INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE MINAS GERAIS - *CAMPUS* OURO BRANCO
SISTEMAS DE INFORMAÇÃO

Filipe Borges Franco de Assis

O USO DA IA GENERATIVA NO AUMENTO DA PRODUTIVIDADE NO DESENVOLVIMENTO DE SOFTWARE

Ouro Branco - MG
2026

FILIPE BORGES FRANCO DE ASSIS

O USO DA IA GENERATIVA NO AUMENTO DA PRODUTIVIDADE NO DESENVOLVIMENTO DE SOFTWARE

Trabalho de conclusão de curso apresentado ao Curso de Sistemas de Informação do Instituto Federal de Educação, Ciência e Tecnologia de Minas Gerais - *Campus* Ouro Branco para a obtenção do título de Bacharel em Sistemas de Informação.

Orientador: Prof. Lucas Portela Costa da Silva

Ouro Branco - MG
2026

A848u Assis, Filipe Borges Franco de.

O uso da IA generativa no aumento da produtividade no desenvolvimento de software. Filipe Borges Franco de Assis. – 2026.

22f.il.

Orientador: Lucas Portela Costa da Silva.

Trabalho de Conclusão de Curso (Sistemas de Informação) – Instituto Federal de Minas Gerais. *Campus* Ouro Branco, 2026.

1. Inteligência artificial. 2. Inteligência artificial generativa. 3. Desenvolvimento de software. 4. Produtividade.5. Engenharia de software. I. Silva, Lucas Portela Costa da. II. Instituto Federal de Minas Gerais. *Campus* Ouro Branco. III. Título.

CDU: 004.41

Catalogação: Márcia Margarida Vilaça - CRB-6/2235
Biblioteca do Instituto Federal de Minas Gerais, *Campus* Ouro Branco

MINISTÉRIO DA EDUCAÇÃO
SECRETARIA DE EDUCAÇÃO PROFISSIONAL E TECNOLÓGICA
INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE MINAS GERAIS
***CAMPUS* OURO BRANCO**
Av. Afonso Sardinha, nº 90, Bairro Pioneiros, CEP: 36.420-000, Ouro Branco - Minas Gerais
(31) 3742-2149 – gabinete.ourobranco@ifmg.edu.br

ANEXO II – ATA DE CONCLUSÃO DE TCC

Aos 19 dias do mês de janeiro de 2026, às 20:05 horas, Filipe Borges Franco de Assis, aluno(a) regularmente matriculado no Curso de Sistemas de Informação do Instituto Federal de Minas Gerais, campus Ouro Branco, matrícula 0072378, concluiu o seu Trabalho de Conclusão de Curso por meio de:

( ) Publicação do artigo intitulado _____ na revista/conferência _____, cujo comprovante de aceitação será anexado a esta ata, recebendo a nota _____ pelo trabalho. Eu, na qualidade de orientador do aluno, lavrei a presente ata atestando a conclusão do trabalho, a qual será assinada por mim e pelo aluno.

_____

Professor Orientador

_____

Aluno

(X) Defesa em sessão pública realizada às 20h05 horas, na sala De Convenções do Instituto Federal de Minas Gerais, campus Ouro Branco, na presença da banca examinadora composta pelos docentes:

1 -Carlos Eduardo Paulino Silva

2 -Ederson Naves Fernandes Gonçalves Júnior

3 -Daniela Costa Terra

4 -Marcio Assis Miranda

do artigo intitulado O Uso da IA Generativa no Aumento da Produtividade no Desenvolvimento de Software

MINISTÉRIO DA EDUCAÇÃO
SECRETARIA DE EDUCAÇÃO PROFISSIONAL E TECNOLÓGICA
INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE MINAS GERAIS
***CAMPUS* OURO BRANCO**
Av. Afonso Sardinha, nº 90, Bairro Pioneiros, CEP: 36.420-000, Ouro Branco - Minas Gerais
(31) 3742-2149 – gabinete.ourobranco@ifmg.edu.br

A banca examinadora, após reunião em sessão reservada, deliberou pela Aprovação do referido trabalho, atribuindo a nota 97 . Eu, na qualidade de presidente da banca examinadora, lavrei a presente ata que será assinada por mim, pelos demais examinadores e pelo aluno.

Observações pertinentes à defesa:

_____

_____

_____

NOME E ASSINATURA DOS COMPONE

ORIENTADO

gov.br Documento assinado digitalmente
LUCAS PORTELA COSTA DA SILVA
Data: 26/01/2026 12:51:13-0300
Verifique em https://validar.iti.gov.br

_____

Professor Orientador

gov.br Documento assinado digitalmente
CARLOS EDUARDO PAULINO SILVA
Data: 23/01/2026 13:38:57-0300
Verifique em https://validar.iti.gov.br

_____

Examinador 1

gov.br Documento assinado digitalmente
EDERSON NAVES FERNANDES GONCALVES JUN
Data: 23/01/2026 16:32:24-0300
Verifique em https://validar.iti.gov.br

_____

Examinador 2

gov.br Documento assinado digitalmente
DANIELA COSTA TERRA
Data: 24/01/2026 15:16:30-0300
Verifique em https://validar.iti.gov.br

_____

Examinador 3

gov.br Documento assinado digitalmente
MARCIO ASSIS MIRANDA
Data: 26/01/2026 12:04:21-0300
Verifique em https://validar.iti.gov.br

_____

Examinador 4

gov.br Documento assinado digitalmente
FILIPE BORGES FRANCO DE ASSIS
Data: 23/01/2026 11:28:01-0300
Verifique em https://validar.iti.gov.br

_____

Aluno

MINISTÉRIO DA EDUCAÇÃO
SECRETARIA DE EDUCAÇÃO PROFISSIONAL E TECNOLÓGICA
INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE MINAS GERAIS
***CAMPUS* OURO BRANCO**
Av. Afonso Sardinha, nº 90, Bairro Pioneiros, CEP: 36.420-000, Ouro Branco - Minas Gerais
(31) 3742-2149 – gabinete.ourobranco@ifmg.edu.br

DECLARAÇÃO ANTI-PLÁGIO

Eu, Filipe Borges Franco de Assis, estudante do curso Bacharelado em Sistemas de Informação do IFMG – Campus Ouro Branco, declaro, para os devidos fins e efeitos, e para fazer prova junto ao IFMG – Campus Ouro Branco, que, **sob as penalidades previstas no art. 299 do Código Penal Brasileiro**, que é de minha criação

o Trabalho de Conclusão de Curso que ora apresento.

