UNIVERSIDAD NACIONAL DE EDUCACIÓN A DISTANCIA

FACULTAD DE EDUCACIÓN

MÁSTER UNIVERSITARIO EN FORMACIÓN DEL PROFESORADO DE EDUCACIÓN SECUNDARIA OBLIGATORIA Y BACHILLERATO, FORMACIÓN PROFESIONAL Y ENSEÑANZAS DE IDIOMAS

Especialidad de informática

TRABAJO FIN DE MÁSTER

De la pizarra al código asistido: Análisis del impacto pedagógico de la Inteligencia Artificial en la enseñanza de programación web en Formación Profesional

Lidia Contreras Ochando

Dirigido por: Francisco Iniesto Carrasco

CURSO 2024/2025

Agradecimientos

Quiero agradecer al Colegio Sorolla Secundaria-Profesional y a Grupo Sorolla Educación por haberme dado la oportunidad de llevar a cabo este estudio en un centro que siempre ha sido mi casa.

A mi tutor, Francisco, por su orientación y confianza durante todo el proceso.

A mis padres, gracias por el apoyo constante y por no extrañarse de que "me pase la vida estudiando".

Y a mí misma, por no perder la curiosidad ni la ilusión por investigar.

Lidia Contreras Ochando :)
Valencia, septiembre de 2025

Resumen

La integración de la inteligencia artificial generativa en la educación representa uno de los cambios más significativos en las metodologías de enseñanza contemporáneas. Este estudio analiza el impacto pedagógico de los asistentes de IA en la enseñanza de programación web en los ciclos formativos de Sistemas Microinformáticos y Redes (SMR) y Administración de Sistemas Informáticos en Red (ASIR).

Se implementó un diseño cuasi-experimental con grupo de control histórico, comparando el rendimiento académico y las percepciones de 47 estudiantes que utilizaron herramientas de IA generativa durante el curso 2024/2025 con 34 estudiantes formados con metodologías tradicionales en 2023/2024. La intervención incorporó formación en ingeniería de prompts, depuración asistida y desarrollo de proyectos integrados en los módulos de programación web (HTML, CSS, JavaScript, PHP).

Los resultados muestran una mejora moderada en el rendimiento académico, con desplazamiento de la distribución de calificaciones hacia tramos superiores y desaparición de notas inferiores a 6. El análisis cualitativo revela la mayor parte del alumnado percibe que la IA facilita la comprensión de la programación, desarrollando competencias de formulación intencional, iteración con mejora y verificación crítica. Sin embargo, se identifica una autoeficacia reducida al programar sin asistencia y limitaciones en la autoexplicación técnica.

La investigación concluye que la integración estructurada de asistentes de IA reconfigura el aprendizaje hacia la orquestación del proceso de programación más que hacia la producción de código, requiriendo adaptaciones metodológicas que equilibren la asistencia tecnológica con el desarrollo de la autonomía conceptual.

Palabras clave: inteligencia artificial generativa, formación profesional, programación web, metodologías pedagógicas, ingeniería de prompts

Abstract

The integration of generative artificial intelligence in education represents one of the most significant changes in contemporary teaching methodologies. This study analyzes the pedagogical impact of AI assistants in web programming education within the vocational training cycles of Microcomputer Systems and Networks (SMR) and Network Computer Systems Administration (ASIR).

A quasi-experimental design with historical control group was implemented, comparing the academic performance and perceptions of 47 students who used generative AI tools during the 2024/2025 academic year with 34 students trained with traditional methodologies in 2023/2024. The intervention incorporated prompt engineering training, assisted debugging, and integrated project development in web programming modules (HTML, CSS, JavaScript, PHP).

Results show moderate improvement in academic performance, with a shift in grade distribution toward higher ranges and disappearance of grades below 6. Qualitative analysis reveals that the students perceive that AI facilitates programming comprehension, developing competencies in intentional formulation, iterative improvement, and critical verification. However, reduced self-efficacy when programming without assistance and limitations in technical self-explanation are identified.

The research concludes that structured integration of AI assistants reconfigures learning toward orchestrating the programming process rather than code production, requiring methodological adaptations that balance technological assistance with the development of conceptual autonomy.

Keywords: generative artificial intelligence, vocational training, web programming, pedagogical methodologies, prompt engineering

Glosario

ABP (Aprendizaje Basado en Proyectos) Metodología didáctica centrada en la realización de proyectos integrados como medio para adquirir conocimientos.

Algoritmo Secuencia finita de pasos que resuelve un problema o realiza una tarea en programación.

API (Interfaz de Programación de Aplicaciones) Conjunto de funciones y procedimientos que permite la interacción entre distintos programas.

Autocompletado Funcionalidad que permite predecir y completar fragmentos de código automáticamente en el editor.

Blackbox AI Asistente de IA para generación y depuración de código, disponible como extensión y herramienta web.

ChatGPT Asistente conversacional basado en modelos LLM, capaz de generar respuestas y código a partir de indicaciones en lenguaje natural.

Codificación / Refactorización Escritura de código fuente para desarrollar programas; la refactorización implica modificar dicho código para mejorar su calidad sin alterar su funcionalidad.

CSS Hoja de estilos en cascada utilizada para definir la presentación visual de los elementos HTML en una web.

Depuración Proceso de detección y corrección de errores en el código de un programa.

DOM (Document Object Model) Representación estructurada de los elementos de una página web, que permite manipulación dinámica mediante programación.

DTD (Document Type Definition) Especificación que define la estructura y reglas de un documento XML.

GitHub Copilot Herramienta de autocompletado inteligente basada en IA para código, integrada en varios editores y entornos de desarrollo.

HTML Lenguaje de marcado utilizado para estructurar páginas web.

IA Tecnología capaz de simular procesos cognitivos humanos, como aprendizaje y toma de decisiones, mediante algoritmos informáticos.

IA generativa Rama de la IA que puede crear contenido nuevo (texto, imágenes, código) a partir de patrones aprendidos en grandes conjuntos de datos.

IDE (Entorno de Desarrollo Integrado) Aplicación que combina editor de código, depurador y otras utilidades para facilitar la programación.

JavaScript Lenguaje de programación que permite agregar interactividad y lógica en el lado cliente de una página web.

Keyframes Concepto de CSS y animación que permite definir los puntos clave de una transición o animación en elementos web.

LLM (Large Language Model) Modelo de lenguaje de gran escala, entrenado con grandes volúmenes de texto para generar y comprender lenguaje natural y código.

Modelo Transformer Arquitectura de redes neuronales profundas especialmente eficiente en procesamiento de lenguaje natural y generación de código.

Pair programming Técnica de desarrollo en la que dos programadores trabajan juntos en una misma estación para producir código más eficiente y con menos errores.

PHP Lenguaje de programación de propósito general, usado principalmente en el desarrollo de aplicaciones web y en el lado servidor.

Prompt / Ingeniería de prompts Instrucción en lenguaje natural que se proporciona a una IA generativa para guiar la creación de contenido, incluyendo código; la ingeniería de prompts optimiza su formulación para obtener mejores resultados.

Responsive Capacidad de una página web para adaptarse a distintos tamaños y dispositivos, garantizando una experiencia de usuario óptima.

SQL / MySQL Lenguaje de consulta estructurado y un sistema de gestión de bases de datos relacional muy usado en aplicaciones web.

Usabilidad Medida que evalúa la facilidad de uso y aprendizaje de una herramienta, aplicación o interfaz.

Índice general

Índice de tablas

Capítulo 1

Introducción

La programación ha sido una disciplina fundamental en el desarrollo tecnológico desde mediados del siglo XX, evolucionando paralelamente al avance de la informática. En las primeras décadas, programar significaba trabajar directamente con lenguajes de bajo nivel y hardware rudimentario y el conocimiento se transmitía de forma artesanal entre pioneros [1]. En estos primeros tiempos, esta tarea estaba reservada principalmente a especialistas con formación matemática avanzada que interactuaban con los ordenadores utilizando lenguajes máquina y ensamblador [2, 3]. Con la aparición de lenguajes de alto nivel, como FORTRAN, COBOL, C, Pascal o Java, la programación se hizo más accesible, entrando así en el ámbito académico formal [4, 5, 6, 7]. Este cambio generó una necesidad creciente de desarrollar metodologías efectivas para enseñar esta habilidad [1].

Tradicionalmente, la enseñanza de la programación ha seguido un enfoque basado en la sintaxis y la resolución de problemas. Los estudiantes aprenden primero los fundamentos de un lenguaje específico: variables, estructuras de control, funciones y objetos, aplicando luego estos conocimientos a la resolución progresiva de problemas cada vez más complejos.

Esta manera tradicional de enseñar a programar se ha centrado típicamente en clases teóricas acompañadas de sesiones prácticas, donde los estudiantes deben escribir código desde cero para resolver ejercicios algorítmicos clásicos. Este enfoque presenta limitaciones importantes, ampliamente documentadas en la literatura especializada [8, 9] y muchos principiantes se enfrentan a obstáculos cognitivos significativos, incluyendo dificultades para interiorizar conceptos abstractos fundamentales. Además, la sintaxis rígida de los lenguajes provoca frustración, generando desaliento y desmotivación ([10, 11]).

En el ámbito específico de la programación web, los desafíos inherentes al aprendizaje se amplifican considerablemente debido a la necesidad de dominar múltiples lenguajes y tecnologías que interactúan entre sí. El desarrollo web involucra principalmente HTML, CSS, JavaScript y PHP, que deben integrarse para construir una aplicación completa. En este contexto multiparadigma, un principiante no solo debe dominar la sintaxis individual de cada lenguaje, sino también entender cómo estos interactúan para lograr un resultado coherente: el HTML define la estructura del contenido, el CSS la presentación visual, JavaScript

proporciona la lógica interactiva del lado del cliente y PHP permite interacción con el servidor [12]. Además, la naturaleza visual inherente al desarrollo web añade una capa adicional de complejidad, ya que los estudiantes deben no solo lograr que su código funcione correctamente desde el punto de vista lógico, sino también que produzca los resultados visuales deseados [13]. La metodología tradicional de enseñanza de la programación puede quedarse corta en módulos específicos de programación web, donde la carga cognitiva se incrementa notablemente al exigir a los estudiantes asimilar simultáneamente conceptos clásicos de programación junto con nociones específicas del desarrollo web, elevando considerablemente el riesgo de confusión y frustración en el aprendizaje [14].

Para abordar estas limitaciones en los métodos tradicionales de enseñanza de la programación, se han explorado diversas estrategias y desarrollado herramientas destinadas a simplificar el proceso mediante la automatización parcial o total de tareas complejas. Inicialmente, estas herramientas tomaban formas relativamente simples, como editores con resaltado de sintaxis, sistemas de autocompletado de código y entornos de desarrollo integrados (IDEs) con retroalimentación inmediata sobre errores [15, 16]. Sin embargo, con el avance significativo del campo de la inteligencia artificial, estas herramientas han evolucionado hacia enfoques más ambiciosos que permiten automatizar tareas que anteriormente requerían intervención manual constante.

El campo de la programación inductiva (Inductive Programming, IP) - un área de la Inteligencia Artificial (IA) - estudia cómo automatizar tareas de programación rutinarias y repetitivas [17, 18], permitiendo a los usuarios centrarse en problemas de mayor complejidad [19] . Una de las líneas más relevantes dentro de este contexto es la programación por ejemplos (Programming by Example, PBE), técnica en la cual el usuario proporciona ejemplos concretos de entradas y salidas deseadas, y el sistema sintetiza automáticamente programas que cumplen esas especificaciones. Popularizada por [20] con su trabajo "Your Wish is My Command: Programming by Example", esta metodología ha evolucionado considerablemente, integrando técnicas avanzadas de aprendizaje automático para generar soluciones más sofisticadas y robustas.

Estos avances en la automatización de tareas de programación han preparado el terreno para la actual revolución impulsada por los grandes modelos de lenguaje. La aparición de los modelos generativos de gran escala, especialmente la familia GPT (Generative Pre-trained Transformer), representa un salto cualitativo significativo en la capacidad de las máquinas de generar tanto lenguaje natural como código de programación [21]. A diferencia de los sistemas anteriores, diseñados generalmente para resolver tareas específicas, estos modelos han sido entrenados con vastas cantidades de texto y código fuente, lo que les permite generalizar y adaptarse a una amplia gama de lenguajes de programación y contextos complejos [22, 23], produciendo soluciones sofisticadas que abarcan no solo la escritura de código, si no también tareas como la depuración, optimización, traducción entre lenguajes, refactorización y detección de vulnerabilidades de seguridad. Gracias a este tipo de modelos, las herramientas

basadas en IA han comenzado a integrarse rápidamente en el flujo de trabajo cotidiano de la industria [24, 25]. Además, este tipo de asistencia se ha convertido rápidamente en uno de los recursos más utilizados en los entornos integrados de desarrollo modernos, acelerando significativamente tareas repetitivas o rutinarias y redefiniendo de forma profunda la manera en que se concibe el acto de programar [26, 27, 28].

Como hemos visto, el contraste entre la manera tradicional de programar y el nuevo paradigma asistido por IA es notable. Tradicionalmente, los programadores debían memorizar sintaxis, buscar manualmente en documentación y depurar código línea por línea. El proceso de aprendizaje era arduo, con una curva de aprendizaje pronunciada [29]. En contraposición, los entornos modernos asistidos por IA permiten a los programadores expresar su intención en lenguaje natural, recibir sugerencias de código relevantes, identificar automáticamente errores y obtener explicaciones detalladas. Esta transformación reduce significativamente la carga cognitiva asociada con los aspectos mecánicos de la programación, permitiendo a los desarrolladores centrarse en la lógica de alto nivel y la resolución de problemas [21]. Para los estudiantes de informática, estos asistentes pueden actuar como tutores personalizados, proporcionando retroalimentación inmediata y adaptandose a su nivel de comprensión.

No obstante, esta transición también implica retos pedagógicos considerables. Algunos educadores advierten que el uso indiscriminado de asistentes inteligentes en contextos educativos podría llevar a los estudiantes a completar tareas de programación sin comprender plenamente el código generado, diluyendo así la adquisición de habilidades fundamentales [30, 31]. Este fenómeno obligaría a repensar profundamente la enseñanza de la programación, creando la necesidad urgente de equilibrar cuidadosamente las metodologías tradicionales con las ventajas prácticas que ofrece la inteligencia artificial [32]. El director de la Escuela Técnica Superior de Ingeniería Informática de la UPV, afirmó en un artículo que hoy la IA ha dado “el salto de los congresos científicos a nuestras conversaciones cotidianas” subrayando que la IA “ya no es tendencia, es parte de la rutina” y que se debe formar a profesionales que comprendan tanto sus fundamentos técnicos como sus implicaciones éticas y sociales [33].

La integración de la inteligencia artificial en la enseñanza de programación no implica simplemente adoptar una nueva herramienta tecnológica, sino que representa un replanteamiento profundo sobre qué significa enseñar y aprender programación en la era actual. Este trabajo fin de máster se sitúa precisamente en esta intersección, explorando cómo implementar eficazmente este nuevo paradigma en la formación profesional, aprovechando las ventajas pedagógicas y prácticas que ofrece la IA, al tiempo que aborda críticamente los desafíos educativos y metodológicos que conlleva.

1.1. Motivación

En el contexto de la formación profesional, especialmente en los ciclos de Sistemas Microinformáticos y Redes (SMR) y Administración de Sistemas Informáticos en Red (ASIR), se

observa de manera recurrente una serie de dificultades que afectan tanto al aprendizaje como a la motivación del alumnado. Entre los problemas más destacados se encuentran la baja motivación inicial y la ansiedad ante la complejidad de los lenguajes de programación. Estas dificultades se ven acentuadas por la necesidad de integrar diferentes lenguajes y tecnologías en proyectos web funcionales: En el caso específico del ciclo de SMR, la integración de lenguajes como JavaScript con tecnologías básicas como HTML y CSS suele representar un reto considerable; Asimismo, en el ciclo de ASIR, los estudiantes deben dominar la programación en PHP junto con la gestión de bases de datos MySQL y la integración con tecnologías de desarrollo web. La combinación de múltiples tecnologías incrementa la carga cognitiva y la percepción de dificultad por parte de los estudiantes.

La incidencia de estos problemas no solo repercute en el rendimiento académico, sino que afecta directamente a la empleabilidad y a la preparación de los futuros profesionales. El mercado laboral tecnológico actual, caracterizado por una rápida adopción de herramientas basadas en inteligencia artificial generativa, demanda perfiles que dominen no solo los fundamentos clásicos de la programación, sino también el uso eficiente de asistentes inteligentes y modelos de lenguaje avanzados. Esta tendencia se refleja en la reciente afirmación de Salvador Más, CEO de GPTAdvisor, durante el Devfest celebrado en la Universitat Politècnica de València en 2024: "Ya no contratamos a ningún programador junior que no sea experto en ChatGPT, porque nos ralentiza" [34]. Esta afirmación refleja con claridad las expectativas actuales del mercado laboral tecnológico, en el que el dominio de herramientas basadas en inteligencia artificial generativa, como ChatGPT, ha pasado a ser considerado esencial. Por tanto, un titulado sin experiencia en estas herramientas podría ser percibido como menos competitivo, retrasando potencialmente al equipo profesional al que se incorpore.

