UNIVERSIDADE DO VALE DO RIO DOS SINOS
UNIDADE ACADÊMICA DE GRADUAÇÃO
CURSO DE BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

GUILHERME CLOSS FRAGA

INTELIGÊNCIA ARTIFICIAL E *VIBE CODING* NO PROCESSO DE DESENVOLVIMENTO DE *SOFTWARE*:
Um estudo comparativo entre desenvolvedores com e sem o uso de ferramentas de IA

São Leopoldo
2025

GUILHERME CLOSS FRAGA

INTELIGÊNCIA ARTIFICIAL E *VIBE CODING* NO PROCESSO DE DESENVOLVIMENTO DE *SOFTWARE*:
Um estudo comparativo entre desenvolvedores com e sem o uso de ferramentas de IA

Artigo apresentado como requisito parcial para obtenção do título de Bacharel em Ciência da Computação, pelo Curso de Ciência da Computação da Universidade do Vale do Rio dos Sinos (UNISINOS)

Orientador(a): Prof.ª Dra. Rosemary Francisco

São Leopoldo
2025

INTELIGÊNCIA ARTIFICIAL E *VIBE CODING* NO PROCESSO DE DESENVOLVIMENTO DE *SOFTWARE*: Um estudo comparativo entre desenvolvedores com e sem o uso de ferramentas de IA

Guilherme Closs Fraga[1]
Rosemary Francisco[2]

Resumo: Lorem ipsum.

Palavras-chave: lorem ipsum.

Abstract: Lorem ipsum.

Keywords: lorem ipsum.

1 INTRODUÇÃO

A Inteligência Artificial (IA) é considerada uma das tecnologias mais transformadoras da atualidade, impulsionando mudanças em diversos setores e despertando interesse tanto no meio acadêmico quanto no mercado. Trata-se do uso de algoritmos e modelos computacionais capazes de realizar tarefas tradicionalmente dependentes da cognição humana, como análise de dados, raciocínio lógico e tomada de decisão (CÂNDIDO, 2022; AGUIAR, 2023). Segundo relatório da McKinsey (2025), a utilização de ferramentas de IA generativa pode aumentar a produtividade de desenvolvedores em tarefas de codificação, permitindo a execução de certas atividades até duas vezes mais rápido que processos totalmente manuais. Apesar desses avanços, a adoção dessas tecnologias levanta questões sobre dependência excessiva, erosão de habilidades humanas e confiabilidade das soluções geradas.

No contexto do desenvolvimento de *software*, a presença da IA tem se tornado particularmente evidente. Ferramentas que combinam aprendizado de máquina e processamento de linguagem natural auxiliam desenvolvedores em diversas etapas do ciclo de vida do *software*, desde a escrita de código até a detecção de falhas e a otimização de desempenho (COUTINHO et al., 2024). Um exemplo é o *GitHub Copilot*, lançado em 2021 pela *GitHub* em parceria com a *OpenAI*, que utiliza modelos de IA Generativa (também chamadas de IA Gen) para sugerir trechos de código em tempo real (GITHUB, 2021). Da mesma forma, agentes de codificação baseados em IA, alinhados ao conceito de vibe coding, permitem que o desenvolvedor interaja de forma mais conversacional com a ferramenta, gerando código de maneira assistida, mas ainda mantendo controle sobre decisões críticas (CACM, 2025). Essas práticas têm sido discutidas como um avanço na programação assistida, trazendo benefícios de produtividade e colaboração

[1]Graduando em Ciência da Computação pela Unisinos. Email: guifraga8@edu.unisinos.br
[2]Doutora em Administração. Email: rosemaryf@unisinos.br

entre humano e máquina.

Esses recursos, ao mesmo tempo em que oferecem praticidade e aumentam a produtividade, também levantam debates relevantes. Há questionamentos sobre até que ponto a utilização de IA generativa pode impactar o desenvolvimento de habilidades de raciocínio lógico e resolução de problemas, uma vez que o programador pode se tornar excessivamente dependente da ferramenta (VERASTEGUI; RUIZ; VILLANUEVA, 2025). Além disso, o código gerado pode apresentar erros lógicos, violações de boas práticas e vulnerabilidades de segurança, exigindo revisão crítica por parte do profissional (DEVOPS, 2024). Estudos experimentais indicam resultados variados: enquanto tarefas simples podem ser concluídas significativamente mais rápido com IA, contextos complexos podem exigir maior tempo de revisão e adaptação do código (PENG et al., 2023; METR, 2025).

Por outro lado, pesquisas recentes indicam que a adoção de IA no desenvolvimento de *software* pode representar ganhos significativos em termos de produtividade e redução do tempo necessário para a entrega de soluções. Um estudo piloto com profissionais de *software* demonstrou percepções majoritariamente positivas quanto ao impacto desses recursos na produtividade individual (COUTINHO et al., 2024). Essa realidade torna essencial a investigação de como essas ferramentas impactam o trabalho de desenvolvedores em contextos práticos. Assim, mais do que um recurso de conveniência, a IA se apresenta como um instrumento que pode redefinir processos e metodologias de engenharia de *software*.

Diante desse cenário, torna-se essencial investigar como a IA generativa influencia o desempenho de desenvolvedores. Este trabalho propõe um estudo comparativo entre dois grupos de participantes: um com acesso a ferramentas de IA Generativa e outro sem acesso a esses recursos, aplicando um desafio técnico de programação. O objetivo é avaliar variáveis como eficiência, tempo de resolução e capacidade para solucionar os problemas propostos, fornecendo uma investigação sobre vantagens e limitações do uso dessas tecnologias.

Assim, o objetivo geral desta pesquisa é analisar o uso de ferramentas de IA Generativa no Desenvolvimento de *Software* por meio de um desafio técnico comparativo entre desenvolvedores com e sem acesso a essas ferramentas. A questão de pesquisa que orienta este estudo é: de que maneira o uso de ferramentas de IA Generativa contribui ou afeta o desempenho de desenvolvedores em atividades práticas de programação? Destacam-se também os objetivos específicos necessários para atingir a conclusão deste trabalho, sendo eles:

a) desenvolver uma plataforma *web* simples para disponibilização do desafio técnico e coleta de dados de desempenho dos participantes;

b) aplicar o desafio técnico a dois grupos de desenvolvedores, sendo um autorizado a utilizar ferramentas de IA e outro restrito ao uso dessas tecnologias;

c) comparar os resultados obtidos pelos grupos em relação ao tempo de conclusão, qualidade do código entregue e abordagem adotada na resolução do problema;

d) analisar as vantagens, limitações e possíveis implicações do uso de ferramentas de IA no

processo de desenvolvimento de *software*.

