EL IMPACTO DE LA INTELIGENCIA ARTIFICIAL EN LOS PROGRAMADORES. ¿MEJORA LA PRODUCTIVIDAD O FOMENTA LA DEPENDENCIA?

Sebastian Alejandro Rico Tautiva

INSTITUCIÓN UNIVERSITARIA PASCUAL BRAVO

FACULTAD DE INGENIERÍA

TECNOLOGÍA EN DESARROLLO DE SOFTWARE

MEDELLÍN

2025

EL IMPACTO DE LA INTELIGENCIA ARTIFICIAL EN LOS PROGRAMADORES. ¿MEJORA LA PRODUCTIVIDAD O FOMENTA LA DEPENDENCIA?

Sebastian Alejandro Rico Tautiva

Trabajo de grado para optar al título de Tecnólogo en Desarrollo de Software

Asesor

Magister Yudy Andrea Quintero Tangarife

INSTITUCIÓN UNIVERSITARIA PASCUAL BRAVO

FACULTAD DE INGENIERÍA

TECNOLOGÍA EN DESARROLLO DE SOFTWARE

MEDELLÍN

2025

Contenido

Tablas

Figuras

Introducción

En los últimos años, la inteligencia artificial se ha integrado en diferentes áreas tecnológicas, incluyendo el desarrollo de software. Herramientas como GitHub Copilot, ChatGPT y otras plataformas automatizadas permiten generar código, depuración, encontrar soluciones a los errores y documentar sistemas, facilitando y mejorando muchas tareas que antes eran exclusivamente humanas.

Sin embargo, a pesar de los grandes avances y beneficios evidentes que trae esta tecnología, también han comenzado a surgir preocupaciones con respecto a cómo puede afectar las habilidades lógicas y analíticas de los programadores. Se teme que la dependencia de estas herramientas pueda reducir la capacidad de resolver problemas por cuenta propia, cambiar la forma en que se obtienen conocimientos de programación y afectar la calidad del aprendizaje en nuevos desarrolladores.

Según los estudios de la consultora (McKinsey, 2023) muestran que el 38% de los desarrolladores en entornos laborales usan herramientas de inteligencia artificial al menos una vez por día. Así mismo, encuestas realizadas por Stack Overflow en 2023 indican que muchos programadores sienten que estas herramientas les hacen trabajar más rápido, pero también admiten haber perdido práctica en tareas que antes dominaban.

Por lo cual, surgió la necesidad de analizar de manera objetiva el verdadero impacto de la inteligencia artificial en el desarrollo profesional de los programadores. Este estudio tuvo como objetivo determinar si estas herramientas se están usando como un recurso que potencia la productividad y la calidad del trabajo, o si al contrario, están generando una dependencia que afecta la autonomía, la creatividad y las capacidades analíticas de quienes las utilizan. A través de esta investigación, se buscó aportar una reflexión crítica sobre el equilibrio necesario entre el aprovechamiento de la tecnología y el fortalecimiento de las competencias humanas dentro del campo del desarrollo de software.

Resumen

La inteligencia artificial ha cambiado el campo del desarrollo de software, introduciendo herramientas que ayudan a los programadores en la generación de código, la depuración de errores y la documentación de sistemas. Plataformas como GitHub Copilot y ChatGPT representan un avance significativo en la automatización de tareas, mejorando la productividad y eficiencia del trabajo de los desarrolladores. Sin embargo, el uso continuo de estas tecnologías ha generado dudas sobre el posible impacto en las habilidades cognitivas, analíticas y de resolución de problemas de los programadores.

Este proyecto analizó el impacto del uso de herramientas de inteligencia artificial en la productividad, autonomía y desarrollo profesional de los programadores. Mediante una revisión sistemática de literatura y el análisis de estudios de caso recientes, se identificaron los principales beneficios y riesgos relacionados a su uso.

El estudio busca aportar a la comprensión del equilibrio necesario entre el uso de la inteligencia artificial y la preservación de las habilidades humanas esenciales en la programación, promoviendo un uso ético, responsable y formativo de estas tecnologías dentro del entorno académico y profesional.

Palabras clave: inteligencia artificial, productividad, programación, dependencia tecnológica, pensamiento crítico, autonomía.

Abstract

Artificial intelligence has transformed the field of software development by introducing tools that assist programmers in code generation, debugging, and system documentation. Platforms such as GitHub Copilot and ChatGPT represent a significant advancement in task automation, enhancing developer productivity and work efficiency. However, the continued use of these technologies has raised concerns about their potential impact on programmers' cognitive, analytical, and problem-solving skills.

This project analyzed the impact of using artificial intelligence tools on the productivity, autonomy, and professional development of programmers. Through a systematic literature review and the analysis of recent case studies, the main benefits and risks associated with their use were identified.

The study seeks to contribute to the understanding of the necessary balance between the use of artificial intelligence and the preservation of essential human skills in programming. Its goal is to promote an ethical, responsible, and formative use of these technologies within both academic and professional environments.

Keywords: Artificial Intelligence, Productivity, Programming, Technological Dependence, Critical Thinking, Autonomy.

Glosario

Inteligencia Artificial: Campo de estudio de la informática que se enfoca en el diseño y desarrollo de sistemas capaces de realizar tareas que normalmente requieren inteligencia humana, como el razonamiento, el aprendizaje, la percepción y la toma de decisiones. Se basa en algoritmos, redes neuronales, aprendizaje automático y procesamiento de datos masivos para simular comportamientos cognitivos**.**

GitHub Copilot: Herramienta de asistencia basada en inteligencia artificial desarrollada por GitHub y OpenAI que utiliza modelos de lenguaje entrenados en grandes cantidades de código fuente. Genera sugerencias automáticas de código y funciones en tiempo real, facilitando la escritura y comprensión de programas.

ChatGPT: Modelo de lenguaje avanzado creado por OpenAI que emplea técnicas de aprendizaje profundo para procesar y generar texto de forma coherente y contextual. En el ámbito del desarrollo de software, puede asistir en la generación de código, documentación técnica, depuración y análisis de problemas lógicos.

Automatización: Uso de tecnologías, sistemas o algoritmos para ejecutar tareas sin intervención humana directa. En la ingeniería de software, la automatización reduce la carga manual en actividades como pruebas, compilación, integración continua o generación de código.

Modelos de Lenguaje: Sistemas basados en redes neuronales profundas entrenadas con grandes volúmenes de datos textuales para comprender y generar lenguaje natural. En la actualidad, estos modelos son el núcleo funcional de asistentes de programación y chatbots inteligentes.

Dependencia Tecnológica: Condición en la que un individuo se vuelve excesivamente dependiente del uso de herramientas o tecnologías externas, disminuyendo su capacidad de actuar de manera autónoma. En este estudio, se analiza la dependencia que pueden generar las herramientas de IA en los programadores.

Pensamiento Crítico: Habilidad cognitiva que permite analizar, evaluar y tomar decisiones fundamentadas con base en el razonamiento lógico. En programación, se manifiesta al identificar errores, optimizar algoritmos y plantear soluciones efectivas sin depender de asistentes automáticos.

1. Planteamiento del problema

1.1 Descripción

En los últimos años, la inteligencia artificial se ha integrado en diferentes áreas tecnológicas, incluyendo el desarrollo de software. Herramientas como GitHub Copilot, ChatGPT y otras plataformas automatizadas permiten generar código, depuración, encontrar soluciones a los errores y documentar sistemas, facilitando y mejorando muchas tareas que antes eran exclusivamente humanas.

Sin embargo, a pesar de los grandes avances y beneficios evidentes que trae esta tecnología, también han comenzado a surgir preocupaciones con respecto a cómo puede afectar las habilidades lógicas y analíticas de los programadores. Se teme que la dependencia de estas herramientas pueda reducir la capacidad de resolver problemas por cuenta propia, cambiar la forma en que se obtienen conocimientos de programación y afectar la calidad del aprendizaje en nuevos desarrolladores.

Según los estudios de la consultora (McKinsey, 2023) muestran que el 38% de los desarrolladores en entornos laborales usan herramientas de inteligencia artificial al menos una vez por día.

Lo cual, llevó a preguntarse si estas tecnologías estaban promoviendo una mayor eficiencia o si, por el contrario, podrían estar generando una forma de dependencia tecnológica que podría afectar en gran medida negativamente el desarrollo profesional de los futuros programadores y su pensamiento crítico.

1.2 Formulación del problema

¿Teniendo en cuenta el uso creciente de herramientas de inteligencia artificial en el desarrollo de software, mejora la productividad y eficiencia de los programadores o fomenta una dependencia que afecta sus habilidades y autonomía profesional?.

2. Justificación

La importancia de este proyecto radica en que permitió analizar una situación actual y en crecimiento dentro del campo del desarrollo de software. La inteligencia artificial representa una de las transformaciones más grandes en el ámbito tecnológico, y su impacto en los profesionales de la programación puede tener consecuencias importantes a corto y largo plazo.

Desde el punto de vista social, este estudio puede ayudar a entender cómo se ven afectados los procesos de aprendizaje de los estudiantes de diferentes ocupaciones y programación, también se puede analizar la forma en que las empresas podrían adaptar sus modelos de formación y evaluación de talento.

A nivel tecnológico, ayuda a identificar si la Inteligencia Artificial realmente está mejorando el trabajo del desarrollador o, si por el contrario podría estar afectando el pensamiento crítico y la resolución de problemas complejos. También en términos prácticos, busca generar conciencia para realizar un uso equilibrado y ético de la inteligencia artificial en la industria del software.

3. Objetivos

3.1 Objetivo general

Analizar el impacto del uso de herramientas de inteligencia artificial en la productividad, autonomía, eficiencia y habilidades de los programadores actuales, evaluando si estas herramientas mejoran el rendimiento o fomentan una dependencia tecnológica.

3.2 Objetivos específicos

- Identificar y reunir investigaciones científicas recientes relacionadas con el uso de herramientas de inteligencia artificial en el desarrollo de software y su influencia en las habilidades, productividad y autonomía de los programadores.
- Analizar de manera crítica la evidencia existente en la literatura académica sobre los efectos positivos y negativos del uso de herramientas como ChatGPT y GitHub Copilot.
- Clasificar los estudios seleccionados según las dimensiones de productividad, eficiencia, autonomía profesional y desarrollo de habilidades técnicas.
- Proponer recomendaciones basadas en la evidencia analizada para promover un uso equilibrado, ético y formativo de la inteligencia artificial en el ámbito del desarrollo de software.

4. Marco teórico

4.1 La inteligencia artificial en el desarrollo de software

En los últimos años, la inteligencia artificial ha dejado de ser una promesa del futuro para convertirse en una herramienta normalizada dentro del mundo del desarrollo de software. Su llegada ha cambiado la forma en que se escriben líneas de código, se corrigen errores, se documentan procesos y hasta se diseñan soluciones difíciles. Herramientas como ChatGPT o GitHub Copilot no solo están facilitando el trabajo de los desarrolladores, sino que han comenzado a integrarse como parte natural de sus rutinas (Iftekhar Ahmed et al., 2025).

Esta transformación ha sido posible gracias al avance del aprendizaje automático y el procesamiento del lenguaje natural, estas tecnologías permiten a las máquinas entender y generar código de manera coherente y eficiente. Como resultado, muchos programadores ahora pueden centrarse en tareas más creativas o estratégicas, mientras dejan en las inteligencias artificiales las funciones repetitivas o mecánicas (Christoph Treude & Margaret-Anne Storey, 2025). Sin embargo, este avance también nos invita a reflexionar sobre qué papel queremos que juegue la inteligencia artificial en nuestro trabajo.

