de software, as LLMs estão sendo exploradas para acelerar processos, como a geração de documentação para descrições de funções, comentários de código e explicações técnicas. Ferramentas como GitHub Copilot e Amazon CodeWhisperer são exemplos práticos dessa aplicação, mostrando como modelos de linguagem podem reduzir o esforço necessário para manter a documentação atualizada. A eficácia dessa metodologia de geração de documentação é comprovada por estudos técnicos que demonstram ganhos em eficiência e na qualidade da saída gerada. Por exemplo, o trabalho de Zürcher (2024) mostra que empresas já utilizam modelos de linguagem para gerenciar as documentações internas com o objetivo de garantir a segurança e a privacidade dos dados. De forma semelhante, o projeto de Dhyani et al. (2024) sobre a automação da geração de documentação de APIs concluiu que a abordagem utilizada aprimorou significativamente a eficiência e a qualidade da saída do modelo, o que é comprovado pela redução do tempo de resposta e pela precisão da documentação gerada. Essas constatações reforçam a necessidade de soluções que conciliem eficiência com segurança e personalização.

2.3 A Engenharia de Prompt como Habilidade Crítica para o Gerenciamento de IA

A engenharia de prompt, definida como a formulação estratégica de instruções e contextos para induzir um LLM a gerar respostas relevantes e precisas, é um campo essencial para o sucesso da aplicação de LLMs em ambientes corporativos. Estudos como o de Sahoo etal. (2024) descrevem essa disciplina como uma "força transformadora" que desbloqueia o vasto potencial dos modelos de linguagem. A capacidade de projetar prompts claros e estruturados é fundamental para o controle sobre o comportamento do modelo, reduzindo a ambiguidade e o viés, e alinhando suas saídas aos objetivos estratégicos da organização.

A engenharia de prompt enfrenta desafios técnicos, como a ambiguidade na formulação da instrução e a limitação do tamanho do contexto, que podem levar a respostas imprecisas ou inconsistentes. No entanto, em ambientes controlados, a engenharia de prompt é frequentemente combinada com componentes auxiliares, como retrievers e formatters, para alcançar consistência e desempenho. Isso se torna uma competência técnica diferenciada, pois a precisão das respostas e a fidelidade às informações fornecidas como contexto dependem diretamente da qualidade das instruções. O estudo de Schulhoff et al. (2024) explora esses problemas e aponta formas de mitigá-los, como a atenção à ambiguidade e aos vieses, garantindo que o sistema de IA atue como uma ferramenta previsível e confiável, crucial para a governança de dados e a mitigação de falhas.

2.4 A Decisão Estratégica da Execução Local e a Governança de Dados

A privacidade e a segurança de dados são preocupações críticas na adoção de soluções baseadas em IA generativa, especialmente em contextos que envolvem informações sensíveis. A pesquisa da McKinsey & Company, citada pela XITE Create (2024), revela que mais de 50% das empresas consideram a IA generativa um risco à segurança de dados. Nesse cenário, a escolha de

Revista GESTO: Revista de Gestão Estratégica de Organizações
Santo Ângelo | v. 14 | n. 1 | p. 65-76 | 2026 | DOI: http://dx.doi.org/10.31512/gesto.v14i1.2447

um modelo de execução local (*on-premise*) em detrimento de uma solução baseada em nuvem não é apenas uma preferência técnica, mas uma decisão de inteligência estratégica para mitigar riscos de segurança e privacidade.

A execução local, como a proporcionada pela plataforma Ollama, oferece o controle total sobre os dados, flexibilidade de operações e conformidade legal com legislações como a LGPD no Brasil e a GDPR na Europa. Essa abordagem elimina a necessidade de enviar dados sensíveis para servidores externos, protegendo a empresa de possíveis vazamentos ou uso indevido. Em uma revisão dos riscos de privacidade em LLMs, Yan et al. (2024) afirmam que "os LLMs personalizados podem armazenar e processar dados sensíveis ao usuário, como conversas pessoais, pesquisas, consultas, ou histórico de navegação. Se não for adequadamente protegido, estes dados podem ser vulneráveis a acesso ou uso indevido não autorizado, levando a violações de privacidade e danos potenciais aos indivíduos". Embora, como apontado pelos autores, essa abordagem demande um maior esforço técnico e infraestrutura dedicada, o nível superior de segurança e previsibilidade justifica o investimento, tornando a solução mais resiliente e adequada para o ambiente corporativo com alta exigência de governança de dados.