As seções seguintes deste documento estão dispostas da seguinte forma: a segunda seção contempla a fundamentação teórica do trabalho, que discorre sobre o desenvolvimento de *software* e IA Generativa aplicada ao desenvolvimento de *software* propriamente. A terceira seção apresenta os trabalhos relacionados a esta pesquisa e uma breve comparação entre os mesmos. Na quarta seção apresenta-se a metodologia do trabalho, descrevendo como foi conduzida todas as etapas e elaborações da pesquisa. Na quinta seção está a proposta de modelo do trabalho, apresentando brevemente a arquitetura da plataforma, tecnologias utilizadas e seu fluxo de uso do lado dos participantes. Na sexta seção apresenta-se os resultados e discussões após coleta de dados por meio do experimento aplicado como um todo.

Por fim, ao fechamento do trabalho na última seção, encontra-se as conclusões e considerações finais levantadas com o experimento, assim como apontamentos para trabalhos futuros. Espera-se que os resultados forneçam evidências consistentes para compreender o papel da IA como aliada no processo de desenvolvimento de *software*, oferencedo resultados para pesquisadores, profissionais e instituições interessadas em compreender e orientar o uso dessas ferramentas emergentes.

2 FUNDAMENTAÇÃO TEÓRICA

Com base na temática que o presente trabalho visa abordar, se faz necessário discorrer sobre dois tópicos fundamentais para a construção e consolidação do mesmo, sendo eles: Desenvolvimento de *Software* e IA Generativa aplicada ao desenvolvimento de *software*. Nas seções seguintes, os conceitos são explorados e detalhados a fim de estabelecer os fundamentos teóricos necessários para compor este trabalho.

2.1 Desenvolvimento de *Software*

Um *software* refere-se a todos os componentes não físicos de um computador, redes de computadores ou dispositivos móveis. Em linhas gerais, refere-se aos programas e aplicações, como o próprio sistema operacional, que fazem aquele aparelho (ou ferramenta) operar de acordo com o que o usuário necessita. Isto é, *software* se trata de uma coleção de instruções, dados ou programas utilizados para operar computadores e executar determinadas tarefas (COUTINHO; BEZERRA, 2021).

Em outras palavras, pode-se dizer que é um termo genérico que se refere à aplicações, *scripts* e programas executáveis em um determinado dispositivo. Assim, o diferencia de *hardware*, que descreve os aspectos físicos de um aparelho. Sendo assim, pode-se estabelecer o *software* como um componente variável de um computador, enquanto o *hardware* é constante (SAKURAI; ZUCHI, 2018)

De acordo com Maynard (2015), existem basicamente dois tipos de *software*, sendo eles:

software de aplicativo e *software* de sistema. Um aplicativo seria aquele que atende uma necessidade específica ou executa determinadas tarefas. E, segundo Schwab (2019), *software* de sistema seria a parte essencial para que o *hardware* de um computador opere, servindo como a plataforma para a execução de aplicativos. Ademais, pode-se mencionar outros tipos de *software*, como os de programação, que são ferramentas necessárias para desenvolvedores, *middlewares*, responsáveis por atuarem entre o *software* do sistema e aplicativos e até mesmo *drivers*, que são responsáveis por operarem dispositivos e periféricos de computadores.

Por fim, a sincronia entre diferentes tipos de *software* e *hardware*, é peça fundamental para o funcionamento eficiente de sistemas computacionais. Enquanto um aplicativo permite que seus usuários realizem determinadas tarefas, um sistema assegura que o *hardware* opere corretamente, dando suporte para diferentes aplicações. Com a constante crescente de complexidade em sistemas modernos, a presença de um *middleware* eficaz se torna indispensável para assegurar que diferentes *softwares* funcionem em harmonia. Da mesma forma que, *drivers* são indispensáveis para comunicação entre o sistema e periféricos, o que amplia as funcionalidades de um computador, resultando na melhoria da experiência do usuário.

Os *softwares*, diferentemente do hardware, não são tangíveis e representam os componentes lógicos responsáveis pelo funcionamento de um computador ou dispositivo móvel. De acordo com Cândido (2022), o sistema operacional constitui o software fundamental, sendo exemplos mais conhecidos o *Windows*, *macOS* e *Linux*, que variam em suas interfaces de usuário. Além dele, outros programas complementam as funções básicas, como o *Microsoft Office* e o *MediaPlayer* (NETO, 2019), além de *softwares* utilitários, como antivírus, ferramentas de *backup* e manutenção de disco, que garantem desempenho e segurança. O avanço tecnológico também possibilitou o surgimento da virtualização, permitindo a execução de diferentes sistemas operacionais em uma única máquina física e ampliando a flexibilidade no uso dos recursos.

O *software* pode ainda estar integrado diretamente ao *hardware*, como em painéis eletrônicos de automóveis ou leitores de *Blu-Ray*, caracterizando os chamados "embutidos" (COUTINHO; BEZERRA, 2021). Historicamente, *software* e *hardware* eram concebidos como uma única unidade, sendo o termo "*software*" empregado pela primeira vez por John W. Tukey em 1958 (MAYNARD, 2015). Somente vinte anos depois, o governo dos Estados Unidos determinou que a IBM passasse a contabilizar *hardware* e *software* separadamente, o que abriu espaço para empresas especializadas (SAKURAI; ZUCHI, 2018). Essa dissociação entre *hardware* e *software* impulsionou a inovação, permitindo atualizações independentes, acelerando ciclos de desenvolvimento e fomentando novos modelos de negócio baseados em *software*, que hoje ocupam posição central na economia digital.

Schwab (2019) destaca que o trabalho do programador consiste, de forma essencial, em traduzir requisitos e algoritmos para uma linguagem de programação que o computador consiga interpretar e executar. A partir desse processo, é possível desenvolver diferentes tipos de *software*, como aplicativos, jogos e sistemas de controle para robôs. Um exemplo histórico ligado à ideia de programação foi a invenção de um *tear* "programável" pelo francês Joseph-Marie

Jacquard, que representou um marco na Revolução Industrial. O equipamento não possuía processador, sendo controlado por meio de cartões perfurados. Conforme explica Maynard (2015), esses cartões não utilizavam valores binários como 0 e 1, mas funcionavam através da presença ou ausência de perfurações: um furo indicava uma ação de movimento, enquanto a ausência significava outra. A criação de Jacquard é considerada uma das precursoras da automação moderna.