4.2 Productividad contra dependencia

Se ha generado un gran debate en este tema, el cual se centra en el equilibrio entre el aumento de la productividad y el riesgo de caer en una dependencia excesiva. Es innegable que estas herramientas pueden agilizar en gran medida tareas comunes: según algunos estudios, la inteligencia artificial puede reducir hasta un 50% el tiempo necesario para refactorizar código, generar funciones o encontrar errores simples. Pero no todo es ganancia inmediata.

Hay algunos críticos que informan sobre una posible pérdida de habilidades fundamentales. ¿Qué pasa si dejamos de pensar cómo resolver un problema porque confiamos en que la

inteligencia artificial lo hará por nosotros? Algunos expertos advierten que esta confianza puede disminuir la capacidad de análisis, de razonamiento lógico y hasta la creatividad de los programadores (Agnia Sergeyuk et al., 2025).

Aquí surge una pregunta importante: ¿Se están formando desarrolladores capaces de crear desde cero, o usuarios que solo saben completar lo que una máquina sugiere? Esta discusión se conecta con el concepto de automatización cognitiva, que plantea que cada vez más tareas mentales humanas están siendo delegadas a sistemas inteligentes (Brooke N. Macnamara et al., 2024). Una situación que, si no se maneja con conciencia, puede cambiar profundamente el rol del programador.

4.3 Impacto de la inteligencia artificial

Con la expansión de estas herramientas, el perfil del programador moderno está cambiando. Ya no se trata solo de dominar lenguajes como Python, Java o JavaScript, sino de entender cómo trabajar junto a sistemas inteligentes. Es decir, no basta con saber programar: hay que saber colaborar con la inteligencia artificial, comprobar lo que produce, adaptar el trabajo, y mantener una mirada crítica ante los resultados.

Por eso, el pensamiento crítico y la capacidad de evaluación son más importantes que nunca. Los desarrolladores deben aprender no solo a usar estas herramientas, sino a identificar cuándo están fallando, cuándo es mejor confiar en su propia lógica, y cómo complementar el trabajo de la máquina con su propio juicio (Feiming Li et al., 2025).

De la misma manera , la inteligencia artificial se ha convertido en una de las herramientas más notables. Esta tecnología, que incluye aplicaciones como ChatGPT, tiene la capacidad de transformar la educación al generar contenido, facilitar la comprensión, personalizar el aprendizaje y apoyar diversas disciplinas (Thomas K.F. Chiu, 2024).

El impacto de estas herramientas en la participación de los estudiantes es multifacético. El compromiso estudiantil se define como una construcción multidimensional, que incluye componentes conductuales, cognitivos y emocionales (Zhanxin Hao et al., 2025).

A nivel educativo, este cambio también exige una transformación. Las instituciones deben preparar a los estudiantes no solo para escribir código, sino para hacerlo de forma ética, autónoma y con una comprensión profunda de las implicaciones que tiene el uso de la inteligencia artificial en su práctica profesional.

4.4 Uso responsable de herramientas de inteligencia artificial

Como cualquier tecnología poderosa, la inteligencia artificial debe usarse con responsabilidad. Aunque sus resultados pueden parecer precisos o hasta brillantes, la realidad es que las sugerencias generadas por modelos como ChatGPT o Copilot no siempre son correctas. Incluso, pueden contener errores graves, vulnerabilidades de seguridad, o conflictos con licencias de código que no siempre son visibles a simple vista (Onur Ceran, 2025).

De la misma manera, diversos autores han sugerido la necesidad recurrente de formar a la sociedad en general y a los alumnos en particular sobre la ética de la inteligencia artificial. Para un desarrollo sostenible y responsable de estas tecnologías, es imprescindible contar con un nivel elevado de comprensión, conocimiento y habilidades éticas en todos los sectores (Lucas J. Wiese et al., 2025). Esto refuerza la idea de que el futuro de la programación no se basará únicamente en el dominio técnico, sino en la capacidad de pensar con razonamiento crítico, actuar éticamente y colaborar de forma segura con la inteligencia artificial.

Esto nos recuerda que, detrás de cada sugerencia, debe haber un desarrollador que sepa verificar, comprender y decidir si lo generado es adecuado. También es importante tener presente el aspecto ético: usar inteligencia artificial no debería implicar apropiarse de

fragmentos de código sin saber su origen o autoría, aunque sea funcional debe estar el razonamiento primero.

Por eso, más allá del entusiasmo por la innovación, es necesario fomentar una cultura de supervisión, validación y aprendizaje constante. No se trata de rechazar la inteligencia artificial, sino de usarla como un apoyo que potencie nuestras capacidades, sin reemplazar la esencia de lo que significa programar (Satyam Kumar Navneet & Joydeep Chandra, 2025).

5. Metodología

5.1 Tipo de proyecto

Esta monografía es de tipo investigativo con enfoque descriptivo-analítico, orientado a desarrollar una revisión sistemática de literatura. El propósito es analizar de manera crítica la evidencia científica encontrada sobre el impacto del uso de herramientas de inteligencia artificial en la productividad, autonomía, eficiencia y habilidades de los programadores.

El enfoque descriptivo permite detallar las características observadas en los estudios seleccionados, mientras que el enfoque analítico permite examinar los resultados para establecer relaciones, patrones y vacíos de conocimiento.

5.2 Método

El método de investigación corresponde a una revisión sistemática de literatura, que se caracteriza por ser un proceso estructurado, reproducible y transparente de búsqueda, selección, análisis y síntesis de información científica.

El estudio seguirá un enfoque cualitativo, basado en la interpretación de resultados teóricos y empíricos reportados por investigaciones previas. Además, se aplicará un razonamiento inductivo y analítico: inductivo, porque a partir de hallazgos particulares de múltiples estudios se buscarán conclusiones generales; y analítico, porque se examinarán críticamente los métodos, resultados y limitaciones de las fuentes seleccionadas.

El desarrollo de la revisión sistemática se llevará a cabo en las siguientes fases:

A) **Formulación de la pregunta de investigación:**

   ¿El uso de herramientas de inteligencia artificial mejora la productividad y eficiencia de los programadores, o fomenta una dependencia que afecta sus habilidades y autonomía profesional?

B) **Diseño del protocolo de búsqueda:**

Se definieron las bases de datos académicas, palabras clave y criterios de inclusión y exclusión de los estudios a considerar.

Criterio	Descripción
1. Herramientas de IA Generadoras de Código	Se incluyeron estudios que analizan el uso de herramientas de inteligencia artificial como GitHub Copilot y Chat GPT en contextos de desarrollo de software.
2. Impacto en la Productividad o Dependencia	Se consideraron investigaciones que evalúan cómo estas herramientas afectan la productividad o autonomía de los programadores.
3) Relevancia con las Preguntas de Investigación	Solo se incluyeron estudios que responden o contribuyen a las preguntas clave del proyecto sobre rendimiento, habilidades o dependencia tecnológica.
4) Tipo de Estudio	Se aceptaron estudios empíricos (experimentos, encuestas, entrevistas, estudios de caso) y estudios teóricos (revisiones, análisis comparativos o revisiones sistemáticas previas) que aportan evidencia o discusión sobre el tema.

Tabla 1. Criterios de inclusión para la aceptación de estudios.

Criterio	Descripción
1. Fuera del ámbito Herramientas de IA Generadoras de Código	Se excluyeron los estudios que no analizan directamente herramientas de inteligencia artificial aplicadas al desarrollo de software, como GitHub Copilot, ChatGPT.
2. Falta de relación con el desarrollo o práctica de la programación	Se descartaron investigaciones centradas en cursos generales de informática, inteligencia artificial o ciencias de la computación sin un enfoque claro en la programación o en la generación de código asistida por IA
3) Ausencia de evidencia empírica o análisis teórico relevante	No se incluyeron estudios sin de datos empíricos, experiencias de usuarios, comparaciones o marcos teóricos que aporten a las preguntas de investigación relacionadas con productividad, autonomía o dependencia tecnológica de los programadores.
4) Accesibilidad	Se excluyeron artículos que no estén disponibles de manera gratuita o de acceso abierto.

Tabla 2. Criterios de exclusión para la aceptación de estudios.

C) Búsqueda y selección de literatura:

Se realizó una búsqueda completa en bases de datos científicas internacionales para identificar artículos, tesis y publicaciones relevantes entre los años 2022 y 2025.

D) Extracción y síntesis de información:

Se recopilaron los principales hallazgos, comparando resultados y clasificándolos según las dimensiones de productividad, eficiencia, autonomía profesional y desarrollo de habilidades.

E) Interpretación y conclusiones:

Se presentó un análisis crítico de la evidencia encontrada, identificando patrones, vacíos de conocimiento y posibles líneas de investigación futuras.

5.3 Instrumentos de recolección de información

El principal instrumento será una revisión sistemática de literatura, en la cual se registró la información relevante de cada estudio seleccionado. Así mismo, se emplearon estrategias de búsqueda avanzada utilizando operadores booleanos (AND, OR, NOT) y filtros necesarios que permitan obtener literatura pertinente al tema investigado.

Componente	Palabras clave
1) Herramientas de IA generadoras de código	(“GitHub Copilot” OR “ChatGPT” OR “AI code generation” OR “generative AI” OR “AI-assisted programming”)
2) Impacto en productividad, habilidades o dependencia	(“programmer productivity” OR “developer efficiency” OR “technological dependence” OR “automation impact”)
3) Tipo de estudio buscado	(“systematic review” OR “literature review” OR “empirical study” OR “case study” OR “experimental study”)

Tabla 3. Criterios de exclusión para la aceptación de estudios.

5.4 Fuentes de información

Fuentes

- Artículos científicos publicados en revistas indexadas en bases de datos como Scopus, ScienceDirect, Google Scholar y arXiv.
- Tesis académicas y documentos de investigación institucional que analicen el impacto de la inteligencia artificial en el desarrollo de software o en las competencias de los programadores.
- Reportes técnicos de organizaciones y consultoras reconocidas, como McKinsey & Company, Stack Overflow, OpenAI y GitHub, que ofrecen datos actualizados sobre el uso de IA en entornos de programación.

6. Resultados

6.1. Estrategia de Búsqueda y Selección Sistemática

La fase inicial y fundamental de toda revisión sistemática consiste en una estrategia de búsqueda rigurosa que asegure la identificación de la totalidad de la evidencia relevante existente. Para cumplir con este principio, el presente estudio se basó en los lineamientos PRISMA, un protocolo reconocido internacionalmente para la estructuración de revisiones sistemáticas. La implementación de este protocolo permitió construir un proceso de selección en fases (identificación, filtrado, selección e inclusión), tal como se visualiza en el Diagrama de Flujo PRISMA (Figura 1). Este diagrama documenta de forma precisa el recorrido desde los resultados de la búsqueda en las bases de datos hasta la conformación del cuerpo final de estudios analizados.