Art. 299 do Código Penal Brasileiro, que dispõe sobre o crime de *Falsidade Ideológica*:

"Omitir, em documento público ou particular, declaração que dele devia constar, ou nele inserir ou fazer inserir declaração falsa ou diversa da que devia estar escrita, com o fim de prejudicar direito, criar obrigação ou alterar verdade sobre fato juridicamente relevante:

Pena – reclusão, de 1 (um) a 5 (cinco) anos, e multa, se o documento é público, e reclusão de 1 (um) a 3 (três) anos, e multa, se o documento é particular.

Parágrafo único. Se o agente é funcionário público, e comete o crime prevalecendo-se do cargo, ou se a falsificação ou alteração é de assentamento de registro civil, aumenta-se a pena de sexta parte".

Este crime engloba plágio e compra fraudulenta de documentos científicos. Por ser verdade, e por ter ciência do referido artigo, firmo a presente declaração.

Ouro Branco, 23*de* janeiro *de* 2026

gov.br Documento assinado digitalmente
FILIPE BORGES FRANCO DE ASSIS
Data: 23/01/2026 11:28:41-0300
Verifique em https://validar.iti.gov.br

Assinatura do aluno: _____

RESUMO

Este trabalho investigou sistematicamente o potencial da Inteligência Artificial Generativa (IAG) para aumentar a produtividade no desenvolvimento de software. Desenvolveu-se uma ferramenta protótipo baseada no modelo Gemini-2.5-Flash para automatizar a tradução de requisitos em código-fonte. Por meio de um experimento controlado (*bootcamp*) com grupos de controle e experimental, avaliaram-se métricas objetivas (tempo e qualidade do código) e subjetivas (esforço cognitivo e satisfação). Uma abordagem mista validou ganhos significativos de produtividade. Os resultados quantificam melhorias de eficiência e fornecem evidências sobre a integração ideal humano-IA em processos ágeis, oferecendo recomendações práticas para a adoção responsável de IAG na Engenharia de Software.

Palavras-chave: Inteligência Artificial; Inteligência Artificial Generativa; Desenvolvimento de Software; Produtividade; Otimização de Processos; Engenharia de Software.

ABSTRACT

This study systematically investigated the potential of Generative Artificial Intelligence (GAI) to increase productivity in software development. A prototype tool based on the Gemini-2.5-Flash model was developed to automate the translation of software requirements into source code. Through a controlled experiment (*bootcamp*) with control and experimental groups, objective metrics (time and code quality) and subjective metrics (cognitive effort and satisfaction) were evaluated. A mixed-methods approach validated significant productivity gains. The results quantify efficiency improvements and provide evidence on effective human–AI integration in agile processes, offering practical recommendations for the responsible adoption of GAI in Software Engineering.

Keywords: Artificial Intelligence; Generative Artificial Intelligence; Software Development; Productivity; Process Optimization; Software Engineering.

SUMÁRIO

1. Introdução

O desenvolvimento de software é uma atividade complexa que envolve múltiplas etapas, marcada por desafios que vão desde a concepção de soluções até sua implementação e manutenção. Um dos pontos mais críticos e suscetíveis a falhas nesse processo é a transição entre a especificação de requisitos e a escrita do código propriamente. Estudos empíricos demonstram que fatores relacionados à gestão inadequada de projetos e problemas organizacionais são as principais causas de fracasso em projetos de software [Verner et al. 2008]. A ambiguidade inerente à linguagem natural, somada a diferentes interpretações por parte de analistas e desenvolvedores, cria uma lacuna que frequentemente resulta em software que não atende plenamente às necessidades do usuário.

Paralelamente a esses desafios persistentes, o campo da Inteligência Artificial (IA) tem passado por uma transformação acelerada, impulsionada principalmente pelos avanços em IA Generativa e nos Modelos de Linguagem de Grande Escala (*LLMs*). Essas tecnologias demonstraram uma capacidade notável de compreender, processar e gerar conteúdo em linguagem natural e em código com um nível de sofisticação sem precedentes [Jiang et al. 2024]. No contexto da Engenharia de Software, essas ferramentas estão sendo cada vez mais aplicadas para otimizar o ciclo de vida do desenvolvimento, automatizando tarefas como a geração de código, a elaboração de casos de teste e a criação de documentação técnica [Jiang et al. 2024].

Apesar do potencial evidente, a aplicação da IA Generativa para mitigar a lacuna fundamental entre requisitos e implementação ainda é um campo em exploração [Norheim et al. 2024]. Sendo assim, a questão central que motivou esta pesquisa foi: de que forma a IA Generativa pode ser sistematicamente empregada para traduzir requisitos de software em código funcional, e qual o impacto real dessa automação na produtividade dos desenvolvedores? A hipótese central foi que a utilização de uma ferramenta especializada, baseada em LLM, poderia atuar na redução de ruído entre as fases de concepção (*"Sprint 0"*) e implementação (*"Sprint 1"*). A ferramenta não visa substituir o desenvolvedor, mas servir como um elo auxiliar no ciclo, conectando as intenções do Analista de Requisitos/PO à execução técnica do Desenvolvedor e Tester, reduzindo a incidência de erros de interpretação, servindo como uma ponte eficaz entre a especificação e a codificação.

