MATHEUS GERMANO DA COSTA

INTELIGÊNCIA ARTIFICIAL NO DESENVOLVIMENTO DE *SOFTWARE*

CAMPINAS

2024

MATHEUS GERMANO DA COSTA

INTELIGÊNCIA ARTIFICIAL NO DESENVOLVIMENTO DE *SOFTWARE*

Trabalho de Conclusão de Curso apresentado como exigência parcial para obtenção do diploma do Curso de Tecnologia em Análise e Desenvolvimento de Sistemas do Instituto Federal de Educação, Ciência e Tecnologia Câmpus Campinas.

Orientador: Prof. Ricardo Barz Sovat

CAMPINAS

2024

FICHA CATALOGRÁFICA

Ficha Catalográfica
Instituto Federal de São Paulo – Campus Campinas
Biblioteca “Pedro Augusto Pinheiro Fantinatti”
Rosangela Gomes - CRB8/8461

Costa, Matheus Germano da
C837i Inteligência artificial no desenvolvimento de software. / Matheus Germano da Costa.
Campinas, SP: [s.n.], 2024.
43 f. : il.

Orientador: Dr. Ricardo Barz Sovat
Trabalho de Conclusão de Curso (graduação) – Instituto Federal de Educação, Ciência e Tecnologia de São Paulo Campus Campinas. Curso de Tecnologia em Análise e Desenvolvimento de Sistemas, 2024.

1. Impactos. 2. Inteligência artificial. 3. Software. I. Instituto Federal de Educação, Ciência e Tecnologia de São Paulo Campus Campinas, Curso de Análise e Desenvolvimento de Sistemas. II. Título.

ATA N.º 16/2024 - TADS-CMP/DAE-CMP/DRG/CMP/IFSP

Ata de Defesa de Trabalho de Conclusão de Curso - Graduação

Na presente data, realizou-se a sessão pública de defesa do Trabalho de Conclusão de Curso intitulado **INTELIGÊNCIA ARTIFICIAL NO DESENVOLVIMENTO DE *SOFTWARE***, apresentado pelo aluno **Matheus Germano da Costa (CP301388X)** do Curso **SUPERIOR DE TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS** (Campus Campinas). Os trabalhos foram iniciados às **19h30m** pelo Professor presidente da banca examinadora, constituída pelos seguintes membros:

Membros	Instituição	Presença (Sim/Não)
Ricardo Barz Sovat (Presidente/Orientador)	IFSP-CMP	SIM
Carlos Eduardo Beluzo(Examinador 1)	IFSP-CMP	SIM
Everton Josué Meyer da Silva (Examinador 2)	IFSP-CMP	SIM

Observações:

A banca examinadora, tendo terminado a apresentação do conteúdo da monografia, passou à arguição do candidato. Em seguida, os examinadores reuniram-se para avaliação e deram o parecer final sobre o trabalho apresentado pelo aluno, tendo sido atribuído o seguinte resultado:

[**X**] Aprovado [ ] Reprovado

Proclamados os resultados pelo presidente da banca examinadora, foram encerrados os trabalhos e, para constar, eu lavrei a presente ata que assino em nome dos demais membros da banca examinadora.

Campus Campinas, 2 de dezembro de 2024

Documento assinado eletronicamente por:

- **Ricardo Barz Sovat**, **PROFESSOR ENS BASICO TECN TECNOLOGICO**, em 02/12/2024 22:34:34.
- **Everton Josue Meyer da Silva**, **PROFESSOR ENS BASICO TECN TECNOLOGICO**, em 03/12/2024 09:34:34.
- **Carlos Eduardo Beluzo**, **PROFESSOR ENS BASICO TECN TECNOLOGICO**, em 11/12/2024 21:15:46.

Este documento foi emitido pelo SUAP em 02/12/2024. Para comprovar sua autenticidade, faça a leitura do QRCode ao lado ou acesse https://suap.ifsp.edu.br/autenticar-documento/ e forneça os dados abaixo:

Código Verificador: 853133
Código de Autenticação: 0146d929d8

Dedico este trabalho a todos que estiveram ao meu lado e me apoiaram de alguma forma ao longo de seu desenvolvimento. Também o dedico aos pesquisadores da área de tecnologia da informação, cujas descobertas e análises contribuíram para o avanço do conhecimento e para a realização deste estudo.

AGRADECIMENTOS

Agradeço, primeiramente, à minha família, que me proporcionou a oportunidade de estudar e me ofereceu todo o apoio ao longo desta jornada. Em seguida, agradeço ao meu orientador, Ricardo Barz Sovat, por aceitar embarcar comigo nesse desafio. Um agradecimento especial aos meus amigos Matheus Costa e Renan Oliveira, que estiveram presentes em diversos momentos do curso e de quem aprendi muito. Por fim, agradeço a todos os professores e servidores do IFSP Campus Campinas que, de alguma forma, contribuíram para que eu chegasse até aqui.

"Sorte é o que acontece quando
a preparação encontra a oportunidade".
Lucius Annaeus Seneca

RESUMO

Este trabalho tem como objetivo investigar e analisar a aplicação da inteligência artificial no desenvolvimento de *software*. A inteligência artificial tem se mostrado uma área promissora, capaz de trazer benefícios significativos para a indústria de desenvolvimento de programas. No contexto atual, em que a demanda por *software* é cada vez maior e mais complexa, a inteligência artificial oferece soluções que podem auxiliar os desenvolvedores a enfrentar desafios e otimizar o processo de desenvolvimento. Através de algoritmos e técnicas de aprendizado de máquina, a inteligência artificial pode automatizar tarefas repetitivas, melhorar a qualidade dos programas e acelerar o ciclo de desenvolvimento, porém isso pode causar maus hábitos e vícios se não utilizada corretamente. Neste trabalho será investigado, através de formulários e testes de algoritmos, como ferramentas de inteligência artificial podem impactar no processo de desenvolvimento de *software* pelo programador, investigando os benefícios e malefícios causados pelo uso dessa tecnologia. Por fim, espera-se que este trabalho contribua para a compreensão dos impactos da inteligência artificial no desenvolvimento de *software*, auxiliando os profissionais da área na tomada de decisões e no aproveitamento dos benefícios proporcionados por essas tecnologias em constante evolução.

Palavras-chave: impactos; inteligência artificial; *software*.

ABSTRACT

This work aims to investigate and analyze the application of artificial intelligence in software development. Artificial intelligence has proven to be a promising area, capable of bringing significant benefits to the software industry. In the current context, where the demand for programs is increasing and becoming more complex, artificial intelligence offers solutions that can help developers tackle challenges and optimize the development process. Through algorithms and machine learning techniques, artificial intelligence can automate repetitive tasks, improve software quality, and accelerate the development cycle. However, if not used correctly, it can lead to bad habits and dependencies. In this work, we will investigate, through form and algorithm testing, how artificial intelligence tools can impact the software developer's development process, exploring the benefits and drawbacks caused by the use of this technology. Finally, it is expected that this research will contribute to understanding the impacts of artificial intelligence in software development, assisting professionals in the field in making informed decisions and harnessing the benefits provided by these constantly evolving technologies.

Keywords: artificial intelligence; impacts; software.

LISTA DE FIGURAS

LISTA DE GRÁFICOS

LISTA DE QUADROS

LISTA DE SIGLAS

DRY - Don’t Repeat Yourself

GPT - Generative Pre-trained Transformer

IA - Inteligência Artificial

KISS - Keep It Simple, Stupid

LLM - Large Language Model

PIB - Produto Interno Bruto

SOLID - Single responsibility principle, Open–closed principle, Liskov substitution principle, Interface segregation principle, Dependency inversion principle

SUMÁRIO

1 INTRODUÇÃO

De acordo com Damioli, Roy e Vertesy (2021), as últimas décadas testemunharam grandes desenvolvimentos na tecnologia de Inteligência Artificial (IA) que vêm sendo amplamente utilizados em diversas áreas, incluindo o desenvolvimento de *software*, e causando um debate sobre o seu impacto na sociedade, de acordo com Makridakis (2017).

Recentemente, a *Open AI*, uma organização de pesquisa sobre IA dos Estados Unidos, lançou ao público o *Chat GPT*, em Novembro de 2022 (Styve; Virkki; Naeem, 2024). Desde então a utilização dele e de ferramentas parecidas é bem comentada e utilizada dentro da comunidade de desenvolvimento de *software*.

"A utilização de IA pode ser uma ferramenta valiosa quando utilizada de forma correta, uma vez que pode ajudar os programadores que a utilizam a lidar com problemas mais complexos, aumentar a eficiência no processo de resolução dos mesmos e um possível maior crescimento profissional" (Damioli; Roy; Vertesy, 2021, tradução própria). Por outro lado, ferramentas de IA podem não estar disponíveis em todas as situações. Para programadores inexperientes, isto pode gerar uma dependência excessiva e até mesmo resultar em códigos de baixa qualidade, fazendo com que o programador jovem não desenvolva o suficiente para que ele saiba como resolver os problemas.

De acordo com Cui et al. (2024), as ferramentas de IA generativa ajudam mais os trabalhadores com menor habilidade ou menor experiência e, de acordo com Styve, Virkki e Naeem (2024), elas estão aqui para ficar e os usuários devem aprender a usá-las de forma responsável.

Styve, Virkki e Naeem (2024) afirmam que desde a introdução de ferramentas de IA, estudantes as usam para gerar código sem compreender totalmente o que ele está fazendo e que, em paralelo, ter o pensamento crítico é parte fundamental das melhores práticas de programação.

O objetivo deste trabalho é analisar como a utilização de ferramentas de IA pode afetar o processo de desenvolvimento dos programadores, destacando tanto os benefícios quanto às limitações que essa tecnologia pode trazer para a formação e aprimoramento de profissionais da área.

Assim, este trabalho busca contribuir para uma melhor compreensão do papel das inteligências artificiais no desenvolvimento de *software*, a fim de fornecer subsídios para uma utilização mais consciente e eficiente dessas ferramentas pelos programadores.

