CENTRO FEDERAL DE EDUCAÇÃO TECNOLÓGICA CELSO SUCKOW DA FONSECA
CURSO DE SISTEMAS DE INFORMAÇÃO

UTILIZAÇÃO DA INTELIGÊNCIA ARTIFICIAL COMO FERRAMENTA DE APOIO AO DESENVOLVIMENTO DE SOFTWARE SEGURO

MUSTAFA RIBEIRO DE ALMEIDA NETO

Orientador: Nilson Mori Lazarin
CEFET/RJ Campus Nova Friburgo

NOVA FRIBURGO
AGOSTO - 2025

MUSTAFA RIBEIRO DE ALMEIDA NETO

UTILIZAÇÃO DA INTELIGÊNCIA ARTIFICIAL COMO FERRAMENTA DE APOIO AO DESENVOLVIMENTO DE SOFTWARE SEGURO

Trabalho de Conclusão de Curso (Graduação) apresentado ao Curso de Sistemas de Informação do Centro Federal de Educação Tecnológica Celso Suckow da Fonseca, como requisito parcial para a obtenção do título de Bacharel em Sistemas de Informação.

Orientador: Nilson Mori Lazarin
CEFET/RJ Campus Nova Friburgo

CENTRO FEDERAL DE EDUCAÇÃO TECNOLÓGICA CELSO SUCKOW DA FONSECA
CURSO DE SISTEMAS DE INFORMAÇÃO
NOVA FRIBURGO
AGOSTO - 2025

UTILIZAÇÃO DA INTELIGÊNCIA ARTIFICIAL COMO FERRAMENTA DE APOIO AO DESENVOLVIMENTO DE SOFTWARE SEGURO

Monografia apresentada ao Curso de Sistemas de Informação do Centro Federal de Educação Tecnológica Celso Suckow da Fonseca, como requisito parcial para a obtenção do título de Bacharel.

MUSTAFA RIBEIRO DE ALMEIDA NETO

Banca Examinadora:

gov.br Documento assinado digitalmente
NILSON MORI LAZARIN
Data: 02/09/2025 11:02:09-0300
Verifique em https://validar.iti.gov.br

Presidente, Prof. Me. Nilson Mori Lazarin (CEFET/RJ) (Orientador(a))

gov.br Documento assinado digitalmente
LEONARDO PIO VASCONCELOS
Data: 28/08/2025 10:54:12-0300
Verifique em https://validar.iti.gov.br

Prof. Dr. Leonardo Pio Vasconcelos (UFPI)

gov.br Documento assinado digitalmente
RAFAEL ELIAS DE LIMA ESCALFONI
Data: 28/08/2025 10:41:23-0300
Verifique em https://validar.iti.gov.br

Prof. Dr. Rafael Elias de Lima Escalfoni (CEFET/RJ)

RAFAEL GUIMARAES RODRIGUES:04387178728

Assinado de forma digital por RAFAEL GUIMARAES RODRIGUES:04387178728
Dados: 2025.09.01 12:18:43 -03'00'

Prof. Me. Rafael Guimarães Rodrigures (CEFET/RJ)

NOVA FRIBURGO

AGOSTO 2025

CEFET/RJ – Sistema de Bibliotecas / Biblioteca Uned Nova Friburgo

A447u Almeida Neto, Mustafa Ribeiro de.
Utilização da inteligência artificial como ferramenta de apoio ao desenvolvimento de software seguro. / Mustafa Ribeiro de Almeida Neto. – Nova Friburgo, RJ : 2025.
xii, 37f.: il. (color.) : em PDF.

Trabalho de Conclusão de Curso (Curso e Sistemas de Informação) - Centro Federal de Educação Tecnológica Celso Suckow da Fonseca, 2025.
Bibliografia: f. 35-37.
Orientador: Nilson Mori Lazarin.

1. Sistemas de Informação. 2. Desenvolvimento de Software. 3. Inteligência Artificial Generativa. I. Lazarin, Nilson Mori (Orientador). II. Título.

CDD 658.4038

Elaborada pela bibliotecária Cristina Rodrigues Alves CRB7/5932

Dedico este trabalho aos meus pais, Mustafa Almeida e Raquel Almeida.

Agradecimentos

Agradeço, em primeiro lugar, aos meus pais, Mustafa Almeida e Raquel Almeida, pelo amor incondicional e incentivo em todas as etapas da minha vida. Às minhas irmãs, Jéssica e Halana, que sempre estiveram presentes oferecendo suporte, palavras de encorajamento e companheirismo nos momentos mais desafiadores.

À Carla Schaus, por ser uma presença essencial na minha vida, com seu carinho e apoio inabalável. Sou grato também aos amigos que conquistei durante essa caminhada acadêmica, pelos momentos compartilhados, pela troca de conhecimentos e pelo incentivo mútuo.

Agradeço especialmente ao meu orientador, professor Nilson Mori Lazarin, pela dedicação, orientação e contribuições valiosas que foram fundamentais para o desenvolvimento deste trabalho.

“A segurança não é um produto, mas um processo.” (Bruce Schneier)

Resumo

A crescente complexidade do desenvolvimento de software e a evolução das ameaças cibernéticas exigem a integração da segurança em todas as fases do ciclo de vida do software. Este trabalho investiga o uso de Inteligência Artificial Generativa como ferramenta de apoio em tempo real para o desenvolvimento seguro. Para isso, foi criada uma extensão para o Visual Studio Code que integra múltiplos provedores de IA para analisar trechos de código, identificar vulnerabilidades e sugerir correções. A eficácia da abordagem foi validada por meio de dois experimentos: uma prova de conceito, com análise de códigos reconhecidamente vulneráveis e comparação de vulnerabilidades analisadas por desenvolvedores, e um estudo de caso no sistema de código aberto WeGIA. Os resultados indicam que a integração da IA no ambiente de desenvolvimento é uma estratégia promissora para reforçar a segurança do software, atuando como recurso complementar à revisão humana especializada.

Palavras-chave: Desenvolvimento Seguro de Software. Inteligência Artificial Generativa. Análise de Vulnerabilidades. Ambiente de Desenvolvimento Integrado.

Abstract

The increasing complexity of software development and the continuous evolution of cybersecurity threats demand the integration of security throughout the entire software development lifecycle. This work investigates the use of Generative Artificial Intelligence as a real-time support tool for secure software development. To this end, an extension for Visual Studio Code was developed, integrating multiple AI providers to analyze code snippets, identify vulnerabilities, and suggest corrections. The effectiveness of this approach was validated through two experiments: a proof of concept, which included the analysis of known vulnerable code and a comparison of vulnerabilities identified by developers, and a case study conducted on the open-source system WeGIA. The results indicate that integrating AI into the development environment is a promising strategy to enhance software security, serving as a complementary resource to expert human review.

Keywords: Secure Software Development. Generative Artificial Intelligence. Vulnerability Analysis. Integrated Development Environment.

Lista de Figuras

Lista de Abreviaturas e Siglas

IA	Inteligência Artificial
SSI	Segurança de Sistemas da Informação
GenAI	Inteligência Artificial Generativa (Generative Artificial Intelligence)
IDE	Ambiente de Desenvolvimento Integrado (Integrated Development Environment)
LLM	Modelos de Linguagem de Grande Escala (Large Language Models)
VSCode	Visual Studio Code

Sumário

1 Introdução

Diante da crescente integração digital e da interconexão de dados, onde praticamente todas as transações e atividades são realizadas de forma virtual, a informação tornou-se um dos ativos mais valiosos para as organizações. A quantidade e a velocidade com que os dados são gerados, transmitidos e armazenados são impressionantes. Os dados gerados por redes sociais, comércio eletrônico e Internet das Coisas geram aproximadamente 2,5 quintilhões de bytes por dia, ou quase 30.000 GB por segundo (BAILLY; MEYFROIDT; TIMSIT, 2017). Esse ambiente de alta conectividade e interdependência amplia a necessidade de proteção da informação contra diversos riscos, desde ataques cibernéticos até falhas de segurança internas.

A Segurança de Sistemas da Informação (SSI) é obtida pela implementação de um conjunto adequado de controles, incluindo políticas, processos, procedimentos, estrutura organizacional e funções de software e hardware (ABNT, 2013). Dessa forma, a segurança de software desempenha um papel fundamental na proteção da informação. Durante o desenvolvimento de software, etapas como coleta de requisitos, análise, design, implementação e teste não apenas garantem a funcionalidade e a qualidade do software, mas também são oportunidades para integrar medidas de segurança (GLUSHKOVA, 2023).

A implementação de medidas de segurança durante a fase de design do desenvolvimento de software é crucial, pois é neste estágio que as bases para um sistema seguro são estabelecidas. Incorporar segurança desde o início pode significativamente reduzir os custos associados a correções tardias de vulnerabilidades (HALKIDIS *et al.*, 2008). No entanto, a complexidade e o custo inicial dessas medidas podem ser um desafio para muitas organizações, especialmente aquelas com recursos limitados. Dessa forma, a Inteligência Artificial (IA) pode desempenhar um papel transformador, oferecendo soluções que automatizam a identificação e a mitigação de riscos, tornando a segurança uma parte integral e contínua do ciclo de desenvolvimento de software.

A IA tem transformado o dia a dia das pessoas e organizações, revelando-se uma ferramenta valiosa capaz de auxiliar em diversas tarefas (ELIAS, 2023). Em se tratando de SSI, a IA pode ser aplicada de diversas formas, seja na análise de tráfego de rede, seja na análise de logs, no reconhecimento de arquivos maliciosos, ou ainda na busca por vulnerabilidades (ALMEIDA; SILVA, 2020).

Neste contexto, este trabalho apresentará uma abordagem de integração do ambiente de desenvolvimento com IA Generativa, visando apoiar os desenvolvedores em tempo de design, de forma que estes possam produzir códigos mais seguros.

1.1 Motivação

A Inteligência Artificial representa uma das maiores revoluções da era digital e vem transformando significativamente diversas profissões e setores. Na área da saúde, por exemplo, a IA tem sido aplicada com sucesso no suporte ao diagnóstico, na personalização de tratamentos, na triagem de pacientes com base em riscos e até mesmo na automação de tarefas burocráticas por meio do reconhecimento de voz e geração de texto. Ferramentas baseadas em IA generativa, como chatbots, também têm sido utilizadas para fornecer orientações personalizadas aos pacientes, além de apoiar profissionais na redação de prontuários e relatórios clínicos de forma eficiente (VIDAL-ALABALL *et al.*, 2024). Da mesma forma, no contexto do desenvolvimento de software, essas capacidades podem ser adaptadas para auxiliar desenvolvedores na identificação de vulnerabilidades e na tomada de decisões para melhorar a segurança do software.

1.2 Definição do Problema

Com o avanço da transformação digital e o uso massivo das redes sociais, as organizações estão cada vez mais expostas a riscos cibernéticos que podem afetar diretamente sua reputação. Segundo Maskuri, o uso inadequado das redes sociais e as falhas de segurança estão entre os principais fatores que aumentam o risco reputacional, exigindo das organizações um investimento contínuo em práticas de cibersegurança (MASKURI *et al.*, 2023).

Esse cenário se torna ainda mais alarmante ao considerar que o Brasil foi classificado como o segundo país mais vulnerável a ataques cibernéticos, registrando mais de 7 bilhões de ameaças apenas no primeiro semestre de 2023 (Trend Micro, 2023). Por isso, é fundamental aprimorar as práticas de desenvolvimento que incorporem a segurança de software desde o início do ciclo de vida, focando na prevenção, identificação e mitigação de vulnerabilidades antes que possam ser exploradas.

Nos últimos anos, os ataques cibernéticos tornaram-se mais sofisticados, explorando vulnerabilidades em hardware, software, redes e dispositivos móveis. A popularização de dispositivos conectados, como smartphones e tecnologias de Internet das Coisas (IoT), ampliou consideravelmente a superfície de ataque. Além disso, práticas como o hacking as a service (HaaS) têm facilitado o acesso a ferramentas automatizadas de ataque, permitindo até mesmo que atores com pouco conhecimento técnico executem ações maliciosas altamente impactantes. Estima-se que esses ataques causem prejuízos de bilhões de dólares por ano no mundo todo (ASLAN *et al.*, 2023).

Uma das principais causas para a recorrência dessas brechas em segurança é a formação insuficiente dos profissionais da área. Segundo Costa, muitos egressos de cursos

superiores não se sentem preparados para lidar com os desafios de segurança de software, o que contribui para a ocorrência de erros críticos durante o desenvolvimento (COSTA *et al.*, 2018).

O avanço das tecnologias de Inteligência Artificial Generativa (GenAI) representa uma abordagem promissora e transformadora no apoio ao desenvolvimento de software seguro. Soluções baseadas em GenAI têm o potencial de auxiliar desenvolvedores durante suas atividades, fornecendo análises automatizadas e sugestões para a mitigação de vulnerabilidades — inclusive para profissionais com menor nível de experiência. Diante desse cenário, este trabalho busca responder à seguinte questão de pesquisa:

"Como a utilização de inteligência artificial pode apoiar, em tempo real, as práticas de desenvolvimento de código seguro?"

1.3 Objetivos

O principal objetivo deste trabalho é explorar como a integração da Inteligência Artificial no ambiente de desenvolvimento pode aprimorar a segurança do software. Para alcançar este objetivo, o trabalho se desdobra nos seguintes objetivos específicos:

- Desenvolver uma extensão para o ambiente de desenvolvimento que se integre a um modelo de IA Generativa para fornecer sugestões de segurança durante o design do software.
- Realizar um estudo de caso para avaliar a eficácia da abordagem proposta, analisando como as sugestões de segurança fornecidas pela IA influenciam o processo de desenvolvimento e a qualidade do código.
- Comparar a eficácia da abordagem proposta, baseada em IA, com a revisão de segurança tradicional realizada por uma equipe de especialistas humanos para identificar suas vantagens e limitações.