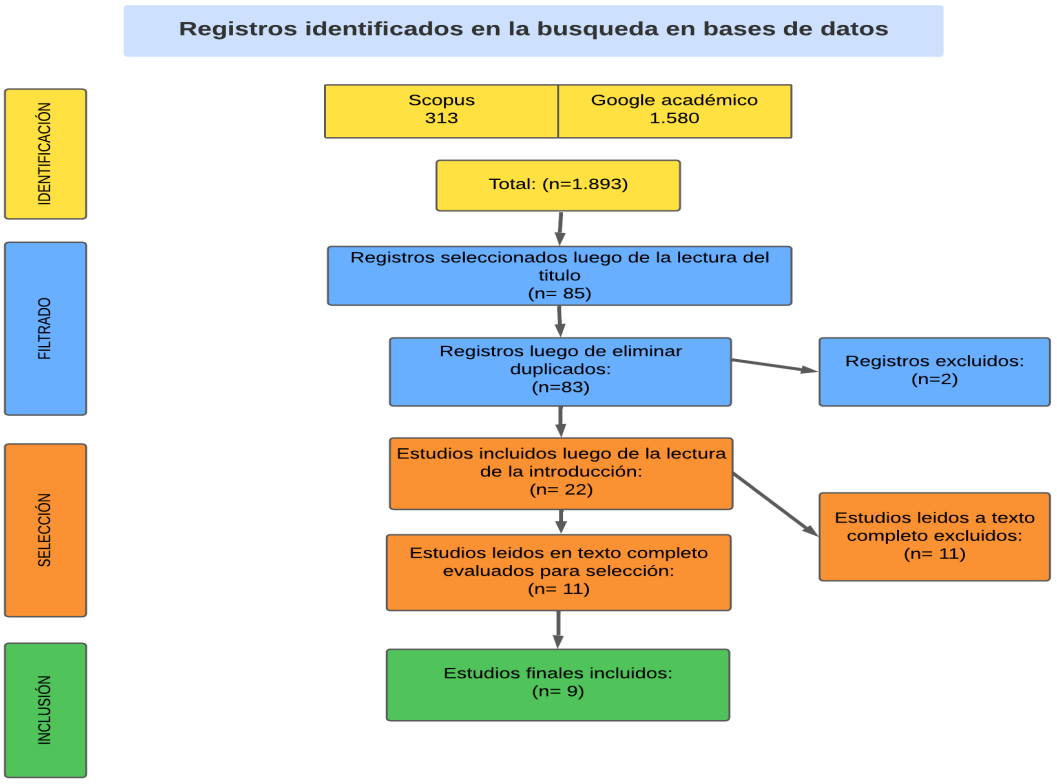

Figura 1. Diagrama de flujo PRISMA.

Estrategia de identificación y filtrado:

El Diagrama PRISMA (Figura 1) no solo describe la cantidad de estudios, sino que demuestra la aplicación rigurosa de unos criterios de inclusión y exclusión mencionados anteriormente, diseñados para garantizar que la muestra final sea la más relevante y de mayor calidad para responder la pregunta de investigación. El análisis de esta selección muestra las decisiones críticas que dieron forma al cuerpo de análisis:

De 1.893 a 85 registros filtrado por título y resumen
La reducción en esta fase (de 1.893 a 85 registros) se debió a la aplicación estricta de criterios de exclusión. Se descartaron estudios que, aunque mencionaban palabras clave como "IA" o "desarrollo de software", se centraban en:

A) Temas ajenos: Investigaciones aplicadas a medicina, finanzas o biología que no aplicaban sus hallazgos al contexto del desarrollo de software.
B) Tipos de publicación no consideradas para selección: Artículos cortos, resúmenes de conferencias sin texto completo, o literatura no revisada por pares.
C) Enfoque técnico no alineado: Estudios que analizaban únicamente la arquitectura de modelos de IA, sin evaluar su impacto en los programadores o en el proceso de desarrollo.

De 85 a 22 estudios evaluación de introducción y metodología
En esta etapa, se realizó una evaluación cualitativa inicial. Se excluyeron 63 estudios principalmente porque:

A) Falta de evidencia: Eran artículos de opinión, propuestas teóricas o marcos conceptuales que, si bien son valiosos, no presentaban datos recogidos de desarrolladores o proyectos reales como por ejemplo: experimentos, encuestas, estudios de caso.

B) Calidad de la fuente: Se priorizaron publicaciones en revistas indexadas y conferencias de alto prestigio, descartando aquellas de procedencia dudosa o sin un proceso de revisión por pares.

De 22 a 11 estudios evaluación a texto completo:

Esta fue la fase de filtrado más exigente. La lectura completa de los 22 artículos permitió aplicar los criterios con mayor precisión, excluyendo 11 estudios porque:

A) No abordaban las dimensiones necesarias del objetivo: Muchos medían únicamente la velocidad de codificación (productividad), pero no exploraban su influencia en las habilidades, la autonomía o los aspectos éticos, que son centrales para esta investigación.

B) Evidencia insuficiente o sesgada: Estudios con un tamaño de muestra muy pequeño, un diseño experimental débil, o un análisis de resultados poco convincente.

C) Redundancia temática: Algunos artículos replicaban hallazgos ya reportados de manera más sólida y comprehensiva por otros estudios en la lista.

Los 9 estudios finales

La selección final de 9 es un proceso de filtrado riguroso que priorizó la calidad sobre la cantidad. Esta muestra representa la combinación perfecta entre:

A) Relevancia temática: Abordan directamente la influencia de herramientas como Copilot y ChatGPT en programadores.

B) Solidez metodológica: Presentan evidencia robusta (experimentos controlados, encuestas a gran escala, estudios de caso en entornos reales).

C) Alineación con los objetivos específicos: Sus hallazgos permiten analizar de manera crítica los efectos positivos, negativos y las dimensiones de productividad, habilidades y autonomía.

6.2. Tabla de características de literatura elegida.

Con el objetivo de resumir y organizar los hallazgos clave de la revisión sistemática de literatura, los estudios primarios seleccionados fueron consolidados en una tabla analítica (Tabla 4). Esta herramienta fue diseñada para permitir una comparación sistemática de las conclusiones centrales de cada investigación.

La tabla presenta para cada estudio un ID único, el título del artículo, sus conclusiones principales y la categoría a la cual fue incluida dependiendo del análisis de cada estudio. La inclusión de este último elemento es fundamental, ya que concentra la esencia del aporte de cada investigación.

ID	Titulo del articulo	Conclusiones
E1	The Impact of Code-Generating AI on the Work of Programmers (Perú, 2025)	Las IA generadoras de código, como GitHub Copilot, aumentan la productividad notablemente, especialmente en desarrolladores novatos, acelerando las tareas hasta en un 55.8%. Sin embargo, su uso conlleva riesgos, como la generación de código inseguro y la posible dependencia que limite el desarrollo de habilidades fundamentales.Por ello, es crucial un enfoque equilibrado que combine estas herramientas con verificaciones de seguridad, supervisión experta y formación continua. Esto asegura que las ganancias en eficiencia no comprometan la calidad del software ni el crecimiento profesional de los programadores.
E2	Generative AI and Developer Workflows: How GitHub Copilot and ChatGPT Influence Solo and Pair Programming (Noruega, 2025)	La integración de IA Generativa, como GitHub Copilot, ha transformado el flujo de trabajo del desarrollador ofreciendo beneficios clave, como un aumento en la productividad y la automatización de tareas rutinarias. Sin embargo, su implementación presenta desafíos, ya que la calidad de sus sugerencias no es infalible y requiere una verificación constante por parte del desarrollador. Esto establece una dinámica de equilibrio esencial: la IA es ideal para tareas repetitivas, pero la experiencia humana sigue siendo indispensable para la resolución de problemas complejos. Por lo tanto, su potencial se maximiza como un complemento bajo supervisión humana, no como un reemplazo. Finalmente, se concluye que es crucial investigar el impacto a largo plazo en las habilidades de los desarrolladores y establecer prácticas éticas para su uso responsable.

E3	Trust and reliance on AI — An experimental study on the extent and costs of overreliance on AI (Alemania, 2024)	Los hallazgos demuestran que, en un contexto general, las personas tienden a depender excesivamente de los consejos de la IA al tomar decisiones financieramente riesgosas y con implicaciones éticas, aun cuando esto conduce frecuentemente a resultados negativos para ellas mismas y para terceros. Sorprendentemente, esta sobre-dependencia es mayor cuando el consejo de la IA contradice la información disponible y el propio criterio de los participantes. El estudio concluye que el mero hecho de saber que un consejo es generado por una IA basta para que las personas confíen en él de más, sin que exista una razón transparente que justifique dicha confianza. Este experimento pionero, al ser independiente de un dominio específico e interactivo, revela un problema crítico en la toma de decisiones asistida por IA: las personas pueden seguir las recomendaciones de la IA incluso sin motivos para confiar en sus capacidades. Estos resultados subrayan la urgente necesidad de desarrollar una "alfabetización en IA" para que los usuarios utilicen la tecnología de forma apropiada y los diseñadores creen sistemas más efectivos.
E4	A Large-Scale Survey on the Usability of AI Programming (Estados Unidos, 2024)	Este estudio investiga la usabilidad de asistentes de programación con IA como GitHub Copilot mediante una encuesta cualitativa a 410 desarrolladores. Los resultados revelan que las principales motivaciones de uso son la autocompletación, la aceleración de tareas y el recordatorio de sintaxis, más que la generación de ideas creativas. Si bien se reconoce su alto rendimiento, se identifica una brecha crítica: las herramientas no satisfacen completamente las necesidades de los desarrolladores, particularmente en la incorporación de requisitos no funcionales en el código generado. El estudio concluye proponiendo mejoras en el diseño de interacciones que otorguen a los desarrolladores mayor control sobre la salida de la IA, y facilite los instrumentos para replicar la investigación.
E5	Harnessing the Potential of Gen-AI Coding Assistants in Public Sector Software Development (Singapur, 2024)	Este estudio concluye que los asistentes de IA como GitHub Copilot incrementan significativamente la productividad en proyectos del sector público, acelerando la codificación entre 21-28% y mejorando la productividad general en un 12%. Un caso documentado muestra hasta un 50% de mejora en equipos pequeños, facilitando servicios más ágiles a la ciudadanía. Aunque el rendimiento es ligeramente inferior al observado en la industria privada (30-100%), debido a requisitos de datos locales y variabilidad en la experiencia de los desarrolladores, el potencial de eficiencia sigue siendo considerable. Para su implementación, se recomienda clasificar el código para usar opciones en la nube como Copilot, o recurrir a alternativas auto-alojadas como Code Llama cuando sea necesario. La institución promueve la adopción estratégica de estas herramientas mediante licencias centralizadas y mejoras en su gobernanza, posicionando al sector público a la vanguardia de la innovación tecnológica mientras garantiza su uso compliant y efectivo.

