Universidade Federal do Piauí
Centro de Ciências da Natureza
Programa de Pós-Graduação em Ciência da Computação

Explorando o Potencial de *Large Language Models* para a Educação em Engenharia de Software

Bruno Mendes de Carvalho Castelo Branco

Teresina-PI, 15 de maio de 2025

Bruno Mendes de Carvalho Castelo Branco

Explorando o Potencial de *Large Language Models* para a Educação em Engenharia de Software

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Ciência da Computação da UFPI (linha de pesquisa: Computação Inteligente), como parte dos requisitos necessários para a obtenção do Título de Mestre em Ciência da Computação.

Universidade Federal do Piauí – UFPI

Centro de Ciências da Natureza

Programa de Pós-Graduação em Ciência da Computação

Orientador: Prof. Dr. Guilherme Amaral Avelino

Coorientador: Prof. Dr. Vinicius Ponte Machado

Teresina-PI

15 de maio de 2025

FICHA CATALOGRÁFICA
Universidade Federal do Piauí
Sistema de Bibliotecas UFPI - SIBi/UFPI
Biblioteca Setorial do CCN

B816e Branco, Bruno Mendes de Carvalho Castelo.
Explorando o potencial de *Large Language Models* para a educação em Engenharia de Software / Bruno Mendes de Carvalho Castelo Branco. -- 2025.
92 f.

Dissertação (Mestrado) - Universidade Federal do Piauí. Centro de Ciências da Natureza. Programa de Pós-Graduação em Ciências da Computação, Teresina, 2025.
"Orientador: Prof. Dr. Guilherme Amaral Avelino. Coorientador: Prof. Dr. Vinicius Ponte Machado".

1. Inteligencia Artificial Generativa. 2. Engenharia de Software. 3. Estatística. 4. Processamento de Linguagem Neural. I. Avelino, Guilherme Amaral. II. Machado, Vinícius Ponte. II. Titulo.

CDD 006.3

Bibliotecária: Caryne Maria da Silva Gomes - CRB3/1461

Bruno Mendes de Carvalho Castelo Branco

Explorando o Potencial de *Large Language Models* para a Educação em Engenharia de Software

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Ciência da Computação da UFPI (linha de pesquisa: Computação Inteligente), como parte dos requisitos necessários para a obtenção do Título de Mestre em Ciência da Computação.

Trabalho aprovado. Teresina-PI, 15 de maio de 2025:

Documento assinado digitalmente
gov.br
GUILHERME AMARAL AVELINO
Data: 26/05/2025 11:45:25-0300
Verifique em https://validar.iti.gov.br

Prof. Dr. Guilherme Amaral Avelino
Orientador

Documento assinado digitalmente
gov.br
VINICIUS PONTE MACHADO
Data: 27/05/2025 07:31:17-0300
Verifique em https://validar.iti.gov.br

Prof. Dr. Vinicius Ponte Machado
Co-Orientador

Documento assinado digitalmente
gov.br
PEDRO DE ALCANTARA DOS SANTOS NETO
Data: 27/05/2025 09:59:55-0300
Verifique em https://validar.iti.gov.br

Prof. Dr. Pedro de Alcantara dos Santos Neto
Docente Interno ao Programa

Documento assinado digitalmente
gov.br
DAVI VIANA DOS SANTOS
Data: 26/05/2025 11:52:30-0300
Verifique em https://validar.iti.gov.br

Prof. Dr. Davi Viana dos Santos
Docente Externo ao Programa

Teresina-PI
15 de maio de 2025

Aos meus, à ciência, à sociedade.

Agradecimentos

Agradeço aos meus pais, Fred e Ana, que desde o início dos planos de fazer um Mestrado me deram suporte para que eu seguisse em frente, mesmo nos momentos em que pensei em desistir, naqueles em que não queria falar sobre nada — apenas ficar ali — ou nos momentos em que meu psicológico não estava alinhado com aquilo que sou.

Agradeço, também, ao meu irmão André, por me dar suporte emocional nesta jornada, por ouvir parte das minhas lamúrias durante o processo, por me permitir exercitar os conhecimentos que adquiri nas disciplinas do Mestrado e pelas discussões sobre o uso de ChatGPT nas nossas horas vagas.

Estendo minha gratidão ao meu orientador, professor doutor Guilherme Amaral Avelino, por ter aceitado me acolher dentro do Programa de Pós-Graduação em Ciência da Computação da UFPI, ainda que não me conhecesse previamente, e por ter se mostrado solícito durante o desenrolar dos trabalhos, mesmo com os compromissos de um Coordenador do PPGCC e de um professor universitário.

Gostaria também de expressar minha gratidão ao meu coorientador, professor doutor Vinícius Ponte Machado, que, lá atrás, aceitou o contato de um desconhecido entusiasta de Inteligência Artificial, que viria a se tornar, mais tarde, seu aluno nas disciplinas do PPGCC e seu coorientando.

Por fim, agradeço aos demais membros da minha família, aos meus amigos da vida, do Rotaract, do trabalho e do próprio Mestrado, além dos mestres que passaram pela minha formação, seja nas graduações anteriores à minha formatura, seja agora na pós-graduação.

O Mestrado se mostrou uma experiência bastante desafiadora, principalmente porque muita coisa aconteceu em paralelo, interferindo na linearidade dessa jornada. O espectro da desistência rondava meus pensamentos com relativa frequência, mas vocês estiveram aqui para me dar a força necessária para continuar.

Por isso, este resultado não é só meu, é de vocês também. Muito obrigado!

"O indivíduo floresce quando o individualismo morre."
Nadejda Krupskaia

Resumo

O uso de ferramentas baseadas em modelos de linguagem de grande escala (LLMs), como o ChatGPT, tem crescido rapidamente em diversos domínios, inclusive na educação. No entanto, ainda são escassas as investigações empíricas sobre como essas ferramentas impactam o processo de aprendizagem em disciplinas como Engenharia de Software. Este trabalho tem como objetivo investigar como essas ferramentas podem ser utilizadas de forma eficaz em um ambiente real de ensino, durante as atividades de Levantamento de Requisitos e Testes. Para isso, foi conduzido um experimento com estudantes da disciplina Engenharia de Software II do curso de Ciência da Computação da Universidade Federal do Piauí, disponibilizando uma ferramenta de chat que utiliza com o modelo GPT-4o. Participaram 16 alunos, cujas 126 interações com a ferramenta foram analisadas com o uso de testes estatísticos, processamento de linguagem natural e questionários. Os resultados indicam que, embora o tamanho dos *prompts* tenha influência na satisfação dos alunos, o conteúdo dos *prompts* e a extensão das respostas geradas tiveram impacto mais relevante na avaliação positiva das interações. Além disso, estudantes com experiência profissional apresentaram uma abordagem mais técnica, com maior uso de trechos de código. A tarefa de Testes foi percebida como mais adequada ao uso da IA em comparação com a de Levantamento de Requisitos, devido à menor dificuldade ao adequar o conteúdo produzido pelo modelo nas suas aplicações. Este estudo contribui com evidências sobre o uso pedagógico de LLMs, propondo diretrizes para sua adoção em atividades práticas no ensino de Engenharia de Software e para avaliar o que foi produzido pelos alunos.

Palavras-chave: Inteligência Artificial Generativa, Engenharia de Software, Educação, Estatística, Processamento de Linguagem Natural.

Abstract

The use of tools based on large language models (LLMs), such as ChatGPT, has been rapidly expanding across various domains, including education. However, empirical investigations into how these tools impact the learning process in disciplines like Software Engineering remain scarce. This study aims to investigate how such tools can be effectively used in a real educational setting, specifically during Requirements Elicitation and Testing activities. To this end, an experiment was conducted with students enrolled in the Software Engineering II course in the Computer Science program at the Federal University of Piauí, using a custom chat tool powered by the GPT-4o model. Sixteen students participated, and a total of 126 interactions with the tool were analyzed through statistical testing, natural language processing, and questionnaires. The results indicate that while the length of the prompts influences student satisfaction, the content of the prompts and the length of the responses had a more significant impact on the likelihood of a positive evaluation. Additionally, students with professional experience demonstrated a more technical approach, with greater inclusion of code snippets in their prompts. The Testing activity was perceived as more suitable for using the AI tool compared to Requirements Elicitation, due to fewer difficulties in adapting the model's output to their project context. This study contributes empirical evidence on the pedagogical use of LLMs and proposes guidelines for their adoption in practical learning activities in Software Engineering, as well as for evaluating the content produced by students.

Keywords: Generative Artificial Intelligence, Software Engineering, Education, Statistics, Natural Language Processing.

Lista de ilustrações

Lista de tabelas

Sumário

1 Introdução

Este capítulo trará uma breve descrição do panorama do tema principal, do contexto, motivação, definição do problema, objetivos, justificativa e questões de pesquisa.

1.1 Contexto e Motivação

Até recentemente, era bastante comum o uso de Redes Neurais Recorrentes ou Convolucionais para qualquer tarefa que envolvesse o processamento de linguagem natural. Cada etapa – ou estado – do processamento era descrita em função do anterior e da posição do *input* dele no texto, o que tornava bastante custoso gerar *outputs* coerentes, mesmo aplicando técnicas como paralelização. No entanto, Vaswani et al. propuseram, primeiramente em 2017, uma nova arquitetura – o *Transformer* – no artigo *Attention is All You Need*, cujo funcionamento se baseia em um mecanismo de autoatenção (VASWANI et al., 2017).

Desde então, emergiram ferramentas que utilizam *Large Language Models* (LLMs), as quais possuem diversos casos de uso, inclusive a geração de código. A principal diferença desses modelos para as ferramentas de *code completion* – como o *Intellisense* – reside no fato de as LLMs serem capazes de receber, como *input*, textos escritos em linguagem natural e retornarem, a partir deles, código (YETIŞTIREN et al., 2023).

Dentre as ferramentas disponíveis, destacam-se o *GitHub Copilot* da Microsoft, desenvolvido a partir da plataforma Codex da OpenAI (CHEN et al., 2021), capaz de analisar o contexto no arquivo de código editado e oferecer sugestões (GITHUB, 2023); e o *ChatGPT*, da própria OpenAI, que, além de usos mais gerais como sumarizar informações, também pode ser utilizado para "gerar e debugar código, automatizar tarefas repetitivas e aprender novas APIs"(OPENAI, 2024b).

Desde o lançamento dessas ferramentas no mercado, diversos estudos têm sido realizados com o intuito de avaliar a precisão do código gerado (DENNY; KUMAR; GIACAMAN, 2023), coletar a avaliação qualitativa dos seus usuários (WANG et al., 2023b); (VAITHILINGAM; ZHANG; GLASSMAN, 2022), testar estratégias de elaboração de *prompts* (MASTROPAOLO et al., 2023) e elencar possíveis vantagens e desvantagens no seu uso, comparando seu desempenho com humanos (ASARE; NAGAPPAN; ASOKAN, 2023).

A fim de verificar o estado da arte do uso de ferramentas de IA Generativa dentro do contexto de Engenharia de Software, Nguyen et al. (NGUYEN-DUC et al., 2023) fizeram um mapeamento sistemático onde, com base no SWEBOK (SOCIETY, 2014),

elencaram algumas prioridades de pesquisa, as quais se seguem: Gestão de Engenharia, Engenharia de Requisitos, Projeto de Software, Manutenção e Evolução de Software, Garantia de Qualidade, Implementação de Software, Processos e Ferramentas de Software, Competências Profissionais, Educação em Engenharia de Software, Aspectos Macro e IA Fundamental.

Para cada um desses 11 tópicos, os autores elencaram o seu histórico de pesquisa, algumas possíveis questões de pesquisa, além de avaliar o estado da arte, os atuais desafios e as perspectivas futuras sobre os estudos dentro da área, envolvendo LLMs (NGUYEN-DUC et al., 2023).

1.2 Definição do Problema

A popularização de novas tecnologias impacta diversos setores que permeiam as nossas vidas, trazendo consigo novos desafios e oportunidades, o que não é diferente dentro do contexto da educação (AGUIAR, 2023). A pandemia de Covid-19 trouxe consigo uma aceleração desse processo, afetando diretamente todos os níveis de educação. Com isso, surgiram preocupações relacionadas à regulamentação dessas ferramentas e à otimização das interações dos alunos com elas (ANJOS; FRANCISCO, 2021).

Discute-se na literatura como esses avanços, inclusive no campo da Inteligência Artificial, influenciaram na mudança do perfil dos estudantes, haja vista que a maior facilidade de acesso à informação potencializa a aprendizagem e aumenta a sua independência (PARREIRA; LEHMANN; OLIVEIRA, 2021). No entanto, também podem dificultar a profundidade de análise desses conteúdos e interferir na percepção da realidade por parte dos estudantes (PARREIRA; LEHMANN; OLIVEIRA, 2021). Por conta disso, é importante avançarmos no sentido de desenvolver maneiras eficientes de adotar essas tecnologias nos currículos educacionais e de educar os estudantes para que a facilidade de navegar por vários conteúdos não seja confundida com a compreensão profunda destes (RODRIGUES; RODRIGUES, 2023).

Essas preocupações, juntamente com a lacuna na investigação dos impactos reais no aprendizado, riscos de dependência excessiva das ferramentas de IA e a necessidade do desenvolvimento de habilidades para formular perguntas eficazes (*prompt engineering*), também se aplicam ao ensino da Engenharia de Software (NGUYEN-DUC et al., 2023).

Neste contexto, é essencial investigar como as ferramentas de IA generativa estão influenciando o aprendizado dos estudantes de Engenharia de Software, tanto em termos técnicos quanto cognitivos, o que foi feito neste trabalho, a partir de uma pesquisa exploratória.

Portanto, definiu-se como objetivo investigar como otimizar o uso de ferramentas

baseadas em LLMs para promover o aprendizado em Engenharia de Software, especialmente nas tarefas de Levantamento de Requisitos e Testes. Para guiá-lo, foram elaboradas e respondidas as seguintes questões de pesquisa:

- **RQ1:** Qual a percepção dos alunos sobre a adequação do modelo GPT para as tarefas de Levantamento de Requisitos e Testes em Engenharia de Software?
- **RQ2:** Quais características dos *prompts* dos alunos causaram interações satisfatórias com a ferramenta?
- **RQ3:** Como a experiência prévia dos alunos com desenvolvimento de software influencia na maneira como utilizam o modelo?

1.3 Estrutura do Trabalho

O presente trabalho está dividido da seguinte forma: o Capítulo 1 fez uma introdução ao trabalho, delineando o problema de pesquisa e oferecendo uma visão abrangente do contexto e das questões que motivam o estudo. O Capítulo 2 aborda a preparação teórica necessária para o entendimento do estudo, apresentando as bases teóricas, quando aplicáveis, de Processos de Software, Estatística e Inteligência Artificial.

O Capítulo 3 expõe estudos anteriores relacionados ao uso de IA Generativa no âmbito do desenvolvimento de software e o Capítulo 4 detalha a abordagem metodológica adotada na pesquisa.

O Capítulo 5 apresenta os resultados obtidos após a realização do experimento, a partir dos quais são realizadas as interpretações e elaboradas as respostas das questões de pesquisa. No Capítulo 6, é discutido com mais profundidade as implicações dos resultados do estudo, bem como são elencadas as suas ameaças à validade. Finalmente, o Capítulo 7 sumariza o que foi obtido de resultados desse experimento e suas possíveis contribuições para o estado da arte no uso de IA Generativa dentro do contexto do ensino em Engenharia de Software, além de elencar possíveis trabalhos futuros.

2 Referencial Teórico

2.1 Processo de Desenvolvimento de Software

Um processo de software é definido como um conjunto de atividades metodológicas compostas por ações de engenharia de software. Cada ação é definida por um conjunto de tarefas, que determinam os artefatos de software a serem produzidos, os fatores de garantia de qualidade exigidos e os marcos do projeto para indicar progresso (VALENTE, 2020). Dentro dele, estão inclusos tanto o desenvolvimento de software a partir do zero como a extensão e modificação de sistemas existentes ou a configuração de componentes de software adquiridos (SOMMERVILLE, 2011).

De acordo com Sommerville, as atividades fundamentais que estão inclusas dentro de um processo de software são:

- **Especificação de Software**: Definição da funcionalidade do software e das restrições operacionais.
- **Projeto e Implementação**: Produção do software conforme as especificações.
- **Validação de Software**: Verificação para garantir que o software atenda às necessidades do cliente.
- **Evolução de Software**: Modificação do software para atender aos requisitos dos clientes que vierem a surgir.

A seguir, serão elencados alguns subprocessos que foram avaliados dentro do ensino da disciplina de Engenharia de Software e, consequentemente, fazem parte do escopo deste trabalho.

2.1.1 Levantamento de Requisitos

Os requisitos de um sistema são as descrições do que ele deve fazer, os serviços que ele oferece e as restrições que se aplicam ao seu funcionamento (SOMMERVILLE, 2011). Esses requisitos são divididos entre os Requisitos Funcionais, que definem o que o sistema deve ser capaz de fazer, e os Requisitos Não-Funcionais, que especificam as restrições aplicáveis ao seu bom funcionamento (VALENTE, 2020).

Sommerville também classifica os requisitos em Requisitos de Usuário, que são escritos em linguagem natural e diagramas para os usuários, e Requisitos de Sistema, que são descrições mais detalhadas das funções, serviços e restrições do software. Os

requisitos não funcionais incluem Requisitos de Produto, que tratam do desempenho do software; Requisitos Organizacionais, que lidam com políticas e procedimentos do cliente e desenvolvedor; e Requisitos Externos, que derivam de fatores como legislações em vigor e normas técnicas (SOMMERVILLE, 2011).

Os requisitos, ao serem elaborados, são estruturados no Documento de Especificação de Requisitos, acessado por clientes, engenheiros de sistema e desenvolvedores (SOMMERVILLE, 2011). Devido à natureza dinâmica do desenvolvimento de software, os requisitos podem mudar com o tempo, tornando o documento de requisitos desatualizado assim que o produto está finalizado, o que leva à adoção da sua coleta de forma incremental em métodos ágeis, como o Extreme Programming, e escritos na forma de estórias de usuário, sendo então priorizados (SOMMERVILLE, 2011; VALENTE, 2020).

Bons requisitos devem ser corretos, precisos, completos, consistentes e verificáveis, de modo a evitar imprecisões, ambiguidades e assegurar que tudo seja implementado conforme o previsto (VALENTE, 2020).

O processo de Engenharia de Requisitos envolve estudo de viabilidade, elicitação e análise de requisitos, especificação, validação e gerenciamento de requisitos (SOMMERVILLE, 2011). Em (FERNÁNDEZ et al., 2017), os autores identificaram problemas comuns na prática de Engenharia de Requisitos, como requisitos incompletos ou ocultos, falhas de comunicação entre a equipe de projeto e o cliente, e alvos móveis que causam mudanças frequentes nas necessidades e, consequentemente, nos requisitos do projeto.

A definição correta dos requisitos é de suma importância para o desenvolvimento de software, pois problemas nessa etapa podem levar a retrabalho e aumentar o custo do produto. A partir dos requisitos, podem ser elaborados casos de uso, que descrevem as interações do usuário com o sistema e incluem fluxos normais e extensões, como situações de erro (VALENTE, 2020).