Maynard (2015) aponta que, em 1843, Augusta Ada King-Noel, mais conhecida como Ada Lovelace, desempenhou um papel marcante ao colaborar com Charles Babbage no projeto do *Analytical Engine*, uma calculadora mecânica de uso geral que acabou não sendo concluída. Durante esse processo, Ada elaborou diversas anotações e descreveu um método para calcular os números de Bernoulli por meio da máquina, o que, segundo Coutinho e Bezerra (2021), é considerado o primeiro programa de computador da história. Posteriormente, em 1941, Konrad Ernst Otto Zuse desenvolveu o Z3, reconhecido como o primeiro sistema de computação totalmente automatizado e programável do mundo.

O Z3 foi construído a partir dos conceitos idealizados por Babbage e tinha como diferencial a capacidade de realizar operações, como a adição, em menos de um quarto de segundo. O propósito central de sua criação estava ligado à busca por maior eficiência nos cálculos. Nos primeiros anos, os *softwares* eram desenvolvidos para máquinas específicas e eram comercializados em conjunto com o *hardware*. Já na década de 1980, passaram a ser distribuídos em mídias físicas, como disquetes, e mais tarde em CDs e DVDs. Atualmente, grande parte dos *softwares* é obtida por meio da *Internet*, seja diretamente nos sites dos fabricantes ou através de plataformas de distribuição de aplicativos.

Os modelos de ciclo de vida de *software* descrevem as etapas que um sistema percorre desde sua concepção até sua desativação, organizando a sequência de atividades necessárias para o desenvolvimento. O objetivo central desses modelos é garantir qualidade, reduzir custos e otimizar o tempo de entrega. Entre as fases mais comuns estão:

- Análise de Requisitos: onde são levantadas e documentadas as necessidades do sistema;
- Design: define a arquitetura, fluxos de trabalho e tecnologias a serem utilizadas;
- Implementação: fase em que o código é desenvolvido com base no planejamento estabelecido;
- Testes: que envolvem diferentes tipos de verificação para assegurar a confiabilidade do *software*;
- Manutenção: responsável por corrigir falhas e implementar melhorias após a entrega do sistema.

De acordo com Cândido (2022), observa-se nos últimos anos uma tendência de desprofissionalização no aprendizado de programação, já que muitas pessoas têm adquirido essas habilidades de forma autodidata, recorrendo a tutoriais *online* em vez de cursos formais. Além

disso, o *networking* global, aliado às diversas plataformas digitais, tem favorecido a formação de grandes comunidades que reúnem tanto entusiastas quanto programadores experientes, tornando cada vez menos nítida a distinção entre esses grupos.

Ao mesmo tempo, percebe-se um avanço significativo nos padrões e na qualidade do desenvolvimento de *software*. Os sistemas estão cada vez mais sofisticados e complexos, o que aumenta a demanda por soluções digitais. Esse contexto reflete uma sociedade em constante transformação, onde as necessidades mudam rapidamente, gerando uma procura crescente por Desenvolvedores *Full Stack* (um profissional de tecnologia que possui a capacidade de trabalhar em todas as camadas de um projeto de *software*) em detrimento de profissionais especializados apenas em uma área.

A complexidade crescente dos sistemas requer profissionais capazes de atuar em todas as camadas do desenvolvimento, abrangendo desde o *Frontend* (interface do usuário) até o *Backend* (servidor, banco de dados e lógica de negócios de uma aplicação). Nesse cenário, os Desenvolvedores *Full Stack* ganham destaque por sua flexibilidade e capacidade de adaptação às mudanças do mercado e às demandas dos usuários. Sua habilidade em compreender e integrar diferentes componentes de um sistema favorece um desenvolvimento mais unificado e eficiente, além de melhorar a comunicação e colaboração dentro das equipes. Assim, em um ambiente em que a agilidade e a adaptabilidade são fatores essenciais, esses profissionais se consolidam como recursos estratégicos para empresas que buscam manter a competitividade.

O desenvolvimento de *software* pode ser entendido como um conjunto de atividades voltadas ao projeto, criação, implementação e manutenção de programas que orientam o funcionamento de computadores, sendo independente do *hardware* e responsável por torná-los programáveis (SCHWAB, 2019).

Conforme aponta Cândido (2022), o desenvolvimento de *software* deve ser integrado ao desenvolvimento de produtos mecânicos e elétricos, o que faz com que o papel dos desenvolvedores de *software* seja menos delimitado em comparação ao dos engenheiros. Esses profissionais podem focar em áreas específicas do projeto, como a codificação, enquanto o ciclo de vida do *software* envolve a transformação de requisitos em funcionalidades, o trabalho em equipes multifuncionais, a gestão de processos e equipes, além de atividades de teste e manutenção.

De acordo com Schwab (2019), a atuação dos desenvolvedores de *software* vai além das equipes de programação, envolvendo também cientistas e fabricantes de dispositivos e *hardware*, que contribuem para a criação de códigos, mesmo não sendo os principais responsáveis pelo desenvolvimento. Essa participação não se limita às indústrias tradicionais de tecnologia da informação, abrangendo também empresas que não se dedicam exclusivamente à produção de *software* ou semicondutores.

A ampliação do papel dos desenvolvedores de *software* evidencia a crescente integração da tecnologia da informação com diversas áreas, como ciência, medicina, engenharia e artes. Atualmente, o desenvolvimento de *software* é fundamental para múltiplos setores, desde a indústria automotiva até o entretenimento, impulsionando a inovação e a transformação digital. Na in-

dústria automotiva, esses profissionais são essenciais para o desenvolvimento de sistemas de condução autônoma e tecnologias avançadas de segurança. Já no cinema, os efeitos visuais e a animação digital dependem fortemente do *software* para criar mundos e personagens realistas. Dessa forma, os desenvolvedores de *software* exercem um papel cada vez mais diversificado e influente, moldando o futuro em diferentes áreas do conhecimento e da atividade humana.