2 JUSTIFICATIVA

S*oftwares* são utilizados diariamente, já estão "embutidos" no dia a dia da maioria da população. De acordo com o Centro Regional de Estudos para o Desenvolvimento da Sociedade da Informação (2023), 84% da população brasileira com 10 anos ou mais utiliza a internet e, consequentemente, utiliza *softwares*, em computadores, *notebooks*, celulares ou *tablets*.

O desenvolvimento de *software* ocorre há muito tempo e vem passando por mudanças significativas nas últimas décadas (Styve; Virkki; Naeem, 2024). Após a liberação ao público de ferramentas de IA, desenvolvedores passaram a utilizar essas ferramentas no seu dia a dia, tirando dúvidas de como resolver um erro ou de como fazer certa tarefa. De acordo com Cui et al. (2024), especificamente sobre o *Github Copilot*, a ferramenta analisa o contexto do projeto e gera trechos de código, comentários e documentação relevantes. Dessa forma, esse recurso pode economizar tempo dos desenvolvedores e potencialmente melhorar a qualidade do código, oferecendo sugestões que eles talvez não conheçam. Porém, como em todas as ferramentas baseadas em *Large Language Models* (LLMs), o *Copilot* pode se equivocar e, se os desenvolvedores confiarem nele sem revisão, ele pode potencialmente introduzir erros ou diminuir a qualidade do código. Tais mudanças podem acarretar em impactos no mercado de trabalho.

De acordo com Davis (2023), o *Stack Overflow* demitiu 28% de sua equipe após se passar um ano de uma contratação em massa no início do avanço das IAs. Segundo ele, isso claramente apresentava uma "ameaça" a um fórum de ajuda em programação já que as pessoas ficavam confortáveis em usar *chatbots*. Em dezembro de 2022, a empresa proibiu temporariamente os usuários de responderem perguntas com a ajuda deles.

Por um lado, utilizar IA pode ser valioso quando utilizado de forma correta, ajudando a lidar com problemas mais complexos e aumentando a eficiência do desenvolvimento (Damioli; Roy; Vertesy, 2021). Por outro lado, segundo Kabir et al. (2023, tradução própria) — "Nossos resultados mostram que 52% das respostas geradas pelo *Chat GPT* são incorretas e 62% são verbosas. Além disso, aproximadamente 78% das respostas sofrem de inconsistência".

Por isso, é fundamental analisar cuidadosamente o uso de ferramentas de IA no desenvolvimento de *software*. Quando usadas corretamente, essas ferramentas podem ser grandes aliadas, aumentando a eficiência, ajudando a encontrar erros e melhorando a qualidade do código. No entanto, se utilizadas sem senso crítico, elas podem introduzir erros e

prejudicar o desenvolvimento, especialmente quando o usuário confia cegamente nos resultados.

2.1 Motivação

A motivação para o desenvolvimento deste projeto surgiu da minha experiência diária como desenvolvedor de *software*, enfrentando desafios de diferentes níveis de complexidade. Com o surgimento de novas ferramentas de IA que prometem auxiliar ou até mesmo substituir desenvolvedores, senti uma grande empolgação inicial. Por vezes, parecia algo mágico, digno de "filmes de ficção científica". No entanto, ao utilizá-las no meu trabalho, percebi que a experiência não era tão simples. Essas ferramentas, de fato, aumentaram minha produtividade em alguns aspectos, mas, em outros, acabavam gerando atrasos.

Os códigos gerados frequentemente apresentavam qualidade mediana ou vinham com erros que eu mesmo precisava identificar e corrigir, apontando as falhas e suas causas. O que mais me incomodava, porém, era ver pessoas ao meu redor confiando cegamente nos resultados dessas ferramentas, propagando um alarmismo de que todos nós seríamos substituídos.

Sempre tentei adotar uma postura analítica e crítica, avaliando os desafios com atenção e ponderando os caminhos possíveis. Foi essa característica, somada ao cenário que se formava à minha volta, que me motivou a estudar os impactos das ferramentas de IA. Meu objetivo é compreender tanto os benefícios quanto os riscos que elas podem trazer, contribuindo para o amadurecimento da comunidade de desenvolvimento de *software* em uma área que se mostra cada vez mais promissora.

3 OBJETIVOS

3.1 Objetivo geral

Essa pesquisa tem como objetivo investigar como a IA impacta no processo de desenvolvimento de *software*, no aprendizado do desenvolvedor e no mercado de trabalho, apontando benefícios e malefícios de seu uso diário a curto e longo prazo.

3.2 Objetivos específicos

- Analisar as informações presentes nas referências sobre conceito de IA, desenvolvimento de *software*, a utilização de IA no dia a dia do desenvolvedor, impactos nas capacidades cognitivas do usuário e como funcionam as IAs Generativas;
- Realizar pesquisas com o público-alvo por meio de um formulário, contendo perguntas sobre ferramentas de IA e desenvolvimento de *software*, englobando estudantes e profissionais da área de diversas senioridades;
- Realizar testes de algoritmos disponíveis na plataforma *Leet Code* com o *Chat GTP* na última versão 4o;
- Determinar a complexidade dos algoritmos gerados pelos *Chat GPT* através da notação *Big O*, considerando tempo de processamento e alocação de memória, e comparar os resultados, indicando pontos positivos e negativos.

4 FUNDAMENTAÇÃO TEÓRICA

A introdução da IA em processos de tecnologia é uma área em crescimento e que conta com bastante investimento. Em 2021, empresas privadas investiram cerca de US$ 93,5 bilhões nessa área ao redor do mundo, mais que o dobro investido em 2020 (Zhang et al., 2022), por isso, atualmente, é de extrema importância na área da tecnologia entender o que são IAs, como funcionam, quais suas finalidades e as consequências dela.

As IAs não estão presentes apenas na vida de quem trabalha com tecnologia, ela está no cotidiano das pessoas, em seus celulares, casas e trabalho. Atualmente, há assistentes virtuais muito utilizados, que são integrados com IA, como Google Assistant, presente nos celulares Android, Siri, presente nos produtos Apple, Cortana, da Microsoft e a Alexa, da Amazon.

De acordo com Davis (2023), desenvolvedores estão se acostumando a usar assistentes de programação, como o *Chat GPT* ou *Bing Chat*, e as ferramentas que utilizam IA presentes em seus produtos, preferindo isso invés de sites bem conhecidos de ajuda em programação, como o *Stack Overflow*.

O uso dessas ferramentas não é visto com maus olhos, e nem deve ser, a IA tem como um de seus objetivos automatizar processos, facilitar a vida de quem a utiliza, aumentar a eficiência de trabalhos e trazer informações de forma facilitada, porém deve ser usada cautelosamente e com um olhar mais crítico, visando que IAs não são perfeitas e vão sim errar, serem inconsistentes.

4.1 Desenvolvimento de *software*

O desenvolvimento de *software* tem sido um pilar central na transformação digital, permitindo a criação de sistemas que sustentam desde aplicações comerciais até soluções complexas de automação. Com o avanço de tecnologias emergentes, como a inteligência artificial, práticas consolidadas e conceitos fundamentais ganham novas interpretações.

4.1.1 Investimentos na área

O setor de desenvolvimento de *software* tem sido um dos maiores receptores de investimentos globais, refletindo a crescente dependência de soluções digitais. Segundo o estudo "Mercado Brasileiro de Software: Panorama e Tendências 2024", da Associação Brasileira das Empresas de Software (ABES), os investimentos globais com tecnologia da informação acumularam US$3,2 trilhões no ano de 2023, sendo 31% em software, 26% em

serviços e 41% em *hardware*. Esses recursos têm sido aplicados na modernização de sistemas, ampliação de plataformas de colaboração e desenvolvimento de aplicações específicas para atender às demandas de mercados em transformação.

Ainda de acordo com o estudo da ABES (2024), no Brasil, os negócios do setor estão operando em 24 estados e foram responsáveis por gerar 260 mil trabalhos diretos em 2023.

4.1.2 Estrutura de dados e algoritmos

Estruturas de dados e algoritmos são pilares essenciais do desenvolvimento de software, permitindo que sistemas sejam eficientes, escaláveis e resilientes. Estruturas de dados são formas organizadas de armazenar e gerenciar informações, enquanto algoritmos são conjuntos de passos lógicos que resolvem problemas computacionais.

Entre as estruturas de dados fundamentais estão:

- Arrays: São coleções de elementos armazenados em posições consecutivas de memória. Permitem acesso rápido (O(1)) quando o índice é conhecido, mas possuem custo linear (O(n)) para buscas.
- Listas Ligadas: Compostas por nós que armazenam dados e um ponteiro para o próximo elemento. São flexíveis em operações de inserção e remoção (O(1), com o ponteiro correto), mas possuem busca linear (O(n)).
- Pilhas e Filas:
    - Pilha (*Stack*): Segue o princípio LIFO (*Last In, First Out*), onde o último elemento inserido é o primeiro a ser removido.
    - Fila (*Queue*): Segue o princípio FIFO (*First In, First Out*), onde o primeiro elemento inserido é o primeiro a sair.
- Tabelas Hash: Estruturas de mapeamento chave-valor que permitem buscas e inserções rápidas (O(1) no caso médio), dependendo de um bom mecanismo de dispersão (hashing).

Essas estruturas são utilizadas em conjunto com algoritmos para otimizar a manipulação e o processamento de dados. Um exemplo clássico é a busca binária em *arrays* ordenados, que reduz significativamente o tempo de busca (O(log n)) em comparação à busca linear.

4.1.3 Complexidade de algoritmo

A análise da complexidade de um algoritmo é essencial para medir sua eficiência em termos de tempo de execução e espaço de memória requeridos. Esse tipo de análise desempenha um papel crucial na avaliação da viabilidade de soluções computacionais, especialmente em cenários onde o desempenho é um fator crítico. Ferramentas como a notação Big-O são amplamente utilizadas para classificar algoritmos de acordo com o comportamento em diferentes escalas de entrada, fornecendo categorias como O(1) (constante), O(log n) (logarítmica) e $O(n^2)$ (quadrática) (Sedgewick; Wayne, 2011; Bhargava, 2016).