Ante este escenario, resulta evidente que mantener enfoques pedagógicos tradicionales, que excluyen o minimizan el uso de inteligencia artificial generativa, supone dejar en desventaja a los estudiantes frente a las exigencias reales del sector. Carece de sentido mantener al alumnado alejado de estas tecnologías en su formación inicial. Al contrario, la incorporación de asistentes de inteligencia artificial generativa como ChatGPT podría aportar importantes beneficios educativos, mejorando tanto el aprendizaje como la motivación del estudiantado. De hecho, la experiencia cotidiana en el aula revela que muchos estudiantes ya recurren de forma autónoma a estas herramientas para resolver ejercicios y superar los obstáculos que encuentran en el aprendizaje. Sin embargo, la utilización no guiada de estas herramientas puede derivar en dependencia y en la falta de comprensión profunda de los conceptos fundamentales.

La introducción de asistentes de IA generativa en el proceso formativo, de manera estructurada y supervisada, se presenta como una oportunidad para transformar estos retos en ventajas pedagógicas. Por un lado, puede contribuir a reducir la ansiedad y la frustración asociadas al aprendizaje inicial de la programación, facilitando la comprensión y la integración de los distintos lenguajes y tecnologías. Por otro, permite alinear la formación

profesional con las demandas actuales del mercado laboral, dotando al alumnado de competencias híbridas que combinan el dominio técnico tradicional con la capacidad de interactuar eficazmente con herramientas inteligentes. En este sentido, la motivación principal de este trabajo radica en explorar y fundamentar la integración de la inteligencia artificial generativa en la enseñanza de la programación web en formación profesional, con el objetivo de mejorar tanto la experiencia de aprendizaje como la empleabilidad de los futuros titulados.

1.2. Justificación

La presente investigación se justifica por varios factores convergentes que señalan la necesidad de explorar y fundamentar la integración de la inteligencia artificial generativa en la enseñanza de la programación web dentro de la formación profesional, específicamente en los ciclos de SMR y ASIR.

En primer lugar, existe una carencia significativa de estudios científicos que aborden la aplicación real y los efectos de estas herramientas en el contexto específico de la formación profesional en España. Si bien la literatura académica recoge un creciente interés por el uso de la IA en la educación y en la enseñanza de lenguajes de programación, la mayoría de los trabajos se centran en niveles educativos superiores, dejando un vacío importante en el conocimiento sobre su impacto en los ciclos formativos. Diversos autores subrayan que la FP es una etapa escasamente investigada en general, particularmente en lo referente a la aplicación de tecnologías digitales avanzadas en el aula. La literatura existente sugiere incluso que los estudiantes de FP presentan, en promedio, menores niveles de competencia digital en comparación con alumnos de etapas formativas superiores, empleando la tecnología más para ocio personal que con fines académicos [35, 36].

Dada la escasez de estudios específicos sobre el uso de IA generativa en la formación profesional, resulta imprescindible investigarlo científicamente. Una justificación central de este trabajo es precisamente contribuir a llenar ese vacío empírico, aportando datos y análisis sobre cómo asistentes de IA pueden integrarse pedagógicamente en asignaturas de programación web de SMR y ASIR.

En segundo lugar, este estudio se alinea con las directrices y prioridades establecidas por las instituciones educativas y los organismos de evaluación de la calidad docente, que promueven la innovación pedagógica y la actualización de las metodologías didácticas para adaptarlas a las necesidades del siglo XXI. De hecho, la nueva legislación de FP insiste en la necesidad de promover la innovación e investigación aplicada en los centros, garantizando la incorporación de las nuevas tecnologías derivadas de la transformación digital. La integración de la inteligencia artificial generativa en la enseñanza de la programación web representa una oportunidad para transformar la práctica docente y mejorar la experiencia de aprendizaje del alumnado, desarrollando competencias clave para su futuro profesional [37].

Por último, hay que considerar que la digitalización educativa debe ir acompañada de

cambios pedagógicos profundos; no basta con dotar de tecnología, sino con redefinir el rol del docente y del estudiante en entornos enriquecidos con IA. La inteligencia artificial generativa no solo ofrece nuevas herramientas para facilitar el aprendizaje de la programación, sino que también plantea desafíos fundamentales sobre la naturaleza de la creatividad, la autoría y la evaluación en la era digital. Al abordar estas cuestiones desde una perspectiva crítica y fundamentada, este estudio puede contribuir a enriquecer el debate académico y a orientar las políticas educativas en el ámbito de la formación profesional.

1.3. Preguntas de investigación y objetivos

La integración de la inteligencia artificial generativa en la enseñanza de la programación web plantea interrogantes fundamentales sobre su impacto real en el aprendizaje y en el desarrollo de competencias técnicas en los ciclos de formación profesional. Aunque la IA promete transformar la educación y facilitar el proceso de enseñanza-aprendizaje, persisten dudas sobre su influencia en el rendimiento académico, la motivación estudiantil y la adquisición de habilidades esenciales para el futuro profesional del alumnado. Además, existe incertidumbre acerca de cómo equilibrar el uso de estas herramientas con los métodos tradicionales, garantizando el cumplimiento de los objetivos curriculares.

Con estos desafíos en mente, este trabajo se propone responder a las siguientes preguntas de investigación:

- ¿Cómo afecta la integración de IA generativa al rendimiento académico en programación web comparado con métodos tradicionales en FP?
- ¿Qué factores moderan la relación entre el uso de IA y el desarrollo de competencias técnicas en ciclos SMR/ASIR?
- ¿Qué estrategias pedagógicas optimizan la integración IA-tradicional en módulos de desarrollo web?

El objetivo general de esta investigación es evaluar la efectividad pedagógica de la integración de la inteligencia artificial generativa en la enseñanza de la programación web en los ciclos de formación profesional de SMR y ASIR.

Para alcanzar este objetivo general, se plantean los siguientes objetivos específicos:

- **O1: Analizar el impacto en el rendimiento académico**
    1. Comparar el rendimiento académico de los estudiantes que utilizan herramientas de IA generativa con el de aquellos formados con metodologías tradicionales.
- **O2: Conocer la percepción estudiantil**

1. Recoger opiniones y valoraciones sobre la utilidad, la facilidad de uso y la eficacia percibida de las herramientas de IA generativa en el aprendizaje de la programación web.
2. Identificar los aspectos positivos y negativos del uso de la IA desde la perspectiva del alumnado, incluida la preferencia entre métodos tradicionales, uso de IA o combinación de ambos.

- **O3: Establecer las competencias desarrolladas**

1. Determinar las competencias específicas (resolución de problemas, creatividad, pensamiento crítico) que los estudiantes adquieren al emplear asistentes de IA generativa.
2. Valorar si la IA facilita la adquisición de habilidades transversales clave para su futura empleabilidad.

- **O4: Analizar patrones de uso y habilidades técnicas asociadas**

1. Identificar las herramientas de IA generativa más utilizadas y la frecuencia de uso.
2. Evaluar el progreso en la redacción de *prompts* estructurados y la percepción de dificultad al elaborarlos.
3. Analizar las estrategias de corrección de código generado por la IA y la confianza percibida al programar sin asistencia.

- **O5: Evaluar el cambio en el rol del docente**

1. Analizar cómo la integración de IA generativa modifica las estrategias pedagógicas y las tareas del profesorado en el aula de programación web.
2. Identificar las nuevas competencias y habilidades docentes necesarias para utilizar de forma eficaz y ética estas herramientas.

- **O6: Proponer recomendaciones metodológicas**

1. Diseñar pautas y buenas prácticas para la implementación responsable de asistentes de IA generativa en ciclos formativos de programación web.
2. Formular orientaciones para integrar la IA en el currículo, garantizando el equilibrio entre apoyo tecnológico y desarrollo de la autonomía del estudiante.

1.4. Referencia al ámbito de esta especialidad y contexto educativo

El presente trabajo se enmarca en el *Máster Universitario en Formación del Profesorado de Educación Secundaria Obligatoria y Bachillerato, Formación Profesional y Enseñanzas*

de Idiomas, con especialidad en Informática, tal y como recoge la Memoria de Verificación del título de la UNED[1]. Este máster tiene entre sus competencias específicas la "innovación docente e iniciación a la investigación educativa" (CE6), que incluye la aplicación de propuestas docentes innovadoras, el análisis crítico de buenas prácticas y el diseño de proyectos de investigación e innovación en el ámbito educativo.

Asimismo, el programa formativo promueve el desarrollo de competencias transversales como la capacidad de "gestionar y planificar la actividad profesional" (CT1), "utilizar de forma eficaz y sostenible las herramientas y recursos de la sociedad del conocimiento" (CT5) y "desarrollar actitudes éticas y de compromiso social" (CT7). Estas competencias son fundamentales para diseñar e implementar metodologías didácticas basadas en tecnologías emergentes, como los asistentes de IA generativa, garantizando un uso responsable y orientado al aprendizaje significativo.

En el ámbito docente, el máster define como competencia general (CG2) la "planificación, desarrollo y evaluación del proceso de enseñanza y aprendizaje", atendiendo al nivel y formación previa de los estudiantes. La investigación propuesta contribuye directamente a esta competencia, al explorar cómo integrar de manera efectiva asistentes de IA en el currículo de los ciclos formativos SMR y ASIR, adaptando las estrategias pedagógicas a las necesidades del alumnado y a la realidad profesional actual.

Finalmente, el máster enfatiza la importancia de las metodologías de "investigación y evaluación educativas" (CE6.4), que permite al futuro profesorado diseñar, implementar y analizar proyectos de innovación pedagógica. Este TFM aplica de manera práctica dichas metodologías al contexto de la programación web en FP, generando datos empíricos sobre el impacto de la IA generativa en el aprendizaje, la motivación y la empleabilidad de los estudiantes. De este modo, la investigación no solo refuerza las competencias tecnológicas y didácticas adquiridas en el máster, sino que también aporta conocimiento aplicable a la mejora continua de las prácticas docentes en FP.

1.5. Organización del documento

La organización del presente documento sigue una estructura que garantiza la comprensión del estudio realizado. El documento se inicia con una introducción en el Capítulo 1, en la que se exponen la motivación, la justificación y los objetivos fundamentales de la investigación, así como las preguntas de investigación que guían el trabajo. A continuación, se contextualiza el ámbito de la especialidad y el entorno educativo en el que se desarrolla el estudio, situando el trabajo dentro del Máster Universitario en Formación del Profesorado

[1]MÁSTER UNIVERSITARIO EN FORMACIÓN DEL PROFESORADO DE EDUCACIÓN SECUNDARIA OBLIGATORIA Y BACHILLERATO, FORMACIÓN PROFESIONAL Y ENSEÑANZAS DE IDIOMAS (Documentación oficial): `https://www.uned.es/universidad/inicio/estudios/masteres/master-universitario-en-formacion-del-profesorado-de-educacion-secundaria-obligatoria-y-bachillerato-formacion-profesional-y-ensenanzas-de-idiomas.html?idContenido=11`

de la UNED y en el contexto concreto de la formación profesional informática.

El Capítulo 2 está dedicado al estado del arte. En esta sección se analizan los antecedentes teóricos y empíricos sobre la inteligencia artificial generativa, con especial atención a las características y evolución de los asistentes de IA en programación, así como los principales resultados de investigaciones previas en el ámbito educativo y sus vacíos respecto a la formación profesional.

El Capítulo 3 describe la metodología empleada en el estudio, detallando el diseño de investigación, el contexto institucional, los participantes y la muestra, el calendario y desarrollo de la intervención, los recursos utilizados, el procedimiento didáctico, las variables e indicadores de análisis, y las técnicas estadísticas y de análisis cualitativo aplicadas.

En el Capítulo 4 se presentan los resultados empíricos obtenidos, estructurados en diferentes apartados según las variables de estudio. Se describen la muestra, la evolución del rendimiento académico, los contrastes entre cohortes, y el análisis de las percepciones estudiantiles recogidas a través de cuestionarios. De igual modo, se analiza en profundidad la evidencia recogida a partir de la resolución de ejercicios prácticos, distinguiendo entre tareas de ingeniería de prompts y actividades de depuración asistida con IA.

La discusión de los hallazgos se desarrolla en el Capítulo 5. En esta sección se sintetizan los resultados, se interpretan desde una perspectiva pedagógica y metodológica, y se discuten tanto las aportaciones como las principales limitaciones del estudio. Se plantean reflexiones sobre los cambios necesarios en la enseñanza de la programación web asistida por IA, alineadas con las competencias profesionales demandadas por el sector y la experiencia observada en el aula.

Finalmente, en el capítulo6 se exponen las conclusiones del trabajo, resaltando las implicaciones para la formación profesional, la contribución al conocimiento científico y las líneas futuras de investigación. El documento concluye con la relación completa de bibliografía y fuentes utilizadas, así como con los anexos donde se incluyen el cuestionario empleado (Anexo A), las respuestas de los estudiantes a las preguntas abiertas (Anexo B) y los enunciados de ejercicios prácticos empleados durante la intervención (Anexos C y D).

Capítulo 2

Estado del Arte

La irrupción de nuevas tecnologías de inteligencia artificial capaces de generar contenido, conocidas como IA generativa, está teniendo un gran impacto en muchos campos. En educación y, más concretamente, en la enseñanza de la programación web, estas herramientas prometen transformar las metodologías tradicionales.

A continuación, se ofrece una contextualización general sobre qué es la IA generativa, por qué ha ganado protagonismo en el ámbito educativo reciente, cuál es su potencial como herramienta didáctica en programación y de qué manera podría aplicarse en entornos de FP como los ciclos de SMR y ASIR.

2.1. Inteligencia artificial e IA generativa

2.1.1. Definición y evolución histórica de la IA

La inteligencia artificial (IA) se define en la literatura científica como la simulación de capacidades cognitivas humanas por sistemas informáticos, abarcando procesos como percepción, razonamiento, aprendizaje y planificación [38]. Una definición ampliamente aceptada en el ámbito académico sostiene que la IA es la capacidad de un sistema para interpretar datos externos, aprender de ellos y emplear ese aprendizaje para alcanzar objetivos específicos mediante una adaptación flexible, lo que implica procesos autónomos de aprendizaje y toma de decisiones. Según [39], una definición operacionalmente útil de IA debe reflejar la adaptación en contextos de conocimiento y recursos insuficientes, enfatizando la capacidad de los sistemas para operar en condiciones inciertas y dinámicas.

El origen formal de la IA como disciplina científica se remonta a la década de 1950, destacando el artículo de Alan Turing, *"Computing Machinery and Intelligence"* [40], donde se pregunta si las máquinas pueden "aprender" y plantea el famoso *Test de Turing* como criterio para evaluar la inteligencia de las máquinas. El término "inteligencia artificial" fue acuñado por John McCarthy en la conferencia de Dartmouth en 1956, considerada el punto fundacional del campo [41]. En sus primeras décadas, la IA estuvo marcada por un entusiasmo

considerable debido a los logros iniciales en razonamiento automatizado y resolución de problemas de manera autónoma[38]. Arthur Samuel es ampliamente reconocido por desarrollar el primer programa de ordenador capaz de aprender, en 1952, con su software para jugar a las damas, que mejoraba su desempeño con la experiencia. Samuel también acuñó el término *machine learning* (aprendizaje automático) en 1959 [42, 43]. En 1958, Frank Rosenblatt introdujo una de las primeras redes neuronales artificiales capaces de aprender mediante el desarrollo del perceptrón [44].

Sin embargo, las limitaciones computacionales y teóricas condujeron a periodos de estancamiento conocidos como "inviernos de la IA", caracterizados por una disminución significativa en la financiación y el interés, especialmente en los años 1970 y finales de los 80 [45]. Estos ciclos de expectativas y decepciones han sido objeto de análisis en la literatura, destacando la importancia de abordar la IA como una tecnología sujeta a limitaciones inherentes y no como una panacea tecnológica.

Desde principios del siglo XXI, y especialmente en la última década, la IA ha experimentado un resurgimiento impulsado por el aumento exponencial del poder de cómputo, la disponibilidad de grandes volúmenes de datos y el desarrollo de nuevas metodologías de aprendizaje automático y redes neuronales profundas (*deep learning*) [46]. Los avances en arquitecturas de redes neuronales, como los *transformers*, han permitido superar barreras previas, logrando desempeños superhumanos en tareas complejas como la visión por computador y el procesamiento de lenguaje natural [47].

Un desarrollo particularmente relevante es la aparición de los grandes modelos de lenguaje (LLMs), que han revolucionado la generación y comprensión automática de texto, alcanzando niveles de coherencia y contextualización sin precedentes [48]. Estos modelos, basados en arquitecturas de aprendizaje profundo, han demostrado su utilidad en tareas avanzadas de investigación científica, redacción, revisión y planificación experimental. Además, el paradigma de la IA generativa ha abierto nuevas aplicaciones en la creación de entornos virtuales interactivos, generación de imágenes y simulaciones complejas [49].

La IA que conocemos ahora se caracteriza por sistemas capaces de aprender patrones complejos de manera autónoma, adaptarse a entornos dinámicos y aproximarse cada vez más al comportamiento inteligente humano, lo que ha impulsado una nueva era de aplicaciones innovadoras en múltiples dominios científicos y tecnológicos.

2.1.2. ¿Qué es la IA generativa? Características y ejemplos

La IA generativa (*Generative AI*, IAGen) es una rama de la inteligencia artificial basada en modelos probabilísticos que aprenden distribuciones subyacentes de conjuntos de datos para sintetizar nuevas instancias mediante muestreo estocástico controlado. Formalmente, estos modelos optimizan una función objetivo que maximiza la verosimilitud logarítmica de los datos de entrenamiento. Es decir, es un paradigma enfocado en crear contenido nuevo a

partir de patrones aprendidos de datos existentes. Este área de la IA engloba arquitecturas como los *Transformers* [47], los *GANs* (Generative Adversarial Networks) [50] y los *VAEs* (Variational Autoencoders) [51], cada una con mecanismos específicos para aproximar la función objetivo. En lugar de limitarse a reconocer patrones o tomar decisiones basadas en datos, como hacen otros sistemas de IA, los modelos generativos producen resultados inéditos: textos nunca antes escritos, imágenes jamás dibujadas, código no programado previamente, etc., manteniendo la coherencia y estilo propios del conjunto de entrenamiento.