1.4 Metodologia

Inicialmente, realizou-se o levantamento de requisitos a partir das demandas identificadas na disciplina de Segurança e Auditoria de Sistemas, o que deu origem ao design e à especificação funcional da extensão MN-Analise para o Visual Studio Code. Em seguida, procedeu-se ao desenvolvimento da ferramenta, implementando a integração com provedores de IA generativa de modo a viabilizar comparações de segurança em diferentes cenários.

A segunda fase foi a de validação experimental, projetada para avaliar a eficácia da extensão. Para isso, foi realizada uma prova de conceito utilizando um conjunto de códigos

reconhecidamente vulneráveis, extraídos de repositórios públicos. A análise consistiu em comparar as vulnerabilidades detectadas pela ferramenta com as falhas conhecidas e, também, comparar com a capacidade de detecção de desenvolvedores humanos.

A terceira fase correspondeu à aplicação em um estudo de caso, onde a ferramenta foi utilizada para analisar o código-fonte do sistema WeGIA, um software de código aberto. Nesta etapa, foram coletadas métricas de desempenho da IA, como Verdadeiros Positivos, Falsos Positivos e Falsos Negativos, para avaliar a precisão e a revocação da ferramenta na identificação de diferentes tipos de vulnerabilidades em um ambiente de desenvolvimento real. A análise dos resultados permitiu uma avaliação objetiva das vantagens e limitações da abordagem proposta.

1.5 Estrutura

Este trabalho está organizado em sete capítulos, cada um com um propósito específico para desenvolver de forma clara e coerente a pesquisa sobre a utilização da inteligência artificial como ferramenta de apoio ao desenvolvimento de software seguro. No Capítulo 1, são apresentados o contexto, a motivação, os objetivos e a metodologia deste estudo. Em seguida, o Capítulo 2 revisa os principais conceitos de Inteligência Artificial, IA Generativa e Ambientes de Desenvolvimento Integrado (IDE), fornecendo a base teórica necessária para a compreensão das etapas subsequentes.

O Capítulo 3 descreve o protocolo de Mapeamento Sistemático da Literatura (MSL) e analisa criticamente os estudos relevantes que investigam abordagens semelhantes. No Capítulo 4, detalha-se a arquitetura e a implementação da extensão MN-Analise para o VSCode, incluindo as integrações com diferentes provedores de IA (OpenAI, Gemini e Ollama) e o fluxo de funcionamento interno da ferramenta.

A Prova de Conceito é apresentada no Capítulo 5, onde são descritos os experimentos conduzidos com códigos reconhecidamente vulneráveis e a análise comparativa dos resultados obtidos pela extensão em relação ao desempenho de desenvolvedores humanos. No Capítulo 6, a extensão é aplicada ao sistema WeGIA, demonstrando seu comportamento em um ambiente real, detalhando critérios de avaliação, distribuição das vulnerabilidades identificadas e resultados práticos.

Por fim, o Capítulo 7 sintetiza os principais achados, discute as contribuições científicas e práticas deste trabalho e aponta direções para pesquisas futuras, encerrando o trabalho com reflexões sobre o impacto da IA na segurança de software.

2 Fundamentação Teórica

A integração de IA em ambientes de desenvolvimento visa fornecer aos desenvolvedores sugestões automatizadas para melhorar a segurança do software durante o processo de desenvolvimento. Essa integração é realizada por meio de extensões específicas que utilizam modelos de IA para analisar o código e identificar possíveis vulnerabilidades. Neste capítulo, abordaremos os conceitos fundamentais que sustentam essa abordagem, incluindo Desenvolvimento de Software, Ambientes de Desenvolvimento Integrado (IDE), Inteligência Artificial (IA) e IA Generativa.

2.1 Desenvolvimento de Software

O desenvolvimento de software pode ser entendido como o conjunto de processos nos quais componentes de código são planejados, implementados, refinados e integrados para formar um produto de software. Esse processo envolve não apenas a criação de novos sistemas, mas também a modernização e integração de soluções já existentes, sempre sob o desafio de atender metas previamente definidas de qualidade, eficiência e prazo. O desenvolvimento de software está tradicionalmente associado ao Ciclo de Vida do Desenvolvimento de Software (SDLC), que organiza essas atividades em etapas bem definidas, garantindo que o produto final seja entregue dentro de padrões estabelecidos e em conformidade com os requisitos técnicos e de negócio (STAVRIDIS; DRUGGE, 2023).

O Ciclo de Vida do Desenvolvimento de Software é composto por etapas que estruturam todo o processo, desde a concepção até a manutenção do sistema. Essas fases incluem o planejamento, no qual são definidos os objetivos do projeto; a análise de requisitos, que identifica as necessidades do usuário; o design, responsável pela modelagem da solução; a codificação e testagem, que envolvem a implementação prática e a verificação de erros; a implantação, em que o software é disponibilizado para uso; e, por fim, a manutenção, que garante sua evolução e correção ao longo do tempo. A Figura 1 ilustra esse processo em forma cíclica, destacando a interdependência entre as etapas e a necessidade de revisões constantes para assegurar a qualidade e a adequação do produto desenvolvido (KLOPPER; GRUNER; KOURIE, 2007).

Figura 1 – Ciclo de Vida do Desenvolvimento de Software

2.2 Ambiente de Desenvolvimento Integrado (IDE)

Ambientes de Desenvolvimento Integrado (IDE) desempenham um papel fundamental no desenvolvimento de software há muitos anos, atuando como ferramentas que integram diversos recursos essenciais para a criação e manutenção de aplicações. IDEs modernas oferecem funcionalidades como compilação contínua, testes automatizados, depuração integrada e refatoração de código, contribuindo significativamente para o aumento da produtividade dos desenvolvedores. Além disso, funcionam como um kit de ferramentas abrangente projetado especificamente para facilitar todo o processo de concepção e desenvolvimento de software, ao consolidar em uma única interface gráfica recursos que anteriormente estariam dispersos em várias ferramentas separadas, promovendo maior eficiência e integração ao fluxo de trabalho do desenvolvedor (SHUKLA, 2024).

Os IDEs também se destacam por integrarem funcionalidades que auxiliam os programadores a rastrear erros e organizar melhor o fluxo de desenvolvimento. Entre essas funcionalidades, destaca-se a integração com sistemas de controle de versão, como o Git, que pode ser utilizado como uma extensão do ambiente. Essa integração torna o processo de colaboração mais seguro e eficiente, permitindo o gerenciamento de mudanças, a criação de ramificações para desenvolvimento paralelo e a reversão de alterações problemáticas sempre que necessário (HöRNEMALM, 2023).

As extensões, portanto, configuram-se como um dos recursos mais relevantes dentro dos IDEs, pois permitem expandir suas funcionalidades de acordo com as necessidades de cada projeto. Esse mecanismo de personalização vai além de ferramentas como o Git e abre espaço para integrações inovadoras, como extensões que incorporam Inteligência Artificial ao ambiente de desenvolvimento.

2.3 Inteligência Artificial

Inteligência Artificial (IA) é o ramo da ciência da computação dedicado a desenvolver máquinas e programas capazes de realizar tarefas que normalmente requerem inteligência humana, como percepção, raciocínio e tomada de decisão. O termo “inteligência artificial” foi cunhado por John McCarthy – considerado o pai da IA – em 1956. Na ocasião, em conjunto com Marvin Minsky, McCarthy definiu a IA como “a ciência e a engenharia de fazer máquinas inteligentes, especialmente programas de computador inteligentes”, demarcando o início formal desse campo de pesquisa. Esse conceito fundacional estabeleceu a meta central da área: construir sistemas computacionais com comportamentos inteligentes e capacidade de aprendizado semelhantes aos de um ser humano, ainda que limitados aos domínios específicos para os quais foram programados ou treinados (FRAIJ; VáRALLYAI, 2021).

Partindo dessa definição, coloca-se o desafio de estabelecer critérios objetivos para reconhecer o que são sistemas inteligentes e inteligências artificiais. Foi com esse propósito que em 1950, Alan Turing formulou seu famoso Teste de Turing, onde um computador seria considerado “inteligente” se um avaliador humano, trocando mensagens escritas, não conseguisse distinguir suas respostas das de uma pessoa. Superar tal desafio exige a integração de múltiplas disciplinas da IA, como o processamento de linguagem natural para a comunicação, a representação de conhecimento para contextualizar informações, o raciocínio automático para formular respostas coerentes e o aprendizado de máquina para se adaptar a novas situações. Posteriormente, pesquisadores ampliaram a proposta para o "Teste de Turing Total", que também demanda interação física com pessoas e objetos, incorporando a necessidade de visão computacional e robótica para a percepção e manipulação do ambiente (RUSSELL; NORVIG, 2021).

Assim, a evolução da Inteligência Artificial percorreu diferentes fases ao longo das décadas, passando dos sistemas baseados em regras e especialistas, predominantes entre os anos 1950 e 1970, para os avanços em redes neurais e técnicas de aprendizado de máquina que marcaram o final do século XX. Já no século XXI, o desenvolvimento de algoritmos de aprendizado profundo (deep learning) e a disponibilidade massiva de dados permitiram aplicações cada vez mais sofisticadas, como o reconhecimento de imagens, voz e linguagem natural (GLUSHKOVA, 2023).

Com o progresso do campo ao longo das décadas, as aplicações da IA expandiram-se de forma exponencial onde antes eram restritas. Inicialmente confinada a laboratórios de pesquisa e domínios acadêmicos, a IA passou a permear diversos setores da indústria e do cotidiano, sendo considerada por alguns como a quarta revolução industrial. Encontram-se sistemas de IA em áreas tão variadas quanto a saúde (por exemplo, auxiliares de diagnóstico por imagem), finanças (análise de risco e detecção de fraudes), transporte (veículos autônomos) e serviços pessoais (assistentes virtuais em smartphones) (TAI, 2020).

Essa evolução deu-se também pela popularização de IAs conversacionais e modelos generativos de linguagem que ampliou o acesso e o interesse pela tecnologia. Tais modelos passaram a ser incorporados em sites e plugins, o que impulsionou seu uso em diferentes aplicações. Exemplos como GPT-3.5 e GPT-4 da OpenAI ou o Gemini do Google contam com centenas de bilhões de parâmetros treináveis e vastos conjuntos de dados, permitindo gerar conteúdos coerentes em múltiplos domínios. (FERNANDEZ; CORNELL, 2024). Esse movimento marca a transição para uma nova etapa da área: a Inteligência Artificial Generativa.

2.3.1 Inteligência Artificial Generativa

A inteligência artificial generativa (GenAI) é uma classe de tecnologias de inteligência artificial que geram novos conteúdos como texto, imagem, música ou vídeo, analisando padrões em dados existentes (BRYNJOLFSSON; LI; RAYMOND, 2023). Tais tecnologias têm ganhado relevância principalmente com o surgimento de modelos linguísticos capazes de gerar respostas coerentes e contextuais.

Entre esses modelos, destacam-se os Large Language Models (LLMs), uma categoria específica de modelo de IA Generativas. Os LLMs são treinados em enormes quantidades de dados, tipicamente não rotulados, para entender e gerar respostas textuais semelhantes às humanas (YANG *et al.*, 2022).

Em essência, um modelo LLM prevê a próxima palavra de uma sequência textual com base nas anteriores e no contexto construído até aquele ponto. No entanto, é importante lembrar que o modelo não possui compreensão semântica ou intenção, apenas manipula probabilidades para gerar texto que parece coerente (SHANAHAN, 2024).

2.4 Inteligência Artificial Generativa no Desenvolvimento de Software

No campo do desenvolvimento de software, a inteligência artificial generativa tem se consolidado como uma aliada estratégica para aumentar a produtividade e melhorar a qualidade do código. Essas tecnologias vêm sendo aplicadas em atividades que tradicional-

mente demandam tempo e esforço dos desenvolvedores, como a automação de testes, a detecção de falhas e a sugestão de melhorias no código-fonte (LEITE; RIBEIRO, 2023).

Uma das maiores contribuições da GenAI nesse contexto é a capacidade de apoiar o processo criativo e técnico de programação por meio da geração assistida de código. Ferramentas como o GitHub Copilot exemplificam essa tendência, oferecendo recomendações contextuais diretamente no ambiente de desenvolvimento e acelerando a implementação de soluções. Esse tipo de integração já é adotado por grande parte dos desenvolvedores profissionais, indicando uma transformação significativa nas práticas de engenharia de software (FERNANDEZ; CORNELL, 2024).

Um reflexo direto desse impacto da inteligência artificial generativa pode ser observado na queda acentuada da participação em comunidades tradicionais de desenvolvedores, como o Stack Overflow. O número de perguntas feitas mensalmente na plataforma despencou para níveis semelhantes aos de seu lançamento em 2009, após anos sendo referência no suporte entre programadores. Esse declínio coincide com a popularização de modelos de linguagem, como o ChatGPT, que oferecem respostas rápidas, personalizadas e sem as barreiras de moderação típicas do Stack Overflow (OROSZ, 2025). Assim, evidencia-se que a adoção crescente de ferramentas de IA não apenas transformou a forma como o código é produzido, mas também alterou significativamente os hábitos de aprendizado e suporte da comunidade de desenvolvedores, consolidando um novo paradigma no ecossistema de engenharia de software, conforme ilustrado na Figura 2.