4.1.3.1 Complexidade de tempo

A complexidade de tempo refere-se à relação entre o tempo de execução de um algoritmo e o tamanho de sua entrada (n). A notação Big-O é usada para descrever o comportamento do algoritmo no pior caso, oferecendo uma métrica padronizada para comparação entre diferentes soluções. Exemplos incluem:

- O(1): Tempo constante, independentemente do tamanho da entrada. Um exemplo típico é o acesso direto a elementos em *arrays*;
- O(log n): Crescimento logarítmico, como na busca binária, onde o espaço de busca é reduzido pela metade a cada iteração;
- O(n): Crescimento linear, observado em algoritmos que percorrem todos os elementos de uma lista;
- $O(n^2)$: Crescimento quadrático, característico de algoritmos menos eficientes como a ordenação por seleção (*selection sort*).

4.1.3.2 Complexidade de espaço

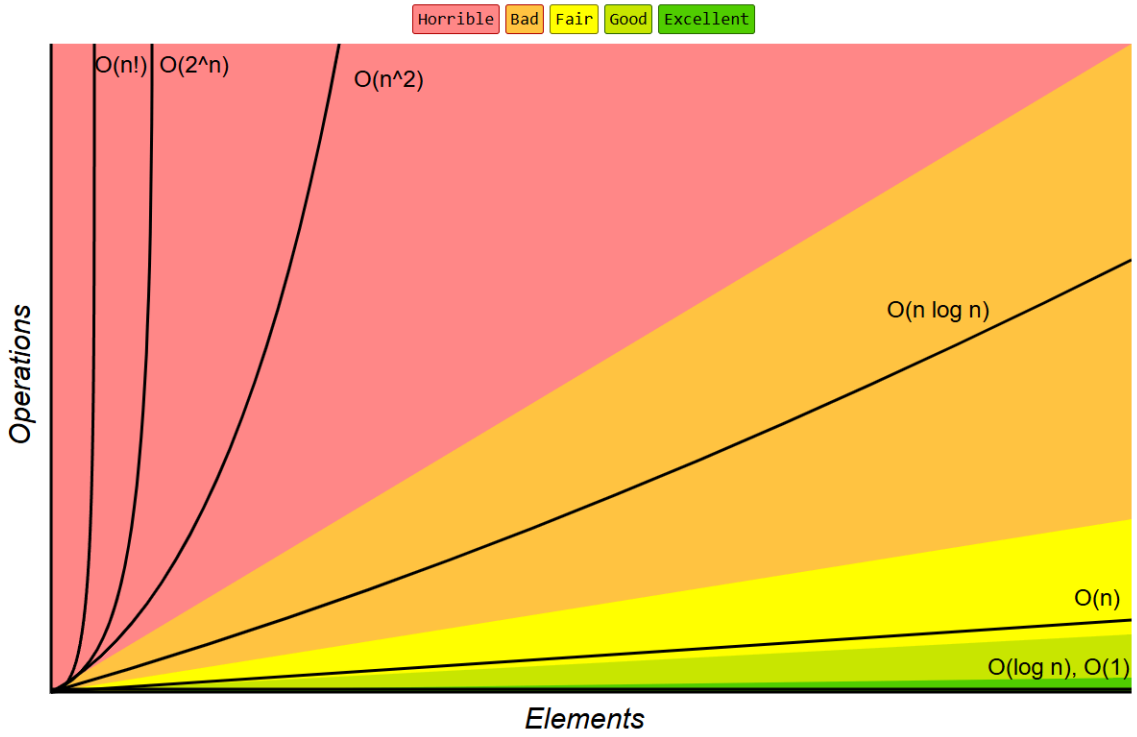
A complexidade de espaço avalia a quantidade de memória adicional exigida por um algoritmo durante sua execução. Ela inclui:

- Memória usada por variáveis temporárias;
- Espaço necessário para chamadas recursivas;
- Estruturas auxiliares empregadas no processamento.

Essa métrica é especialmente importante para garantir que um algoritmo seja viável em sistemas com restrições de recursos. Na figura 1 podemos observar o gráfico com os diversos

valores de complexidade possíveis para um algoritmo. A figura 2 exemplifica as complexidades possíveis de algoritmos comuns.

Figura 1 - Gráfico de complexidade na notação Big-O



Fonte: Big-O Cheat Sheet (2024)

Figura 2 - Operações de estrutura de dados comuns

Data Structure	Time Complexity								Space Complexity
	Average				Worst				Worst
	Access	Search	Insertion	Deletion	Access	Search	Insertion	Deletion	
Array	Θ(1)	Θ(n)	Θ(n)	Θ(n)	O(1)	O(n)	O(n)	O(n)	O(n)
Stack	Θ(n)	Θ(n)	Θ(1)	Θ(1)	O(n)	O(n)	O(1)	O(1)	O(n)
Queue	Θ(n)	Θ(n)	Θ(1)	Θ(1)	O(n)	O(n)	O(1)	O(1)	O(n)
Singly-Linked List	Θ(n)	Θ(n)	Θ(1)	Θ(1)	O(n)	O(n)	O(1)	O(1)	O(n)
Doubly-Linked List	Θ(n)	Θ(n)	Θ(1)	Θ(1)	O(n)	O(n)	O(1)	O(1)	O(n)
Skip List	Θ(log(n))	Θ(log(n))	Θ(log(n))	Θ(log(n))	O(n)	O(n)	O(n)	O(n)	O(n log(n))
Hash Table	N/A	Θ(1)	Θ(1)	Θ(1)	N/A	O(n)	O(n)	O(n)	O(n)
Binary Search Tree	Θ(log(n))	Θ(log(n))	Θ(log(n))	Θ(log(n))	O(n)	O(n)	O(n)	O(n)	O(n)
Cartesian Tree	N/A	Θ(log(n))	Θ(log(n))	Θ(log(n))	N/A	O(n)	O(n)	O(n)	O(n)
B-Tree	Θ(log(n))	Θ(log(n))	Θ(log(n))	Θ(log(n))	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(n)
Red-Black Tree	Θ(log(n))	Θ(log(n))	Θ(log(n))	Θ(log(n))	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(n)
Splay Tree	N/A	Θ(log(n))	Θ(log(n))	Θ(log(n))	N/A	O(log(n))	O(log(n))	O(log(n))	O(n)
AVL Tree	Θ(log(n))	Θ(log(n))	Θ(log(n))	Θ(log(n))	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(n)
KD Tree	Θ(log(n))	Θ(log(n))	Θ(log(n))	Θ(log(n))	O(n)	O(n)	O(n)	O(n)	O(n)

Fonte: Big-O Cheat Sheet (2024)

4.1.4 Boas práticas de programação

As práticas de codificação representam um conjunto de princípios e diretrizes que orientam o desenvolvimento de *software* com foco em clareza, manutenção e eficiência. Robert C. Martin destaca em suas obras, como *Clean Code: A Handbook of Agile Software Craftsmanship* (2008) e *Clean Architecture: A Craftsman's Guide to Software Structure and Design* (2017), que seguir boas práticas é essencial para garantir a sustentabilidade dos projetos de *software*.

4.1.4.1 Conceitos fundamentais

Martin (2008) define que o código deve ser limpo, ou seja, fácil de ler, compreender e modificar. Ele enfatiza que o código é, acima de tudo, uma forma de comunicação entre desenvolvedores, e que sua qualidade impacta diretamente na eficiência das equipes e na redução de custos ao longo do ciclo de vida do *software*. Entre as práticas mais relevantes, destacam-se:

- SOLID: Um conjunto de cinco diretrizes para a programação orientada a objetos que visa criar sistemas mais modulares e resilientes. Esses princípios incluem a responsabilidade única, o fechamento para modificação e abertura para extensão, e a inversão de dependências, entre outros. Quando aplicados corretamente, promovem maior flexibilidade e escalabilidade no *design* de *software* (Martin, 2008).
- DRY (*Don't Repeat Yourself*): Este princípio, também mencionado em *The Pragmatic Programmer* (Hunt; Thomas, 1999), é frequentemente reforçado por Martin. Ele estabelece que a duplicação de código deve ser evitada, pois aumenta o risco de inconsistências e dificulta a manutenção do *software*.
- KISS (*Keep It Simple, Stupid*): Praticar a simplicidade no desenvolvimento é uma recomendação recorrente em *Clean Code*. Martin afirma que soluções simples reduzem a propensão a erros e facilitam a compreensão do código por outros desenvolvedores.

4.1.4.2 Impacto no desenvolvimento de software

Práticas bem definidas ajudam a evitar problemas como dívidas técnicas e a complexidade desnecessária do sistema. *Softwares* bem projetados são mais fáceis de adaptar às mudanças, o que é crucial em ambientes ágeis. Por exemplo, ao aplicar os princípios de

código limpo, os desenvolvedores podem reduzir significativamente o tempo de depuração e os custos associados a defeitos em produção (Martin, 2017).

A maneira como o código está sendo desenvolvido também é importante em termos de manter um código de alta qualidade. Métodos como *pair programming*, *peer reviews*, etc., também devem ser usados ativamente em um ambiente de aprendizagem (Styve; Virkki; Naeem, 2024).

4.2 Inteligência Artificial

A IA é um campo científico e tecnológico que visa desenvolver sistemas computacionais capazes de executar tarefas que normalmente exigiriam inteligência humana, como aprendizado, raciocínio, percepção e resolução de problemas. Segundo Russel e Norvig (2021), a IA busca criar máquinas e programas que possam simular processos cognitivos, utilizando técnicas como aprendizado de máquina, redes neurais e algoritmos inteligentes para processar dados, identificar padrões e tomar decisões de forma autônoma e cada vez mais sofisticada. Essa tecnologia abrange múltiplas subáreas, como processamento de linguagem natural, visão computacional e robótica, com aplicações que se expandem rapidamente em setores como saúde, educação, finanças, entretenimento e pesquisa científica, prometendo revolucionar a forma como interagimos com tecnologias e resolvemos problemas complexos.