2.1.1 Modelos de Desenvolvimento de *Software*

O desenvolvimento de *software* pode ser conduzido por diferentes abordagens, conhecidas como modelos de processo de *software*. Esses modelos têm como objetivo estruturar e organizar as atividades necessárias para transformar requisitos em um produto final de qualidade. Existem diversos modelos conhecidos, entretanto, os mais populares e utilizados acabam sendo o Modelo Cascata e os Modelos Ágeis.

O Modelo Cascata (ou *Waterfall*) foi um dos primeiros modelos formais de processo de *software*. Proposto por Winston Royce em 1970, consiste em uma sequência linear de fases: levantamento de requisitos, análise, projeto, implementação, testes, implantação e manutenção. Cada fase deve ser concluída antes do início da próxima, funcionando como uma "cascata" de atividades (SOMMERVILLE, 2011). Entre suas vantagens podemos citar:

- Estrutura simples e de fácil entendimento;
- Definição clara das fases e entregas;
- Útil em projetos com requisitos bem definidos e estáveis.

Por outro lado, há também desvantagens que valem ser destacadas deste modelo, sendo elas:

- Pouca flexibilidade diante de mudanças nos requisitos;
- Risco de identificação tardia de falhas, geralmente apenas na fase de testes;
- Dificuldade em lidar com projetos complexos ou inovadores.

Os Modelos Ágeis surgiram como uma alternativa aos modelos tradicionais, sendo formalizados em 2001 com o Manifesto Ágil. Esses modelos priorizam colaboração com o cliente, entregas rápidas e contínuas, adaptação às mudanças e interações entre pessoas mais do que processos rígidos (BECK et al., 2001).

Entre os métodos ágeis mais utilizados, principalmente nos dias atuais, estão o *Scrum*, o *Kanban* e o *Extreme Programming* (conhecido também como *XP*). Todos eles seguem o princípio de dividir o projeto em pequenos ciclos (iterações), resultando em entregas frequentes e incrementais do produto. Como as principais vantagens, podemos citar:

- Alta flexibilidade para mudanças de requisitos;
- Entregas rápidas que geram valor contínuo ao cliente;
- Melhora na comunicação e na colaboração entre equipe e *stakeholders*.

E, das suas principais desvantagens:

- Pode gerar falta de documentação formal, dependendo da equipe;
- Requer alto envolvimento do cliente no processo;
- Menos indicado para projetos de missão crítica que demandam forte rastreabilidade.

Sendo assim, evidencia-se que cada modelo de desenvolvimento possui suas características, vantagens e limitações. A escolha do modelo mais adequado depende do contexto do projeto, da complexidade do sistema, do nível de risco, do grau de inovação e da estabilidade dos requisitos. Modelos tradicionais como Cascata são mais estruturados, os modelos Ágeis oferecem maior flexibilidade e adaptabilidade, características cada vez mais valorizadas no mercado atual. Na Tabela 1, temos um comparativo resumido dos principais pontos dos modelos citados.

Tabela 1 – Comparativo entre Modelo Cascata e Modelos Ágeis

Modelo	**Resumo**	**Vantagem**	**Desvantagem**	**Indicado para**
Cascata	Linear e sequencial	Simples e claro	Pouca flexibilidade	Projetos pequenos e estáveis
Ágeis	Iterativo e adaptável	Entregas rápidas	Pouca documentação	Projetos dinâmicos e mutáveis

Fonte: Elaborado pelo autor

2.2 IA Generativa (aplicada ao desenvolvimento de *software*)

A Inteligência Artificial (IA) consolidou-se como uma das áreas mais relevantes da Ciência da Computação, devido à sua capacidade de simular processos cognitivos humanos como aprendizagem, raciocínio e tomada de decisão por meio de algoritmos e modelos matemáticos (CÂNDIDO, 2022; AGUIAR, 2023). Nos últimos anos, a evolução tecnológica permitiu que esses sistemas ultrapassassem o campo do reconhecimento de padrões e da automação, alcançando um estágio em que são capazes de gerar novos conteúdos, fenômeno que deu origem ao conceito de Inteligência Artificial Generativa. Segundo Uniphore (2024), a IA Generativa refere-se ao ramo da IA que cria resultados originais sejam eles textos, imagens, sons ou códigos, a partir de grandes volumes de dados já existentes, distinguindo-se de modelos meramente classificatórios ou preditivos. Conforme aponta TechTarget (2024), esse processo ocorre por meio da análise de padrões e estruturas aprendidas, sendo ativado a partir de instruções em linguagem natural, conhecidas como prompts.

No campo do desenvolvimento de software, a IA Generativa tem desempenhado papel crescente ao possibilitar a geração automática de trechos de código, documentação, testes e até refatoração de programas. Idrisov e Schlippe (2024) compararam códigos produzidos por ferramentas de IA e por programadores humanos, identificando que, embora as soluções geradas

automaticamente sejam satisfatórias em tarefas de baixa complexidade, ainda apresentam limitações em contextos que exigem maior profundidade técnica ou conhecimento de domínio. Em estudo semelhante, Yu (2025) observou que assistentes de codificação generativa, como Codeium e ferramentas similares, aumentam a produtividade em atividades repetitivas, mas enfrentam dificuldades quando aplicados a tarefas que demandam compreensão arquitetural ou integração com sistemas legados.

Outro aspecto recorrente na literatura é a avaliação da qualidade do código gerado. Yetiştiren et al. (2023) realizaram um comparativo entre GitHub Copilot, Amazon CodeWhisperer e ChatGPT, utilizando o benchmark HumanEval. Os resultados apontaram variações significativas em critérios como correção, confiabilidade e manutenibilidade, indicando que tais ferramentas funcionam melhor como apoio ao programador do que como substitutos plenos. A segurança do software também se apresenta como desafio relevante: de acordo com Alwageed e Khan (2025), embora a IA Generativa possa contribuir para práticas de codificação segura, existe o risco de introdução de vulnerabilidades e de não conformidade com padrões regulatórios.

Por fim, é importante destacar as discussões éticas e sociais associadas ao uso da IA Generativa. Afreen, Mohaghegh e Doborjeh (2025) ressaltam que problemas como a contextualização insuficiente do código, a geração de erros sutis e as preocupações relacionadas a direitos autorais e privacidade ainda carecem de soluções adequadas. Nesse sentido, observa-se que a IA Generativa aplicada ao desenvolvimento de software apresenta grande potencial de aumento de produtividade e inovação, mas exige acompanhamento crítico, validações constantes e reflexões acerca de suas limitações e implicações sociais.