2.1.2 Testes de Software e Desenvolvimento Dirigido a Testes (TDD)

Testes de software consistem em um conjunto de instruções planejadas e executadas sistematicamente para verificar se o programa cumpre suas funções e, caso contrário, corrigir defeitos antes do uso comercial (SOMMERVILLE, 2011). Tradicionalmente, os testes eram realizados em uma etapa separada, mas com a adoção de métodos ágeis, como o Test-Driven Development (TDD), essa prática foi integrada ao ciclo de desenvolvimento, onde testes são implementados antes das funcionalidades (VALENTE, 2020).

Testes eficazes são realizados em diferentes níveis: testes de unidade, que verificam pequenos trechos de código; testes de integração, que avaliam funcionalidades completas; e testes de sistema, que simulam o uso do sistema pelo usuário final (VALENTE, 2020). Além disso, testes podem ser classificados como caixa-preta, focados em requisitos funcionais,

ou caixa-branca, que analisam o funcionamento interno do código (PRESSMAN; MAXIM, 2016).

No TDD, o desenvolvimento de software é feito em etapas incrementais, onde para cada incremento é desenvolvido um teste automatizado. O código é implementado apenas após o teste falhar inicialmente, e o processo é repetido até que todos os testes sejam bem-sucedidos (SOMMERVILLE, 2011). Essa abordagem ajuda a evitar a omissão de testes, facilita a escrita de código altamente testável e melhora o design do sistema (VALENTE, 2020).

2.2 Conceitos Estatísticos

A seguir, serão apresentados alguns conceitos estatísticos que foram importantes para o desenvolvimento do estudo, em especial seus testes.

2.2.1 Testes não-paramétricos

São testes que independem de suposições sobre a distribuição da população, diferentemente dos testes paramétricos, e se aplicam a casos onde, dentre outras dificuldades, os dados coletados não seguem uma distribuição normal (SPIEGEL, 1993). Como eles foram executados por meio de bibliotecas pertencentes à linguagem de programação *Python* e não são o foco deste estudo, seus conceitos não serão aprofundados.

2.2.1.1 Teste de Shapiro-Wilk

É um teste de hipótese que avalia se os dados seguem uma distribuição normal (AJEE; VALSAN; SANKARAN, 2024), sendo este o melhor teste de aderência à normalidade (LEOTTI; BIRCK; RIBOLDI, 2005; SPIEGEL, 1993). A fórmula do Teste de Shapiro-Wilk é dada por:

$$W = \frac{\left(\sum_{i=1}^{n} a_i x_{(i)}\right)^2}{\sum_{i=1}^{n}(x_i - \bar{x})^2} \tag{2.1}$$

Onde:

- $x_{(i)}$ é o i-ésimo valor ordenado da amostra,
- $\bar{x}$ é a média da amostra,
- a_i são os coeficientes calculados a partir da distribuição normal e da amostra,
- n é o número total de elementos da amostra.

2.2.1.2 Teste U de Mann-Whitney

É um teste que verifica se duas amostras são originadas a partir de uma mesma população e uma tende a conter valores maiores que a outra (SPIEGEL, 1993). É dado por:

$$U = \min(U_1, U_2) \tag{2.2}$$

Onde:

$$U_1 = R_1 - \frac{n_1(n_1 + 1)}{2} \tag{2.3}$$

$$U_2 = R_2 - \frac{n_2(n_2 + 1)}{2} \tag{2.4}$$

Sendo:

- R_1 é a soma dos postos do primeiro grupo,
- R_2 é a soma dos postos do segundo grupo,
- n_1 é o tamanho da amostra do primeiro grupo,
- n_2 é o tamanho da amostra do segundo grupo.

2.2.1.3 Correlação de Spearman

É uma medida de associação entre duas variáveis, utilizada para os casos onde os dados não atendem ao pressuposto de normalidade (SPIEGEL, 1993). Sua fórmula é dada por:

$$\rho_s = 1 - \frac{6 \sum d_i^2}{n(n^2 - 1)} \tag{2.5}$$

Onde:

- d_i é a diferença entre os *rankings* de cada par de observações,
- n é o número total de pares de dados.

O valor de ρ_s varia entre -1 e +1, sendo o primeiro caso uma correlação perfeita negativa e, no segundo, perfeita positiva, significando que duas variáveis variam de maneira inversa ou juntas, respectivamente. Quando 0, representa a independência entre elas (SPIEGEL, 1993).

2.2.1.4 Regressão Logística

Regressão é uma descrição algébrica da relação entre duas ou mais variáveis, sendo elas uma variável dependente e uma ou mais variáveis independentes. No primeiro caso, consiste em uma regressão simples e, no segundo, em uma regressão múltipla (TRIOLA, 2008).

Uma regressão baseada em dados antigos pode não ser válida para dados mais novos e, também, ela é válida apenas para a população da qual se extraíram os dados. Existem vários modelos matemáticos que podem ser utilizados para a regressão, e a decisão depende do quão bem os dados se ajustam ao gráfico de uma função (TRIOLA, 2008).

Para os casos em que a variável dependente é qualitativa, utiliza-se a regressão logística, que prevê a probabilidade de um evento específico — no caso, da variável dependente assumir determinada categoria (FIGUEIRA, 2006).

A fórmula da Regressão Logística é dada por:

$$P(Y = 1 \mid X) = \sigma(X) = \frac{1}{1 + e^{-z}} \tag{2.6}$$

Onde:

- $P(Y = 1 \mid X)$ é a probabilidade predita de que o evento $Y = 1$ ocorra, dado o vetor de características X,
- $\sigma(X)$ é a função sigmoide, que mapeia qualquer valor real para o intervalo $(0, 1)$,
- e é a base do logaritmo natural, aproximadamente igual a 2.71828,
- z é a combinação linear das variáveis independentes, calculada por:

$$z = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots + \beta_n X_n \tag{2.7}$$

- β_0 é o intercepto, representando o valor da predição quando todas as variáveis independentes são zero,
- β_i é o coeficiente associado à variável independente X_i, indicando sua importância no modelo,
- X_i é o valor da i-ésima variável independente,
- n é o número total de variáveis independentes.

É possível observar o efeito do aumento de uma unidade em uma variável independente X_i na probabilidade do evento, ao considerar todas as demais constantes. Assim, obtém-se a razão de chances (ou *Odds Ratio*) (FIGUEIRA, 2006), que é dada por:

$$\text{Odds Ratio} = e^{\beta_i} \tag{2.8}$$

Onde:

- Odds Ratio é a razão de chances associada ao aumento de uma unidade na variável independente X_i,
- β_i é o coeficiente associado à variável independente X_i, indicando a variação na razão de chances para cada unidade adicional em X_i.

2.3 Conceitos de Inteligência Artificial

A seguir, serão apresentados os conceitos dos subcampos da Inteligência Artificial relacionados ao presente estudo: Processamento de Linguagem Natural e IA Generativa.

2.3.1 Processamento de Linguagem Natural

Processamento de Linguagem Natural (PLN) é um campo de pesquisa que se ocupa em desenvolver modelos computacionais e processos, a fim de resolver problemas práticos de compreensão da linguagem humana (OTTER; MEDINA; KALITA, 2019). O adjetivo "natural"é referente à linguagem falada por humanos e, dentro da Ciência da Computação, é uma área que está ligada à de Inteligência Artificial (CASELI; NUNES, 2023).

As análises feitas neste estudo foram:

- **N-gramas**: grupo com n palavras que aparecem em sequência. Um trigrama, por exemplo, é um grupo com três palavras que aparecem em sequência dentro de um dado texto (CASELI; NUNES, 2023). Na literatura, a análise de unigramas é sugerida como um dos processos de tratamento do corpus durante a mineração (MOURA et al., 2010), além de ser um dos elementos a serem avaliados para saber do que se trata o texto e qual o seu conteúdo para, dentre outras aplicações, detectar plágios (TORREJÓN; RAMOS, 2010).
- **Coocorrência**: no caso do estudo, nos referimos à matriz palavra-palavra, que é o número de vezes em que uma palavra da linha aparece no mesmo texto da palavra da coluna (CASELI; NUNES, 2023). Na literatura, a coocorrência já foi utilizada para descobrir a associação entre pesquisas científicas por meio de palavras-chave (FRANCO; FARIA, 2019), bem como a avaliação do que é mais abordado em estudos sobre determinado assunto, para encontrar padrões e possíveis pontos pouco explorados (MARQUES; MARQUES; MACULAN, 2021; SANTOS; REATEGUI; CAREGNATO, 2022).

2.3.2 Inteligência Artificial Generativa

Inteligência Artificial Generativa, como sugerido pelo nome, refere-se a modelos de IA capazes de gerar conteúdo novo e com coerência a partir de dados fornecidos (FEUERRIEGEL et al., 2024). Um componente central desses modelos é o mecanismo de autoatenção, que permite que a computação seja condicionada a todos os vetores ocultos da Rede Neural. No contexto de processamento de texto, por exemplo, cada palavra de uma entrada é codificada em vetores, que são analisados em conjunto para captar dependências globais dentro do dado (RUSSELL; NORVIG, 2022).

Entretanto, a autoatenção pode levar à perda de nuances importantes em dados complexos. Para mitigar isso, os pesquisadores desenvolveram o conceito de Atenção Multifacetada, que divide uma entrada em várias partes e aplica autoatenção a cada uma delas separadamente. Esse método não só preserva mais informações, mas também destaca elementos importantes para o contexto (RUSSELL; NORVIG, 2022).

Avançando nessa tecnologia, o modelo *Transformer*, introduzido no trabalho de (VASWANI et al., 2017), revolucionou o uso de autoatenção ao permitir que modelos de IA captem informações globais sem a necessidade de passar por sequências de dados de maneira linear. O *Transformer* é composto por camadas de *encoders* e *decoders*. Cada camada do Transformer processa sequências de símbolos, onde os *encoders* focam na codificação completa da entrada, enquanto os *decoders* trabalham na geração de saída baseada nesta codificação, considerando apenas as palavras até a posição *n-1* para cada palavra na posição *n*. Em cada camada, uma sequência de saída é gerada e alimentada na próxima camada, com a adição de conexões residuais para manter o contexto (RUSSELL; NORVIG, 2022). Ao final, uma função *softmax* seleciona a palavra mais apropriada ao contexto.

Os modelos de linguagem de larga escala (LLMs) têm demonstrado capacidades notáveis na geração de código a partir de descrições textuais. O *Codex*, por exemplo, é um modelo GPT ajustado com base em um vasto conjunto de dados de código disponível publicamente no *GitHub*. Um estudo revelou que, em um conjunto de avaliação chamado *HumanEval*, o *Codex* conseguiu resolver 28.8% dos problemas propostos, uma melhoria significativa em relação aos modelos anteriores como GPT-3 e GPT-J, que resolveram 0% e 11.4%, respectivamente (CHEN et al., 2021).

Os LLMs também têm a capacidade de analisar grandes volumes de código existente para extrair padrões, melhorar a refatoração e facilitar a compreensão de sistemas complexos. Essa aplicabilidade amplia o papel dos LLMs no desenvolvimento de software, desde a escrita de funções autônomas até a documentação automática e o suporte ao entendimento de código. No entanto, apesar de seus avanços, os LLMs ainda enfrentam desafios em termos de eficiência de amostras de treinamento e a capacidade de lidar com

especificações de código mais complexas e abstratas (CHEN et al., 2021).

2.3.2.1 IAs Generativas na Engenharia de Software

A Inteligência Artificial Generativa (GenAI) tem demonstrado grande potencial em diversas áreas da Engenharia de Software, impulsionada por avanços em arquiteturas de redes neurais, como o *Transformer* (VASWANI et al., 2017), e por modelos de linguagem de larga escala (LLMs) treinados em vastos conjuntos de dados de código (CHEN et al., 2021). Essa tecnologia tem a capacidade de automatizar tarefas, auxiliar na tomada de decisões e aumentar a produtividade dos desenvolvedores (DOHMKE; IANSITI; RICHARDS, 2023).

Na Engenharia de Requisitos (RE), a GenAI pode automatizar a elicitação, análise e classificação de requisitos, facilitando a comunicação entre *stakeholders* e a formalização das necessidades do software. Modelos de linguagem podem gerar requisitos abstratos, mas compreensíveis, embora desafios como a alucinação de requisitos e a falta de dados específicos para ajuste fino dos modelos ainda precisem ser superados (EZZINI et al., 2023; NGUYEN-DUC et al., 2023).

No design de software, ferramentas como o ChatGPT podem sugerir arquiteturas de sistema, identificar padrões de design apropriados e auxiliar na criação de diagramas, agilizando os processos de modelagem e prototipação. No entanto, a integração da GenAI em ambientes reais é limitada devido à falta de estudos em empresas (NGUYEN-DUC et al., 2023).

Durante a implementação, a GenAI pode oferecer suporte na geração, conclusão e sumarização de código. Ferramentas baseadas em LLMs são capazes de gerar código funcional a partir de descrições em linguagem natural (DAKHEL et al., 2023; IMAI, 2022), completar trechos de código de forma inteligente (SUN et al., 2022), e auxiliar na refatoração e documentação (MADAAN et al., 2023; NOEVER; WILLIAMS, 2023; AHMED; DEVANBU, 2023). No entanto, ainda existem desafios na compreensão precisa das entradas dos usuários e na garantia da segurança do código gerado (NGUYEN-DUC et al., 2023).

Na garantia de qualidade (SQA), a GenAI tem sido aplicada na automação de testes, previsão de falhas e análise de cobertura de código (LIU et al., 2023; YUAN et al., 2023; WANG et al., 2023a). A engenharia de prompts adequada e o refinamento dos modelos são essenciais para melhorar a eficácia da GenAI nessas tarefas (NGUYEN-DUC et al., 2023).

Na manutenção de software, a GenAI pode auxiliar na tradução de linguagens legadas para modernas, identificação de bugs e geração de *patches* (WEISZ et al., 2021). Apesar de ser eficaz na tradução e refatoração de código, a sua compreensão semântica e a limitação no contexto continuam sendo desafios (NGUYEN-DUC et al., 2023).

3 Trabalhos Relacionados

O presente capítulo apresenta trabalhos relevantes que tratam do uso de IAs Generativas, especificamente Large Language Models (LLMs), tanto dentro do processo de Desenvolvimento de Software *per se*, quanto no contexto educacional, com ênfase na Engenharia de Software, a fim de trazer o estado da arte.

Serão discutidas diferentes abordagens de integração de LLMs em cenários de aprendizagem, como a criação de tutores personalizados, a utilização de padrões de prompts para melhorar a modularidade do código e a simulação de sistemas e APIs.

3.1 LLMs no Contexto de Desenvolvimento de Software

(WHITE et al., 2023c) apresentam "padrões de *prompt*"reutilizáveis para diferentes atividades em Engenharia de Software. Esses padrões são projetados de modo a permitir que desenvolvedores estabeleçam regras que promovam a modularidade e a reutilização do código, além de simular sistemas, explorar alternativas arquiteturais desde as fases iniciais de design e acrescentar restrições a ele, quando pertinente.

Complementando o trabalho anterior, (WHITE et al., 2023a) focam na criação de um catálogo de padrões de *prompt* com o objetivo de melhorar especialmente a desambiguação de especificações e simulação de APIs, oferecendo uma abordagem que permite aos desenvolvedores explorar rapidamente diferentes possibilidades de design de código e de especificação de APIs, além de facilitar a simulação de mudanças nos sistemas de software com eficiência e menos propensa a erros.

(SUN et al., 2022) discutem a aplicação de modelos de linguagem de grande escala (LLMs) no desenvolvimento de *software*, destacando seu uso em tarefas como tradução de código, autocompletar e conversão de linguagem natural em código, exemplificado por ferramentas como o *GitHub Copilot*. São destacadas a capacidade de aumentar a produtividade dos desenvolvedores, mas também os desafios com relação à qualidade de saída e necessidade de explicabilidade, destacando a última, pois os desenvolvedores precisam entender a lógica, as entradas, as saídas e o desempenho dos modelos para utilizá-los de forma eficaz em seus fluxos de trabalho.

(DöDERLEIN et al., 2022) exploram o uso do *Codex* e *GitHub Copilot* no desenvolvimento de software, investigando sua eficácia na tarefa e a sua sensibilidade à variação dos parâmetros de entrada, como *prompts* e temperatura, mostrando que, quando configurados corretamente, podem melhorar a qualidade do código gerado. No entanto, a performance dos LLMs varia de acordo com as configurações específicas de cada problema, apontando

para a necessidade de uma compreensão mais profunda para otimizar seu uso em diferentes cenários de programação.

3.2 LLMs no Contexto do Ensino em Engenharia de Software

A utilização dessas ferramentas nos currículos demonstrou impactos variados na aprendizagem dos alunos. No curso CS50 — o curso introdutório de Ciência da Computação oferecido por Harvard na modalidade online, a integração de IA para orientá-los proporcionou uma experiência comparável à tutoria individual, embora houvesse desafios relacionados a como limitar o uso desenfreado pelos alunos, o que resultou na adoção de um sistema de vidas. Como resultado, os autores relataram um aprendizado mais profundo por parte dos estudantes (LIU et al., 2024).

No estudo de (LYU et al., 2024), por outro lado, foi avaliado o uso do *CodeTutor* — uma aplicação web que utiliza a API do OpenAI para oferecer suporte interativo aos alunos em tempo real, a qual mostrou melhorias importantes nas habilidades de pensamento computacional, embora a motivação para enfrentar desafios complexos não tenha sido impactada expressivamente, levando os autores a sugerirem que estratégias pedagógicas devessem ser desenvolvidas para melhorar o pensamento crítico e a autorregulação dos estudantes (LYU et al., 2024). Complementando o estudo anterior, (RAHMAN; WATANOBE, 2023) enfatizam a necessidade de desenvolver tarefas que exijam pensamento crítico por parte dos estudantes, de modo a guiar o seu uso da IA nas atividades de aprendizado.

Os padrões de uso das ferramentas de IA variaram entre os projetos analisados. A utilização de sistemas como o GPT-3.5-Turbo no *APAS Artemis* — um Sistema Automatizado de Avaliação de Programação utilizado para avaliar automaticamente exercícios de programação em cursos de engenharia de software — revelou vantagens em termos de *feedback* oportuno e de escalabilidade, mas também levantou preocupações sobre respostas genéricas e o potencial impacto negativo na progressão da aprendizagem dos alunos (FRANKFORD et al., 2024). O uso do ChatGPT em um curso de Programação Orientada a Objetos (POO) mostrou-se útil para a otimização e comparação de código, mas os ajustes durante o curso foram necessários para minimizar respostas muito diretas e estimular o raciocínio dos estudantes, independentemente das informações fornecidas pelas respostas da IA (KOSAR et al., 2024).

Além disso, em experiências envolvendo o desenvolvimento de software, o ChatGPT ajudou a melhorar a eficiência dos alunos e sua compreensão do conteúdo, mas destacou-se a necessidade do uso equilibrado da tecnologia para evitar dependências prejudiciais (WASEEM et al., 2024). (JOšT; TANESKI; KARAKATIč, 2024) relataram benefícios semelhantes aos do estudo anterior, acrescentando que a dependência excessiva dessas ferramentas pode prejudicar o desempenho acadêmico, em termos de nota, além da

habilidade dos alunos em resolver problemas de forma independente.

