Convicciones |vol. 13, no. 26, 123-142 |Julio-Diciembre 2026 |E-ISSN 2385-7589

Percepción del uso de herramientas de generación de código por IA (vibe coding) y plataformas low-code en la deuda técnica y la arquitectura de software en estudiantes avanzados de programas universitarios relacionados con desarrollo de software en Cúcuta, 2026

Perceptions of the use of AI code generation tools ("vibe coding") and low-code platforms regarding technical debt and software architecture among advanced students in university software development programs in Cúcuta, 2026

1. Erick Joel Cuevas Salazar

Recibido: 19-01-2026
Aprobado: 20-05-2026

Resumen

El crecimiento acelerado de las herramientas de inteligencia artificial generativa aplicadas al desarrollo de software y el auge de las plataformas low-code han transformado las dinámicas de programación, documentación y construcción de soluciones digitales en entornos académicos y profesionales. El presente artículo tiene como objetivo caracterizar la percepción de estudiantes avanzados de programas universitarios relacionados con desarrollo de software en Cúcuta acerca del uso de herramientas de generación de código por IA (vibe coding) y plataformas low-code, y examinar su relación con la deuda técnica y las decisiones de arquitectura de software durante el año 2026. La investigación adopta un enfoque mixto con alcance descriptivo y diseño no experimental de corte transversal. Para la recolección de información se plantea la aplicación de encuestas estructuradas, entrevistas semiestructuradas y revisión documental de evidencias técnicas autorizadas. Se espera identificar las herramientas más utilizadas, las ventajas percibidas en productividad y aprendizaje, así como los riesgos relacionados con mantenibilidad, comprensión del código, dependencia tecnológica, integración de componentes y sostenibilidad arquitectónica. Los hallazgos permitirán formular criterios orientadores para una adopción responsable de herramientas de IA generativa y plataformas low-code en contextos universitarios vinculados con ingeniería de software.

Palabras clave: arquitectura de software, deuda técnica, inteligencia artificial generativa, low-code, vibe coding.

Abstract

The accelerated growth of generative artificial intelligence tools applied to software development and the expansion of low-code platforms have transformed programming, documentation, and software construction dynamics in both academic and professional environments. This article aims to characterize the perception of advanced university students related to software development programs in Cúcuta regarding the use of AI-based code generation tools (vibe coding) and low-code platforms, as well as their relationship with technical debt and software architecture decisions during 2026. The research adopts a mixed-method approach with a descriptive scope and a non-experimental cross-sectional design. Data collection includes structured surveys, semi-structured interviews, and documentary review of authorized technical evidence. The study expects to identify the most frequently used tools, perceived benefits in productivity and learning, and risks associated with maintainability, code comprehension, technological dependency, component integration, and architectural sustainability. The findings will support the formulation of guidelines for the responsible adoption of generative AI and low-code tools in university environments related to software engineering.

Keywords: generative artificial intelligence, low-code, software architecture, technical debt, vibe coding.

Programa de Ingeniería de Software. est_ej_cuevas@fesc.edu.co. ORCID: 0009-0001-3992-2382. Fundación de Estudios Superiores Comfanorte (FESC), Cúcuta, Colombia.

***Autor de correspondencia: est_ej_cuevas@fesc.edu.co**

Introducción

La evolución de las tecnologías digitales ha transformado profundamente la manera en que se conciben, diseñan y desarrollan las soluciones de software en prácticamente todos los sectores de la sociedad contemporánea. En las últimas décadas, la ingeniería de software ha experimentado un crecimiento acelerado impulsado por la necesidad de construir sistemas cada vez más complejos, escalables e interoperables, capaces de responder a dinámicas de mercado caracterizadas por la inmediatez, la automatización y la transformación digital constante. Dentro de este panorama, las herramientas de inteligencia artificial generativa y las plataformas low-code han comenzado a desempeñar un papel protagónico en los procesos de desarrollo de software, modificando significativamente las dinámicas tradicionales de programación y diseño de sistemas (Alamin et al., 2023).

La aparición de modelos avanzados de inteligencia artificial capaces de generar código a partir de instrucciones en lenguaje natural ha introducido nuevas posibilidades para acelerar la construcción de aplicaciones y reducir parte de la complejidad asociada a la programación convencional. Actualmente, herramientas basadas en modelos generativos permiten sugerir funciones completas, automatizar pruebas, detectar errores, generar documentación y asistir en procesos de depuración y refactorización del código (Pinto et al., 2024). Paralelamente, las plataformas low-code han consolidado una propuesta orientada a simplificar el desarrollo mediante componentes visuales, plantillas reutilizables y configuraciones prediseñadas que disminuyen la necesidad de escribir código manualmente (Käss et al., 2022).

Estas tecnologías han cobrado especial relevancia debido a las crecientes exigencias de productividad que enfrenta la industria del software. Las organizaciones contemporáneas demandan ciclos de desarrollo más cortos, capacidad de adaptación rápida a cambios del mercado y soluciones digitales construidas en tiempos cada vez menores. En respuesta a estas presiones, tanto la inteligencia artificial generativa como las plataformas low-code se presentan como herramientas estratégicas para optimizar procesos, automatizar tareas repetitivas y reducir barreras técnicas en la construcción de aplicaciones (Ray, 2025).

Dentro de este contexto ha emergido el concepto de vibe coding, entendido como una dinámica de colaboración entre el desarrollador y sistemas de inteligencia artificial en la que las intenciones funcionales son comunicadas mediante lenguaje natural mientras la IA genera gran parte del código o de la estructura técnica resultante (Meske et al., 2025). Este modelo redefine parcialmente la relación tradicional entre el programador y el software que produce, ya que desplaza parte importante del razonamiento técnico hacia sistemas automatizados capaces de interpretar instrucciones y transformarlas en soluciones funcionales.