A análise do referencial teórico demonstra a natureza multifacetada do problema da documentação de software e a complexidade de se aplicar soluções de IA generativa em um contexto corporativo. A documentação, que evoluiu de um mero registro para um ativo estratégico e uma ferramenta de gestão, exige abordagens inovadoras que se alinhem às práticas ágeis. Os LLMs, com sua capacidade de automação e flexibilidade, emergem como o caminho natural para essa inovação, mas sua implementação exige uma consideração cuidadosa de aspectos como segurança, privacidade e controle. A engenharia de prompt, por sua vez, mostra-se como a competência-chave para transformar a promessa teórica dos LLMs em uma realidade prática e confiável, enquanto a decisão pela execução local de modelos de linguagem é apresentada como a escolha estratégica para mitigar os riscos inerentes à manipulação de dados sensíveis. O conjunto dessas abordagens teóricas, que abarcam desde a gestão do conhecimento até a governança de dados, estabelece a base para os procedimentos metodológicos e a construção do modelo experimental detalhados a seguir.

3 Procedimentos metodológicos: a construção do modelo experimental

O modelo experimental, nomeado de "O Oráculo", foi arquitetado para demonstrar a viabilidade de um sistema de automação de documentação de software, inspirando-se em um modelo simplificado de chatbot assistente para processos internos. O escopo do chatbot foi expandido para se tornar uma via de duas mãos, capaz de receber novas informações e instruções, agregando-as à sua própria base de dados para uso futuro. O projeto foi arquitetado para abordar os desafios e vantagens de um chatbot que é alimentado e retorna informações sobre os processos internos de equipes de desenvolvimento.

3.1 Arquitetura do Sistema e Seleção de Ferramentas

A arquitetura do sistema se baseia na orquestração de componentes por meio do framework LangChain. A escolha do LangChain foi justificada por sua modularidade e capacidade de integrar modelos de linguagem com diversas fontes de dados e ferramentas externas, características que,

Revista GESTO: Revista de Gestão Estratégica de Organizações
Santo Ângelo | v. 14 | n. 1 | p. 65-76 | 2026 | DOI: http://dx.doi.org/10.31512/gesto.v14i1.2447

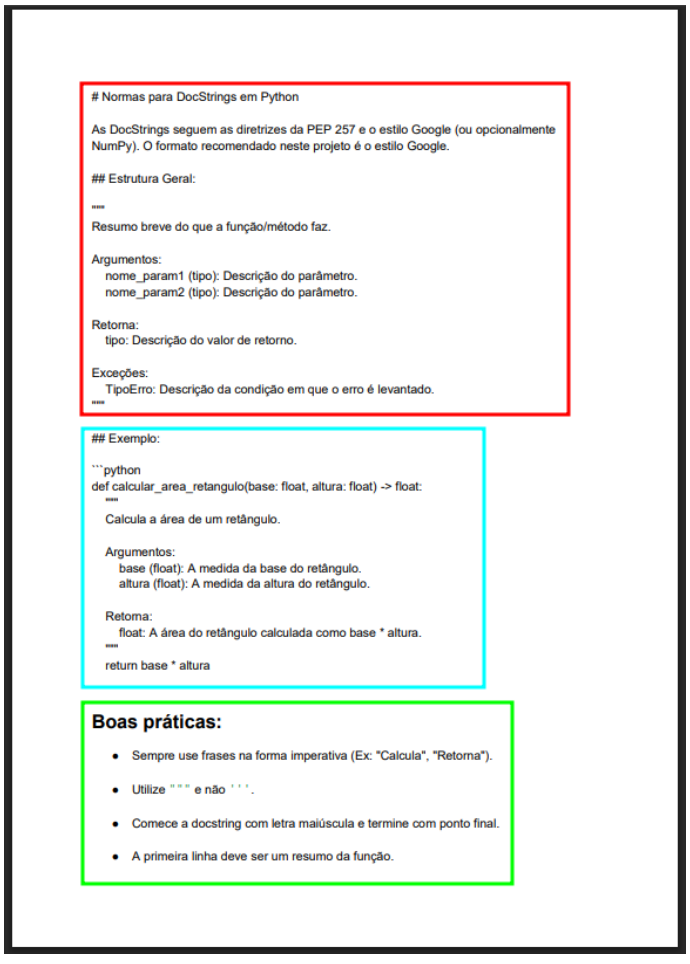
segundo o estudo comparativo da Conferência IJGIS (2024), o tornam a ferramenta mais versátil para interações complexas e escaláveis.