No trabalho de (MONTEIRO et al., 2025), foi feito um experimento em duas etapas, sendo a primeira envolvendo três usuários de diferentes níveis de experiência utilizando o próprio ChatGPT com o modelo GPT-3.5-Turbo para desenvolver uma página web e, a segunda, envolvendo um sistema personalizado sendo utilizando por 14 estudantes sem experiência. Os autores observaram uma dificuldade no uso do ChatGPT por desenvolvedores novatos e, como resposta, sugerem uma interface própria para esse público desenvolver suas atividades.

No sistema *TeachYou*, que adotou uma abordagem "*Reflect-Respond*", a qual alterna entre receber ajuda dos estudantes e questioná-los ativamente sobre o código produzido, foram obtidos resultados que indicam que essa dinâmica pode promover uma melhor construção de conhecimento pelos alunos (JIN et al., 2024).

Em alguns casos, como no estudo de (PETROVSKA et al., 2024), a implementação de atividades que incentivam a crítica das saídas da IA resultou em uma melhor compreensão dos materiais pelos alunos, sem que eles se tornassem excessivamente dependentes dessas tecnologias e, como consequência, a maioria deles demonstrou uma atitude positiva em relação ao uso dessas ferramentas no contexto educacional.

3.3 Diferenciais do Estudo

Embora muitos dos trabalhos revisados tenham explorado o impacto das IAs generativas em contextos diversos, o presente estudo se diferencia ao coletar, para análise, os dados de uso, por meio de uma ferramenta personalizada para auxiliar no ensino dentro de uma disciplina de Engenharia de Software, utilizando o GPT-4o como modelo base.

Ao contrário de abordagens mais genéricas, como as discutidas em estudos como (KOSAR et al., 2024) e (FRANKFORD et al., 2024), que analisam o uso de LLMs em cenários amplos de educação em programação, a ferramenta do nosso trabalho utiliza, além do texto enviado pelos alunos, *system prompts* especificamente desenhados para melhorar o contexto, adequando-o às tarefas e reduzindo a quantidade de informações necessárias a ser fornecida diretamente pelos alunos, permitindo que eles possam focar apenas nas suas necessidades.

Outros estudos, como o de (WASEEM et al., 2024), destacam a necessidade de um uso equilibrado para evitar dependências tecnológicas, o que levou nossa metodologia a implementar um sistema de gerenciamento de uso, com a limitação por "vidas"e por contexto, buscando incentivar o uso inteligente por parte dos alunos, direcionando o seu uso para a prática da disciplina.

Os alunos participantes deste estudo receberam um treinamento em *prompt en-*

gineering baseado em (WHITE et al., 2023c; WHITE et al., 2023a), onde aprenderam a aplicar alguns padrões reutilizáveis para as diferentes atividades da disciplina. O objetivo desta abordagem é maximizar os ganhos por meio da adoção de estratégias de *prompts* e da alfabetização em IA, mais especificamente LLMs, como proposto em (LYU et al., 2024), o que constitui mais um diferencial do presente trabalho.

Por fim, por meio da coleta dos dados realizada durante o uso da ferramenta pelos estudantes, foram feitas análises qualitativas e quantitativas do conteúdo produzido por eles, envolvendo a aplicação de testes estatísticos, aliados a análises textuais por meio do emprego de técnicas de processamento de linguagem natural. Com base nesses resultados, relacionaram-se as respostas dos formulários aplicados após cada uma das duas atividades da disciplina.

Esta análise multidisciplinar, além de configurar mais um diferencial do estudo, também é sua principal contribuição, visto que pode servir como um norte para experimentos futuros dentro do ensino em geral.

O quadro comparativo a seguir mostra as principais diferenças entre o presente estudo e aqueles mais relacionados, conforme os seguintes critérios, extraídos das suas respectivas metodologias e resultados:

- **Critério 1**: Utilizou-se de técnicas de PLN para analisar o conteúdo dos *prompts*?
- **Critério 2**: Testou-se estatisticamente os dados de uso para a análise?
- **Critério 3**: Os participantes foram treinados para usar a ferramenta/tecnologia analisada?
- **Critério 4**: O foco do estudo era no Ensino em Engenharia de Software?

Tabela 1 – Quadro comparativo entre o presente estudo e os mais relacionados.

Trabalho	**Critério 1**	**Critério 2**	**Critério 3**	**Critério 4**
(LIU et al., 2024)	Não	Não	Sim	Não
(LYU et al., 2024)	Não	Sim	Sim	Não
(FRANKFORD et al., 2024)	Não	Não	Não	Sim
(WASEEM et al., 2024)	Não	Não	Não	Sim
(JOšT; TANESKI; KARAKATIč, 2024)	Não	Sim	Não	Sim
Este trabalho	Sim	Sim	Sim	Sim

4 Metodologia

Neste capítulo, será apresentada em detalhes a metodologia utilizada para o desenvolvimento desta pesquisa, desde o desenvolvimento do chat que consome a API da *OpenAI* até a análise dos dados.

4.1 Estudo Exploratório

O presente trabalho consistiu em um estudo exploratório realizado em um ambiente real de ensino de Engenharia de Software, onde os alunos utilizaram a ferramenta desenvolvida para auxiliá-los durante as atividades de Levantamento de Requisitos e Testes de suas respectivas aplicações. Foram armazenados, entre outros dados, os *prompts* dos alunos, as respostas da ferramenta a esses *prompts*, o nível de expertise dos alunos ("apenas estudante", quando não trabalham; "estagiário", quando atuam no mercado, porém em caráter de aprendizado; e "profissional", quando trabalham no mercado com carteira assinada ou pessoa jurídica) e, opcionalmente, sua avaliação da interação — *prompt* + resposta — que poderia ser positiva, na forma de "curtida", ou negativa, na forma de "não curtida". Esses dados, aliados aos questionários aplicados, compuseram o conjunto de dados analisados. A seguir, serão descritas com mais detalhes cada uma das etapas do trabalho.

4.1.1 Desenvolvimento e Implementação da Ferramenta

Primeiramente, foi planejada e desenvolvida a ferramenta utilizada para coletar informações sobre o uso e controlar o que seria produzido pelos estudantes. Para isso, foram utilizados *Python* com *FastAPI* no *back-end*, *Firebase* para o armazenamento de dados, *Flutter* no *front-end* e a API da OpenAI com o modelo GPT-4o. A API foi hospedada na plataforma *Heroku*, enquanto a aplicação foi hospedada no *Netlify*. A *stack* foi escolhida com base na praticidade no desenvolvimento, bem como pela expertise do autor com as tecnologias envolvidas. O ambiente é bastante similar ao do ChatGPT, no qual os alunos podem iniciar a conversa selecionando o assunto sobre o qual desejam interagir.

Além disso, foi implementado um sistema de “vidas”, inspirado no sistema de *tokens* de uso feito por (LIU et al., 2024), onde cada *prompt* enviado consome uma vida, que é recuperada a cada 5 minutos, contados a partir da data e hora do último *prompt*. Com essa abordagem, buscou-se estimular, por meio da escassez de usos, o planejamento dos *prompts* e, consequentemente, o uso inteligente da ferramenta, além de reduzir os custos com a *OpenAI*, *Firebase* e os serviços de *deploy*.

Mais detalhes das telas de *login* podem ser visualizados na figura 1 e da tela do

chat, com a visão da lista de conversas, seus respectivos assuntos, quantidade de vidas e tempo para recuperação, na figura 2.

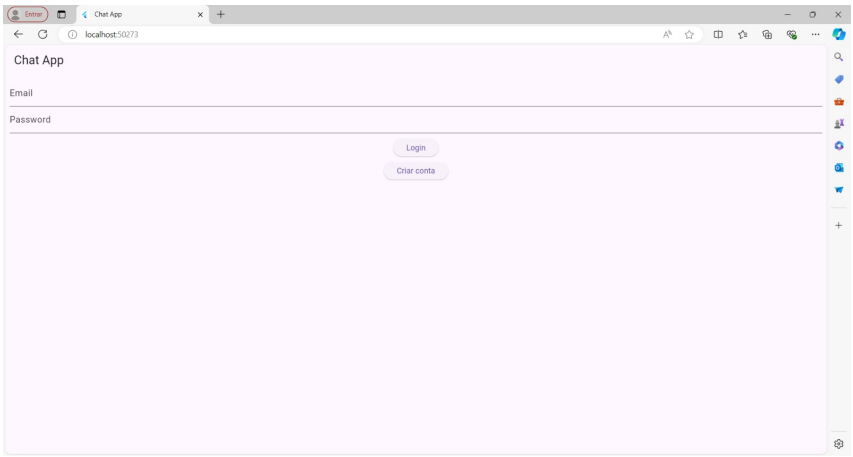

Figura 1 – Tela de Login da Ferramenta.

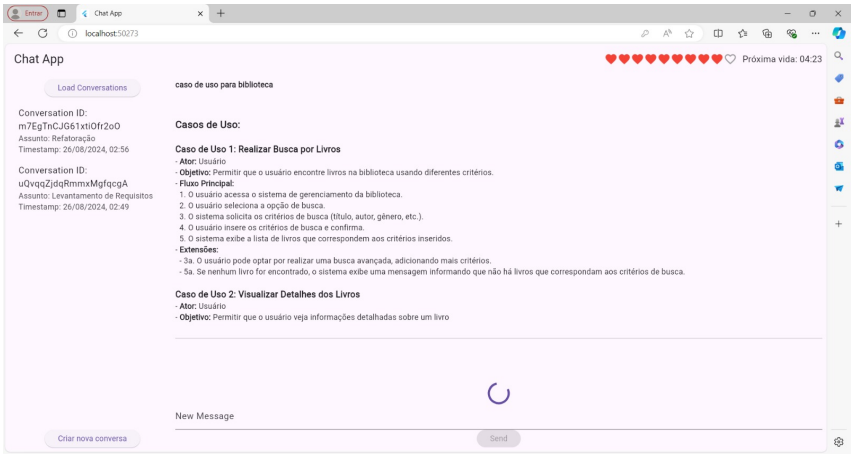

Figura 2 – Tela Inicial da Ferramenta.

4.1.2 Estratégias de Prompts

Com base nos conceitos de Engenharia de Software levantados durante a revisão de literatura, foram elaborados os *system prompts* (*prompts* de sistema), levando em consideração as boas práticas recomendadas pela documentação oficial (OPENAI, 2024a; OPENAI, 2024c; OPENAI, 2024d), os quais forneceram um contexto adicional sobre a atividade a ser desempenhada pelo usuário, estabelecendo regras para o funcionamento do modelo dentro da aplicação. O objetivo era permitir que os alunos se concentrassem apenas no que precisavam.

A estrutura de um *system prompt* é a seguinte: a primeira parte indica para o modelo qual é o tema do conhecimento sobre o qual ele irá trabalhar; a segunda parte contém o conteúdo do conhecimento propriamente dito; e a terceira parte são as regras que irão delimitar suas respostas, como a restrição na geração de código, de modo a evitar que

os alunos utilizem a ferramenta sem um papel ativo no desenvolvimento das atividades. O fluxo está ilustrado na figura 3.

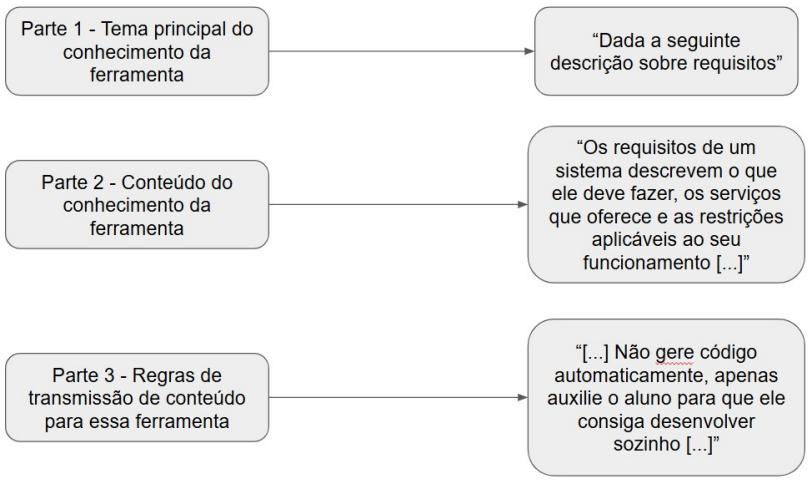

Figura 3 – Estrutura básica de um *system prompt*.

Como os *system prompts* foram elaborados com muitos detalhes, espera-se que os *prompts* dos usuários sejam mais curtos. Com isso em mente, as respostas foram limitadas a 700 *tokens*, valor suficiente para que a ferramenta retorne soluções viáveis. Durante a sua elaboração, os *system prompts* foram testados dentro do ChatGPT, com o mesmo modelo utilizado na API (GPT-4o). O conteúdo completo dos *system prompts* elaborados encontra-se no apêndice A.

4.1.3 Dados Armazenados

O armazenamento do sistema foi feito com o *Firebase*, um banco de dados NoSQL que trabalha com a estrutura de documentos. Todos os dados coletados foram salvos em três coleções: *Users* — a de usuários, *Conversations* — a de conversas, e *Prompts* — a de *prompts* enviados, suas respostas e as avaliações dos usuários, conforme ilustrado nas Tabelas 2, 3 e 4, respectivamente.

Tabela 2 – Tabela da Coleção `Users`

Campo	Tipo	Descrição
email	string	Endereço de e-mail do usuário
enrollment	string	Número de matrícula do usuário
expertise_level	string	Nível de experiência do usuário
last_prompt_time	timestamp	Data e hora da última interação com o sistema
lives	inteiro	Número de "vidas"restantes do usuário
password_hash	string	Hash da senha do usuário (armazenada de forma segura)

Tabela 3 – Tabela da Coleção `Conversations`

Campo	Tipo	Descrição
id	string	Identificador único da conversa
conversation_number	inteiro	Número sequencial da conversa
system_prompt	string	Prompt do sistema para a conversa
theme	string	Tema da conversa
timestamp	timestamp	Data e hora em que a conversa foi criada
user_id	string	Identificador do usuário que iniciou a conversa

Tabela 4 – Tabela da Coleção `Prompts`

Campo	Tipo	Descrição
id	string	Identificador único do prompt
conversation_id	string	Identificador da conversa associada
disliked_interaction	booleano	Indica se a interação foi desgostada (verdadeiro/falso)
liked_interaction	booleano	Indica se a interação foi apreciada (verdadeiro/falso)
prompt_number	inteiro	Número sequencial do prompt dentro da conversa
prompt	string	O texto do prompt enviado ao modelo
response	string	A resposta do modelo ao prompt
timestamp	timestamp	Data e hora em que o prompt foi enviado
user_id	string	Identificador do usuário que enviou o prompt

4.1.4 Jornada do Usuário

Ao abrir a ferramenta, o usuário visualiza uma tela inicial, onde poderá fazer o login ou se registrar. Após autenticar-se, o sistema buscará as conversas salvas no banco de dados, além de possibilitar a criação de novas conversas. Quando o usuário escrever e enviar o *prompt*, este será, simultaneamente, salvo no banco do *Firebase* e enviado para a API da *OpenAI*, que retornará a resposta com os parâmetros definidos.

Uma vez retornada a resposta, ela será não apenas exibida ao usuário, como também armazenada no banco do *Firebase*, garantindo que o histórico fique salvo. O esquema está ilustrado na Figura 4.

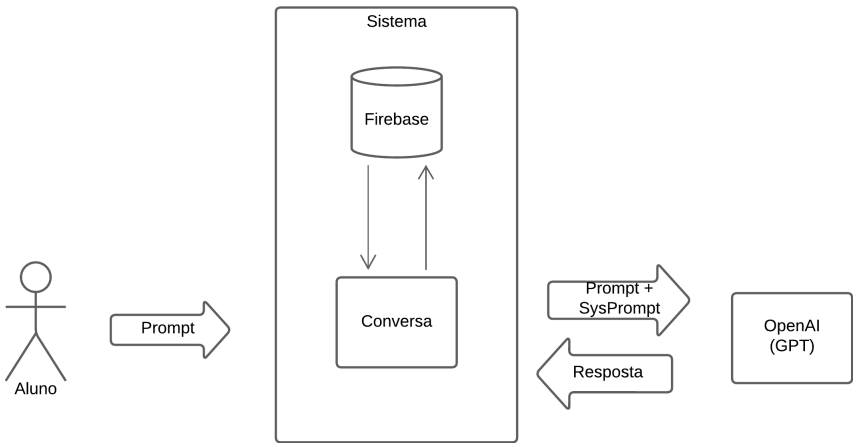

Figura 4 – Jornada de uso da ferramenta pelo usuário.

4.2 Realização do Experimento

Na segunda etapa do estudo, foram ministradas aulas aos alunos apresentando a ferramenta e as atividades que iriam desempenhar com ela. Esses tópicos serão abordados com mais detalhes a seguir.

4.2.1 A Disciplina

O presente estudo foi desenvolvido durante a disciplina de Engenharia de Software II, do curso de Bacharelado em Ciência da Computação da Universidade Federal do Piauí. No decorrer da grade, é ensinado aos alunos metodologias ágeis de desenvolvimento de software e, concomitantemente a isso, os alunos formam grupos, os quais escolhem um tema e, à partir dele, desenvolve um MVP (*minimum viable product*, ou produto mínimo viável) que atue em cima das demandas elencadas.

O processo de desenvolvimento passa por todas as etapas dos processos de software, de modo que, após a realização das aulas teóricas sobre eles, aos alunos é dado um prazo para aplicar os conhecimentos adquiridos sobre a etapa estudada e desenvolver os artefatos inerentes a cada um.

Um exemplo desse fluxo é o que se segue: para a etapa de Levantamento de Requisitos, os alunos assistem a aulas teóricas sobre os assuntos relacionados a esse processo e, depois, é dado um prazo para que eles apliquem os conhecimentos adquiridos e desenvolvam o processo para gerar o documento de requisitos do software que irão desenvolver. Ao final, a documentação é corrigida e, então, atribuída uma nota aos alunos.

O que o estudo fez foi, juntamente com as aulas teóricas sobre o processo em questão, ensinar aos alunos maneiras de utilizar as LLMs para auxiliar nessas atividades, detalhado melhor a seguir.

4.2.2 Treinamento dos Alunos

Para cada uma das duas atividades de desenvolvimento de software às quais a ferramenta foi aplicada, foi ministrada uma aula de treinamento. Nela, foi explicada brevemente cada uma dessas atividades, bem como a importância de um padrão de *prompt* para o uso mais eficiente do modelo, aplicado a cada uma.

Para a elaboração do material de aula, especialmente dos padrões sugeridos, foram utilizados os conteúdos presentes em (AMATRIAIN, 2023; WHITE et al., 2023b; OPENAI, 2024c; OPENAI, 2024d) como base, com alguns exemplos de uso, visando auxiliar os alunos na adoção dos padrões para otimizar seus resultados. Os padrões sugeridos para Levantamento de Requisitos estão no Apêndice B e, para os Testes, no Apêndice C.

4.2.3 Questionário de Avaliação

Para complementar os dados de uso coletados, foi aplicado um questionário de avaliação para cada uma das duas atividades, com o objetivo de coletar *feedback* detalhado dos alunos sobre sua experiência com a ferramenta. As perguntas foram estruturadas utilizando a escala de *Likert*, perguntas de sim ou não e perguntas de texto aberto, a fim de capturar de forma mais precisa os sentimentos dos alunos.