Mais de 90% das empresas de grande porte já têm algum processo implementado por IA (ABES, 2024). Acredita-se que a IA melhorou drasticamente a capacidade da humanidade, ou também sugere-se que ela é apenas uma ferramenta que auxilia em tarefas simples. Já alguns temem que a IA seja a ruína da humanidade, com humanos podendo ser substituídos. No momento, é claro que ela tem um impacto benéfico no fluxo, na produtividade e na satisfação no trabalho, porém não parece uma panaceia, uma bala de prata (DORA, 2024).

4.2.1 Investimentos na área

Empresas líderes, como Microsoft e Meta, têm liderado os investimentos em IA. Segundo a CNN Brasil (2024), a Meta destinou bilhões ao desenvolvimento de sistemas baseados em IA, prevendo aceleração em custos operacionais para acompanhar a demanda. Da mesma forma, a Microsoft redirecionou recursos significativos para integrar IA em suas soluções empresariais, evidenciando o potencial estratégico da tecnologia (Infomoney, 2024).

> Estimativas sugerem que os gigantes da tecnologia líderes investirão aproximadamente US$1 trilhão no desenvolvimento de IA nos próximos cinco anos. Isso se alinha bem com uma estatística apresentada no capítulo ‘Inteligência

> artificial: adoção e atitudes' de que 81% dos entrevistados dizem que suas empresas mudaram recursos para o desenvolvimento de IA (DORA, 2024, tradução própria).

A IA chega ao público com maior presença embarcada nos dispositivos no ano de 2024. Estimativas indicam que os dispositivos com IA embarcada serão os grandes impulsionadores de vendas a partir de 2026 (ABES, 2024).

4.2.2 IAs Generativas

IAs Generativas são um tipo de IA capaz de gerar conteúdos e ideias, abrangendo áreas como conversação, narrativas, imagens, vídeos e música. Essa tecnologia demonstra a capacidade de aprender e interpretar linguagens humanas, linguagens de programação, bem como conhecimentos avançados em áreas como arte, química e biologia, entre outras, elas tentam prever características dada uma certa entrada. Elas aprendem a distribuição de diferentes características de dados e seus relacionamentos (AWS, 2024), atualmente muito utilizada pela comunidade através do *Chat GPT.*

Styve, Virkki e Naeem (2024) dizem que o advento de ferramentas de IA Generativa, como *Chat GPT*, *GitHub Copilot* e *Amazon Code Whisperer*, melhorou ainda mais o kit de ferramentas do desenvolvedor, pois essas ferramentas podem ser usadas para uma ampla variedade de tarefas, como geração de código, documentação, comentários e revisão. No entanto, estudos revelam que as soluções geradas por essas ferramentas podem apresentar vulnerabilidades e inconsistências, exigindo análise crítica por parte dos desenvolvedores. As IAs generativas podem impulsionar um aumento de 7%, cerca de US$7 trilhões, no Produto Interno Bruto (PIB) global e aumentar a produtividade em 1,5 ponto percentual ao longo de dez anos (AWS, 2024).

4.2.2.1 Large Language Models

LLMs são, como o nome indica, grandes modelos de linguagem que utilizam a arquitetura de *transformers*, agrupando um conjunto de redes neurais que consistem em um codificador e um decodificador com capacidades de *self-attention*. O codificador e o decodificador extraem significados de uma sequência de texto e entendem as relações entre palavras e frases nele através dessa arquitetura. Esses modelos são treinados através de quantidade de dados massivos de textos e códigos-fonte de diversas fontes (Styve; Virkki; Naeem, 2024) e são focados especificamente em tarefas baseadas em linguagem, como resumos, geração de textos, classificação, conversação humanizada e extração de informações (AWS, 2024).

4.2.3 Adoção pelos desenvolvedores

A IA Generativa representa um avanço transformador no desenvolvimento de *software*. As ferramentas mais comuns, como o *Chat GPT,* estão revolucionando a produtividade dos programadores, oferecendo recursos inovadores de geração de código, documentação e revisão (Styve; Virkki; Naeem, 2024).

Dados mostram que 67% dos desenvolvedores relatam melhorias significativas em seus processos de codificação, porém de acordo com Cui et al. (2024), a IA generativa ajuda mais os trabalhadores com menor capacidade ou menor experiência. A IA tem demonstrado potencial para:

- Reduzir a complexidade do código
- Acelerar processos de revisão
- Aumentar a produtividade individual em aproximadamente 2,1%

Embora existam desafios, a capacidade de aprendizado crítico e a adaptação responsável são fundamentais para maximizar os benefícios dessas ferramentas tecnológicas.

4.2.4 Adoção pelas empresas

O cenário corporativo demonstra um compromisso robusto com a transformação digital. Impressionantes 81% das organizações já redirecionaram recursos para o desenvolvimento de capacidades em IA, com estimativas de investimento próximas a 1 trilhão de dólares nos próximos cinco anos (DORA, 2024).

Um exemplo concreto desta revolução tecnológica é a Google, onde mais de 25% de todo novo código é gerado por IA e subsequentemente revisado por engenheiros, conforme declaração do CEO Sundar Pichai (The Verge, 2024). Essa estratégia não apenas impulsiona a produtividade, mas também demonstra o potencial transformador da IA no ambiente corporativo. Principais benefícios organizacionais identificados:

- Melhoria na satisfação profissional
- Otimização de fluxos de trabalho
- Potencial de inovação incrementada
- Aceleração dos processos de desenvolvimento de software

As empresas que adotarem uma abordagem estratégica, combinando tecnologia com desenvolvimento de habilidades críticas, estarão melhor posicionadas para aproveitar o potencial transformador da IA.

5 METODOLOGIA

A pesquisa adotou uma metodologia de campo e descritiva, visando comparar o impacto de ferramentas de IA no desenvolvimento de *software* e no desenvolvedor que as utiliza.

5.1 Revisão bibliográfica

Para a escrita do projeto, uma base de conhecimento teórico deveria ser adquirida ao longo de seu desenvolvimento e este objetivo foi concluído através da busca e leitura de diversos artigos, *posts* e vídeos sobre desenvolvimento de *software*, inteligência artificial, inteligência artificial generativa e ferramentas de inteligência artificial.

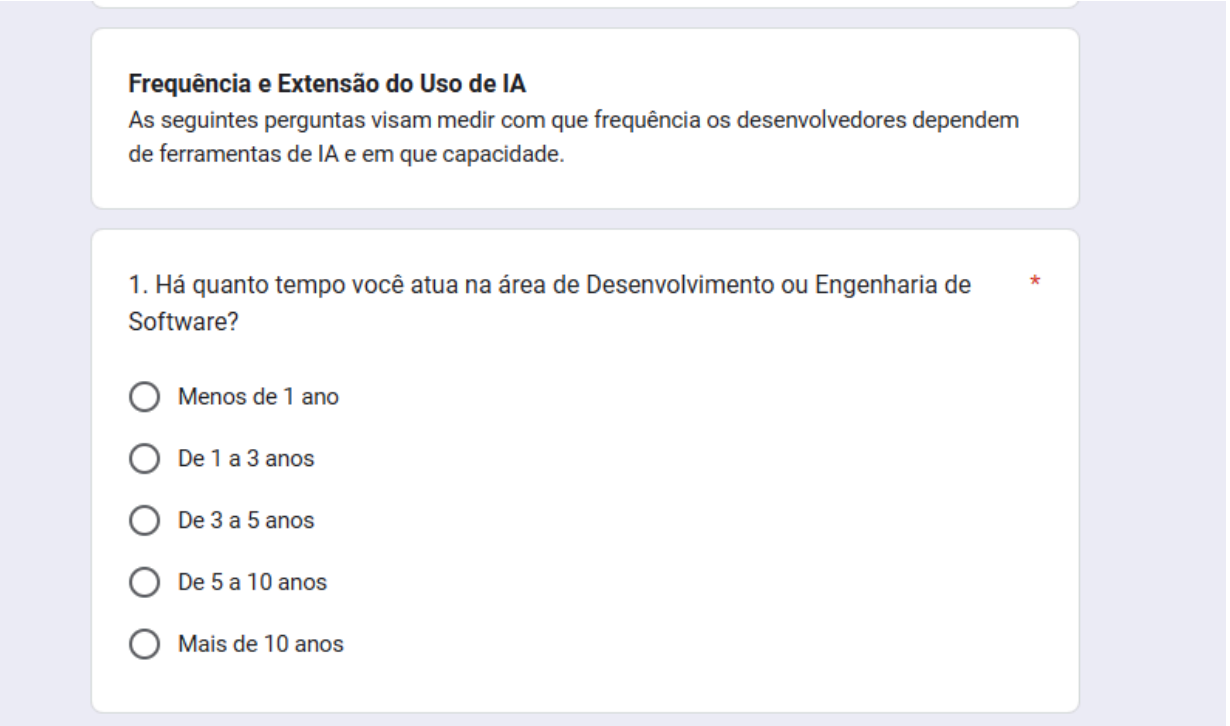
5.2 Coleta de dados

Foram utilizados dois métodos distintos para a coleta de dados quantitativos e qualitativos, visando alcançar uma compreensão mais aprofundada sobre a percepção dos programadores em relação ao impacto das ferramentas de IA no aprendizado e sua performance diante de requisições comuns no dia a dia dos usuários. Os instrumentos de coleta utilizados foram:

5.2.1 Questionário

Foi elaborado um questionário contendo 15 perguntas objetivas e 2 dissertativas mostrado na figura 3. Seu objetivo é explorar as experiências, percepções e desafios enfrentados pelo público-alvo durante o período de estudo e trabalho no desenvolvimento de *software*. No total, foram obtidas 50 respostas, fornecendo uma base significativa de informações para análise.