A interface de usuário foi implementada com a biblioteca de código aberto Streamlit, que permite a criação rápida de aplicações web interativas em Python, priorizando a simplicidade de implementação e a praticidade no desenvolvimento. O modelo de linguagem escolhido foi o llama3.1:8b-instruct-q4_K_S, executado localmente via Ollama. A decisão de utilizar um modelo on-premise foi uma escolha estratégica para garantir a privacidade dos dados da empresa, conforme discutido na seção 2.4 O modelo Llama3.1, com seus 8 bilhões de parâmetros e um contexto de até 128 mil tokens, oferece um desempenho robusto e é otimizado com métodos de Supervised Fine-Tuning e Reinforcement Learning with Human Feedback, o que garante um comportamento inicial aprimorado e uma resposta mais coerente.

3.2 Google Drive como Base de Dados Estratégica

A centralização da base de dados do sistema no Google Drive foi uma decisão estratégica para garantir que a solução pudesse se conectar a uma fonte de dados segura e amplamente utilizada, integrando-se ao fluxo de trabalho já existente de muitas equipes. Para viabilizar o acesso direto aos arquivos do Google Drive, foi necessário implementar um processo de autenticação OAuth 2.0, utilizando credenciais geradas no Google Cloud Console. O processo inicia-se com a criação de um arquivo credentials.json, que armazena as credenciais do cliente OAuth. O trecho de código abaixo é responsável por inicializar a autenticação no carregamento da aplicação pelo Streamlit:

Figura 1. Código responsável pela inicialização dos serviços do OAuth 2.0.



Fonte: Print screen do autor

Revista GESTO: Revista de Gestão Estratégica de Organizações
Santo Ângelo | v. 14 | n. 1 | p. 65-76 | 2026 | DOI: http://dx.doi.org/10.31512/gesto.v14i1.2447

Nesse trecho, destacam-se o uso do método run_local_server para capturar o token de autenticação, a definição de permissões via SCOPES e a criação dos serviços drive_service e docs_service para manipular arquivos e documentos. A função foi otimizada com o decorator

@st.cache_resource do Streamlit, impedindo que o fluxo de autenticação se repita em cada execução da aplicação. O processo de manipulação de arquivos foi desenvolvido com uma abordagem modular, alinhada com boas práticas de engenharia de software. Funções específicas foram criadas para: navegar por diretórios ou criá-los (get_or_create_folder_by_path); carregar o conteúdo de arquivos de forma recursiva (load_docs_recursive); e criar novos documentos (create_google_doc).

3.3 Organização da Base de Conhecimento Estratégica

A base de conhecimento do modelo foi estruturada no Google Drive, em pastas contendo arquivos de referência para três padrões de documentação técnica: Docstrings (Python), JavaDoc (Java) e OpenAPI/Swagger (REST APIs). Essa organização reflete uma abordagem de engenharia de prompt que combina instruções normativas, boas práticas e exemplos práticos, uma técnica que, conforme apontam LIU et al. (2023) e WEI et al. (2022), melhora significativamente o desempenho dos modelos de linguagem em tarefas específicas.

