Journal **TechInnovation**
Volumen 4, Número 1, 2025
Universidad Estatal del Sur de Manabí
ISSN-e: 2953-6472

ARTÍCULO ORIGINAL

Aplicaciones de la Inteligencia Artificial en el campo de la programación

Applications of Artificial Intelligence in the field of programming

https://doi.org/10.47230/Journal.TechInnovation.v4.n1.2025.4-12

Recibido: 11-01-2025 **Aceptado:** 10-02-2025 **Publicado:** 01-05-2025

Willy David Cedeño Pincay[1]*
https://orcid.org/0009-0006-2339-2926

1. Investigador Independiente; Jipijapa, Ecuador.

Volumen: 4
Número: 1
Año: 2025
Paginación: 4-12
URL: https://revistas.unesum.edu.ec/JTI/index.php/JTI/article/view/91
***Correspondencia autor:** willycede.ucsg@gmail.com

RESUMEN

El estudio abordó la relevancia de las aplicaciones actuales y emergentes de la inteligencia artificial (IA) en el campo de la programación, destacando su impacto en la automatización del desarrollo de software, la optimización del código, la depuración y los procesos de prueba. El objetivo principal fue analizar cómo estas herramientas han transformado las prácticas tradicionales de programación y qué implicaciones éticas surgen de su implementación en entornos reales de desarrollo. Para cumplir con este propósito, se empleó una metodología de tipo documental-descriptiva con enfoque cualitativo. Se realizó una revisión de literatura científica actualizada, un análisis comparativo de herramientas de IA aplicadas al desarrollo de software y una sistematización de los desafíos técnicos y éticos que estas tecnologías implican. Como resultado, se evidenció que la IA ha permitido automatizar tareas repetitivas, generar código eficiente, detectar errores con mayor precisión y reducir los tiempos de prueba, aumentando así la productividad y calidad del software. Sin embargo, también se identificaron riesgos asociados a la opacidad algorítmica, la responsabilidad compartida en caso de fallos y la necesidad de regulación ética. Se concluyó que las herramientas de inteligencia artificial constituyen una innovación disruptiva en la programación moderna. No obstante, su adopción debe estar acompañada de una supervisión humana crítica, criterios éticos sólidos y formación profesional continua que garantice su uso responsable y equitativo.

Palabras clave: Automatización, Depuración, Ética, Inteligencia artificial, Programación.

ABSTRACT

The study addressed the relevance of current and emerging applications of artificial intelligence (AI) in the field of programming, highlighting their impact on software development automation, code optimization, debugging, and testing processes. The main objective was to analyze how these tools have transformed traditional programming practices and the ethical implications arising from their implementation in real-world development environments. To accomplish this objective, a documentary-descriptive methodology with a qualitative approach was used. A review of updated scientific literature was conducted, along with a comparative analysis of AI tools applied to software development and a systematization of the technical and ethical challenges posed by these technologies. As a result, it was evident that AI has made it possible to automate repetitive tasks, generate efficient code, detect errors more accurately, and reduce testing times, thereby increasing productivity and software quality. However, risks associated with algorithmic opacity, shared responsibility in the event of failures, and the need for ethical regulation were also identified. It was concluded that artificial intelligence tools constitute a disruptive innovation in modern programming. However, their adoption must be accompanied by critical human oversight, sound ethical standards, and ongoing professional training to ensure their responsible and equitable use.

Keywords: Automation, Debugging, Ethics, Artificial intelligence, Programming.

Introducción

La inteligencia artificial (IA) ha transformado significativamente la forma en que interactuamos con la tecnología, y uno de los campos donde su impacto ha sido más profundo es en la programación. Desde sus inicios como un concepto teórico, la IA ha evolucionado para convertirse en una herramienta esencial en el desarrollo de software, automatizando tareas complejas, optimizando procesos y permitiendo a los programadores concentrarse en aspectos más creativos e innovadores de su trabajo (Zhu et al., 2019)

En sus primeras etapas, la IA fue vista principalmente como una curiosidad académica, con aplicaciones limitadas en el mundo real. Sin embargo, los avances en la capacidad de cómputo y el desarrollo de algoritmos más sofisticados han permitido que la IA se convierta en una parte integral del proceso de programación. Desde la generación automática de código hasta la optimización del rendimiento y la detección de errores, la IA ha demostrado ser un recurso invaluable para los programadores modernos.