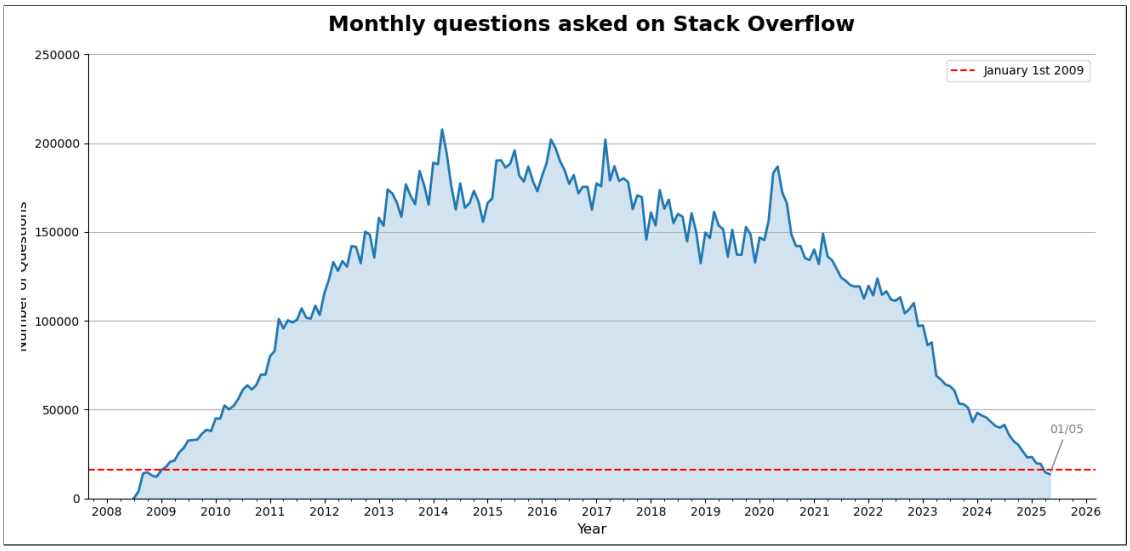

Figura 2 – Declínio no número de perguntas mensais no Stack Overflow, retornando a níveis de 2009 (OROSZ, 2025).

3 Trabalhos Relacionados

Neste capítulo, serão apresentados os trabalhos relevantes que abordam temas similares ao desta pesquisa. Para a identificação e análise crítica desses trabalhos, foi realizado um Mapeamento Sistemático da Literatura (MSL). De acordo com Kitchenham e Charters (2007), o MSL possibilita uma revisão ampla e estruturada de estudos primários dentro de um tópico de pesquisa, permitindo identificar, avaliar e interpretar os trabalhos relevantes com foco em uma área ou tema de investigação (KITCHENHAM; CHARTERS, 2007).

3.1 Protocolo de Mapeamento

O protocolo seguido para a realização deste MSL foi inspirado nas diretrizes de Kitchenham e Charters (2007) e compreendeu as seguintes etapas: i) definição do objetivo da pesquisa; ii) formulação das questões de investigação; iii) seleção das fontes de dados relevantes; iv) elaboração da *string* de busca; e v) definição dos critérios de inclusão, exclusão e avaliação da qualidade dos estudos.

O objetivo da pesquisa foi definido com base na seguinte questão: *Como a utilização de inteligência artificial pode apoiar, em tempo real, as práticas de desenvolvimento de código seguro?*

Com base nesse objetivo, foram formuladas as seguintes questões de pesquisa:

- **Q1:** Quais são os contextos e domínios de aplicação da IA voltada à segurança de software?
- **Q2:** Quais técnicas têm sido utilizadas para promover segurança no desenvolvimento de software?
- **Q3:** Quais tipos de vulnerabilidades ou falhas são abordadas por essas soluções?
- **Q4:** Quais ferramentas ou abordagens têm se mostrado eficazes no apoio à segurança via IA?

A fonte de dados utilizada para a realização da busca foi o Google Scholar, em razão de sua ampla cobertura de artigos científicos e acessibilidade. A justificativa para essa escolha metodológica reside tanto na gratuidade e facilidade de uso da plataforma, que democratiza o acesso à descoberta científica, quanto em sua abrangência documental. A cobertura do Google Scholar não é apenas quantitativamente superior à de bases como Web of Science e Scopus, mas também qualitativamente mais diversa, incluindo tipologias essenciais como teses, livros e anais de conferência, cruciais para uma revisão de literatura abrangente e atualizada (MARTíN-MARTíN *et al.*, 2018).

A *string* de busca foi elaborada com base na estratégia Population, Intervention, Comparison, Outcome (PICO) (SANTOS; PIMENTA; NOBRE, 2007), onde a população corresponde a *"software development"*, a intervenção refere-se a abordagens *"AI-based"*, e o desfecho esperado envolve práticas de *"Code Review"* mediadas por ferramentas *"Tool"*. O critério de comparação foi desconsiderado por não se aplicar ao objetivo exploratório deste estudo. Assim, a *string* genérica utilizada nas fontes de busca foi definida como:

`"software development"` AND `"AI-based"` AND `"Code Review"` AND `"Tool"`

Para garantir a qualidade e a relevância dos estudos incluídos, foram definidos os seguintes critérios:

Critérios de Inclusão (CI):

- CI1: Estudos que abordem o uso de IA para segurança no desenvolvimento de software;
- CI2: Estudos publicados a partir de 2023, de modo a contemplar as pesquisas mais recentes;
- CI3: Disponibilidade do texto completo em português ou inglês.

Critérios de Exclusão (CE):

- CE1: Estudos que abordam temas semelhantes sem apresentar contribuições adicionais ou relevantes em relação a trabalhos já incluídos;
- CE2: Trabalhos cujo acesso ao texto completo seja restrito por *paywall*, sem possibilidade de obtenção via acesso institucional ou fontes alternativas confiáveis;
- CE3: Trabalhos que não tenham sido publicados formalmente ou que não tenham passado por revisão por pares (ex.: repositórios como *arXiv*).

Após a aplicação dos critérios CE1 a CE3, os estudos selecionados passaram pela triagem final, com leitura completa (aceitação) e avaliação da contribuição efetiva para os objetivos da pesquisa.

Tabela 1 – Resumo do processo de seleção dos estudos

Etapa	Descrição	Encontrados / Restantes	Removidos
0	Busca inicial	147	–
1	Critério de Exclusão 1 (CE1)	147 $\rightarrow$ 73	74
2	Critério de Exclusão 2 (CE2)	73 $\rightarrow$ 36	37
3	Critério de Exclusão 3 (CE3)	36 $\rightarrow$ 28	8
4	Triagem final (aceitação)	28 $\rightarrow$ 3	25

Os três estudos selecionados e analisados com maior profundidade foram:

- *AICodeReview: Advancing code quality with AI-enhanced reviews*;

- *ChatGPT as a Software Development Tool: The Future of Development*;
- *The influence of Artificial Intelligence on productivity in Software Development*.

A revisão desses trabalhos busca não apenas situar a pesquisa dentro do contexto atual, mas também identificar lacunas no conhecimento existente e possíveis áreas para contribuição original.

3.2 AICodeReview: Advancing code quality with AI-enhanced reviews

(ALMEIDA *et al.*, 2024) apresentaram o *AICodeReview*, uma ferramenta projetada para aprimorar a qualidade e a eficiência dos processos de revisão de código. A pesquisa aborda a crescente complexidade dos projetos de software modernos e a necessidade de ferramentas automatizadas para apoiar os desenvolvedores.

O objetivo central do trabalho foi desenvolver uma ferramenta, na forma de um *plugin* para a IDE IntelliJ, que utiliza IA para automatizar a avaliação de código. A ferramenta visa analisar de forma abrangente o código para identificar problemas de sintaxe e semântica, além de propor resoluções viáveis. Com isso, os autores buscaram otimizar o tempo e o esforço dedicados à revisão, resultando em um avanço na qualidade geral do software.

A abordagem consiste na integração do modelo GPT-3.5 diretamente no IDE do desenvolvedor. O AICodeReview funciona como um assistente que, sob demanda, analisa o código selecionado pelo usuário. A ferramenta oferece funcionalidades como:

- Revisão de código multilíngue: Suporte para linguagens como Java, Kotlin, C++ e Python;
- Sugestões personalizáveis: Permite ajustar o nível de detalhe e a precisão das recomendações;
- Compatibilidade: Integra-se com todos os produtos da família JetBrains, como PyCharm e Android Studio.

Para validar a eficácia da ferramenta, os autores conduziram uma avaliação preliminar com 12 estudantes de graduação com conhecimentos em programação Java e no uso do IntelliJ. Os participantes foram divididos em dois grupos: um grupo experimental, que utilizou o AICodeReview, e um grupo de controle, que realizou a revisão de código de forma manual.

A metodologia empregou uma abordagem onde ambos os grupos revisaram os mesmos arquivos de código contendo cerca de 30 tipos diferentes de "*code smells*", indícios de problemas no código que, embora não representem erros de funcionamento imediato, indicam más práticas de programação ou escolhas de design que podem prejudicar a legibilidade, a manutenção e a evolução do software.

Foram avaliadas as seguintes métricas: Tempo gasto para revisar cada arquivo, Número de *code smells*"detectados e número de *code smells* refatorados (corrigidos). Os resultados indicaram que o grupo que utilizou o AICodeReview obteve um desempenho significativamente superior em todas as métricas: reduziu o tempo médio de revisão (15,2 min contra 22,5 min), detectou um número maior de *code smells* (28 contra 20) e alcançou um número superior de refatorações bem-sucedidas (25 contra 13).

As principais contribuições do estudo incluem a introdução da ferramenta AICodeReview, a avaliação empírica demonstrando sua eficácia e *insights* sobre técnicas de revisão de código baseadas em IA.

3.3 ChatGPT as a Software Development Tool: The Future of Development

(HöRNEMALM, 2023) investiga e avalia como o ChatGPT pode ser utilizado como uma ferramenta de apoio nas atividades diárias de desenvolvedores de software. O estudo explora a eficácia da ferramenta, os riscos associados ao seu uso e as possíveis melhorias na experiência do desenvolvedor.

O objetivo principal do trabalho foi avaliar o ChatGPT como uma ferramenta para o desenvolvimento de software. Para isso, a pesquisa buscou responder a três questões centrais: O ChatGPT é uma ferramenta eficaz para desenvolvedores? Existem riscos associados ao uso de ferramentas de IA? O que pode ser feito para melhorar a experiência do desenvolvedor ao usar essas ferramentas?

O modelo de IA na pesquisa foi o ChatGPT, especificamente a versão GPT-4, que foi lançada durante o período do estudo. Os participantes receberam acesso a uma conta do ChatGPT Plus para garantir o uso do modelo mais avançado disponível no momento da coleta de dados.

A metodologia foi estruturada em duas fases distintas. Na fase de exploração inicial, foram conduzidas entrevistas com cinco desenvolvedores sênior, com o objetivo de obter uma compreensão aprofundada da profissão. Os *insights* coletados nessa etapa serviram de base teórica e auxiliaram na definição das tarefas a serem utilizadas na fase seguinte.

Na fase de coleta de dados, sete desenvolvedores participaram de um experimento prático. Utilizando o método de observação direta, os participantes realizaram cinco tarefas em ambiente remoto, como escrever testes unitários, depurar código, redigir e-mails e elaborar um plano Scrum. Ao término das atividades, foram conduzidas entrevistas para reunir informações sobre a experiência, a percepção de eficácia e o nível de confiança na ferramenta.

Os resultados indicaram que o ChatGPT foi eficaz para aumentar a produtividade dos

desenvolvedores, principalmente em tarefas de codificação e planejamento. A ferramenta demonstrou ser excelente na geração de testes unitários e na depuração de código, além de oferecer um bom ponto de partida para o planejamento de projetos no formato Scrum.

Em tarefas de comunicação, como a redação de e-mails e descrições de pull requests, a ferramenta se mostrou menos eficiente. Os participantes consideraram o texto gerado excessivamente formal, prolixo e desalinhado com o tom de comunicação típico em ambientes de trabalho, exigindo edições manuais significativas. A maioria dos participantes também expressou o desejo de que as ferramentas fossem mais integradas ao software existente que já utilizam.

Um dos achados mais relevantes foi a identificação de um risco associado ao nível de experiência do usuário. Constatou-se que desenvolvedores juniores tendem a confiar mais nas respostas do ChatGPT do que desenvolvedores sêniores, evidenciando uma correlação entre menor experiência profissional e maior vulnerabilidade ao uso acrítico de ferramentas de IA generativa. Nesse sentido, torna-se essencial que organizações que adotem o ChatGPT implementem mecanismos robustos de suporte e supervisão, a fim de mitigar riscos especialmente entre profissionais menos experientes.

3.4 The influence of Artificial intelligence on productivity in Software development

(GLUSHKOVA, 2023) realiza uma investigação quantitativa para analisar a influência da Inteligência Artificial, especificamente do ChatGPT, na produtividade em diferentes estágios do ciclo de vida do desenvolvimento de software. O estudo busca fornecer dados empíricos sobre o impacto de uma ferramenta de IA generativa.

O objetivo da pesquisa foi investigar e quantificar o impacto do ChatGPT na produtividade ao longo das várias fases do processo de desenvolvimento de software. Para isso, o trabalho buscou analisar como a percepção desse impacto varia de acordo com características dos profissionais, como nível de experiência, cargo e tamanho da empresa em que atuam. Além disso, a pesquisa se propôs a identificar quais estágios do desenvolvimento são mais beneficiados pelo uso da ferramenta

A abordagem adotada foi um estudo quantitativo baseado em uma pesquisa (*survey*) online, complementado por uma análise qualitativa dos comentários abertos dos participantes. A pesquisa foi projetada para coletar dados sobre a percepção de produtividade de profissionais que já utilizavam o ChatGPT em seu trabalho, permitindo uma análise estatística das suas experiências e opiniões.

A metodologia foi baseada na aplicação de um questionário online a 150 profissionais da área de desenvolvimento de software, distribuído através de plataformas como LinkedIn,

Telegram e WhatsApp entre março e julho de 2023. A amostra foi diversificada, incluindo diferentes cargos (desenvolvedores, gerentes de produto, analistas), níveis de senioridade (de júnior a sênior) e contextos organizacionais.