Na figura 2 está um dos documentos utilizados exemplificando a estrutura textual mencionada. Nele estão marcadas as seções com instruções normativas, exemplo prático e boas práticas, respectivamente em vermelho, ciano e verde.

Figura 2. Documento DocStrings.docx utilizado como referência do padrão de documentação DocStrings.

```python
import streamlit as st
from google_auth_oauthlib.flow import InstalledAppFlow
from googleapiclient.discovery import build
from config import SCOPES

@st.cache_resource
def get_gdrive_and_docs_services():
    flow = InstalledAppFlow.from_client_secrets_file("credentials.json", SCOPES)
    creds = flow.run_local_server(port=0)
    drive_service = build('drive', 'v3', credentials=creds)
    docs_service = build('docs', 'v1', credentials=creds)
    return drive_service, docs_service
```

Fonte: Print screen do autor

A escolha de uma base de conhecimento externa ao modelo de linguagem é um pilar da arquitetura RAG. Essa separação permite que o conhecimento do modelo seja atualizado dinamicamente sem a necessidade de um novo treinamento completo. O acesso seguro a essa base de dados, garantido pelo processo de autenticação via OAuth 2.0 e a delegação de domínio, é crucial para a gestão de riscos e a governança de dados em um ambiente corporativo.

3.4. A Evolução da Engenharia de Prompt: da Abordagem Simples ao Pipeline Estratégico

A metodologia de engenharia de prompt do projeto evoluiu de uma abordagem inicial e generalista para uma estratégia de pipeline em duas camadas. Inicialmente, foram realizados testes com prompts simples, baseados em zero-shot e few-shot prompting, onde o modelo era contextualizado com sua função e, em alguns casos, com exemplos de entradas e saídas. Esta abordagem inicial demonstrou ser eficaz para solicitações simples, como perguntas técnicas sobre

Revista GESTO: Revista de Gestão Estratégica de Organizações
Santo Ângelo | v. 14 | n. 1 | p. 65-76 | 2026 | DOI: http://dx.doi.org/10.31512/gesto.v14i1.2447

padrões de documentação, mas apresentava falhas em tarefas mais complexas. O modelo, embora capaz de entender a tarefa, gerava respostas inconsistentes, com excesso de tokens e contextualizações desnecessárias, e falhava em limitar o escopo de suas respostas à base de conhecimento fornecida, o que gerava "alucinações" ou desvios do conteúdo. A tentativa de resolver esses problemas adicionando mais regras e exemplos ao prompt monolítico se mostrou ineficaz, pois quanto mais variáveis eram introduzidas, mais as respostas divergiam, indicando uma dificuldade do modelo em gerenciar múltiplas instruções complexas em um único prompt.

Para resolver essas limitações, foi adotada uma estratégia que se baseia em um princípio de engenharia de software: dividir um problema complexo em partes menores e mais gerenciáveis. A nova abordagem, um pipeline de processamento em duas camadas, consistiu em:

- **Prompt de Classificação:** Um primeiro prompt, especializado em analisar a solicitação do usuário, tinha como única função classificá-la em uma das três categorias pré-definidas: requisicao_de_busca, requisicao_de_documentacao ou requisicao_de_edicao_da_documentacao. O retorno era um objeto JSON que, além da classificação, extraía os parâmetros essenciais para a próxima etapa (e.g., padrão, escopo, diretório).
- **Prompt de Execução:** Um segundo prompt, mais simples e focado, recebia o resultado da primeira etapa. Com base na classificação e nos parâmetros extraídos, o sistema fornecia ao modelo um contexto otimizado e específico para a tarefa, resultando em respostas mais precisas e objetivas.

Essa transformação metodológica demonstrou que a previsibilidade e a confiabilidade de um sistema de IA não dependem apenas da capacidade do modelo, mas da inteligência na orquestração das instruções, reconfigurando o processo para mitigar a ambiguidade e as falhas de interpretação.