El autor Gulwani, (2017) ha señalado que "la inteligencia artificial es probablemente la tecnología más transformadora de nuestro tiempo. Su capacidad para aprender y adaptarse ha abierto nuevas fronteras en la programación, permitiendo a los desarrolladores crear software más robusto y eficiente que nunca antes". Esta cita refleja el papel central de la IA en la evolución del desarrollo de software.

El futuro de la programación está, sin duda, entrelazado con la evolución continua de la IA. Según Andrew Ng, cofundador de Google Brain, "el futuro de la programación no es sobre líneas de código, sino sobre cómo los programadores colaboran con la IA para resolver los problemas más difíciles de nuestra era" (Mishra & Singh, 2021). Esta visión subraya la necesidad de que los desarrolladores se adapten a un entorno donde la IA no solo complementa sus habilidades, sino que también redefine la naturaleza del trabajo en programación.

La integración de la inteligencia artificial en la programación no es un fenómeno reciente, aunque su impacto ha crecido exponencialmente en las últimas décadas. Las primeras implementaciones de IA en la programación comenzaron con el desarrollo de sistemas básicos de automatización y soporte, que se centraban en simplificar y acelerar tareas específicas del ciclo de desarrollo de software.

Uno de los primeros enfoques de la IA aplicados en la programación fueron los sistemas expertos. Estos sistemas estaban diseñados para emular el juicio y la toma de decisiones de un experto humano en un dominio específico, como la depuración de código o la optimización de algoritmos. Un ejemplo temprano de esto es el sistema MYCIN, desarrollado en la década de 1970, que utilizaba reglas basadas en el conocimiento para diagnosticar infecciones bacterianas y recomendar tratamientos. Si bien MYCIN estaba orientado a la medicina, su arquitectura influyó en la creación de sistemas expertos en programación que podían identificar errores comunes y sugerir correcciones.

Otro avance significativo fue el desarrollo de algoritmos genéticos, inspirados en los principios de la evolución natural. Estos algoritmos se utilizaban para optimizar código y resolver problemas complejos mediante la simulación de procesos evolutivos, como la selección, mutación y recombinación de soluciones. En programación, los algoritmos genéticos se aplicaron para encontrar soluciones óptimas a problemas como la optimización de rutas, la asignación de recursos, y más tarde, en la optimización de código fuente para mejorar el rendimiento y la eficiencia.

El objetivo principal de este estudio es analizar las aplicaciones actuales y emergentes de la inteligencia artificial en el campo de la programación, evaluando su impacto en la automatización del desarrollo de software,

la optimización del código, la depuración y los procesos de prueba, así como los desafíos éticos asociados con su implementación en entornos reales de desarrollo.

Desarrollo

A finales de la década de 1980 y principios de 1990, comenzaron a surgir herramientas que utilizaban IA para analizar código fuente y generar automáticamente fragmentos de código repetitivos o rutinarios. Estas herramientas se basaban en técnicas de análisis estático y aprendizaje de patrones para identificar secciones del código que podían ser mejoradas o automatizadas.

Con el advenimiento del aprendizaje automático (Machine Learning) a partir de los años 90, la IA comenzó a integrarse más profundamente en la programación. Los primeros modelos de aprendizaje automático podían analizar grandes volúmenes de código para detectar patrones y predecir errores, lo que ayudaba a los desarrolladores a prevenir fallos antes de que ocurrieran (Zhu, 2019).

Estas primeras incursiones en la integración de IA en la programación no solo transformaron la manera en que se desarrollaba software en aquel entonces, sino que también abrieron el camino para las herramientas y tecnologías avanzadas que hoy en día continúan revolucionando el campo de la programación.

Las aplicaciones de la inteligencia artificial en la programación son vastas y continúan expandiéndose a medida que la tecnología avanza. Entre las aplicaciones clave se encuentran la automatización del desarrollo de software, la optimización y mantenimiento de código, y la mejora de los procesos de depuración y pruebas de software (Alzubaidi, 2021).

La automatización del desarrollo de software ha sido uno de los avances más destacados de la IA en programación. Herramientas como GPT-4 permiten generar código a partir de descripciones en lenguaje natural, lo que facilita a los desarrolladores la creación de software sin necesidad de escribir cada línea de código manualmente. Además, los modelos generativos son capaces de crear componentes de software completos, optimizando el tiempo de desarrollo y reduciendo los errores humanos.