E6	ChatGPT in Introductory Programming: Counterbalanced Evaluation of Code Quality, Conceptual Learning, and Student Perceptions (Estados Unidos, 2025)	Este estudio cuasi-experimental evaluó el impacto de ChatGPT en un curso introductorio de programación (CS1). Los resultados demuestran que el acceso a la herramienta mejora consistentemente la calidad del código en corrección, legibilidad y resolución de advertencias, además de reducir significativamente los tiempos de finalización de tareas. Sin embargo, sus efectos sobre la comprensión conceptual fueron dispares: mientras en el tema de funciones no mostró beneficios claros, en el tema de estructuras sí se observaron mejoras. Las percepciones estudiantiles fueron mayoritariamente positivas, valorando especialmente su utilidad para generación de código y depuración. Se concluye que ChatGPT fortalece habilidades procedimentales y eficiencia, pero requiere una integración estructurada para favorecer la comprensión conceptual. Futuras investigaciones deberían ampliar la muestra, explorar efectos longitudinales y diseñar estrategias de andamiaje que optimicen tanto las ganancias procedimentales como conceptuales, incorporando además consideraciones éticas para una adopción sostenible en educación en programación.
E7	Which is a better programming assistant? A comparative study between chatgpt and stack overflow (China, 2023)	El estudio comparativo revela que ChatGPT es superior para tareas de algoritmos y uso de librerías, generando código de mayor calidad en menos tiempo. Por el contrario, Stack Overflow demuestra mayor eficacia en depuración, gracias a su capacidad para resolver errores específicos y ofrecer enlaces de referencia. La encuesta post-experimento identificó que la ventaja de ChatGPT radica en su rapidez para generar código e ideas, mientras Stack Overflow depende de la disponibilidad de contenido relevante.
E8	The Effects of GitHub Copilot on Computing Students' Programming Effectiveness, Efficiency, and Processes in Brownfield Coding Tasks (Estados Unidos, 2025)	La integración de IA en programación demuestra una dualidad fundamental: mientras genera ganancias significativas de velocidad en el trabajo con código heredado y transforma los flujos de desarrollo tradicionales hacia un modelo basado en prompts, su implementación revela desafíos críticos. Los programadores más efectivos adoptan un enfoque estratégico, siendo selectivos con las sugerencias y realizando ediciones manuales, mientras que surge una preocupante brecha de comprensión donde los estudiantes utilizan código generado sin entender su lógica o integración.
E9	The Impact of AI on Developer Productivity: Evidence from GitHub Copilot (Estados Unidos, 2023)	Este estudio constituye el primer experimento controlado que mide el impacto de la IA generativa en desarrolladores profesionales, demostrando que GitHub Copilot incrementa la productividad en un 55.8%. Sin embargo, los beneficios varían según la experiencia: los programadores noveles se benefician más que los experimentados. La investigación, centrada en tareas estandarizadas, señala la necesidad de ampliar la investigación a proyectos colaborativos reales y evaluar el impacto en la calidad del código, cuyas implicaciones en seguridad y rendimiento resultan cruciales.

Tabla 4. Análisis de estudios seleccionados.

El grupo de estudios analizados (E1–E9) muestra una tendencia clara: las herramientas de inteligencia artificial como GitHub Copilot y ChatGPT sí mejoran la productividad en el desarrollo de software, pero también generan riesgos importantes que afectan las habilidades y la autonomía de los programadores.

En primer lugar, varios estudios (Christoph Treude & Margaret-Anne Storey, 2025; Kevin KB Ng et al., 2024; Kevin Paul Rivas et al., 2025; Sida Peng et al., 2023) coinciden en que la productividad aumenta de forma significativa, especialmente en tareas repetitivas o de autocompletado. Por ejemplo, (Kevin Paul Rivas et al., 2025; Sida Peng et al., 2023) reportan incrementos superiores al 50% en ciertos escenarios, mientras que (Kevin KB Ng et al., 2024) demuestra mejoras relevantes incluso en el sector público. Sin embargo, estos mismos trabajos dejan claro que la IA no garantiza una mejor comprensión del código, sino una mayor velocidad.

Por otro lado, otros estudios evidencian riesgos relacionados con la dependencia. El estudio (Artur Klingbeil et al., 2024), aunque no es específico de programación, demuestra que las personas pueden confiar demasiado en los sistemas de IA incluso cuando la recomendación es incorrecta. Esto se conecta directamente con los hallazgos de (Md Istiak Hossain Shihab et al., 2025; Shiza Andleeb et al., 2025), donde los estudiantes aceptan código generado sin entenderlo completamente, lo que crea una brecha de comprensión que afecta su autonomía como desarrolladores.

También se observa que la IA no cubre todas las necesidades del desarrollo de software. El estudio (Jenny T. Liang et al., 2024) señala que los asistentes actuales no consideran adecuadamente los requisitos no funcionales, mientras que (Jinrun Liu et al., 2023) menciona que, aunque ChatGPT genera código rápidamente, otras plataformas como Stack Overflow siguen siendo superiores en temas de depuración. Esto reafirma que la IA es útil, pero no reemplaza totalmente el criterio técnico humano.

Finalmente, la revisión muestra que el impacto de la IA depende mucho del contexto: en educación (Md Istiak Hossain Shihab et al., 2025; Shiza Andleeb et al., 2025) el riesgo de dependencia es mayor; en entornos profesionales (Kevin KB Ng et al., 2024; Kevin Paul Rivas et al., 2025; Sida Peng et al., 2023) la productividad aumenta, pero siguen existiendo dudas sobre la calidad del código y la sostenibilidad de estos beneficios a largo plazo.

6.3.Resultados de la Búsqueda Bibliográfica en Scopus

Como primer paso en la identificación de literatura relevante, se ejecutó una búsqueda sistemática en Scopus. La estrategia se construyó mediante una combinación de términos clave relacionados con“GitHub Copilot” OR “ChatGPT” OR “AI code generation” OR “generative AI” OR “AI-assisted programming” OR (“programmer productivity” OR “developer efficiency” OR “technological dependence” OR “automation impact”. El propósito de esta búsqueda fue buscar el volumen y la evolución en el tiempo de publicaciones relacionadas con la colaboración entre programación e inteligencia artificial, permitiendo identificar tendencias y vacíos en la literatura. Las Figuras desde 2 hasta 6 presentan la distribución anual de los documentos encontrados por esta consulta. Estos gráficos ofrecen una primera evidencia cuantitativa del material científico disponible y sirven como punto de partida para el posterior proceso de filtrado y selección, mostrando la base documental sobre la cual se aplicarán los criterios de inclusión y exclusión.

En la figura 2 se observa los resultados de la búsqueda del término *“GitHub Copilot”* sumado a limitaciones en la búsqueda como: “Generative ai”, “Code Generation”, “Artificial Intelligence”, “Ai Assistans” y “Automated Code Generation”, con un rango de tiempo entre 2022 y 2025 . El gráfico demuestra un pico importante de publicaciones entre 2023 y 2025, tiempo en el cual Copilot se convirtió en una de las herramientas de IA más utilizadas en entornos profesionales y educativos en cuanto a la programación. Este comportamiento demuestra que Copilot es uno de los temas con mayor atención científica reciente, lo que explica por qué varios de los estudios seleccionados se enfocan específicamente en evaluar su impacto.

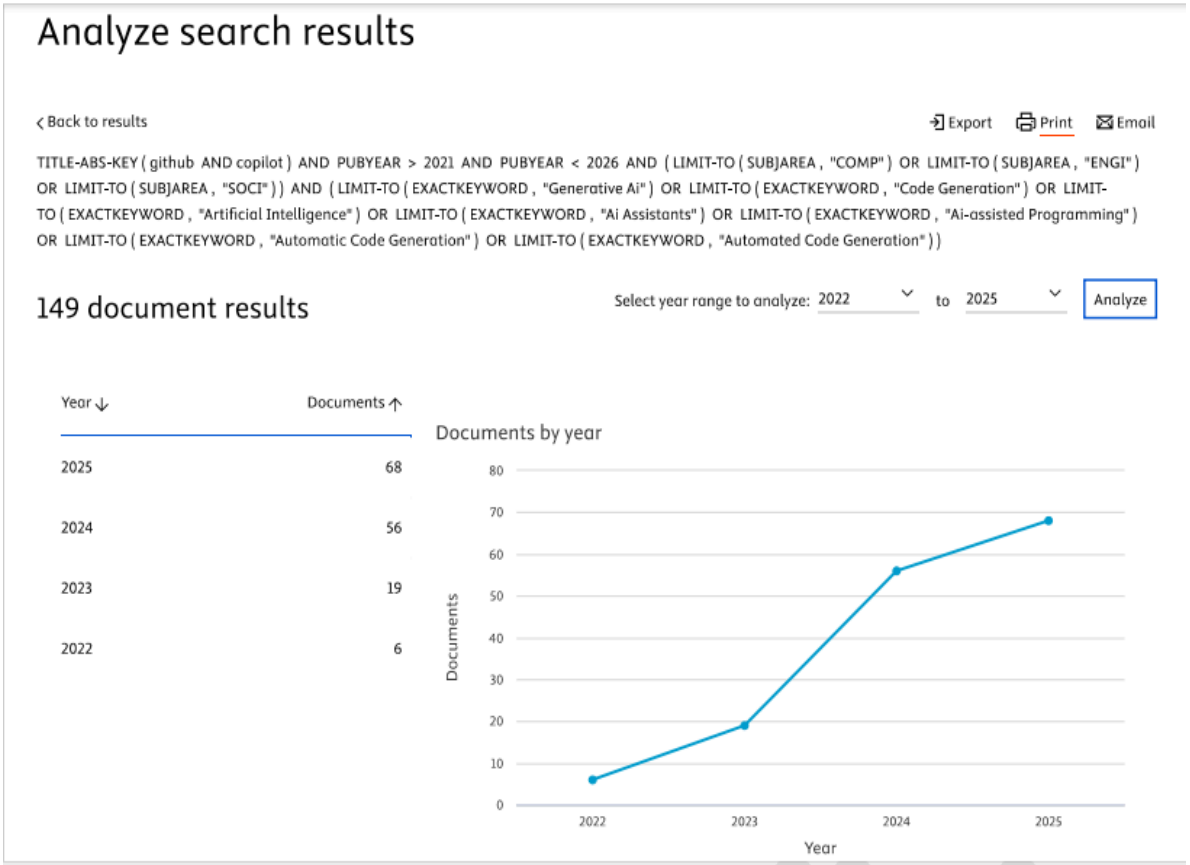

Figura 2. Número de estudios para la búsqueda: 'GitHub Copilot', clasificados por año de publicación.

Fuente: Elaboración por medio de los resultados de búsqueda y filtros aplicados en Scopus.

En la figura 3 se observa los resultados de la búsqueda del término *“ChatGPT”* sumado a limitaciones en la búsqueda como: “Code Generation”, “Artificial Intelligence”, “Ai Assistans” y “Automated Code Generation”, con un rango de tiempo entre 2022 y 2025. El gráfico muestra un crecimiento muy fuerte en 2023 y 2024, seguido de una caída considerable en 2025. Esta disminución puede deberse a un desplazamiento temático hacia otros modelos, o al hecho de que en 2023–2024 se produjo la mayor cantidad de estudios exploratorios, mientras que en 2025 comenzaron investigaciones más especializadas, menos amplias en número pero más profundas en análisis.