4 Análise e discussão dos resultados

A avaliação do modelo experimental se concentrou em sua capacidade de processar, de forma precisa e confiável, as solicitações dos usuários, validando a eficácia da estratégia de engenharia de prompt de duas camadas. O desempenho do sistema foi analisado em termos de assertividade, fidelidade à base de conhecimento e coerência nas respostas. O conjunto de testes foi concebido para simular o ambiente de uso em uma equipe de desenvolvimento, abrangendo os três tipos de requisição mapeados: busca de informações, documentação de código e edição de documentos.

4.1 Avaliação de Desempenho do Modelo e da Estratégia de Prompts

A fase inicial de testes, com prompts mais simples (*zero-shot* e *few-shot*), revelou resultados inconsistentes e limitados, especialmente em tarefas complexas. O modelo tendia a gerar respostas com excesso de tokens, divagando sobre o assunto e falhando em limitar seu escopo ao conteúdo da base de conhecimento, o que levava a "alucinações". Por exemplo, ao receber uma pergunta sobre a estrutura de um padrão de documentação, o modelo, embora capaz de fornecer a resposta correta, frequentemente incluía contextualizações desnecessárias. A tentativa de fazer com que o modelo buscasse informações em uma base de dados interna recém-criada falhou completamente, pois ele não era capaz de restringir as respostas ao novo conhecimento, gerando informações genéricas ou erradas.

A transição para a estratégia de pipeline de duas camadas resolveu essas limitações de forma significativa. O primeiro prompt, focado exclusivamente na classificação da intenção do usuário, demonstrou um nível de assertividade notável. Foram realizados testes com diversos tipos de formulações, incluindo perguntas ambíguas, e o modelo classificou-as corretamente em 100% dos

Revista GESTO: Revista de Gestão Estratégica de Organizações
Santo Ângelo | v. 14 | n. 1 | p. 65-76 | 2026 | DOI: http://dx.doi.org/10.31512/gesto.v14i1.2447

casos, como evidenciado em testes finais. Ele conseguia, por exemplo, distinguir uma pergunta técnica sobre "como documentar com Javadoc" (uma requisicao_de_busca) de uma solicitação para "documentar este método com Javadoc" (uma requisicao_de_documentacao). O uso de uma estrutura de retorno em JSON permitiu extrair de forma precisa os parâmetros necessários (e.g., padrão, escopo), simplificando a etapa seguinte do processamento.

Na segunda etapa do pipeline, o prompt de execução, que recebia um contexto focado e otimizado a partir da primeira etapa, também apresentou um desempenho superior. Os testes de documentação de código mostraram que o modelo foi capaz de analisar trechos de código e gerar a documentação correta, seguindo as normas dos padrões fornecidos (Docstrings, Javadoc, etc.). Em solicitações de busca, o modelo respondeu de forma objetiva, técnica e direta, utilizando informações da base de conhecimento de forma precisa. O modelo demonstrou, inclusive, a capacidade de identificar quando a informação solicitada não existia na base de dados, informando claramente ao usuário, o que é crucial para evitar as "alucinações" e construir confiança no sistema.

A Tabela 1 sintetiza a evolução da metodologia de engenharia de prompt e seus respectivos resultados e implicações.

Tabela 1. Sumário da Evolução da Metodologia de Engenharia de Prompt e Resultados

Estratégia de Prompt	**Descrição**	**Resultados Obtidos**	**Implicações Estratégicas**
Zero-Shot e Few-Shot (Iniciais)	Contextualização simples e com exemplos.	Êxito em tarefas simples. Falhas na consistência das respostas, imprecisão e geração de excesso de tokens em solicitações complexas.	Falta de previsibilidade e confiabilidade. Baixo valor em um ambiente corporativo, onde a precisão é crítica para a gestão.
Pipeline de Duas Camadas (Final)	Separação lógica da tarefa em duas etapas: classificação da intenção do usuário (1º prompt) e execução da tarefa com contexto otimizado (2º prompt).	Classificação perfeita das solicitações e respostas objetivas, precisas e coerentes.	Aumento significativo da confiabilidade e precisão do sistema. Permite a criação de fluxos de trabalho robustos, automatizando processos complexos de forma segura e controlável.