As respostas de ambos os questionários, juntamente com os dados de uso coletados, foram analisadas e resultaram nas interpretações feitas no Capítulo 6. Cada um dos formulários continha o Termo de Consentimento para Coleta de Dados, com concordância obrigatória para que houvesse a coleta das respostas (Apêndice D), e as questões foram adaptadas para a tarefa específica, seja Levantamento de Requisitos, seja Testes (Apêndice E).

4.3 Análise dos Dados

A última etapa da metodologia envolve a análise dos dados coletados durante as etapas anteriores. Para este fim, foi utilizada a linguagem de programação *Python*, principalmente por meio das bibliotecas especializadas em ciência de dados, como Pandas, Scipy, Numpy, entre outras; e em processamento de linguagem natural, como a NLTK. Um *endpoint* específico foi elaborado para exportar os dados de maneira estruturada no formato CSV, conforme ilustrado na Tabela 5.

Tabela 5 – Campos exportados em CSV das coleções `conversations`, `prompts` e `users`

Campo	**Tipo**	**Descrição**
enrollment	string	Número de matrícula do usuário
expertise_level	string	Nível de experiência do usuário
conversation_id	string	Identificador da conversa associada
theme	string	Tema da conversa
liked_interaction	booleano	Indica se a interação foi curtida (verdadeiro/falso)
disliked_interaction	booleano	Indica se a interação foi "descurtida"(verdadeiro/falso)
prompt	string	O texto do *prompt* enviado ao modelo
response	string	A resposta do modelo ao *prompt*
prompt_tokens	inteiro	Quantidade de *tokens* do *prompt*
response_tokens	inteiro	Quantidade de *tokens* da resposta

A metodologia de análise dos dados será descrita em três seções separadas: uma para as análises e testes estatísticos, outra para as análises do conteúdo dos *prompts* com processamento de linguagem natural e, por fim, a análise das respostas dos alunos aos formulários.

4.3.1 Estatística

Neste processo, foi realizado o tratamento dos dados, removendo interações sem conteúdo relevante. Os resultados foram, então, plotados em gráficos, com o objetivo de identificar variáveis de interesse para os testes estatísticos, o que resultou na escolha dos *tokens* dos *prompts*, *tokens* das respostas, nível de expertise e curtidas.

Além disso, considerando o baixíssimo número de "descurtidas", decidiu-se que as interações seriam classificadas, em relação à avaliação dos usuários, da seguinte forma:

- *Curtidas*: aquelas interações que foram explicitamente curtidas (*liked_interaction* = *true*);
- *Não Curtidas*: aquelas interações que foram explicitamente reprovadas (*disliked_interaction* = *true*) ou que não foram nem curtidas nem reprovadas (*disliked_interaction* = *false* e *liked_interaction* = *false*).

Com essas classificações, as colunas foram normalizadas para que seus valores se tornassem compatíveis com as bibliotecas utilizadas.

Antes dos testes estatísticos, foram feitas análises da proporção entre os *prompts*, as respostas, seus respectivos atributos e as demais variáveis, com o intuito de obter outros *insights*, além dos resultados fornecidos pelos testes.

Em seguida, foi realizado o teste de normalidade em ambas as quantidades de *tokens* para verificar se seguiam uma distribuição normal. Com base nos resultados, optou-se por realizar a Correlação de Spearman, o teste U de Mann-Whitney e, por fim, duas Regressões Logísticas, cada uma com *cross-validation* de 10 *folds*. Quando pertinente, foram avaliadas as razões de chances, por meio das quais se estimou o impacto das variáveis dependentes nas variáveis independentes analisadas.

4.3.2 Processamento de Linguagem Natural

Neste processo, foram pré-processados apenas os conteúdos dos *prompts*, visto que eles é que são diretamente produzidos pelos alunos. Após a remoção das *stopwords* em português, contou-se o número de palavras, ordenando-as por esse valor e, a partir daí, removeram-se também as palavras mais frequentes nos padrões de *prompt* ministrados ou cujas frequências destoavam muito das anteriores.

Os *prompts* foram analisados com a classificação feita para: interações curtidas, interações não curtidas e o nível de expertise dos alunos. Em cada caso, foram avaliadas as palavras mais frequentes por meio das *wordclouds*, as sequências de palavras mais comuns por meio dos trigramas — para descobrir do que trata os *prompts* — e as palavras que

mais aparecem juntas em um mesmo *corpus*, por meio das coocorrências, para tentar identificar padrões.

4.3.3 Análise das Respostas aos Formulários

Por fim, foram analisadas as respostas dos alunos aos formulários. Apesar dos esforços realizados no sentido de engajar os alunos no experimento, o número de respondentes foi pequeno, o que levou à necessidade de realizar um pré-processamento para selecionar apenas as respostas daqueles que produziram algum *prompt*.

As análises foram feitas separadamente para as questões objetivas e subjetivas, referentes ao Levantamento de Requisitos e aos Testes, respectivamente, relacionando as respostas com os resultados obtidos nas análises anteriores, quando possível.

4.4 Conclusão

A metodologia do estudo, descrita nesta seção, envolveu os seguintes passos: desenvolver uma ferramenta que utilize o modelo GPT-4o, treinar os alunos para utilizá-la em Levantamento de Requisitos e Testes, coletar os dados de uso, coletar a experiência dos alunos por meio de formulários e realizar a análise desses dados.

Com isso, foram obtidos os resultados utilizados para responder às questões de pesquisa elencadas no Capítulo 1.

5 Resultados

Nesta seção, serão apresentados os resultados da aplicação da metodologia descrita no capítulo anterior.

5.1 Apresentação dos Dados

Como descrito anteriormente, o experimento foi realizado no contexto da disciplina de Engenharia de Software II do curso de Ciência da Computação da Universidade Federal do Piauí. Ela contou com 24 alunos, dos quais 16 participaram do experimento, utilizando ativamente a ferramenta com os *prompts*. Após a remoção das interações sem conteúdo relevante, restaram, ao todo, 126 *prompts*.

A Tabela 6 mostra a distribuição desses *prompts*, de acordo com o nível de expertise dos alunos em desenvolvimento de *software*, por nível de expertise, o percentual de *prompts* de cada um e a quantidade de *prompts* por aluno.

Tabela 6 – Tabela de Alunos e Prompts por Nível de Expertise

Nível de Expertise	Quantidade de Alunos	Quantidade de Prompts	Percentual de Prompts	Prompts por Aluno
Apenas Estudante	12	105	83.33%	8,75
Estagiário	4	21	16.67%	5,25

É possível observar que, apesar das duas distribuições estarem semelhantes, a proporção de *prompts* dos usuários cadastrados como "Estagiário"em relação ao total é ligeiramente menor do que a dos estudantes sem experiência prévia (5,25 no primeiro caso contra 8,75 no segundo), o que pode indicar que o primeiro grupo precisou de menos *prompts* para realizar suas atividades.

Além disso, os alunos utilizaram a ferramenta muito mais na tarefa de Levantamento de Requisitos do que na tarefa de Testes. Isso, por si só, não necessariamente indica que o GPT seja mais adequado para a primeira tarefa, uma vez que, na segunda etapa do experimento, o nível de engajamento foi menor.

Ao analisar a quantidade de *tokens* dos *prompts* e das respostas por tema, obtemos o gráfico da Tabela 7.

Tabela 7 – Total de Tokens (Prompts e Respostas) por Tema

Tema	Total de Tokens dos Prompts	Total de Tokens das Respostas
Levantamento de Requisitos	11136	58565
Testes	4962	10235

Como esperado, a quantidade total de *tokens* no Levantamento de Requisitos foi maior do que no de Testes, devido ao menor número de *prompts*. Para contornar essa

discrepância e possibilitar uma análise comparativa, foram calculadas três razões, chamadas de Razão 1, Razão 2 e Razão 3, definidas da seguinte forma:

$$\text{Razão 1} = \frac{\text{Total de tokens dos prompts do tema}}{\text{Total de prompts do tema}} \tag{5.1}$$

$$\text{Razão 2} = \frac{\text{Total de tokens das respostas do tema}}{\text{Total de prompts do tema}} \tag{5.2}$$

$$\text{Razão 3} = \frac{\text{Total de tokens das respostas do tema}}{\text{Total de tokens dos prompts do tema}} \tag{5.3}$$

Os resultados dessas três razões para os dados coletados encontram-se na Tabela 8.

Tabela 8 – Comparação das razões entre os dois temas

Tema	**Razão 1**	**Razão 2**	**Razão 3**
Levantamento de Requisitos	105,06	552,50	5,26
Testes	248,10	511,75	2,06

Comparando os valores, é possível perceber que os *prompts* para a tarefa de Levantamento de Requisitos, no geral, foram menores do que os da tarefa de Testes. No entanto, se compararmos com o tamanho das respostas fornecidas pelo GPT — que tiveram o valor da Razão 2 praticamente igual — podemos perceber que foi necessário menos texto nos *prompts* de Levantamento de Requisitos para obter respostas tão grandes quanto as de Testes, o que pode indicar que, nesse aspecto, o GPT precisou de mais contexto para gerar respostas do mesmo tamanho para a segunda tarefa, como observado a partir do valor da Razão 3 (2,06 desta *versus* 5,26 da primeira).

Além de interagir com a ferramenta, o usuário também poderia, opcionalmente, curtir ou não curtir expressamente a interação. A comparação entre o total de *prompts* e cada uma dessas interações está na Figura 5.

No gráfico, observa-se que, do total de *prompts*, 32,54% receberam avaliação positiva, 8,73% receberam avaliação negativa e 58,73% não receberam nenhuma. Por ser um campo opcional, é esperado que a quantidade de avaliações seja menor do que o total de interações. Devido à baixa quantidade de avaliações negativas, os dados foram separados nas classes "curtida"e "não curtida"(a soma das "descurtidas"com as que não foram nem uma, nem outra). Dessa forma, é possível realizar testes e análises estatísticas com relevância.

5.1.1 Testes Estatísticos

5.1.1.1 Teste de Normalidade

Para investigar se as quantidades de *tokens*, tanto dos *prompts* quanto das respostas, seguem o padrão normal, foi realizado o teste *Shapiro-Wilk*. Tanto para os *Prompts* quanto

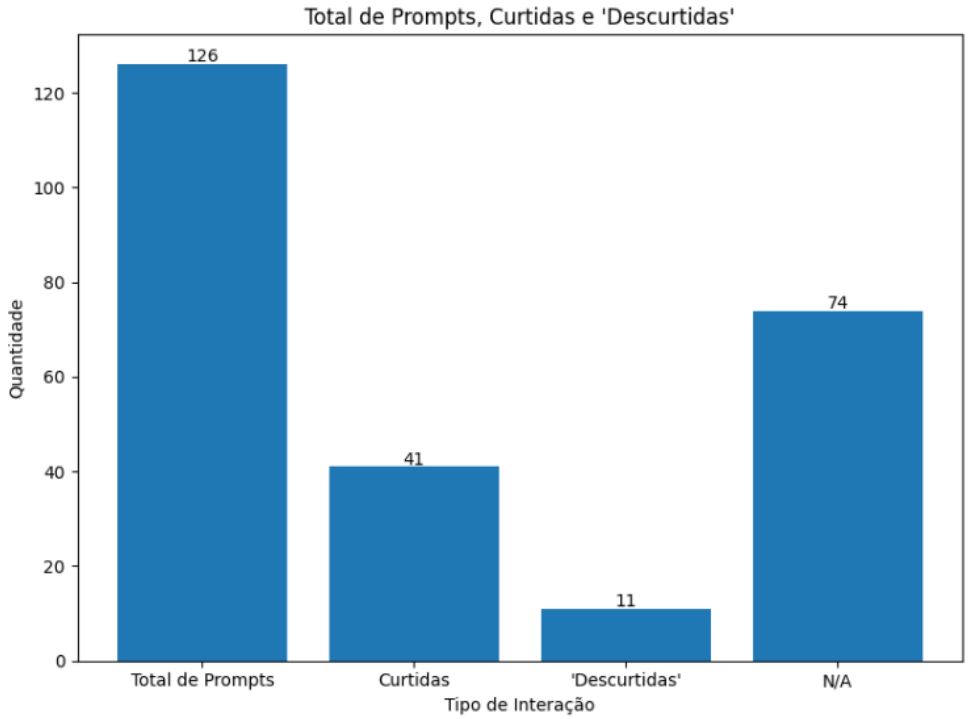

Figura 5 – Total de *prompts* e de avaliações das respectivas respostas.

para as *Respostas*, as hipóteses nulas H_0 foram rejeitadas, devido ao p-valor abaixo de 0,05. Logo, ambas não seguem uma distribuição normal.

5.1.1.2 Correlação de *Spearman*

Foram avaliadas as relações entre os *tokens* dos *prompts* e das respostas, entre o nível de expertise e os *tokens* dos *prompts*, além das três variáveis anteriores com a curtida da interação.

Aqui, vale destacar as seguintes correlações: entre os *tokens* dos *prompts* e das respostas, não tiveram uma que fosse estatisticamente significativa, o que pode ter influência da limitação nestes valores durante o desenvolvimento da ferramenta ($\rho = 0{,}1189$ e p-valor $= 0{,}1847$); e da interação ser curtida com ambas as quantidades de *tokens* dos *prompts* e das respostas, que tiveram correlações positivas e estatisticamente significativas ($\rho = 0{,}2273$ e p-valor $= 0{,}0105$; $\rho = 0{,}18$ e p-valor $= 0{,}0437$, respectivamente).

5.1.1.3 Teste U de *Mann-Whitney*

Investigou-se a possibilidade de haver uma diferença entre uma interação ser curtida ou não, avaliando o número de *tokens* dos *prompts* e o das respostas. De acordo com os testes, ambos são estatisticamente significativos para uma interação ser curtida ou não pelo usuário, sendo essa relação mais forte para os *prompts*, devido ao p-valor menor (p-valor $= 0{,}0111$ e $0{,}0444$, respectivamente).

5.1.1.4 Regressão Logística

Para avaliar os efeitos diretos dos tamanhos das respostas e dos *prompts*, além do nível de expertise, na probabilidade de uma interação ser curtida, bem como se o nível de expertise influencia no tamanho dos *prompts*, foram feitas duas Regressões Logísticas.

Para o segundo modelo, especificamente, é importante notar que, como o número de *tokens* dos *prompts* é uma variável contínua, adotou-se a seguinte heurística: calculou-se a mediana desses dados e, então, criou-se uma nova variável dicotômica, cujos valores assumem as seguintes regras:

- Se o número de *tokens* for maior do que a mediana, o valor dessa variável será "verdadeiro";
- Se esse número for menor ou igual à mediana, o valor será "falso".

Satisfeitas as pendências, as regressões foram feitas e ajustadas com validação cruzada de 10 *folds*. Não foi encontrada uma regressão com dados estatisticamente significativos para o segundo modelo. No caso do primeiro, a Tabela 9 mostra a variação percentual da probabilidade de uma interação ser curtida, por variável.

Tabela 9 – Coeficientes, Razões de Chances para 100 *Tokens* e Variação Percentual

Variável	**Coeficiente**	**Razão de Chances**	**Variação Percentual**
Constante	-1,856175	0,1563	-84,37%
Prompt (*Tokens*)[1]	0,000221	1,0224	+2,24%
Resposta (*Tokens*)[1]	0,002099	1,2335	+23,35%
Expertise	-0,593467	0,5524	-44,76%

Apenas os *tokens* das respostas, aqui, apresentaram impacto estatisticamente significativo (p-valor = 0,03164). Os números da **Tabela 9** mostram que, para cada 100 *tokens* a mais na resposta, há um aumento de 23,35% na probabilidade de uma interação ser curtida. Esses resultados corroboram com a hipótese de que, no contexto do estudo, as respostas tiveram um impacto muito mais expressivo na decisão dos usuários do que os *prompts* iniciais.

5.1.2 Análise do Conteúdo dos Textos

A seguir, serão apresentados os resultados da análise dos *prompts* produzidos pelos alunos, haja vista que, mesmo que os *tokens* da resposta tenham tido um impacto maior, os *prompts* é que são diretamente manipulados pelos usuários.

[1] As razões de chances para *Prompt* e Resposta foram calculadas para um aumento de 100 *tokens*, ou cerca de 75 palavras.

5.1.2.1 Palavras Mais Frequentes

A primeira investigação realizada foi a respeito das palavras mais frequentes em todos os *prompts*, com o resultado ilustrado na Tabela 10. Ao analisá-la, é possível perceber que muitas das palavras são comuns em descrições de alto nível sobre Levantamento de Requisitos e Testes, não agregando muito valor por si mesmas, uma vez que também estão presentes nos padrões de *prompt* sugeridos aos alunos. Além disso, algumas palavras apareciam muito mais do que as outras, interferindo na visualização dos dados. Portanto, elas foram removidas da análise, a fim de obter uma visão mais precisa do real conteúdo produzido pelos alunos. A Tabela 12 apresenta as palavras removidas para melhorar a qualidade da avaliação, e a Tabela 11 mostra o resultado do filtro.

Tabela 10 – As 20 palavras mais frequentes nos *prompts*.

Palavra	**Frequência**
usuário	169
sistema	92
livros	82
usuários	82
uso	67
criar	48
const	45
import	44
requisitos	42
pode	38
casos	37
descrição	34
string	33
perfil	32
permitir	31
medicamentos	31
paciente	31
caso	31
posts	28
software	28

Tabela 11 – As 20 palavras mais frequentes, após filtros.

Palavra	**Frequência**
criar	48
const	45
import	44
descrição	34
string	33
perfil	32
permitir	31
medicamentos	31
paciente	31
posts	28
software	28
post	27
notificações	27
atores	26
base	24
recomendações	24
comunidade	24
remedius	24
text	24

5.1.2.2 Nuvem de Palavras

Para uma melhor visualização das frequências das palavras, foram geradas *word-clouds*, nas quais as palavras possuem tamanhos proporcionais ao número de vezes em que aparecem nos *prompts*. Serão analisados os gráficos de Curtidas *versus* Não Curtidas e de estudantes sem experiência profissional *versus* aqueles que têm experiência como estagiários.

Tabela 12 – Palavras removidas dos *prompts* para análise

Palavra	**Frequência**
usuário	169
sistema	92
usuários	82
livros	82
uso	67
requisitos	42
pode	38
casos	37
caso	31
deve	19
teste	11
testes	9
requisito	7

Os gráficos para os *prompts* pertencentes às interações que foram curtidas estão ilustrados na Figura 6 e, para as não curtidas, na Figura 7, ambas após a remoção das que constam na Tabela 12.

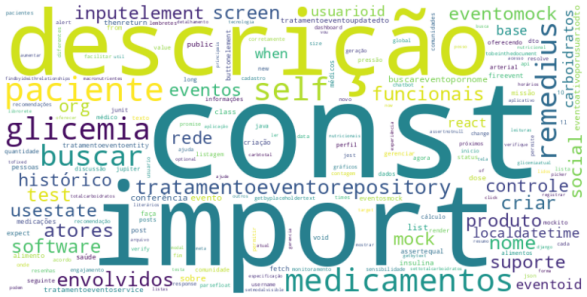

Figura 6 – Nuvem de palavras dos *prompts* para interações curtidas.