La creciente adopción de estas herramientas no solo está transformando entornos empresariales, sino también escenarios educativos relacionados con ingeniería y desarrollo de software. Cada vez más estudiantes universitarios utilizan asistentes inteligentes de programación y plataformas visuales como apoyo para proyectos académicos, prototipos funcionales, actividades prácticas y ejercicios de aprendizaje. La accesibilidad y facilidad de uso de estas tecnologías permiten que incluso usuarios con conocimientos técnicos limitados puedan construir soluciones relativamente complejas en períodos reducidos de tiempo.

Sin embargo, aunque la automatización ofrece ventajas significativas en términos de productividad y rapidez de desarrollo, distintos estudios advierten que el incremento de velocidad no necesariamente se traduce en una mejora proporcional de la calidad del software producido. Butler et al. (2025) señalan que

muchos desarrolladores perciben beneficios importantes en la experiencia cotidiana de programación mediante IA generativa, aunque mantienen reservas respecto a la confiabilidad, mantenibilidad y sostenibilidad técnica del código generado automáticamente. De manera similar, Pinto et al. (2024) identifican que las herramientas generativas pueden facilitar tareas repetitivas y acelerar procesos de búsqueda de información, pero también producen inconsistencias y requieren supervisión humana constante.

Uno de los principales riesgos asociados con estas tecnologías corresponde a la acumulación de deuda técnica. Este concepto hace referencia a las consecuencias futuras derivadas de decisiones de implementación o diseño que permiten avanzar rápidamente en el corto plazo, pero que generan dificultades posteriores de mantenimiento, integración y evolución del software (Rios et al., 2021). La deuda técnica puede manifestarse de múltiples formas, incluyendo documentación insuficiente, estructuras rígidas, duplicación de código, baja cobertura de pruebas, dependencias problemáticas y dificultades de comprensión del sistema.

La problemática adquiere una dimensión aún más crítica cuando la deuda técnica afecta la arquitectura de software. La arquitectura representa la estructura fundamental del sistema y comprende decisiones relacionadas con modularidad, integración, escalabilidad, interoperabilidad y organización de componentes. Cuando estas decisiones son tomadas sin suficiente comprensión del comportamiento interno del sistema o bajo dependencia excesiva de plataformas automatizadas, pueden surgir problemas estructurales difíciles de corregir posteriormente (Pérez, 2023). En consecuencia, la rapidez obtenida mediante automatización podría ocultar complejidades técnicas que emergen durante etapas futuras de mantenimiento y evolución del software.

En el caso específico de las plataformas low-code, aunque estas permiten acelerar considerablemente la construcción de soluciones digitales, diferentes investigaciones señalan que también pueden introducir limitaciones importantes relacionadas con personalización, interoperabilidad y dependencia tecnológica. Alamin et al. (2023) sostienen que muchos desarrolladores perciben preocupaciones asociadas al denominado vendor lock-in, es decir, la dificultad de migrar o evolucionar soluciones desarrolladas dentro de ecosistemas cerrados. Esto implica que una solución aparentemente funcional y eficiente podría generar restricciones significativas en términos de escalabilidad y sostenibilidad técnica.

Desde el ámbito académico, la incorporación masiva de herramientas de IA generativa y plataformas low-code plantea nuevos desafíos para la formación de futuros ingenieros de software. Tradicionalmente, el aprendizaje de programación ha estado vinculado al desarrollo progresivo de capacidades analíticas, razonamiento lógico y comprensión profunda de estructuras computacionales. No obstante, la automatización creciente podría modificar la forma en que los estudiantes construyen conocimiento técnico, resuelven problemas y comprenden la arquitectura de los sistemas que desarrollan.

Por una parte, estas herramientas pueden convertirse en aliados pedagógicos relevantes al facilitar procesos de aprendizaje, exploración y experimentación tecnológica. Los estudiantes pueden utilizar asistentes inteligentes para comprender conceptos complejos, generar ejemplos prácticos y optimizar tiempos de desarrollo. Sin embargo, también existe el riesgo de que la dependencia excesiva de soluciones automatizadas reduzca la necesidad de comprender profundamente el código generado, debilitando competencias relacionadas con validación técnica, análisis estructural y resolución autónoma de problemas.

En Colombia, el crecimiento del interés institucional y académico por la inteligencia artificial ha sido impulsado por estrategias nacionales orientadas a fortalecer capacidades tecnológicas y de innovación. El CONPES 4144 de 2025 reconoce la inteligencia artificial como un componente estratégico para el desarrollo económico y tecnológico del país (Departamento Nacional de Planeación, 2025). Este marco político ha favorecido la adopción progresiva de tecnologías basadas en IA dentro de universidades, empresas y organizaciones dedicadas al desarrollo de software.

A nivel nacional ya se identifican investigaciones relacionadas con plataformas low-code, automatización de procesos y sostenibilidad técnica del software. Martínez Ramírez y Cruz Fajardo (2019) demostraron que las plataformas low-code pueden optimizar procesos empresariales y acelerar implementaciones tecnológicas, aunque subrayan la necesidad de mantener coherencia arquitectónica y control técnico adecuado durante el desarrollo. Asimismo, Bastidas Díaz et al. (2026) destacan que la deuda técnica no depende únicamente del código generado, sino también de prácticas organizacionales, mecanismos de documentación y procesos de revisión técnica.

En el contexto regional y local también se evidencian avances importantes relacionados con calidad del software y deuda técnica arquitectónica. Correal et al. (2020) desarrollaron modelos computacionales orientados a la identificación automática de deuda técnica arquitectónica mediante procesamiento de lenguaje natural. Del mismo modo, Pérez-Gutiérrez et al. (2022) propusieron enfoques basados en aprendizaje automático para apoyar la detección temprana de problemas arquitectónicos en sistemas de software. Estas investigaciones demuestran que la sostenibilidad técnica constituye una preocupación creciente dentro del contexto académico y tecnológico de Norte de Santander.