Figura 3 - Captura de tela de uma parte do questionário



Fonte: Autoria própria (2024)

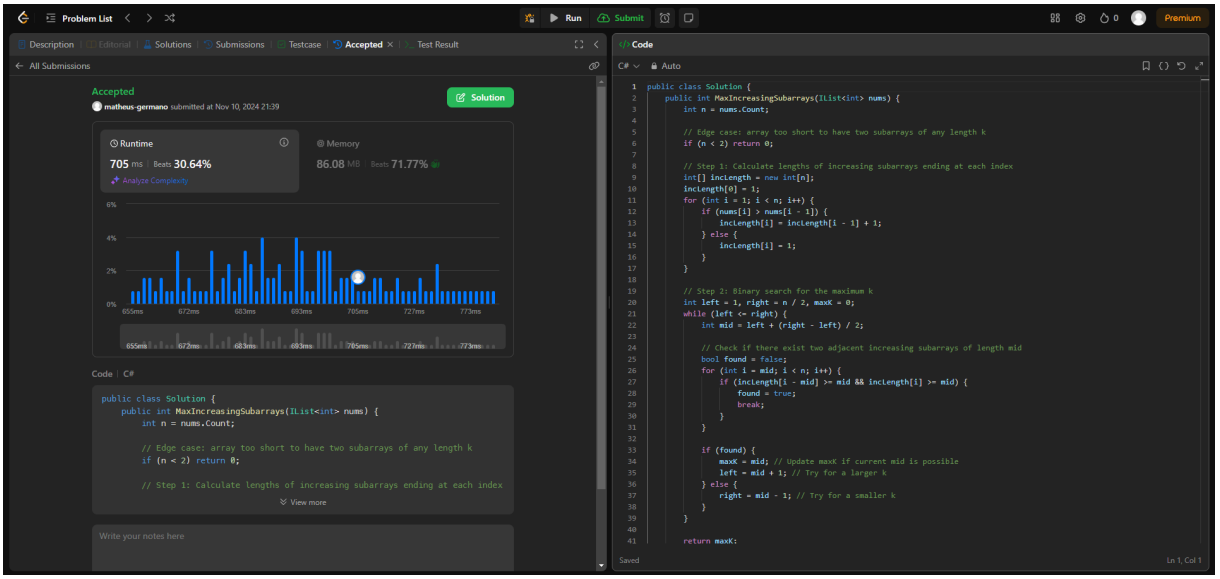
5.2.1.1 Participantes

A população-alvo do questionário é composta por desenvolvedores estudantes ou que já atuam na área. A amostra utilizada foi selecionada através de divulgação do questionário em grupos diversos que contém estudantes e desenvolvedores que já estão no mercado de trabalho.

5.2.2 Testes de algoritmos

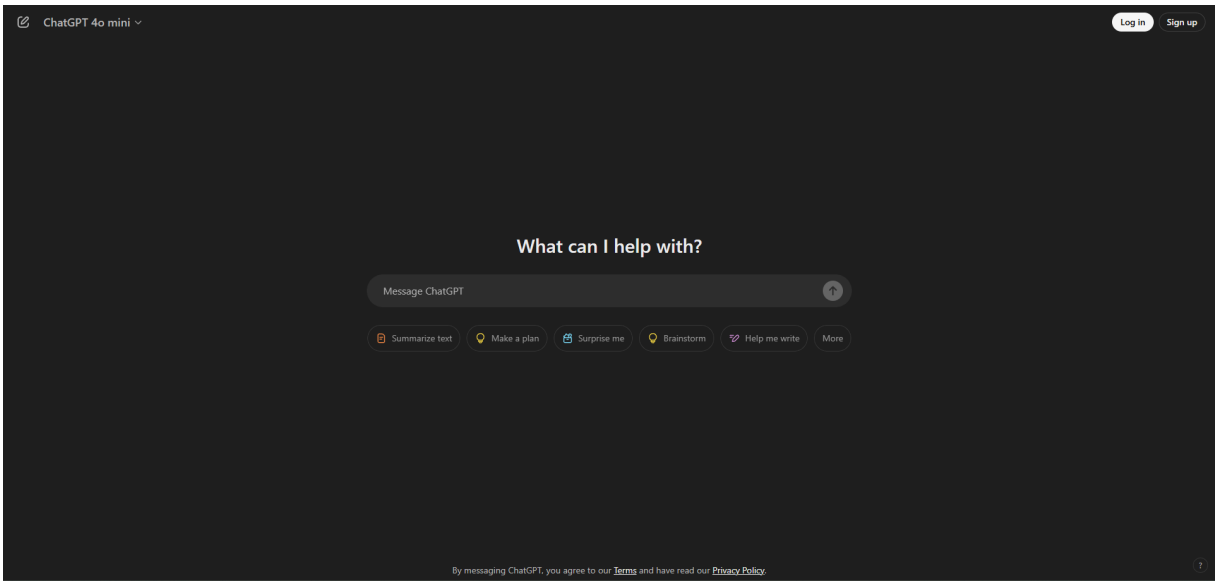
Para avaliar a eficiência e a qualidade das ferramentas de IA, foram realizados testes de algoritmos utilizando as plataformas *Leet Code* e *Chat GPT*. Um exemplo de submissão pode ser visto na figura 4. A figura 5 mostra a página inicial de interação com o *chatbot.*

Figura 4 - Captura de tela da submissão do desafio no *Leet Code*



Fonte: Autoria própria (2024)

Figura 5 - Captura de tela da página inicial do *Chat GPT*



Fonte: Autoria própria (2024)

5.2.2.1 ***Execução dos testes***

Foram selecionados três desafios distintos com níveis de dificuldade categorizados como fácil, médio e difícil, a serem resolvidos em três linguagens de programação diferentes: C#, C++ e Python. A escolha dessas linguagens foi fundamentada em sua presença no top 5 do ranking TIOBE de 2024. Para padronizar os testes, foi elaborado um prompt padrão que

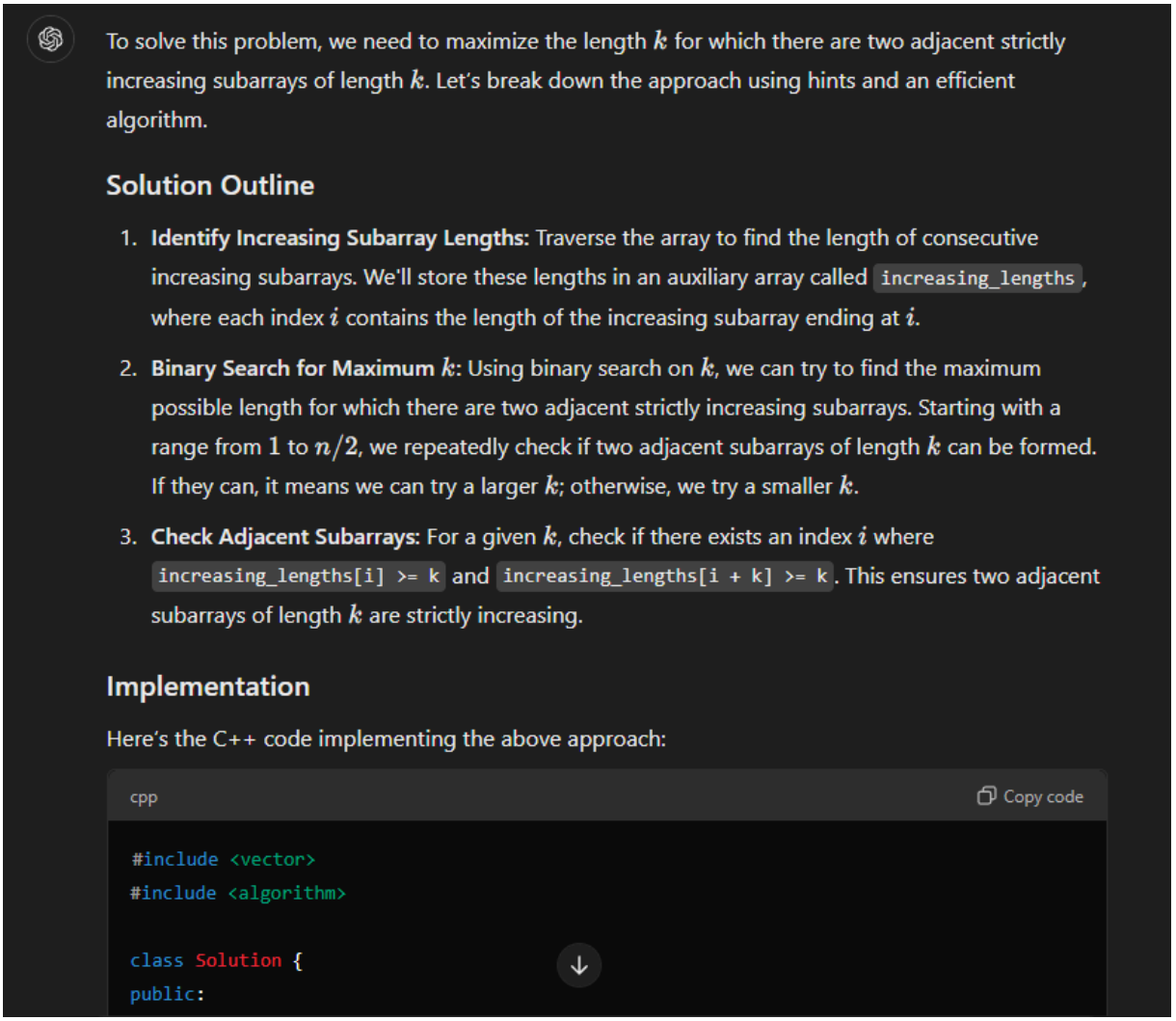
seria enviado ao *Chat GPT* no início de cada tentativa de resolução do desafio, contendo todas as instruções descritas na plataforma *Leet Code*.

O prompt utilizado foi estruturado da seguinte maneira:

> "*I am trying to solve an algorithm test about {challenge_title}. The description of it is:*
> *{test description}*
>
> *I also have {n} hints to help find the solution:*
> *{hints}*
>
> *Code to implement:*
> *{code}*"

Com base nesse *prompt*, o desafio era apresentado ao *chatbot*, que deveria responder com uma solução na linguagem de programação especificada. A solução seria submetida à plataforma para avaliação. Caso o resultado fosse negativo (falha), o erro era descrito ao *chatbot*, que deveria propor uma correção para uma nova tentativa. Esse ciclo de iterações era repetido até que o desafio fosse solucionado com sucesso ou que o número de tentativas chegasse ao limite de 15 execuções. A figura 6 mostra o resultado de uma das interações feitas no processo de resolução dos algoritmos.