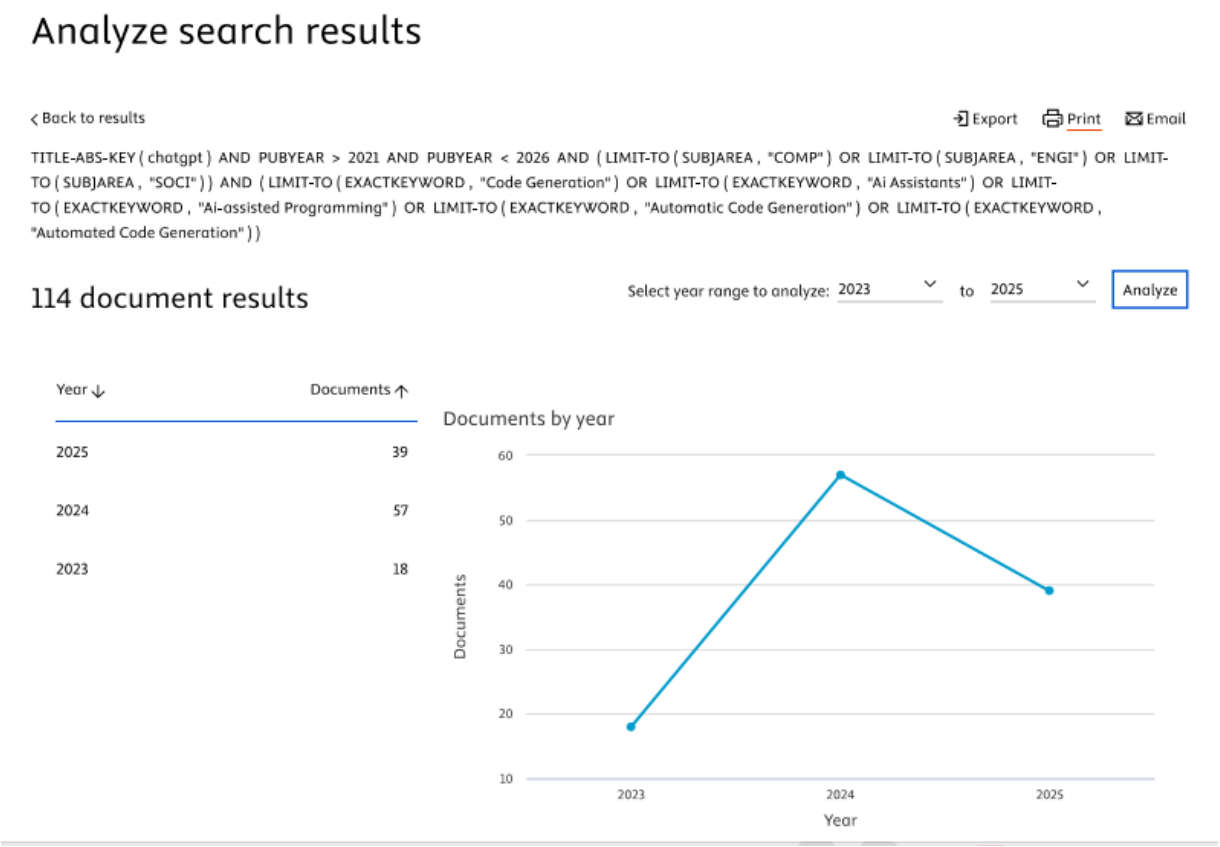

Figura 3. Número de estudios para la búsqueda: “ChatGPT”, clasificados por año de publicación.

Fuente: Elaboración por medio de los resultados de búsqueda y filtros aplicados en Scopus.

En la figura 4 se observa los resultados de la búsqueda del término *“programmer productivity”* sumado a limitaciones en la búsqueda como: “Code Generation”, “Ai Assistans” y “Automated Code Generation”, con un rango de tiempo entre 2022 y 2025. En este gráfico se evidencia cómo el interés académico en los efectos de la IA sobre la productividad ha aumentado de manera significativa entre 2023 y 2024, pero con una disminución muy notable en el 2025, coincidiendo con la adopción masiva de herramientas como GitHub Copilot y ChatGPT. Esta tendencia indica que a pesar de una pequeña cantidad de resultados, la productividad es un enfoque bastante estudiado dentro del campo.

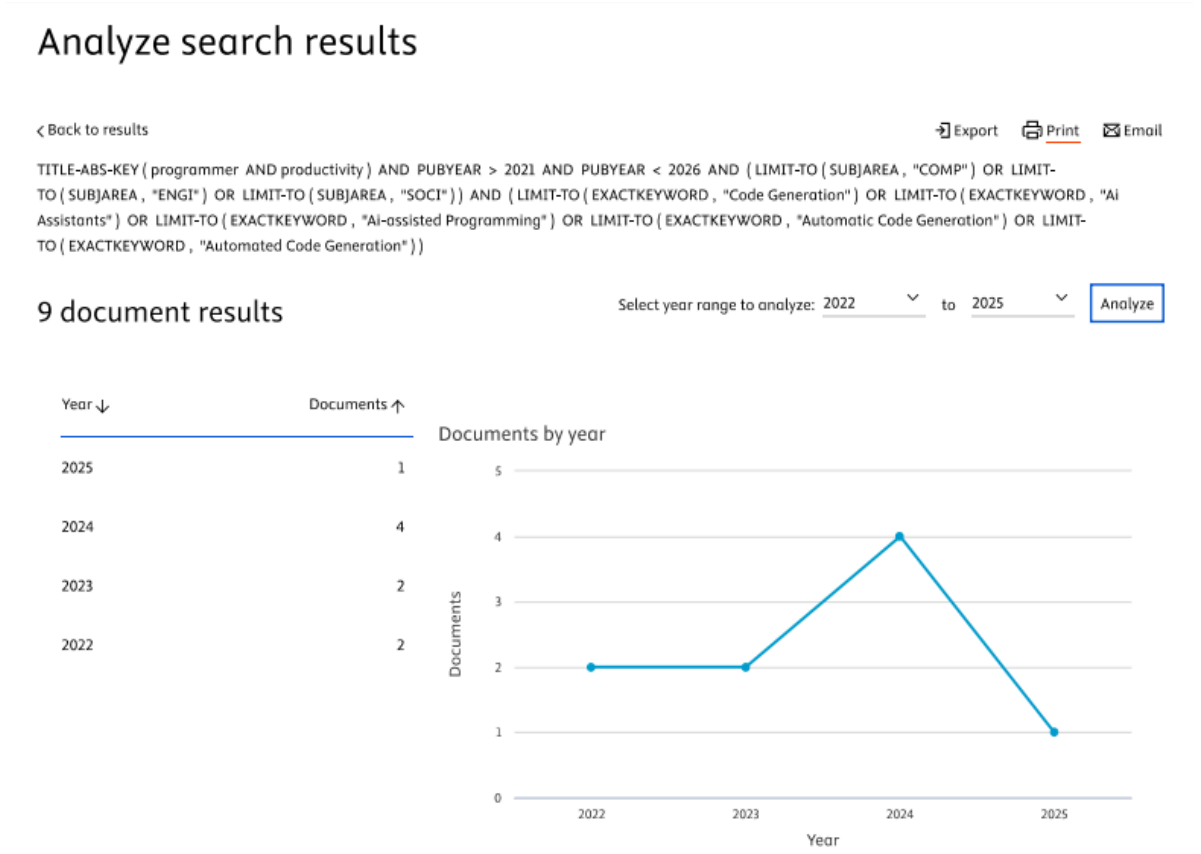

Figura 4. Número de estudios para la búsqueda: 'Programming productivity', clasificados por año de publicación.

Fuente: Elaboración por medio de los resultados de búsqueda y filtros aplicados en Scopus.

En la figura 5 se observa los resultados de la búsqueda del término *“Developer efficiency”* sumado a limitaciones en la búsqueda como: “Code Generation”, “Ai Assistans” y “Automated Code Generation”, con un rango de tiempo entre 2022 y 2025. En este gráfico se evidencia un comportamiento similar al de productividad, con un aumento notable en los estudios entre 2022 y 2024, con una caída en la cantidad de estudios en el 2025. La relación entre eficiencia y herramientas de IA aparece como un tema creciente, especialmente en contextos educativos y de programación básica.

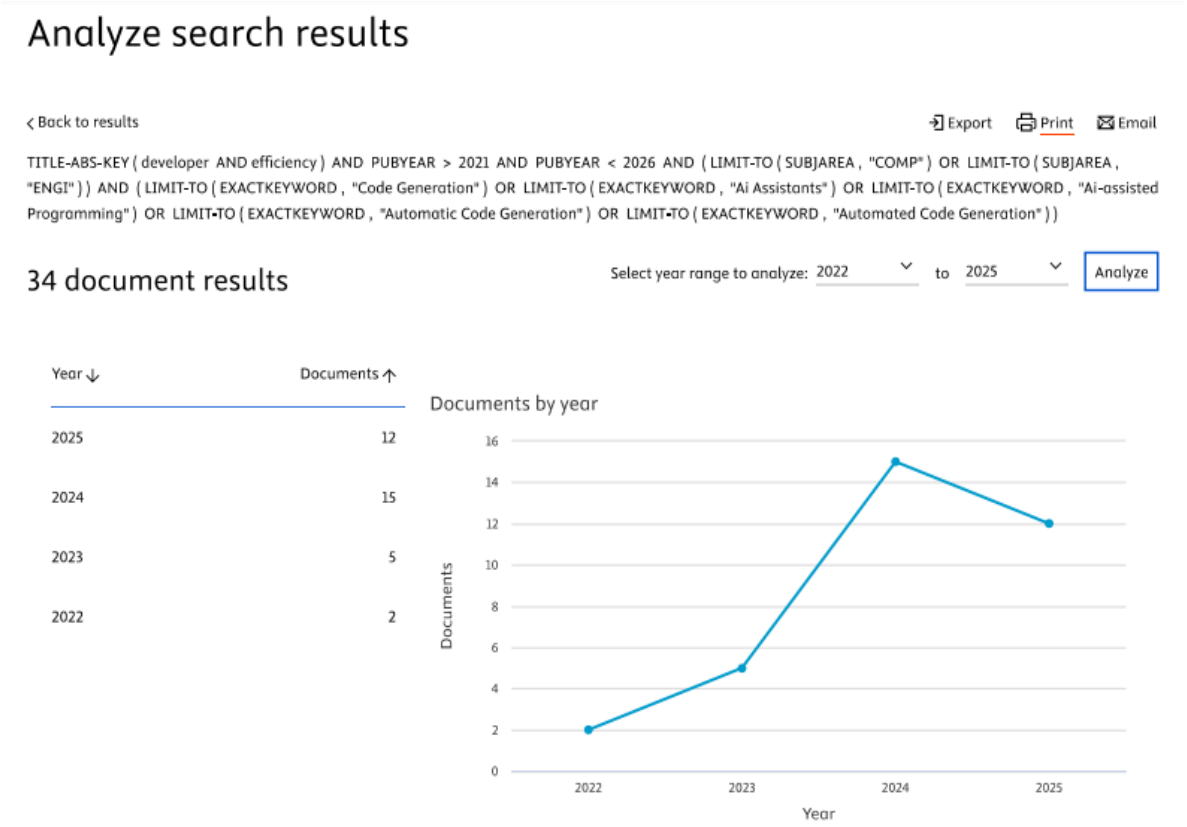

Figura 5. Número de estudios para la búsqueda: “Developer efficiency”, clasificados por año de publicación.

Fuente: Elaboración por medio de los resultados de búsqueda y filtros aplicados en Scopus.

En la figura 6 se observa los resultados de la búsqueda del término *“Automation Impact”* sumado a limitaciones en la búsqueda como: “Code Generation”, “Ai Assistans” y “Automated Code Generation”, con un rango de tiempo entre 2022 y 2025. En esta búsqueda se aprecia que el número de estudios orientados al impacto general de la IA también crece a partir de 2023, pero con un comportamiento menos elevado que en productividad. Este gráfico evidencia que, aunque existe interés por los efectos globales de estas tecnologías, la literatura aún es incipiente y está en consolidación y crecimiento. Esto justifica la necesidad de un análisis crítico como el que se realizó posteriormente en este proyecto.

Figura 6. Número de estudios para la búsqueda: “Automation Impact”, clasificados por año de publicación.