A pesquisa utilizou escalas Likert para medir a percepção de impacto em diversas dimensões da produtividade, como velocidade na conclusão de tarefas, melhoria na qualidade do trabalho, redução de erros e aprimoramento na capacidade de resolução de problemas. Os dados coletados foram analisados com técnicas estatísticas como análise de correlação e ANOVA (Análise de Variância) para identificar diferenças significativas entre os grupos de respondentes.

Os resultados confirmaram um impacto positivo e moderado do ChatGPT na produtividade geral. Os maiores benefícios foram percebidos nas fases de Documentação e Codificação, que obtiveram as maiores médias de economia de tempo percebida. Em contrapartida, o estágio de Implantação (Deployment) foi o que apresentou o menor benefício, segundo os participantes.

Uma das descobertas mais relevantes foi a identificação de uma correlação inversa entre o nível de experiência e o benefício percebido. Profissionais com menor experiência (júnior/estagiário) relataram ganhos substanciais em produtividade, qualidade e aprendizado, enquanto desenvolvedores mais experientes perceberam impactos menos expressivos. Esse resultado reforça a tendência já observada por (HöRNEMALM, 2023), segundo a qual desenvolvedores juniores demonstram maior propensão a confiar e depender das respostas do ChatGPT. Além disso, o estudo evidenciou que os usuários da versão paga (ChatGPT 4.0) relataram ganhos de produtividade significativamente superiores em comparação aos usuários da versão gratuita.

3.5 Comparativo

A literatura demonstra um crescente interesse na aplicação de Inteligência Artificial para otimizar o ciclo de vida do desenvolvimento de software. Os trabalhos analisados exploram o uso de IAs generativas sob a ótica da qualidade do código, da experiência do desenvolvedor e da produtividade. Nossa proposta se insere nesse contexto, porém, direciona o foco para um nicho crítico e ainda pouco explorado: a integração de análises de segurança de software em tempo real, diretamente no ambiente de desenvolvimento.

Para contextualizar as contribuições desta pesquisa, a Tabela 2 compara os três estudos selecionados com a abordagem proposta neste trabalho, destacando objetivos, abordagens e metodologias.

Tabela 2 – Análise comparativa entre os trabalhos relacionados e a proposta atual

Característica	**Almeida et al. (2024)**	**Hörnemalm (2023)**	**Glushkova (2023)**	**Este Trabalho**
Objetivo Principal	Melhorar a qualidade e a eficiência da revisão de código, detectando erros de sintaxe e semântica.	Avaliar o ChatGPT como ferramenta para atividades diárias de desenvolvimento (codificação, comunicação e planejamento).	Investigar o impacto percebido do ChatGPT na produtividade em diferentes estágios do desenvolvimento de software.	Aprimorar a segurança do software por meio da integração, no IDE, de uma ferramenta de IA para detecção de vulnerabilidades.
Abordagem	Plugin (AICodeReview) para a IDE IntelliJ.	Uso direto da interface web do ChatGPT (externa ao IDE).	Pesquisa de opinião (*survey*) com profissionais de software.	Extensão para a IDE Visual Studio Code.
Modelo de IA	GPT-3.5.	GPT-4.	Predominantemente GPT-3.5 (versão gratuita utilizada pela maioria) e GPT-4.	Suporte a múltiplos provedores (ChatGPT, Google Gemini, Ollama).
Metodologia	Avaliação experimental com 12 estudantes, comparando o uso da ferramenta com a revisão manual de *code smells*.	Estudo qualitativo em duas fases: entrevistas com desenvolvedores sênior para base teórica e observação de 7 desenvolvedores realizando tarefas.	Análise quantitativa a partir de *survey* com 150 profissionais, incluindo estatística descritiva, correlação e ANOVA.	Prova de conceito com códigos vulneráveis e estudo de caso em projeto de software de código aberto (*WeGIA*).

A análise da Tabela 2 evidencia que, embora todos os trabalhos convirjam para o potencial transformador da IA, cada um aborda uma faceta distinta do problema. Este trabalho se diferencia e complementa os demais da seguinte forma:

Em contraste com o estudo de (ALMEIDA *et al.*, 2024), que introduziu um plugin para IntelliJ IDEA focado na melhoria da qualidade geral do código (detecção de *code smells*), a extensão para o Visual Studio Code proposta neste trabalho visa especificamente a detecção de vulnerabilidades de segurança.

O trabalho de (HöRNEMALM, 2023) revelou um forte desejo dos profissionais por ferramentas mais integradas ao seu fluxo de trabalho. A abordagem proposta neste trabalho atua diretamente nessa questão ao desenvolver uma extensão para IDE, eliminando a necessidade de alternar entre o editor de código e uma interface de chat externa.

Enquanto (GLUSHKOVA, 2023) oferece uma visão abrangente sobre a *percepção* de produtividade por meio de uma *survey*, nosso trabalho se propõe a realizar uma análise prática. Ao aplicar a extensão em um projeto de código aberto real e comparar seus resultados com a análise de especialistas, buscamos evidências concretas sobre a eficácia da ferramenta na detecção de falhas, indo além da percepção subjetiva de produtividade para uma avaliação objetiva de sua contribuição real para a segurança do código.

4 Proposta

A proposta deste trabalho consiste em avaliar o impacto da utilização de Inteligência Artificial (IA) durante a fase de desenvolvimento de software, especificamente no apoio à identificação de vulnerabilidades no código. Para isso, foi desenvolvida uma extensão para um ambiente de desenvolvimento integrado (IDE), que utiliza modelos de inteligência artificial generativa com o objetivo de detectar vulnerabilidades e sugerir melhorias no código fonte.

A avaliação da eficácia da extensão será realizada por meio de um experimento estruturado em três etapas complementares. A primeira etapa consiste na análise de códigos reconhecidamente vulneráveis, obtidos a partir de um repositório público que reúne exemplos de vulnerabilidades amplamente documentadas. Na segunda etapa, é realizada uma análise comparativa baseada no estudo de (COSTA *et al.*, 2018) e tem como propósito mensurar o desempenho da extensão em relação ao desempenho humano. Por fim, a aplicabilidade da extensão em um ambiente real por meio da análise de um projeto de software de código aberto.

4.1 Projeto

O projeto da extensão foi concebido com o propósito de apoiar desenvolvedores na identificação de vulnerabilidades em tempo real, diretamente no ambiente de desenvolvimento. Para isso, buscou-se estabelecer requisitos claros que orientassem a implementação e servissem como base para a avaliação da ferramenta. Entre os objetivos principais, destaca-se a necessidade de permitir a seleção de trechos de código no editor e o envio desses fragmentos para análise, de forma transparente e integrada ao fluxo de trabalho do desenvolvedor. A extensão também foi planejada para se conectar a diferentes provedores de inteligência artificial, como OpenAI (ChatGPT), Google Gemini e Ollama, oferecendo ao usuário flexibilidade na escolha do mecanismo de análise.

Outro aspecto relevante considerado no projeto diz respeito à forma como os resultados da análise deveriam ser apresentados. A decisão foi de que eles seriam exibidos no próprio Visual Studio Code, por meio de um painel lateral interativo, a fim de preservar a usabilidade e evitar a necessidade de alternar entre múltiplas ferramentas. Além disso, estabeleceu-se que a extensão deveria suportar múltiplos idiomas, inicialmente português e inglês, de modo a ampliar seu alcance e permitir que o desenvolvedor configurasse a linguagem de acordo com suas preferências.

No que se refere a requisitos não funcionais, a portabilidade da solução é assegurada

pelo fato de a extensão ser executada dentro do próprio Visual Studio Code, o qual já possui suporte nativo para diferentes sistemas operacionais, como Windows, Linux e macOS. Dessa forma, não foi necessário implementar adaptações específicas para cada ambiente, uma vez que a compatibilidade é garantida pela infraestrutura da IDE, permitindo que a extensão seja utilizada de forma transparente em qualquer plataforma suportada pelo VSCode.

A escolha do Visual Studio Code como IDE para o desenvolvimento da extensão foi motivada não apenas por sua ampla adoção na comunidade de desenvolvedores, mas também por sua posição de destaque entre as ferramentas mais utilizadas mundialmente. Em agosto de 2025, o VSCode ocupava a segunda colocação no ranking global, com uma participação de 15,66%, logo atrás do Visual Studio, que liderava com 28,97%. Esse cenário reforça a relevância da plataforma, uma vez que seu uso disseminado garante maior impacto, alcance e aderência da solução junto aos desenvolvedores. A Figura 3 ilustra esse panorama.

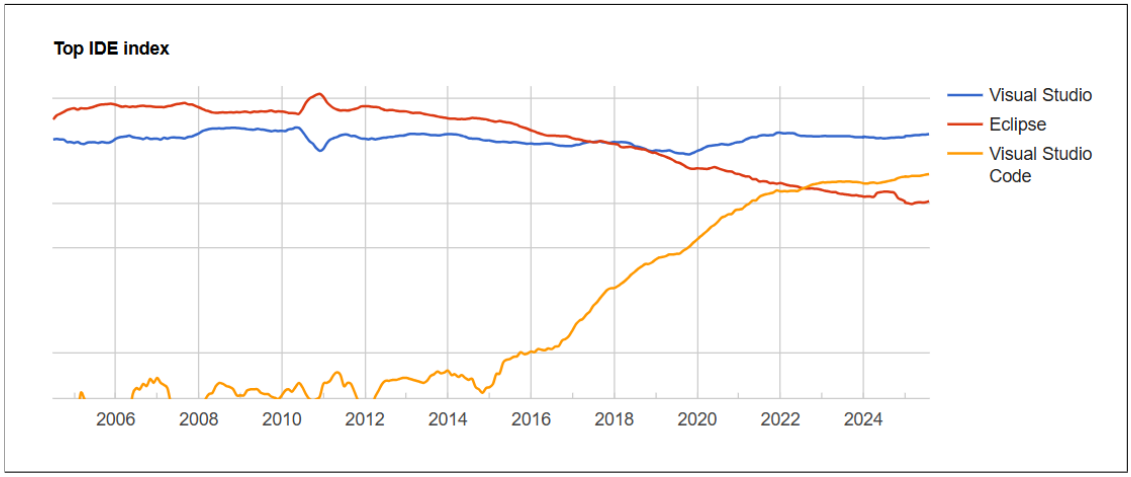

Figura 3 – Ranking das IDEs mais utilizadas mundialmente em agosto de 2025

Por fim, o projeto considerou algumas premissas e limitações que norteiam a utilização da extensão. O desempenho da análise está diretamente relacionado à qualidade do provedor de IA selecionado, o que significa que resultados distintos podem ser obtidos conforme a tecnologia escolhida. Além disso, a extensão não executa correções automáticas no código, restringindo-se à detecção de vulnerabilidades e à sugestão de melhorias. No caso da integração com o Ollama, reconhece-se que a execução de modelos localmente pode demandar elevado consumo de memória e espaço em disco, o que pode impactar a experiência do usuário em determinados ambientes. Esses pontos, entretanto, não representam restrições impeditivas, mas sim aspectos considerados no delineamento do projeto para garantir maior clareza quanto às suas capacidades e limitações.

4.2 Implementação

O funcionamento da extensão pode ser compreendido a partir de um fluxo de interações que envolve o desenvolvedor, o Visual Studio Code, a própria extensão e o provedor de inteligência artificial escolhido, esse processo é ilustrado na Figura 4.

Inicialmente, o desenvolvedor escreve seu código no editor do Visual Studio Code e, quando deseja submetê-lo à análise, executa o comando disponibilizado pela extensão, denominado *"Analisar Código"*. A partir desse momento, o editor aciona a extensão *MN-Analise*, que passa a controlar o fluxo da operação. O primeiro passo realizado pela extensão é verificar qual provedor de inteligência artificial está configurado e, em seguida, validar as credenciais ou parâmetros necessários, como a chave de API ou o endpoint local no caso do Ollama. Essa validação garante que a comunicação com o serviço selecionado possa ser estabelecida de forma correta e segura.

Com a configuração confirmada, a extensão prepara a entrada que será enviada ao modelo, construindo um *prompt* fixo em português ou inglês, de acordo com a escolha do usuário.

- **Em Português:** *"Detecte as vulnerabilidades no código e explique, se tiver uma."*
- **Em Inglês:** *"Detect the vulnerabilities in the code and explain, if there is one."*

Este prompt é então combinado com o trecho de código selecionado pelo usuário, compondo a mensagem que será enviada ao modelo de IA para análise.

Esse conjunto de informações é então transmitido via requisição HTTP para a API do provedor de IA. No provedor, o código e o *prompt* são processados e analisados, resultando em uma resposta que descreve as vulnerabilidades encontradas e, quando possível, sugestões de melhoria.

Após a conclusão do processamento, a resposta é devolvida à extensão, que trata o conteúdo recebido e organiza os resultados de forma adequada para apresentação no Visual Studio Code. Para isso, é gerado dinamicamente um painel do tipo *WebView*, exibido ao lado do editor, no qual o desenvolvedor pode visualizar as vulnerabilidades identificadas e compreender melhor o diagnóstico. Por fim, o painel é renderizado e apresentado ao usuário, permitindo que o ciclo se conclua com a análise visível diretamente no ambiente de desenvolvimento.