A transição de um prompt monolítico para o pipeline de duas camadas é uma transformação metodológica que reflete a aplicação de um princípio de programação clássico para a IA. Ao documentar essa jornada, o estudo ilustra que a inovação é um processo iterativo e adaptativo, no qual as falhas iniciais são reconfiguradas para gerar uma solução mais robusta, um ponto central da "reconfiguração de processos".

4.2 Ganhos em Produtividade e Gestão de Conhecimento (Análise Gerencial)

A automação da documentação de software por meio do modelo proposto gera ganhos significativos que se traduzem em vantagens gerenciais e estratégicas. A principal contribuição está na otimização do tempo e do esforço de trabalho. Ao automatizar a criação de documentação para trechos de código, o modelo elimina a necessidade de os desenvolvedores realizarem manualmente uma tarefa repetitiva e muitas vezes negligenciada. Isso permite que eles redirecionem seu tempo e energia para atividades de maior valor agregado, como o desenvolvimento de novas funcionalidades ou a resolução de problemas complexos, resultando em um **aumento direto na produtividade** e na velocidade de entrega de projetos.

Revista GESTO: Revista de Gestão Estratégica de Organizações
Santo Ângelo | v. 14 | n. 1 | p. 65-76 | 2026 | DOI: http://dx.doi.org/10.31512/gesto.v14i1.2447

Outra vantagem crucial reside na **padronização e na gestão do conhecimento**. A documentação técnica, quando gerada pelo chatbot, segue as normas e os padrões da base de conhecimento de forma consistente, o que melhora a clareza e a colaboração entre as equipes. A automação também confere uma constância na execução e longevidade dos padrões adotados, reduzindo a divergência de informações em ambientes com diferentes estilos de trabalho. Adicionalmente, a solução **mitiga um risco estratégico crucial**: a perda de conhecimento organizacional. Ao centralizar o conhecimento técnico em um formato acessível e interativo, a empresa se torna menos dependente da expertise de indivíduos específicos. Isso não apenas **reduz o custo e a dificuldade de treinamento** de novos membros, mas também minimiza o impacto da rotatividade de pessoal. O chatbot atua como um repositório de conhecimento ativo, tornando a informação acessível e permanente, permitindo que novos integrantes sejam mais independentes e precisem de menos orientação para regras de negócio básicas.

4.3 Análise da Flexibilidade e Escalabilidade da Solução

A arquitetura modular e a seleção de ferramentas do projeto conferem flexibilidade e escalabilidade para a solução, tornando-a estrategicamente viável para adoção em um ambiente corporativo. A escolha do **LangChain** como framework de orquestração foi decisiva. Sua modularidade permite a fácil integração com outras fontes de dados além do Google Drive, bem como a adaptação para diferentes modelos de LLM. A capacidade do LangChain de criar *pipelines* e integrar Chains, Agents e Tools significa que a solução pode ser expandida para executar tarefas mais complexas no futuro, como interagir com sistemas de versionamento de código (Git) ou com outras bases de dados.

A decisão de utilizar a plataforma **Ollama** para executar o modelo localmente foi um fator determinante para a escalabilidade da solução. Enquanto modelos de alta complexidade demandam infraestrutura robusta, o modelo llama3.1:8b-instruct-q4_K_S, com seus 8 bilhões de parâmetros, opera de forma eficiente com apenas 8 GB de RAM. Essa característica torna a solução acessível para a maioria das empresas que podem implementá-la em sua própria infraestrutura, sem a necessidade de um investimento maciço em hardware dedicado. A natureza *on-premise* do sistema não apenas garante a segurança e a privacidade, mas também oferece flexibilidade para personalizar o modelo, adaptando-o às necessidades específicas da organização sem depender de APIs de terceiros.