Fuente: Elaboración por medio de los resultados de búsqueda y filtros aplicados en Scopus.

Con base en los gráficos obtenidos desde Scopus se puede observar una línea de tiempo, en la cual se identifica una tendencia clara en la evolución de los estudios relacionados con el uso de herramientas como ChatGPT, GitHub Copilot y la IA generativa en la programación. En los primeros años evaluados, el número de publicaciones es bajo, lo cual es comprensible considerando que las primeras herramientas de IA generativa aplicadas al desarrollo de software empezaron a tener un reconocimiento real solo a partir de 2022.

A partir de allí, entre 2023 y 2024 se evidencia un crecimiento exponencial en las publicaciones. Este aumento coincide con la popularización de ChatGPT, el lanzamiento de versiones más avanzadas de Copilot y el interés académico por medir su impacto en productividad, calidad del código y habilidades de los programadores. Los estudios encontrados sobre productividad, eficiencia y uso de GitHub Copilot refuerzan esta tendencia, ya que la mayoría se concentra en estos dos años (Chi-In Chang et al., 2025).

Sin embargo, es llamativa la caída significativa en 2025, especialmente en temas relacionados directamente con ChatGPT y su influencia en programación, como se observa en los resultados específicos de la búsqueda “ChatGPT”. Esta disminución no necesariamente significa una pérdida de interés, sino más bien un cambio en las líneas de investigación, algo que ya se evidencia en trabajos recientes. Por ejemplo, el estudio (Christoph Treude & Margaret-Anne Storey, 2025) señala que la introducción de la IA generativa ha transformado profundamente la ingeniería de software, desplazando la atención desde herramientas particulares como ChatGPT hacia nuevos marcos conceptuales, relaciones socio-técnicas y cambios estructurales en el proceso de desarrollo. De manera similar, (Kevin Krings et al., 2025) describe la emergencia de paradigmas pos-programación, como el vibe coding, donde el desarrollo ya no se entiende como escribir código asistido por IA, sino como una interacción fluida con agentes inteligentes capaces de generar y coordinar sistemas completos. Esta transición ayuda a explicar por qué en 2025 disminuyen los estudios centrados en “ChatGPT + programación tradicional”: la atención investigativa se desplaza hacia agentes autónomos y nuevos modelos de interacción, lo

cual reduce la presencia de artículos bajo los términos clásicos de búsqueda. A esto se suma que muchos estudios de 2023–2024 fueron exploratorios y se enfocaron en productividad y eficiencia, mientras que las líneas más recientes se orientan hacia aspectos más complejos, como impactos éticos, dependencia cognitiva, calidad del código y transformaciones en los procesos de desarrollo, temas que no siempre utilizan el término “ChatGPT” como descriptor principal.

6.4. Análisis crítico de los Efectos positivos y negativos de las herramientas de IA

Tras la identificación y categorización de la literatura, se procedió a un análisis crítico de la evidencia para buscar los efectos positivos y negativos del uso de herramientas como ChatGPT y GitHub Copilot. Este análisis revela un panorama combinado, donde los beneficios coexisten, y a menudo están ligados, a desafíos y riesgos significativos. A continuación, se presentan estos hallazgos en la Tabla 5, para posteriormente discutir los patrones más relevantes que surgen del contraste de la evidencia.

Categorías de análisis	Efectos positivos	Efectos negativos	Estudios involucrados (ID)	Análisis
Productividad y eficiencia	- Aceleración en tareas repetitivas (55% más rápido, E2). -Generación rápida de código boilerplate (E4, E7).	- Sobrecarga por revisar código sugerido (E1). -Tiempo perdido en producir prompts (E5).	E1, E2, E4, E5, E7	La ganancia en velocidad es clara en tareas simples, pero puede verse contrarrestada por la capacidad técnica en lógica compleja.
Calidad del código	- Menor cantidad de errores de sintaxis (E3). -Mejores prácticas en patrones comunes (E8).	- Multiplicación de vulnerabilidades de seguridad (E6). -Código superficial que no pasa revisión (E1, E5).	E1, E3, E5, E6, E8	La calidad mejora en la forma, pero se afecta en el fondo (seguridad, lógica). La revisión humana es irreemplazable.

Habilidades y aprendizaje	- Curva de aprendizaje acelerada para nuevos lenguajes (E9). -Actúa como tutor para resolver dudas (E3).	- Deterioro de la habilidad para depurar sin ayuda (E5). - Falsa sensación de competencia (E9).	E3, E5, E9	Efecto dual. Es una herramienta pedagógica poderosa si se usa con conciencia, pero un riesgo para la internalización de conocimientos.
Autonomía	-Mayor confianza para abordar tecnologías desconocidas (E3, E8).	- Dependencia situacional en la IA para resolver problemas (E5). -Ansiedad ante la falta de la herramienta (E9).	E3, E5, E8, E9	La autonomía se redefine, no necesariamente se pierde. Se pasa de "saber codificar" a "saber dirigir y corregir a la IA".

Tabla 5. Análisis crítico de los Efectos positivos y negativos de las herramientas de IA

Con base en la tabla anterior surgió el siguiente análisis:

Observación 1: La Paradoja de la Productividad

El análisis evidencia una paradoja en la productividad. Si bien estudios como (Jinrun Liu et al., 2023; Viktoria Stray et al., 2025) reportan aumentos de velocidad superiores al 50%, esta ganancia es condicional. Investigaciones como (Kevin KB Ng et al., 2024; Kevin Paul Rivas et al., 2025) señalan que la productividad neta puede verse afectada por la 'sobrecarga cognitiva' de revisar y corregir sugerencias incorrectas o insuficientes, un efecto que empeora en problemas de alta complejidad.

Observación 2: Calidad Superficial vs. Calidad Profunda

En cuanto a la calidad del código, se identifica una clara división. Existe un acuerdo en que la IA reduce errores de sintaxis y sugiere estructuras estéticamente mejores (Artur Klingbeil et al., 2024; Md Istiak Hossain Shihab et al., 2025). Sin embargo, surge una preocupación sobre la 'calidad profunda': estudios como (Shiza Andleeb et al., 2025) alertan sobre la

generación de código con vulnerabilidades de seguridad sutilmente integradas, lo que implica que la herramienta no sustituye el criterio experto y la revisión de seguridad.

Observación 3: El Aprendizaje amplificado y el aprendizaje defectuoso

El impacto en las habilidades de los programadores es quizás el más confuso. Por un lado, la literatura sugiere un 'efecto de amplificación' para el aprendizaje, donde los desarrolladores novatos pueden acelerar su comprensión (Artur Klingbeil et al., 2024; Sida Peng et al., 2023). Por otro lado, se observa un 'efecto de atrofia' en habilidades fundamentales como la depuración y la resolución profunda de problemas (Kevin KB Ng et al., 2024), planteando un riesgo a largo plazo para la formación de expertos autónomos.

6.5. Categorización de la Literatura Seleccionada

Con el fin de estructurar y analizar los hallazgos de la revisión sistemática de literatura, los nueve estudios seleccionados fueron clasificados en categorías temáticas. Esta clasificación, surge del análisis del contenido y las conclusiones de cada artículo, permite agrupar la investigación en torno a ejes centrales. Las categorías establecidas son: Interacción humano-IA en desarrollo de software; Calidad del código; Aprendizaje y formación con IA; Automatización y generación de código; Productividad y eficiencia; y Dependencia y autonomía del programador. La Figura 7 presenta, mediante un gráfico de barras, la distribución de los estudios en estas categorías, ofreciendo una visión cuantitativa inicial del panorama investigativo.

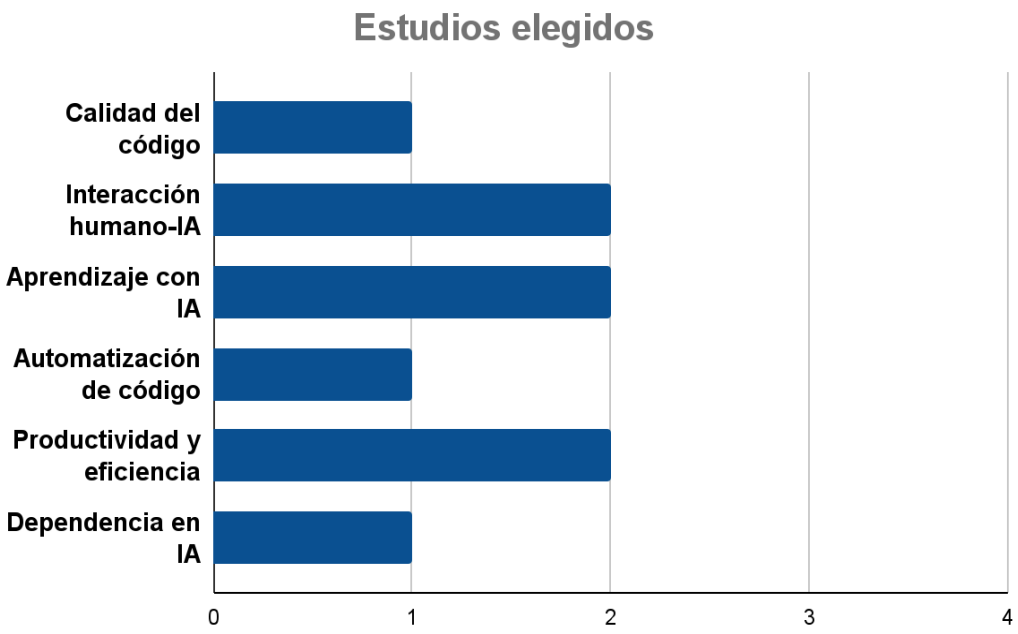

Figura 7. Gráfico de barras de estudios elegidos según categorías.

La Figura 7 presenta la categorización temática de los nueve estudios seleccionados para la revisión, los cuales fueron escogidos de manera deliberada por su pertinencia para responder la pregunta de investigación planteada. En este sentido, la distribución observada en la gráfica no refleja la disponibilidad general de literatura en cada categoría, sino la relevancia de los estudios para abordar las dimensiones clave de dicha pregunta.

La gráfica muestra que las categorías de Interacción humano–IA, Aprendizaje con IA y Productividad y eficiencia concentran dos estudios cada una. Esta predominancia se explica porque estos temas constituyen los ejes centrales necesarios para analizar cómo los programadores colaboran con herramientas de IA, cómo estas influyen en sus procesos de aprendizaje y en qué medida impactan los niveles de rendimiento y optimización del trabajo. Estos estudios aportan la base empírica y conceptual para examinar el potencial de la IA generativa en la práctica profesional del desarrollo de software.

Por otra parte, las categorías de Calidad del código, Automatización del código y Dependencia en IA, cada una representada por un único estudio, cumplen un papel complementario dentro del análisis. Aunque su cantidad es menor, estos temas permiten explorar aspectos críticos directamente vinculados con los riesgos y tensiones asociados al uso de IA: desde la fiabilidad del código generado, hasta los posibles efectos en la autonomía del programador y la aparición de comportamientos de dependencia. Estas áreas, aunque menos numerosas en la selección, resultan esenciales para comprender la otra dimensión de la pregunta de investigación: los posibles impactos negativos o no deseados.

6.6. Tabla de clasificación según las dimensiones analizadas

Teniendo en cuenta las dimensiones de productividad, eficiencia, autonomía y habilidades técnicas, se elaboró la siguiente tabla de clasificación. En ella se organizan los estudios seleccionados (E1–E9) según los principales aspectos evaluados en cada investigación, lo que permite identificar de manera estructurada los enfoques, aportes y limitaciones presentes en la literatura analizada.