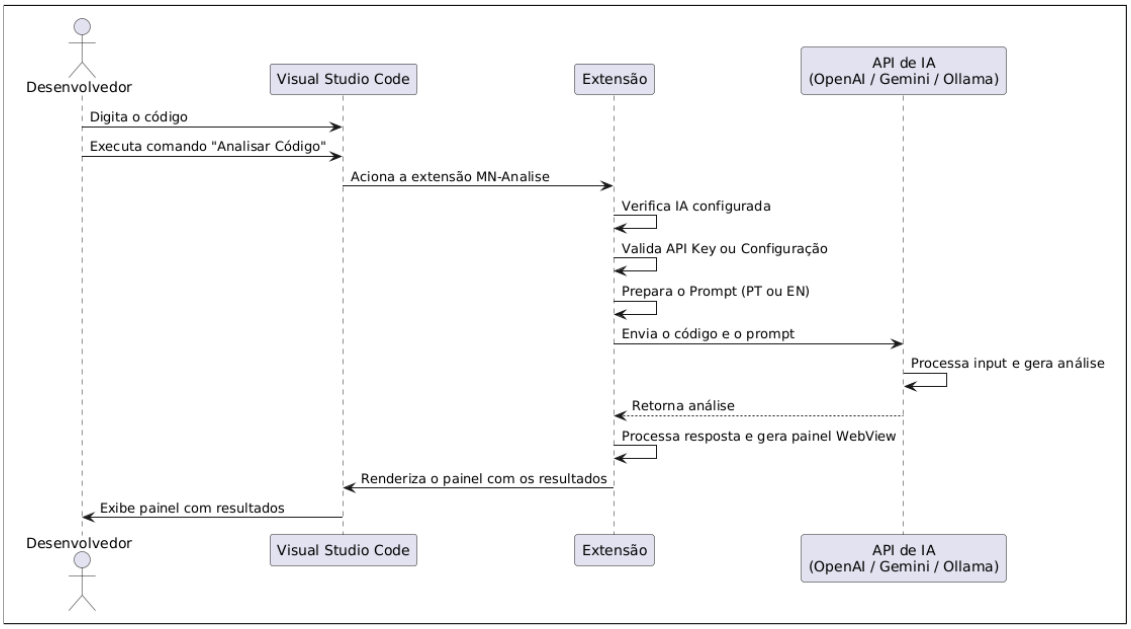

Figura 4 – Diagrama de sequência da extensão proposta

4.2.1 Desenvolvimento da Extensão no Visual Studio Code

A extensão foi construída utilizando o framework oficial de desenvolvimento do Visual Studio Code, o processo iniciou-se com a configuração do ambiente, que envolveu a instalação do Node.js e o uso das ferramentas *yo* e *generator-code* para a geração da estrutura inicial do projeto em JavaScript. Esse esqueleto forneceu a base para a modularização da solução, que foi projetada de forma a separar claramente responsabilidades entre configuração, lógica de negócio e integração com provedores externos.

A arquitetura da extensão é organizada em módulos distintos. O arquivo *package.json* concentra as informações de configuração, incluindo metadados (nome, versão, autor e descrição), comandos registrados e dependências externas, como a biblioteca oficial do Google Gemini (@google/generative-ai). O arquivo *extension.js* atua como ponto de entrada da aplicação, responsável por registrar o comando principal *"mn-analise.analyzeCodeCommand"* e controlar o ciclo de vida da extensão por meio das funções *activate* e *deactivate*. O módulo *config.js*, por sua vez, centraliza o gerenciamento das configurações fornecidas pelo usuário, como chave de API, provedor de IA e idioma do prompt. Já o núcleo da lógica de negócio está encapsulado no módulo *codeAnalyzerService.js*, que implementa a comunicação com as APIs de IA e organiza os resultados para posterior exibição no editor.

O empacotamento e a publicação da extensão no repositório oficial do Visual Studio Code, de modo que esta fique disponível para uso público, requer criar uma conta na plataforma do Visual Studio Code Publisher disponível em <https://dev.azure.com>. Em seguida, foi instalada a ferramenta de linha de comando *VSCE* com o comando "npm install -g vsce". O empacotamento da extensão foi feito com o comando "vsce package", que gera

um arquivo ".vsix" pronto para distribuição manual ou envio ao marketplace. A publicação oficial foi realizada com os comandos "vsce publish".

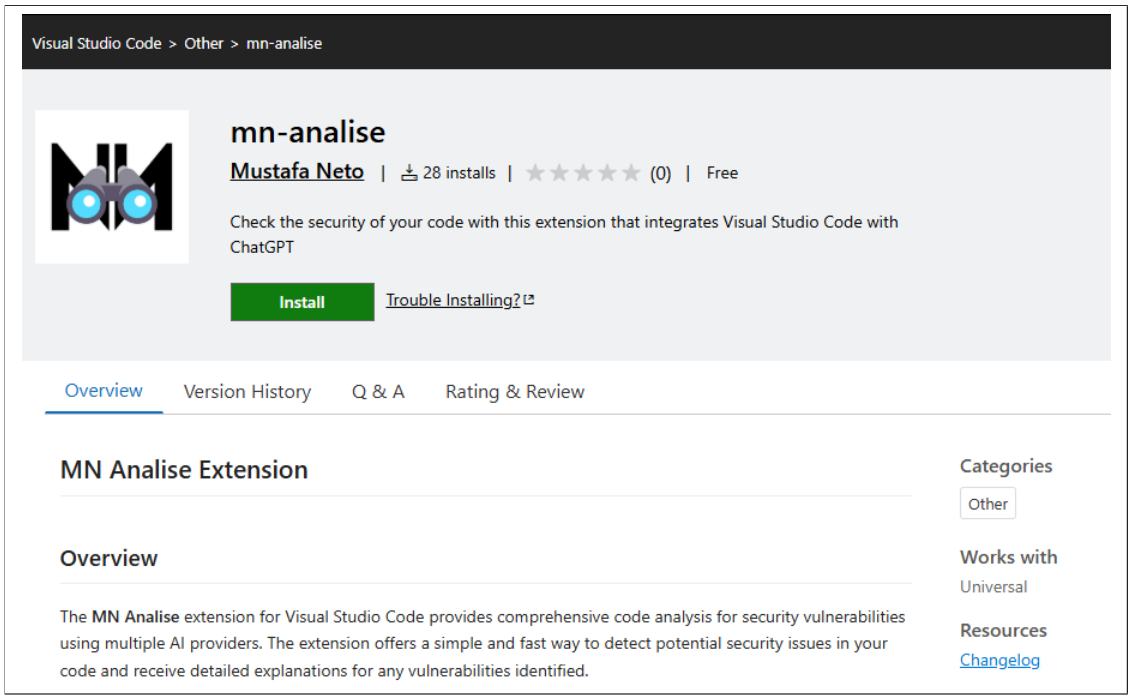

Figura 5 – Extensão disponível no Marketplace - Microsoft

4.2.2 Integração com o ChatGPT (OpenAI)

A integração da extensão com o ChatGPT é realizada por meio da API oficial da OpenAI. Para que a comunicação com o modelo seja possível, é necessário que o usuário obtenha uma chave de API, a qual funciona como uma credencial de acesso segura. Esse processo inicia-se com o acesso à plataforma de desenvolvedores da OpenAI, disponível em <https://platform.openai.com/>. O usuário deve realizar login (ou criar uma conta, caso ainda não possua). Após o login, é necessário navegar até a seção "API keys" no menu lateral e criar uma nova chave clicando em "Create new secret key". A chave gerada é exibida uma única vez e, por motivos de segurança, deve ser copiada e armazenada em local seguro, pois não será possível visualizá-la novamente.

Com a chave em mãos, o usuário deve acessar as configurações do Visual Studio Code, e localizar o grupo de configurações da extensão, identificado por "mn-analise". A chave obtida deve ser inserida no campo "mn-analise.openaiApiKey" e, no campo "mn-analise.aiProvider", deve ser selecionada a opção "chatgpt", habilitando assim o uso do modelo da OpenAI pela extensão.

4.2.3 Integração com o Gemini (Google)

A integração com o modelo Gemini, desenvolvido pelo Google, é realizada por meio do Google AI Studio e do SDK oficial fornecido para JavaScript. Assim como na integração com o ChatGPT, é necessário gerar uma chave de API. O processo começa com o acesso ao site do Google AI Studio, disponível em <https://aistudio.google.com/>, seguido do login com uma conta Google. Dentro da plataforma, o usuário deve navegar até a seção de chaves e selecionar a opção "Get API key". Após a geração da chave, ela será exibida para que o usuário possa copiá-la e armazená-la em local seguro.

Para utilizar o Gemini na extensão, o usuário deve novamente acessar as configurações do VS Code e inserir a chave obtida no campo "mn-analise.geminiApiKey". Em seguida, o campo "mn-analise.aiProvider" deve ser configurado com o valor "gemini", permitindo que o sistema envie as requisições de análise diretamente à API do Google.

4.2.4 Integração com o Ollama

Diferentemente das integrações anteriores baseadas em serviços na nuvem, o Ollama é uma plataforma voltada para a execução local de modelos de linguagem, permitindo que os dados processados permaneçam exclusivamente no ambiente do desenvolvedor. Essa abordagem oferece vantagens significativas em termos de segurança, privacidade e controle sobre a informação, pois não há necessidade de transmissão de dados sensíveis para servidores externos.

A utilização do Ollama requer a instalação prévia do software, que pode ser obtido no site oficial (<https://ollama.com/>). O usuário deve baixar o instalador compatível com seu sistema operacional (Windows, macOS ou Linux) e realizar a instalação. Após a conclusão, o Ollama será executado como um serviço em segundo plano.

Com o Ollama instalado, o próximo passo consiste no download de um modelo. Isso é feito por meio do terminal, utilizando o comando "ollama run <nome-do-modelo>". Por exemplo, para utilizar o modelo Llama 3 da Meta, o comando é "ollama run llama3". Esse procedimento pode consumir alguns gigabytes de espaço e levar algum tempo, dependendo da conexão e do modelo selecionado.

A configuração da extensão para uso com o Ollama é realizada diretamente nas configurações do VS Code. O campo "mn-analise.aiProvider"deve ser definido como "ollama". Em seguida, no campo "mn-analise.ollamaModel", o usuário deve inserir o nome exato do modelo baixado, como por exemplo "llama3". O campo "mn-analise.ollamaEndpoint" já vem preenchido com o endereço padrão (http://localhost:11434) e somente precisa ser alterado caso o Ollama esteja sendo executado em uma porta ou máquina diferente.

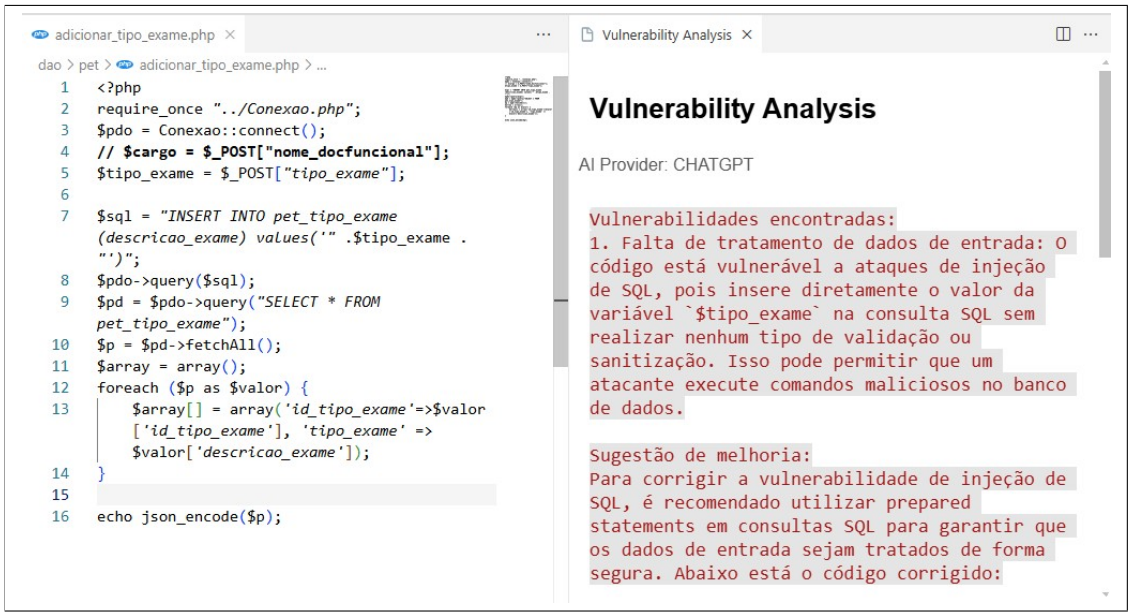

Figura 6 – Ilustração do uso da extensão desenvolvida no VSCode

4.3 Reprodutibilidade

Para garantir a transparência e permitir a validação e replicação dos resultados apresentados neste trabalho, todos os artefatos de software desenvolvidos foram disponibilizados publicamente. O código-fonte completo da extensão "MN-Analise", juntamente com as instruções detalhadas para instalação, configuração e uso, pode ser acessado em seu repositório no GitHub, no endereço <https://github.com/mustafaneto/code-analyzer>.

Adicionalmente, foi criado um vídeo demonstrativo para ilustrar o funcionamento da ferramenta na prática. O conteúdo do vídeo inicia com o processo de instalação da extensão diretamente no Visual Studio Code, acessando a aba de extensões do *Marketplace*, pesquisando por *MN-Analise* e selecionando a opção *Instalar*. Em seguida, é mostrado que a extensão passa a estar disponível no menu lateral da IDE, contendo os botões *Search Vulnerabilities* e *Settings*. O botão *Settings* é utilizado para configurar as credenciais de API e selecionar o idioma de uso da ferramenta.

Na sequência, o vídeo apresenta um tutorial para cada provedor de inteligência artificial suportado pela extensão. Primeiro, demonstra-se como gerar a chave de API no ChatGPT (OpenAI), depois no Gemini (Google) e, por fim, no Ollama. Para cada provedor, é exibida a extensão em execução, utilizando o botão *Search Vulnerabilities* para analisar trechos de código em tempo real e mostrar os resultados no painel lateral do VSCode. O vídeo completo está disponível no YouTube[1].