Figura 7 – Nuvem de palavras dos *prompts* para interações não curtidas.

As Figuras 8 e 9 apresentam as *wordclouds* dos estudantes sem experiência profissional e dos estagiários, respectivamente, sem as palavras da Tabela 12.

Figura 8 – Nuvem de palavras dos *prompts* de alunos apenas estudantes.

Figura 9 – Nuvem de palavras dos *prompts* de alunos estagiários.

Analisando os *wordclouds*, pode-se perceber que, entre os *prompts* das interações

curtidas e não curtidas, havia uma maior presença de código naqueles pertencentes ao primeiro grupo, apesar da diferença não ser tão expressiva. Ao fazer a mesma avaliação para os níveis de experiência dos estudantes participantes do estudo, é possível detectar uma clara diferença entre os grupos, com destaque para a *wordcloud* dos estagiários bastante equilibrada e praticamente tomada por trechos de código.

5.1.2.3 Análise de Trigramas

Para avaliar do que se tratam os *prompts* dos alunos e ter uma noção do seu conteúdo, foram gerados os trigramas e suas respectivas frequências de aparição no texto. Estas trincas foram geradas para aqueles *prompts* que foram curtidos ou não, e também para aqueles alunos que são apenas estudantes e para aqueles que são, também, estagiários, sem as palavras da Tabela 12.

Separando-se os *prompts* cujas interações foram avaliadas positivamente ou não, obtemos os resultados nas Figuras 10 e 11, respectivamente.

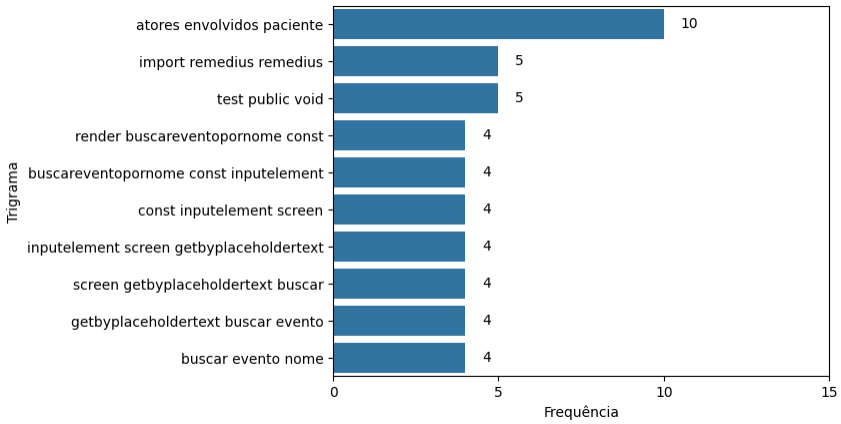

Figura 10 – Trigramas dos *prompts* curtidos.

Figura 11 – Trigramas dos *prompts* não curtidos.

Para as interações feitas por aqueles alunos que são apenas estudantes e pelos estagiários, foram obtidos os trigramas ilustrados nas Figuras 12 e 13, respectivamente.

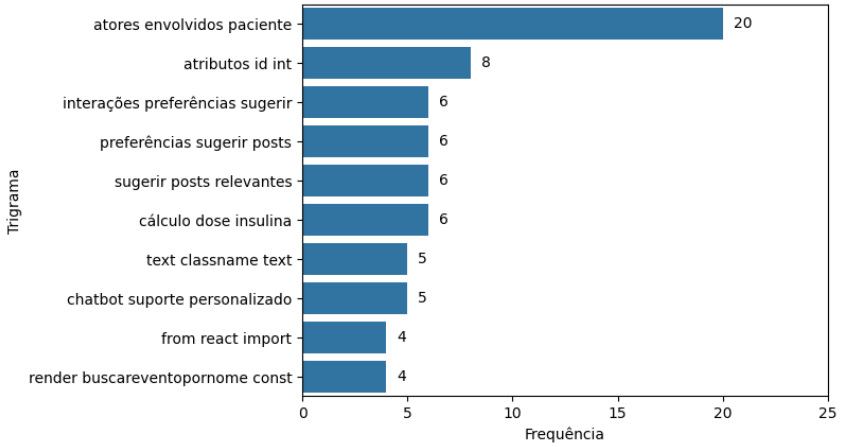

Figura 12 – Trigramas dos alunos que são apenas estudantes.

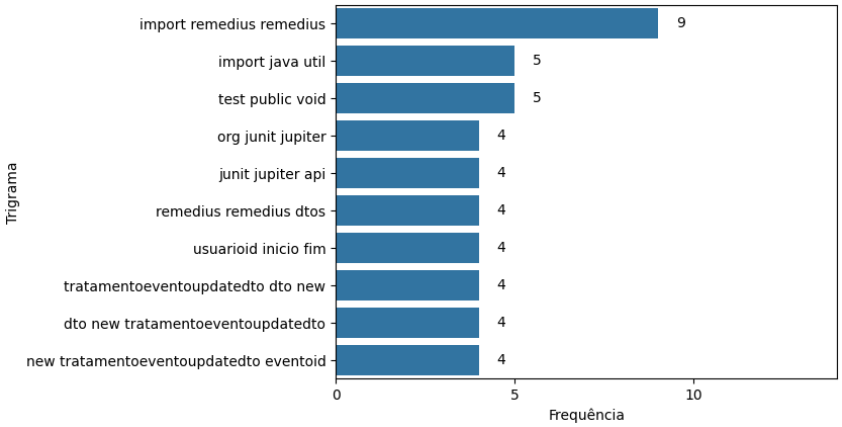

Figura 13 – Trigramas dos alunos que são estagiários.

As interações curtidas apresentaram mais trechos de código nos trigramas, quando comparadas com aquelas não curtidas (oito na primeira *versus* três na segunda), o que indicaria que interações contendo código tinham mais chances de serem curtidas pelo usuário.

Outra diferença marcante aparece ao comparar os trigramas dos alunos que são apenas estudantes com os daqueles que também estagiavam. Quando analisados os *prompts* dos estudantes, estes, no geral, continham palavras de alto nível, relacionadas às regras de negócio dos seus respectivos *softwares* e, no caso dos estagiários, apresentavam quase que em sua totalidade palavras relacionadas a código em todos os trigramas mais frequentes.

5.1.2.4 Análise de Coocorrência

Por fim, foi realizada a análise de coocorrência, separando os *prompts* entre aqueles que foram curtidos ou não, e também entre os que foram feitos por alunos que são apenas

estudantes e por aqueles que são, também, estagiários, sem as palavras da Tabela 12. Como resultado, obtiveram-se os gráficos com as 20 coocorrências mais frequentes, os quais serão apresentados a seguir.

Filtrando os *prompts* cujas interações foram avaliadas positivamente ou não, obtiveram-se as coocorrências ilustradas nas Figuras 14 e , respectivamente.

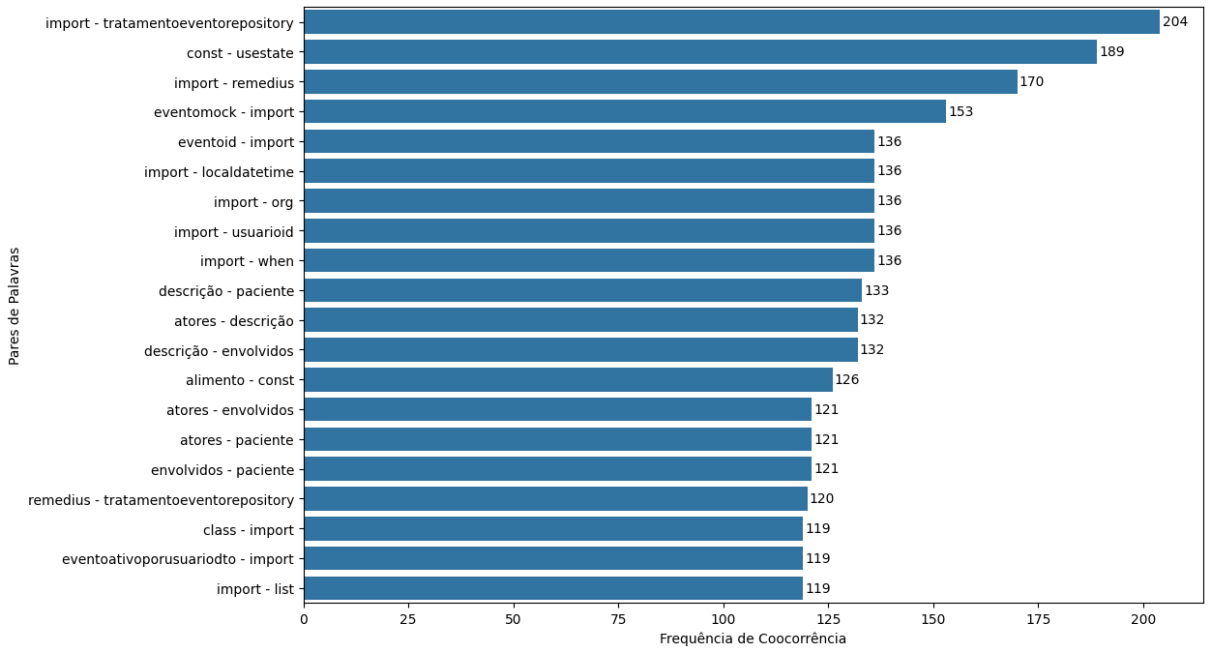

Figura 14 – Coocorrências de palavras nos *prompts* curtidos.

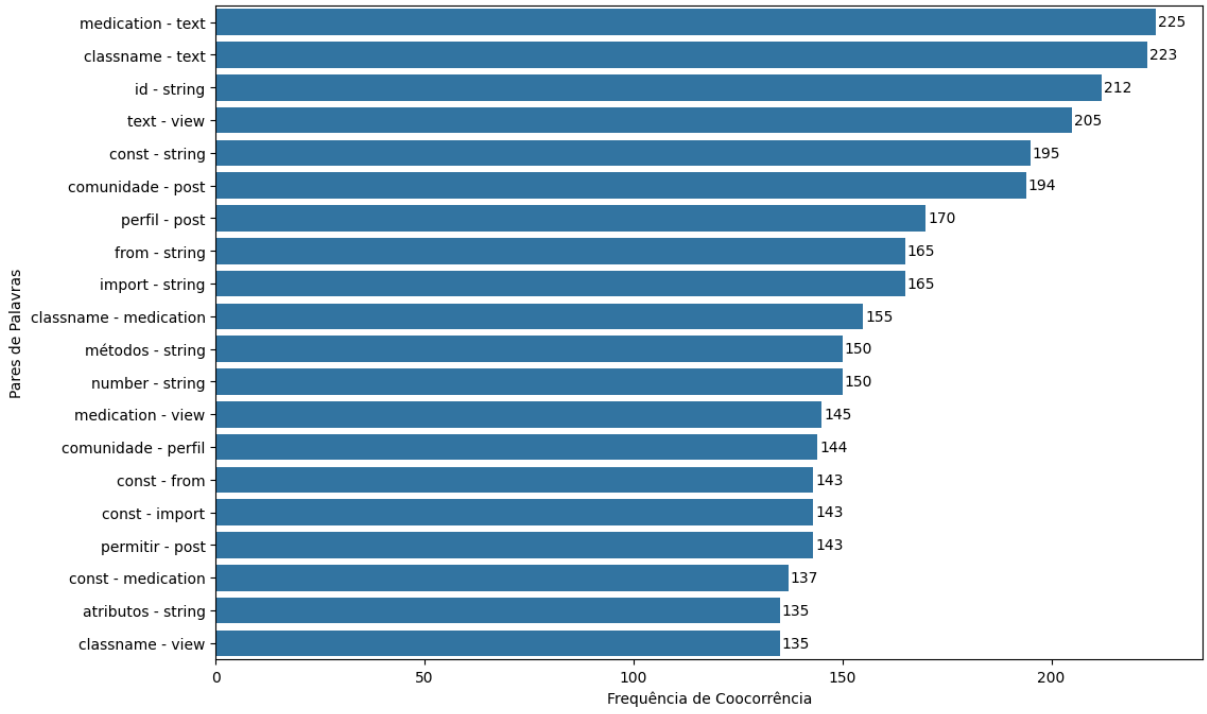

Figura 15 – Coocorrências de palavras nos *prompts* não curtidos.

Os *prompts* dos alunos que não possuem experiência profissional e os dos que estagiam resultaram nas coocorrências ilustradas nas Figuras 16 e 17, respectivamente.

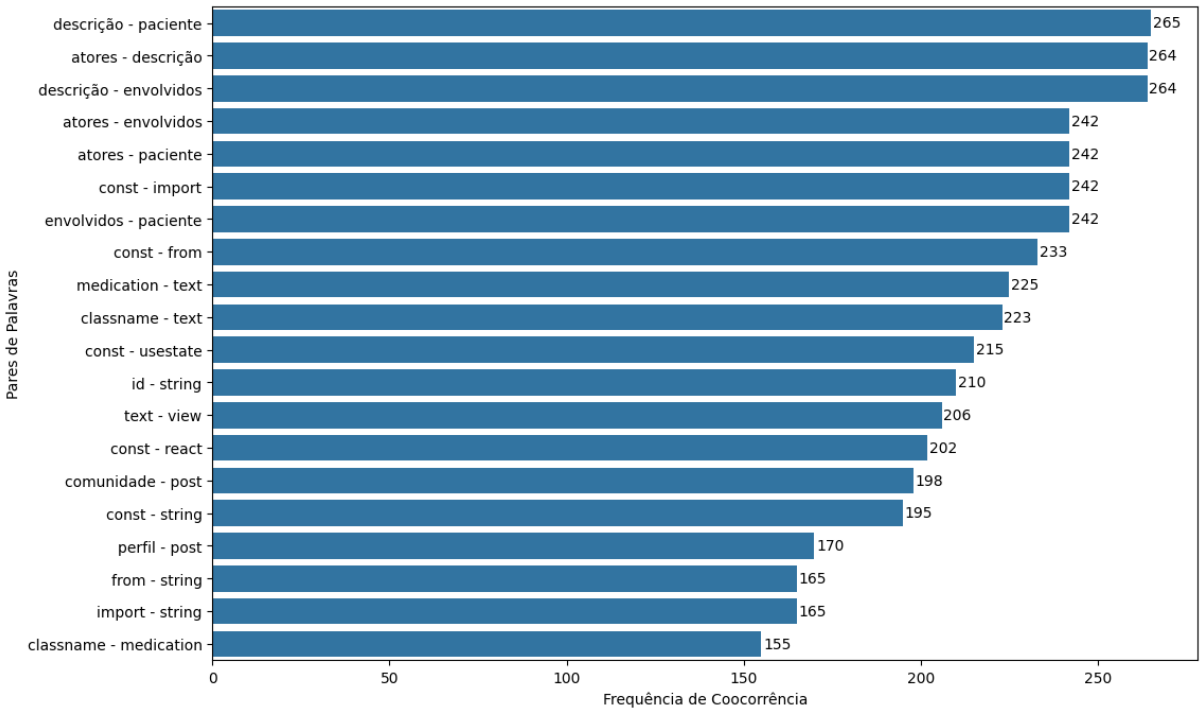

Figura 16 – Coocorrências de palavras nos *prompts* de estudantes.

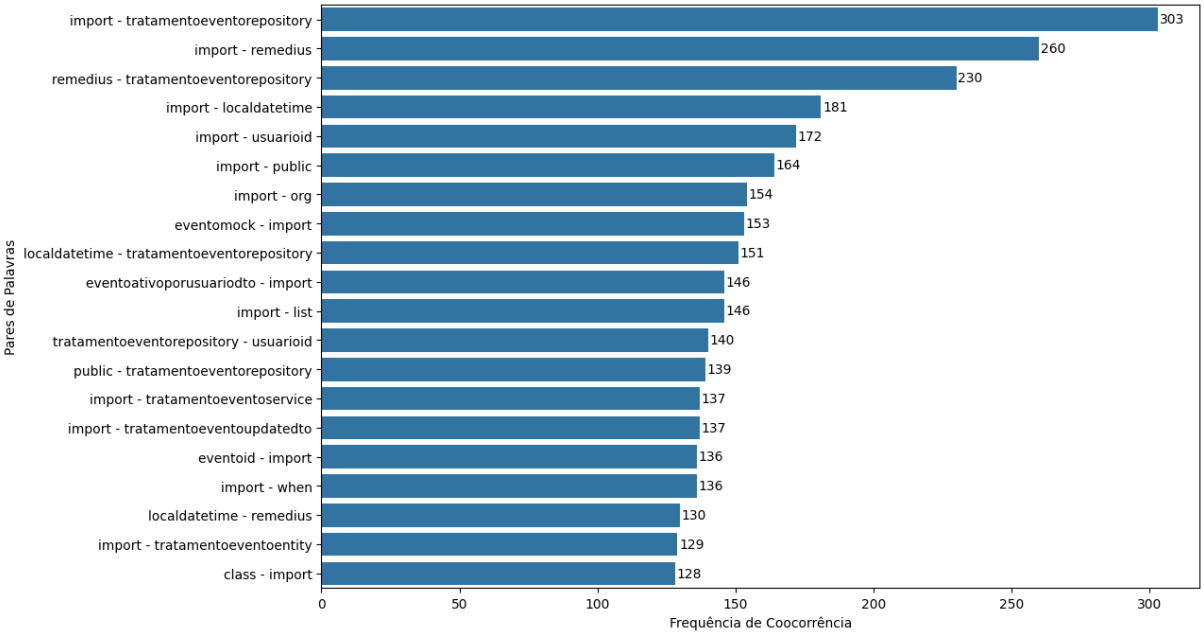

Figura 17 – Coocorrências de palavras nos *prompts* de estagiários.

Diferentemente dos trigramas, as interações curtidas apresentaram menos coocorrências com código, se comparadas com aquelas não curtidas (14 grupos com 1999 registros no primeiro grupo *versus* 16 grupos com 2683 registros na segundo), o que pode se explicar pela menor quantidade de *prompts* pertencentes ao grupo de curtidas.

Porém, em concordância com os trigramas, há uma diferença marcante entre as coocorrências dos *prompts* daqueles alunos que são apenas estudantes e os daqueles que também estagiavam. As 20 mais frequentes dos estagiários contém, pelo menos, uma palavra associada a código, diferentemente dos alunos sem experiência profissional, reforçando o indício de que os estagiários tinham uma maior probabilidade de utilizar código em seus *prompts*.

5.1.2.5 Formulários

Como mencionado anteriormente, dos 24 alunos da disciplina, 16 utilizaram a ferramenta. Destes, apenas seis responderam a pelo menos um dos formulários: três responderam apenas ao formulário de Levantamento de Requisitos, dois apenas ao de Testes e um respondeu a ambos. Curiosamente, duas pessoas que não utilizaram a ferramenta também responderam ao primeiro formulário, totalizando oito pessoas que responderam a pelo menos um dos formulários. Todos os estudantes usuários que responderam ao questionário, na época do cadastro de suas contas, eram apenas estudantes.

A Figura 18 mostra a distribuição dos alunos entre o total de matriculados, o total daqueles que utilizaram a ferramenta, o total de respondentes e quantos desses não participaram efetivamente do experimento. A Figura 19 mostra a distribuição dos respondentes que utilizaram a ferramenta.