Las características principales de la IA generativa incluyen su capacidad de generalización (puede combinar conceptos aprendidos de forma novedosa) y su dependencia de un aprendizaje no supervisado o auto-supervisado a gran escala. Un elemento clave es que estos modelos no operan con reglas explícitas pre-programadas, sino que aprenden tras exponerse a enormes corpus de datos (texto, imágenes, audio, código, etc.) y ajustan millones o billones de parámetros internos hasta capturar las regularidades del lenguaje o de las imágenes, por lo que luego pueden sintetizar nuevas instancias que parecen creadas por humanos [52]. Por ejemplo, un modelo generativo de imágenes entrenado con millones de fotografías de paisajes puede generar una imagen completamente nueva de un paisaje imaginario pero realista, combinando montañas, ríos y cielos de manera coherente.

Existen ya numerosos modelos representativos de IA generativa que ilustran sus capacidades en distintos dominios. En el campo del lenguaje natural, destacan los modelos de la familia GPT (Generative Pre-trained Transformer) de OpenAI. Son modelos de lenguaje de última generación capaces de producir texto con sorprendente fluidez, utilizados en aplicaciones como ChatGPT para mantener conversaciones, responder preguntas o redactar textos variados. Estos modelos de lenguaje, entrenados con conjuntos masivos de textos, pueden responder a prácticamente cualquier consulta con un texto articulado, imitando estilos de escritura diversos.

2.2. Asistentes de IA para programación

2.2.1. Definición y funcionalidad general

Los asistentes de programación basados en IA son herramientas de software que actúan como *"pair programming"*[1] virtuales, empleando modelos de lenguaje avanzados para ayudar a escribir código de forma más rápida y con menos esfuerzo. Estas herramientas analizan el contexto del código y las indicaciones en lenguaje natural proporcionadas por el desarrollador, y generan sugerencias o bloques de código relevantes mediante técnicas de procesamiento de lenguaje natural entrenadas en enormes corpus de código fuente [53]. Por ejemplo, un asistente puede completar automáticamente una línea de código o incluso funciones enteras a

[1] *Pair programming*: Técnica de desarrollo de software en la que dos programadores trabajan juntos en una misma estación de trabajo

partir de una descripción en español o inglés de lo que se desea programar. Además de autocompletar código, muchos asistentes pueden explicar fragmentos de código en lenguaje natural, detectar errores comunes o sugerir correcciones, e incluso generar comentarios o documentación automáticamente [54].

Es importante destacar que el desempeño de un asistente de IA depende tanto de la calidad de su modelo subyacente como de las indicaciones (*prompts*[2]) que reciba. Estudios recientes resaltan que lograr resultados útiles a menudo requiere formular correctamente las instrucciones: "el generador de código produce resultados de alta calidad, pero esto depende de la calidad del mensaje del *prompt*" [54], de modo que el usuario debe aprender a interactuar de forma efectiva con el asistente. En cualquier caso, los asistentes no garantizan exactitud absoluta: al basarse en patrones estadísticos, pueden sugerir código erróneo, inseguro o subóptimo. Por ello, el rol del desarrollador sigue siendo revisar críticamente y probar el código generado. A pesar de estas limitaciones, la adopción de asistentes de IA en programación ha crecido de forma exponencial. Tan solo un año después de su lanzamiento, GitHub Copilot fue activado por más de un millón de desarrolladores y generó más de tres mil millones de líneas de código aceptadas en proyectos reales [55].

2.2.2. Herramientas destacadas: comparativa de asistentes de programación

En la actualidad existe un ecosistema diverso de asistentes de programación basados en IA, ofrecidos tanto por grandes compañías tecnológicas como por startups especializadas. A continuación, se presenta una revisión comparativa de algunas de las herramientas más destacadas, describiendo sus características principales y diferencias relevantes[3]:

- **GitHub Copilot:** Es uno de los primeros asistentes de IA integrados en entornos de desarrollo. Desarrollado por GitHub en colaboración con OpenAI y Microsoft, Copilot funciona como una extensión que se conecta a editores populares (Visual Studio Code, JetBrains, etc.) para proveer sugerencias de código en tiempo real [56]. Se basa en el modelo Codex (derivado de GPT-3) y está entrenado con una vasta cantidad de código público de GitHub. Copilot sobresale en la autocompleción inteligente: el desarrollador puede escribir un comentario descriptivo (ejemplo: "función que calcule números primos") y Copilot sugerirá directamente la implementación en el lenguaje elegido.

[2] *Prompt*: Un prompt en el contexto de la inteligencia artificial es una instrucción, conjunto de instrucciones o texto inicial que se proporciona a un modelo generativo para guiar la generación de respuestas o contenido específico.

[3] La lista de herramientas de inteligencia artificial presentada en este trabajo es representativa en el momento de su redacción. Sin embargo, el panorama de la IA evoluciona a gran velocidad, por lo que es probable que algunas herramientas mencionadas queden obsoletas o sean reemplazadas por nuevas soluciones en un corto periodo de tiempo. Se recomienda consultar fuentes actualizadas para obtener información vigente.

GitHub Copilot cuenta con una función de chat (Copilot Chat) en versiones recientes, permitiendo al usuario hacer preguntas sobre el código o solicitar explicaciones. Si bien es una herramienta de pago mediante suscripción, es gratuita para estudiantes verificados y proyectos open-source, facilitando su adopción en entornos educativos [55].

- **ChatGPT (OpenAI):** Si bien ChatGPT no es exclusivamente una herramienta de programación, se ha convertido en un asistente omnipresente para desarrolladores por su capacidad de resolver dudas y generar código a través de conversación natural. Basado en la familia de modelos GPT de OpenAI, ChatGPT permite mantener un diálogo con la IA: el usuario puede describir un problema o preguntar cómo implementar cierta funcionalidad, y el modelo responde con explicaciones detalladas e incluso con el código completo en el lenguaje solicitado. Estudios en entornos educativos han observado que ChatGPT efectivamente "facilita tareas de programación mediante el uso estratégico de *prompts*", guiando a los estudiantes hacia información relevante cuando formulan bien sus preguntas [57]. Su enfoque generalista lo hace muy flexible, aunque no se integra directamente en el IDE por defecto y desarrolladores encuestados destacan que ChatGPT tiene cierta tendencia a sonar convincente aunque la respuesta sea incorrecta [54].

- **Amazon CodeWhisperer:** Es la propuesta de Amazon Web Services (AWS) en este campo. CodeWhisperer es un asistente de código gratuito para desarrolladores individuales que se integra con IDEs como Visual Studio Code, PyCharm, IntelliJ, etc. Su funcionamiento es similar a Copilot, ofreciendo sugerencias de código contextuales en tiempo real dentro del editor [58]. Una característica diferenciadora es su optimización para el ecosistema AWS: CodeWhisperer puede reconocer cuando el desarrollador trabaja con APIs o servicios de Amazon y sugerir código específico para esos casos.

- **Blackbox AI:** Blackbox es un asistente de programación impulsado por IA desarrollado por una startup, que ha ganado popularidad al reclamar millones de usuarios. A diferencia de otras soluciones integradas en IDE, Blackbox ofrece múltiples vías de uso: una aplicación web/móvil y extensiones para navegadores/editores que permiten invocar al asistente en diferentes entornos [59]. Su premisa es entender instrucciones en lenguaje natural y traducirlas a código en el lenguaje que el usuario elija. En entornos como Visual Studio Code, Blackbox se manifiesta de forma similar a Copilot, sugiriendo código a medida que se escribe y permitiendo aceptar la sugerencia con atajos [60].

- **DeepSeek:** Es una herramienta emergente en el panorama de asistentes de IA, fruto de una startup con origen en China que ha captado la atención por su estrategia disruptiva [61]. DeepSeek se distingue por apostar por una plataforma abierta y eficiente:

ha desarrollado sus propios modelos de lenguaje especializados en programación (DeepSeek Coder [62]), entrenados desde cero con conjuntos masivos de datos (mezclando código y texto técnico [63]). A pesar de ser más recientes, los modelos de DeepSeek han demostrado rendimiento de vanguardia en generación y comprensión de código, con resultados competitivos a los de GPT-4 en tareas de autocompletar código, corrección de bugs y explicación de código. Una característica destacada es su capacidad de manejar contextos extensos: DeepSeek Coder puede procesar archivos grandes y mantener la coherencia a través de distintas partes de un proyecto, lo que resulta útil en proyectos complejos donde se requiere entender interdependencias entre módulos. DeepSeek ofrece una interfaz de chatbot (similar a ChatGPT) y plugins para IDE, aunque su presencia global todavía es limitada en comparación con herramientas establecidas.

- **Google Gemini (Code Assist):** Google, a través de su división DeepMind, ha desarrollado Gemini, una familia de modelos concebidos para ser multimodales y altamente eficientes. En el contexto de asistentes de programación, Google lanzó Gemini Code Assist, un servicio impulsado por la versión 2.0 de Gemini específicamente optimizada para tareas de codificación. Gemini Code Assist se integró a finales de 2024 con Google Cloud y como extensión en editores (VS Code, JetBrains), y desde 2025 está disponible gratuitamente para usuarios individuales [64]. Ofrece completado de código prácticamente ilimitado y una función de Code Review asistido por IA. Su integración con las herramientas de Google implica que también ofrece características como chat con la IA sobre el código (similar a Copilot Chat) y potencial para conectarse con otros servicios de Google.

2.2.3. Casos de uso documentados en desarrollo web

En los últimos años ha aumentado drásticamente el uso de asistentes de inteligencia artificial para desarrollo web. La adopción en la industria es alta: encuestas recientes revelan que el 76 % de los desarrolladores ya usan o planean usar herramientas de IA en su proceso de desarrollo [65]. Las grandes empresas también reportan un impacto notable: por ejemplo, Google indicó que más del 25 % de su código nuevo es generado con ayuda de IA [64]. Estos asistentes están integrándose en los flujos de trabajo profesionales porque soportan múltiples lenguajes (desde Python o Java hasta lenguajes web como JavaScript, HTML, CSS y PHP), facilitando tanto el *front-end* como el *back-end*. Un estudio que analizó discusiones de desarrolladores en foros identificó que JavaScript es, junto con Python, de los lenguajes más usados con asistentes como Copilot [66]. La satisfacción de los programadores con estas herramientas es elevada (en torno al 90 % en algunos casos [67]), lo que refleja la rápida asimilación de la IA como "compañero de programación" en entornos profesionales.

Diversos estudios documentan mejoras en la productividad al emplear asistentes de código basados en grandes modelos de lenguaje. En 2024, un estudio con más de 4.800 desarrolla-

dores en empresas (Microsoft, Accenture, etc.) encontró que quienes usaban un asistente, como GitHub Copilot, completaban un 26 % más de tareas semanalmente en promedio [68]. Además de acelerar la escritura de código, los asistentes ayudan a ahorrar tiempo en tareas repetitivas: en un caso de estudio empresarial, los desarrolladores estimaron un ahorro cercano al 20 % del tiempo gracias a la autocompleción inteligente [69]. Muchos programadores valoran que la herramienta les permite enfocarse en partes más complejas o creativas del proyecto mientras delegan el código rutinario. También se han observado mejoras en la calidad percibida del código: por ejemplo, Copilot puede actuar como "pseudo-revisor" sugiriendo correcciones y buenas prácticas, contribuyendo a reducir errores comunes. Asimismo, más de la mitad de los desarrolladores encuestados afirman que la IA les permite codificar más rápido, buscar menos en internet y completar antes las tareas repetitivas, contribuyendo en general a un flujo de trabajo más eficiente [65].

A pesar del intenso interés industrial y la abrumante cantidad de estudios publicados desde que estas tecnologías generativas emergieron, existen lagunas significativas en la investigación sobre su aplicación específica en tecnologías web fundamentales (HTML, CSS, JavaScript y PHP). Incluso GitHub, en su estudio más reciente, admite que la programación asistida por IA es un campo relativamente nuevo y que existe poca investigación previa en la que apoyarse [70]. La mayor parte de las investigaciones analizadas se concentra en lenguajes como Python, Java o C++, relegando las tecnologías web a un papel secundario. Una pequeña parte de los artículos incluyen casos prácticos en desarrollo *front-end/back-end*, y estos suelen tratar HTML/CSS como complementos, no como foco principal [71, 72, 73, 74]. No existen protocolos estandarizados para evaluar calidad o eficiencia al utilizar IA en proyectos web integrados (PHP + JS + HTML). La investigación no ha seguido el ritmo de la implementación práctica, particularmente en tecnologías básicas de desarrollo web. Este desfase justifica la necesidad de estudios empíricos que analicen cómo los asistentes de IA afectan en la creación de interfaces web (HTML/CSS), la implementación de lógica cliente-servidor (JS/PHP) o la depuración de código multiplataforma.

2.3. IA en la Educación

2.3.1. Integración de la IA en entornos educativos.

La inteligencia artificial generativa ha irrumpido en los sistemas educativos con una velocidad sin precedentes, transformando metodologías pedagógicas y procesos de aprendizaje [75, 76]. Los estudios recientes coinciden en que su adopción masiva se debe a tres factores clave: accesibilidad tecnológica, interfaces intuitivas y capacidad de personalización. Herramientas como ChatGPT han demostrado una rápida adopción en entornos universitarios, con aumentos documentados del 90 % en el uso de IA para asignaciones académicas tras su implementación estructurada [77, 78, 72]. Esta integración se extiende desde la creación auto-

matizada de materiales didácticos hasta la tutoría personalizada, abarcando tanto educación superior como formación técnica [79, 80, 81].

La implementación de flujos de trabajo con IA ha incrementado la adopción de estas herramientas entre estudiantes, evidenciando su potencial para optimizar procesos académicos [78]. Sin embargo, la integración efectiva requiere marcos pedagógicos claros, como señala la investigación de [82], que subraya la necesidad de combinar alfabetización digital con criterios éticos para evitar dependencias tecnológicas acríticas.

Beneficios generales

La literatura científica reciente documenta que, cuando se utilizan de forma ética y adecuada, las herramientas de IA generativa pueden transformar los procesos de enseñanza-aprendizaje en todos los niveles educativos. Entre sus aplicaciones destacan la automatización en la creación de materiales didácticos, la generación de ejercicios y cuestionarios personalizados, la elaboración de resúmenes, la ayuda en la redacción de textos y la retroalimentación instantánea a los estudiantes. Además, la IA generativa fomenta la creatividad y la competencia digital, y facilita la inclusión educativa mediante la adaptación de contenidos para estudiantes con necesidades específicas [83, 84, 85].

Retos y consideraciones éticas

A pesar de los beneficios que la IA puede aportar en educación, su integración también plantea desafíos importantes. Destacan la necesidad de garantizar la veracidad y calidad de la información generada, mantener la integridad académica y formar tanto a docentes como a estudiantes en el uso ético de estas herramientas. La UNESCO y otros organismos internacionales subrayan la importancia de un enfoque regulatorio y formativo que preserve el papel central de los docentes y asegure que la educación siga siendo un proceso humano y social, evitando la automatización excesiva y el riesgo de dependencia tecnológica [86, 87, 88, 81].

2.3.2. Asistentes de IA en la enseñanza de programación

La integración de asistentes de IA en la enseñanza de programación ha generado un interés creciente, con estudios que muestran resultados contradictorios sobre su impacto pedagógico. Investigaciones recientes revelan que la mayoría de los estudiantes universitarios utilizan herramientas como GitHub Copilot o ChatGPT para resolver ejercicios de programación, aunque su eficacia varía según el nivel de experiencia y el contexto educativo [89, 90].

Los estudios coinciden en que estos sistemas mejoran la eficiencia operativa en tareas rutinarias. Por ejemplo, en un experimento controlado con estudiantes novatos, el uso de GitHub Copilot redujo un 40 % el tiempo requerido para completar ejercicios básicos de

Python, aunque no mostró mejoras significativas en la autoeficacia o la comprensión conceptual [89]. Un estudio en Estonia con 210 estudiantes mostró que el uso moderado de IA (2-3 veces/semana) se asociaba con un aumento del 15 % en calificaciones teóricas, pero una reducción del 10 % en habilidades prácticas de depuración del código [91]. En [92] concluyen que los usuarios avanzados (3 o más años programando) son un 40 % más eficientes corrigiendo errores lógicos en código generado por IA que los principiantes.

Para lenguajes como C, los asistentes basados en LLMs han demostrado ser particularmente útiles en la detección de errores sintácticos, con tasas de resolución autónoma del 78 % para problemas básicos [93]. En cursos introductorios de Java, el 63 % de los docentes reportan que ChatGPT facilita la creación de materiales didácticos personalizados, especialmente en la generación de ejemplos contextualizados [94].

Los trabajos, en general, coinciden en tres ventajas transversales:

1. **Feedback inmediato y personalizado:** La retroalimentación en tiempo real reduce la frustración y acelera el progreso según experimentos quasi-experimentales con C++ y Python [95, 94].

2. **Incremento de motivación y autoeficacia:** Integrar IA generativa eleva la motivación y la satisfacción del alumnado en grupos de programación introductoria [96].