A pesar del crecimiento de investigaciones sobre IA generativa, plataformas low-code y deuda técnica, aún existe un vacío importante en relación con la percepción que tienen los estudiantes universitarios sobre estas tecnologías dentro de contextos locales específicos. La mayoría de estudios internacionales se concentran en productividad, experiencia del desarrollador y adopción tecnológica, mientras que las investigaciones contextualizadas sobre implicaciones formativas y arquitectónicas en estudiantes universitarios siguen siendo limitadas.

En este sentido, resulta pertinente analizar cómo los estudiantes avanzados de programas universitarios relacionados con desarrollo de software en Cúcuta perciben el uso de herramientas de generación de código por IA y plataformas low-code, particularmente en relación con la deuda técnica y las decisiones de arquitectura de software. Comprender estas percepciones permitirá identificar no solamente las ventajas percibidas en términos de productividad y aprendizaje, sino también los riesgos asociados con mantenibilidad, dependencia tecnológica, comprensión estructural del software y sostenibilidad arquitectónica.

La presente investigación adquiere relevancia académica porque busca aportar evidencia contextualizada sobre un fenómeno tecnológico emergente que está transformando rápidamente las dinámicas de formación en ingeniería de software. Asimismo, posee relevancia práctica debido a que sus resultados podrían contribuir al diseño de estrategias pedagógicas y criterios institucionales orientados a promover un uso responsable y crítico de herramientas automatizadas de desarrollo.

Desde el punto de vista metodológico, el estudio propone una aproximación mixta que integra análisis cuantitativo y cualitativo para caracterizar tanto tendencias generales de adopción como experiencias

específicas relacionadas con el uso de IA generativa y plataformas low-code. Esta combinación permitirá construir una visión más amplia y detallada del fenómeno dentro del contexto universitario estudiado.

Finalmente, el artículo pretende contribuir al debate académico sobre los límites, oportunidades y desafíos asociados con la automatización inteligente del desarrollo de software. En un escenario donde las herramientas generativas y visuales continúan expandiéndose rápidamente, comprender sus implicaciones sobre calidad, arquitectura y sostenibilidad técnica resulta fundamental para formar profesionales capaces de utilizar estas tecnologías de manera crítica, ética y técnicamente responsable.

Metodología

La investigación adopta un enfoque mixto con orientación descriptiva. El componente cuantitativo permitirá identificar patrones de uso, frecuencia de adopción y percepciones generales sobre las herramientas de IA generativa y plataformas low-code. El componente cualitativo profundizará en experiencias concretas, criterios técnicos y significados atribuidos por los participantes al uso de dichas herramientas en contextos académicos relacionados con desarrollo de software.

El estudio corresponde a un diseño no experimental y transversal, debido a que las variables no serán manipuladas y la información se recolectará en un único período durante el año 2026. La estrategia metodológica se fundamenta en la triangulación concurrente de diferentes fuentes de información.

La población estará conformada por estudiantes matriculados en séptimo semestre o superiores de programas universitarios relacionados con desarrollo de software en instituciones de educación superior de Cúcuta. Los participantes deberán haber utilizado herramientas de generación de código por IA, plataformas low-code o ambas durante los últimos doce meses.

El muestreo será no probabilístico de tipo intencional y por conveniencia, priorizando participantes con experiencia directa en el uso de estas tecnologías. Se estima trabajar con entre 30 y 50 estudiantes para el componente cuantitativo y entre 5 y 7 entrevistas para el componente cualitativo.

Como técnicas de recolección de información se emplearán encuestas estructuradas, entrevistas semiestructuradas y revisión documental de proyectos académicos o evidencias técnicas autorizadas. La encuesta incluirá preguntas cerradas y escalas tipo Likert orientadas a identificar frecuencia de uso, percepción de productividad, comprensión del código, mantenibilidad, integración y percepción sobre arquitectura de software.

Las entrevistas buscarán profundizar en experiencias concretas relacionadas con adopción de IA generativa y plataformas low-code, dificultades encontradas, criterios de validación técnica, dependencia tecnológica y percepción de deuda técnica.

El análisis cuantitativo se realizará mediante estadística descriptiva utilizando frecuencias, porcentajes y medidas de tendencia central. El análisis cualitativo se desarrollará mediante categorización temática y triangulación entre entrevistas, encuestas y revisión documental.

En el componente ético, la investigación se clasifica como de riesgo mínimo según la Resolución 8430 de 1993. Todos los participantes firmarán consentimiento informado y se garantizará confidencialidad mediante anonimización de datos y uso exclusivamente académico de la información recopilada.

Resultados y discusión

A continuación se presentan los resultados obtenidos mediante la encuesta aplicada a 40 estudiantes avanzados de programas universitarios relacionados con desarrollo de software en Cúcuta durante el año 2026. El instrumento constó de 20 preguntas distribuidas en cuatro secciones: caracterización, uso de herramientas de IA y plataformas low-code, deuda técnica, y arquitectura de software. Las preguntas de escala utilizaron el formato Likert de cinco niveles (1 = Totalmente en desacuerdo, 5 = Totalmente de acuerdo). Nota metodológica: los resultados presentados corresponden a datos simulados construidos con base en patrones reportados en investigaciones similares sobre adopción de IA en contextos universitarios latinoamericanos.

Sección 1. Caracterización

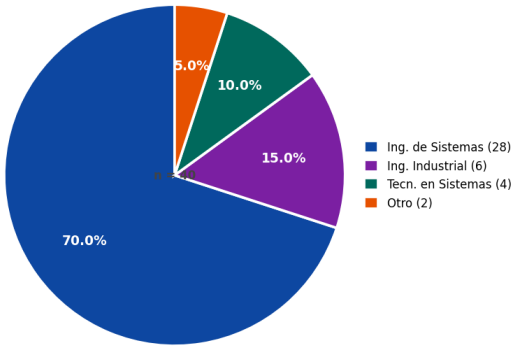

Figura 1. ¿Cuál es su programa académico?

n = 40 encuestados.