Portanto, este trabalho investigou essa hipótese por meio do desenvolvimento e da avaliação de uma ferramenta-protótipo. O objetivo não foi apenas construir uma nova aplicação, mas utilizá-la como um instrumento para medir e comprovar o impacto da IA Generativa na produtividade. A pesquisa buscou responder se, ao fornecer aos desenvolvedores um código-base gerado diretamente a partir dos requisitos, foi possível obter ganhos mensuráveis em termos de tempo, qualidade e esforço.

A relevância desta pesquisa abrange as dimensões técnica, econômica e acadêmica, ao validar abordagens que otimizam fases críticas do desenvolvimento, reduzem retrabalho e fornecem evidências empíricas sobre a aplicação de IA. O trabalho estrutura-se apresentando os objetivos e a fundamentação teórica, seguidos pela descrição da metodologia experimental e a discussão dos resultados obtidos.

2. Objetivos

2.1. Objetivo Geral

Avaliar e demonstrar o impacto positivo da Inteligência Artificial Generativa na produtividade do desenvolvimento de software, utilizando como estudo de caso uma ferramenta desenvolvida em Python e integrada ao modelo Gemini-2.5-Flash, projetada para automatizar a conversão de requisitos de software em código-fonte.

2.2. Objetivos Específicos

- Desenvolver um protótipo funcional de uma ferramenta que utilizou o modelo Gemini-2.5-flash, em linguagem Python, para traduzir requisitos textuais em código-fonte;
- Conduzir um estudo controlado com profissionais de desenvolvimento de software, utilizando um protocolo definido para coleta de dados quantitativos e qualitativos, a fim de avaliar a produtividade, qualidade do código e experiência dos participantes;
- Validar estatisticamente a hipótese de que a integração da IA Generativa, por meio da ferramenta protótipo desenvolvida, promoveu um aumento mensurável da produtividade no desenvolvimento de software.

3. Referencial Teórico

Este capítulo apresenta a fundamentação teórica que sustenta esta pesquisa. Diferente de abordagens puramente históricas, foca-se aqui nos pilares técnicos que sustentam a solução proposta: a Inteligência Artificial Generativa, a Engenharia de Requisitos e a Engenharia de Prompts, culminando na análise de trabalhos correlatos.

3.1. Inteligência Artificial Generativa e LLMs

A Inteligência Artificial Generativa constitui uma categoria de IA capaz de criar conteúdo novo, como textos, imagens ou código, a partir de padrões aprendidos em grandes conjuntos de dados [Naveed et al. 2024, Center for Security and Emerging Technology (CSET) 2023]. Diferente de sistemas tradicionais voltados apenas para classificação ou análise, sua função central é a geração de conteúdos originais.

Dentro deste campo, os Grandes Modelos de Linguagem (*LLMs*) focam no processamento e geração de linguagem natural e código. A arquitetura *Transformer* [Vaswani et al. 2017] é a base da maioria dos LLMs atuais, permitindo o processamento paralelo de sequências textuais e a captura de contextos complexos. Essa capacidade é fundamental para a Engenharia de Software, pois permite que modelos "entendam" especificações em linguagem natural e as traduzam para linguagens de programação formais [Jiang et al. 2024].

3.2. Engenharia de Requisitos e o Desafio da Ambiguidade

A Engenharia de Requisitos é a base para o desenvolvimento de sistemas de software de qualidade, envolvendo a descoberta, análise, documentação e verificação dos serviços e restrições do sistema [Sommerville 2011]. Segundo Pressman [Pressman 2014], a

compreensão correta dos requisitos é frequentemente a fase mais difícil do processo, pois erros nesta etapa propagam-se exponencialmente para o design e a codificação.

Um dos maiores desafios nesta disciplina é a ambiguidade inerente à linguagem natural. Diferente das linguagens formais (código), a linguagem humana permite múltiplas interpretações. Quando um requisito ambíguo é passado para o desenvolvimento, gera-se retrabalho e falhas no produto final [Verner et al. 2008].

A importância de requisitos bem estruturados no processo de geração automática reside no fato de que a IA atua como um "intérprete estrito". Se a entrada for vaga, a saída (o código) será inconsistente ou incorreta. Portanto, a estruturação prévia dos requisitos, que consiste em transformar linguagem natural vaga em especificações lógicas, é condição *sine qua non* para a eficácia da automação.

3.3. Engenharia de Prompts e Qualidade de Código

Com a ascensão dos LLMs, surgiu a Engenharia de Prompts como uma nova disciplina técnica. Ela refere-se à prática sistemática de projetar, refinar e otimizar as entradas (*prompts*) dadas aos modelos de IA para obter as saídas mais precisas e úteis possíveis [White et al. 2023].

A Engenharia de Prompts afeta diretamente a qualidade do código gerado ao atuar como uma camada de validação lógica. Técnicas como *Chain-of-Thought* (Cadeia de Pensamento), nas quais se instrui o modelo a explicar seu raciocínio passo a passo antes de gerar o código, ajudam a reduzir as chamadas "alucinações" (respostas plausíveis, mas incorretas) [Bang et al. 2023]. Ao fornecer contexto explícito e restrições de arquitetura no prompt, garante-se que o código gerado siga padrões de projeto, como MVC[1] ou DAO[2], e não seja apenas um conjunto de scripts desconexos.

3.4. Desafios Éticos, Privacidade e Propriedade Intelectual

A integração de LLMs no fluxo de desenvolvimento impõe desafios que transcendem a eficiência técnica, exigindo atenção redobrada à privacidade e à propriedade intelectual. O primeiro risco reside na submissão de requisitos e lógicas de negócio a modelos de terceiros (como Gemini ou ChatGPT), o que pode acarretar vazamento de segredos industriais caso os dados sejam utilizados para o retreinamento dos algoritmos [Center for Security and Emerging Technology (CSET) 2023]. Simultaneamente, embora os modelos gerem código de forma probabilística, criando sequências novas em vez de meramente copiar trechos, a fronteira do plágio não intencional e a classificação jurídica dessas obras derivadas permanecem em debate. Diante disso, a literatura atual sugere que o desenvolvedor humano deve atuar como o garantidor final da originalidade e da licitude do uso daquele código em projetos comerciais, uma vez que a utilização da ferramenta não o isenta da responsabilidade ética sobre o artefato final [Jiang et al. 2024].