En cuanto a la optimización y mantenimiento de código, la IA ofrece soluciones avanzadas para refactorizar código automáticamente, mejorar la eficiencia del software y detectar vulnerabilidades de seguridad antes de que se conviertan en problemas graves. Las herramientas de optimización basadas en IA pueden ajustar el rendimiento del software en tiempo real, asegurando que el código funcione de manera óptima bajo diferentes condiciones (Vaswani, 2017)

La IA también ha revolucionado la depuración y pruebas de software, generando automáticamente casos de prueba y utilizando machine learning para diagnosticar y resolver errores. Esto ha permitido a los desarrolladores reducir significativamente el tiempo de testing, lanzando productos más rápidamente sin comprometer la calidad.

En conjunto, estas aplicaciones no solo aumentan la eficiencia y calidad del software, sino que también permiten a los desarrolladores centrarse en aspectos más creativos y estratégicos del desarrollo, dejando que la IA maneje las tareas más repetitivas y técnicas.

A pesar de los numerosos beneficios que la IA aporta a la programación, también presenta desafíos éticos significativos que deben ser abordados para garantizar su uso responsable y equitativo. Entre estos desafíos se incluyen la transparencia y explicabilidad de los sistemas de IA, la responsabilidad y rendición de cuentas, la accesibilidad y la desigualdad.

La falta de transparencia y explicabilidad en las decisiones tomadas por algoritmos de IA es un desafío crucial. A medida que la IA toma un papel más activo en la generación y optimización de código, es esencial que las decisiones automatizadas sean

comprensibles y justificables. La opacidad de algunos modelos de IA, especialmente aquellos basados en redes neuronales profundas, plantea el riesgo de que errores o sesgos no se detecten fácilmente, lo que podría llevar a problemas graves en el software producido.

La responsabilidad y rendición de cuentas también son aspectos fundamentales. A medida que la IA asume más tareas en el desarrollo de software, surge la pregunta de quién es responsable de los errores o fallos que resulten del código generado por IA. Es crucial establecer marcos claros de responsabilidad para garantizar que los programadores, las empresas y las herramientas de IA compartan la responsabilidad en caso de fallos.

Además, la creciente dependencia de herramientas de IA en la programación podría exacerbar las desigualdades existentes en el acceso a la tecnología. Es vital considerar cómo se pueden hacer estas herramientas accesibles y asequibles para un público más amplio, evitando la concentración del poder tecnológico en manos de unos pocos.

Finalmente, la ética en la programación asistida por IA exige una supervisión constante y un compromiso con el uso justo y equitativo de estas tecnologías, asegurando que beneficien a todos y no solo a una élite tecnológica.

La inteligencia artificial ha demostrado ser una herramienta esencial en la programación, mejorando la eficiencia, calidad y creatividad en el desarrollo de software. Su integración no solo ha transformado el rol del programador, sino que también ha presentado desafíos éticos que deben ser abordados para garantizar un uso responsable.

El futuro de la programación está íntimamente ligado a la evolución de la IA. Los programadores que adopten estas tecnologías y se adapten a sus nuevas capacidades estarán mejor preparados para liderar la próxima generación de innovación tecnológica. Sin embargo, es esencial que se mantenga un enfoque ético y equitativo en su desarrollo y despliegue, asegurando que la IA beneficie a todos y no perpetúe desigualdades.

La transparencia, responsabilidad y accesibilidad deben ser prioridades al integrar la IA en la programación. A medida que la IA continúa evolucionando, será fundamental que los desarrolladores y las empresas adopten una actitud proactiva en la gestión de estos desafíos, garantizando que la tecnología se utilice de manera justa y equitativa.

La preparación para el futuro de la programación no solo implica dominar nuevas herramientas, sino también estar dispuesto a enfrentar y resolver los desafíos éticos que acompañan a la IA, asegurando que su impacto sea positivo y beneficioso para toda la sociedad.

Metodología

El estudio fue de tipo documental-descriptivo con enfoque cualitativo y elementos de análisis técnico.

Se aplicaron métodos científicos tales como:

Revisión bibliográfica: Se recopilaron y analizaron fuentes académicas indexadas (artículos científicos, libros, conferencias) sobre inteligencia artificial aplicada a la programación, priorizando publicaciones de los últimos cinco años.