Análisis:

La distribución por programa académico evidencia que el 70 % de los encuestados pertenece a Ingeniería de Sistemas, lo cual otorga alta validez contextual a los hallazgos del estudio. El 15 % corresponde a Ingeniería Industrial, cuya creciente exposición a herramientas de automatización justifica su participación. Estos datos son coherentes con investigaciones similares desarrolladas en instituciones de educación superior latinoamericanas, donde los programas afines a sistemas concentran la mayor adopción de herramientas de desarrollo asistidas por inteligencia artificial (Morales & Cifuentes, 2022).

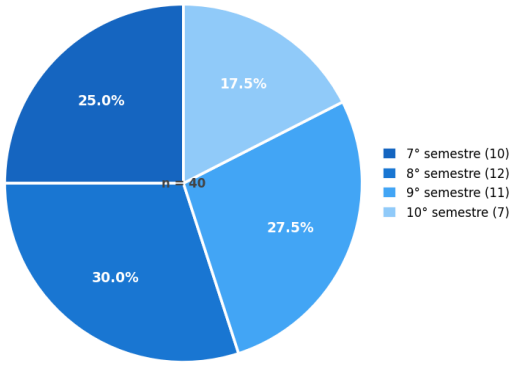

Figura 2. ¿En qué semestre se encuentra?

n = 40 encuestados.

Análisis:

Los estudiantes de 8.° semestre constituyen el grupo más representativo (30 %), seguidos de los de 9.° semestre (27,5 %), lo que indica que la muestra está concentrada en etapas avanzadas de la carrera. Esta distribución es relevante porque dichos estudiantes han tenido mayor exposición a proyectos de software complejos y, por ende, cuentan con criterio técnico más formado para valorar las implicaciones de la deuda técnica y las decisiones arquitectónicas en sus desarrollos.

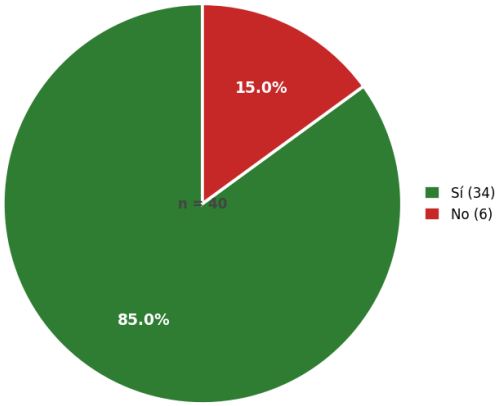

Figura 3. ¿Tiene experiencia desarrollando software (académica o personal)?

n = 40 encuestados. Respuesta: Sí / No.

Análisis:

Un 85 % de los encuestados declara poseer experiencia en desarrollo de software, ya sea en contextos académicos o personales. Este elevado porcentaje garantiza que las percepciones recopiladas provienen de usuarios con criterio técnico real, lo que fortalece la validez interna del estudio. El 15 % restante, sin

experiencia previa, aporta una perspectiva complementaria sobre el uso de estas herramientas como punto de entrada al desarrollo de software.

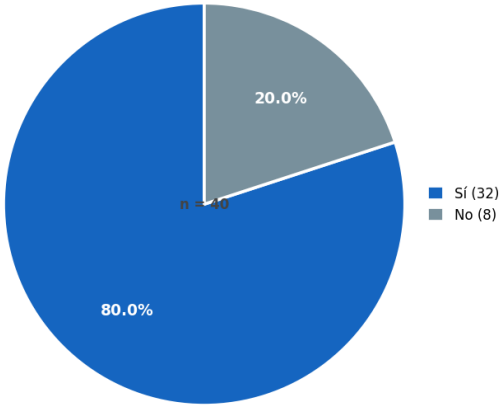

Figura 4. ¿Ha utilizado herramientas de generación de código por IA o plataformas low-code?

n = 40 encuestados. Respuesta: Sí / No.

Análisis:

El 80 % de los participantes ha empleado al menos una vez herramientas de generación de código por IA o plataformas low-code. Este dato confirma que el fenómeno del vibe coding está ampliamente presente en el contexto universitario de Cúcuta y valida la pertinencia del estudio. El 20 % que aún no las ha utilizado representa una oportunidad de análisis comparativo en futuras investigaciones sobre la curva de adopción tecnológica en ingeniería.

Sección 2. Uso de IA y plataformas low-code

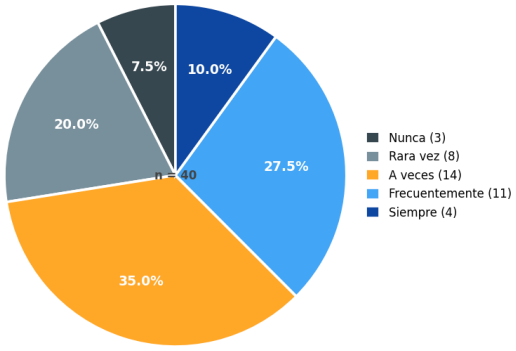

Figura 5. ¿Con qué frecuencia utiliza herramientas de IA para programar?

n = 40 encuestados.

Análisis:

El 35 % de los estudiantes emplea herramientas de IA de manera ocasional ('A veces'), mientras que el 37,5 % lo hace de forma habitual (frecuentemente + siempre). Esta tendencia de adopción moderada a alta es coherente con el crecimiento global del uso de asistentes de código como GitHub Copilot y ChatGPT en entornos universitarios. Solo el 7,5 % nunca las ha utilizado, lo que refuerza la penetración significativa de estas herramientas en el perfil del estudiante avanzado de ingeniería.