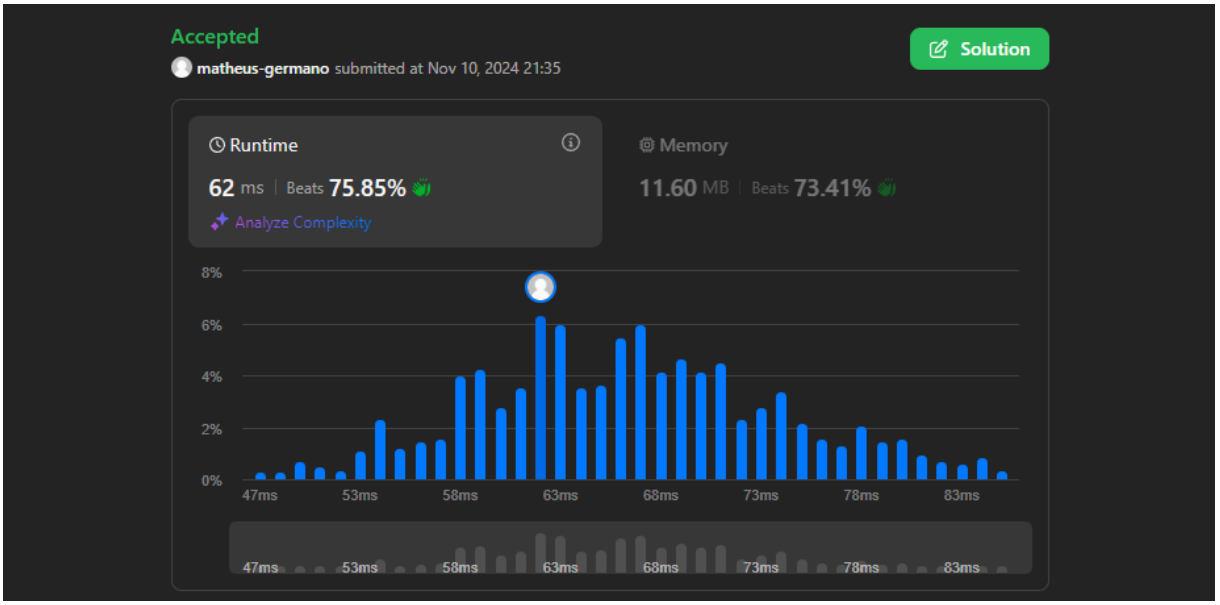
Figura 6 - Captura de tela da resposta do *Chat GPT* sobre a solicitação de um desafio



Fonte: Autoria própria (2024)

Nos cenários com resultados positivos e corretos, o desempenho do *Chat GPT* foi avaliado por meio de métricas como tempo de execução, alocação de memória e suas respectivas complexidades, conforme disponibilizadas pela plataforma *Leet Code*. Essa abordagem permitiu validar se o *chatbot* produzia algoritmos com qualidade e eficiência, garantindo critérios objetivos para análise. A figura 7 apresenta o resultado de uma das submissões feitas na plataforma.

Figura 7 - Captura de tela das métricas da submissão de um desafio no *Leet Code*



Fonte: Autoria própria (2024)

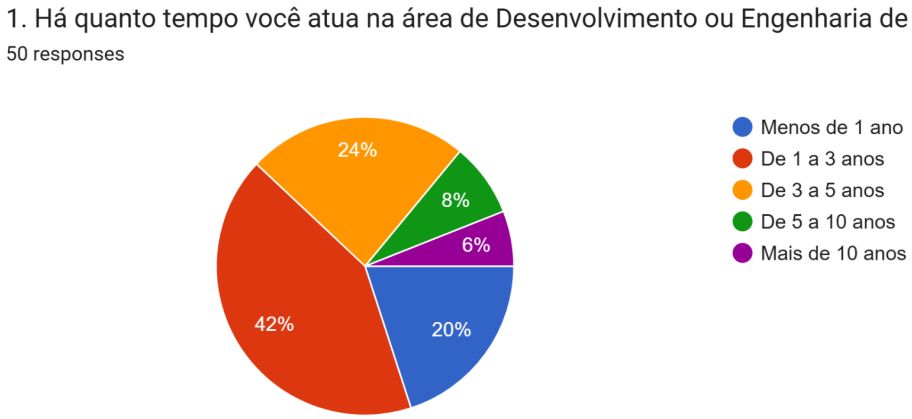
5.3 Análise de dados

Após a execução de todas as etapas propostas na metodologia, consolidou-se uma base sólida de conhecimento e dados para o desenvolvimento do projeto. Essa base foi composta por informações vindas de artigos de alta relevância sobre o tema, respostas obtidas junto ao público-alvo por meio do questionário aplicado, além dos testes de algoritmos realizados utilizando as plataformas *Leet Code* e *Chat GPT*.

5.3.1 Resultado a partir do questionário

Foram obtidas 50 respostas no questionário proposto ao público-alvo. Os participantes apresentaram uma ampla variação de experiência profissional, desde indivíduos com menos de um ano de atuação na área (20%) até profissionais com mais de 5 anos de experiência (14%). Esse dado permitiu observar as percepções de impacto da IA em diferentes estágios da carreira. No gráfico 1 pode ser visto o resultado das respostas de uma das questões presentes no formulário.

Gráfico 1 - Captura de tela do resultado da questão 1 do questionário



Fonte: Autoria própria (2024)

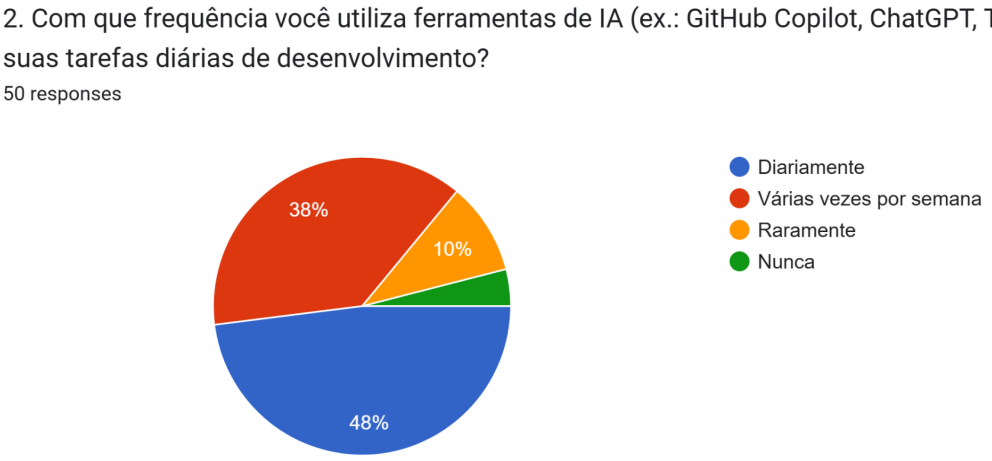
5.3.1.1 Frequência e finalidades de uso

A maioria dos respondentes indicou utilizar ferramentas de IA frequentemente, com 48% relatando uso diário e 38% afirmando utilizar várias vezes por semana. As finalidades principais incluíram:

- Escrita de código: mencionada como a principal aplicação.
- Refatoração e otimização de código: usada frequentemente para melhorar a qualidade do código existente.
- Geração de documentação: destacada como uma atividade beneficiada pela automação proporcionada pela IA.
- Depuração de código e compreensão de sistemas legados: tarefas citadas por vários profissionais como mais acessíveis com a ajuda da IA.

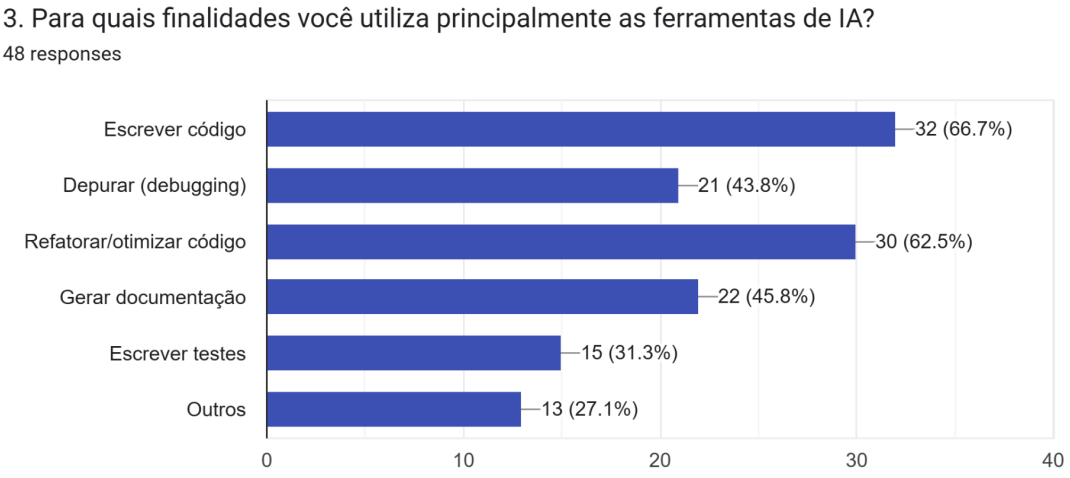
Apenas 2 pessoas responderam que nunca utilizam ferramentas de IA para auxílio. Os resultados relacionados a esses dados podem ser observados nos gráficos 2 e 3.

Gráfico 2 - Captura de tela do resultado da questão 2 do questionário



Fonte: Autoria própria (2024)

Gráfico 3 - Captura de tela do resultado da questão 3 do questionário



Fonte: Autoria própria (2024)

5.3.1.2 Impacto na produtividade e economia de tempo

Os dados indicam que, aproximadamente, 92% dos respondentes relataram aumento significativo na produtividade após a adoção de ferramentas de IA, com 54,2% afirmando que "aumentou muito". Em relação à economia de tempo, 60,4% estimaram um ganho de 1 a 3 horas por dia, enquanto 16,7% indicaram economias de 3 a 5 horas.

5.3.1.3 Melhoria na compreensão e resolução de problemas

Grande parte dos participantes (81,2%) concordou que as ferramentas de IA melhoraram sua compreensão do código e a capacidade de resolver problemas complexos. No entanto, 16,7% destacaram que o impacto foi mínimo ou inexistente.