3. **Apoyo a la docencia:** Manuales prácticos para docentes como [97] defienden que Copilot y ChatGPT liberan tiempo de corrección y permiten concentrarse en diseño de tareas de mayor nivel.

Sin embargo, la literatura también identifica dos desafíos críticos:

1. **Dependencia cognitiva:** El 42 % de los estudiantes novatos que usan Copilot desarrollan dificultades para depurar código sin asistencia, mostrando una brecha entre la productividad inmediata y la comprensión profunda [89, 92, 98].

2. **Limitaciones técnicas:** Los LLMs presentan tasas de error del 22 % en explicaciones de conceptos avanzados, requiriendo supervisión docente constante [99, 90]. Se han documentado alucinaciones y errores lógicos que los estudiantes copian sin verificar, elevando la tasa de respuestas incorrectas hasta en un 61 % cuando la IA se equivoca [100].

3. **Integridad académica:** Varios estudios relacionan el uso intensivo de chatbots con un aumento del plagio de código y la dificultad de detectarlo [100, 98].

2.3.3. Vacíos en la investigación de IA aplicada en ciclos formativos

Pese a todos los avances, ningún estudio analiza específicamente el impacto de introducir inteligencia artificial en los currículos de los ciclos de formación profesional. Las revisiones

más recientes confirman que la evidencia se limita a educación universitaria, bachillerato o a informes corporativos, dejando sin explorar contextos orientados al empleo inmediato, con escasa atención a: La adaptación de IA para proyectos de programación, especialmente de desarrollo web; Métricas de competencia técnica en entornos de FP; Estrategias antiplagio en evaluaciones prácticas [101]. Esta ausencia contrasta con la demanda industrial, que pide incorporar la IA en flujos de trabajo profesionales y demandan graduados capaces de trabajar con estas herramientas.

Aunque los asistentes de IA muestran potencial para transformar la enseñanza de programación, su implementación en FP requiere enfoques pedagógicos innovadores que mitiguen riesgos y aprovechen oportunidades específicas del contexto técnico-profesional. La ausencia de estudios en ciclos subraya la relevancia de investigaciones empíricas que exploren estas dinámicas en entornos reales de FP.

Capítulo 3

Metodología

En las siguientes secciones se detalla el contexto y escenario, los participantes, los instrumentos de recogida de datos y las técnicas de análisis empleadas.

3.1. Diseño de la investigación

El estudio descrito en este trabajo adopta un diseño mixto, al integrar métodos cuantitativos (calificaciones) y cualitativos (diario de *prompts*, observación participante, cuestionario de percepción). Esta combinación permite captar no sólo el impacto numérico de la inteligencia artificial generativa sobre el rendimiento, sino también las experiencias, motivaciones y estrategias que desarrollan los estudiantes.

3.1.1. Enfoque general

Para este estudio se ha optado por un enfoque explicativo secuencial que sigue dos fases:

1. *Fase cuantitativa*: medición y comparación de resultados académicos entre la cohorte 2024/25 (incluyengo IA generativa) y la cohorte 2023/24 (metodología tradicional).

2. *Fase cualitativa*: recogida de diarios de *prompts* y ejercicios de uso de IA y cuestionarios de percepción para profundizar en la interpretación de los hallazgos cuantitativos.

3.1.2. Diseño específico

El estudio se configura como un cuasi-experimento con grupo de comparación histórico. No fue posible la asignación aleatoria de los participantes, dada la imposibilidad de alterar la organización académica del centro o las aulas, de modo que se contrastaron:

- **Grupo experimental** (curso 2024/25): 47 estudiantes distribuidos en tres módulos de programación web —Aplicaciones web (AW) en 2ºSMR (24 estudiantes), Lenguajes

de Marcas y Sistemas de Información (LM) en 1ºASIR (14 estudiantes) e Implantación de Aplicaciones Web (IAW) en 2ºASIR (9 estudiantes)— que utilizaron asistentes de IA tras un período inicial de fundamentos de los diferentes lenguajes sin IA.

- **Grupo control histórico** (curso 2023/24): 34 estudiantes de los mismos módulos (salvo LM) y ciclos (salvo 1ºASIR) formados con las mismas unidades didácticas, pero sin soporte de IA generativa.

Todos los estudiantes del grupo experimental realizaron primero una primera parte teorico-práctica de conocimientos fundamentales sin IA; después, se introdujo la ingeniería de *prompts*, la depuración asistida y, por último, proyectos completos con IA. La intervención se extendió desde septiembre de 2024 hasta abril de 2025.

3.1.3. Justificación del diseño

Este diseño mixto y cuasiexperimental resulta adecuado por varias razones que permiten responder a los objetivos clave de la investigación:

- Permite cuantificar la efectividad pedagógica de la IA generativa mediante calificaciones objetivas y compararlas con un curso anterior homogéneo en currículo y evaluación.
- Incorpora la visión cualitativa de los actores (alumnado y docente), esencial para entender procesos como la motivación, la autoeficacia y la evolución en la redacción de *prompts*.
- Se ajusta a las limitaciones logísticas del contexto (tamaño reducido de grupos, ausencia de asignación aleatoria y tiempos lectivos de 55 minutos por clase).
- Facilita la transferencia de resultados a entornos reales de FP, donde normalmente se trabaja con cohortes sucesivas más que con grupos paralelos.

3.2. Contexto del estudio

3.2.1. Centro e institución

El estudio se llevó a cabo en *Enseñanzas Profesionales Sorolla*[1], un centro de Formación Profesional integrado en *Colegio Sorolla Secundaria–Profesional*[2], centro privado-concertado perteneciente a *Grupo Sorolla Educación*[3] y situado en la ciudad de Valencia (Comunidad Valenciana). El colegio ofrece todas las etapas educativas, desde Infantil hasta Bachillerato, y

[1]Enseñanzas Profesionales Sorolla: `https://www.profesionalsorolla.es/`
[2]Colegio Sorolla: `https://colegiosorolla.es/`
[3]Grupo Sorolla Educación: `https://gruposorollaeducacion.es/`

cuatro ciclos formativos concertados: SMR[4], Administración y Finanzas, APSD[5] y Farmacia y Parafarmacia, además del ciclo privado de ASIR[6].

3.2.2. Módulos implicados

La intervención se aplicó en tres módulos de programación web de los ciclos de Formación Profesional relacionados con informática. La lengua de docencia en los tres grupos es el castellano; cada ciclo dispone de su propio aula-taller equipada con un ordenador de sobremesa por estudiante, lo que garantiza la autonomía y continuidad del trabajo individual durante el curso. Todos los módulos relacionados con tecnologías web y bases de datos los imparte la misma docente (autora de este trabajo).

Tabla 3.1: Módulos, ciclos y carga horaria semanal.

Ciclo	Módulo	Lenguajes impartidos	Horas/sem
2.º SMR	AW	HTML, CSS, JavaScript	4
1.º ASIR	LM	HTML, CSS, XML, JSON, jQuery	3
2.º ASIR	IAW	PHP, MySQL	5

3.2.3. Calendario de la intervención

La intervención se desarrolló durante el primer y segundo trimestre del curso 2024-2025 (entre septiembre de 2024 y marzo de 2025) para los segundos cursos (2º SMR y 2º ASIR)[7], y también durante parte del tercer trimestre para el primer curso de ASIR[8].

La secuencia didáctica se organizó en varias fases:

- **Fase inicial** (aprox. 6-8 semanas): Fundamentos de los lenguajes y ejercicios sin IA.
- **Introducción a la IA e ingeniería de prompts** (2-3 semanas): Formación específica y primeras prácticas guiadas.
- **Depuración de código con IA** (1-2 semanas): Ejercicios específicos de corrección y análisis de código.
- **Desarrollo de proyectos integrados e integración de nuevos lenguajes** (Resto del curso): Progresiva incorporación de asistentes de IA en proyectos de mayor envergadura, con seguimiento y evaluación continua.

[4]Sistemas Microinformáticos y Redes
[5]Atención a Personas en Situación de Dependencia
[6]Administración de Sistemas Informáticos en Red
[7]A partir de marzo los segundos cursos realizan la formación en empresa.
[8]El primer curso de ASIR realizó la formación en empresa en mayo.

Todas las clases se desarrollaron en los respectivos talleres informáticos, con recursos audiovisuales (pantalla y pizarra) para demostraciones en directo y grabación de explicaciones, que posteriormente se subieron a YouTube[9] para su consulta.

3.2.4. Recursos tecnológicos

Cada estudiante dispuso de un ordenador de sobremesa con Windows 10/11, acceso a internet. El software base incluyó Visual Studio Code[10] como editor principal, XAMPP[11] para servidores locales y GitHub Pages[12] o Azure[13] para la publicación de proyectos web, según el ciclo.

En cuanto a herramientas de IA, se emplearon principalmente la GitHub Copilot[14] (licencia educativa y, posteriormente, versión gratuita integrada en VS Code) y herramientas con interfaz web como ChatGPT[15], Claude[16], Deepseek[17] y Blackbox[18], aunque los estudiantes eligieron libremente la herramienta que preferían utilizar[19].

3.3. Participantes

El estudio involucró dos cohortes de estudiantes de los mismos ciclos formativos del mismo centro educativo, correspondientes a cursos académicos consecutivos (2023/2024 y 2024/2025).

3.3.1. Muestra experimental (2024/2025) - Con asistentes de IA

La muestra del estudio está compuesta por la totalidad de los estudiantes matriculados en los grupos implicados durante el curso 2024/2025. Participaron en total 47 alumnos:

- 2.º SMR: 24 estudiantes *(edad 18-43, 1 mujer)*
- 1.º ASIR: 14 estudiantes *(edad 19–31, 2 mujeres)*
- 2.º ASIR: 9 estudiantes *(edad 20-38, 1 mujer)*

[9] Todos los vídeos están accesibles en el canal de YouTube: `https://www.youtube.com/@liconoc`
[10] Visual Studio Code: `https://code.visualstudio.com/`
[11] XAMPP: `https://www.apachefriends.org/es/index.html`
[12] GitHub pages: `https://docs.GitHub.com/es/pages`
[13] Azure: `https://azure.microsoft.com/es-es`
[14] GitHub copilot: `https://GitHub.com/features/copilot`
[15] ChatGPT: `https://chatgpt.com/`
[16] Claude: `https://claude.ai/`
[17] Deepseek: `https://www.deepseek.com/en`
[18] Blackbox: `https://www.blackbox.ai/`
[19] Las herramientas empleadas por los estudiantes fueron variando a lo largo del curso por la rápida actualización de las diferentes versiones de los modelos de IA generativa (como las diferentes versiones de ChatGPT o Claude), adaptaciones (cambios en la extensión de Copilot) o aparición de nuevas herramientas que al inicio del curso no existían (como Deepseek).

3.3.2. Muestra de control (2023/2024) - Sin asistentes de IA

Para el análisis comparativo, se utilizaron los datos de rendimiento académico del curso anterior correspondientes a los mismos módulos y ciclos formativos[20]. La muestra de control estuvo compuesta por 34 estudiantes[21]:

- 2.º SMR: 24 estudiantes *(2 mujeres)*
- 2.º ASIR: 10 estudiantes *(1 mujer)*

3.3.3. Descripción de la muestra

La cohorte del curso 2023-2024 se componía de 34 estudiantes, mientras que la del curso 2024-2025 amplía el tamaño muestral hasta 47 estudiantes. Se mantiene la fuerte sobrerrepresentación masculina —92 % de varones frente al 8 % de mujeres—, tendencia que ya se observaba el curso anterior (91 % vs 9 %). La Figura 3.2 y la Figura 3.3 sintetizan la distribución de género por grupo y curso.

Tabla 3.2: Distribución de estudiantes por sexo y grupo (curso 2023–2024)

Grupo	H	M	Total
2º ASIR	9	1	10
2º SMR	22	2	24
Total	**31**	**3**	**34**

Tabla 3.3: Distribución de estudiantes por sexo y grupo (curso 2024–2025)

Grupo	H	M	Total
1º ASIR	12	2	14
2º ASIR	8	1	9
2º SMR	23	1	24
Totales	**43**	**4**	**47**

En cuanto a la edad (Tabla 3.4), el rango total oscila entre 18 y 43 años, con una mediana de 20-21. 1º ASIR concentra la mayoría de alumnos en el tramo 19-23, mientras que 2º ASIR presenta mayor dispersión (casos puntuales de 29 y 38 años). 2º SMR agrupa casi dos tercios de sus estudiantes en los 18-19 años, lo que evidencia su perfil marcadamente juvenil.

[20]El grupo de 1.º ASIR no participó en el curso 2023/2024 porque la docente no impartía clase en ese módulo, por lo que la comparación para este grupo no fue posible.

[21]La edad de esta cohorte no está disponible por un cambio en la aplicación académica usada en el centro de estudios.

Tabla 3.4: Distribución de edad y sexo por grupo (curso 2024–2025).

Grupo	Edad	H	M	Total
1º ASIR	19	5	0	6
	20	3	1	4
	21	2	0	3
	23	2	0	2
	31	0	1	1
	Subtotal	**12**	**2**	**14**
2º ASIR	20	1	0	1
	21	1	0	1
	22	2	0	2
	23	1	0	1
	25	2	0	2
	29	0	1	1
	38	1	0	1
	Subtotal	**8**	**1**	**9**
2º SMR	18	16	1	17
	19	3	0	3
	21	1	0	1
	22	1	0	1
	29	1	0	1
	43	1	0	1
	Subtotal	**23**	**1**	**24**
Total general		**43**	**4**	**47**

3.3.4. Características y comparabilidad de las muestras

En cuanto a la experiencia previa en programación, la mayoría de los estudiantes de SMR no tenían experiencia más allá de posibles actividades con Scratch en secundaria. En 1º ASIR, aproximadamente la mitad provenía de bachillerato sin experiencia en programación, y la otra mitad de ciclos de SMR, con conocimientos básicos de HTML y CSS, y en algunos casos JavaScript. Los estudiantes de 2.º ASIR tenían conocimientos de HTML y CSS, aprendidos el curso anterior.

Respecto a necesidades educativas especiales, se identificaron casos de TDAH en ambas cohortes: tres casos en el curso 2024/2025 y dos casos en el curso 2023/2024.

Ambas muestras fueron impartidas por la misma docente (autora del presente trabajo), utilizando el mismo temario base, con la diferencia de que en el curso 2024/2025 se incorporó el uso de asistentes de inteligencia artificial.

3.4. Procedimiento

3.4.1. Metodología general

La docencia en los ciclos formativos de 2º SMR, 2º ASIR y 1º ASIR se organizó mediante una metodología común, caracterizada por el predominio de clases prácticas presenciales, participación activa del alumnado y evaluación continua.

Metodología de enseñanza

- Introducción teórica y explicaciones conceptuales en pizarra al inicio de cada sesión.
- Programación en directo proyectada por la docente, permitiendo al alumnado replicar los ejercicios en sus propios ordenadores.
- Resolución de dudas y participación activa mediante la salida a la pizarra para proponer soluciones y explicar fragmentos de código.
- Realización de tareas individuales o en grupo relacionadas con el contenido visto en la clase.

Secuencia de contenidos por grupo

El diseño curricular en los distintos grupos se estructuró secuencialmente para garantizar la adquisición progresiva de competencias en desarrollo web y programación, adaptando la profundidad y la complejidad de los contenidos al nivel y experiencia previa del alumnado:

- **2º SMR:**
  - **Primera evaluación:** HTML (etiquetas, listas, enlaces, imágenes, tablas) y CSS (fundamentos, selectores y estructura de caja).
  - **Segunda evaluación:** Introducción a JavaScript.
- **2º ASIR**[22]**:**
  - **Primera evaluación:** Repaso de fundamentos de HTML, CSS y JavaScript; introducción a la programación en PHP; procesamiento de formularios web y conexión con bases de datos MySQL.
- **1º ASIR:**
  - **Primera evaluación:** Fundamentos de HTML y CSS.
  - **Segunda evaluación:** Introducción a XML, DTD/XMLSchema y JSON.

- **Tercera evaluación:** Manejo de jQuery y manipulación básica del DOM, incluyendo eventos y animaciones.

Criterios de evaluación

El sistema de evaluación aplicado en los módulos de informática del centro de estudios se estructura en diferentes componentes y ponderaciones, diseñados para valorar de forma integral tanto el desempeño técnico como las competencias transversales del alumnado:

- **Actividades en el aula (40 %)**
  - 30 %: Actividades prácticas realizadas en el aula a lo largo del curso.
  - 10 %: Comprensión lectora y redacción de documentación asociada a las prácticas.
- **Proyecto (20 %)**
  - Proyectos realizados bajo metodología ABP (Aprendizaje Basado en Proyectos), integrando distintos módulos y promoviendo el trabajo interdisciplinar.
- **Examen teórico (15 %)**
  - Prueba teórica sobre los contenidos impartidos en el aula.
- **Examen práctico (15 %)**
  - Prueba práctica de ejecución sobre las actividades desarrolladas en clase.
- **Competencia socio-profesional (10 %)**
  - Se valora la cortesía, código de presencia, puntualidad, responsabilidad y autonomía.

3.4.2. Diferenciación metodológica por incorporación de IA en la cohorte experimental

Tras la adquisición de los conceptos fundamentales en cada lenguaje, se introdujo en los tres grupos experimentales (curso 2024/2025) la integración de fundamentos de inteligencia artificial e ingeniería de prompts como elemento diferenciador respecto a las cohortes anteriores:

1. **Introducción teórica a la IA:** Formación básica sobre IA generativa, funcionamiento de modelos de lenguaje y principios de la ingeniería de prompts enfocados a la programación asistida.