3.5. Trabalhos Correlatos

Esta seção apresenta uma síntese dos principais trabalhos identificados por meio da Revisão Sistemática da Literatura, abordando o uso da IA Generativa no desenvolvi-

[1] *Model-View-Controller*: Padrão arquitetural que divide a aplicação em três componentes interconectados (Modelo, Visão e Controlador), separando a representação da informação da interação do usuário.

[2] *Data Access Object*: Padrão de projeto estrutural que abstrai e encapsula o acesso a uma fonte de dados (banco de dados), isolando a camada de persistência da lógica de negócio.

mento de software.

[Li et al. 2024] evidenciam ganhos significativos de produtividade com o uso de ferramentas como o *GitHub Copilot*, destacando que a automação de tarefas repetitivas permite aos desenvolvedores focar em aspectos mais criativos. [Norheim et al. 2024], contudo, alertam para riscos de dependência e para a importância da validação humana, argumentando que a produtividade deve considerar não apenas a velocidade, mas também a qualidade e manutenibilidade.

No campo de testes de software, [Tahvili and Hatvani 2020] e [Dias 2023] demonstram o potencial da IA para ampliar cobertura e eficiência, enquanto [Ståhlberg 2024] reforça que a supervisão humana continua essencial. Na documentação, [Bhosale et al. 2025] apontam que os LLMs podem reduzir o esforço manual, embora exijam revisão criteriosa.

Tabela 1 – Resumo dos Trabalhos Correlatos

Aplicação	**Ferramenta / Abordagem**	**Benefício**	**Limitação**	**Autor(es)**
Geração de Código	Ferramentas baseadas em LLMs (ex.: GitHub Copilot)	Redução do tempo em tarefas rotineiras de codificação.	Requer supervisão humana e validação constante.	Li (2024) Norheim (2024)
Otimização de Processos	Ferramentas analíticas baseadas em IA	Análise de dados históricos e detecção de gargalos.	Requer adaptação cultural, de processos e capacitação.	Ståhlberg (2024) Silva (2023)

3.5.1. Lacuna de Pesquisa e Diferenciação do Trabalho Atual

A análise dos trabalhos correlatos revela uma lacuna importante. A maioria dos estudos, como o de [Li et al. 2024] e os avaliados por [Norheim et al. 2024], foca no uso de assistentes de codificação para autocompletar trechos de código durante a digitação (*code completion*) ou na geração de artefatos isolados, como testes e documentação.

Diferente dessas abordagens, esta pesquisa propõe o uso da IA Generativa em uma etapa anterior e mais abrangente: a tradução de documentos de requisitos inteiros para uma estrutura de código inicial funcional. O diferencial deste trabalho reside na utilização da Engenharia de Prompts para estruturar a arquitetura do software na fase inicial (“Sprint 0”), preenchendo a lacuna semântica entre a especificação do analista e a implementação técnica, em vez de apenas acelerar a digitação de sintaxe.

4. Metodologia

Esta pesquisa adotou uma abordagem mista, combinando uma revisão sistemática da literatura com o desenvolvimento de um artefato experimental baseado em IA Generativa. A validação empírica ocorreu em ambiente controlado (*bootcamp*), permitindo a

coleta e análise de dados quantitativos e qualitativos sobre a produtividade no ciclo de desenvolvimento [Verner et al. 2008].

4.1. Abordagem da Pesquisa

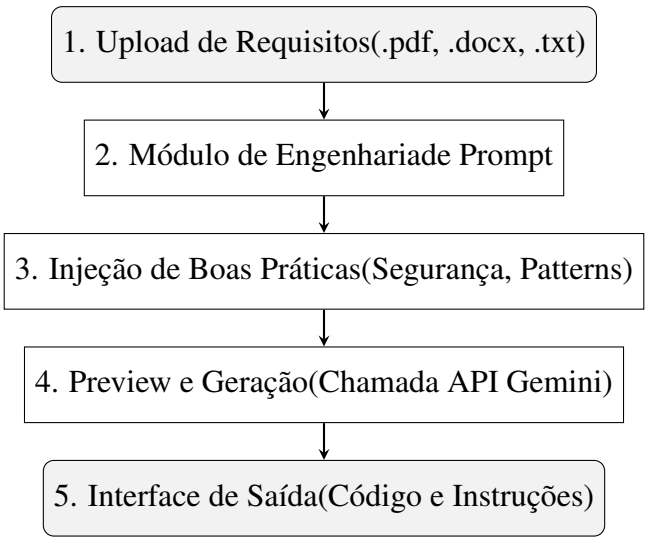
A pesquisa foi classificada como aplicada e experimental, por ter implementado e avaliado uma ferramenta concreta que utilizou IA generativa para transformar requisitos em código, medindo seu impacto direto na produtividade do desenvolvimento. A abordagem metodológica foi do tipo quali-quantitativa (mista), combinando elementos de investigação para obter uma compreensão mais abrangente do fenômeno estudado.

4.2. Desenvolvimento da Ferramenta Experimental

Foi desenvolvida uma ferramenta em Python que integra a API do modelo Gemini-2.5-Flash. O desenvolvimento seguiu os princípios da Prototipagem Experimental aplicada à pesquisa científica. A ferramenta atua como um orquestrador entre o usuário e o LLM, garantindo que a entrada de dados (requisitos) seja enriquecida com contexto técnico antes de ser processada. Para garantir a replicabilidade e transparência deste estudo, o código-fonte completo da ferramenta foi disponibilizado publicamente em repositório aberto[3].

Fluxo de Processamento

O fluxo de funcionamento da aplicação foi desenhado para eliminar a "página em branco" do desenvolvedor. Conforme ilustrado na Figura 1, o processo inicia-se com o upload do documento de requisitos (suportando formatos .docx, .pdf, .txt e .md). Em seguida, o sistema processa esse texto bruto e o insere em um *pipeline* de Engenharia de Prompt, onde são injetadas as regras de arquitetura e boas práticas.