5.3.1.4 Dependência e riscos percebidos

Embora a maioria utilize as ferramentas como apoio, aproximadamente 16% dos respondentes relataram uma dependência ocasional da IA e dificuldades em resolver problemas sem seu auxílio. Além disso, problemas de segurança e desempenho no código gerado foram mencionados por 60% dos participantes, o que demonstra uma necessidade de atenção a esses aspectos.

5.3.1.5 Avaliação da qualidade do código

Quanto à qualidade do código gerado, 60% avaliaram como mediana, e 30% indicaram alta qualidade. A legibilidade e a facilidade de manutenção foram mencionadas como áreas a serem aprimoradas.

5.3.1.6 Recomendação e impactos de longo prazo

Quase todos os participantes recomendaram o uso de ferramentas de IA para o desenvolvimento de *software*. Além disso, 62% acreditam que o impacto a longo prazo será positivo, embora 8% tenham expressado preocupações sobre possíveis efeitos negativos no desenvolvimento de habilidades manuais de programação.

5.3.1.7 Vantagens e desafios

Entre as vantagens citadas, destacaram-se:

- Agilidade e praticidade: os participantes enfatizaram o ganho de eficiência em tarefas como escrita de código e correção de bugs.
- Facilidade na resolução de problemas: a IA foi mencionada como um apoio significativo para tarefas complexas.

Por outro lado, os desafios incluem:

- Dependência excessiva das ferramentas: mencionada como um risco em potencial.
- Dificuldades em depurar código gerado automaticamente: apontadas como um problema em situações específicas.

5.3.2 Resultados obtidos pelos testes de algoritmos

O *Chat GPT* demonstrou uma performance satisfatória para problemas de dificuldade fácil e média, conseguindo solucionar os desafios dentro do limite estabelecido de 15 tentativas. No entanto, em problemas classificados como difíceis, o modelo não foi capaz de apresentar uma solução válida em nenhuma das linguagens testadas, mesmo após múltiplas revisões e ajustes baseados no feedback fornecido. Os resultados dos testes de algoritmos podem ser vistos no quadro 1.

Quadro 1 - Quadro de resultados dos testes de algoritmos realizados com o *Chat GPT*

Nível de dificuldade	Linguagem de programação	Número total de submissões	Número de tentativas	Resultado	Complexidade do algoritmo em tempo	Complexidade do algoritmo em espaço
Fácil	C#	2	1	Sucesso	O(N*K)	O(1)
Fácil	C++	53	3	Sucesso	O(N)	O(N)
Fácil	Python	36	1	Sucesso	O(N*K)	O(1)
Médio	C#	3	1	Sucesso	O(N log N)	O(N)
Médio	C++	57	1	Sucesso	O(N log N)	O(N)
Médio	Python	43	11	Sucesso	O(N log N)	O(N)
Difícil	C#	0	15	Falha	N/A	N/A
Difícil	C++	18	15	Falha	N/A	N/A
Difícil	Python	8	15	Falha	N/A	N/A

Fonte: Autoria própria (2024)

5.3.2.1 Problemas fáceis

Para os desafios classificados como fáceis, o *Chat GPT* apresentou um desempenho consistente em todas as linguagens testadas:

- Em C#, a solução foi alcançada em apenas 1 tentativa.
- Em C++, o modelo necessitou de 3 tentativas para submeter uma resposta correta.
- Em Python, assim como em C#, a solução foi apresentada na primeira tentativa.

Esse desempenho sugere que o modelo é capaz de interpretar problemas simples e propor soluções com algoritmos de maneira eficiente.

5.3.2.2 Problemas médios

Nos problemas de dificuldade média, observou-se maior variação no desempenho, porém sem muita relevância:

- Em C# e C++, o modelo solucionou as questões na primeira tentativa, apresentando algoritmos com complexidade O(N log N) no pior caso.
- Em Python, o *Chat GPT* precisou de 11 tentativas para atingir o resultado esperado, indicando que o modelo pode apresentar inconsistências dependendo da linguagem e da interpretação de requisitos específicos.

Apesar das variações, o modelo demonstrou capacidade de resolver problemas mais desafiadores, embora exigisse mais ajustes em determinadas situações.

5.3.2.3 Problemas difíceis

Em contraste, para os desafios de nível difícil, o *Chat GPT* não conseguiu apresentar uma solução válida em nenhuma das linguagens testadas, mesmo após 15 tentativas.

Em C#, C++ e Python, apesar das múltiplas revisões no código e ajustes baseados no feedback das submissões, o modelo não foi capaz de atender aos requisitos das questões.

6 DISCUSSÃO E TRABALHOS FUTUROS

Ao longo da pesquisa e realização dos testes, foi possível observar certos comportamentos, impactos positivos e impactos negativos da utilização de ferramentas de IA no processo de desenvolvimento de *software*, seja profissionalmente ou como estudante.

Com base nos resultados do formulário aplicado ao público-alvo, podemos concluir que ferramentas de IA estão cada vez mais presentes no dia a dia dos desenvolvedores e tendem a se tornar ainda mais essenciais. Em termos de produtividade, por exemplo, uma adoção de apenas 25% dessas ferramentas já proporciona um aumento aproximado de 2,1% no desempenho individual. Embora esse valor possa parecer pequeno, quando escalado para equipes, times e setores inteiros, o impacto pode ser significativamente mais expressivo (DORA, 2024).

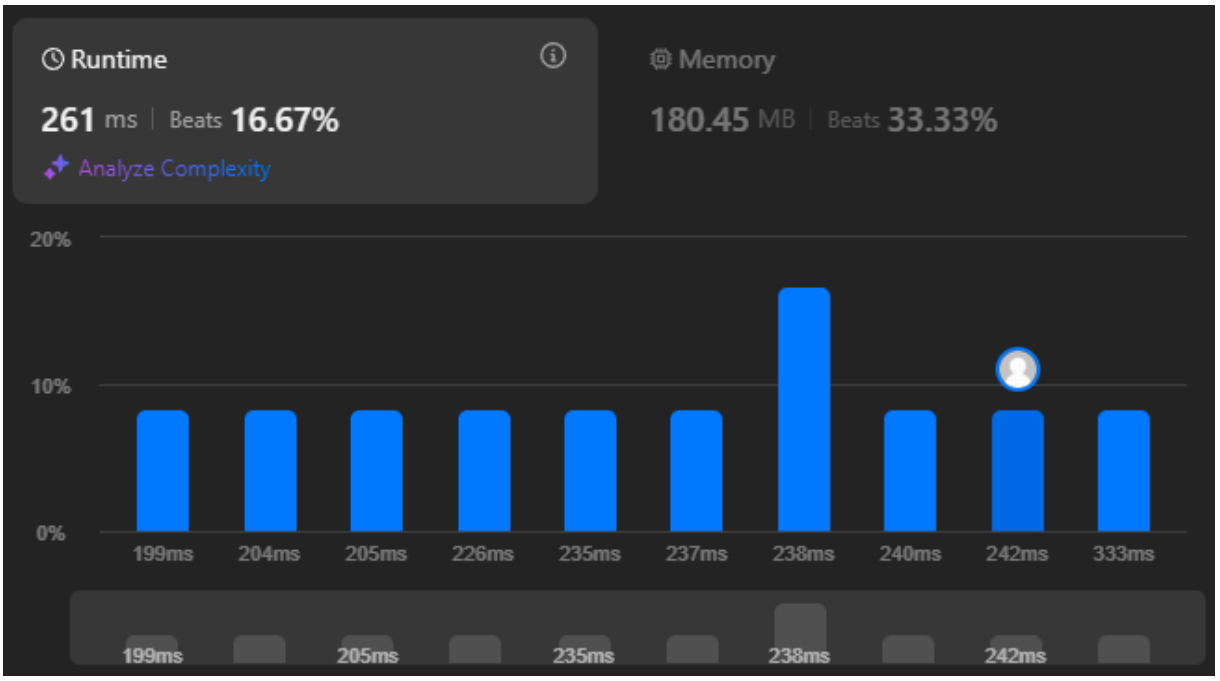
Além disso, observou-se que as ferramentas de IA foram amplamente aceitas nos processos de desenvolvimento de *software*, com feedback positivo da comunidade. Elas são utilizadas para uma variedade de finalidades e frequentemente recomendadas entre colegas. Paralelamente, surgem novos *fine-tunings* de modelos que aprimoram a experiência do usuário e os resultados alcançados. Um exemplo notável é o Claude, que vem ganhando popularidade. “Claude é um assistente de IA de última geração criado pela Anthropic e treinado para ser seguro, preciso e protegido para ajudar você a fazer seu melhor trabalho.” (Anthropic, 2024).

De um ponto de vista mais acadêmico, de acordo com Styve, Virkki e Naeem (2024), as ferramentas de IA vieram para ficar e os alunos devem aprender a usá-las de forma responsável em seu aprendizado e entrega de tarefas. Porém, nessas mesmas situações, os estudantes buscam por respostas e, muitas vezes, utilizam a primeira que encontram, sem analisar o código, o que ele faz, ou por que ele funciona, não questionando sua qualidade e seu impacto nos projetos. Essa prática geralmente leva à falta de compreensão e dificulta a capacidade do aluno de aprender e aumentou consideravelmente após a introdução de ferramentas de IA em seus processos.

Notavelmente, o *Chat GPT* apresentou uma significativa limitação nos testes de algoritmos de nível de complexidade difícil, não conseguindo concluir com sucesso o desafio proposto em nenhuma das três linguagens avaliadas. Mesmo nos desafios de níveis fácil e médio, embora tenham sido concluídos com êxito, os resultados demonstraram uma qualidade apenas mediana. É possível observar que as submissões de outros usuários na plataforma apresentaram desempenho superior, evidenciando uma margem de melhoria para as soluções

geradas pelo *chatbot*. Na figura 11 é exemplificada a relação entre a eficiência da submissão de um dos testes de algoritmo com as de outros usuários.