3 TRABALHOS RELACIONADOS

Para fins de melhor localizar os trabalhos com temáticas relacionadas para o presente estudo, foi utilizado a plataforma de busca especializada em literatura acadêmica e científica Google Scholar. Os procedimentos de pesquisa e seleção consistiram em utilizar palavras chaves em português como “inteligência artificial generativa”, “desenvolvimento de software”, “produtividade”, “suporte”, “otimização” e “performance”, variando também os termos em inglês. Por fim, para selecionar os trabalhos a seguir, foi considerado o contexto do estudo de cada autor, conceitos abordados, tecnologias mencionadas e/ou utilizadas e problemáticas ressaltadas, que melhor correlacionassem com o presente estudo.

3.1 Trabalhos Acadêmicos

Dado o contexto do aprimoramento e evolução de Inteligências Artificiais Generativas nos anos recentes, tal como suas aplicabilidades para otimização de tarefas voltadas para a área de desenvolvimento de software, diversos estudos têm sido realizados com o intuito de explorar

ainda mais os conceitos técnicos e práticos. A seguir, são apresentados estudos e pesquisas relacionados ao tema, estabelecendo um paralelo com a temática abordada para este trabalho.

Gomes (2023) apresenta uma análise comparativa de diferentes Inteligências Artificiais Generativas (IAs Generativas) aplicadas no desenvolvimento de software, mas particularmente focando em ferramentas auxiliares como *GitHub Copilot*, *Codeium*, *Tabnine* e *CodeGeeX*. Para fins práticos, seu trabalho focou em experimentos utilizando *GitHub Copilot* e *Codeium* para mensurar a eficiência das ferramentas.

Como resultado, o Gomes (2023) aponta que ambas as tecnologias se mostram extremamente úteis no processo de otimização do desenvolvimento, produtividade e qualidade de códigos, com a escolha entre elas dependendo das necessidades particulares do projeto e das preferências do profissional. Entretanto, o destaque ficou para a ferramenta *GitHub Copilot*, que apresentou uma leve vantagem, com uma capacidade superior de compreensão para construção de uma lógica de desenvolvimento mais coesa.

Em Silva et al. (2025), é realizado o teste prático de viabilidade de desenvolvimento de uma aplicação *web* para gerenciamento de sócios torcedores de um clube de futebol, utilizando IA Generativa como apoio na construção da aplicação. Ainda que, neste estudo os autores desenvolveram uma solução simples, utilizando tecnologias como *JavaScript*, *PHP*, *HTML5* e entre outras, o trabalho demonstrou que utilizar de Inteligência Artificial otimiza o processo de desenvolvimento, desde a geração de código, identificação de falhas e sugestões de melhores práticas para a construção de soluções.

De acordo com os autores e os resultados obtidos (SILVA et al., 2025), a adoção de IA Generativa no processo de desenvolvimento contribui tanto para acelerar a produção, quanto para a qualidade e otimização da aplicação. Para o estudo em questão, os autores contaram com o suporte de modelos de IA Generativa como *GPT-3.5* e *Claude 1*.

Em contrapartida, no trabalho de Ferreira (2023), o autor aborda sobre a influência das IAs Generativas na área de desenvolvimento de software, com base em um estudo qualitativo, combinando cenários exploratórios e descritivos. Aqui, o autor menciona o uso de ferramentas como o *ChatGPT*, *Bard*, *GitHub Copilot* para tratar sobre a temática de modelos baseados no conceito de Processamento de Linguagem Natural (PLN).

Ao longo de seu estudo, Ferreira (2023) aborda questões como o impacto da Inteligência Artificial na programação, trazendo dados e referências, para dissertar sobre tópicos como os desafios e benefícios do uso de ferramentas de IA. Sobre os desafios, o autor menciona a questão da dependência do uso de IA, limitando assim o desenvolvimento de habilidades próprias por parte do programador e a substituição do desenvolvedor por IA, assunto fortemente debatido desde a evolução das IAs Generativas nos anos recentes.

Santos (2024) investigou a influência das ferramentas de IA Generativas, citando *ChatGPT* e *GitHub Copilot*, com foco no processo de ensino e aprendizagem das instituições de ensino superior na área de desenvolvimento de *software*. O objetivo deste estudo consistiu em identificar oportunidades e desafios no uso destas tecnologias. Para isso, o autor realizou um comparativo

com grupos de alunos iniciantes, estagiários no setor de tecnologia, alocando-os em um projeto simulado.

Para tanto o autor (SANTOS, 2024) seguiu um delineamento *crossover* 2x2 AB/BA, isto é, alternando os grupos entre o uso e não uso de IA. Após a coleta de dados, foi mensurado o conhecimento técnico dos grupos, tal como qualidade dos artefatos gerados, avaliação do produto desenvolvido e provas simuladas para testar o conhecimento adquirido. Aqui, o autor ressalta que os grupos que fizeram o uso de IA obtiveram melhores avaliações nos artefatos desenvolvidos, desde a documentação ao desenvolvimento de *software* propriamente dito. Entretanto, a nível de conhecimento, não refletiu em avaliações melhores nas provas simuladas (notas de avaliações, propriamente).

Costa (2024) explorou como o uso de IA Generativa pode otimizar e aprimorar o processo de desenvolvimento de *software* no contexto empresarial. O autor conduziu uma extensa revisão bibliográfica com o intuito de identificar as ferramentas e metodologias mais atuais e eficazes no mercado. A partir de então, selecionou e implementou *frameworks* como *Spring Boot* para a construção de uma aplicação escalável em *Java*, utilizando o *ChatGPT* como ferramenta de suporte durante o processo.

Ademais, o autor (COSTA, 2024) fez o uso de ferramentas de teste e documentação de *APIs*, tais como *Postman* e até mesmo *JUnit*, como *framework* para validação de testes unitários. Em seu trabalho, o autor utilizou a IA Generativa através de *prompts* para criação de todos os requisitos, códigos e validações do seu projeto. Como resultado, ele evidenciou uma melhoria na eficiência tanto do desenvolvimento quanto da qualidade da aplicação entregue. Entretanto, ressaltou a importância de que, o conhecimento prévio para uso de IAs Generativas é fundamental, para que se torne possível o uso da mesma como ferramenta de apoio no processo de desenvolvimento de *software*.