Ao final do processamento, a interface apresenta o resultado dividido em duas seções: o lado esquerdo contendo os blocos de código gerado e o lado direito contendo

[3]Disponível em: `https://github.com/FilipeBrges/prompt-code-gen`

instruções detalhadas, dicas de implementação e explicações sobre as decisões arquiteturais tomadas pela IA.

Estrutura do Prompt Gerado

Um componente crítico da ferramenta é a montagem do *System Prompt*. Diferente de um chat comum, a ferramenta compila um "super-prompt" estruturado. O Quadro 1 apresenta um exemplo real da estrutura enviada ao modelo durante o experimento (partes extensas foram suprimidas para brevidade).

Quadro 1 – Estrutura do Prompt Montado pela Ferramenta (Exemplo Simplificado)

```
## Document Requirements
   [...]
   (Conteudo variavel extraido do documento de requisito do usuario)
   [...]

## Prompt Engineering Context
### Desenvolvedor Senior
   [...] (Define persona, stack tecnologica e contexto)

## Best Practices to Apply
### [Backend] Boas Praticas
- Separe o codigo por camadas (controladores, servicos, repositorios).
   [...] (Lista completa de padroes arquiteturais injetada aqui)

### [Geral] Seguranca
- Nunca armazene senhas em texto plano.
   [...] (Lista completa de validacoes de seguranca injetada aqui)

## Output Requirements
(Instrucoes ocultas para formatacao da resposta JSON/Markdown) [...]
```

A efetividade deste artefato foi avaliada com base em indicadores de eficiência e qualidade das tarefas geradas pela IA em comparação com as produzidas manualmente durante o *bootcamp*.

4.3. Planejamento e Execução do Bootcamp

Esta etapa correspondeu ao planejamento, organização e execução do *bootcamp*, que serviu como ambiente controlado para a coleta de dados empíricos da pesquisa. O *bootcamp* foi concebido para avaliar a efetividade do uso de uma ferramenta auxiliar no ciclo de desenvolvimento, permitindo comparar o desempenho de participantes que utilizaram a solução baseada em IA generativa com aqueles que realizaram o desenvolvimento manualmente.

Inicialmente, foi definido o escopo do projeto: o desenvolvimento de um Gerenciador de Tarefas, a ser implementado em Java, no console, com requisitos de persistência e validação, considerado de nível de dificuldade básico. A amostra foi composta por 6 duplas de programadores (totalizando 12 participantes), divididas equitativamente: 3 duplas no Grupo Experimental (Com IA) e 3 duplas no Grupo de Controle (Sem IA). O evento contou com a participação de estudantes de graduação do curso de Sistemas de Informação do Instituto Federal de Minas Gerais – Campus Ouro Branco, convidados por meio de divulgação aberta. A participação foi voluntária, com a maioria por estudantes

que já atuam na área de desenvolvimento com diferentes níveis de experiência, de modo a compor uma amostra diversificada e representativa.

O ambiente de desenvolvimento foi preparado com antecedência, contemplando todas as ferramentas necessárias para a execução das atividades, incluindo a configuração da ferramenta experimental, os editores de código e demais recursos de apoio. Foram elaborados documentos de requisitos, guias de uso da ferramenta e questionários de avaliação, de forma a padronizar o processo de coleta de dados.

O grupo experimental utilizou a ferramenta desenvolvida em Python integrada ao modelo Gemini-2.5-Flash para auxiliar na transformação de requisitos em código, realizando ajustes, refinamentos e documentando os resultados obtidos. Já o grupo de controle executou o mesmo projeto sem o auxílio da ferramenta, desenvolvendo o código de maneira tradicional, com base apenas nos requisitos fornecidos.

O *bootcamp* teve duração limite previamente definida, suficiente para permitir o desenvolvimento completo do projeto proposto. Durante sua execução, foram coletados dados quantitativos e qualitativos, incluindo: o tempo gasto em cada etapa; as métricas de desempenho; as principais dificuldades encontradas e as soluções adotadas pelos participantes. Ao final, foram aplicados questionários estruturados e realizadas entrevistas semiestruturadas para compreender a percepção dos participantes quanto à produtividade e experiência de uso. Além disso, foi conduzida uma avaliação técnica do código produzido por ambos os grupos, considerando critérios como qualidade, eficiência e aderência aos requisitos estabelecidos.

4.4. Análise dos Dados

A análise dos dados coletados durante e após o *bootcamp* foi conduzida por meio de uma abordagem mista, combinando métodos quantitativos e qualitativos para garantir uma compreensão abrangente dos efeitos do uso da ferramenta baseada em IA generativa no processo de desenvolvimento de software.

No componente quantitativo, buscou-se mensurar a produtividade e a qualidade técnica utilizando métricas consolidadas na literatura. Adotou-se a contagem de Linhas de Código (LOC) para mensurar a dimensão física do software produzido, pois, embora isoladamente não indique qualidade, quando analisada em conjunto com o tempo de desenvolvimento, serve como um indicador eficaz de produtividade bruta [Pressman 2014]. Complementarmente, utilizou-se a Complexidade Ciclomática, proposta por [McCabe 1976], que quantifica o número de caminhos linearmente independentes através do código fonte. A escolha desta métrica justifica-se por sua capacidade de indicar a testabilidade e o esforço de manutenção necessário: quanto maior a complexidade não gerenciada, maior a probabilidade de erros ocultos.

Além dessas métricas diretas, foram aplicados questionários estruturados com escala Likert ao final do *bootcamp*, a fim de mensurar percepções sobre o impacto da ferramenta na produtividade, facilidade de uso e satisfação geral. Os dados obtidos foram analisados por meio de métodos estatísticos apropriados, incluindo análise de correlação, testes de significância e avaliação de tendências, buscando validar a existência de diferenças significativas entre os grupos.