[1] <https://youtu.be/-TUOCCtkHns?si=YFAQyLv3Qccf1Fg9>

5 Prova de Conceito

Neste capítulo, apresenta-se a prova de conceito desenvolvida para demonstrar a viabilidade da utilização da IA como ferramenta de apoio ao desenvolvimento seguro de software, conduzida por meio da extensão integrada ao modelo ChatGPT (GPT-3.5-turbo).

O objetivo central foi investigar se a ferramenta seria capaz de identificar vulnerabilidades em tempo de desenvolvimento, comparando seus resultados com bases de código reconhecidamente vulneráveis e com estudos prévios envolvendo desenvolvedores humanos. Pretendeu-se analisar se o desempenho da IA poderia igualar ou até superar o de profissionais em cenários específicos.

Os experimentos descritos constituem a base de um trabalho preliminar apresentado na 15ª edição do *Computer on the Beach*, no artigo "Uma extensão para o VSCode que utiliza o ChatGPT como ferramenta de apoio ao desenvolvimento de software seguro" (NETO; LAZARIN, 2024).

Dois experimentos complementares foram realizados: o primeiro mediu a eficácia da extensão na identificação de vulnerabilidades em códigos vulneráveis; o segundo consistiu em uma análise comparativa com o estudo de Costa et al. (2018), contrastando o desempenho da IA com o de desenvolvedores humanos na detecção de falhas críticas.

5.1 Seleção e Execução em Códigos Vulneráveis

O repositório *Vulnerable Code Snippets*, disponível no GitHub (Security, Snoopy, 2024) possui códigos que simulam diferentes tipos de vulnerabilidades comuns, como *SQL Injection*, *Session Hijacking*, *Sensitive Data Exposure*, entre outras. O objetivo deste experimento foi avaliar a capacidade da ferramenta de identificar corretamente essas vulnerabilidades em tempo de desenvolvimento.

Durante os testes, os códigos foram submetidos individualmente através da extensão. A interação com a ferramenta se dá pela seleção do trecho vulnerável diretamente no VSCode, com a extensão enviando automaticamente o código selecionado para análise do modelo de IA, solicitando especificamente a identificação e explicação das vulnerabilidades encontradas.

Ao todo, foram testados 34 diferentes tipos de vulnerabilidades. A extensão apresentou uma taxa de eficácia de 88,24%, identificando corretamente 30 das 34 vulnerabilidades analisadas. Vale destacar que, mesmo nos casos em que não houve identificação direta da vulnerabilidade, como em *Code Execution* e *PostMessage Security*, a IA ofereceu recomendações pertinentes para aprimorar a segurança do código. As únicas vulnerabilidades não

detectadas foram relacionadas a *Connection String Injection* e *Sensitive Data Exposure*.

5.2 Análise Comparativa com Desenvolvedores

O segundo experimento teve como objetivo comparar a eficácia da extensão baseada em IA com a análise feita por desenvolvedores humanos. Para isso, foram utilizados quatro trechos de códigos-fonte contendo vulnerabilidades, os quais haviam sido previamente analisados no estudo conduzido por Costa et al. (2018).

No estudo de referência, foram avaliadas as habilidades de desenvolvedores web na identificação de vulnerabilidades em aplicações. Os resultados revelaram que, embora os participantes tivessem desempenho satisfatório em algumas categorias, apenas 61% foram capazes de identificar corretamente a vulnerabilidade específica de *Session Hijacking*, uma falha crítica com potencial de comprometimento severo da segurança das aplicações.

Ao submeter os mesmos trechos de código à análise da extensão desenvolvida com IA Generativa, observou-se que a ferramenta conseguiu identificar corretamente todas as vulnerabilidades presentes, incluindo falhas complexas como *Session Hijacking*. Esse resultado evidencia o elevado potencial da Inteligência Artificial como ferramenta de apoio ao desenvolvimento seguro, demonstrando capacidade não apenas de acompanhar, mas em certos casos até superar, o desempenho de profissionais experientes na identificação de ameaças à segurança do software.

Tabela 3 – Resumo dos Experimentos da Prova de Conceito

Experimento	**Descrição**	**Resultados**
Análise de Códigos Reconhecidamente Vulneráveis	Códigos obtidos do repositório *Vulnerable Code Snippets* no GitHub foram analisados individualmente pela extensão *MN-Analise*, solicitando a identificação e explicação das vulnerabilidades.	34 vulnerabilidades testadas. A ferramenta identificou corretamente 30, resultando em uma taxa de acerto de **88,24%**. Ofereceu sugestões úteis nos casos parcialmente detectados.
Análise Comparativa com Desenvolvedores	Trechos de código já avaliados por desenvolvedores no estudo de Costa et al. (2018) foram reanalisados pela extensão. O objetivo foi comparar o desempenho da IA com o de profissionais humanos.	A extensão identificou corretamente todas as vulnerabilidades, incluindo falhas críticas como *Session Hijacking*, superando os **61%** de acerto de humanos.

6 Estudo de Caso

Com o objetivo de avaliar a aplicabilidade prática da extensão desenvolvida, foi conduzido um estudo de caso utilizando o sistema WeGIA — Web Gerenciador para Instituições Assistenciais (LAZARIN; ESCALFONI; FERREIRA, 2024). O WeGIA é um software livre projetado para auxiliar na gestão de instituições assistenciais, oferecendo módulos como cadastro de pessoas, controle de patrimônio, memorandos institucionais, gestão de saúde, cadastro de animais, entre outros.

Este sistema, por ser de código aberto, apresenta uma excelente oportunidade para aplicação de ferramentas de apoio à segurança, permitindo validar a eficácia da extensão em um ambiente real, com código desenvolvido de forma colaborativa e sujeito às mesmas limitações e desafios encontrados em sistemas corporativos.

A metodologia adotada consistiu na realização de uma varredura completa no repositório do WeGIA. Para cada ocorrência de possível vulnerabilidade detectada pela extensão, uma *issue* foi aberta no repositório. Este processo contou com a valiosa colaboração da própria equipe de desenvolvimento do WeGIA, que auxiliou na tarefa de documentar cada achado, registrando o arquivo analisado, o parâmetro potencialmente vulnerável e o tipo de vulnerabilidade associado.

Para validar os resultados obtidos, as respostas fornecidas pela extensão foram comparadas com a análise conduzida pelo grupo de pesquisa de vulnerabilidades **CVE Hunters**[1]. A comparação foi realizada de forma criteriosa, considerando não apenas se a IA e a equipe humana identificaram o mesmo parâmetro como vulnerável em um determinado arquivo, mas também se atribuíram corretamente o tipo de vulnerabilidade presente. Dessa forma, a avaliação contemplou tanto a precisão na detecção dos parâmetros sensíveis quanto na correta classificação das vulnerabilidades encontradas.

6.1 Resultados em Casos Reais (WeGIA)

Para avaliar a eficácia da extensão desenvolvida, foi realizada uma varredura completa no repositório do sistema WeGIA. Como resultado desse processo, foram abertas um total de 59 issues, cada uma representando uma ocorrência de potencial vulnerabilidade detectada pela ferramenta.

As vulnerabilidades foram categorizadas de acordo com o seu tipo, conforme apresentado na Tabela 4.

[1] O CVE Hunters (<https://www.cvehunters.com/>) é um grupo de pesquisa focado em identificar e documentar vulnerabilidades em projetos de código aberto.

Tabela 4 – Categorias de Vulnerabilidades Identificadas

	Vulnerabilidade
1	Injeção de Código
2	XSS (Cross-site Scripting)
3	LFI (Local File Inclusion)
4	SQL Injection
5	Redirecionamento Não Autorizado
6	Falta de Validação em Uploads
7	Directory Traversal / Path Traversal
8	Falta de Proteção Contra CSRF

6.1.1 Critérios de Avaliação

Para a análise dos resultados, foram considerados dois aspectos complementares:

- A **classificação correta do tipo de vulnerabilidade** associada a cada ocorrência.
- A **detecção dos parâmetros** considerados sensíveis ou vulneráveis dentro dos arquivos.

Cada ocorrência foi analisada e classificada em uma das seguintes categorias, conforme descrito abaixo:

- **Verdadeiro Positivo (VP)**: Quando a extensão detectou uma vulnerabilidade, seja em um parâmetro específico ou na classificação do tipo, e essa detecção foi confirmada pela equipe de segurança. Ou seja, tanto a IA quanto os analistas humanos concordaram que há uma vulnerabilidade naquela ocorrência.
- **Falso Positivo (FP)**: Quando a extensão indicou uma vulnerabilidade (em um parâmetro ou na classificação do tipo), mas essa não foi confirmada pela equipe de segurança na análise manual.
- **Falso Negativo (FN)**: Quando uma vulnerabilidade foi identificada pela equipe de segurança, porém não foi detectada pela extensão durante o processo de varredura.

6.1.2 Distribuição das Vulnerabilidades

A Figura 7 apresenta a quantidade de ocorrências para cada tipo de vulnerabilidade detectada no repositório do sistema WeGIA. Esta distribuição permite compreender quais tipos de falhas foram mais recorrentes no sistema analisado.

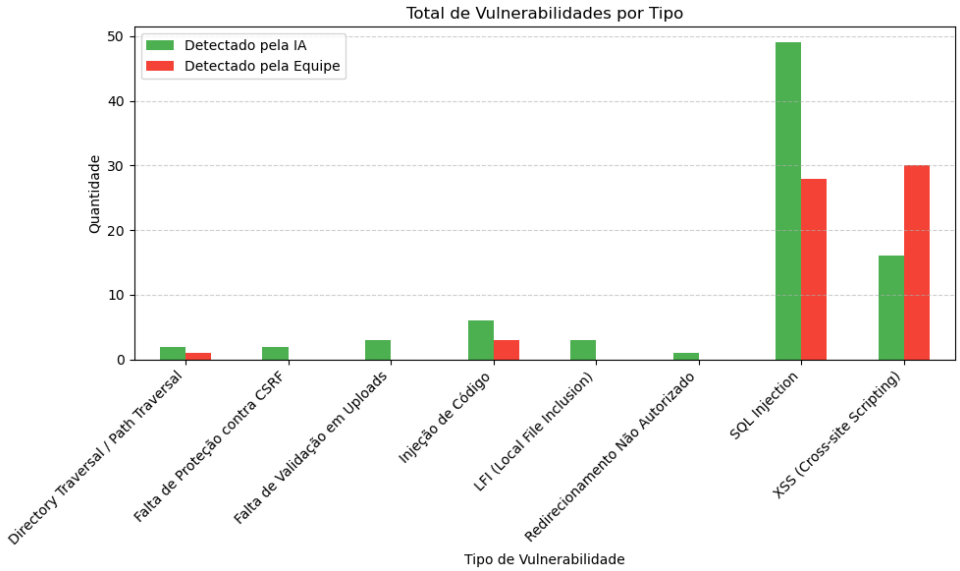

Figura 7 – Distribuição da quantidade de cada tipo de vulnerabilidade.

Observou-se que, dentre as 59 *issues* analisadas, as vulnerabilidades com maior ocorrência foram SQL Injection e XSS (Cross-site Scripting). A ferramenta baseada em IA detectou um total de 49 ocorrências de SQL Injection e 16 de XSS, enquanto a equipe de segurança identificou 28 ocorrências de SQL Injection e 30 de XSS.

6.1.3 Análise por Tipo de Vulnerabilidade (Geral)

Nesta etapa, o objetivo da análise foi exclusivamente comparar se o tipo de vulnerabilidade apontado pela ferramenta de IA coincidia com o tipo identificado pela equipe de segurança humana. Desta forma, a avaliação desconsiderou os parâmetros ou trechos de código específicos, focando unicamente na correspondência da classificação (por exemplo, se ambos identificaram a falha como 'SQL Injection').

Os resultados quantitativos desta análise estão apresentados a seguir:

- **Verdadeiros Positivos (VP):** 37
- **Falsos Positivos (FP):** 46
- **Falsos Negativos (FN):** 25
- **Precisão:** 44,58%
- **Revocação (Recall):** 59,68%
- **F1-Score:** 51,03%

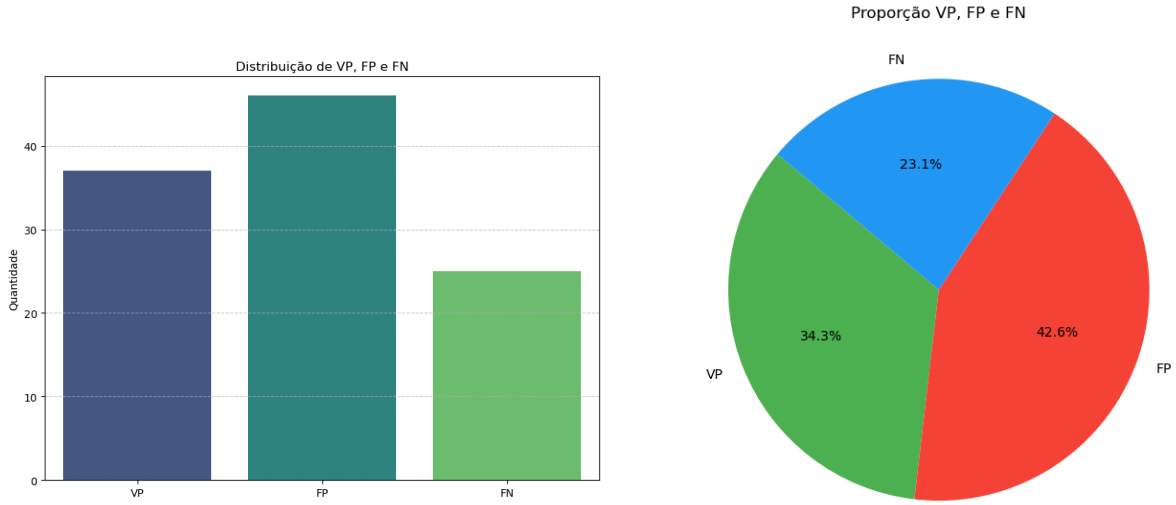

Figura 8 – Distribuição e proporção na análise por tipo de vulnerabilidade.