Por fim, com base nos estudos analisados e considerando o crescimento acelerado das ferramentas de Inteligência Artificial Generativa na atualidade, evidencia-se que o profissional — independentemente da área de atuação — pode, e deve, tirar proveito dessas tecnologias para se manter em constante aprendizado e adaptado às novas realidades do mercado. Ou seja, não é necessário dominar todos os princípios técnicos por trás dessas ferramentas, mas sim saber utilizá-las estrategicamente, visando otimização, desempenho e melhores resultados.

Dessa forma, a proposta deste estudo é analisar, com base em dados e indicadores, o quanto o uso da Inteligência Artificial Generativa pode contribuir, junto aos profissionais da área de desenvolvimento de *software*, para a entrega de soluções mais ágeis. Para delimitar o foco principal do trabalho, o segmento analisado será o desenvolvimento de software, considerando o papel do desenvolvedor no cenário atual do mercado de trabalho.

Com base no levantamento de todas as informações dos trabalhos relacionados ao tema, é possível agrupar as principais características de cada um deles para fins de comparação. Dada as informações levantadas, separa-se as características em quatro parâmetros, sendo eles:

- **Tema principal:** define qual foi o principal tema abordado ao longo do estudo;

- **Metodologia aplicada:** define o modelo utilizado para o estudo em questão;
- **Tecnologias/Inteligências Artificiais utilizadas:** se aplicável, define as ferramentas e/ou tecnologias utilizadas para fins de estudo, tal como o modelo de Inteligência Artificial utilizado ou analisado;
- **Objetivo:** define o objetivo do modelo, ou seja, qual problemática o modelo buscou esclarecer ao seu decorrer.

A partir do resultado do levantamento de trabalhos relacionados, foi possível elaborar a Tabela 2, onde mostra o conteúdo dos critérios apontados em cada estudo, por ordem de apresentação.

Tabela 2 – Comparativo entre os trabalhos relacionados

Autores	**Tema**	**Metodologia**	**Tecnologias/IA**	**Objetivo**
Gomes (2023)	Comparação de IAs no desenvolvimento de *software*	Testes práticos comparativos	*GitHub Copilot*, *Codeium*, *Tabnine*, *CodeGeeX*	Avaliar eficiência de ferramentas de IA no desenvolvimento
Silva et al. (2025)	Aplicação de IA na criação de sistemas *web*	Estudo de caso com experimento	*JavaScript*, *PHP*, *HTML5*, *GPT-3.5*, *Claude 1*	Verificar ganho de produtividade com uso de IA
Ferreira (2023)	Impacto da IA na programação	Estudo qualitativo exploratório	*ChatGPT*, *Bard*, *GitHub Copilot*	Analisar benefícios e riscos do uso da IA
Santos (2024)	IA no ensino de tecnologia	Experimento com grupos (*crossover*)	*ChatGPT*, *GitHub Copilot*	Medir impacto da IA na aprendizagem e desempenho técnico
Costa (2024)	Uso de IA Generativa no desenvolvimento de *software* empresarial	Revisão bibliográfica com aplicação prática	*Spring Boot*, *Java*, *ChatGPT*, *Postman*, *JUnit*	Otimizar e melhorar eficiência e qualidade no desenvolvimento de aplicações

Fonte: Elaborado pelo autor

4 MATERIAIS E MÉTODOS

Este trabalho tem como principal objetivo investigar a utilização de IA Generativa no processo de desenvolvimento de *software*. A partir do conhecimento consolidado com base na fundamentação teórica, o presente estudo consiste em um breve experimento, dividido em dois grupos participantes do mesmo.

Sendo assim, para atingir os objetivos, a pesquisa percorreu cinco fases. A primeira consistiu em analisar trabalhos relacionados, almejando identificar tópicos existentes sobre o assunto ou que explorassem conceitos e práticas no que dizem respeito ao desenvolvimento de *software*

com auxílio de IA. Com base nesta análise, foi possível traçar um paralelo entre os estudos relacionados e o modelo proposto.

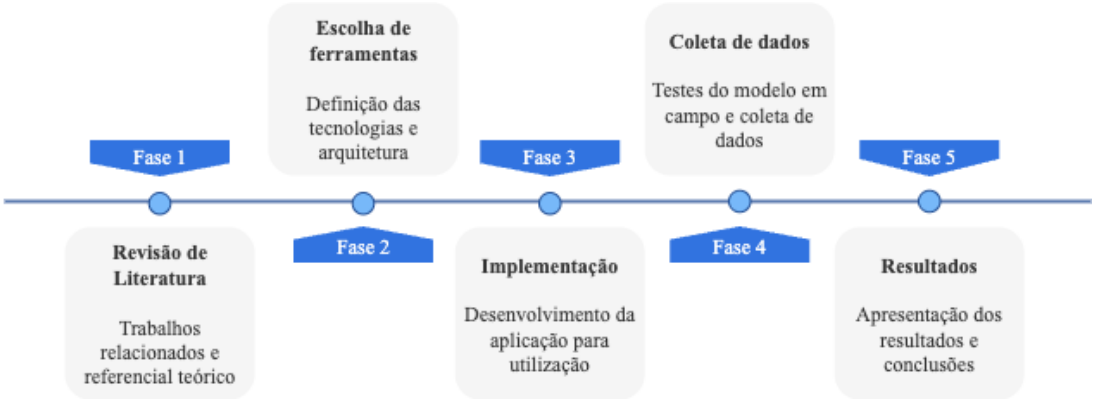
Na segunda fase, definiu-se as tecnologias, bem como a arquitetura da aplicação *web* proposta para o desenvolvimento do trabalho. Com isso, estabelece-se o alicerce para o desenvolvimento da solução e para o cumprimento dos objetivos do trabalho.

A terceira fase consistiu no desenvolvimento da aplicação *web* que foi utilizada como ambiente de execução do desafio técnico, assim como a elaboração do desafio disponibilizado dentro da plataforma. O objetivo central da ferramenta foi de oferecer, de forma prática e organizada, o enunciado do desafio, permitindo que os participantes iniciassem a atividade e, ao final, submetessem suas soluções. Além disso, a aplicação foi responsável por registrar automaticamente o tempo total despendido por cada participante, desde o momento em que iniciou a tarefa até a finalização do envio, viabilizando a coleta de dados confiáveis para a análise posterior.