Já o componente qualitativo envolveu uma análise aprofundada dos cenários prá-

ticos de transformação de requisitos em código. Os participantes utilizaram a ferramenta em condições controladas que simularam contextos reais de desenvolvimento, e os resultados foram avaliados quanto à eficácia da geração de código funcional e de qualidade para diferentes tipos de requisitos. Entrevistas e questionários com perguntas abertas foram conduzidos para explorar as percepções subjetivas dos participantes sobre a experiência de uso da ferramenta, permitindo identificar padrões, dificuldades, benefícios percebidos e sugestões de aprimoramento.

Por fim, foi realizada a triangulação dos dados quantitativos e qualitativos, de modo a verificar convergências e divergências entre as evidências coletadas. Essa abordagem integrada fortaleceu as conclusões da pesquisa, possibilitando compreender não apenas se houve impacto na produtividade, mas também como esse impacto foi percebido e em quais condições se manifestou com maior intensidade, fornecendo subsídios para análises críticas e recomendações futuras.

Tabela 2 – Detalhamento das Métricas de Produtividade e Métodos de Coleta e Análise

Dimensão	**Métrica Específica**	**Tipo de Dado**	**Instrumento**	**Método de Análise**
Objetiva (Eficiência)	Tempo de Desenvolvimento	Quantitativo	Logs de Atividade	Estatística Descritiva
	Volume de Código e Complexidade (*LOC*, componentes)	Quantitativo	Ferramentas de Análise Estática	Estatística Descritiva
	Qualidade do Código (Defeitos, Manutenibilidade)	Quantitativo	Ferramentas de Análise	Análise de Defeitos
Subjetiva (Experiência)	Esforço Cognitivo Percebido	Qualitativo / Likert	Questionários (*Likert*)	Estatística (*Likert*)
	Satisfação do Desenvolvedor	Qualitativo / Likert	Questionários (*Likert*)	Estatística (*Likert*)
	Dificuldades e Benefícios	Qualitativo	Entrevistas e Perguntas Abertas	Entrevistas
Processo (Contexto)	Interação com o requisito recebido	Qualitativo	Observação e Perguntas Abertas	Entrevistas

4.5. Limitações Metodológicas

Reconhecem-se as seguintes limitações metodológicas: Variabilidade na experiência e na proficiência dos desenvolvedores com as ferramentas; Possível viés de seleção na amostra de participantes; A rápida evolução das ferramentas de IA pode tornar alguns resultados rapidamente desatualizados; Potenciais vulnerabilidades de segurança no código-fonte gerado pela ferramenta, cuja identificação e mitigação dependem diretamente da atuação dos participantes; Dificuldade em

isolar o impacto específico das ferramentas de IA de outros fatores que afetam a produtividade. Essas limitações serão consideradas na análise dos resultados e nas conclusões da pesquisa.

4.6. Ética na Pesquisa

A pesquisa seguirá os princípios éticos recomendados para estudos com seres humanos, garantindo o uso responsável de todos os dados coletados. Caso seja necessário obter opiniões ou feedbacks dos participantes, será aplicado um Termo de Consentimento Livre e Esclarecido (TCLE), assegurando anonimato e confidencialidade, além de transparência quanto aos objetivos e métodos da pesquisa.

5. Resultados e Discussão

Este capítulo apresenta a análise integrada dos dados coletados. A investigação é conduzida através de três eixos: análise objetiva (métricas de engenharia), subjetiva (esforço e satisfação) e triangulação qualitativa.

5.1. Análise Objetiva: Desempenho e Qualidade Técnica

A avaliação objetiva mensurou o produto final entregue pelos dois grupos (Com IA e Sem IA). Os resultados consolidados de eficiência e qualidade são apresentados na Tabela 3.

Tabela 3 – Consolidado das métricas objetivas de eficiência e qualidade

Métrica	**Sem IA (média)**	**Com IA (média)**	**Variação**
Eficiência e Estrutura			
Tempo de Desenv. (min)	75.00	24.33	-67.56%
LOC Médio	90.00	107.00	+18.89%
Complexidade Ciclomática	27.00	34.33	+27.16%
Qualidade Técnica (Escala 1-5)			
Corretude	2.7	3.7	+37.50%
Legibilidade	4.0	4.7	+16.67%
Manutenibilidade	2.7	4.3	+62.50%
Aderência a padrões	3.3	4.3	+30.00%
Tratamento de erros	3.3	5.0	+50.00%

5.1.1. Eficiência Temporal e Estrutura

A redução de 67,56% no tempo de desenvolvimento (Tabela 3) sugere-se que a IAG atua como um acelerador de "arranque", eliminando o tempo gasto com *boilerplate*. O aumento nas métricas de complexidade e LOC no grupo com IA não representa desorganização, mas sim uma maior sofisticação arquitetural, decorrente da inclusão automática de rotinas robustas de tratamento de erros e modularização.

5.1.2. Avaliação da Qualidade (QA)

A variação de +62,50% em Manutenibilidade é o dado mais expressivo. A análise individual revelou que 66% dos projetos do Grupo Sem IA apresentaram falhas críticas (ex: erros de *Scanner* ou lógica de IDs). Em contraste, o código do Grupo Com IA foi consistentemente funcional, com o Grupo 6 atingindo nota máxima (5/5) devido a uma arquitetura desacoplada que nenhum grupo manual replicou no tempo disponível.

5.2. Análise Subjetiva: O Fator Humano

Os dados subjetivos (Tabela 4) indicam que a IAG altera drasticamente a experiência de desenvolvimento.

Tabela 4 – Consolidado das métricas subjetivas de experiência

Indicador	Sem IA	Com IA	Impacto
Esforço Cognitivo (1-5)	2.38	1.38	-42.02%
Satisfação (1-5)	4.08	4.67	+14.46%

A redução de 42% no esforço cognitivo sugere que a IA atua como um "amortecedor", absorvendo a complexidade da implementação. Embora o grupo sem IA tenha reportado alta satisfação (4.08), a análise qualitativa indica um contentamento "apesar do atrito". Já a média de 4.67 do grupo com IA reflete uma jornada fluida. Portanto, a satisfação superior é uma consequência direta da eficiência técnica: melhores resultados com menor desgaste.