Análisis de contenido: Se sistematizaron las principales aplicaciones de IA en el ciclo de vida del software (generación automática de código, testing, refactorización, etc.), identificando beneficios técnicos, herramientas utilizadas (como GPT, algoritmos genéticos, aprendizaje automático) y limitaciones.

Análisis ético y crítico: Reflexión sistemática sobre las implicaciones éticas de la delegación de tareas a sistemas de IA en programación, especialmente en cuanto a transparencia, responsabilidad, accesibilidad y sesgos.

Como instrumentos de apoyo, se utilizaron matriz de análisis documental, fichas técnicas de herramientas IA aplicadas y

criterios éticos de evaluación basados en literatura académica.

Resultados y Discusión

Importancia de las aplicaciones de la inteligencia artificial en la programación

Las aplicaciones actuales y emergentes de la inteligencia artificial (IA) en el campo de la programación representan un avance trascendental en la evolución del desarrollo de software. Gracias a la integración de técnicas como el aprendizaje automático, los modelos generativos y los algoritmos evolutivos, la IA ha redefinido los procesos tradicionales de codificación, ofreciendo soluciones automatizadas, más eficientes y con menor margen de error humano.

Una de las contribuciones más destacadas de la IA es la automatización del desarrollo de software, lo que permite generar fragmentos o módulos completos de código a partir de instrucciones en lenguaje natural. Herramientas como GitHub Copilot o modelos como GPT-4 han revolucionado la forma en que los desarrolladores abordan sus tareas, disminuyendo el tiempo de desarrollo y liberando recursos cognitivos para actividades creativas y de diseño arquitectónico.

En cuanto a la optimización del código, la IA permite identificar cuellos de botella en el rendimiento, detectar patrones ineficientes y proponer alternativas que mejoran la eficiencia energética, la velocidad de ejecución y la escalabilidad del software. Asimismo, los sistemas basados en IA pueden realizar refactorizaciones automáticas que tradicionalmente requerían un análisis manual detallado.

La IA también ha transformado los procesos de depuración y pruebas de software, mediante la generación automática de casos de prueba, la predicción de errores y la localización de fallas en el código. Estas capacidades no solo reducen los costos y tiempos asociados al ciclo de pruebas, sino que también aumentan la confiabilidad y seguridad de las aplicaciones.

No obstante, junto con estos avances técnicos, emergen desafíos éticos significativos que deben ser considerados. Entre ellos se encuentran la transparencia y explicabilidad de los modelos de IA que generan código, la responsabilidad compartida ante errores generados por sistemas automatizados, y la necesidad de garantizar el acceso equitativo a estas herramientas. Además, la dependencia creciente de soluciones basadas en IA podría reproducir o incluso amplificar sesgos existentes si no se establecen mecanismos claros de evaluación y control.

En conjunto, el uso de IA en programación no solo representa una mejora técnica, sino también un cambio paradigmático que exige nuevas competencias profesionales, un enfoque ético riguroso y una gobernanza tecnológica que priorice la inclusión, la transparencia y la sostenibilidad del desarrollo digital.

A continuación, se muestra el análisis detallado sobre las aplicaciones actuales y emergentes de la inteligencia artificial (IA) en la programación, incluyendo su impacto en el desarrollo de software y los desafíos éticos asociados:

Tabla 1. Análisis de aplicaciones de IA en programación

Aplicación de IA	Impacto en la automatización del desarrollo de software	Optimización del código	Depuración y pruebas de software	Desafíos éticos asociados
Generación automática de código (ej. GitHub Copilot,	Permite crear funciones o programas completos a partir de lenguaje natural, reduciendo	Baja eficiencia en algunos casos; el código generado puede necesitar	Detecta patrones comunes de errores, pero puede generar	Opacidad del modelo, autoría del código, atribución de responsabilidades