A quarta fase compreendeu a aplicação prática do modelo e a coleta de dados. Para tanto, foram selecionados participantes por conveniência, ou seja, profissionais de desenvolvimento de *software* previamente conhecidos pelo pesquisador e que se dispuseram a colaborar com o estudo. Após a seleção, os desenvolvedores foram distribuídos de forma aleatória em dois grupos: um com autorização para utilizar ferramentas de IA e outro restrito ao desenvolvimento sem esses recursos. Essa etapa tem como propósito garantir condições controladas para comparar os desempenhos e mensurar a utilização da IA nos resultados obtidos.

Após a coleta dos resultados, a quinta e última fase consistiu na apresentação e validação dos dados, bem como na elaboração das considerações finais do estudo, de forma a contribuir para a área de pesquisa em questão. A Figura 1 ilustra as fases da pesquisa, contendo uma breve descrição de cada uma.

Figura 1 – Fases da pesquisa



Fonte: Elaborado pelo autor

4.1 Proposta de Solução

Conforme mencionado anteriormente, observa-se que o impacto da Inteligência Artificial no desenvolvimento de *software* ainda é um tema em debate, sendo comum que as análises existentes se restrinjam a relatos de uso ou percepções subjetivas por parte dos desenvolvedores. Em muitos casos, a avaliação da influência da IA sobre fatores como produtividade e tempo de execução de tarefas ocorre de forma fragmentada, sem um método padronizado que permita mensurar de maneira objetiva os efeitos reais de tais ferramentas.

Dentro desse cenário, a proposta do estudo surge com a elaboração do um desafio técnico, disponibilizado através da plataforma *web* desenvolvida especificamente para este fim. A ferramenta funcionou como um ambiente de avaliação controlado, no qual os participantes foram previamente organizados em dois grupos: um grupo com autorização para utilizar ferramentas de IA durante a resolução do desafio e outro grupo para realizar a atividade sem recorrer a tais recursos.

A proposta buscou, assim, criar condições equivalentes para ambos os grupos, garantindo que todos recebessem o mesmo enunciado do desafio, apresentado diretamente na plataforma. Para cada participante foi disponibilizado um *link* único, correspondente ao grupo ao qual pertencia. A partir desse acesso, o desenvolvedor pôde visualizar as instruções, iniciar a tarefa e, ao concluir, submeter os arquivos com sua solução.

A plataforma também atuou como mecanismo de registro, armazenando automaticamente o tempo total gasto por cada participante, desde o momento em que deu início ao desafio até a finalização da entrega. Dessa forma, além da coleta da solução proposta, foi possível analisar métricas objetivas como tempo de conclusão, resolução dos problemas propostos (*bugs*) e até mesmo, eventuais diferenças de abordagem entre os grupos.

5 MODELO PROPOSTO

Esta seção tem como objetivo descrever brevemente a ferramenta desenvolvida e utilizada para a execução deste trabalho, incluindo sua arquitetura, funcionalidades e tecnologias utilizadas. O modelo proposto consiste em uma aplicação *web* que centraliza as informações do desafio técnico, disponibilizando-o de forma prática e organizada para os desenvolvedores participantes do estudo.

A aplicação coleta o nome e o cargo atual do participante no mercado de trabalho. O nome é utilizado exclusivamente para exibir uma mensagem personalizada de boas-vindas. Funciona como um identificador simples durante a sessão, mantendo o usuário ativo no desafio, sem necessidade de cadastro, *tokens* ou *logins* complexos. Dessa forma, os nomes dos participantes não são utilizados na apresentação dos resultados do estudo, garantindo anonimato e simplicidade na utilização da ferramenta.

5.1 Visão Geral

Lorem ipsum.

5.2 Arquitetura e Tecnologias

Lorem ipsum.

5.3 Funcionalidades

A seguir serão apresentadas as principais funcionalidades da plataforma proposta para o desenvolvimento deste modelo, detalhando cada um delas. Nesta seção estão as telas dos fluxos de uso e etapas dentro da plataforma. As demais telas estão disponíveis no Apêndice 7.

5.4 Formulário Inicial

Ao acessar a plataforma *web* através do *link* fornecido juntamente com as instruções iniciais via *WhatsApp* para cada participante, o desenvolvedor necessita preencher um breve formulário informando seu "Nome e Sobrenome"(que é somente de uso interno na plataforma), seu "Cargo atual"e então pressionar o botão "Registrar". Os campos "Nome e Sobrenome"e "Cargo atual"são obrigatórios para o participante poder avançar.

5.5 Tela Inicial e Informações do Desafio

Após o preenchimento do formulário inicial, o participante é direcionado para a primeira tela da plataforma, que neste caso é a tela do desafio técnico propriamente. Porém, nesta tela, antes de exibir os detalhes do desafio, é exibido mais algumas informações que reforçam a necessidade de atenção e dedicação do participante antes de dar início ao teste.

Estando de acordo, o participante clica no botão "De acordo"e na mesma tela é renderizada todos os detalhes e imagens de cenários de exemplo, descrevendo o desafio técnico que o desenvolvedor irá solucionar. A tela inicial e as informações do desafio encontram-se na Figura **??**.

Ao descer um pouco a tela, o desenvolvedor encontra as imagens de cenários de exemplo que ilustram o objetivo final das funcionalidades que a solução precisa atender para concluir o desafio de modo satisfatório, como exemplo da Figura **??**.

Ao final da tela, está o botão para realizar o *download* do código-fonte do desafio técnico e logo abaixo, o botão para iniciar o desafio, que só é habilitado após o participante acionar o anterior, ou seja, realizar o *download*. Nesta parte também foi colocado mais um aviso reforçando a importância do participante preparar seu ambiente de desenvolvimento antes de clicar em "ini-

ciar desafio", pois ao acionar este botão, uma rota da API é chamada que armazena no Banco de Dados o horário de início (*timestamp*) do desenvolvedor, registrando assim o momento exato em que o participante iniciou a atividade.

5.6 Desafio Técnico

Lorem ipsum.

5.7 Tela de Upload

Lorem ipsum.