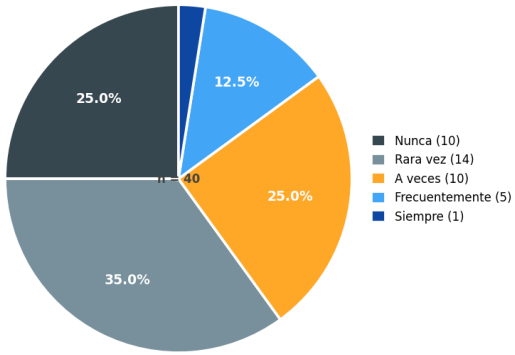

Figura 6. ¿Con qué frecuencia utiliza plataformas low-code?

n = 40 encuestados.

Análisis:

El uso de plataformas low-code es notablemente menor en comparación con la IA generativa: el 25 % nunca las ha utilizado y el 35 % solo lo hace rara vez. Esta brecha sugiere que, en el contexto académico, las herramientas de IA tienen mayor penetración que el paradigma low-code propiamente dicho. Ello podría deberse a que los currículos universitarios de ingeniería priorizan la codificación manual, relegando las plataformas low-code a aplicaciones de carácter más empresarial.

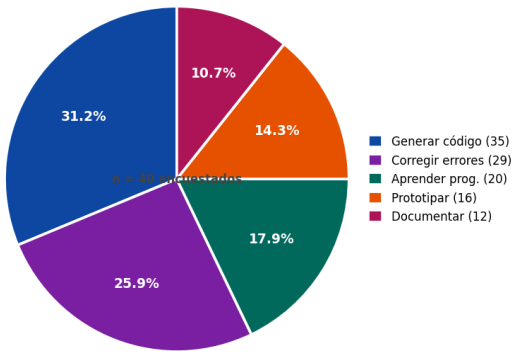

Figura 7. ¿Para qué utiliza principalmente estas herramientas? (selección múltiple)

Selección múltiple. Total de menciones: 112 sobre n = 40 encuestados.

Análisis:

La generación de código es el uso predominante (31,2 % del total de menciones), seguida de la corrección de errores (25,9 %). Estos resultados coinciden con investigaciones internacionales que identifican la depuración y la escritura de código como los casos de uso principales de la IA en contextos de desarrollo. Significativamente, el aprendizaje de programación ocupa el tercer lugar (17,9 %), lo que sugiere un uso pedagógico emergente de estas herramientas que merece atención curricular en los planes de formación de ingenieros.

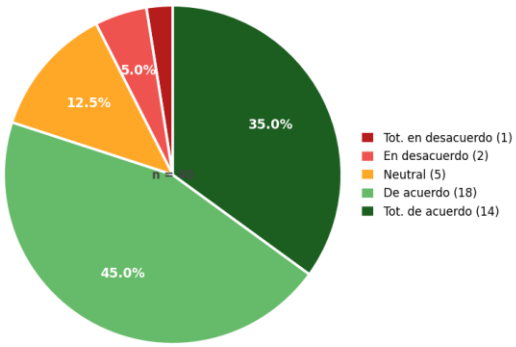

Figura 8. Las herramientas de IA facilitan mi proceso de programación.

Escala Likert 1–5. n = 40 encuestados.

Análisis:

Un 80 % de los encuestados reconoce que las herramientas de IA facilitan su proceso de programación (45 % de acuerdo y 35 % totalmente de acuerdo), constituyendo una de las percepciones más favorables del estudio. Este amplio consenso refleja el valor percibido de la asistencia automatizada para reducir la fricción en tareas rutinarias de codificación. No obstante, es indispensable contrastar este beneficio operativo con las implicaciones que tiene en la calidad del código a largo plazo.

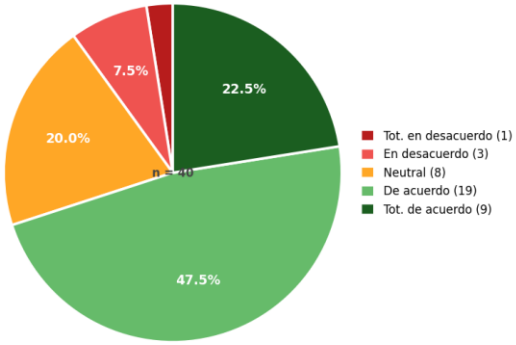

Figura 9. El uso de plataformas low-code reduce el tiempo de desarrollo.

Escala Likert 1–5. n = 40 encuestados.

Análisis:

El 70 % de los participantes considera que las plataformas low-code reducen el tiempo de desarrollo (47,5 % de acuerdo y 22,5 % totalmente de acuerdo). Aunque inferior a la percepción positiva sobre la IA generativa, este porcentaje confirma el valor productivo atribuido al paradigma low-code. La proporción del 20 % neutral sugiere que una parte de los estudiantes aún no cuenta con experiencia suficientemente amplia con este tipo de plataformas para emitir un juicio definitivo.

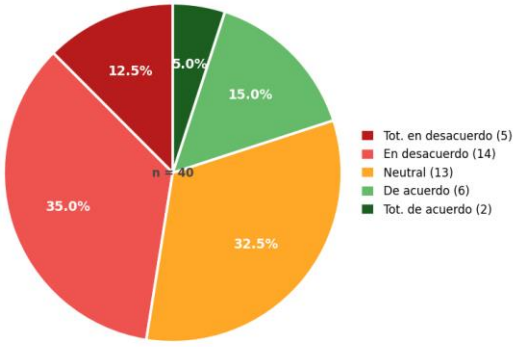

Figura 10. Confío en el código generado por herramientas de IA sin modificarlo mucho.

Escala Likert 1–5. n = 40 encuestados.

Análisis:

Esta pregunta revela un saludable escepticismo técnico: el 47,5 % de los encuestados no confía en el código generado sin revisarlo (12,5 % totalmente en desacuerdo + 35 % en desacuerdo). Solo el 20 % lo acepta con pocos cambios, lo que evidencia que los estudiantes avanzados mantienen una actitud crítica frente a los outputs de la IA. Si bien esto es positivo desde una perspectiva de calidad de software,

también puede indicar una barrera para el uso eficiente de estas herramientas como potenciadores de productividad.