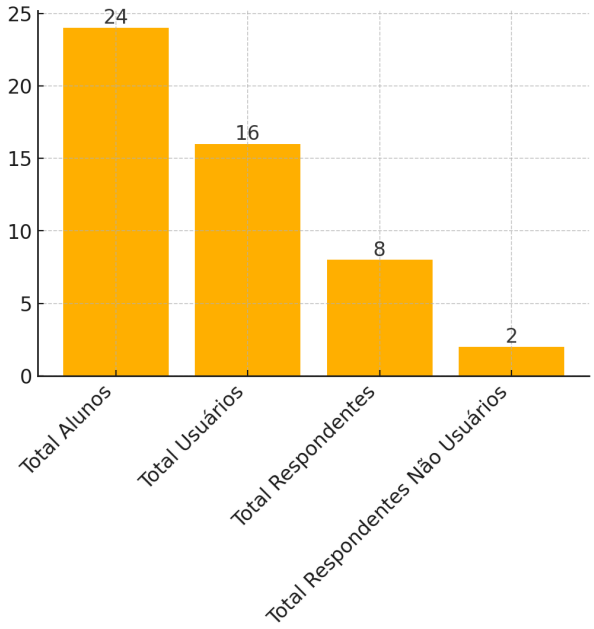

Figura 18 – Distribuição dos alunos matriculados na turma, usuários da ferramenta e respondentes do questionário.

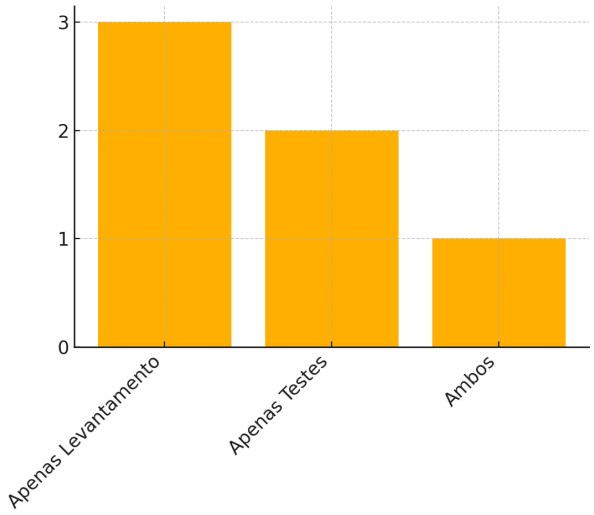

Figura 19 – Distribuição dos alunos que responderam um ou ambos os questionários.

A seguir, serão analisadas as respostas para cada um dos formulários. No caso específico do Levantamento de Requisitos, somente foram analisadas as respostas de alunos que também utilizaram a ferramenta. Mesmo com esse cuidado, é importante ressaltar que esses dados não terão tanta representatividade como teriam caso todos os usuários tivessem respondido. As perguntas e suas respectivas numerações estão no Apêndice E.

A Tabela 13 apresenta cada um dos alunos e quais questionários eles responderam. A partir de agora, os alunos serão referidos pelos seus respectivos pseudônimos:

Tabela 13 – Relação dos alunos que responderam os formulários de Levantamento de Requisitos (1) e Testes (2)

Matrícula	**Pseudônimo**	**Resposta 1?**	**Resposta 2?**
***060	Aluno 1	Sim	Sim
***406	Aluno 2	Sim	Não
***215	Aluno 3	Sim	Não
***091	Aluno 4	Sim	Não
***479	Aluno 5	Não	Sim
***239	Aluno 6	Não	Sim

Todas as respostas para as perguntas objetivas do questionário do Apêndice E estão na Tabela 14.

Tabela 14 – Respostas dos alunos às perguntas objetivas do formulário de Levantamento de Requisitos

	Q3	**Q4**	**Q5**	**Q6**	**Q7**	**Q8**	**Q9**	**Q13**	**Q14**
Aluno 1	0-2 horas	5 - Ótimo	3 - Regular	3 - Regular	2 - Ruim	3 - Regular	Sim	Não	Não utilizei outra estratégia
Aluno 2	0-2 horas	4 - Bom	3 - Regular	4 - Bom	4 - Bom	4 - Bom	Sim	Não	Não utilizei outra estratégia
Aluno 3	0-2 horas	4 - Bom	3 - Regular	5 - Ótimo	5 - Ótimo	3 - Regular	Não	Não	Não utilizei outra estratégia
Aluno 4	0-2 horas	5 - Ótimo	5 - Ótimo	5 - Ótimo	2 - Ruim	5 - Ótimo	Sim	Não	Não utilizei outra estratégia

De acordo com a tabela, é possível observar que todos os alunos dedicaram um curto período de tempo à tarefa, com nenhum deles relatando ter gasto mais de 2 horas (Q3).

Em relação ao impacto da ferramenta no processo de aprendizado (Q4), as percepções foram predominantemente positivas. Dois alunos classificaram o impacto como "Bom" e os outros dois como "Ótimo". Entretanto, a adequação da ferramenta para desempenhar a tarefa propriamente dita (Q5) teve uma avaliação inferior, sendo considerada "Regular" por três alunos e "Ótimo" por outro.

Quanto à adequação da ferramenta para aprender os conceitos relacionados à tarefa (Q6), a percepção geral dos alunos foi de neutra a positiva, com avaliações variando de "Regular" a "Ótimo". No entanto, o esforço adicional necessário para adaptar o conteúdo gerado pela ferramenta (Q7) variou bastante, indo de "Ruim" a "Ótimo", indicando que a experiência não foi uniforme entre os participantes.

O nível geral de satisfação com a ferramenta (Q8) apresentou resultados neutros, mas tendendo ao positivo. Esse padrão reflete-se nas respostas sobre o uso futuro da ferramenta (Q9): apenas um dos alunos declarou que não a utilizaria novamente para essa tarefa específica. Além disso, nenhum dos alunos utilizou outro padrão de *prompt* que não fosse o apresentado em sala de aula (Q13 e Q14).

Questionado sobre a dificuldade com o uso da ferramenta (Q10), o Aluno 1 destacou a dificuldade de aprofundar os requisitos para além do superficial, enquanto os Alunos 2 e 3 relataram problemas com relação à inserção do contexto. O Aluno 1 justificou sua dificuldade pelo fato de a ferramenta não ser muito boa para a inserção de textos grandes, enquanto o Aluno 2 atribuiu a dificuldade à sua própria aplicação. Por fim, o Aluno 4 apontou como principal obstáculo a dificuldade de imaginar casos de uso bem específicos.

Como principal aspecto positivo (Q11), o Aluno 1 destacou a ajuda que a ferramenta deu ao sugerir ideias que ele não tivera antes, além do ganho de tempo na realização das suas atividades, o que também foi destacado pelo Aluno 4. O Aluno 3 mencionou a ausência da necessidade de fornecer regras extras no *prompt* relacionadas à Engenharia de Software, o que dialoga com a avaliação do Aluno 2, que destacou a assertividade da ferramenta, bem como a completude da sua resposta para o Levantamento de Requisitos.

Sobre os padrões de *prompts* sugeridos aos alunos em sala de aula (Q12), todos os alunos os avaliaram positivamente. O Aluno 1 destacou que levará esse padrão para além da pesquisa, enquanto o Aluno 2 considerou o padrão de interação reversa — em que a ferramenta faz perguntas ao aluno para que ele responda e, com base nisso, são feitos os Requisitos — como bastante útil durante o trabalho. O Aluno 3 mencionou que, apesar dos padrões terem sido úteis, ele os utilizou pouco. Por fim, o Aluno 4 comentou que os padrões agilizaram ainda mais seu processo.

Nenhum dos respondentes fez um relato mais detalhado sobre a sua experiência com o uso do GPT para a tarefa de Levantamento de Requisitos (Q15).

Para a tarefa de Testes, três alunos responderam ao todo, sendo que apenas um também respondeu ao questionário anterior. A relação das perguntas objetivas com suas respectivas respostas está na Tabela 15:

Tabela 15 – Respostas dos alunos às perguntas do formulário de Testes

	Q3	**Q4**	**Q5**	**Q6**	**Q7**	**Q8**	**Q9**	**Q13**	**Q14**
Aluno 1	0-2 horas	3 - Regular	3 - Regular	4 - Bom	3 - Regular	4 - Bom	Sim	Sim	Sim
Aluno 5	0-2 horas	4 - Bom	5 - Ótimo	4 - Bom	3 - Regular	4 - Bom	Sim	Sim	Não
Aluno 6	0-2 horas	5 - Ótimo	5 - Ótimo	5 - Ótimo	4 - Bom	5 - Ótimo	Sim	Não	Não utilizei outra estratégia

Todos os alunos relataram ter gasto até 2 horas na realização da tarefa (Q3). No que diz respeito ao impacto da ferramenta na aprendizagem (Q4), as percepções foram predominantemente positivas, variando de "Regular" a "Ótimo". Um dos alunos avaliou o impacto como "Regular", outro como "Bom" e o último como "Ótimo". Quanto à adequação da ferramenta para a realização da tarefa propriamente dita (Q5), dois dos três alunos avaliaram a ferramenta como "Ótimo", enquanto o terceiro avaliou como "Regular". Portanto, a percepção geral é positiva, embora haja alguma variação.

Quando questionados sobre a adequação da ferramenta para aprender conceitos relacionados à tarefa de Testes (Q6), as avaliações foram positivas, variando de "Bom" a "Ótimo". O esforço adicional necessário para adaptar o conteúdo gerado pela ferramenta para essa tarefa (Q7) foi considerado neutro a bom. Dois alunos avaliaram esse esforço como "Regular" e um como "Bom", sugerindo que, embora não tenha sido trivial, também não foi excessivamente trabalhoso para a maioria.

O nível geral de satisfação com a ferramenta (Q8) foi considerado positivo. Todos os respondentes avaliaram a satisfação com a ferramenta como "Bom" ou "Ótimo". Em relação ao uso futuro da ferramenta (Q9), todos os alunos indicaram que a utilizariam novamente para a tarefa de Testes.

Apenas um dos alunos relatou ter utilizado outra estratégia para elaborar os *prompts* (Q13). Curiosamente, entre aqueles que tentaram outra estratégia (Q14), um declarou ter obtido melhores resultados, enquanto o outro não observou melhora significativa.

Questionado sobre a principal dificuldade com o uso da ferramenta para Testes (Q10), o Aluno 1 citou o ato de pensar em como implementar o TDD e planejar os testes antes de implementar qualquer coisa. Os Alunos 5 e 6 citaram o mesmo problema: aliar o uso da ferramenta à realidade das suas aplicações, especialmente no que diz respeito ao contexto.

Como aspecto positivo (Q11), o Aluno 1 mencionou que a ferramenta trouxe ideias que ele não teria sozinho, enquanto o Aluno 5 destacou a facilidade para gerar os códigos

e adaptá-los às necessidades do projeto. Por fim, o Aluno 6 ressaltou que a ferramenta o direcionou para os pontos certos e otimizou o tempo de desenvolvimento dessa etapa.

Ao opinar sobre os padrões de *prompts* sugeridos em sala de aula para esta atividade (Q12), todos os alunos os avaliaram positivamente, com destaque para o Aluno 1, que mencionou o fato de já ter utilizado alguns deles sem saber, e que os demais seriam adotados por ele em outras áreas; e para o Aluno 5, que comentou como os padrões ajudaram a melhorar sua escrita de *prompts*.

Apenas o Aluno 5 complementou as respostas anteriores descrevendo sua experiência (Q15), dizendo que, apesar da ferramenta ser útil, não foi tão utilizada por ele nesta etapa do desenvolvimento. Ele justificou isso pelo fato de ter dado preferência a outras ferramentas com as quais já tinha mais experiência, como o *GitHub Copilot*.

Esta, especificamente, pode ter sido preferida devido ao seu foco maior em programação e pela praticidade de já ser integrada às principais IDEs do mercado, facilitando o uso e a aquisição de contexto, já que o Copilot possibilidade anexar os arquivos para enviar juntamente com os *prompts*.

5.2 Resposta às Questões de Pesquisa

A seguir, serão respondidas cada uma das três questões de pesquisa levantadas no Capítulo 1.

5.2.1 Qual a percepção dos alunos sobre a adequação da ferramenta para as tarefas de Levantamento de Requisitos e Testes em Engenharia de Software?

A percepção dos alunos variou. No Levantamento de Requisitos, os alunos que responderam ao formulário relataram uma percepção mista, mas geralmente positiva, sobre a ferramenta. Dentre os apontamentos feitos, destacaram que o modelo foi útil para ajudá-los a ter novas ideias e a estruturar os requisitos das suas aplicações, assim como dificuldades ao tentar trabalhar com a inserção de contexto em *prompts* mais longos.

A necessidade de adaptar as respostas geradas pela IA nos seus trabalhos também foi mencionada, indicando que, apesar da utilidade do modelo GPT, foram exigidos ajustes nas respostas para que elas se adequassem melhor à tarefa.

Em contraste, para a tarefa de Testes, os alunos demonstraram uma percepção mais positiva em relação à adequação do GPT. A ferramenta foi vista como mais eficaz, especialmente em termos de geração de código para testes automatizados, o que a levou a ser percebida como mais adequada para o uso da IA, possivelmente pelo caráter mais

objetivo da tarefa de Testes.

Resposta RQ1: A adequação percebida da ferramenta variou dependendo da tarefa, sendo mais bem recebida pelos alunos em Testes. No Levantamento de Requisitos, embora útil, a adaptação do conteúdo gerado e a inserção de contexto de alto nível prejudicaram sua avaliação pelos usuários.

5.2.2 Quais características dos *prompts* dos alunos resultaram em interações satisfatórias com a ferramenta?

A análise das características dos *prompts* indica que o que mais influencia uma interação ser curtida está relacionado à maior frequência de *prompts* com código, o que sugere que os alunos que forneceram contextos mais específicos gostaram mais das respostas geradas. Isso é consistente com as observações de que os *prompts* vagos, especialmente aqueles excessivamente conceituais ou genéricos, tendem a resultar em respostas menos satisfatórias.

Apesar do conteúdo dos *prompts* ter um grande impacto na avaliação, o impacto do seu tamanho na probabilidade de uma interação ser curtida foi baixo, sendo esse um atributo mais relevante no caso das respostas: quanto maiores, mais chances tinham de serem bem avaliadas. Isso sugere que *prompts* mais específicos e respostas mais completas foram os elementos-chave para uma melhor avaliação das interações pelos alunos.

Resposta RQ2: *Prompts* mais específicos, incluindo código e um contexto claro, influenciaram na satisfação dos usuários com a ferramenta.

5.2.3 Como a experiência prévia dos alunos com desenvolvimento de software influencia na maneira como utilizam o modelo?

A experiência prévia dos alunos influenciou significativamente a maneira como utilizaram a ferramenta para suas atividades acadêmicas, como observado na análise das diferenças entre estudantes e estagiários.

O segundo grupo adotou uma abordagem mais objetiva em seus *prompts*, com participantes frequentemente incluindo trechos de código para enriquecer o contexto. Por outro lado, os estudantes sem experiência profissional tendem a produzir *prompts* mais generalistas, com uma maior predominância de termos relacionados ao domínio do problema.

Resposta RQ3: Alunos com experiência no mercado utilizaram a ferramenta de maneira mais técnica, com ampla presença de trechos de código e vocabulário técnico, enquanto os que são apenas estudantes adotaram uma abordagem mais de alto nível, com perguntas mais abrangentes e relacionadas ao domínio do problema no qual estavam trabalhando.

6 Discussão

Esta seção apresenta a discussão dos principais achados obtidos neste estudo, abordando com mais detalhes os resultados obtidos no capítulo anterior.

6.1 Análise geral

Como ilustrado na Tabela 6, a quantidade de estudantes sem experiência profissional era o triplo da daqueles que estagiavam na área (12 no primeiro caso contra 4 no segundo), o que também se refletiu na quantidade de *prompts* gerada por cada um dos grupos.

Se compararmos a quantidade de *prompts* por participantes de cada grupo, pode-se deduzir que os estagiários precisaram de menos interações para desempenhar suas tarefas, com cada um produzindo, em média, 5,25 *prompts*, contra 8,75 daqueles que são apenas estudantes, uma diferença de 60%, convergindo com os achados de (WHITE et al., 2023c; WHITE et al., 2023a), onde padrões bem estabelecidos de *prompts* podem resultar em maior eficiência na interação com ferramentas de IA, especialmente quando aplicados por usuários mais experientes.

Mesmo se houvesse a mesma quantidade de alunos pertencentes aos dois grupos, essa é uma diferença esperada, devido à maior capacidade técnica do segundo. Estes incluíram mais trechos de código nos seus *prompts*, como mostram as nuvens de palavras das Figuras 8 e 9. Ainda, a análise dos trigramas e das coocorrências mais frequentes fortalecem essa observação.

Esses achados corroboram com as observações de (DöDERLEIN et al., 2022) e (SUN et al., 2022) sobre como a clareza dos *prompts* pode otimizar a qualidade das respostas obtidas, especialmente quando utilizados trechos de código.

A possibilidade de coletar o *feedback* dos alunos na forma de curtidas e "não curtidas"de forma opcional também trouxe mais um *insight*: os alunos ativamente se dispuseram a avaliar mais aquelas interações que lhes agradaram do que o contrário. Os testes estatísticos mostraram uma relação positiva entre o tamanho dos *prompts* e das respostas com a probabilidade de uma interação ser avaliada positivamente pelos usuários, por meio dos resultados do teste de correlação de *Spearman*.

No entanto, essa relação foi mais relevante com as respostas do que com as perguntas (visível na razão de chances de 23,35% para cada 100 *tokens* no primeiro caso contra 2,24% no segundo), mostrando que, mais importante do que o tamanho do contexto fornecido, é o seu conteúdo, haja vista que tem relação direta com o que constará na resposta da IA. Este achado está de acordo com (LYU et al., 2024), que enfatiza que, embora a quantidade

de informação possa ser útil, o foco na qualidade dos *prompts* é essencial para a eficácia do uso das ferramentas de IA.

Consoante a essa conclusão, as diferenças entre os trigramas dos *prompts* curtidos e os dos não curtidos mostram que aquelas perguntas feitas pelo usuário com conteúdo de caráter mais técnico, como trechos de código, foram mais curtidas. As interações curtidas apresentavam oito trigramas com código, contra três daquelas que não foram curtidas. Quando somadas as frequências de cada um, a diferença fica mais expressiva: os trigramas com código somaram 34 ocorrências no primeiro caso, contra 17 no segundo. Considerando que a quantidade de curtidas foi cerca de 1/3 do total de *prompts*, esta observação se realça ainda mais.

Depreende-se daí que, mesmo que os *prompts* sejam longos, eles podem gerar respostas insatisfatórias caso sejam vagos ou excessivamente conceituais. Essa observação está alinhada com (WASEEM et al., 2024), que destaca que o uso prático das ferramentas de IA é preferido pelos usuários em relação a interações excessivamente genéricas, ou seja, o nível de detalhamento do *prompt* importa mais do que o seu tamanho propriamente dito.