[22] En la segunda evaluación de 2º ASIR se introdujo Wordpress, que no se aborda en esta memoria.

2. **Ingeniería de prompts:** Desarrollo de habilidades para la redacción efectiva de instrucciones (prompts), enfatizando claridad, especificidad y secuenciación, así como la importancia de la iteración y depuración de resultados.
3. **Depuración y revisión asistida:** Depurar código y revisión de buenas prácticas a través de agentes conversacionales.
4. **Integración en evaluaciones y aumento del nivel de exigencia:** Una vez consolidada la competencia básica, el uso de IA fue habilitado para ejercicios, proyectos y exámenes.

Adaptación de los criterios de evaluación

La estrategia evaluativa del estudio difiere de manera fundamental entre la cohorte control y la cohorte experimental, adaptándose a las competencias y herramientas disponibles en cada contexto.

Cohorte 2023/2024 (sin IA): El formato de evaluación se basaba principalmente en la resolución de fragmentos de código. Este formato permitía asegurar la viabilidad de la pruebas y ejercicios en el tiempo establecido, valorando la comprensión conceptual y la destreza manual sin apoyos automáticos.

Ejemplo: La estructura de los exámenes requería que el alumnado completara fragmentos de código como los siguientes:

```
<!-- Un span con un enlace dentro. El texto del enlace es Ofertas.
El enlace es un ancla a id="ofertas" -->
...
<!-- Una celda de la tabla con id="tabla-izquierda". Dentro tiene una imagen
(imagenes/blackfriday.png). La imagen también es un enlace. -->
```

Cohorte 2024/2025 (con IA): Con la adopción plena de asistentes de IA, el diseño de evaluación evolucionó para que los exámenes y proyectos permitieran y estimularan un uso genuino, autónomo y crítico de las herramientas:

- Se plantearon proyectos web completos, enunciados más elaborados que requerían planificación, desarrollo, integración, documentación y defensa técnica.
- Se elaboraron portafolios web avanzados (2º SMR, 1º ASIR) y tiendas online completas con integración de base de datos y áreas personales (2º ASIR).
- Se valoró no solo el resultado final, sino también la calidad de los prompts utilizados, la capacidad de iterar y depurar, la reflexión crítica sobre el proceso, y la publicación/documentación en formato profesional.

Justificación metodológica del cambio:

- El acceso a IA hacía trivial la resolución de fragmentos aislados de código: en pruebas simuladas, los sistemas generaban las respuestas completas en cuestión de segundos.
- Se incrementó el nivel de exigencia al incorporar la creación de proyectos reales y documentados.
- Se mantuvo la duración de los exámenes para garantizar la comparabilidad, aumentando proporcionalmente la dificultad y el alcance de los proyectos.
- Los nuevos criterios permiten valorar cómo el alumnado integra la IA en procesos creativos, colaborativos y de resolución de problemas reales, alineándose con el cambio de paradigma en la profesión.

Criterios de corrección y rúbricas:

- *Modalidad sin IA:* Rúbricas dirigidas principalmente a la corrección funcional y sintáctica de los fragmentos entregados, medida objetiva de los apartados completados.
- *Modalidad con IA:* Rúbricas ampliadas que contemplan estructura, estilo y navegación del sitio, implementación de CSS avanzado, documentación y originalidad, publicación en GitHub, uso reflexivo de la IA, justificación de decisiones técnicas y cumplimiento de requisitos avanzados estipulados en cada proyecto.

3.5. Variables y métricas

- **Variable independiente**: La condición didáctica, es decir, cohorte con integración de asistentes de IA (curso 2024/25) frente a cohorte histórica sin IA (2023/24). Dado que en el grupo control no hubo 1º ASIR (LM), los contrastes inferenciales solo se realizan en 2º ASIR y 2º SMR, que tienen módulos comparables entre cursos.
- **Variables dependientes cuantitativas primarias**: (i) calificación media por estudiante (0–10) en los módulos comparables, y (ii) su distribución por tramos de nota, empleadas para análisis descriptivos y contrastes por cohorte.
- **Variables dependientes secundarias**: Los ítems del cuestionario post-intervención (n=37) que incluyen el uso declarado de herramientas, la frecuencia, la competencia en prompting, la dificultad percibida, el impacto en el aprendizaje, la calidad del código y las preferencias metodológicas; se tratan como variables categóricas (frecuencias/porcentajes por opción).

- **Evidencias de desempeño con IA en ejercicios prácticos**: Se operacionalizan siete indicadores binarios (presencia=1/ausencia=0) en cada entrega, incluyendo la formulación intencional, la iteración con mejora, el diagnóstico/depuración, la autoexplicación, la verificación básica, la autonomía crítica y la reflexión sobre límites de la IA. Se calculan porcentajes de presencia por indicador (n=45 en ingeniería de *prompts* y n=35 en depuración).
- **Covariables descriptivas**: El grupo (1.º ASIR, 2.º ASIR, 2.º SMR), el sexo y la edad, usadas para describir la muestra y valorar la comparabilidad entre cohortes.

Todas estas variables se describen en el capítulo 4.

3.6. Técnicas de análisis

- Se realiza un análisis descriptivo de la muestra (frecuencias, porcentajes, medias y desviaciones típicas) y de las calificaciones por cohorte (tablas de distribución).
- Para contrastes inferenciales entre cursos se aplica la *t de Student* para muestras independientes en 2.º ASIR y 2.º SMR, con $\alpha = 0{,}05$ (bilateral), reportando t, p y tamaño del efecto d de Cohen. Se informa además de la potencia observada y del tamaño muestral estimado para $1 - \beta = 0{,}80$ (umbral aproximado $n \approx 100$ por grupo), que contextualizan la interpretación de no-significatividad con efectos moderados.
- El cuestionario se analiza con estadística descriptiva (frecuencia y porcentaje por opción). Dado el tamaño muestral y la finalidad exploratoria, no se aplican modelos multivariantes; cuando procede, se discuten patrones consistentes entre ítems.
- Para los ejercicios prácticos, se utiliza análisis de contenido dirigido con esquema de codificación deductivo basado en los siete indicadores operativizados. La fiabilidad se garantiza mediante definición previa de criterios, ejemplos positivos/negativos y calibración interna; al tratarse de un único codificador (docente-investigadora), se reconoce como limitación la ausencia de acuerdo intercodificador, mitigada con auditoría de trazas (portafolios y capturas) y transparencia en el anexo de enunciados.

3.7. Consideraciones éticas

La intervención se integra en la docencia ordinaria y respeta los principios de minimización de riesgos, voluntariedad y anonimización. El cuestionario fue voluntario, no recogió datos especialmente protegidos y se analizó de forma agregada; las evidencias de ejercicios se anonimizan (sin nombres ni identificadores personales). Los datos se almacenaron en repositorios institucionales bajo control docente y con acceso restringido.

En términos de integridad académica, se explicitó al alumnado el uso permitido de asistentes de IA y se adaptaron las rúbricas para evaluar pensamiento crítico y autoría (no solo producto final).

Se actúa conforme a la normativa vigente de protección de datos (tratamiento con fines docentes y de investigación educativa, anonimización de resultados, conservación limitada), y a las recomendaciones internacionales sobre IA en educación (uso responsable, supervisión docente, prevención de dependencia y de alucinaciones), contextualizadas en el Capítulo 2.

Capítulo 4

Resultados

4.1. Rendimiento académico

4.1.1. Distribución global de calificaciones

La tabla 4.1 compara la frecuencia de las notas medias entre cursos. La cohorte 2024-2025 muestra un desplazamiento hacia notas más altas ($\overline{x}$ = 8,0) frente a 2023-2024 ($\overline{x}$ = 7,4), junto a una ligera reducción de la dispersión (DT 1,0-1,5 vs 1,6-1,8). Estos datos revelan la desaparición de puntuaciones inferiores a 6 y un incremento del segmento 8-10.

Tabla 4.1: Distribución de las calificaciones medias por curso (nº de estudiantes por nota media).

Nota media	**2024–2025**	**2023–2024**
5	0	4
6	6	5
7	5	3
8	11	7
9	18	11
10	7	4

4.2. Contrastes inferenciales

Se aplicó la prueba t de Student para contrastar 2º ASIR y 2º SMR entre cohortes (Tabla 4.2). No se detectaron diferencias significativas (2º ASIR: $t = -0,89$, p = 0,387; 2º SMR: $t = -1,25$, p = 0,219). Sin embargo, los tamaños del efecto ($d = 0,40$ y $d = 0,36$) indican una mejora moderada que la prueba no alcanza a confirmar debido a la baja potencia (0,13–0,23). El cálculo a priori sitúa en $n \approx 100$ alumnos por grupo el umbral para lograr $1 - \beta = 0,80$.

Tabla 4.2: Resultados t Student y tamaños del efecto.

Grupo	n_{23-24}	n_{24-25}	t	p	d
2º ASIR	10	9	-0,89	0,387	0,40
2º SMR	24	24	-1,25	0,219	0,36

4.3. Análisis del cuestionario post-intervención

4.3.1. Descripción del cuestionario y de la muestra

A final de curso, antes de la formación en empresa, se distribuye un cuestionario post-intervención voluntario entre los estudiantes de la cohorte del curso 2024-2025[1]. Respondieron al cuestionario 37 estudiantes, distribuidos de la siguiente manera:

- **1º ASIR**: 12 respuestas (32 % del total).
- **2º ASIR**: 6 respuestas (16 % del total).
- **2º SMR**: 19 respuestas (51 % del total).

Si se pondera por el tamaño real de cada grupo (sección 3.3.3), la tasa de participación fue alta y bastante homogénea: 86 % en 1º ASIR (12/14), 67 % en 2º ASIR (6/9) y 79 % en 2º SMR (19/24). Esto refuerza la representatividad de los resultados, especialmente para 2º SMR, que aporta más de la mitad de las respuestas.

El cuestionario constó de 20 ítems: 14 preguntas cerradas (Likert o elección múltiple) y 6 abiertas. Los ítems cubren siete bloques temáticos: uso de herramientas de IA, frecuencia y hábitos, competencia en *prompting*, dificultad percibida, impacto en el aprendizaje, calidad del código generado y preferencias metodológicas. En las subsecciones siguientes se analiza cada bloque, presentando los resultados cuantitativos y una síntesis de las respuestas abiertas.

4.3.2. Uso de herramientas de IA

La pregunta 2 permitía seleccionar varias opciones; los 37 estudiantes marcaron un total de 115 herramientas (media = 3,1 por estudiante). La Tabla 4.3 muestra los porcentajes sobre el total de encuestados.

El análisis de los resultados revela una adopción universal de ChatGPT (100 % de los encuestados), consolidándose como herramienta principal y punto de entrada a la IA generativa en el aula. Blackbox (65 %) y GitHub Copilot (54 %) destacan como las opciones preferidas para tareas de autocompletado, depuración y refactorizado de código, especialmente integradas en entornos como Visual Studio Code, en sintonía con la metodología seguida en el curso.

[1]El cuestionario completo se encuentra en el anexo A.

Tabla 4.3: Resultados de la pregunta 2: Uso declarado de herramientas de IA (n = 37).

Herramienta	**n**	**% sobre muestra**
ChatGPT	37	100 %
Blackbox	24	65 %
GitHub Copilot	20	54 %
Claude	15	41 %
Gemini	9	24 %
Otras	10	27 %

Los modelos alternativos Claude (41 %) y Gemini (24 %) presentan una presencia significativa, posiblemente motivada por la curiosidad del alumnado o por las restricciones de acceso a ChatGPT. Finalmente, la categoría "Otras" (27 %) indica una tendencia incipiente hacia la exploración de nuevas soluciones (DeepSeek, Perplexity, TalkAI o Monica), lo que refleja una diversidad creciente en el ecosistema de herramientas empleadas por el estudiantado.

4.3.3. Hábitos previos y frecuencia de uso

Antes de usar IA, ¿cómo resolvías dudas de programación?

Antes de incorporar asistentes de inteligencia artificial al flujo de trabajo, los estudiantes recurrían principalmente a fuentes humanas y a la búsqueda en línea para resolver dudas de programación. En la pregunta 3, cada participante podía seleccionar hasta dos opciones; las 37 respuestas generaron 67 selecciones en total. La Tabla 4.4 recoge la frecuencia y el porcentaje de estudiantes que eligieron cada recurso.

Tabla 4.4: Resultados de la pregunta 3: Preferencias de ayuda antes de la IA (n = 37).

Recurso	**n**	**% sobre muestra**
Preguntar al profesor/a	25	37 %
Buscar en internet	20	30 %
Preguntar a compañeros/as	14	21 %
Documentación / libros	8	12 %
Otras	0	0 %

El análisis muestra que la docencia presencial desempeñaba un papel central en la resolución de dudas, siendo la profesora la principal fuente de consulta para la mayoría del alumnado. Un 30 % complementaba esta ayuda con búsquedas en internet, lo que indica un grado relevante de autonomía digital. El 21 % valoraba la consulta a compañeros/as, señalando un entorno de trabajo colaborativo dentro del aula. Por último, el uso de documentación o manuales escritos resultaba minoritario (12 %), reflejando la preferencia por fuentes rápidas y directas de información frente a los recursos más formales y tradicionales.

¿Con qué frecuencia usas IA para programar?

La pregunta 4 del cuestionario investigó la frecuencia de uso de asistentes de IA en tareas de programación. La Tabla 4.5 recoge los resultados globales de la cohorte encuestada.

Tabla 4.5: Resultados de la pregunta 4: Frecuencia de uso (n = 37).

Categoría	n	% sobre muestra
Siempre	18	49 %
Frecuentemente	17	46 %
Ocasionalmente	2	5 %
Nunca	0	0 %

Los resultados evidencian una integración muy elevada de la inteligencia artificial en la dinámica de programación. Prácticamente la mitad del alumnado (49 %) declara emplear asistentes de IA en todas sus tareas de programación y un 46 % adicional lo hace con frecuencia. La opción "ocasionalmente" es puntual (5 %) y ningún estudiante indicó no utilizarlos nunca, lo que pone de manifiesto la penetración casi universal de estas herramientas en el flujo de trabajo tras la intervención docente. Esta distribución homogénea refuerza la idea de que el uso de IA se ha consolidado como parte esencial y habitual en la resolución de problemas de programación a nivel formativo.

4.3.4. Competencia en *prompting* y dificultad percibida

¿Crees que has mejorado tu manera de escribir prompts estructurados desde que empezó el curso?

La Tabla 4.6 presenta la percepción del alumnado respecto a la mejora en la escritura de prompts estructurados tras la intervención formativa.

Tabla 4.6: Resultados de la pregunta 5: Percepción de mejora (n = 37).

Respuesta	n	% sobre muestra
Sí, ahora sé cómo estructurarlo	33	92 %
No, los escribo igual	2	6 %
No estoy seguro/a	1	2 %

Los datos reflejan una percepción muy mayoritaria de mejora: el 92 % del alumnado considera haber adquirido o perfeccionado la capacidad de redactar *prompts* de manera más estructurada a lo largo del curso. Dos estudiantes señalan que no han variado su forma de escribirlos y sólo uno manifiesta dudas al respecto. Este resultado apunta a una evolución positiva generalizada en una competencia clave para el aprovechamiento eficiente de la inteligencia artificial generativa en el desarrollo de tareas de programación.

¿Qué dificultad crees que tiene escribir un buen *prompt*?

La pregunta 6 del cuestionario aborda la percepción de dificultad para redactar *prompts* de calidad por parte del alumnado. La Tabla 4.7 recoge la distribución de respuestas sobre esta cuestión.

Tabla 4.7: Resultados de la pregunta 6: Dificultad percibida (n = 37).

Respuesta	n	% sobre muestra
Neutral	21	57 %
Difícil	7	19 %
Fácil	7	19 %
Muy difícil	1	3 %
Muy fácil	1	3 %

La mayoría del alumnado (57 %) manifiesta una valoración "neutral" respecto a la dificultad de escribir un buen prompt, sin identificar dicha tarea como especialmente compleja ni sencilla. Las percepciones de dificultad y facilidad se distribuyen de manera equivalente (19 % en cada caso), mientras que sólo una minoría muy pequeña sitúa la experiencia en los extremos. Esta distribución sugiere que, tras el proceso de formación e integración de IA en el aula, los estudiantes perciben la escritura de *prompts* como una habilidad asumible.

4.3.5. Impacto percibido en el aprendizaje

¿La IA te ha ayudado a entender cómo programar?

La Tabla 4.8 muestra la percepción estudiantil acerca de la influencia de la inteligencia artificial en su comprensión de la programación.

Tabla 4.8: Resultados de la pregunta 7: Influencia de la IA en la comprensión de la programación(n = 37).

Respuesta	n	% sobre muestra
Sí, mucho	19	51 %
Sí, algo	15	41 %
No, no ha influido	3	8 %

La inmensa mayoría del alumnado (92 %) manifiesta que el uso de asistentes de IA ha contribuido positivamente a su comprensión de la programación, con un 51 % indicando que la ayuda ha sido "mucho" y un 41 % adicional que afirma que "algo". Sólo una minoría (3 estudiantes) no reconoce influencia atribuible a la IA en este aspecto. Estos resultados describen una percepción ampliamente favorable respecto al impacto formativo de las herramientas de inteligencia artificial.

¿Cómo de seguro te sientes escribiendo código sin IA?

La pregunta 8 del cuestionario aborda el nivel de confianza del alumnado al programar sin el apoyo de asistentes de inteligencia artificial. La Tabla 4.9 recoge la distribución de respuestas.