4.4 O Protótipo como Exemplo de Inteligência Estratégica

O modelo experimental desenvolvido exemplifica a **inteligência estratégica** na prática ao abordar os desafios da gestão de software de forma proativa. O estudo não apenas propõe uma nova ideia para a documentação técnica, mas também demonstra como a tecnologia pode ser utilizada para reconfigurar um processo de trabalho tradicional. A solução LLM é apresentada como uma forma de **"utilização de dados, informações e conhecimento para a tomada de decisões estratégicas"**. O chatbot atua como um repositório de conhecimento ativo, que permite à equipe de desenvolvimento obter informações de forma rápida, precisa e confiável, o que é fundamental para a agilidade e a eficácia na resolução de problemas.

A automação de tarefas rotineiras libera tempo e recursos humanos para o foco em projetos de maior complexidade, onde a tomada de decisão é mais crítica. A confiabilidade das respostas e a mitigação de "alucinações" do modelo garantem que as decisões sejam baseadas em informações sólidas da base de conhecimento da empresa. Dessa forma, o protótipo transcende sua função técnica, tornando-se uma ferramenta de gestão que apoia a agilidade, a resiliência e a capacidade

Revista GESTO: Revista de Gestão Estratégica de Organizações
Santo Ângelo | v. 14 | n. 1 | p. 65-76 | 2026 | DOI: http://dx.doi.org/10.31512/gesto.v14i1.2447

de adaptação da organização, alinhando-se diretamente aos objetivos de inovação e transformação estratégica do evento CIGECO.

5 Considerações finais

Este trabalho demonstrou que a automação da documentação de software por meio de LLMs é uma abordagem promissora e tecnicamente viável. A pesquisa validou a eficácia de uma arquitetura baseada em *Retrieval-Augmented Generation* (RAG) e em um pipeline de engenharia de prompt em duas camadas, que se mostrou superior às abordagens mais simples ao garantir a precisão e a confiabilidade das respostas. A decisão estratégica de utilizar um modelo de execução local on-premise reforça a relevância do estudo para o contexto corporativo, onde a governança e a privacidade de dados são imperativos.

O estudo contribui para a área de gestão e inovação ao apresentar um modelo prático e seguro que aplica a inovação da IA para resolver um problema crônico de gestão do conhecimento. Ele oferece um framework conceitual para que outras organizações considerem a implementação de soluções semelhantes, ao mesmo tempo em que endereça as principais preocupações de segurança e privacidade. A automação da documentação, por meio de uma ferramenta como o chatbot, não se resume a uma otimização técnica, mas se configura como um processo de transformação que impacta diretamente na produtividade, na padronização e na resiliência da equipe de desenvolvimento. A capacidade do sistema de fornecer informações precisas e evitar a perda de conhecimento individual o estabelece como uma ferramenta de inteligência estratégica, que utiliza dados e conhecimento para apoiar a tomada de decisões no ambiente de desenvolvimento.

Apesar dos resultados positivos, é importante reconhecer as limitações inerentes ao escopo experimental deste trabalho. A avaliação do modelo foi restrita a um ambiente de testes controlado, o que representa um desafio de validação em um cenário real de uso por uma equipe de desenvolvimento. A ausência de uma análise aprofundada da usabilidade em um ambiente de produção real, por exemplo, é um ponto que trabalhos futuros podem e devem abordar. Além disso, a natureza dos modelos RAG — que são otimizados para a geração de texto — impõe barreiras à integração direta com ferramentas externas, como editores de código e sistemas de versionamento, mesmo com a capacidade do modelo de classificar a intenção do usuário para tal ação.

Em resposta a essas limitações e como uma reflexão para a evolução do trabalho, sugere-se uma expansão da pesquisa. As próximas etapas poderiam incluir a integração do modelo a sistemas de versionamento como o Git, o uso de *embeddings* personalizados para refinar a recuperação de informações e a realização de testes com usuários reais para avaliar a usabilidade e o impacto prático da solução. Outra linha de pesquisa promissora envolve a exploração de agentes autônomos que atuem de forma proativa na manutenção e atualização da documentação de projetos em tempo real, indo além de uma interface de chatbot reativa.