Comparando os trigramas dos alunos apenas estudantes com os dos estagiários, percebe-se que há uma clara diferença nos conteúdos dos *prompts*. Enquanto o primeiro grupo focava em instruções de alto nível, o segundo utilizava bastante código. Apenas dois dos dez trigramas dos que são apenas estudantes continham código, em contraste com todos os trigramas dos estagiários.

Esse comportamento pode estar relacionado ao que foi observado por (RAHMAN; WATANOBE, 2023) e (HAINDL; WEINBERGER, 2024): estudantes sem experiência utilizam as ferramentas principalmente para adquirir conhecimento básico e aprender conceitos de programação. Já os estagiários, por terem mais experiência, utilizam as ferramentas de IA de forma mais prática e direcionada à geração de código.

A comparação entre as coocorrências das classes "curtidas"*versus* "não curtidas"se contrapôs aos achados anteriores. Nos *prompts* de interações curtidas, 14 das 20 coocorrências mais frequentes contêm, pelo menos, um trecho de código, contra 16 para as interações não curtidas. A soma total das frequências no primeiro caso é 1999, enquanto que no segundo caso foi 2683.

A diferença pode estar relacionada ao fato de que a quantidade de interações curtidas corresponde a apenas cerca de 1/3 do total. Portanto, o que foi encontrado aqui se relaciona com o achado do trabalho de (DöDERLEIN et al., 2022), que destacam que, embora o uso de código aumente a qualidade da interação, ele precisa ser utilizado de forma adequada para ser efetivo.

Analisando o caso do nível de expertise dos estudantes, é possível observar um comportamento semelhante ao dos trigramas. As coocorrências com código foram 9 de 20,

com uma frequência total de 1.951, para os alunos sem experiência, contra todas as 20 para os estagiários e uma frequência total de 3.272.

Esses resultados, somados aos anteriores, mostram que, possivelmente, há uma relação entre a presença de códigos nos *prompts* e uma interação ser curtida. Eles também mostram uma clara diferença entre o conteúdo das perguntas dos estudantes e dos estagiários, sendo que, no último caso, predominou o uso de código, alinhando-se com (LYU et al., 2024) e (PETROVSKA et al., 2024), que apontam que usuários mais experientes tendem a utilizar a IA de forma mais eficiente.

Apesar disso, o nível de expertise não apresentou influência estatisticamente significativa direta sobre a probabilidade de uma interação ser avaliada positivamente, sugerindo que, embora alunos com experiência prévia em desenvolvimento tendam a escrever *prompts* mais objetivos, tais características não garantiram por si só interações significativamente melhores com a ferramenta. A percepção de qualidade parece estar mais diretamente associada às respostas do GPT, independente da expertise com desenvolvimento dos alunos, como também fora relatado por (KOSAR et al., 2024).

Uma diferença relevante observada entre as tarefas foi a maior eficiência ao gerar respostas para Levantamento de Requisitos, ao comparar as razões entre os *tokens* das respostas e os dos *prompts*, presentes na Tabela 8. No entanto, a percepção dos alunos que responderam aos questionários foi de que a ferramenta estava mais adequada à tarefa de Testes.

Este contraste pode estar relacionado com o esforço adicional que os alunos relataram em adaptar as respostas do GPT na primeira tarefa, além dos valores destas razões terem sido influenciados pela baixa adesão na segunda tarefa. (SUN et al., 2022) e (DöDERLEIN et al., 2022) também apontaram que diferentes tarefas de desenvolvimento requerem abordagens diferenciadas de uso das ferramentas de IA, especialmente em atividades que exigem maior contextualização, como a de Levantamento de Requisitos.

Tal divergência entre ambas reforça que a efetividade percebida das ferramentas depende, também, da integração correta com as práticas educacionais, conforme dito por (PETROVSKA et al., 2024) e (JOšT; TANESKI; KARAKATIč, 2024). Em (NGUYEN-DUC et al., 2023), também é apontada a necessidade de que essas ferramentas sejam implementadas de maneira adequada ao currículo educacional para maximizar seus benefícios.

Um aspecto destacado pelos alunos nos questionários foi o valor da ferramenta na economia de tempo e na geração de novas ideias, algo frequentemente reportado por trabalhos anteriores, como os de (WASEEM et al., 2024) e (LYU et al., 2024). Além disso, a aplicação dos padrões de *prompts* apresentados em sala foi percebida de maneira positiva pelos alunos, sugerindo que a padronização de boas práticas no uso das IAs generativas

pode ser uma estratégia eficaz para maximizar seus benefícios, dialogando diretamente com as recomendações de (RAHMAN; WATANOBE, 2023) e (JIN et al., 2024) sobre a alfabetização em IA e adoção de estratégias específicas para o seu uso.

6.2 Implicações Práticas

À partir do que foi encontrado no estudo e do debate feito em torno do que fora encontrado em outros trabalhos relacionados a este, é possível convergir para a ideia de que o uso com *prompts* mais objetivos e, quando aplicável, com mais trechos de código, foi o que apresentou melhores resultados, de acordo com o *feedback* dos alunos. No entanto, não é uma relação tão certeira, como observado nos gráficos de coocorrência e, também, como já encontrado em (DöDERLEIN et al., 2022), sugerindo que outros fatores, como a clareza do conteúdo do *prompt*, também desempenharam um papel importante.

Em termos pedagógicos, essa ideia sugere que, simplesmente incentivar o uso de trechos de código ou a criação de prompts mais longos não garante uma melhor experiência, sendo necessário o ensino de boas práticas de construção de *prompts*, com ênfase na clareza e na objetividade nas instruções passadas às LLMs. Além disso, estratégias que promovam o aprendizado prático, como o uso de exemplos que mostrem como as LLMs podem ser úteis em situações do cotidiano de um engenheiro de software, podem melhorar o processo de aprendizagem especialmente daqueles alunos com menos experiência.

A análise do conteúdo escrito pelos alunos, por ter apresentado diferenças expressivas entre níveis de experiência diferentes, à partir das análises do conteúdo produzido por cada um deles, aponta para uma possibilidade: a avaliação do grau de maturidade técnica dos alunos e do seu progresso na disciplina, de acordo com a forma que eles elaboram seus *prompts*.

Outra abordagem que também pode ser utilizada é a personalização do uso destas ferramentas de acordo com o nível de conhecimento do aluno. Para os iniciantes, o foco poderia ser em como estruturar um *prompt* de maneira clara, além de entender o que esperar das respostas. Já para alunos mais avançados, pode-se focar em interações mais complexas, como a criação de código ou resolução de problemas técnicos específicos. Assim, garante-se a eficácia da ferramenta de IA, independente do grau de conhecimento do aluno.

Ou seja, não só é fundamental o ensino de boas práticas de elaboração de *prompts*, mas também a adequação das demandas de uso das LLMs ao nível de conhecimento dos alunos.

6.3 Ameaças à Validade

Este estudo possui algumas limitações importantes que devem ser destacadas. Primeiramente, a baixa adesão dos alunos aos formulários reduz a capacidade de generalização dos resultados obtidos neles para toda a turma. Isso provavelmente ocorreu devido ao fato de que a adoção ou não da ferramenta não impactava negativamente os alunos, o que levou alguns a preferirem outras ferramentas, como o *Copilot*. Além disso, é importante destacar que os resultados obtidos neste estudo podem diferir para outras populações. Portanto, mais repetições são necessárias para se obter uma análise conclusiva.

Outra ameaça à validade está relacionada às restrições impostas ao modelo GPT nas respostas (limitação de *tokens*), que podem ter influenciado na relação entre o tamanho dos *prompts* e das respostas. É possível que, sem essa restrição, outros padrões de interação pudessem ter surgido, além de, possivelmente, resultar em um perfil de avaliação diferente.

Por fim, como as avaliações foram feitas por meio da opção facultativa de "curtida" nas interações, pode haver viés relacionado ao comportamento dos alunos ao avaliarem suas interações com o GPT. Como destacado por (WASEEM et al., 2024) e (FRANKFORD et al., 2024), métricas de avaliação baseadas em aprovação voluntária tendem a favorecer respostas que aparentam ser mais completas ou diretamente aplicáveis, o que pode mascarar deficiências importantes na interação, como a falta de informações relevantes, ou seja: a resposta pode ter sido percebida como positiva, porém na verdade não foi.

7 Conclusão

Este estudo teve como objetivo explorar a percepção dos alunos sobre o uso de modelos de linguagem generativa (LLMs), como o GPT-4o, para auxiliar em tarefas de Engenharia de Software, atuando nas atividades de Levantamento de Requisitos e Testes. A metodologia adotada foi um estudo exploratório com alunos de uma disciplina de Engenharia de Software, onde foram analisados dados de interações com a ferramenta, avaliações dos alunos e respostas aos questionários.

Os resultados indicam que a experiência prévia dos alunos com desenvolvimento de software influenciou o conteúdo dos seus *prompts*, uma vez que estudantes com experiência como estagiários produziram perguntas mais objetivas, frequentemente incorporando trechos de código. Por outro lado, os alunos sem experiência tendiam a elaborar *prompts* mais gerais, focando mais no domínio do problema do que nos aspectos técnicos. Além disso, verificou-se que o nível de satisfação dos alunos com a ferramenta foi mais elevado quando as respostas fornecidas pelo modelo eram maiores. Tais achados sugerem a ideia de que a qualidade do *prompt* e o tamanho da resposta gerada são fatores determinantes para uma interação ser avaliada positivamente.

Em relação à adequação da ferramenta para as tarefas de Levantamento de Requisitos e Testes, a primeira foi percebida como mais desafiadora, uma vez que os alunos relataram dificuldades ao inserir contexto e ajustar as respostas geradas pela IA. Por outro lado, a tarefa de Testes foi considerada mais adequada para o uso da ferramenta, principalmente devido à natureza mais objetiva da tarefa e à geração de código automatizado, que os alunos acharam mais útil.

Espera-se que este estudo contribua, com seus erros e acertos, para o estado da arte do uso dessas ferramentas de IA no contexto de aprendizado, não só da Engenharia de Software, mas também de outras áreas, juntamente com uma metodologia abrangente de análise dos dados coletados, formas de avaliar a percepção dos usuários e sugestões de trabalhos futuros com base nas lacunas que ainda permanecem.

7.1 Trabalhos Futuros

Os resultados e limitações deste estudo apontam para diversas oportunidades de pesquisa que podem ser exploradas para aprofundar a compreensão sobre o uso de modelos de linguagem generativa (LLMs) no ensino e na prática de Engenharia de Software.

Uma limitação importante foi a quantidade reduzida de participantes e o viés potencial relacionado à adesão facultativa dos alunos. Para melhorar a generalização dos

resultados, estudos futuros devem ser realizados com amostras maiores, pois isso poderia fornecer resultados mais robustos estatisticamente. Além disso, outras análises são possíveis de serem feitas em futuras replicações do presente estudo, como relacionar os *prompts* e os *feedbacks* com as notas obtidas pelos alunos, além de investigar o real tempo de uso deles, para além do que os próprios relataram.

Além disso, é necessário investigar maneiras mais eficazes de integrar essas ferramentas no currículo educacional. Trabalhos futuros devem se concentrar no desenvolvimento e avaliação de estratégias pedagógicas específicas que incentivem o uso eficiente das LLMs nas atividades das disciplinas.

Outra linha promissora de pesquisa seria investigar como os estudantes que utilizaram LLMs durante sua formação acadêmica se adaptam ao uso dessas ferramentas no ambiente profissional. Estudos poderiam analisar se o treinamento em *prompt engineering* adquirido na academia se traduz em ganhos de eficiência no uso dessas ferramentas no mercado de trabalho.

Além disso, seria relevante identificar possíveis diferenças no uso de LLMs entre profissionais iniciantes e experientes, avaliando como o conhecimento adquirido durante o período acadêmico influencia sua utilização prática. Essa abordagem poderia fornecer dados importantes sobre a eficácia do ensino de LLMs como ferramenta educativa e sua aplicabilidade no mercado de trabalho.

Por fim, a replicação do experimento utilizando outros modelos disponíveis no mercado, como *DeepSeek* e *Gemini*, com a posterior comparação dos resultados obtidos no presente estudo, pode contribuir para a generalização dos achados.

Referências

AGUIAR, J. J. B. Inteligência artificial e tecnologias digitais na educação: oportunidades e desafios. *Open Minds International Journal*, v. 4, n. 2, p. 183–188, 2023. 2

AHMED, T.; DEVANBU, P. Few-shot training LLMs for project-specific code-summarization. In: *Proceedings of the 37th IEEE/ACM International Conference on Automated Software Engineering*. New York, NY, USA: Association for Computing Machinery, 2023. (ASE '22). Disponível em: <https://doi.org/10.1145/3551349.3559555>. 12

AJEE, K. L.; VALSAN, A.; SANKARAN, R. Choosing the right statistical test: A guide for data analysis. *Amrita Journal of Medicine*, v. 20, n. 2, p. 86–88, April-June 2024. 7

AMATRIAIN, X. *Prompt Engineering 201: Advanced methods and toolkits*. 2023. <https://amatria.in/blog/prompt201#cot>. Acessado em: 2 de dezembro de 2024. Disponível em: <https://amatria.in/blog/prompt201#cot>. 21

ANJOS, C. I. dos; FRANCISCO, D. J. Educação infantil e tecnologias digitais: reflexões em tempos de pandemia. *Zero-a-seis*, Universidade Federal de Santa Catarina (UFSC), v. 23, n. 2, p. 125–146, 2021. 2

ASARE, O.; NAGAPPAN, M.; ASOKAN, N. *Is GitHub's Copilot as Bad as Humans at Introducing Vulnerabilities in Code?* 2023. 1

CASELI, H. M.; NUNES, M. G. V. (Ed.). *Processamento de Linguagem Natural: Conceitos, Técnicas e Aplicações em Português*. [S.l.]: BPLN, 2023. <https://brasileiraspln.com/livro-pln>. ISBN 978-65-00-80693-9. 10

CHEN, M. et al. *Evaluating Large Language Models Trained on Code*. 2021. 1, 11, 12

DAKHEL, A. M. et al. GitHub Copilot AI pair programmer: Asset or liability? *Journal of Systems and Software*, v. 203, p. 111734, 2023. 12

DENNY, P.; KUMAR, V.; GIACAMAN, N. Conversing with copilot: Exploring prompt engineering for solving cs1 problems using natural language. In: *Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 1*. New York, NY, USA: Association for Computing Machinery, 2023. (SIGCSE 2023), p. 1136–1142. ISBN 9781450394314. Disponível em: <https://doi.org/10.1145/3545945.3569823>. 1

DOHMKE, T.; IANSITI, M.; RICHARDS, G. Sea change in software development: Economic and productivity analysis of the ai-powered developer lifecycle. *arXiv preprint arXiv:2306.15033*, 2023. 12

DöDERLEIN, J.-B. et al. Piloting copilot and codex: Hot temperature, cold prompts, or black magic? 10 2022. Disponível em: <http://arxiv.org/abs/2210.14699>. 13, 43, 44, 45, 46

EZZINI, S. et al. Ai-based question answering assistance for analyzing natural-language requirements. In: *2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE)*. Los Alamitos, CA, USA: IEEE Computer Society, 2023. p. 1277–1289. 12

FERNÁNDEZ, D. M. et al. Naming the pain in requirements engineering: Contemporary problems, causes, and effects in practice. *Empirical software engineering*, Springer, v. 22, p. 2298–2338, 2017. 6

FEUERRIEGEL, S. et al. Generative ai. *Business & Information Systems Engineering*, Springer, v. 66, n. 1, p. 111–126, 2024. 11

FIGUEIRA, C. V. Modelos de regressão logística. 2006. 9

FRANCO, N. M. G.; FARIA, L. I. L. de. Colaboração científica intraorganizacional: análise de redes por coocorrência de palavras-chave. *Em questão*, p. 87–110, 2019. 10

FRANKFORD, E. et al. *AI-Tutoring in Software Engineering Education: Experiences with Large Language Models in Programming Assessments*. 2024. Disponível em: <https://arxiv.org/abs/2404.02548v2>. 14, 15, 16, 47

GITHUB. *Sobre o GitHub Copilot for Individuals*. 2023. Disponível em: <https://docs.github.com/pt/copilot/overview-of-github-copilot/about-github-copilot-for-individuals>. 1

HAINDL, P.; WEINBERGER, G. Students' experiences of using chatgpt in an undergraduate programming course. *IEEE Access*, v. 12, p. 43519–43529, 2024. Disponível em: <https://doi.org/10.1109/ACCESS.2024.3380909>. 44

IMAI, S. Is GitHub Copilot a substitute for human pair-programming? an empirical study. In: *2022 IEEE/ACM 44th International Conference on Software Engineering: Companion Proceedings (ICSE-Companion)*. [S.l.: s.n.], 2022. p. 319–321. ISSN 2574-1926. 12

JIN, H. et al. Teach ai how to code: Using large language models as teachable agents for programming education. In: *Proceedings of the CHI Conference on Human Factors in Computing Systems (CHI '24)*. Honolulu, HI, USA: ACM, New York, NY, USA, 2024. Disponível em: <https://doi.org/10.1145/3613904.3642349>. 15, 46

JOšT, G.; TANESKI, V.; KARAKATIč, S. The impact of large language models on programming education and student learning outcomes. *Applied Sciences*, v. 14, n. 4115, 2024. Disponível em: <https://doi.org/10.3390/app14104115>. 14, 16, 45

KOSAR, T. et al. Computer science education in chatgpt era: Experiences from an experiment in a programming course for novice programmers. *Mathematics*, v. 12, n. 629, 2024. Disponível em: <https://doi.org/10.3390/math12050629>. 14, 15, 45

LEOTTI, V. B.; BIRCK, A. R.; RIBOLDI, J. Comparação dos testes de aderência à normalidade kolmogorov-smirnov, anderson-darling, cramer–von mises e shapiro-wilk por simulação. *Anais do 11º Simpósio de Estatística Aplicada à Experimentação Agronômica*, 2005. 7

LIU, H. et al. *AutoTestGPT: A System for the Automated Generation of Software Test Cases Based on ChatGPT*. 2023. Available at SSRN 4584792. 12

LIU, R. et al. Teaching cs50 with ai: Leveraging generative artificial intelligence in computer science education. In: *Proceedings of the 55th ACM Technical Symposium on Computer Science Education (SIGCSE 2024)*. Portland, OR, USA: ACM, New York, NY, USA, 2024. Disponível em: <https://doi.org/10.1145/3626252.3630938>. 14, 16, 17

LYU, W. et al. Evaluating the effectiveness of llms in introductory computer science education: A semester-long field study. In: *Proceedings of the Eleventh ACM Conference on Learning @ Scale (L@S '24)*. Atlanta, GA, USA: ACM, New York, NY, USA, 2024. Disponível em: <https://doi.org/10.1145/3657604.3662036>. 14, 16, 43, 45

MADAAN, A. et al. Learning performance-improving code edits. *arXiv:2302.07867*, 2023. 12

MARQUES, F. B.; MARQUES, Y. B.; MACULAN, B. C. M. dos S. Coocorrência de palavras-chave em dados abertos da capes: teses e dissertações em ciência da informação. *Múltiplos Olhares em Ciência da Informação*, 2021. 10

MASTROPAOLO, A. et al. *On the Robustness of Code Generation Techniques: An Empirical Study on GitHub Copilot*. 2023. 1