Tabla 4.9: Resultados de la pregunta 8: Auto-eficacia sin IA (n = 37).

Respuesta	n	% sobre muestra
Algo inseguro	24	65 %
Bastante seguro	5	14 %
Muy seguro	3	8 %
Nada seguro	5	14 %

Los resultados reflejan una tendencia marcada hacia la inseguridad percibida a la hora de programar sin la ayuda de la IA: un 65 % del alumnado se declara "algo inseguro" y un 14 % expresa sentirse "nada seguro" en este contexto. Por el contrario, solo una minoría se identifica como "bastante segura" (14 %) o "muy segura" (8 %) programando de manera autónoma. Esta distribución sugiere una dependencia importante de los asistentes de inteligencia artificial como herramienta de apoyo, y pone de relieve la necesidad de potenciar, en el futuro, estrategias didácticas que refuercen la autoconfianza y la autoeficacia del alumnado en escenarios desasistidos.

4.3.6. Calidad del código generado y estrategias de corrección

¿Con qué frecuencia tienes que corregir el código que te genera la IA?

La pregunta 9 del cuestionario explora la asiduidad con la que el alumnado debe corregir el código proporcionado por los asistentes de inteligencia artificial. La Tabla 4.10 muestra la distribución de respuestas.

Tabla 4.10: Resultados de la pregunta 9: Frecuencia de corrección del código IA (n = 37).

Respuesta	n	% sobre muestra
Frecuentemente	16	43 %
Ocasionalmente	13	35 %
Casi nunca	6	16 %
Siempre	2	5 %

Los datos reflejan que la corrección del código generado por la IA es una práctica habitual: el 43 % de los estudiantes indica que necesita hacerlo "frecuentemente" y un 35 % lo hace "ocasionalmente". Solo un 16 % declara que "casi nunca" corrige y un 5 % afirma hacerlo "siempre". Esta distribución evidencia que, si bien la IA es una herramienta de apoyo

ampliamente utilizada, la mayoría del alumnado mantiene una actitud activa y crítica, revisando y adaptando los resultados obtenidos antes de considerarlos válidos para sus propósitos académicos.

Si la IA genera código incorrecto, ¿cómo lo corriges?

La pregunta 10 permitía seleccionar hasta dos acciones para abordar el código incorrecto generado por IA. La Tabla 4.11 muestra la distribución de las estrategias elegidas por el alumnado.

Tabla 4.11: Resultados de la pregunta 10: Estrategias de corrección (n = 37).

Respuesta	**n**	**% sobre muestra**
Pido a la IA que lo ajuste	32	46 %
Lo modifico yo mismo	30	43 %
Consulto al profesor/a	7	10 %
Pido ayuda a compañeros/as	1	1 %
Otras	0	0,0 %

El análisis de respuestas revela que la mayoría de estudiantes, ante un código incorrecto generado por IA, prefiere en primer lugar solicitar a la propia herramienta una nueva versión ajustada (46 %), mientras que un porcentaje muy similar opta por realizar directamente modificaciones manuales (43 %). Las consultas al profesorado representan un recurso de apoyo (10 %), mientras que la búsqueda de ayuda entre compañeros apenas se utiliza (1 %). Esta tendencia señala tanto la consolidación de la IA como herramienta central de iteración en el aprendizaje, como el desarrollo de autonomía técnica y capacidad de autogestión del alumnado en la resolución de problemas de programación.

4.3.7. Preferencias metodológicas

Comparado con métodos tradicionales, ¿prefieres el uso de IA?

La pregunta 11 del cuestionario explora explícitamente la preferencia del alumnado entre el aprendizaje asistido por inteligencia artificial y los métodos clásicos, entendidos como memorización, consulta de manuales o documentación técnica. La Tabla 4.12 resume las respuestas obtenidas.

Los resultados ponen de relieve una preferencia muy marcada por el aprendizaje asistido por IA: el 89 % de los estudiantes indica preferir este enfoque frente a los métodos tradicionales, mientras que únicamente 3 estudiantes optan por los sistemas clásicos y un alumno se declara indeciso. Esta tendencia mayoritaria sugiere que, tras la experiencia con herramientas de IA en el aula, el alumnado percibe ventajas significativas en cuanto a accesibilidad,

Tabla 4.12: Resultados de la pregunta 11: Preferencia de método de aprendizaje (n = 37).

Respuesta	n	% sobre muestra
Sí, prefiero IA	33	89 %
No, prefiero tradicional	3	8 %
No estoy seguro/a	1	3 %

personalización y eficiencia en la resolución de problemas de programación, consolidando la IA como el método preferente para su propio aprendizaje técnico.

¿La combinación de IA con métodos tradicionales es efectiva?

La pregunta 12 explora la percepción de eficacia derivada de la integración entre métodos tradicionales de aprendizaje y el uso de herramientas de inteligencia artificial en la formación en programación. La Tabla 4.13 recoge la valoración del alumnado.

Tabla 4.13: Resultados de la pregunta 12: Eficacia de la combinación IA con tradicional. (n = 37).

Respuesta	n	% sobre muestra
Muy efectiva	25	68 %
Efectiva	11	30 %
Poco efectiva	1	3 %
Nada efectiva	0	0,0 %

Los resultados muestran un consenso amplio respecto a la conveniencia de combinar enfoques. El 68 % de los encuestados califica la integración de IA con métodos tradicionales como "muy efectiva" y el 30 % la considera "efectiva". Prácticamente no existen opiniones negativas: sólo una respuesta indica que la combinación resulta "poco efectiva", y nadie la valora como ineficaz. En conjunto, esta tendencia reafirma la percepción de que la síntesis entre herramientas innovadoras y metodologías clásicas no solo es bienvenida, sino que potencia la comprensión y el aprendizaje en programación, facilitando una experiencia formativa más completa y flexible.

4.3.8. Distribución del tiempo de actividad

La pregunta 13 del cuestionario solicitaba al alumnado que ordenase tres actividades básicas —generar código con IA, modificar el código generado por IA y programar manualmente desde cero— en función del porcentaje de tiempo que dedicaba a cada una. La Tabla 4.14 recoge la clasificación media de las 35 respuestas.

Los resultados muestran un patrón claro en la gestión del tiempo de trabajo del alumnado. El 80 % declara emplear la mayor parte de su tiempo en utilizar la IA para generar

Tabla 4.14: Resultados de la pregunta 13: Distribución del tiempo de actividad (n = 37).

Actividad	**1.ª opción**	**2.ª opción**	**3.ª opción**
Usar IA para generar código	80 %	11 %	9 %
Modificar código generado por la IA	11 %	77 %	11 %
Escribir código manualmente (sin IA)	9 %	11 %	80 %

código, mientras que la escritura manual se encuentra relegada al último puesto para el 80 % de la muestra. La edición y ajuste del código generado por la IA se sitúa en segunda posición para la mayoría (77 %). Este reparto confirma la dinámica identificada en apartados previos: la inteligencia artificial es utilizada predominantemente como punto de partida para la producción de código, seguida de una fase de revisión y adaptación, con la programación convencional "desde cero" usada de manera minoritaria y residual.

4.3.9. Recomendaciones y visión futura

¿Recomendarías integrar IA en la enseñanza?

La pregunta 14 del cuestionario explora la disposición del alumnado a recomendar la incorporación de inteligencia artificial en los procesos docentes. La Tabla 4.15 muestra una aceptación prácticamente unánime.

Tabla 4.15: Resultados de la pregunta 14: Recomendación de integrar IA en la enseñanza (n = 37).

Respuesta	**n**	**% sobre muestra**
Sí	34	92 %
No	2	5 %
Otras	1	3 %

Estos datos denotan un grado de consenso muy elevado entre el alumnado: el 92 % recomienda incorporar la IA en la enseñanza, mientras que únicamente dos estudiantes (5 %) se manifiestan en contra y uno expresa cierta cautela, proponiendo su integración pero evitando la dependencia exclusiva. La tendencia apunta a una fuerte aceptación del uso educativo de la inteligencia artificial, interpretándola como una mejora significativa para los procesos de aprendizaje, siempre que su uso esté bien guiado y contextualizado dentro de una metodología equilibrada.

¿Crees que haber aprendido a usar la IA te puede ayudar en tu futuro profesional?

La pregunta 18 del cuestionario indaga la percepción de utilidad que el alumnado atribuye al aprendizaje del uso de inteligencia artificial ante su futuro profesional. La Tabla 4.16

resume las respuestas obtenidas.

Tabla 4.16: Resultados de la pregunta 18: Percepción de utilidad futura (n = 37).

Respuesta	**n**	**% sobre muestra**
Sí	34	92 %
No estoy seguro/a	3	8 %
No	0	0 %

Los resultados muestran un respaldo casi unánime respecto al valor profesional de las competencias adquiridas en el uso de IA: el 92 % del alumnado considera que estas habilidades le serán útiles en su carrera, mientras que ningún estudiante manifiesta una opinión negativa y sólo un 8 % expresa incertidumbre al respecto. Esta percepción generalizada indica que la formación en herramientas de inteligencia artificial no solo es valorada como un soporte académico inmediato, sino que es interpretada —de forma mayoritaria— como un capital competencial de alta relevancia para su futura inserción y desempeño en el mundo laboral.

4.3.10. Análisis cualitativo de respuestas abiertas

¿Cómo te ayuda la IA a generar páginas web?

La pregunta abierta 15 del cuestionario invitó al alumnado a describir ejemplos concretos de cómo la inteligencia artificial les había ayudado en la creación de páginas web. El análisis de contenido sobre las 34 respuestas válidas permitió identificar seis categorías principales, que se detallan en la Tabla 4.17 junto con ejemplos textuales representativos [2].

Tabla 4.17: Resultados de la pregunta 15: Ayuda de la IA al generar páginas web (n = 34).

Categoría	**n**	**Ejemplo textual**
Estructurar la página / «plantilla base»	9	«Me ayuda a crear una estructura para luego personalizarla» (ID-3)
Ahorro de tiempo / rapidez	10	«La IA lo hace más rápido que cualquier persona» (ID-4)
Corrección y depuración de errores	6	«Me señala las partes donde fallé y cómo corregirlas» (ID-6)
Generar ideas / creatividad	7	«Me da ideas cuando me quedo en blanco» (ID-18)
Mejora de diseño / estilos modernos	6	«Añade estilos modernos y bonitos de forma rápida» (ID-5)
Aprendizaje de nuevas funciones / comprensión	6	«Aprendo nuevas funciones y otras formas de crear código» (ID-11)

[2]Las respuestas completas a las preguntas de respuesta abierta se encuentran en el Anexo B

El análisis revela que el apoyo de la inteligencia artificial se percibe como especialmente útil para la generación rápida de estructuras base —o plantillas reutilizables— que sirven como punto de partida para desarrollos posteriores, así como para agilizar el proceso de creación web mediante el ahorro considerable de tiempo. Destacan también la utilidad de la IA en la corrección y depuración de errores, la aportación creativa en la generación de ideas y alternativas cuando el alumnado se queda sin recursos, la mejora en el diseño y la incorporación de estilos modernos, así como el aprendizaje efectivo de nuevas funciones o enfoques de codificación. Estas experiencias reflejan que la IA, más allá de acelerar la producción, contribuye a diversificar estrategias de aprendizaje, ampliar la comprensión técnica y democratizar el acceso a recursos actualizados y personalizados en el desarrollo web.

¿Qué habilidades crees que has mejorado gracias al uso de IA en programación?

La pregunta abierta 16 exploraba las habilidades percibidas como mejoradas gracias al uso de inteligencia artificial en programación. El análisis de las 32 respuestas permitió identificar seis categorías principales, recogidas en la Tabla 4.18 junto con ejemplos representativos de la muestra.

Tabla 4.18: Resultados de la pregunta 16: Habilidades que los estudiantes consideran haber mejorado con la IA (n = 32).

Categoría	**n**	**Ejemplo textual**
Comprensión y lectura de código	14	«He aprendido nuevas funciones y formas de crear código» (ID-11)
Eficiencia / ahorro de tiempo	10	«Cosas que tardaba más en realizar ahora tardo minutos» (ID-5)
Depuración y corrección de errores	9	«Me señala las partes donde fallé y cómo corregirlas» (ID-6)
Mejora en la redacción de *prompts*	3	«He mejorado la creación de prompts» (ID-11); «Uso prompts más avanzados» (ID-19)
Creatividad e inspiración de ideas	6	«Me da ideas cuando me quedo en blanco» (ID-18)
Competencias específicas (HTML / CSS / JS)	4	«He mejorado sustancialmente en el uso de CSS» (ID-2)

Los resultados resaltan que la habilidad más frecuentemente identificada fue la comprensión y lectura de código, señalada como principal mejora por 14 estudiantes. Un alto número de participantes también menciona una mayor eficiencia y ahorro de tiempo en la realización de tareas, así como la mejora en la depuración y corrección de errores en los programas desarrollados. Una proporción significativa reconoce haber perfeccionado la redacción de *prompts* —esencial para la interacción efectiva con herramientas de IA— y destaca el incremento en creatividad y generación de ideas. Finalmente, se observa que parte del alumnado atribuye la mejora directa en competencias técnicas específicas (HTML, CSS o JavaScript) al uso de

IA. Estos resultados ponen de manifiesto que la integración de la inteligencia artificial en el proceso de aprendizaje ha impactado en competencias tanto transversales como técnicas clave en la formación en programación web.

¿Qué desafíos has enfrentado al integrar IA en tu proceso de aprendizaje?

El análisis de contenido de las 32 respuestas a la pregunta 17 permitió identificar seis tipos principales de dificultades asociadas al uso de inteligencia artificial en el proceso de aprendizaje, que se resumen en la Tabla 4.19 junto a ejemplos textuales ilustrativos.

Tabla 4.19: Resultados de la pregunta 17: Principales desafíos percibidos al trabajar con IA (n = 32).

Categoría	**n**	**Ejemplo textual**
Formular buenos *prompts* / que la IA entienda	7	«Tener que pedirle muchas veces que arregle lo que se le pide» (ID-1)
Dependencia excesiva / pereza	4	«La IA te hace muy dependiente si no tienes conocimientos básicos» (ID-4)
Errores o alucinaciones; necesidad de validar	6	«A veces la IA me da respuestas que parecen correctas pero tienen errores» (ID-14)
Comprender el código generado / curva de aprendizaje	7	«Intentar comprender código generado sin aún tener mucha base» (ID-3)
Limitaciones de acceso / cuota	2	«El mayor desafío suele ser cuando ChatGPT llega al límite diario» (ID-23)
Ningún desafío reseñable	6	«Casi ninguno, ya que todo es más fácil y lo difícil te lo explica» (ID-8)

Los resultados evidencian que uno de los desafíos más frecuentes es la redacción de *prompts* claros y específicos, de modo que la IA comprenda y ejecute correctamente la petición: varios estudiantes destacan la necesidad de iterar y ajustar sus instrucciones para lograr el resultado esperado. Otra dificultad señalada es el riesgo de desarrollar una dependencia excesiva o actitud pasiva, especialmente en quienes carecen de bases sólidas de programación. La aparición de errores o "alucinaciones" en las soluciones propuestas por la IA exige una actitud constante de validación y análisis crítico por parte del usuario. Por otro lado, algunos participantes subrayan el reto de comprender y asimilar fragmentos de código generados automáticamente sin contar aún con los fundamentos necesarios, lo que puede ralentizar el aprendizaje real. Las limitaciones técnicas, como las restricciones de cuota de uso en herramientas populares (por ejemplo, ChatGPT), también han sido reseñadas como obstáculo puntual. Por último, una parte del alumnado manifiesta no haber experimentado problemas significativos, apreciando ventajas claras en cuanto a accesibilidad y explicación de conceptos gracias al apoyo de la inteligencia artificial.

¿Cómo crees que la IA cambiará la forma de programar y de enseñar a programar?

La pregunta abierta 19 explora las percepciones del alumnado sobre el impacto futuro de la inteligencia artificial en el aprendizaje y la práctica de la programación. El análisis de las 30 respuestas permitió identificar cinco grandes ideas-fuerza, que se resumen en la Tabla 4.20 junto con citas textuales destacadas.

Tabla 4.20: Resultados de la pregunta 19: Cómo la IA cambiará la forma de aprender y programar (n = 30).

Categoría	**n**	**Ejemplo textual**
Aprendizaje y programación más accesibles, rápidas y creativas	14	«La IA hará la programación más accesible, automatizando tareas repetitivas» (ID-13)
El *prompting* será la nueva "sintaxis"	4	«El lenguaje natural será el nuevo lenguaje de programación; saber crear prompts precisos será casi más importante que saber programar» (ID-2)
La IA debe ser un complemento, hace falta base para validar	4	«Es importante tener una base de programación; así la IA se usa como complemento para hacer el trabajo más eficiente» (ID-5)
Riesgo de dependencia y pérdida de trabajos / habilidades	6	«Los alumnos se volverán muy dependientes de la IA» (ID-11); «Acabará con los programadores de bajo y medio nivel» (ID-30)
Mejora de la comprensión gracias a explicaciones interactivas	2	«Aprenderemos más rápido y conseguiremos la información necesaria» (ID-6)

La categoría más mencionada hace referencia a la accesibilidad y rapidez que entrañarán los flujos de trabajo basados en IA, augurando un proceso creativo más ágil y la automatización de tareas repetitivas. Varias respuestas avanzan que el dominio del *prompting* —la capacidad de interactuar con la IA en lenguaje natural— se convertirá en una habilidad fundamental, equiparable o incluso más valiosa que el dominio de lenguajes de programación tradicionales. Un grupo de estudiantes subraya la importancia de conservar una base conceptual sólida de programación, para usar la IA de forma crítica y evitar errores, reafirmando la función de la IA como complemento y no reemplazo. Por otro lado, una parte del alumnado expresa preocupación por el riesgo de dependencia excesiva, la posible pérdida de habilidades técnicas tradicionales y el eventual desplazamiento laboral de perfiles menos

cualificados en programación. Finalmente, algunos participantes destacan el potencial de la IA para facilitar la comprensión de conceptos complejos gracias a explicaciones interactivas y personalizadas, anticipando una aceleración y democratización del aprendizaje técnico en entornos educativos y profesionales.