Estudio	Productividad	Eficiencia	Autonomía	Habilidades
E1	Mejora en productividad de más de 55.8%	Acelera tareas repetitivas	No evalúa riesgo de dependencia	Habilidades superficiales y riesgos en fundamentos
E2	Aumento en la velocidad de codificación	Flujo de trabajo más automatizado	Requiere supervisión constante	No aplica
E3	No evalúa la productividad	No evalúa eficiencia	Alto riesgo de sobre confianza	No aplica
E4	Percibe una productividad positiva	Autocompletado eficiente pero parcial	Autonomía limitada por falta de control sobre la IA	No aplica

E5	Mejora en la proactividad de 12-50% en ciertos casos	Agiliza servicios cotidianos	Autonomía estable, pero con regulación	No aplica
E6	Mejora tiempos de entrega	Código más limpio y entendible	Autonomía parcial	Mejora en los procedimientos de programación
E7	ChatGPT es eficiente para generar código	Mayor eficiencia en algoritmos	Autonomía equilibrada	Mejora en búsqueda de soluciones
E8	Ganancias de velocidad en código generado	Enfoque estratégico basado en los prompts	Autonomía reducida (comprensión reducida)	Mejora la habilidad procedimental
E9	+55.8% en productividad	Mayor velocidad según la experiencia	Dependencia mayor en novatos	No evalúa habilidades directamente

Tabla 6. Clasificación de estudios según dimensiones

Al clasificar los estudios según las categorías mencionadas, se evidencia una tendencia clara: la mayoría de investigaciones reportan mejoras en productividad y eficiencia gracias al uso de herramientas como ChatGPT y GitHub Copilot, especialmente en tareas repetitivas o de autocompletado (Kevin KB Ng et al., 2024; Kevin Paul Rivas et al., 2025; Md Istiak Hossain Shihab et al., 2025; Shiza Andleeb et al., 2025; Sida Peng et al., 2023; Viktoria Stray et al., 2025). Esto confirma que la IA generativa tiene un impacto positivo en la velocidad con la que se desarrollan ciertas actividades de programación.

Sin embargo, al analizar la dimensión de autonomía, los resultados muestran una realidad diferente. Varios estudios destacan riesgos de dependencia, sobreconfianza o necesidad de supervisión constante (Artur Klingbeil et al., 2024; Kevin Paul Rivas et al., 2025; Md Istiak Hossain Shihab et al., 2025), lo que indica que el uso de IA puede reducir la toma de decisiones independientes, sobre todo en desarrolladores con menor experiencia. Solo algunos trabajos evidencian un uso más equilibrado y controlado (Jinrun Liu et al., 2023;

Kevin KB Ng et al., 2024), lo que sugiere que el contexto profesional influye en la forma en que la IA afecta la autonomía.

En cuanto a las habilidades, los estudios coinciden en que la IA fortalece principalmente las competencias procedimentales, como escribir código más rápido o depurar de forma eficiente (Md Istiak Hossain Shihab et al., 2025; Shiza Andleeb et al., 2025). No obstante, también alertan sobre posibles dificultades en el desarrollo de habilidades conceptuales más profundas, como la comprensión lógica o el análisis estructural del código. Esto refuerza la idea de que la IA puede mejorar el rendimiento operativo, pero no necesariamente la formación técnica integral.

En términos más sencillos, la clasificación muestra que los beneficios de la IA se concentran en la productividad y eficiencia, mientras que los desafíos se relacionan con la autonomía y la calidad del aprendizaje. Esta tabla facilita visualizar cómo cada estudio aporta evidencia a dimensiones específicas y permite comprender de manera más clara el impacto general de estas herramientas en el ámbito del desarrollo de software.

6.7. Propuesta de recomendaciones para un uso equilibrado

Partiendo del análisis crítico de la evidencia, que revela una influencia dividida de la IA con beneficios evidentes pero riesgos significativos, en esta sección se propone un conjunto de recomendaciones concretas. Estas sugerencias están diseñadas para maximizar los efectos positivos en la productividad y el aprendizaje, mientras se minimizan los riesgos asociados a la seguridad, la dependencia y la disminución de habilidades. En las tablas 7 a 11, se presentan estos marcos de recomendaciones, trazando de manera explícita la conexión entre cada propuesta y los hallazgos de la literatura que la justifican.

Calidad y ética		
Recomendación	**Evidencia**	**Acción propuesta**
Implementar revisiones de seguridad de código generado por IA	- Generación de código inseguro (E1). - Propagación de vulnerabilidades (E6). - Preocupación por implicaciones en seguridad (E9).	Los hallazgos muestran que la IA no es confiable en seguridad. Se propone una etapa obligatoria en la revisión de código donde un desarrollador experimentado verifique específicamente vulnerabilidades en el código sugerido por la IA.

Tabla 7. Recomendaciones basadas en la evidencia para la calidad y ética.

Habilidades y autonomía		
Recomendación	**Evidencia**	**Acción propuesta**
Diseñar programas de enseñanza en IA para desarrolladores	- Sobre-dependencia incluso contra el propio criterio (E3). - Brecha de comprensión: uso de código sin entenderlo (E8). - Riesgo de limitar habilidades fundamentales (E1).	La evidencia indica una confianza mal utilizada. Un programa de enseñanza ayudaría a evaluar críticamente las sugerencias, entender sus limitaciones y mantener la autoridad cognitiva.

Tabla 8. Recomendaciones basadas en la evidencia para las habilidades y autonomía.

Formación y aprendizaje		
Recomendación	**Evidencia**	**Acción propuesta**
Integrar la IA como "Tutor" en programas de formación, como una herramienta	- Efectos diferentes en la comprensión conceptual (E6). - Acelera la curva de aprendizaje en novatos (E1, E9). - Útil para generación y depuración, pero se necesita integración estructurada (E6).	Para mejorar el beneficio formativo y minimizar la afectación, la IA debe usarse como una herramienta: guías que obliguen al estudiante a explicar el código generado y resolver partes clave por sí mismo.

Tabla 9. Recomendaciones basadas en la evidencia para la formación y aprendizaje.

Productividad		
Recomendación	**Evidencia**	**Acción propuesta**
Establecer políticas de "Uso Apropiado" que definan tareas realizables y de riesgo.	- La IA es ideal para tareas repetitivas, pero la experiencia humana es indispensable para problemas complejos (E2). - Ganancias de productividad en tareas estandarizadas (E9).	Crear una lista interna que especifique en qué tipos de tareas (ej: código estandarizado, documentación) se fomenta el uso de la IA y en cuáles (ej: lógica de negocio, algoritmos complejos) se requiere supervisión estricta.

Tabla 10. Recomendaciones basadas en la evidencia para la productividad.

Autonomía		
Recomendación	**Evidencia**	**Acción propuesta**
Promover un modelo de "Copiloto Humano-IA" con supervisión.	- Los programadores efectivos son selectivos y realizan ediciones manuales (E8). - La calidad de las sugerencias requiere verificación constante (E2).	Se recomienda formalizar el rol del desarrollador como un "director técnico" de la IA, donde su valor no está en escribir cada línea, sino en dirigir, refinar y validar la salida del código.

Tabla 11. Recomendaciones basadas en la evidencia para la autonomía.

Las tablas 6 a 10 presentan un conjunto de recomendaciones como respuesta directa de los hallazgos identificados en los estudios seleccionados. Estas propuestas no son sugerencias aleatorias, sino el resultado de un proceso de síntesis que buscó traducir la evidencia empírica en sugerencias prácticas para un uso equilibrado de herramientas de IA en el desarrollo de software. En conclusión, las recomendaciones responden a la pregunta central de la investigación, articulando cómo maximizar los beneficios de productividad y aprendizaje sin comprometer la autonomía, la seguridad y las habilidades fundamentales del programador.

En primer lugar, las recomendaciones relacionadas con calidad y ética (Tabla 6) surgen debido a la evidencia consistente que muestra que la IA puede generar código con vulnerabilidades, errores lógicos y fallos de seguridad. Estudios como (Kevin Paul Rivas et al., 2025; Shiza Andleeb et al., 2025; Sida Peng et al., 2023) muestran que los modelos no son confiables para garantizar robustez en tareas sensibles. A partir de estos hallazgos se fundamenta la necesidad de asegurar que el código generado cumpla con estándares profesionales antes de integrarse en un sistema real. De esta manera, la recomendación se construye directamente a partir de la identificación del riesgo y ofrece un mecanismo para contrarrestarlo.

En cuanto a habilidades y autonomía (Tabla 7), la evidencia refleja problemas vinculados a la sobredependencia, la aceptación acrítica de sugerencias y la pérdida progresiva de comprensión técnica. Hallazgos como los de (Artur Klingbeil et al., 2024; Kevin Paul Rivas et al., 2025; Md Istiak Hossain Shihab et al., 2025) muestran que algunos programadores utilizan la IA incluso cuando perciben que la respuesta podría no ser correcta, mientras que otros utilizan código sin entenderlo completamente. Por ello, la recomendación de implementar programas de formación en el uso crítico de IA busca contrarrestar estas tendencias, promoviendo que los desarrolladores mantengan su autonomía cognitiva y sus competencias más importantes.

Las recomendaciones orientadas a formación y aprendizaje (Tabla 8) se elaboran con base en estudios que muestran que la IA puede facilitar la comprensión y acelerar el aprendizaje, especialmente en principiantes (Kevin Paul Rivas et al., 2025; Shiza Andleeb et al., 2025; Sida Peng et al., 2023), pero también impactar de manera diferente los procesos conceptuales. Por ello, integrar la IA como “tutor” y no como generador de soluciones completas, refleja un equilibrio entre aprovechar su potencial pedagógico y evitar que reemplace los procesos cognitivos clave del estudiante.

Las recomendaciones centradas en productividad (Tabla 9) surgen de la evidencia que muestra que la IA mejora el rendimiento en tareas repetitivas y estandarizadas (Sida Peng et al., 2023; Viktoria Stray et al., 2025), pero no sustituye la experiencia humana en problemas complejos o decisiones críticas. Esto justifica la acción de establecer políticas internas de “Uso Apropiado”, que delimiten qué tipos de tareas pueden delegarse con seguridad a la IA y cuáles necesitan supervisión experta.

Finalmente, las recomendaciones sobre autonomía (Tabla 10) responden a la evidencia de que los desarrolladores más efectivos no aceptan el código generado de manera automática, sino que ejercen un rol activo de supervisión, edición y validación (Sida Peng et al., 2023; Viktoria Stray et al., 2025). De allí surge la propuesta de un modelo de “copiloto humano–IA”, donde el desarrollador actúa como responsable del proceso y mantiene un rol de dirección técnica.

7. Recomendaciones

A partir de los resultados obtenidos en los estudios E1–E9, se proponen varias recomendaciones orientadas a mejorar el uso de herramientas de IA en el desarrollo de software y evitar los riesgos. En primer lugar, es de mucha importancia implementar procesos de capacitación obligatoria en formación en IA, tanto en instituciones educativas como en empresas. Estudios como (Artur Klingbeil et al., 2024; Md Istiak Hossain Shihab et al., 2025; Shiza Andleeb et al., 2025) demuestran que los usuarios tienden a confiar en exceso en las respuestas de la IA, incluso cuando son incorrectas, lo que puede generar dependencia y afectar la comprensión real del código. Por ello, es importante enseñar a los desarrolladores a evaluar, cuestionar y verificar lo que la IA produce.