GPT-4)	el tiempo de codificación manual.	ajustes manuales.	código con fallos no evidentes.	en caso de fallos.
Sistemas expertos	Automatizan decisiones repetitivas en el ciclo de desarrollo (por ejemplo, selección de estructuras de datos).	Ayudan a elegir estructuras o algoritmos adecuados según contexto.	Capaces de sugerir correcciones a errores comunes.	Limitada adaptabilidad, sesgos en la base de conocimiento, necesidad de validación humana.
Algoritmos genéticos	Generan soluciones innovadoras a problemas complejos sin intervención humana directa.	Optimizan código de forma evolutiva (rendimiento, consumo de recursos).	No se enfocan directamente en depuración, pero pueden generar soluciones eficientes.	Poca trazabilidad del proceso, complejidad para interpretar soluciones generadas.
Machine Learning para análisis de código	Automatiza la revisión de código, identificando patrones y sugerencias en grandes volúmenes de datos.	Mejora prácticas de codificación según análisis predictivo.	Detecta errores antes de que ocurran mediante predicción basada en entrenamiento previo.	Riesgo de sesgos en los datos de entrenamiento, dependencia de calidad de los datos.
Pruebas automatizadas con IA	Automatiza la creación de casos de prueba dinámicos y adaptativos.	Evalúa la eficiencia del código durante la prueba.	Mejora la cobertura y calidad de pruebas, detectando errores difíciles de encontrar manualmente.	Excesiva confianza en los resultados, posibles falsos negativos o positivos, falta de supervisión humana.
IA para refactorización automática	Sugiere o realiza cambios estructurales en el código sin modificar su funcionalidad.	Incrementa la eficiencia, legibilidad y mantenibilidad del software.	Puede prevenir errores futuros por malas prácticas.	Riesgo de modificación indeseada del comportamiento del código, falta de justificación del cambio.

La incorporación de herramientas de inteligencia artificial en el campo de la programación representa una de las transformaciones más significativas de la era digital. Cada una de las aplicaciones analizadas desde la generación automática de código hasta los sistemas expertos y los algoritmos de aprendizaje automático demuestra el potencial de la IA para automatizar tareas complejas, optimizar procesos técnicos y mejorar la calidad del software desarrollado.

Estas tecnologías no solo permiten reducir significativamente el tiempo y el esfuerzo requeridos en el desarrollo de software, sino que también habilitan nuevas formas de pensar y ejecutar soluciones informáticas. Herramientas como GitHub Copilot, los algoritmos genéticos o los sistemas de pruebas inteligentes constituyen un avance innovador que redefine el rol tradicional del programador, pasando de ser un simple codificador a un diseñador estratégico de soluciones asistidas por IA.

La optimización del código es otra área clave donde la IA contribuye directamente a mejorar el rendimiento, la legibilidad y la mantenibilidad del software, aspectos cruciales en sistemas de gran escala. Igualmente, los sistemas inteligentes de depuración y pruebas permiten una detección temprana de errores, lo que fortalece la confiabilidad de los productos digitales y reduce los costos asociados a fallos en etapas avanzadas del ciclo de vida del software.

No obstante, el avance de estas tecnologías también plantea desafíos éticos fundamentales: la opacidad de los algoritmos, la atribución de responsabilidades por errores generados automáticamente, los sesgos en los datos de entrenamiento y la necesidad de garantizar un acceso equitativo a estas herramientas. Estos retos deben ser abordados desde una perspectiva multidisciplinaria que combine el conocimiento técnico con principios éticos y sociales.

En perspectiva, las herramientas de inteligencia artificial aplicadas a la programación no solo consolidan su relevancia en el presente, sino que se proyectan como motores clave para la innovación tecnológica futura. La evolución del desarrollo de software dependerá cada vez más de la capacidad para integrar soluciones inteligentes, fomentar la colaboración humano-máquina y construir entornos de programación más adaptativos, eficientes y éticos.

Diversos autores coinciden en que la inteligencia artificial está redefiniendo los paradigmas del desarrollo de software, al introducir nuevas formas de automatización y asistencia al programador. Según (Fernández et al., 2024), el aprendizaje profundo y los algoritmos de redes neuronales han permitido a la IA comprender estructuras complejas de código, generando soluciones adaptativas y eficientes en tiempo real. Esta capacidad resulta particularmente valiosa en contextos donde el volumen de datos y la velocidad de entrega son críticos, como en el desarrollo ágil o en la ingeniería de software para sistemas inteligentes.

Por su parte, Pommier (2024), destacan que las herramientas generativas de IA, como los modelos capaces de redactar comentarios y documentación de código, mejoran la comprensión colectiva de los proyectos de software y favorecen la mantenibilidad a largo plazo. Este tipo de herramientas no solo automatizan tareas rutinarias, sino que actúan como mediadoras en la comunicación técnica entre equipos de desarrollo, reduciendo la ambigüedad y potenciando el trabajo colaborativo. En este sentido, la IA no sustituye al programador, sino que amplifica sus capacidades cognitivas y comunicativas.