Sugerencias para mejorar el uso de IA en clase

La última pregunta abierta del cuestionario recogía recomendaciones del alumnado para optimizar la integración de la inteligencia artificial en el aula. El análisis cualitativo de las 27 respuestas permitió identificar seis líneas principales de mejora, que se resumen en la Tabla 4.21 junto con ejemplos representativos.

Tabla 4.21: Resultados de la pregunta 20: Sugerencias para mejorar el uso de IA en clase (n = 27).

Categoría	**n**	**Ejemplo textual**
Formación en *prompting* avanzado	9	«Saber crear prompts precisos será casi más importante que saber programar» (ID-2)
Fomentar pensamiento crítico / validar la salida	6	«Comprobar si la información que da la IA es correcta; corregir errores» (ID-11)
Uso equilibrado IA + base teórica (evitar dependencia)	5	«Primero enseñar las bases y luego introducir la IA» (ID-21)
Aprendizaje basado en proyectos reales con IA	5	«Plantear retos para comprobar lo aprendido» (ID-6); «Crear la página con ChatGPT y luego analizar el código» (ID-14)
Catálogo de herramientas / visión global de IA	2	«Listado de IA's para cada uso» (ID-13); «Exposiciones explicando los tipos de IA» (ID-26)
Extender IA a asignaturas teóricas / tareas repetitivas	2	«Implementarla también en asignaturas más teóricas» (ID-24)

Las propuestas formuladas por el alumnado se centran principalmente en la necesidad de profundizar en la formación sobre ingeniería de *prompts*, señalando la habilidad de formular preguntas precisas como elemento clave para aprovechar el potencial de la IA en programación. Igualmente, destacan la importancia de fomentar el pensamiento crítico y la validación activa de las respuestas proporcionadas por la IA, con el objetivo de evitar la aceptación acrítica o la copia directa de código. Muchas respuestas insisten en la conveniencia de mantener un uso híbrido, en el que la base teórica y conceptual preceda al uso intensivo de IA, evitando así una dependencia excesiva de la automatización. Se propone, además, dinamizar el aprendizaje mediante el planteamiento de proyectos o retos reales donde la IA actúe como herramienta de apoyo y análisis posterior. Otras sugerencias minoritarias abogan por ampliar

el repertorio de herramientas conocidas, con exposiciones o catálogos que ayuden a seleccionar la IA más adecuada según el contexto, así como explorar su integración en asignaturas teóricas o aprovechamiento para tareas repetitivas, ampliando así su alcance formativo.

4.4. Análisis de los ejercicios prácticos

Tras el análisis de las percepciones estudiantiles mediante el cuestionario post-intervención, resulta fundamental examinar las evidencias concretas del proceso de aprendizaje a través de los ejercicios prácticos realizados durante la fase de introducción a la IA. Estos ejercicios constituyen un componente esencial para evaluar la adquisición real de competencias en el uso pedagógico de asistentes de inteligencia artificial.

Los ejercicios prácticos se estructuraron en dos modalidades complementarias, diseñadas para desarrollar diferentes aspectos de la competencia y autonomía en el uso de IA generativa:

1. **Ejercicios de ingeniería de *prompts*:** Centrados en el desarrollo de habilidades para formular instrucciones precisas y efectivas dirigidas a modelos de lenguaje, con progresión desde tareas básicas (generación de tablas HTML) hasta elementos avanzados (animaciones con CSS o JavaScript).

2. **Ejercicios de depuración y comparación de herramientas:** Orientados al manejo estratégico de herramientas especializadas, comparación entre modelos de IA, personalización de agentes y resolución de problemas mediante depuración asistida.

Ambas modalidades requirieron documentación completa del proceso, incluyendo capturas de pantalla, justificación de decisiones, descripción de iteraciones y reflexión sobre los resultados obtenidos. Esta aproximación permite analizar no solo la efectividad de las soluciones generadas, sino también la evolución del pensamiento metacognitivo y la capacidad de autorregulación del aprendizaje.

4.4.1. Ingeniería de *prompts*

La ingeniería de *prompts* representa una competencia emergente fundamental en el paradigma de la programación asistida por inteligencia artificial. Los ejercicios diseñados siguieron una progresión didáctica estructurada: desde la generación de estructuras HTML básicas (tablas con datos específicos) hasta la implementación de funcionalidades avanzadas como transiciones CSS y animaciones con *keyframes*[3].

Cada ejercicio requirió que los estudiantes formularan instrucciones en lenguaje natural suficientemente precisas para obtener código funcional que cumpliera especificaciones técnicas concretas. La documentación del proceso incluía la transcripción literal de los *prompts*

[3]El boletín de ejercicios de ingeniería de prompts se puede ver en el Anexo C

utilizados, capturas de pantalla de las respuestas obtenidas, descripción de las iteraciones necesarias para refinar los resultados y reflexión sobre las estrategias empleadas.

Cada estudiante debía (i) formular un *prompt* inicial especificando el resultado deseado, (ii) iterar para ajustar la salida (añadir restricciones, corregir malentendidos, pedir alternativas), y (iii) verificar de forma básica el comportamiento logrado (captura, antes/después, breve prueba).

Con el fin de estimar la utilidad pedagógica, codificamos en una matriz binaria (1/0) la presencia de siete indicadores en el trabajo de cada estudiante:

- **Intencionalidad**: el estudiante formula un objetivo claro antes de interactuar con la IA.
- **Iteración con mejora**: realiza ajustes en el *prompt* o en el código basados en la respuesta previa de la IA.
- **Diagnóstico/depuración**: identifica la causa de un error o el punto a mejorar antes de aplicar cambios.
- **Auto-explicación**: describe con sus propias palabras cómo funciona la solución o el cambio realizado.
- **Verificación básica**: comprueba que la solución obtenida funciona (p. ej., ejecutando el código o revisando la visualización).
- **Autonomía crítica**: decide de forma justificada aceptar, rechazar o modificar la propuesta de la IA.
- **Reflexión sobre límites de la IA**: reconoce errores, sesgos o limitaciones de la herramienta utilizada.

Tabla 4.22: Evidencias de utilidad pedagógica — Ingeniería de *prompts* (global, n=45)

Indicador	**%**
Formulación intencional	97,8
Iteración con mejora	84,4
Diagnóstico / depuración	71,1
Auto-explicación	17,8
Verificación básica	82,2
Autonomía crítica	37,8
Reflexión sobre límites de la IA	44,4

Con un total de 45 trabajos analizados[4], los resultados (en la tabla 4.22) muestran un uso de la IA mayoritariamente dirigido y activo. La formulación intencional roza la universalidad

[4]Algunos alumnos/as no entregaron el trabajo.

(97,8 %): casi todo el alumnado inicia la interacción con una meta clara, lo que confirma que la IA se emplea como herramienta orientada a un objetivo concreto (no como exploración aleatoria). En los textos aparecen metas muy explícitas y operativas, por ejemplo: «Quiero que crees un archivo HTML con una tabla que tenga: Nombre, Precio y Cantidad en stock, y que ocupe todo el ancho de la página», «Quiero que el menú esté junto y no se separe; al pasar el ratón debe cambiar el color de toda la casilla, no solo del texto», o «Hazme un botón con el texto "Hazme grande!" que tarde 2 segundos en agrandarse y que cambie a rojo con letras blancas».

La iteración con mejora es también muy elevada (84,4 %), señal de que, ante salidas incompletas o imprecisas, el alumnado refina sus *prompts* o el código hasta aproximar el resultado deseado. Lo formulan con secuencias como: «Primeramente me realizó esto; así que le indiqué lo siguiente para que lo corrigiera. Con este cambio, ahora sí ha quedado como se buscaba», o «Este prompt estaba mal: no pedía una frase y no tenía botones, así que vuelvo a intentar construirlo completamente con una nueva versión». Acompañando esa iteración, siete de cada diez trabajos evidencian diagnóstico/depuración (71,1 %): se explicita qué falla antes de pedir corrección, lo que sugiere una comprensión operativa del comportamiento del código. Por ejemplo: «Al crear esto faltaban las líneas de separación, así que le pedí que las añadiera», «El menú me abre una ventana vacía; quiero que los enlaces redirijan correctamente a una página web», o «He tenido que cambiar el `href` del CSS porque el nombre del archivo no coincidía y por eso no me cargaba los estilos».

La verificación básica aparece en el 82,2 % de los casos (pruebas mínimas, capturas o comprobaciones visuales), lo que indica hábitos de control funcional. Abundan confirmaciones que muestran comprobación directa del resultado: «Poniendo el código que me ha dado, ha quedado de la siguiente manera», «El resultado es exactamente lo que buscaba; no ha sido necesaria ninguna corrección», «Ahora sí que lo conseguimos» o «Funcionó a la primera y no tuve que hacer nada».

En cambio, la auto-explicación —poner por escrito por qué la solución funciona o qué hace cada cambio— es minoritaria (17,8 %), y conviene reforzarla para consolidar el aprendizaje conceptual más allá de la ejecución. Cuando aparece, se ve con claridad: «Aclaraciones: `display: inline-block` para que la transformación se aplique correctamente. Transformación: `scaleX(-1)` es la que invierte el texto», o «Para que la imagen gire sin parar se tienen que utilizar `@keyframes` y la propiedad `animation`».

En el plano metacognitivo, la autonomía crítica (37,8 %) muestra que una parte del alumnado ya toma decisiones informadas (aceptar, adaptar o descartar propuestas de la IA). Se leen intervenciones donde el estudiante decide y actúa: «Solo es necesario eliminar el título y los estilos CSS para que quede de manera correcta; resultado final después de los cambios manuales», «He modificado el *padding* de `form-container` a 50 px para que respire mejor», o «He cambiado yo manualmente y he quitado las letras del medio y los colores que no quería». La reflexión sobre límites alcanza el 44,4 %, evidenciando que casi la mitad identifica fallos,

sesgos o restricciones de las herramientas y actúa en consecuencia. Se reconoce, por ejemplo: «Como ChatGPT no tiene acceso a la carpeta, solo tuve que agregar manualmente la imagen para que funcionase», «Se equivocó al centrar los campos de introducción de datos; le indiqué cómo corregirlo y lo arregló», o «ChatGPT no me ponía los colores que yo pedía; tuve que repetir el proceso varias veces hasta que aplicó el color al fondo y no a los bordes». Ambos indicadores son positivos, pero aún tienen margen de mejora.

4.4.2. Depuración de código

Este bloque se centró en localizar y corregir errores de HTML/CSS/JS con ayuda de IA y en comparar modelos ante un mismo *prompt*. Las tareas incluyeron: (i) Corrección de etiquetas/propiedades y ajustes del DOM o estilos; (ii) Uso del chat de la IA, agentes o una imagen como base, para obtener un borrador y refinarlo; (iii) Comparación de respuestas (Blackbox, ChatGPT, Claude, Gemini) y justificación de elecciones[5].

Los estudiantes debieron documentar no solo los resultados obtenidos, sino también sus decisiones: selección de modelos según el contexto, configuración de agentes personalizados, análisis crítico de las diferencias entre herramientas y evaluación de la calidad del código generado. Esta aproximación permite examinar el desarrollo de competencias metacognitivas específicas para el uso eficiente y crítico de ecosistemas de IA especializados en programación.

Como en el ejercicio anterior, se codificaron los 7 indicadores de comportamiento reflexivo, con el fin de evaluar el uso pedagógico de la herramienta en la mejora de la comprensión y la práctica de la programación web.

Tabla 4.23: Evidencias de utilidad pedagógica — Depuración con IA (n=35)

Indicador	**%**
Formulación intencional	61,8
Iteración con mejora	67,6
Diagnóstico / depuración	76,5
Auto-explicación	23,5
Verificación básica	82,4
Autonomía crítica	85,3
Reflexión sobre límites de la IA	76,5

La Tabla 4.23 resume 35 entregas[6] y muestra un patrón nítido: los porcentajes más altos en "Autonomía crítica" (85,3 %) y "Verificación básica" (82,4 %) indican que la mayoría de los estudiantes contrastaron las salidas de la IA, compararon modelos y tomaron decisiones sobre cómo proceder. Este comportamiento se aprecia en comentarios como «Blackbox me daba problemas, así que la haremos con Copilot» o «He optado por continuar el trabajo de

[5]El boletín de ejercicios de depuración de código se puede ver en el Anexo D

[6]Los estudiantes de 2ºASIR no hicieron esta tarea por falta de tiempo y algunos estudiantes de los otros grupos no entregaron el trabajo.

forma manual», donde la herramienta se integra en un flujo de trabajo que incluye prueba, evaluación y, si es necesario, sustitución o edición humana. Paralelamente, el 76,5 % en "Diagnóstico/depuración" y en "Reflexión sobre límites" sugiere que la IA fue usada tanto como detector de errores como objeto de mirada crítica; se pueden leer frases como «Blackbox no funciona» o «A pesar de indicarle que añadiera CSS, no lo aplicó», que evidencian una comprensión de que estos sistemas fallan y requieren supervisión. En contraste, la "Auto-explicación" alcanza sólo un 23,5 %: se corrige mucho, pero pocas veces se explicita el por qué técnico de las soluciones, lo que señala una oportunidad curricular para reforzar el razonamiento conceptual que acompaña a la depuración.

En "Formulación intencional" (61,8 %), los estudiantes suelen traducir objetivos de diseño en pedidos dirigidos a la IA, normalmente descomponiendo el problema en dos pasos (estructura y estilo). Aparecen enunciados claros como «Genera el código HTML de una web con una cabecera, una sección principal con un texto, y un pie de página» y su continuación orientada al estilo, «Genérame el CSS de este código para que la cabecera tenga un fondo azul y el texto de la sección principal sea de color gris oscuro con un tamaño de letra de 18px». Este tipo de formulaciones revelan comprensión de los requisitos y facilitan que la IA entregue un punto de partida alineado con el resultado esperado.

La "Iteración con mejora" (67,6 %) crece cuando el alumnado trata las respuestas como hipótesis a refinar. El ciclo de ajuste se documenta con expresiones como «Me lo dio por partes, por lo que le pedí que me lo juntara», donde el propio *prompt* se reajusta para corregir la forma de entrega; con ampliaciones graduales de criterios como «Ahora le especificaremos que se adapte al tipo de pantalla»; o con evidencia de persistencia y cambio de estrategia, «Tras varios intentos y distintos prompts no me daba lo que quería», que desemboca en probar otro modelo o reformular la consigna. Esta dinámica itera entre borrador y verificación, y explica la mejora del resultado sin perder el control humano del proceso.

La "Auto-explicación" (23,5 %) es la dimensión más débil. Hay ejemplos en los que el estudiante justifica el cambio con conocimiento declarativo, como «Las etiquetas de encabezado válidas son de `<h1>` a `<h6>`» o «Explicación: la propiedad estaba mal escrita». Incluso se verbaliza el sentido estructural de la sintaxis con frases como «Añadí la llave { porque es necesaria para definir el bloque de estilos». El bajo porcentaje sugiere que, aunque se realizan arreglos correctos, aún falta verbalizar más el razonamiento técnico que los hace necesarios.

Estos resultados apuntan a un uso ya de la IA como co-asistente de depuración y prototipado, con fortaleza en verificación y toma de decisiones, y un área de mejora clara en la explicitación del razonamiento técnico.

Capítulo 5

Discusión

5.1. Síntesis de hallazgos

La evidencia triangulada (calificaciones, cuestionario y análisis de entregas) describe un aprendizaje asistido por IA que acelera la producción y apoya la comprensión, promueve prácticas de verificación y decisiones críticas sobre el código, y desplaza el tiempo hacia generación y refactorización con IA; al mismo tiempo, visibiliza la necesidad de reforzar la auto-explicación técnica y la confianza sin asistencia. Todos estos resultados permiten responder a las preguntas de investigación y a los objetivos planteados en el Capítulo 1.

5.1.1. Respuesta a las preguntas de investigación

En primer lugar, la integración de asistentes de IA generativa en la enseñanza de la programación web en formación profesional ha producido un impacto observable en el rendimiento académico del alumnado. El análisis comparativo de las cohortes revela un desplazamiento de las calificaciones hacia tramos superiores (media ascendente de 7,4 a 8,0), la desaparición de notas por debajo de 6 y una ligera reducción en la dispersión de las puntuaciones. Aunque los contrastes estadísticos (t de Student) entre cohortes no alcanzan la significación, los tamaños del efecto se consideran moderados, lo que indica mejoras académicas compatibles con la incorporación de IA, si bien estas requieren muestras mayores para confirmación inferencial robusta. La mejora en la media y la reducción de las bajas calificaciones sugieren que la IA facilita la superación de los mínimos académicos a mayor número de estudiantes, trasladando el foco docente de la simple ejecución de fragmentos de código a la gestión integral de proyectos y la documentación del proceso.