A revocação de **59,68%** indica que a ferramenta conseguiu identificar a maioria das vulnerabilidades encontradas pela equipe de segurança. Por outro lado, a precisão de **44,58%** evidencia uma quantidade significativa de falsos positivos, o que sugere uma tendência da Inteligência Artificial em apontar uma quantidade maior de tipos de vulnerabilidades que não foram identificadas pela equipe de segurança. Este comportamento pode estar relacionado a dois fatores: primeiramente, à própria natureza conservadora do modelo, que prioriza a identificação de potenciais vulnerabilidades, mesmo que isso aumente o número de alarmes incorretos; e, em segundo lugar, à possibilidade de que parte dessas vulnerabilidades realmente exista, mas não tenha sido reconhecida pela equipe humana, seja por limitações de tempo, conhecimento específico ou pela complexidade do código analisado. O **F1-Score de 51,03%** representa um equilíbrio moderado entre precisão e revocação, reforçando que a extensão pode ser uma ferramenta valiosa como apoio ao desenvolvimento seguro, desde que utilizada em conjunto com análises manuais para maximizar sua efetividade.

6.1.4 Análise Focada em SQL Injection

A segunda etapa consistiu em uma análise específica sobre as ocorrências em que a equipe de segurança classificou como **SQL Injection**. A escolha desta métrica se justifica por ser uma das vulnerabilidades com maior incidência nas *issues* analisadas, além de representar uma das falhas mais críticas e recorrentes em aplicações web. O objetivo foi avaliar se a extensão baseada em IA foi capaz de identificar corretamente esse tipo específico de vulnerabilidade.

Os resultados quantitativos desta análise estão apresentados a seguir:

- **Verdadeiros Positivos (VP):** 25

- **Falsos Positivos (FP):** 9
- **Falsos Negativos (FN):** 6
- **Precisão:** 73,53%
- **Revocação (Recall):** 80,65%
- **F1-Score:** 76,92%

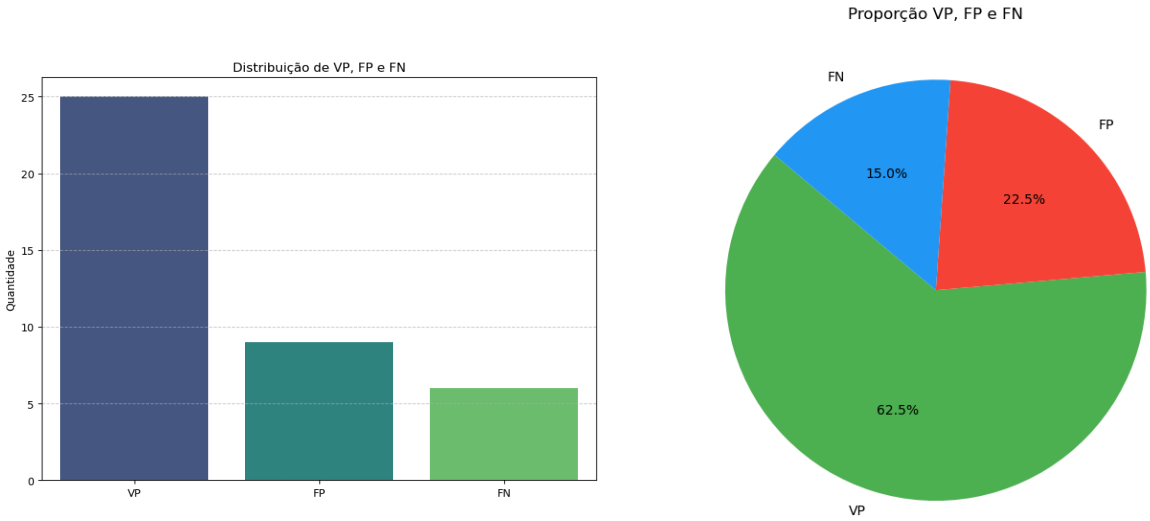

Figura 9 – Distribuição e proporção na análise focada em SQL Injection.

Os resultados demonstram que a extensão apresentou um desempenho significativamente superior na detecção de SQL Injection em comparação com a análise geral. A **revocação de 80,65%** indica que a IA conseguiu identificar a grande maioria das ocorrências reais desse tipo de vulnerabilidade. Além disso, a **precisão de 73,53%** reflete uma redução considerável no número de falsos positivos em relação à análise geral, o que demonstra uma maior assertividade do modelo quando foca especificamente em SQL Injection. O **F1-Score de 76,92%** representa um equilíbrio robusto entre precisão e revocação, indicando que, para esse tipo de falha, a ferramenta baseada em IA se mostra bastante eficiente como apoio no processo de desenvolvimento seguro.

6.1.5 Análise Focada em XSS

A terceira etapa consistiu em uma análise específica sobre as ocorrências em que a equipe de segurança classificou como **XSS (Cross-site Scripting)**. Assim como na etapa anterior, a escolha deste recorte se justifica por se tratar de uma das vulnerabilidades mais presentes nas *issues* analisadas, além de representar uma falha crítica amplamente explorada em aplicações web.

Os resultados quantitativos desta análise estão apresentados a seguir:

- **Verdadeiros Positivos (VP):** 12

- **Falsos Positivos (FP):** 35
- **Falsos Negativos (FN):** 21
- **Precisão:** 25,53%
- **Revocação (Recall):** 36,36%
- **F1-Score:** 30,00%

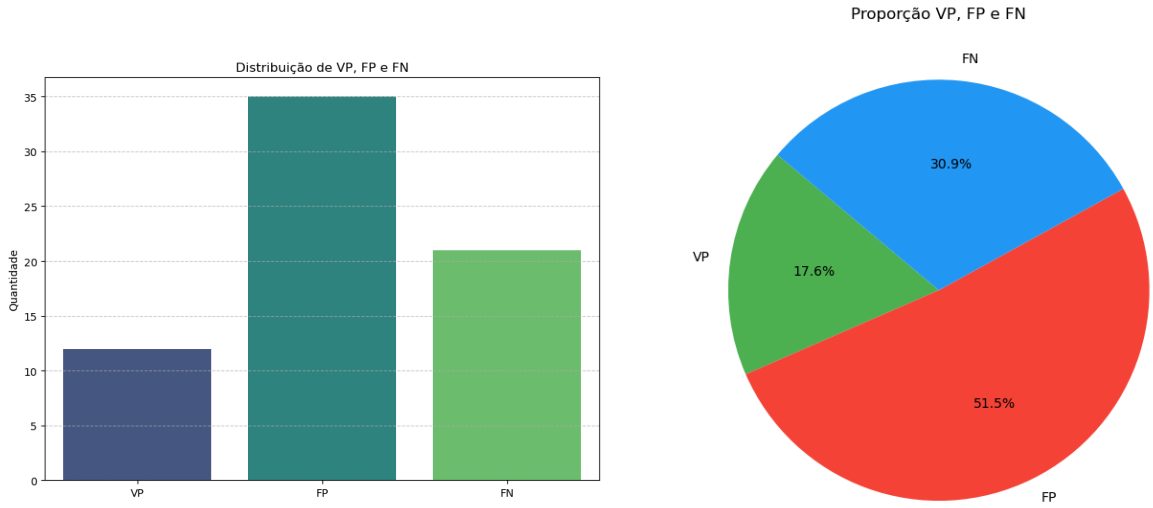

Figura 10 – Distribuição e proporção na análise focada em XSS.

Os resultados desta análise indicam um desempenho inferior da extensão na detecção de vulnerabilidades do tipo XSS em comparação às demais análises realizadas. A **revocação de 36,36%** demonstra que a IA foi capaz de identificar pouco mais de um terço das ocorrências reais desta falha. Além disso, a **precisão de 25,53%** revela uma alta taxa de falsos positivos, indicando que o modelo teve dificuldades em distinguir corretamente os padrões associados ao XSS, gerando muitos alertas incorretos. O **F1-Score de 30,00%** reflete esse desequilíbrio, evidenciando que a detecção desse tipo específico de vulnerabilidade ainda representa um desafio significativo para o modelo de IA utilizado, possivelmente devido à maior dependência de contexto e semântica na identificação de falhas desse tipo.

6.1.6 Análise por Parâmetros Vulneráveis

A quarta e última etapa da análise teve como foco a avaliação da capacidade da extensão em identificar corretamente os **parâmetros vulneráveis** presentes nos arquivos analisados, independentemente do tipo de vulnerabilidade associado. A adoção deste critério se justifica pelo fato de que, muitas vezes, não é possível afirmar com absoluta certeza qual seria o tipo de vulnerabilidade mais adequado em cada ocorrência. Isso se deve, principalmente, às diferentes interpretações que podem ocorrer durante uma análise de segurança, seja pela equipe humana ou pela própria IA. Em diversos casos, uma mesma falha pode ser classificada de formas distintas, dependendo do contexto, da perspectiva de

quem analisa ou da abordagem adotada para avaliar o risco. Dessa forma, a análise por parâmetro busca reduzir esse viés de classificação, concentrando-se na identificação dos elementos do código que estão suscetíveis a ataques, especialmente aqueles diretamente relacionados à manipulação de entradas, como parâmetros em requisições. Este critério é, portanto, fundamental para garantir uma avaliação mais objetiva e alinhada com o risco real, independentemente da categorização específica da vulnerabilidade.

Os resultados quantitativos desta análise estão apresentados a seguir:

- **Verdadeiros Positivos (VP):** 42
- **Falsos Positivos (FP):** 32
- **Falsos Negativos (FN):** 30
- **Precisão:** 56,76%
- **Revocação (Recall):** 58,33%
- **F1-Score:** 57,53%

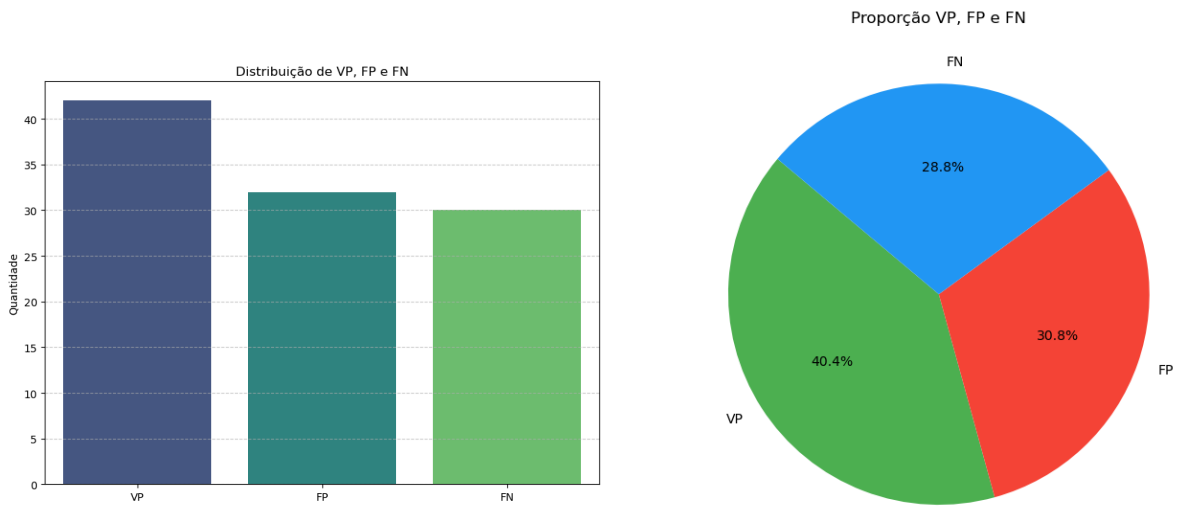

Figura 11 – Distribuição e proporção na análise por parâmetros vulneráveis.

Os resultados desta etapa mostram um desempenho intermediário da extensão na detecção de parâmetros vulneráveis. A **revocação de 58,33%** indica que a ferramenta foi capaz de identificar mais da metade dos parâmetros que realmente representavam risco. Por outro lado, a **precisão de 56,76%** demonstra que, apesar de uma quantidade considerável de falsos positivos, o modelo conseguiu manter um equilíbrio entre acertos e erros. O **F1-Score de 57,53%** reflete esse equilíbrio, posicionando a análise por parâmetros com uma performance melhor que a observada para XSS e na análise por tipo de vulnerabilidade, mas inferior à obtida na análise focada em SQL Injection.

7 Conclusão

Este trabalho explorou a aplicação da Inteligência Artificial Generativa como uma ferramenta de apoio ao desenvolvimento de software seguro, buscando responder à questão de como a IA pode auxiliar desenvolvedores em tempo real. Para este fim, foi desenvolvida uma extensão para o Visual Studio Code, denominada "MN-Analise", que se integra a múltiplos provedores de IA, como ChatGPT, Google Gemini e Ollama, para analisar trechos de código e identificar potenciais vulnerabilidades.

A avaliação da proposta foi realizada em múltiplas etapas. A prova de conceito demonstrou a alta viabilidade da abordagem, alcançando uma taxa de eficácia de 88,24

O estudo de caso, conduzido no sistema de código aberto WeGIA, forneceu uma análise prática da ferramenta em um ambiente real. Os resultados revelaram um desempenho excepcional na detecção de *SQL Injection*, com um F1-Score de 76,92%, indicando alta precisão e revocação para este tipo de falha. Contudo, a análise de *Cross-site Scripting* (XSS) se mostrou um desafio, com um F1-Score de apenas 30,00%, o que sugere uma limitação do modelo em lidar com vulnerabilidades que exigem maior compreensão contextual e semântica. A análise focada na identificação de parâmetros vulneráveis, independentemente do tipo de falha, apresentou um desempenho equilibrado (F1-Score de 57,53%).