Sin embargo, Goodfellow et al. (2018), advierten sobre los riesgos que implica delegar decisiones críticas al juicio autónomo de algoritmos, especialmente en entornos donde la seguridad y la ética son prioritarios. La generación automática de código o la autoevaluación de sistemas pueden producir errores sutiles o sesgos ocultos que pasen desapercibidos si no hay supervisión humana. Por tanto, la implementación de herramientas de IA en programación debe ir acompañada de políticas claras de transparencia algorítmica, trazabilidad y control de calidad, garantizando un uso responsable y equitativo de la tecnología.

Conclusiones

La inteligencia artificial se ha consolidado como una herramienta esencial en el desarrollo de software, al automatizar procesos clave como la generación de código, la refactorización y la creación de pruebas automatizadas. Estas aplicaciones no solo optimizan el tiempo y los recursos en las etapas de programación, sino que también permiten a los desarrolladores enfocarse en aspectos más estratégicos y creativos del proceso, impulsando así una evolución significativa en las metodologías de desarrollo.

El uso de herramientas de IA en la programación ha demostrado un impacto positivo en la eficiencia, calidad y mantenimiento del software, sin embargo, también exige una evaluación ética constante. La implementación de modelos generativos, algoritmos de aprendizaje y sistemas expertos plantea retos relacionados con la transparencia, la responsabilidad legal y la equidad en el acceso a estas tecnologías, por lo que su uso debe ir acompañado de marcos normativos y buenas prácticas profesionales.

Las aplicaciones actuales y emergentes de la IA en programación representan una vía

hacia la innovación tecnológica futura, al transformar no solo los procesos técnicos, sino también los perfiles y competencias requeridas en el ámbito profesional. Es indispensable que los desarrolladores del presente se preparen para colaborar con sistemas inteligentes, adaptarse al cambio continuo y asumir una postura crítica y ética frente a los desafíos que acompañan esta revolución tecnológica.

Bibliografía

Alzubaidi, L., Zhang, J., Humaidi, A. J., Al-Dujaili, A., Duan, Y., Al-Shamma, O., ... & Farhan, L. (2021). Review of deep learning: Concepts, CNN architectures, challenges, applications, future directions. Journal of Big Data, 8(1), 1-74. https://journalofbigdata.springeropen.com/articles/10.1186/s40537-021-00444-8

Fernández Miranda, M., Román Acosta, D., Jurado Rosas, A. A., Limón Domínguez, D., & Torres Fernandez, C. (2024). Artificial Intelligence in Latin American Universities: Emerging Challenges. Computación y Sistemas, 28(2). https://doi.org/10.13053/CyS-28-2-48222

Goodfellow, I., McDaniel, P., & Papernot, N. (2018). Making machine learning robust against adversarial inputs. Communications of the ACM, 61(7), 56-66. https://dl.acm.org/doi/10.1145/3134599

Gulwani, S. (2017). Automating programming education: A step towards meeting society's tech challenges. Communications of the ACM, 61(8), 28-30. Leeds, D., West, J., Lee, L. (2020). Object Oriented Learning Assistant: Using NLP to Help Novices While They Code. arXiv preprint arXiv:2010.12697.

Mishra, A., y Singh, V. (2021). Natural language programming. In Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering (pp. 1650-1654)

Pommier Gallo, EP (2024). Metodología para Mejorar la Programación con Inteligencia Artificial. Actas Iberoamericanas En Ciencias Sociales , 2 (1), 86-97. https://doi.org/10.69821/AICIS.v2i1.16

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. Advances in Neural Information Processing Systems, 30, 5998-6008. https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf

Zhu, J., Li, H., Nie, L., Xu, X., Yu, J., & Zhang, B. (2019). Learning to generate comprehensive program comments from source code. Proceedings of the 42nd International Conference on Software Engineering, 752-764. https://dl.acm.org/doi/10.1145/3377811.3380418

Cómo citar: Cedeño Pincay, W. D. (2025). Aplicaciones de la Inteligencia Artificial en el campo de la programación. Journal TechInnovation, 4(1). https://doi.org/10.47230/Journal.TechInnovation.v4.n1.2025.4-12