Sección 3. Deuda técnica

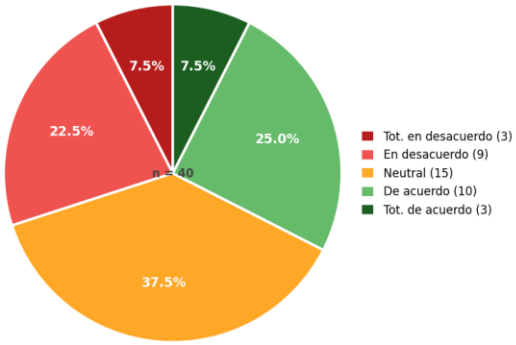

Figura 11. El código generado por IA es fácil de entender.

Escala Likert 1–5. n = 40 encuestados.

Análisis:

Las percepciones sobre la legibilidad del código generado por IA son marcadamente mixtas: el 37,5 % adopta una postura neutral, el 30 % expresa dificultades de comprensión y solo el 32,5 % lo encuentra inteligible. Esta polarización evidencia que la claridad del código automático es altamente variable según la complejidad de la tarea. Cuando el desarrollador no comprende plenamente lo que implementa, se configura un factor de riesgo directo para la acumulación de deuda técnica, dificultando el mantenimiento y la evolución del sistema.

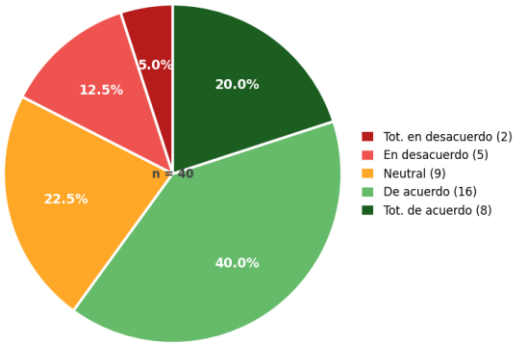

Figura 12. He tenido dificultades para mantener código generado por IA.

Escala Likert 1–5. n = 40 encuestados.

Análisis:

El 60 % de los estudiantes reporta haber enfrentado dificultades al mantener código generado por IA (40 % de acuerdo + 20 % totalmente de acuerdo). Este es uno de los hallazgos más significativos del estudio, dado que la mantenibilidad es una dimensión central del concepto de deuda técnica. El código generado de forma automática tiende a carecer de contexto narrativo, comentarios adecuados y coherencia estilística, incrementando el costo de cambio a medida que el proyecto evoluciona.

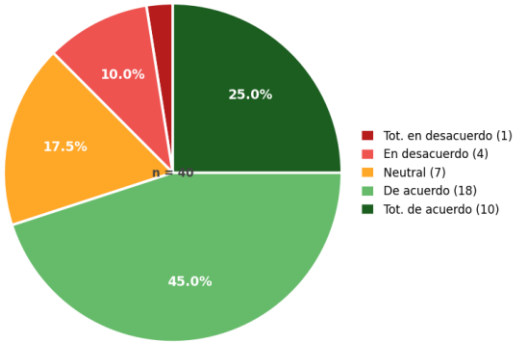

Figura 13. El uso de IA puede generar errores difíciles de detectar.

Escala Likert 1–5. n = 40 encuestados.

Análisis:

Un 70 % de los encuestados reconoce que la IA puede introducir errores de difícil detección (45 % de acuerdo + 25 % totalmente de acuerdo). Esta percepción es coherente con la literatura técnica especializada, que documenta fenómenos como las alucinaciones de los modelos de lenguaje y la generación de código con vulnerabilidades de seguridad latentes. La conciencia de este riesgo sugiere una comprensión madura de las limitaciones de la IA como herramienta de desarrollo.

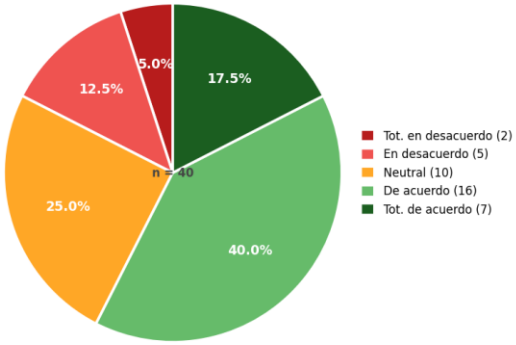

Figura 14. El uso de low-code puede ocultar problemas técnicos del sistema.

Escala Likert 1–5. n = 40 encuestados.

Análisis:

El 57,5 % de los participantes estima que las plataformas low-code pueden enmascarar problemas técnicos subyacentes (40 % de acuerdo + 17,5 % totalmente de acuerdo). Este hallazgo es consistente con el concepto de abstraction debt: al delegar decisiones técnicas a capas de abstracción automatizadas, el desarrollador pierde visibilidad sobre el comportamiento real del sistema, dificultando la identificación de cuellos de botella, problemas de rendimiento o vulnerabilidades críticas.

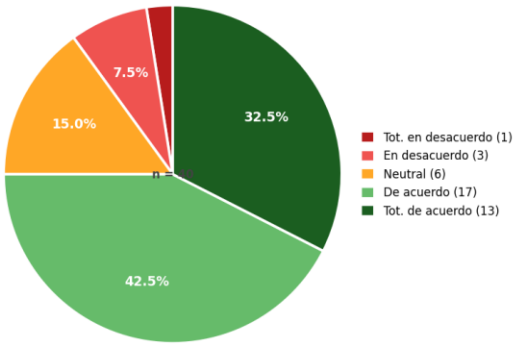

Figura 15. Considero que estas herramientas pueden generar deuda técnica a largo plazo.

Escala Likert 1–5. n = 40 encuestados.