MONTEIRO, M. et al. Nocodegpt: A no-code interface for building web apps with language models. *Software: Practice and Experience*, n/a, n. n/a, 2025. Disponível em: <https://onlinelibrary.wiley.com/doi/abs/10.1002/spe.3432>. 15

MOURA, M. F. et al. Um modelo para a seleção de n-gramas significativos e não redundantes em tarefas de mineração de textos. Campinas: Embrapa Informática Agropecuária, 2010., 2010. 10

NGUYEN-DUC, A. et al. *Generative Artificial Intelligence for Software Engineering – A Research Agenda*. 2023. Disponível em: <https://arxiv.org/abs/2310.18648>. 1, 2, 12, 45

NOEVER, D.; WILLIAMS, K. Chatbots as fluent polyglots: Revisiting breakthrough code snippets. *arXiv:2301.03373 [cs]*, 2023. 12

OPENAI. *Best practices for prompt engineering with the OpenAI API*. 2024. <https://help.openai.com/en/articles/6654000-best-practices-for-prompt-engineering-with-the-openai-api#h_21d4f4dc3d>. Acessado em: 02 de setembro de 2024. 18

OPENAI. *Get answers. Find inspiration. Be more productive.* 2024. Disponível em: <https://openai.com/chatgpt/>. 1

OPENAI. *Prompt engineering best practices for ChatGPT*. 2024. <https://help.openai.com/en/articles/10032626-prompt-engineering-best-practices-for-chatgpt>. Acessado em: 02 de setembro de 2024. 18, 21

OPENAI. *Prompt Engineering Guide: Write Clear Instructions*. 2024. <https://platform.openai.com/docs/guides/prompt-engineering/strategy-write-clear-instructions>. Acesso em: 2 set. 2024. 18, 21

OTTER, D. W.; MEDINA, J. R.; KALITA, J. K. *A Survey of the Usages of Deep Learning in Natural Language Processing*. 2019. 10

PARREIRA, A.; LEHMANN, L.; OLIVEIRA, M. O desafio das tecnologias de inteligência artificial na educação: percepção e avaliação dos professores. *Ensaio: avaliação e políticas públicas em educação*, SciELO Brasil, v. 29, p. 975–999, 2021. 2

PETROVSKA, O. et al. Incorporating generative ai into software development education. In: *Computing Education Practice (CEP '24)*. Durham, United Kingdom: ACM, New York, NY, USA, 2024. Disponível em: <https://doi.org/10.1145/3633053.3633057>. 15, 45

PRESSMAN, R. S.; MAXIM, B. R. *Engenharia de software-8*. [S.l.]: McGraw Hill Brasil, 2016. 7

RAHMAN, M. M.; WATANOBE, Y. Chatgpt for education and research: Opportunities, threats, and strategies. *Applied Sciences*, v. 13, n. 5783, 2023. Disponível em: <https://doi.org/10.3390/app13095783>. 14, 44, 46

RODRIGUES, O. S.; RODRIGUES, K. S. A inteligência artificial na educação: os desafios do chatgpt. *Texto Livre*, SciELO Brasil, v. 16, p. e45997, 2023. 2

RUSSELL, S.; NORVIG, P. *Inteligência Artificial: Uma Abordagem Moderna*. 4. ed. São Paulo: Pearson, 2022. 11

SANTOS, R. A. d.; REATEGUI, E. B.; CAREGNATO, S. E. Análise de coocorrência de palavras na pesquisa brasileira em hiv/aids indexada na web of science no período 1993-2020. *Informação & informação. Londrina, PR. Vol. 27, n. 2 (abr./jun. 2022), p. 248-273*, 2022. 10

SOCIETY, I. C. *Guide to the Software Engineering Body of Knowledge*. 3. ed. Piscataway, NJ, USA: IEEE, 2014. ISBN 978-0-7695-5166-1. 1

SOMMERVILLE, I. *Software Engineering, 9/E*. [S.l.]: Pearson Education India, 2011. 5, 6, 7

SPIEGEL, M. R. Estatística, 3ª edição. *São Paulo: Makron*, v. 1994, 1993. 7, 8

SUN, J. et al. Investigating explainability of generative AI for code through scenario-based design. In: *27th International Conference on Intelligent User Interfaces, IUI '22*. Association for Computing Machinery, 2022. p. 212–228. Disponível em: <https://dl.acm.org/doi/10.1145/3490099.3511119>. 12, 13, 43, 45

TORREJÓN, D. A. R.; RAMOS, J. M. M. Detección de plagio en documentos. sistema externo monolingüe de altas prestaciones basado en n-gramas contextuales. *Procesamiento del lenguaje natural*, v. 45, p. 49–57, 2010. 10

TRIOLA, M. F. Introdução à estatística, 10a. edição. *Rio de Janeiro: LTC editora*, 2008. 9

VAITHILINGAM, P.; ZHANG, T.; GLASSMAN, E. L. Expectation vs. experience: Evaluating the usability of code generation tools powered by large language models. In: *Extended Abstracts of the 2022 CHI Conference on Human Factors in Computing Systems*. New York, NY, USA: Association for Computing Machinery, 2022. (CHI EA '22). ISBN 9781450391566. Disponível em: <https://doi.org/10.1145/3491101.3519665>. 1

VALENTE, M. T. *Engenharia de Software Moderna: Princípios e Práticas para Desenvolvimento de Software com Produtividade*. [S.l.]: Editora: Independente, 2020. 5, 6, 7

VASWANI, A. et al. Attention is all you need. In: *Advances in Neural Information Processing Systems*. [s.n.], 2017. v. 30. Disponível em: <https://doi.org/10.1016/j.caeai.2023.100147>. 1, 11, 12

WANG, J. et al. Software testing with large language model: Survey, landscape, and vision. *arXiv preprint arXiv:2307.07221*, 2023. 12

WANG, R. et al. *Investigating and Designing for Trust in AI-powered Code Generation Tools*. 2023. 1

WASEEM, M. et al. *ChatGPT as a Software Development Bot: A Project-based Study*. 2024. Disponível em: <https://arxiv.org/abs/2310.13648v2>. 14, 15, 16, 44, 45, 47

WEISZ, J. D. et al. Perfection not required? human-AI partnerships in code translation. In: *26th International Conference on Intelligent User Interfaces, IUI '21*. Association for Computing Machinery, 2021. p. 402–412. Disponível em: <https://doi.org/10.1145/3397481.3450656>. 12

WHITE, J. et al. A prompt pattern catalog to enhance prompt engineering with chatgpt. 2 2023. Disponível em: <http://arxiv.org/abs/2302.11382>. 13, 16, 43

WHITE, J. et al. *A Prompt Pattern Catalog to Enhance Prompt Engineering with ChatGPT*. 2023. Disponível em: <https://arxiv.org/abs/2302.11382>. 21

WHITE, J. et al. Chatgpt prompt patterns for improving code quality, refactoring, requirements elicitation, and software design. 3 2023. Disponível em: <http://arxiv.org/abs/2303.07839>. 13, 16, 43

YETIŞTIREN, B. et al. *Evaluating the Code Quality of AI-Assisted Code Generation Tools: An Empirical Study on GitHub Copilot, Amazon CodeWhisperer, and ChatGPT*. 2023. 1

YUAN, Z. et al. No more manual tests? evaluating and improving ChatGPT for unit test generation. *arXiv preprint arXiv:2305.04207*, 2023. 12

Apêndices

APÊNDICE A – PROMPTS DE SISTEMA

A.1 Engenharia de Requisitos

Dada a seguinte descrição sobre requisitos:

Os requisitos de um sistema descrevem o que ele deve fazer, os serviços que oferece e as restrições aplicáveis ao seu funcionamento. Eles são divididos em Requisitos Funcionais, que especificam as ações que o sistema deve realizar, e Requisitos Não-Funcionais, que delineiam as restrições e condições sob as quais o sistema deve operar. Além disso, os requisitos podem ser classificados como de Usuário, escritos em linguagem natural e diagramas para descrever os serviços oferecidos aos usuários e suas restrições, e de Sistema, que são descrições detalhadas do funcionamento do software, incluindo suas funções, serviços e restrições.

Requisitos não-funcionais são subdivididos em três categorias: Requisitos de Produto, que abordam o desempenho do software, como a rapidez das transações; Requisitos Organizacionais, que envolvem políticas e procedimentos da organização cliente e desenvolvedora, como normas a serem seguidas; e Requisitos Externos, que são influenciados por fatores externos, como legislações e normas técnicas. Bons requisitos devem ser corretos, precisos, completos, consistentes e verificáveis, garantindo que não haja ambiguidades, que todas as funcionalidades sejam cobertas, e que seja possível testar se eles estão sendo atendidos.

A partir dos requisitos, elaboram-se os casos de uso, que são escritos da perspectiva do ator que utilizará o sistema para alcançar um objetivo específico. Cada caso de uso inclui um fluxo normal e extensões para representar situações de erro ou variações no processo. Isso ajuda a assegurar que todas as possíveis interações e cenários de uso do sistema sejam considerados, proporcionando uma base sólida para o desenvolvimento e a verificação do software.

Você será um assistente para auxiliar no levantamento de requisitos dos alunos de uma disciplina de Engenharia de Software. PROIBIDO CÓDIGO. PROIBIDO TUDO FORA DE REQUISITOS. LIMITE DE 700 TOKENS NA RESPOSTA. Resposta em português do Brasil.

A.2 Testes de Software

Dada a seguinte descrição sobre testes:

Os testes de software são um conjunto de instruções planejadas e executadas sistematicamente para garantir que o programa funcione conforme esperado e corrigir quaisquer defeitos antes de seu uso comercial. Existem três grupos principais de testes automatizados: testes de unidade, que verificam pequenos trechos de código; testes de integração, que verificam funcionalidades completas do sistema e podem usar componentes externos; e testes de sistema, que simulam sessões de uso do sistema pelo usuário final, sendo mais caros e sensíveis a mudanças.

Os testes de software podem ser realizados de duas maneiras: testes caixa-preta, que avaliam se o software cumpre suas funções sem se preocupar com a estrutura interna, e testes caixa-branca, que analisam o funcionamento interno do código. Testes caixa-preta se concentram nos requisitos funcionais e podem identificar uma classe diferente de erros em comparação com os testes caixa-branca, que envolvem uma análise rigorosa da lógica do código. É essencial selecionar um número limitado de caminhos lógicos importantes para rodar os testes.

Existem várias abordagens para ambas as filosofias de teste. Para testes caixa-branca, temos o teste do caminho básico, que avalia todos os caminhos independentes no grafo de fluxo; o teste de condição, que exercita as condições lógicas; e o teste de ciclo, que avalia os ciclos no fluxo. Para testes caixa-preta, existem o particionamento de equivalência, que divide as entradas em classes de equivalência; a análise de valor limite, que avalia entradas nas fronteiras dos domínios; e o teste de interface, que testa os elementos da interface dos componentes desenvolvidos.

Você será um assistente para auxiliar na elaboração de testes dos softwares dos alunos de uma disciplina de Engenharia de Software. PROIBIDO MAIS QUE 1 FUNÇÃO DE TESTE. PROIBIDO TUDO FORA DE TESTES. LIMITE DE 700 TOKENS NA RESPOSTA. Resposta em português do Brasil

APÊNDICE B – Padrões de *Prompt* - Levantamento de Requisitos, com Exemplos

B.1 Persona

Permite ao modelo atuar como uma persona específica, como um analista de requisitos ou outro papel, como stakeholder ou um usuário final. Exemplo: “Atue como um analista de requisitos e me ajude a identificar os requisitos funcionais e não funcionais de um sistema para um site de e-commerce.”

B.2 Interação Invertida

Faz o LLM perguntar até obter todas as informações necessárias. Exemplo: “Você será responsável por me fazer perguntas sobre os requisitos de um sistema de gestão hospitalar. Pergunte até que tenha todas as informações necessárias para elaborar os requisitos detalhados.”

B.3 Refinamento de Perguntas

Pede ao modelo sugestões de como melhorar suas perguntas ou requisitos. Exemplo: “Como posso melhorar este requisito para torná-lo mais completo e verificável?”

B.4 Template

Estabelece um modelo para garantir que os requisitos sigam um formato padrão. Exemplo: “A seguir está um template para um requisito funcional. Preencha-o com as informações de um sistema de agendamento de consultas médicas: ‘O sistema deve [ação], quando [condição], para [benefício].”’

B.5 Reflexão

Solicita ao LLM que explique sua lógica ou raciocínio por trás de uma resposta. Isso pode ser útil para os alunos que precisam entender por que certos requisitos são importantes e como eles se alinham com os objetivos do sistema. Exemplo: “Explique por que este requisito não-funcional é essencial para um sistema de e-commerce seguro.”

B.6 Abordagens Alternativas

Solicita abordagens alternativas para um mesmo problema, para ter mais opções além de uma única solução. Exemplo: “Quais abordagens alternativas você me sugere para implementar a segurança de login em um sistema de e-commerce?”

B.7 Fact-check List

Pede uma lista com os principais fatores a serem verificados em um determinado requisito. Exemplo: “Liste os principais fatos sobre os requisitos de desempenho que devo verificar no meu sistema de gestão hospitalar.”

APÊNDICE C – Padrões de *Prompt* - Testes, com Exemplos

C.1 Persona

Permite ao modelo atuar como uma persona específica, como um testador. Exemplo: “Atue como um testador de unidade. Crie um teste para a função calcularDesconto() da classe CarrinhoDeCompras, garantindo que ela aplique o desconto correto para um código promocional válido e que falhe para códigos inválidos.”

C.2 Fact-check List

Pede uma lista com os principais fatores a serem verificados em um determinado esquema de testes. Exemplo: “Liste os principais fatos sobre a funcionalidade X que devo testar.”

C.3 Refinamento de Testes

Pede ao modelo sugestões de como melhorar seus testes. Exemplo: “Como posso melhorar este teste: [teste] para aumentar a sua cobertura do código?”

C.4 Reflexão

Solicita ao LLM que explique sua lógica ou raciocínio por trás de uma resposta. Isso pode ser útil para os alunos que precisam entender o porquê de algum teste. Exemplo: “Explique por que este teste deve ser feito: [teste].”

C.5 “Receita”

Solicita ao LLM uma sequência de passos para uma ação, com base em alguns “ingredientes” para atingir um objetivo. Exemplo: “Quero escrever um teste para funcionalidade X utilizando TDD. Eu sei que preciso escrever o teste, depois escrever o código e testá-lo para ver se passa. Qual o passo-a-passo completo utilizando linguagem Y?”

C.6 Abordagens Alternativas

Solicita abordagens alternativas para um mesmo problema, para ter mais opções além de uma única solução. Exemplo: “Quais outros testes você me sugere para verificar este requisito/esta função X?”

APÊNDICE D – Termo de Consentimento Aplicado aos Alunos

Título da Pesquisa: Explorando o Potencial de Large Language Models para a Educação em Engenharia de Software

Pesquisador Responsável: Bruno Mendes de Carvalho Castelo Branco

Orientador: Guilherme Amaral Avelino

Coorientador: Vinícius Ponte Machado

Instituição: Universidade Federal do Piauí

Programa de Mestrado: Programa de Pós-Graduação em Ciência da Computação

Objetivo: Avaliar o impacto das ferramentas de IA Generativa no aprendizado dos alunos em diferentes etapas do processo de desenvolvimento de software.

Procedimento: Após utilizarem a ferramenta para um dos processos de software, você responderá a esse questionário. O preenchimento é destinado **exclusivamente** para os alunos da disciplina de Engenharia de Software II da Universidade Federal do Piauí, porém o anonimato será mantido. Apenas os pesquisadores terão acesso ao que for respondido aqui.

Todas as informações fornecidas serão tratadas com absoluta confidencialidade e utilizadas exclusivamente para fins acadêmicos e científicos. A análise dos dados será realizada de forma agregada, garantindo o anonimato dos participantes e impedindo a identificação individual.

O endereço de e-mail e a matrícula serão utilizados apenas para controlar as respostas recebidas e para integrar as informações coletadas com os dados de uso da ferramenta, visto que ambos estão vinculados à matrícula. Em nenhum momento a identidade dos participantes será divulgada ou associada às suas respostas.

Não há riscos aos participantes, e a resposta feita de forma honesta pode contribuir para que possamos avaliar a eficácia do uso de IA Generativa no seu aprendizado e, se possível, identificar melhores práticas para sua adoção do processo de ensino.

Declaração de Consentimento: Ao selecionar a opção “Concordo”, você declara que

leu e compreendeu as informações fornecidas acima, que teve a oportunidade de tirar quaisquer dúvidas e que aceita participar desta pesquisa.

Marque a opção abaixo para confirmar seu consentimento:

Concordo em participar da pesquisa e autorizo a coleta dos dados. (Campo Obrigatório)

APÊNDICE E – Estrutura do Questionário

E.1 Dados Demográficos

- Matrícula (Texto curto)
- Nível de experiência (Apenas estudante, Estagiário, Profissional)

E.2 Uso da Ferramenta

- Q1: Você utilizou a ferramenta de IA para a atividade de Levantamento de Requisitos / Testes? (Sim/Não)
- Q2: Se não utilizou, qual foi o motivo? (Texto aberto)
- Q3: Quantas horas você dedicou a essa tarefa utilizando a ferramenta de IA? (0-2 horas / 2-4 horas / Mais de 4 horas / Não utilizei)

E.3 Avaliação da Ferramenta

- Q4: Como você avalia o impacto da ferramenta de IA no seu progresso de aprendizagem? (Escala Likert de 1 a 5: 1 - Muito ruim, 2 - Ruim, 3 - Regular, 4 - Bom, 5 - Ótimo)
- Q5: Como você avalia a adequação da ferramenta de IA para essa tarefa? (Escala Likert de 1 a 5: 1 - Muito ruim, 2 - Ruim, 3 - Regular, 4 - Bom, 5 - Ótimo)
- Q6: Como você avalia a adequação da ferramenta de IA para aprender conceitos relacionados a essa tarefa? (Escala Likert de 1 a 5: 1 - Muito ruim, 2 - Ruim, 3 - Regular, 4 - Bom, 5 - Ótimo)
- Q7: Quanto esforço adicional foi necessário para adaptar o conteúdo gerado pela ferramenta de IA para essa tarefa? (Escala Likert de 1 a 5: 1 - Muito ruim, 2 - Ruim, 3 - Regular, 4 - Bom, 5 - Ótimo)

E.4 Considerações Finais

- Q8: De 1 a 5, qual o seu nível de satisfação com a ferramenta? (Escala Likert de 1 a 5: 1 - Muito ruim, 2 - Ruim, 3 - Regular, 4 - Bom, 5 - Ótimo)

- Q9: Você utilizaria novamente o ChatGPT para este fim? (Sim/Não)
- Q10: Qual foi sua maior dificuldade para Levantamento de Requisitos? (Texto aberto)
- Q11: O que você achou mais positivo na sua experiência com a ferramenta para Levantamento de Requisitos / Testes? (Texto aberto)
- Q12: O que você achou dos padrões de *prompt* sugerido para o Levantamento de Requisitos / Testes? (Texto aberto)
- Q13: Você utilizou outra estratégia para a elaboração dos *prompts*? (Sim/Não)
- Q14: Se sim, os resultados foram melhores? (Sim/Não/Não utilizei outra estratégia)
- Q15: Fale mais sobre sua experiência (Texto aberto)