6 ANÁLISE E DISCUSSÃO DOS RESULTADOS

Nesta seção são apresentadas a coleta e análise dos dados levantados a partir das soluções fornecidas pelos participantes, juntamente com o registro do tempo total de duração para a conclusão do desafio de cada desenvolvedor. Foi coletado também dados de *feedback* referente ao experimento através de um questionário aplicado no término do desafio, com o intuito de forncer mais resultados para uma análise relevante do projeto como um todo.

6.1 Coleta de Dados

Lorem ipsum.

6.2 Avaliação dos Resultados

Lorem ipsum.

7 CONSIDERAÇÕES FINAIS E TRABALHOS FUTUROS

Lorem ipsum.

Referências

BECK, K. et al. **Manifesto for Agile Software Development**. 2001. Disponível em: <https://agilemanifesto.org/>. Acesso em: 29 ago. 2025.

COSTA, V. J. L. **ChatGPT: uma análise da ferramenta aplicada no processo de desenvolvimento de software**. Goiânia, GO: [s.n.], 2024. Pontifícia Universidade Católica de Goiás. Disponível em: <https://repositorio.pucgoias.edu.br/jspui/handle/123456789/7929>. Acesso em: 23 ago. 2025.

COUTINHO, E.; BEZERRA, C. Simulação de alocação de recursos em projetos de desenvolvimento de software utilizando teoria das filas. In: **Anais do III Workshop em Modelagem e Simulação de Sistemas Intensivos em Software**. Porto Alegre, RS: SBC, 2021. p. 30–39. ISSN 0000-0000. Disponível em: <https://sol.sbc.org.br/index.php/mssis/article/view/17257>. Acesso em: 24 ago. 2025.

COUTINHO, M. et al. **The Role of Generative AI in Software Development Productivity: A Pilot Case Study**. 2024. Disponível em: <https://arxiv.org/abs/2406.00560>. Acesso em: 12 set. 2025.

CâNDIDO, A. K. R. **Uma revisão sistemática de estudos secundários sobre práticas ágeis de desenvolvimento de software**. Goiânia, GO: [s.n.], 2022. Pontifícia Universidade Católica de Goiás. Disponível em: <https://repositorio.pucgoias.edu.br/jspui/handle/123456789/4436>. Acesso em: 24 ago. 2025.

FERREIRA, I. S. **Benefícios e desafios do uso de IAs na programação**. Salgueiro, PE: [s.n.], 2023. TCC (Sistemas para Internet) – Instituto Federal de Educação, Ciência e Tecnologia do Sertão Pernambucano. Disponível em: <https://releia.ifsertao-pe.edu.br/jspui/handle/123456789/1184>. Acesso em: 21 jul. 2025.

GITHUB. **GitHub Copilot documentation**. 2021. Disponível em: <https://docs.github.com/en/copilot>. Acesso em: 12 set. 2025.

GOMES, P. R. P. **Uma análise preliminar das IA generativa no suporte ao desenvolvimento de software**. Serra, ES: [s.n.], 2023. 61 p. Monografia (Bacharelado em Sistemas de Informação) – Instituto Federal do Espírito Santo. Disponível em: <https://repositorio.ifes.edu.br/handle/123456789/5914>. Acesso em: 21 jul. 2025.

MAYNARD, A. D. Navigating the fourth industrial revolution. **Nature Nanotechnology**, Tempe, Arizona, EUA, v. 10, p. 1005–1006, 2015. Disponível em: <https://www.nature.com/articles/nnano.2015.286>. Acesso em: 24 ago. 2025.

NETO, J. G. **Metodologias ágeis em uma microempresa de desenvolvimento de softwares: um estudo de caso com o uso do Scrum**. Curitiba, PR: [s.n.], 2019. Universidade Tecnológica Federal do Paraná. Disponível em: <http://repositorio.utfpr.edu.br/jspui/handle/1/19439>. Acesso em: 24 ago. 2025.

SAKURAI, R.; ZUCHI, J. D. AS REVOLUÇÕES INDUSTRIAIS ATÉ A INDUSTRIA 4.0. **Revista Interface Tecnológica**, Taquaritinga, SP, v. 15, n. 2, p. 480–491, 2018. Disponível em: <https://revista.fatectq.edu.br/interfacetecnologica/article/view/386>. Acesso em: 24 ago. 2025.

SANTOS, G. P. dos. **Inteligência artificial generativa: o processo de ensino-aprendizagem no ensino superior de tecnologia**. Tese (Dissertação – Mestrado em Administração de Organizações) — Faculdade de Economia, Administração e Contabilidade de Ribeirão Preto, University of São Paulo, Ribeirão Preto, SP, 2024. Disponível em: <https://www.teses.usp.br/teses/disponiveis/96/96132/tde-23012025-124616/en.php>. Acesso em: 22 jul. 2025.

SCHWAB, K. **A Quarta Revolução Industrial**. [S.l.]: Edipro, 2019.

SILVA, I. H. da et al. DESENVOLVIMENTO DE APLICAÇÃO WEB COM INTELIGÊNCIA ARTIFICIAL GENERATIVA COMO AGENTE TUTOR SUPERVISOR NA CODIFICAÇÃO. **ARACÊ**, v. 7, n. 5, p. 22855–22877, 2025. Disponível em: <https://periodicos.newsciencepubl.com/arace/article/view/4942>. Acesso em: 21 jul. 2025.

SOMMERVILLE, I. **Engenharia de Software**. 9. ed. São Paulo: Pearson Addison Wesley, 2011.

APÊNDICE A – DIAGRAMA ER DO BANCO DE DADOS

Lorem ipsum.

APÊNDICE B – TELAS DA APLICAÇÃO

Lorem ipsum.

APÊNDICE C – TEXTO TEXTO

Lorem ipsum.

APÊNDICE D – CENÁRIOS DE TESTES

Lorem ipsum.

APÊNDICE E – TEXTO TEXTO

Lorem ipsum.

APÊNDICE F – TEXTO TEXTO

Lorem ipsum.

APÊNDICE G – RESPOSTAS DO QUESTIONÁRIO

Lorem ipsum.

APÊNDICE H – REQUISITOS FUNCIONAIS DA APLICAÇÃO

Lorem ipsum.

APÊNDICE I – TEXTO TEXTO

Lorem ipsum.

ANEXO A – NOME DO ANEXO

Lorem ipsum.