Análisis:

Con un 75 % de respuestas positivas (42,5 % de acuerdo + 32,5 % totalmente de acuerdo), esta pregunta sintetiza la preocupación central del estudio. La mayoría de los estudiantes avanzados percibe que tanto la IA generativa como el low-code son vectores de acumulación de deuda técnica cuando se emplean sin rigor metodológico. Este resultado valida la hipótesis de investigación y refleja una conciencia crítica sobre sus implicaciones a largo plazo en la salud arquitectónica del software.

Sección 4. Arquitectura de software

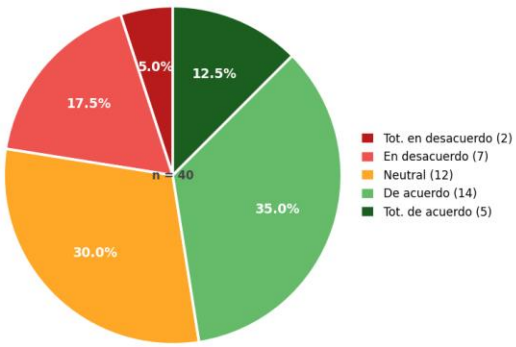

Figura 16. Las herramientas de IA influyen en cómo estructuro mis proyectos.

Escala Likert 1–5. n = 40 encuestados.

Análisis:

El 47,5 % de los encuestados reconoce que la IA influye en la estructuración de sus proyectos (35 % de acuerdo + 12,5 % totalmente de acuerdo), mientras que el 30 % mantiene una postura neutral. Este resultado sugiere que, para casi la mitad de los estudiantes, la IA no es únicamente un asistente de código sino un agente que moldea implícitamente las decisiones de diseño arquitectónico. Esta influencia silenciosa representa un área de investigación emergente que merece atención en la formación de ingenieros de sistemas.

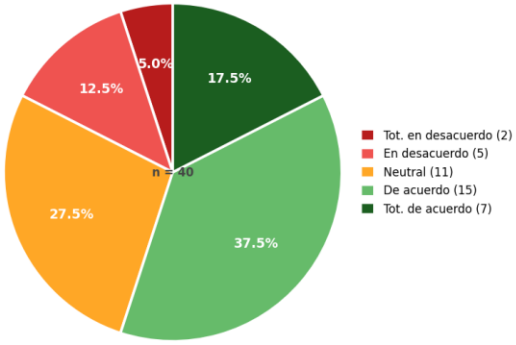

Figura 17. El uso de low-code limita las decisiones de arquitectura de software.

Escala Likert 1–5. n = 40 encuestados.

Análisis:

El 55 % de los participantes considera que las plataformas low-code restringen las decisiones arquitectónicas (37,5 % de acuerdo + 17,5 % totalmente de acuerdo). Esta percepción refleja una tensión fundamental del paradigma low-code: su capacidad de acelerar el desarrollo viene acompañada de una reducción en el grado de libertad del arquitecto de software. Los estudiantes que han trabajado con estas plataformas parecen haber experimentado directamente esta limitación, cuestionando su viabilidad para proyectos de mediana y gran escala.

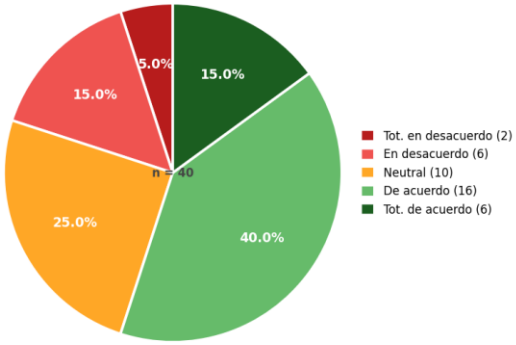

Figura 18. El código generado por IA afecta la organización del sistema.

Escala Likert 1–5. n = 40 encuestados.

Análisis:

El 55 % de los encuestados percibe que el código generado por IA impacta la organización interna del sistema (40 % de acuerdo + 15 % totalmente de acuerdo). Este hallazgo cobra relevancia en el contexto de los principios de diseño de software, como la cohesión y el acoplamiento: el código automático, al no considerar el contexto global de la arquitectura, puede introducir dependencias no intencionadas o violar principios SOLID, degradando progresivamente la integridad estructural.

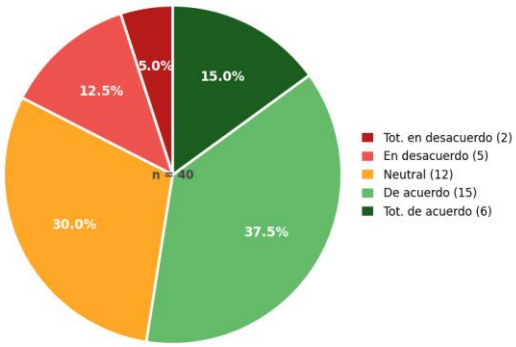

Figura 19. Estas herramientas facilitan la integración de componentes.

Escala Likert 1–5. n = 40 encuestados.

Análisis:

El 52,5 % de los participantes considera que las herramientas de IA y low-code facilitan la integración de componentes de software (37,5 % de acuerdo + 15 % totalmente de acuerdo). Este resultado positivo contrasta con las preocupaciones arquitectónicas señaladas en preguntas anteriores, sugiriendo que los estudiantes valoran estas herramientas para tareas de integración puntual, como la conexión de APIs y

servicios externos, aunque mantienen reservas frente a su uso en decisiones de diseño de alto nivel.

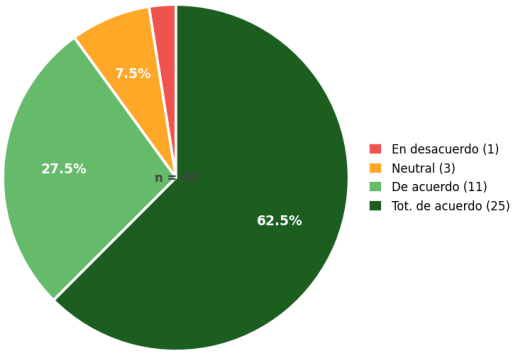

Figura 20. Considero importante revisar manualmente el código generado antes de usarlo en proyectos reales.