5.3. Triangulação Qualitativa: A Mudança na Natureza do Trabalho

A análise cruzada dos dados, realizada pelo confronto entre as métricas quantitativas de desempenho e os relatos qualitativos dos participantes, permitiu identificar as causas raízes das discrepâncias numéricas observadas. Essa triangulação metodológica foi fundamental para compreender não apenas a magnitude dos ganhos de produtividade, mas também a natureza da transformação no fluxo de trabalho operada pela introdução da IA, revelando nuances comportamentais que os números isolados não capturariam.

5.3.1. De Codificação para Auditoria e Aprendizado

Os relatos indicam que a IAG deslocou o esforço de *implementação* para *supervisão* e aprendizado. No Grupo Sem IA, o bloqueio foi causado por micro-problemas (sintaxe, lógica) e pelo atrito de "ter que pesquisar como fazer". No Grupo Com IA, o desafio foi "escolher prompts" e "entender o código gerado". Importante notar que participantes destacaram o auxílio no "[...] entendimento de funcionalidades não dominadas [...]", sugerindo um papel de tutor técnico da ferramenta.

5.3.2. O Impacto da Escrita Manual na Qualidade

A triangulação revela que o processo manual foi a causa da menor qualidade objetiva. A citação "*[. . .] ao fazer sem IA tive que me preocupar mais [. . .] acabamos cometendo alguns erros*

[...]" ilustra que a "preocupação" (esforço cognitivo) induziu ao erro humano. A IA eliminou esse erro de sintaxe, entregando um artefato estruturado para refino.

5.4. Validação Estatística e Limitações

Aplicou-se o Teste-t de Student às métricas. Embora as médias do grupo com IA sejam superiores, o tamanho da amostra ($N = 3$ por grupo) resultou em $p > 0.05$ (ex: $p = 0.15$ para Manutenibilidade), indicando ausência de significância estatística estrita. Assim, os resultados devem ser interpretados como fortes evidências qualitativas e tendências de desempenho, validadas pela consistência entre dados objetivos e relatos, mas demandando estudos maiores para generalização.

6. Conclusão

Este estudo investigou sistematicamente o impacto da Inteligência Artificial Generativa no ciclo de desenvolvimento de software, buscando ir além das métricas superficiais de velocidade. Os dados confirmaram a hipótese inicial de que a ferramenta aumenta a produtividade, mas revelaram uma nuance crítica: o ganho real não provém apenas da aceleração da escrita de código, mas sim da drástica eliminação do atrito cognitivo durante a tradução de requisitos em soluções técnicas.

6.1. A IAG como Multiplicador de Qualidade

Contrariando a noção de que código gerado por IA tem baixa qualidade, o experimento mostrou que, para tarefas definidas, a IA produziu artefatos estruturalmente superiores (+62,50% em manutenibilidade), garantindo padrões e tratamento de erros frequentemente negligenciados sob pressão manual.

6.1.1. A IA na Resolução de Ambiguidades

Os resultados corroboram a teoria sobre ambiguidade de requisitos apresentada na fundamentação. O grupo sem IA apresentou maior incidência de falhas lógicas (nota 2.7 em Corretude), o que indica que a interpretação humana dos requisitos variou, introduzindo erros subjetivos.

A ferramenta, ao aplicar a Engenharia de Prompt, forçou uma interpretação padronizada. Isso responde à questão de como a IA resolve a ambiguidade: ela atua cristalizando as definições vagas em instruções de código precisas antes que o desenvolvedor humano possa introduzir viés ou erro de interpretação. O aumento de 62,50% na manutenibilidade é evidência direta de que requisitos bem estruturados via prompt resultam em código de maior qualidade técnica.

6.2. O Alívio da Carga Cognitiva

A conclusão mais relevante reside na dimensão humana. A redução de 42% no esforço cognitivo valida a percepção de que "*[...] ao fazer sem IA tive que me preocupar mais [...]*". A análise revelou que essa "preocupação" advém tanto da escrita quanto da pesquisa manual. A IA eliminou esse atrito, acelerando a entrega (-67,56% no tempo) e aumentando a satisfação (+14%).

6.3. Implicações Éticas e a Responsabilidade Humana

Os resultados positivos de produtividade não devem obscurecer as implicações éticas do uso da IA Generativa, sendo fundamental estabelecer que o código gerado pela ferramenta proposta não constitui um produto final, mas sim uma "minuta técnica" ou um esboço funcional avançado. Como a IA opera como uma caixa preta estatística sujeita a "alucinações" lógicas e

vulnerabilidades de segurança, a ausência de plágio direto na geração probabilística não elimina a necessidade de validação rigorosa. Dessa forma, a adoção dessa tecnologia altera o perfil de responsabilidade do desenvolvedor, que deixa de ser apenas o autor da sintaxe para se tornar o curador ético e técnico da solução, permanecendo como o único responsável legal e moral por garantir que o software implantado respeite integralmente as normas de privacidade, segurança e propriedade intelectual da organização.

6.4. Considerações Finais

Conclui-se que a IAG transforma o desenvolvedor de "artesão de código" para "arquiteto e auditor". Ao assumir essa postura de validação, o desenvolvedor se aproxima das funções de análise e teste, garantindo que a solução técnica permaneça fiel às especificações originais do negócio.

Portanto, a IA Generativa aumentou a produtividade em tempo, qualidade e experiência, confirmando integralmente a hipótese proposta. Para a indústria, isso sinaliza um aumento de capacidade produtiva através da capacitação tecnológica, permitindo soluções mais robustas com menor desgaste mental. Contudo, é fundamental considerar quais riscos essa mudança de paradigma traz, especialmente quanto à dependência tecnológica e à manutenção da competência técnica humana.