Figura 8 - Captura de tela das métricas da submissão de um desafio médio no *Leet Code*



Fonte: Autoria própria (2024)

Um fato incontestável é que os modelos dessas ferramentas são treinados a partir de textos e códigos-fonte provenientes de repositórios públicos disponíveis na internet. Entretanto, uma de suas principais limitações é a precisão das saídas geradas, o que pode resultar em códigos e explicações imprecisos (Styve; Virkki; Naeem, 2024).

Essa limitação é aprofundada no artigo da *Apple*, "*GSM-Symbolic: Understanding the Limitations of Mathematical Reasoning in Large Language Models*", que, embora baseado em exemplos matemáticos, possui implicações claras para a área de desenvolvimento de *software*. O estudo aponta que, apesar dos LLMs demonstrarem capacidades notáveis em diversos domínios, como processamento de linguagem natural, resposta a perguntas e tarefas criativas, eles apresentam inconsistências notáveis ao responder diferentes variações da mesma pergunta. Além disso, identificou-se uma fragilidade no raciocínio matemático desses modelos, cujo desempenho piora significativamente à medida que aumenta a complexidade ou o número de cláusulas em uma questão. Conforme observado: "Pequenas alterações nos tokens de entrada podem alterar drasticamente as saídas do modelo, indicando um forte viés

de token e sugerindo que esses modelos são altamente sensíveis e frágeis" (Mirzadeh et al., 2024, tradução própria).

O artigo também fornece hipóteses sobre as causas dessa fragilidade no raciocínio dos modelos. Segundo os autores: "Nossa hipótese é que esse declínio se deve ao fato de que os LLMs atuais não são capazes de raciocínio lógico genuíno; em vez disso, eles tentam replicar as etapas de raciocínio observadas em seus dados de treinamento." Além disso, argumentam que: "A literatura sugere que o processo de raciocínio em LLMs é uma correspondência de padrões probabilísticos em vez de raciocínio formal" (Mirzadeh et al., 2024, tradução própria). Esses argumentos podem justificar o porquê houveram tantas tentativas falhas nos testes de algoritmos através do *Chat GPT*.

Diante desse cenário e das pesquisas realizadas, torna-se evidente a necessidade de adotar um pensamento crítico em relação às soluções encontradas, sejam elas geradas por ferramentas de IA ou oriundas de fóruns amplamente conhecidos, como o *Stack Overflow*. É crucial que os desenvolvedores consigam avaliar com discernimento boas e más práticas de codificação e implementação, garantindo que seus projetos mantenham padrões elevados de qualidade e escalabilidade.

Espera-se que essa pesquisa contribua para a compreensão dos impactos da IA no desenvolvimento de *software*, auxiliando os profissionais da área na tomada de decisões e no aproveitamento dos benefícios proporcionados por essas tecnologias em constante evolução.

Em etapas futuras, caso haja oportunidade, o questionário poderá ser divulgado para uma amostra mais ampla e diversificada, abrangendo uma maior variação nos anos de experiência dos participantes. Essa abordagem permitirá a obtenção de resultados mais representativos e detalhados. Além disso, os testes de algoritmos realizados no *Leet Code* poderão ser complementados com experimentos controlados envolvendo diferentes grupos de profissionais e estudantes. Nesse contexto, seriam disponibilizadas ferramentas de IA para alguns grupos, enquanto o acesso seria restrito para outros, possibilitando uma análise comparativa mais aprofundada sobre os impactos do uso dessas tecnologias no desenvolvimento de *software*.

REFERÊNCIAS

ABES - Associação Brasileira das Empresas de Software. **Mercado brasileiro de *software* 2024:** panorama e tendências. São Paulo: ABES, 2024. 117 p. Disponível em: https://abes.com.br. Acesso em: 25 nov. 2024.

AMAZON WEB SERVICES. ***What is generative AI?*** Disponível em: https://aws.amazon.com/what-is/generative-ai/. Acesso em: 25 nov. 2024.

AMMAR, Hany; HAMDI, Mohamed Salah; ABDELMOEZ, Walid. ***Software Engineering Using Artificial Intelligence Techniques:*** *Current State and Open Problems*. Research Gate, 2012. Disponível em: https://aitskadapa.ac.in/e-books/AI&DS/SOFTWARE%20ENGINEERING%20FOR%20AI/Software_Engineering_Using_Artificial_Intelligence.pdf. Acesso em: 4 jun. 2023.

ANTHROPIC. **Claude AI**. Disponível em: https://claude.ai/. Acesso em: 25 nov. 2024.

BHARGAVA, Aditya Y. ***Grokking Algorithms****: An illustrated guide for programmers and other curious people*. 1. ed. Shelter Island: Manning Publications, 2016. Disponível em: https://www.google.com.br/books/edition/Grokking_Algorithms/yzkzEAAAQBAJ. Acesso em: 25 nov. 2024.

BIG O CHEAT SHEET. **Big O Cheat Sheet**. Disponível em: https://www.bigocheatsheet.com/. Acesso em: 25 nov. 2024.

CHEN, Lijia; CHEN, Pingping; LIN, Zhijian. ***Artificial Intelligence in Education***. IEEE Access, 2020. DOI https://doi.org/10.1109/ACCESS.2020.2988510. Disponível em: https://ieeexplore.ieee.org/abstract/document/9069875. Acesso em: 4 jun. 2023.

CHEN, Lingjiao; ZAHARIA, Matei; ZOU, James. ***How Is ChatGPT's Behavior Changing over Time?***. 31 out. 2023. Disponível em: https://arxiv.org/pdf/2307.09009.pdf. Acesso em: 17 nov. 2023.

CUI, Kevin Zheyuan. et al. ***The Effects of Generative AI on High Skilled Work:*** *Evidence from Three Field Experiments with Software Developers.* 2024. Disponível em: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=4945566. Acesso em: 25 nov. 2024.

DAMIOLI, Giacomo; ROY, Vincent Van; VERTESY, Daniel. ***The impact of artificial intelligence on labor productivity***. SpringerLink**,** 2021. Disponível em: https://link.springer.com/article/10.1007/s40821-020-00172-8. Acesso em: 10 jun. 2023.

DAVIS, Wes. ***Stack Overflow cuts 28% of its staff***. The Verge, 16 out. 2023. Disponível em: https://www.theverge.com/2023/10/16/23919004/stack-overflow-layoff-ai-profitability. Acesso em: 16 nov. 2023.

DORA (DevOps Research and Assessment). ***Accelerate:*** *State of DevOps Report*. 2024. 117 p. Disponível em: https://dora.dev. Acesso em: 25 nov. 2024.

KABIR, Samia. et al.. ***Who Answers It Better? An In-Depth Analysis of ChatGPT and Stack Overflow Answers to Software Engineering questions***. 10 ago. 2023. Disponível em: https://arxiv.org/pdf/2308.02312.pdf. Acesso em: 17 nov. 2023.

MAKRIDAKIS, Spyros. ***The forthcoming Artificial Intelligence (AI) revolution:*** *Its impact on society and firms*. Futures**,** 2017. Disponível em: https://www.sciencedirect.com/science/article/pii/S0016328717300046?casa_token=GaqRSg6qF5wAAAAA:MSv7n5Qu46aY1wwO3E6e1SjFLvryQ3xGNXUr4pTcrusNE3WXpyekchqr5tkjINj0NLgilElEwVg7. Acesso em: 10 jun. 2023.

MARTIN, Robert C. ***Clean Code:*** *A Handbook of Agile Software Craftsmanship*. Upper Saddle River: Prentice Hall, 2008. 464 p.

MARTIN, Robert C. ***Clean Architecture****: A Craftsman's Guide to Software Structure and Design*. Upper Saddle River: Prentice Hall, 2017. 432 p.

MIRZADEH, Iman. et al. ***Understanding the Limitations of Mathematical Reasoning in Large Language Models***. ArXiv, [s.l.], v. 2410.05229, 2024. Disponível em: https://arxiv.org/abs/2410.05229. Acesso em: 25 nov. 2024.

Reuters. **Meta projeta forte aceleração em custo de IA após balanço bater projeções; ações caem**. InfoMoney, 30 out. 2024. Disponível em: https://www.infomoney.com.br/mercados/meta-projeta-forte-aceleracao-em-custo-de-ia-apos-balanco-bater-projecoes-acoes-caem/. Acesso em: 25 nov. 2024.

Reuters. **Meta e Microsoft aumentam gastos com IA e preocupam *Wall Street*.** CNN Brasil, 31 out. 2024. Disponível em: https://www.cnnbrasil.com.br/economia/negocios/meta-e-microsoft-aumentam-gastos-com-ia-e-preocupam-wall-street/. Acesso em: 25 nov. 2024.

RUSSELL, Stuart; NORVIG, Peter. **Inteligência artificial**. 4. ed. Rio de Janeiro: LTC, 2021.

SEDEEGEWICK, Robert; WAYNE, Kevin. ***Algorithms***. 4. ed. Princeton: Princeton University, 2011. Disponível em: https://algs4.cs.princeton.edu/home/. Acesso em: 25 nov. 2024.

STYVE, Arne; VIRKKI, Outi T.; NAEEM, Usman. ***Developing Critical Thinking Practices Interwoven with Generative AI Usage in an Introductory Programming Course***. IEEE Global Engineering Education Conference (EDUCON), 2024. Disponível em: https://doi.org/10.1109/EDUCON60312.2024.10578746. Acesso em: 25 nov. 2024.

THE VERGE. ***Google's new code AI tools showcase the future of programming***. The Verge, 29 out. 2024. Disponível em: https://www.theverge.com/2024/10/29/24282757/google-new-code-generated-ai-q3-2024. Acesso em: 25 nov. 2024.

TIOBE Software. ***TIOBE Index***. Disponível em: https://www.tiobe.com/tiobe-index/. Acesso em: 25 nov. 2024.

ZHANG, Daniel. et al. ***The AI Index 2022 Annual Report***. AI Index Steering Committee, Stanford Institute for Human-Centered AI, Stanford University, 2022. Disponível em: https://aiindex.stanford.edu/wp-content/uploads/2022/03/2022-AI-Index-Report_Master.pdf. Acesso em: 17 nov. 2023.