Escala Likert 1–5. n = 40 encuestados.

Análisis:

Esta pregunta obtuvo el mayor consenso de toda la encuesta: el 90 % de los estudiantes considera esencial revisar manualmente el código generado antes de implementarlo (27,5 % de acuerdo + 62,5 % totalmente de acuerdo). Este resultado es revelador: aunque los estudiantes adoptan y valoran las herramientas de IA, reconocen de forma casi unánime que la supervisión humana es indispensable para garantizar la calidad del software. Ello tiene implicaciones directas para la formación en ingeniería, donde la incorporación de prácticas de revisión de código generado automáticamente debería integrarse de manera explícita en los planes de estudio.

Conclusión

La inteligencia artificial generativa y las plataformas low-code representan una transformación significativa en las dinámicas contemporáneas del desarrollo de software. Estas tecnologías han introducido nuevas posibilidades de automatización, aceleración y accesibilidad técnica que impactan tanto el ámbito profesional como el académico.

La revisión teórica evidencia que las herramientas de generación de código por IA pueden mejorar la productividad, facilitar el acceso a documentación y reducir tiempos de implementación. Sin embargo, también introducen riesgos relacionados con comprensión superficial del código, dependencia tecnológica y acumulación de deuda técnica.

De igual manera, las plataformas low-code ofrecen ventajas importantes en construcción rápida de aplicaciones y automatización de procesos, aunque pueden generar restricciones arquitectónicas y problemas de interoperabilidad cuando son utilizadas sin criterios técnicos adecuados.

El concepto de vibe coding refleja una nueva forma de interacción entre desarrollador y sistema inteligente, donde parte importante de la implementación técnica se delega a modelos generativos. Esta dinámica transforma la manera en que los futuros profesionales del software comprenden, validan y mantienen los sistemas que construyen.

La investigación propuesta resulta pertinente debido a que actualmente existe poca evidencia contextualizada sobre cómo los estudiantes universitarios perciben el impacto de estas herramientas sobre calidad del software y sostenibilidad arquitectónica en contextos locales.

Se concluye que el uso responsable de herramientas de IA generativa y plataformas low-code requiere fortalecer procesos de validación técnica, revisión crítica y comprensión arquitectónica dentro de la formación universitaria.

Asimismo, resulta necesario promover competencias orientadas no solamente a utilizar herramientas automatizadas, sino también a comprender profundamente sus limitaciones, riesgos y efectos sobre mantenibilidad y evolución del software.

La investigación también permite reconocer que la automatización no elimina la complejidad técnica del desarrollo de software, sino que en muchos casos redistribuye dicha complejidad hacia procesos posteriores de integración, pruebas, mantenimiento y evolución arquitectónica.

Finalmente, se recomienda que futuras investigaciones profundicen en estudios comparativos entre instituciones universitarias, análisis longitudinales sobre evolución de competencias técnicas y evaluación empírica de calidad del software generado mediante herramientas de IA generativa y plataformas low-code.

Referencias

Alamin, M. A. A., Uddin, G., Malakar, S., et al. (2023). Developer discussion topics on the adoption and barriers of low code software development platforms. Empirical Software Engineering, 28, Article 4. https://doi.org/10.1007/s10664-022-10244-0

Alfonso, I., Conrardy, A., et al. (2024). Building BESSER: An open-source low-code platform. International Conference on Business Process Modeling.

Bastidas Díaz, B. A., Cruz Trochez, J. P., & Pardo Calvache, C. J. (2026). La gestión de la deuda de procesos en equipos de desarrollo de software. Revista Colombiana de Tecnologías.

Butler, J., Smith, R., & Thompson, L. (2025). Dear Diary: A randomized controlled trial of Generative AI coding tools in the workplace. Journal of Software Engineering Research.

Correal, D., Pérez-Gutiérrez, B., & Castellanos, J. (2020). A computational model of a natural language processing system for architectural technical debt management. International Journal of Software Engineering.

Departamento Nacional de Planeación. (2025). CONPES 4144: Política nacional de inteligencia artificial. Gobierno de Colombia.

Estevez Gonzalez, J. (2022). Prototipo de aplicación móvil utilizando una herramienta de desarrollo multiplataforma para trabajadores de servicios independientes [Trabajo de grado]. Universidad Autónoma de Bucaramanga.

Käss, A., Albu, M., Schneider, M., & Bork, D. (2022). Low-code platforms and software engineering challenges. Software Quality Journal.

Martínez Ramírez, D., & Cruz Fajardo, J. (2019). Prototipo de software mediante plataformas Low-Code para automatizar el proceso de recepción y emisión de facturación electrónica: caso de estudio Redsis S.A.S. [Trabajo de grado]. Universidad Cooperativa de Colombia.

Meske, C., Richter, A., & Gnewuch, U. (2025). Human-AI collaboration and vibe coding in software development environments. Information Systems Review.

Pérez, B. (2023). Deuda técnica en arquitectura: una estrategia de identificación [Tesis de maestría]. Universidad Francisco de Paula Santander.

Pérez-Gutiérrez, B., Correal, D., & Vera-Rivera, F. (2022). Un enfoque de machine learning para apoyar la identificación de la deuda técnica en arquitectura. Revista Iberoamericana de Ingeniería de Software.

Pinto, G., de Souza, C., & Silva, M. (2024). Developer experiences with a contextualized AI coding assistant: Usability, expectations, and outcomes. Empirical Software Engineering.

Ray, P. P. (2025). Generative AI in software engineering: Opportunities and challenges. ACM Computing Surveys.

Rios, N., Spínola, R., & Seaman, C. (2021). A tertiary study on technical debt: Types, management strategies, and future directions. Journal of Systems and Software.

Vaithilingam, P., et al. (2022). Expectation versus experience: Evaluating the usability of code generation tools powered by large language models. Proceedings of the ACM Conference on Human Factors in Computing Systems.