Em suma, o modelo desenvolvido é mais do que uma prova de conceito técnica; é um ponto de partida para a transformação digital e a inovação na gestão de equipes de software. O estudo oferece uma base concreta para futuras inovações na interseção entre inteligência artificial e engenharia de software, solidificando a automação da documentação como um pilar estratégico para a gestão do conhecimento e para a competitividade organizacional.

Revista GESTO: Revista de Gestão Estratégica de Organizações
Santo Ângelo | v. 14 | n. 1 | p. 65-76 | 2026 | DOI: http://dx.doi.org/10.31512/gesto.v14i1.2447

Referências

Aghajani, Emad, et al. (2019) **Software documentation issues unveiled**., IEEE/ACM 41st International Conference on Software Engineering (ICSE). IEEE, 2019.[1]

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P.,... & Amodei, D. (2020). **Language models are few-shot learners. Advances in neural information processing systems**, *33*, 1877-1901.[1]

Brynjolfsson, E., & Mcafee, A. N. D. R. E. W. (2017). **The business of artificial intelligence**. *Harvard business review, 7*(1), 1-2.[1]

Chen, Banghao et al. (2025). **Unleashing the potential of prompt engineering for large language models**. Patterns.

Dhyani, P., Nautiyal, S., Negi, A., Dhyani, S. and Chaudhary, P. (2024) **Automated API Docs Generator using Generative AI**, *2024 IEEE International Students' Conference on Electrical, Electronics and Computer Science (SCEECS)*, Bhopal, India, 2024, pp. 1-6.

Fowler, M., & Highsmith, J. (2001). **The agile manifesto**. *Software development, 9*(8), 28-35.

Huang, M.-H., & Rust, R. T. (2018). **Artificial Intelligence in Service**. *Journal of Service Research, 21*(2), 155-172.

Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N.,... & Kiela, D. (2020). **Retrieval-augmented generation for knowledge-intensive nlp tasks**. *Advances in neural information processing systems, 33*, 9459-9474.

Mavroudis, Vasilios. (2024). LangChain Ollama (2023). https://github.com/ollama/ollama/commits/main/?since=2023-01-01&until=2023-08-31.

Pressman e Maxim (2016). **Engenharia de Software: Uma Abordagem Profissional**. 9. ed.

Reynolds, L., & McDonell, K. (2021). **Prompt programming for large language models: Beyond the few-shot paradigm**. *In Extended abstracts of the 2021 CHI conference on human factors in computing systems*, pp. 1-7.

Sahoo, P., Singh, A. K., Saha, S., Jain, V., Mondal, S., & Chadha, A. (2024). **A systematic survey of prompt engineering in large language models: Techniques and applications**. *arXiv preprint arXiv:2402.07927*.

Schulhoff, S., Ilie, M., Balepur, N., Kahadze, K., Liu, A., Si, C.,... & Resnik, P. (2024). **The prompt report: A systematic survey of prompting techniques**. *arXiv preprint arXiv:2406.06608, 5*.

Sommerville, I. (2011) **Software Engineering** 9. ed. Pearson.

Statish, C. J. and Anand M. (2016) **Software Documentation Management Issues and Practices: a Survey**, *Indian Journal of Science and Technology, 9*.

Svensson, A. (2015). **Reducing outdated and inconsistent code comments during software development: The comment validator program**.

Topsakal, Oguzhan & Akinci, T. Cetin. (2023). **Creating Large Language Model Applications Utilizing LangChain: A Primer on Developing LLM Apps Fast**. *International Conference on Applied Engineering and Natural Sciences, 1*, 1050-1056.

Wei, J., Wang, X., Schuurmans, D., Bosma, M., Xia, F., Chi, E.,... & Zhou, D. (2022). **Chain-of-thought prompting elicits reasoning in large language models**. *Advances in neural information processing systems, 35*, 24824-24837.

Yan, B., Li, K., Xu, M., Dong, Y., Zhang, Y., Ren, Z., & Cheng, X. (2024). **On protecting the data privacy of large language models (llms): A survey**. *arXiv preprint arXiv:2403.05156*.

Zürcher, A. (2024) **Developing a Chatbot for Internal Documents**.