Conclui-se que a integração de ferramentas de IA no ambiente de desenvolvimento é uma estratégia promissora e eficaz para aprimorar a segurança do software. A extensão desenvolvida provou ser um recurso valioso, capaz de automatizar a detecção de vulnerabilidades críticas e complementar a análise humana. No entanto, a variação de desempenho entre os diferentes tipos de vulnerabilidades e a incidência de falsos positivos ressaltam que a tecnologia deve ser utilizada como uma ferramenta de apoio, e não como um substituto para a revisão de segurança realizada por especialistas. A abordagem proposta contribui significativamente ao integrar a segurança de forma contínua e acessível no fluxo de trabalho do desenvolvedor, fomentando a criação de códigos mais robustos e seguros desde a sua concepção.

7.1 Limitações Encontradas

Durante a execução deste trabalho, foram encontradas algumas limitações que influenciaram o escopo e a profundidade da análise. A principal dificuldade residiu na validação dos resultados gerados pela Inteligência Artificial. Determinar se uma vulnerabilidade apontada pela ferramenta era, de fato, um positivo verdadeiro exigiria um conhecimento aprofundado em segurança de software que a equipe não possuía integralmente.

Para contornar essa questão, adotou-se a comparação com as análises do grupo de pesquisa **CVE Hunters**. Embora essa abordagem tenha fornecido um referencial valioso, ela serviu como um método comparativo, e não como uma validação absoluta. Ou seja, a análise se restringiu a verificar a concordância entre a IA e os especialistas, sem a capacidade de arbitrar sobre a correção técnica de cada apontamento.

Outro desafio significativo foi a dificuldade em obter acesso a projetos de código aberto adequados para teste e a disponibilidade de ambientes para configurar e executar a ferramenta de varredura de forma ampla, o que limitou a escala da validação em cenários diversificados.

Por fim, a própria arquitetura da extensão, que utilizava um *prompt* fixo para interagir com as IAs, se mostrou uma limitação. Essa abordagem, embora garantisse consistência, restringia a capacidade de adaptar as perguntas a contextos de código mais complexos ou específicos, o que pode ter impactado a qualidade e a profundidade das respostas obtidas dos modelos de linguagem.

7.2 Trabalhos Futuros

Com base nos resultados e aprendizados deste trabalho, diversas oportunidades de pesquisa e desenvolvimento futuro podem ser exploradas para expandir e aprimorar a abordagem proposta:

- **Aprimoramento da Detecção de Vulnerabilidades Contextuais:** Investigar técnicas para melhorar a precisão na detecção de vulnerabilidades complexas como XSS, o que pode envolver o refinamento dos *prompts* ou treinamento de modelos específicos.
- **Análise Comparativa de Modelos de IA:** Realizar um estudo comparativo aprofundado entre os diferentes provedores de IA (ChatGPT, Gemini, Ollama, etc.) para avaliar sistematicamente seus pontos fortes e fracos na detecção de tipos específicos de vulnerabilidades de segurança.
- **Sugestão de Correções Automatizadas:** Evoluir a ferramenta para não apenas identificar, mas também sugerir e, eventualmente, aplicar correções de código automatizadas, guiando o desenvolvedor na mitigação efetiva das falhas encontradas.
- **Estudos de Usabilidade em Larga Escala:** Conduzir estudos com um número maior e mais diversificado de desenvolvedores para coletar feedback sobre a usabilidade da extensão e seu impacto real na produtividade e na qualidade do código em diferentes contextos de desenvolvimento.

Referências

ABNT. **Tecnologia da informação - técnicas de segurança - código de prática para controles de segurança da informação**. [S.l.]: ABNT, 2013. ISBN 9788507046134.

ALMEIDA, A. B. G.; SILVA, J. L. da. Inteligência artificial aplicada à segurança da informação. **Revista Perquirere**, v. 17, n. 2, p. 241–254, maio/ago 2020. Acesso em: 30 jun. 2025. Disponível em: <https://revistas.unipam.edu.br/index.php/perquirere/article/view/2040>.

ALMEIDA, Y.; ALBUQUERQUE, D.; FILHO, E. D.; MUNIZ, F.; de Farias Santos, K.; PERKUSICH, M.; ALMEIDA, H.; PERKUSICH, A. Aicodereview: Advancing code quality with ai-enhanced reviews. **SoftwareX**, v. 26, p. 101677, 2024. ISSN 2352-7110. Disponível em: <https://www.sciencedirect.com/science/article/pii/S2352711024000487>.

ASLAN, ; AKTUğ, S. S.; OZKAN-OKAY, M.; YILMAZ, A. A.; AKIN, E. A comprehensive review of cyber security vulnerabilities, threats, attacks, and solutions. **Electronics**, v. 12, n. 6, 2023. ISSN 2079-9292. Disponível em: <https://www.mdpi.com/2079-9292/12/6/1333>.

BAILLY, S.; MEYFROIDT, G.; TIMSIT, J.-F. What's new in ICU in 2050: big data and machine learning. **Intensive Care Med**, United States, v. 44, n. 9, p. 1524–1527, dez. 2017. Disponível em: <https://doi.org/10.1007/s00134-017-5034-3>.

BRYNJOLFSSON, E.; LI, D.; RAYMOND, L. R. **Generative AI at Work**. [S.l.], 2023. (Working Paper Series, 31161). Disponível em: <http://www.nber.org/papers/w31161>.

COSTA, P. V.; GONÇALVES, W. I.; GONÇALVES, E. D.; LAZARIN, N. M. Nível de conhecimento de desenvolvedores sobre segurança em aplicações web: Pesquisa e análise. In: **Anais da V Escola Regional de Sistemas de Informação do Rio de Janeiro**. Porto Alegre, RS, Brasil: SBC, 2018. p. 92–99. Disponível em: <https://doi.org/10.5753/ersirj.2018.4661>.

ELIAS, S. I. O IMPACTO DA INTELIGÊNCIA ARTIFICIAL NO COMPORTAMENTO ORGANIZACIONAL. **Revista Ilustração**, v. 4, n. 3, p. 33–39, set. 2023. Disponível em: <https://doi.org/10.46550/ilustracao.v4i3.176>.

FERNANDEZ, A. S.; CORNELL, K. A. CS1 with a Side of AI: Teaching Software Verification for Secure Code in the Era of Generative AI. In: **Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 1**. New York, NY, USA: Association for Computing Machinery, 2024. (SIGCSE 2024), p. 345–351. ISBN 9798400704239. Disponível em: <https://doi.org/10.1145/3626252.3630817>.

FRAIJ, J.; VáRALLYAI, L. literature review: Artificial intelligence impact on the recruitment process. **International Journal of Engineering and Management Sciences**, v. 6, p. 108–119, 05 2021. Disponível em: <https://doi.org/10.21791/IJEMS.2021.1.10.>

GLUSHKOVA, D. Trabalho de Conclusão de Curso (Mestrado), **The influence of Artificial Intelligence on productivity in Software Development**. 2023. Rel. Anna D'Ambrosio. Politecnico di Torino, Corso di laurea magistrale in Ingegneria Gestionale (Engineering and Management). Disponível em: <https://webthesis.biblio.polito.it/28435/>.

HALKIDIS, S. T.; TSANTALIS, N.; CHATZIGEORGIOU, A.; STEPHANIDES, G. Architectural risk analysis of software systems based on security patterns. **IEEE Transactions on Dependable and Secure Computing**, v. 5, n. 3, p. 129–142, 2008. Disponível em: <https://ieeexplore.ieee.org/document/4384502>.

HöRNEMALM, A. **ChatGPT as a Software Development Tool: The Future of Development**. [S.l.]: Umeå University, 2023. <https://urn.kb.se/resolve?urn=urn:nbn:se:umu:diva-209909>. Acesso em: 17 maio 2024.

KITCHENHAM, B.; CHARTERS, S. **Guidelines for performing systematic literature reviews in software engineering**. [S.l.], 2007. Disponível em: <https://www.researchgate.net/publication/302924724_Guidelines_for_performing_Systematic_Literature_Reviews_in_Software_Engineering/>.

KLOPPER, R.; GRUNER, S.; KOURIE, D. Assessment of a framework to compare software development methodologies. In: . [s.n.], 2007. p. 56–65. Disponível em: <https://www.researchgate.net/publication/220803599_Assessment_of_a_framework_to_compare_software_development_methodologies>.

LAZARIN, N.; ESCALFONI, R. E.; FERREIRA, V. Wegia: Web gerenciador para instituições assistenciais. In: **Anais do XXI Congresso Latino-Americano de Software Livre e Tecnologias Abertas**. Porto Alegre, RS, Brasil: SBC, 2024. p. 322–330. Disponível em: <https://doi.org/10.5753/latinoware.2024.245668>.

LEITE, E.; RIBEIRO, D. PAPEL TRANSFORMADOR DA INTELIGÊNCIA ARTIFICIAL NA SEGURANÇA. **Revista Interface Tecnológica**, v. 20, p. 181–190, 10 2023. Disponível em: <https://doi.org/10.31510/infa.v20i1.1669>.

MARTíN-MARTíN, A.; ORDUNA-MALEA, E.; THELWALL, M.; Delgado López-Cózar, E. Google scholar, web of science, and scopus: A systematic comparison of citations in 252 subject categories. **Journal of Informetrics**, v. 12, n. 4, p. 1160–1177, 2018. ISSN 1751-1577. Disponível em: <https://www.sciencedirect.com/science/article/pii/S1751157718303249>.

MASKURI, F.; OTHMAN, M.; OSMAN, I.; KASSIM, S.; RAZAK, N. The impact of social media usage on the organisation's reputation risk through its cybersecurity. **International Journal of Academic Research in Business and Social Sciences**, v. 13, n. 2, 2023. Disponível em: <http://dx.doi.org/10.6007/IJARBSS/v13-i2/16199>.

NETO, M. R. d. A.; LAZARIN, N. M. Uma extensão para o VSCode que utiliza o ChatGPT como ferramenta de apoio ao desenvolvimento de software seguro. In: **Anais do XV Computer on the Beach - COTB'24**. Balneário Camboriú - Santa Catarina - Brasil: Universidade do Vale do Itajaí, 2024. p. 310–311. Disponível em: <https://doi.org/10.14210/cotb.v15.p310-311>.

OROSZ, G. **The Pulse #134: Stack Overflow is almost dead**. The Pragmatic Engineer, 2025. Accessed: 20 Aug. 2025. Disponível em: <https://newsletter.pragmaticengineer.com/p/the-pulse-134?ref=blog.pragmaticengineer.com>.

RUSSELL, S. J.; NORVIG, P. **Artificial Intelligence: A Modern Approach**. 4. ed. Hoboken: Pearson, 2021. (Pearson Series in Artificial Intelligence). Inclui referências bibliográficas e índice. LCCN 2019047498. ISBN 9780134610993. Disponível em: <https://lccn.loc.gov/2019047498>.

SANTOS, C. M. d. C.; PIMENTA, C. A. d. M.; NOBRE, M. R. C. A estratégia pico para a construção da pergunta de pesquisa e busca de evidências. **Revista Latino-Americana de Enfermagem**, v. 15, n. 3, p. 508–511, jun. 2007. Disponível em: <https://revistas.usp.br/rlae/article/view/2463>.

Security, Snoopy. **Broken Vulnerable Code Snippets**. 2024. <https://github.com/snoopysecurity/Broken-Vulnerable-Code-Snippets>. Acesso em: 30 jun. 2025.

SHANAHAN, M. Talking about large language models. **Commun. ACM**, Association for Computing Machinery, New York, NY, USA, v. 67, n. 2, p. 68–79, jan. 2024. ISSN 0001-0782. Disponível em: <https://doi.org/10.1145/3624724>.

SHUKLA, A. Cloud-based lightweight modern integrated development environments (ides) and their future. **Journal of Artificial Intelligence Cloud Computing**, 01 2024. Disponível em: <https://www.researchgate.net/publication/378567865_Cloud-Based_Lightweight_Modern_Integrated_Development_Environments_IDEs_and_their_Future>.

STAVRIDIS, A.; DRUGGE, A. **The Rise of Intelligent System Development: A Qualitative Study of Developers' Views on AI in Software Development Processes**. 2023. Disponível em: <https://www.diva-portal.org/smash/record.jsf?pid=diva2:1759153>.

TAI, M. C. The impact of artificial intelligence on human society and bioethics. **Tzu Chi Medical Journal**, v. 32, n. 4, 2020. ISSN 1016-3190. Disponível em: <https://journals.lww.com/tcmj/fulltext/2020/32040/the_impact_of_artificial_intelligence_on_human.5.aspx>.

Trend Micro. **Stepping Ahead of Risk: Trend Micro 2023 Midyear Cybersecurity Threat Report**. 2023. <https://documents.trendmicro.com/images/TEx/articles/Risk_Landscape_infographic-aypd962.png>. Acesso em: 30 jun. 2025.

VIDAL-ALABALL, J.; Panadés Zafra, R.; ESCALé-BESA, A.; MARTINEZ-MILLANA, A. The artificial intelligence revolution in primary care: Challenges, dilemmas and opportunities. **Atención Primaria**, v. 56, n. 2, p. 102820, 2024. ISSN 0212-6567. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0212656723002536>.

YANG, X.; CHEN, A.; POURNEJATIAN, N.; SHIN, H. C.; SMITH, K. E.; PARISIEN, C.; COMPAS, C.; MARTIN, C.; COSTA, A. B.; FLORES, M. G.; ZHANG, Y.; MAGOC, T.; HARLE, C. A.; LIPORI, G.; MITCHELL, D. A.; HOGAN, W. R.; SHENKMAN, E. A.; BIAN, J.; WU, Y. A large language model for electronic health records. **npj Digital Medicine**, v. 5, n. 1, p. 194, dez. 2022. ISSN 2398-6352. Disponível em: <https://www.nature.com/articles/s41746-022-00742-2>.