En segundo lugar, se recomienda utilizar la IA únicamente como apoyo en tareas repetitivas u operativas, evitando confiar en estos sistemas decisiones críticas de diseño o arquitectura. Los estudios (Jenny T. Liang et al., 2024; Jinrun Liu et al., 2023; Viktoria Stray et al., 2025) muestran que herramientas como ChatGPT y GitHub Copilot funcionan muy bien para acelerar actividades mecánicas, pero no sustituyen el criterio humano en tareas complejas. Esto significa que su uso debe ser estratégico y orientado a complementar la labor del programador, no a reemplazarla.

También es necesario que las instituciones educativas integren estrategias de enseñanza que combinen IA con ejercicios de razonamiento y análisis, tal como sugieren (Md Istiak Hossain Shihab et al., 2025; Shiza Andleeb et al., 2025). Aunque la IA facilita la generación de código, los estudiantes pueden llegar a usarlo sin entenderlo, lo que genera una brecha de comprensión. Por ello, se recomienda emplear actividades donde el estudiante deba explicar, corregir o justificar el código generado por IA, fortaleciendo así sus habilidades conceptuales.

Debe generarse una cultura donde la IA sea vista como una herramienta de apoyo, no como un sustituto del programador. Estudios como (Jinrun Liu et al., 2023; Kevin Paul Rivas et al., 2025; Viktoria Stray et al., 2025) coinciden en que la supervisión humana sigue siendo indispensable para garantizar calidad, seguridad y toma de decisiones adecuadas dentro del desarrollo de software. Esto requiere implementar modelos híbridos de trabajo donde la IA acelere procesos, pero las decisiones de gran importancia continúen bajo el control humano.

Se propone reforzar la capacitación en verificación y revisión del código asistido por IA. Según (Md Istiak Hossain Shihab et al., 2025), muchos usuarios aceptan el código generado sin entender su lógica, lo que aumenta riesgos en calidad, seguridad y mantenibilidad. Capacitar a los desarrolladores en técnicas de verificación, depuración y validación permite evitar esta brecha y garantiza que la IA se utilice de forma responsable y sostenible.

Como trabajo futuro se requiere investigar con mayor precisión el fenómeno de la dependencia y la sobreconfianza en herramientas como ChatGPT y GitHub Copilot. Estudios como (Artur Klingbeil et al., 2024; Kevin Paul Rivas et al., 2025; Md Istiak Hossain Shihab et al., 2025) demuestran que los usuarios tienden a confiar demasiado en las recomendaciones de la IA, incluso cuando no las comprenden completamente, pero ninguno presenta métricas estandarizadas que permitan medir este riesgo en escenarios reales de programación. En investigaciones futuras sería fundamental desarrollar instrumentos cuantitativos más estrictos que permitan evaluar la dependencia cognitiva, la pérdida de autonomía y la capacidad de razonamiento independiente.

8. Conclusiones

A partir del análisis realizado sobre los estudios E1–E9, se concluye que el uso de herramientas de inteligencia artificial en el desarrollo de software genera un impacto doble que responde a la pregunta central del proyecto. La evidencia muestra que sí existe una mejora considerable en productividad y eficiencia, especialmente en tareas repetitivas, estructuradas o de bajo nivel de complejidad. Estudios como (Kevin KB Ng et al., 2024; Kevin Paul Rivas et al., 2025; Md Istiak Hossain Shihab et al., 2025; Shiza Andleeb et al., 2025; Sida Peng et al., 2023; Viktoria Stray et al., 2025) demuestran que herramientas como GitHub Copilot y ChatGPT permiten reducir tiempos de desarrollo, incrementar la velocidad de trabajo y facilitar la escritura de código más limpio o más rápido, lo que demuestra un beneficio real para los programadores, tanto en entornos educativos como profesionales.

Sin embargo, estos mismos estudios revelan que este aumento en productividad no siempre se acompaña de un fortalecimiento en habilidades y autonomía profesional. Por el contrario, investigaciones como (Artur Klingbeil et al., 2024; Md Istiak Hossain Shihab et al., 2025; Shiza Andleeb et al., 2025) muestran riesgos de dependencia, sobreconfianza y brechas de comprensión, especialmente entre programadores novatos o estudiantes que tienden a aceptar código generado sin evaluarlo críticamente. Estos hallazgos indican que, aunque la IA acelera el trabajo, también puede limitar el desarrollo de habilidades conceptuales profundas y disminuir la toma de decisiones autónomas si no se usa de manera adecuada.

En conclusión, la evidencia permite decir que las herramientas de IA actualmente funcionan como asistentes altamente efectivos, pero no como reemplazos del razonamiento humano. El impacto positivo depende directamente de cómo se integren en los procesos de aprendizaje y trabajo: cuando se utilizan como apoyo supervisado, la productividad mejora sin comprometer la autonomía; pero cuando se adoptan sin criterios claros, pueden fomentar una dependencia que afecta las habilidades esenciales de programación.

Por ende, el resultado de la investigación indica que la IA sí mejora la productividad y eficiencia de los programadores, pero este efecto positivo puede venir acompañado de riesgos que deben gestionarse por medio de formación adecuada, verificación humana y prácticas de uso responsable. El verdadero reto no está en aceptar o rechazar la IA, sino en aprender a utilizarla de forma equilibrada para mejorar el rendimiento sin sacrificar la autonomía profesional ni el desarrollo de habilidades fundamentales.

9. Bibliografía

Agnia Sergeyuk, Ilya Zakharov, Ekaterina Koshchenko, & Maliheh Izadi. (2025). *Human-AI Experience in Integrated Development Environments: A Systematic Literature Review*. https://arxiv.org/abs/2503.06195

Artur Klingbeil, Cassandra Grützner, & Philipp Schreck. (2024). *Trust and reliance on AI — An experimental study on the extent and costs of*. https://www.sciencedirect.com/science/article/pii/S0747563224002206

Brooke N. Macnamara, Ibrahim Berber, M. Cenk Çavuşoğlu, Elizabeth A. Krupinski, Naren Nallapareddy, Noelle E. Nelson, Philip J. Smith, Amy L. Wilson-Delfosse, & Soumya Ray. (2024). *Does using artificial intelligence assistance accelerate skill decay and hinder skill development without performers' awareness?* https://link.springer.com/article/10.1186/s41235-024-00572-8

Chi-In Chang, Wan-Chong Choi, Iek-Chong Choi, & Huey Lei. (2025). *A Systematic Literature Review of the Practical Applications of Artificial Intelligence-Generated Content (AIGC) Using OpenAI ChatGPT, Copilot, and Codex in Programming Education*. https://dl.acm.org/doi/full/10.1145/3719487.3719519

Christoph Treude, & Margaret-Anne Storey. (2025). *Generative AI and Empirical Software Engineering: A Paradigm Shift*. https://arxiv.org/abs/2502.08108

Feiming Li, Xinyu Yan, Hongli Su, Rong Shen, & Gang Mao. (2025). *An Assessment of Human–AI Interaction Capability in the Generative AI Era: The Influence of Critical Thinking*. https://www.mdpi.com/2079-3200/13/6/62

Iftekhar Ahmed, Aldeida Aleti, Haipeng Cai, Alexander Chatzigeorgiou, Pinjia He, Xing Hu, Mauro Pezzè, Denys Poshyvanyk, & Xin Xia. (2025). *Artificial Intelligence for Software Engineering: The Journey So Far and the Road Ahead*. https://dl.acm.org/doi/10.1145/3719006

Jenny T. Liang, Chenyang Yang, & Brad A. Myers. (2024). *A Large-Scale Survey on the Usability of AI Programming*. https://dl.acm.org/doi/abs/10.1145/3597503.3608128

Jinrun Liu, Xinyu Tang, Linlin Li, Panpan Chen, & Yepang Liu. (2023). *Which is a better programming assistant? A comparative study between chatgpt and stack overflow*. https://arxiv.org/abs/2308.13851

Kevin KB Ng, Liyana Fauzi, Leon Leow, & Jaren Ng. (2024). *Harnessing the Potential of Gen-AI Coding Assistants in Public Sector Software Development*. https://arxiv.org/abs/2409.17434

Kevin Krings, Nino S. Bohn, & Thomas Ludwig. (2025). *(R)evolution of Programming: Vibe Coding as a Post-Coding Paradigm*. https://arxiv.org/abs/2510.12364

Kevin Paul Rivas, Elmo Francisco Tirado, & Marcelino Torres Villanueva. (2025). *The Impact of Code-Generating AI on the Work of Programmers*. https://revistas.ulasalle.edu.pe/innosoft/article/view/228

Lucas J. Wiese, Indira Patil, Daniel S. Schiff, & Alejandra J. Magana. (2025). *AI ethics education: A systematic literature review*. https://www.sciencedirect.com/science/article/pii/S2666920X25000451

McKinsey. (2023). *El estado de la IA en 2023*. https://www.mckinsey.com/featured-insights/destacados/el-estado-de-la-ia-en-2023-el-ano-clave-de-la-ia-generativa/es#/

Md Istiak Hossain Shihab, Christopher Hundhausen, Ahsun Tariq, Summit Haque, Yunhan Qiao, & Brian Wise Mulanda. (2025). *The Effects of GitHub Copilot on Computing Students' Programming Effectiveness, Efficiency, and Processes in Brownfield Coding Tasks*. https://dl.acm.org/doi/full/10.1145/3702652.3744219

Onur Ceran. (2025). *Security Evaluation of AI-Generated Code: A Comparative Study of ChatGPT, Copilot, And Gemini through Static and Dynamic Analysis*. https://dergipark.org.tr/en/pub/gmbd/issue/94345/1720932

Satyam Kumar Navneet, & Joydeep Chandra. (2025). *Rethinking Autonomy: Preventing Failures in AI-Driven Software Engineering*. https://arxiv.org/abs/2508.11824

Shiza Andleeb, Brandon Kantorski, & Jeffrey Carver. (2025). *ChatGPT in Introductory Programming: Counterbalanced Evaluation of Code Quality, Conceptual Learning, and Student Perceptions*. https://arxiv.org/abs/2510.00946

Sida Peng, Eirini Kalliamvakou, Peter Cihon, & Mert Demirer. (2023). *The Impact of AI on Developer Productivity: Evidence from GitHub Copilot*. https://arxiv.org/abs/2302.06590

Thomas K.F. Chiu. (2024). *Future research recommendations for transforming higher education with generative AI*. https://www.sciencedirect.com/science/article/pii/S2666920X23000760

Viktoria Stray, Nils Brede Moe, Nivethika Ganeshan, & Simon Kobbenes. (2025). *Generative AI and Developer Workflows: How GitHub Copilot and ChatGPT Influence Solo and Pair Programming*. https://scholar.google.es/scholar?hl=es&as_sdt=0%2C5&q=Generative+AI+and+Developer+Workflows%3A+How+GitHub+Copilot+and+ChatGPT+Influence+Solo+and+Pair+Programming&btnG=

Zhanxin Hao, Jianxiao Jiang, Jifan Yu, Zhiyuan Liu, & Yu Zhang. (2025). *Student-AI Interaction in an LLM-Empowered Learning Environment: A Cluster Analysis of Engagement Profiles*. https://arxiv.org/abs/2503.01694