Diante disso, sugere-se para pesquisas futuras a expansão da amostra para conferir maior robustez estatística aos dados, bem como a aplicação da abordagem em projetos de maior duração e complexidade. Recomenda-se, ainda, a análise específica dos riscos de dependência tecnológica a longo prazo e a avaliação da ferramenta e da metodologia em domínios de negócio específicos.

Referências

[Bang et al. 2023] Bang, Y., Cahyawijaya, S., Kaye, N., Zhuang, W., Lau, J., et al. (2023). A multitask, multilingual, multimodal evaluation of chatgpt on reasoning, hallucination, and interactivity. *arXiv preprint arXiv:2302.04023*. Disponível em: https://www.afnlp.org/conferences/ijcnlp2023/proceedings/main-long/cdrom/pdf/2023.ijcnlp-long.45.pdf. Acesso em: 28 novembro 2025.

[Bhosale et al. 2025] Bhosale, P., Shinde, S., and Nikam, S. (2025). Ai based code documentation generation. *International Journal of Innovative Research in Technology*, 11(7):1–12. Disponível em: https://ijirt.org/publishedpaper/IJIRT172635_PAPER.pdf. Acesso em: 20 maio 2025.

[Center for Security and Emerging Technology (CSET) 2023] Center for Security and Emerging Technology (CSET) (2023). What are generative ai, large language models, and foundation models? Georgetown University. Disponível em: https://cset.georgetown.edu/article/what-are-generative-ai-large-language-models-and-foundation-models/. Acesso em: 26 maio 2025.

[Dias 2023] Dias, J. C. (2023). Teste de software com ia: Um mapeamento sistemático da literatura. Master's thesis, Universidade Federal de Pernambuco, Recife. Disponível em: https://repositorio.ufpe.br/handle/123456789/50401. Acesso em: 20 maio 2025.

[Jiang et al. 2024] Jiang, J. et al. (2024). A survey on large language models for code generation. *arXiv preprint arXiv:2406.00515*. Disponível em: https://arxiv.org/abs/2406.00515. Acesso em: 23 jul. 2025.

[Li et al. 2024] Li, M. M., Dickhaut, E., Bruhin, O., Wache, H., and Weritz, P. (2024). More than just efficiency: Impact of generative ai on developer productivity. In *Americas Conference on Information Systems (AMCIS)*. Disponível em: https://aisel.aisnet.org/cgi/viewcontent.cgi?article=1371&context=amcis2024. Acesso em: 13 maio 2025.

[McCabe 1976] McCabe, T. J. (1976). A complexity measure. *IEEE Transactions on software engineering*, (4):308–320. Disponível em: http://www.literateprogramming.com/mccabe.pdf. Acesso em: 28 novembro 2025.

[Naveed et al. 2024] Naveed, H., Khan, A. U., Qiu, S., Saqib, M., Anwar, S., Usman, M., Akhtar, N., Barnes, N., and Mian, A. (2024). A comprehensive overview of large language models. *arXiv preprint arXiv:2307.06435v10*. Disponível em: https://arxiv.org/abs/2307.06435. Acesso em: 26 maio 2025.

[Norheim et al. 2024] Norheim, J. J., Rebentisch, E., Xiao, D., Draeger, L., Kerbrat, A., and Weck, O. L. D. (2024). Challenges in applying large language models to requirements engineering tasks. *Design Science*, 10:e13. Disponível em: https://www.cambridge.org/core/journals/design-science/article/challenges-in-applying-large-language-models-to-requirements-engineering-tasks/1FC7666F0A0B4E7091D2D4B2D46321B5. Acesso em: 20 maio 2025.

[Pressman 2014] Pressman, R. S. (2014). *Software Engineering: A Practitioner's Approach*. McGraw-Hill Education, 8 edition. Disponível em: https://lib.zu.edu.pk/ebookdata/Engineering/Data%20Science/Software%20Engineering_%20A%20P

ractitioner%E2%80%99s%20Approach%20by%20Roger%20S.%20Pressman_%20Bruce%20R.%20Maxin%20-McGraw-Hill%20Education%20(2014).pdf. Acesso em: 28 novembro 2025.

[Sommerville 2011] Sommerville, I. (2011). *Software Engineering*. Addison-Wesley, 9 edition. Disponível em: https://engineering.futureuniversity.com/BOOKS%20FOR%20IT/Software-Engineering-9th-Edition-by-Ian-Sommerville.pdf. Acesso em: 28 novembro 2025.

[Ståhlberg 2024] Ståhlberg, V. (2024). Enhancing software development processes with artificial intelligence. Master's thesis, Universidade de Turku, Finlândia. Disponível em: https://www.utupub.fi/bitstream/10024/179187/1/Stahlberg_Vili_opinnayte.pdf. Acesso em: 20 maio 2025.

[Tahvili and Hatvani 2020] Tahvili, S. and Hatvani, L. (2020). *Artificial Intelligence Methods for Optimization of the Software Testing Process*. Elsevier, Amsterdam. Disponível em: https://www.sciencedirect.com/book/9780323919135/artificial-intelligence-methods-for-optimization-of-the-software-testing-process. Acesso em: 13 maio 2025.

[Vaswani et al. 2017] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2017). Attention is all you need. In *Advances in Neural Information Processing Systems 30 (NIPS 2017)*. Disponível em: https://arxiv.org/abs/1706.03762. Acesso em: 27 maio 2025.

[Verner et al. 2008] Verner, J., Sampson, J., and Cerpa, N. (2008). What factors lead to software project failure? In *2008 Second International Conference on Research Challenges in Information Science*, pages 71–80. IEEE. Disponível em: https://ieeexplore.ieee.org/document/4632095/. Acesso em: 23 jul. 2025.

[White et al. 2023] White, J., Fu, Q., Hays, S., Sandborn, M., Olea, C., Bourne, H., and Schmidt, D. C. (2023). A prompt pattern catalog to enhance prompt engineering with chatgpt. *arXiv preprint arXiv:2302.11382*. Disponível em: https://doi.org/10.48550/arXiv.2302.11382. Acesso em: 28 novembro 2025.