Diversos factores aparecen como moduladores de la relación entre el uso de IA y el desarrollo de competencias técnicas: la capacidad de redacción de *prompts*, la actitud crítica ante el código generado y la habilidad para iterar y depurar procesos juegan un papel fundamental. Se ha constatado que la mejora en la competencia en *prompting* es generalizada, pero la autoconfianza programando sin asistencia sigue siendo limitada, subrayando la importancia

de fortalecer la base conceptual independiente de las herramientas. Asimismo, la autonomía crítica a la hora de verificar, modificar o rechazar el código de la IA se consolida como una competencia transversal esencial, superando el simple uso instrumental de las herramientas y promoviendo la auto-regulación y la toma de decisiones técnicas justificadas.

Por otro lado, las estrategias pedagógicas que han demostrado mayor eficacia en la integración de IA y metodología tradicional se caracterizan por el diseño de tareas reales e integradas en las que el proceso sea objeto de evaluación, no solo el producto. Esto implica documentar los procesos realizados, realizar comparaciones entre soluciones automáticas y manuales, justificar las decisiones tomadas y alternar la programación con y sin apoyo de asistentes automáticos. El enfoque en la realización de proyectos, la exigencia de autoexplicación técnica y la valoración explícita de la reflexión sobre los límites, errores y aciertos de la IA se revelan como elementos clave para maximizar el aprendizaje significativo y reducir los riesgos de dependencia instrumental. La evidencia muestra que la combinación híbrida IA-tradicional cuenta con elevada aceptación entre el alumnado y potencia la adquisición de competencias profesionales transferibles al sector laboral actual.

5.1.2. Respuesta a los objetivos del trabajo

O1: Impacto en el rendimiento académico. Como ya hemos comentado anteriormente, en la comparación por cohortes se observa un desplazamiento de la distribución de calificaciones hacia tramos superiores. Aunque este patrón es compatible con ganancias académicas moderadas, se requieren muestras mayores para su verificación inferencial.

O2: Percepción del alumnado e impacto formativo. La adopción de ChatGPT es universal, con uso extendido de otras herramientas; casi la mitad declara usar IA siempre y otra proporción similar de forma frecuente, lo que evidencia integración rutinaria en el flujo de trabajo. A nivel formativo, casi todo el alumnado afirma que la IA les ha ayudado a entender cómo programar. En contraste, aparece una autoeficacia más débil sin asistencia: una mayoría se siente "algo inseguro/a" y unos pocos estudiantes "nada seguros/as" al programar sin IA, mostrando un área de mejora para reforzar la confianza en escenarios desasistidos.

O3: Competencias desarrolladas con tareas prácticas- El análisis de contenido de las entregas documentadas muestra uso competente y dirigido de los asistentes:

- En ingeniería de *prompts* destacan la formulación intencional, la iteración con mejora y la verificación básica, mientras que la auto-explicación técnica queda rezagada.

- En depuración, sobresalen la autonomía crítica, la verificación y el diagnóstico, junto a la reflexión sobre límites de la IA, manteniéndose de nuevo más baja la auto-explicación. Este patrón sugiere progreso en competencias de planificación, control y evaluación del

proceso con IA, y una oportunidad curricular para hacer explícito el razonamiento técnico subyacente.

O4: Patrones de uso, prompting y corrección del código.

- El alumnado reporta mejora clara en la escritura de *prompts* y una dificultad percibida neutra, rasgos congruentes con la progresión didáctica impartida.
- La corrección del código generado por IA es habitual y se afronta mediante reiteración con la IA y edición manual, apoyándose poco en terceros.
- En la gestión del tiempo, la gran mayoría sitúa "usar IA para generar código" como actividad principal, seguida de "modificar lo generado", quedando "escribir manualmente desde cero" como minoritaria, lo que confirma el flujo *prompt→iteración→ajuste* como práctica dominante.

O5: Cambios metodológicos y rol docente. La intervención no se limita a "permitir IA", sino que reconfigura tareas y evaluación: se pasa de ejercicios fragmentarios a proyectos reales (más próximos a lo que el alumnado enfrentará en una empresa) y documentados, se eleva la exigencia (estructura, estilo, publicación, justificación técnica), y las rúbricas incorporan verificación, reflexión y uso responsable de IA, alineando docencia y evaluación con el nuevo paradigma profesional. Este cambio sitúa al docente en un rol de "andamiaje metacognitivo", más que de mera corrección de producto final.

O6: Preferencias metodológicas y transferibilidad. Tras la experiencia, la mayor parte del alumnado prefiere el aprendizaje asistido por IA frente a métodos exclusivamente tradicionales, y perciben efectiva o muy efectiva la combinación IA+tradicional, apoyando un modelo híbrido que preserve prácticas manuales y fortalezca la autoría. Además, recomiendan integrar IA en la enseñanza y anticipan utilidad profesional de las competencias adquiridas, lo que respalda la viabilidad y aceptación del enfoque en contextos similares de FP.

5.2. Interpretación pedagógica de los resultados

Aunque la baja potencia muestral no permite confirmar inferencialmente un beneficio académico, es necesario recalcar que se mantuvo la duración de pruebas elevándose la dificultad y se reorganizó la evaluación hacia proyectos integrados (en algunos casos inter-módulos) y documentados, por lo que el incremento en notas no puede atribuirse a "facilitación" del examen o los trabajos, si no a procesos de trabajo distintos y más exigentes.

El patrón de indicadores en las entregas revela desarrollo de destrezas autorregulatorias asociadas al proceso: planificación (formulación intencional), control (iteración con mejora,

diagnóstico/depuración) y evaluación (verificación, autonomía crítica). Este patrón sugiere que el alumnado aprendió a gestionar la IA (pedir, revisar, aceptar/rechazar) más que a delegar a ciegas. La autoexplicación técnica queda como cuello de botella: se llega al resultado y se contrasta su funcionamiento, pero cuesta hacer explícito el porqué, lo que orienta una necesidad de andamiaje para verbalizar principios y justificar cambios. La exigencia de documentar procesos (capturas, iteraciones, decisiones) fue clave para visibilizar estas conductas y poder codificarlas.

Las respuestas abiertas refuerzan que la IA ayuda a entender cómo programar —"entiendo mejor el código", "ahora comprendo para qué sirve cada sección", "me corrige y explica cómo arreglarlo"—, lo que sugiere que la interacción con la IA funcionó como tutor de lectura de código y de prácticas de prueba, más que como simple generador de soluciones. No obstante, la brecha entre comprender con IA y explicar por escrito el porqué persiste (baja autoexplicación), señalando un objetivo formativo explícito para próximas intervenciones.

Las conductas observadas en depuración y las estrategias de corrección reportadas indican que el alumnado decide, compara modelos y alterna entre reajustar via IA y corregir por cuenta propia, desplazando el foco del "aceptar lo generado" al "orquestar el proceso". Esta autonomía es prometedora, pero debe consolidarse en contextos sin asistencia: la autoeficacia declarada al programar sin IA es modesta, de modo que la autonomía con IA no equivale todavía a independencia sin IA. En términos pedagógicos, conviene alternar tareas con y sin IA, y reforzar criterios de calidad y autoexplicación para transferir decisiones y validaciones al plano conceptual propio.

5.3. Limitaciones y decisiones metodológicas

- **Tamaño muestral y potencia:** Los contrastes entre cohortes arrojan tamaños del efecto moderados sin alcanzar significación estadística, con potencias observadas bajas y un umbral a priori cercano a $n \approx 100$ estudiantes por grupo. En consecuencia, la ausencia de significación no invalida la señal de mejora, pero impone cautela y demanda replicación con muestras mayores.

- **Ausencia de aleatorización y diseño cuasi-experimental:** El estudio compara cohortes históricas consecutivas del mismo centro, impartidas por la misma docente y con el mismo temario base, incorporándose la IA solo en 2024/2025. Este diseño reduce variabilidad docente y curricular, pero no controla sesgos. Además, 1.º ASIR no dispone de cohorte 2023/2024 comparable, de modo que los contrastes inferenciales se limitan a 2.º SMR y 2.º ASIR. Estas condiciones impiden atribuciones causales fuertes y recomiendan interpretar los hallazgos como asociaciones compatibles con un efecto de la intervención.

- **Instrumentación y criterios de evaluación:** La evaluación sin IA se centró en frag-

mentos de código, mientras que con IA se pasó a proyectos integrados, documentación y valoración explícita del uso responsable de IA. El cambio de instrumento evaluativo introduce un posible sesgo de instrumentación que puede influir en la distribución de calificaciones; por ello, el desplazamiento hacia tramos altos debe leerse junto con la evidencia de proceso y no como efecto "puro" de la IA.

- **Codificación cualitativa:** El análisis de ejercicios se basó en una matriz binaria de siete indicadores previamente operativizados. Esta decisión facilita la trazabilidad y el conteo, pero simplifica la intensidad y calidad de las evidencias (p. ej., una autoexplicación mínima y una argumentación sólida puntúan igual).
- **Heterogeneidad y comparabilidad de muestras:** Las cohortes difieren en tamaño y composición, con sobrerrepresentación masculina y variabilidad etaria entre grupos; aunque el análisis principal se circunscribe a módulos presentes en ambos cursos, estas diferencias pueden modular la respuesta a la intervención.

5.4. Cambios docentes para una enseñanza de programación asistida por IA

Enfoque curricular. Reequilibrar los resultados de aprendizaje desde la producción de código hacia la orquestación del proceso: lectura y comprensión de código, diagnóstico/depuración, verificación de soluciones y refactorización. La IA se concibe como herramienta para apoyar estas prácticas, no como sustituto del razonamiento.

Diseño de tareas. Priorizar tareas integradas con trazabilidad del proceso (qué se pidió, qué se aceptó/rechazó y por qué), alternando escenarios con y sin IA para proteger la transferencia conceptual. Incluir comparaciones razonadas entre soluciones (de IA y propias) sin fijar dependencia de una herramienta concreta.

Evaluación. Valorar explícitamente el proceso además del producto: intencionalidad, iteración con mejora, diagnóstico, verificación y autoexplicación técnica. Incorporar evidencias de comprobación (tests/validadores), pequeñas defensas orales o escritas sobre decisiones clave, y trazabilidad del uso de IA. Mantener criterios estables y comunicados al alumnado.

Integridad y ética. Establecer una política clara y positiva de uso de IA (qué está permitido, cómo citarla, cómo evidenciar verificación). Exigir transparencia en el uso y promover prácticas que minimicen dependencias y alucinaciones.

Calidad y sostenibilidad. Introducir estándares básicos de calidad (accesibilidad, seguridad, legibilidad, rendimiento) como parte natural del trabajo con o sin IA.

Capítulo 6

Conclusiones y trabajo futuro

6.1. Conclusiones

La presente investigación ha abordado una cuestión fundamental en el panorama educativo actual: el impacto pedagógico de la inteligencia artificial generativa en la enseñanza de programación web en formación profesional. Los resultados obtenidos, derivados de un diseño cuasi-experimental con estudiantes de los ciclos SMR y ASIR, ofrecen evidencias sobre las posibilidades y desafíos de integrar estas herramientas emergentes en contextos educativos reales.

Los hallazgos revelan una mejora moderada en las calificaciones de los estudiantes que emplearon asistentes de IA. El desplazamiento de la distribución hacia tramos superiores y la desaparición de calificaciones por debajo de 6 sugieren un efecto académico favorable. Aunque los contrastes estadísticos no alcanzaron significación debido al tamaño muestral limitado, los tamaños del efecto moderados son prometedores y coherentes con mejoras sustanciales que requieren validación en estudios con muestras mayores.

Resulta especialmente relevante que esta mejora académica se produjo en un contexto de mayor exigencia evaluativa. El paso de ejercicios fragmentarios a proyectos web completos, documentados y publicados representa un salto cualitativo en la complejidad de las tareas, lo que refuerza la validez de los resultados obtenidos. La IA no facilitó el proceso evaluativo, sino que permitió a los estudiantes abordar desafíos de mayor complejidad profesional.

Los resultados evidencian una apropiación tecnológica exitosa y una valoración claramente positiva. La adopción universal de herramientas especializadas confirman la naturalidad con la que el alumnado integra estas tecnologías en sus flujos de trabajo. Además, la percepción de utilidad formativa es abrumadoramente positiva. Esta preferencia no implica rechazo a lo tradicional, si no valoración de un enfoque híbrido.

En cuanto al rol docente, la investigación confirma una transformación profunda. El profesorado evoluciona desde instructor de sintaxis hacia facilitador de metacognición, enfocándose en desarrollar capacidades de análisis, toma de decisiones y validación crítica. Esta

transición requiere nuevas competencias docentes: dominio de herramientas de IA, diseño de tareas complejas, evaluación de procesos, y mediación en dilemas éticos.

Esta investigación demuestra que la integración estructurada de IA generativa en la enseñanza de programación web puede generar beneficios pedagógicos significativos, siempre que se conciba como herramienta de mediación cognitiva y no como sustituto del pensamiento crítico. El éxito de la intervención radica en el desplazamiento del foco educativo desde la producción mecánica de código hacia el desarrollo de competencias de orden superior: análisis, síntesis, evaluación y creatividad.

Los estudiantes emergen como orquestadores competentes de procesos tecno-humanos complejos, capaces de aprovechar la potencia de la IA mientras mantienen control crítico sobre los resultados. Esta capacidad híbrida –combinando intuición humana con asistencia algorítmica– representa una competencia profesional fundamental en el panorama tecnológico contemporáneo.

La investigación confirma que el futuro de la educación en programación no reside en la resistencia a la IA ni en su adopción acrítica, sino en la construcción de pedagogías que cultiven simultaneamente competencia tecnológica y autonomía intelectual. Este equilibrio complejo pero necesario constituye el desafío y la oportunidad de la educación tecnológica en la era de la inteligencia artificial.

6.1.1. Implicaciones para la formación profesional

Los resultados tienen implicaciones directas para el sistema de formación profesional español. La investigación demuestra que es posible integrar IA generativa de forma pedagógicamente productiva en ciclos formativos, siempre que se mantenga un enfoque estructurado y reflexivo.

La preferencia estudiantil por modelos híbridos respalda estrategias que preserven fundamentos técnicos mientras aprovechan las ventajas de la automatización inteligente. Esta conclusión es especialmente relevante dado que la mayor parte de los estudiantes anticipa utilidad profesional de las competencias adquiridas en IA.

6.1.2. Contribución al conocimiento científico

Esta investigación contribuye al escaso corpus de estudios sobre IA en formación profesional, un nivel educativo sistemáticamente infrainvestigado. Los resultados desafían narrativas simplistas sobre la IA como amenaza o panacea educativa, evidenciando que su valor pedagógico depende críticamente del diseño instruccional y la mediación docente.

La triangulación de evidencias (calificaciones, percepciones, análisis de proceso) fortalece la validez de los hallazgos y ofrece un marco metodológico replicable para investigaciones futuras. Los indicadores de utilidad pedagógica desarrollados (formulación intencional, iteración con mejora, diagnóstico/depuración, auto-explicación, verificación, autonomía crítica,

reflexión sobre límites) constituyen un aporte conceptual para la evaluación de competencias en programación asistida por IA.

6.1.3. Reflexión crítica sobre limitaciones y alcance

Los hallazgos deben interpretarse considerando las limitaciones inherentes al diseño cuasi-experimental empleado. La ausencia de aleatorización, el tamaño muestral limitado, y la especificidad del contexto (centro privado-concertado, docente única, módulos específicos) condicionan la generalización de resultados. No obstante, la coherencia interna de los datos y la solidez del marco teórico respaldan la validez de las conclusiones en contextos similares.

La investigación revela tensiones no resueltas que requieren atención continuada: el equilibrio entre eficiencia y comprensión profunda, la gestión de la dependencia tecnológica, y la necesidad de preservar habilidades fundamentales en entornos crecientemente automatizados.

6.2. Líneas futuras de trabajo

A partir de los hallazgos obtenidos, se identifican las siguientes líneas prioritarias para consolidar y ampliar este modelo de enseñanza:

- **Evaluación longitudinal y transferencia:** Implementar diseños pre-post con tareas equivalentes con y sin IA para medir el progreso individual y la transferencia de aprendizaje. Asimismo, evaluar la permanencia de las competencias adquiridas en contextos reales como la formación en centros de trabajo.
- **Ampliación y diversificación muestral:** Replicar la investigación en diferentes centros y contextos para aumentar la robustez de los resultados y identificar los factores que moderan la efectividad del enfoque.
- **Optimización de instrumentos:** Evolucionar los indicadores actuales hacia niveles de logro más detallados y desarrollar andamiajes específicos (plantillas de prompting, guiones de depuración) mediante estudios controlados.
- **Análisis comparativo de herramientas:** Evaluar diferentes combinaciones de asistentes de IA con criterios objetivos de utilidad pedagógica, evitando sesgos comerciales y priorizando el equilibrio entre apoyo tecnológico y control del proceso de aprendizaje.

Estas líneas de investigación permitirán profundizar en la comprensión del impacto pedagógico de la IA en la formación profesional y contribuir al desarrollo de marcos metodológicos más sólidos para su integración educativa.

Bibliografía

[1] La historia de la programación informática, March 2023. URL `https://www.superprof.es/blog/historia-desarrollo-informatico/`. Section: Programación.

[2] Jean E Sammet. *Programming languages: History and fundamentals*. Prentice-Hall, Inc., 1969.

[3] Luis Llamas. Breve historia de la programación. URL `https://www.luisllamas.es/breve-historia-de-la-programacion/`.

[4] John W Backus, Robert J Beeber, Sheldon Best, Richard Goldberg, Lois M Haibt, Harlan L Herrick, Robert A Nelson, David Sayre, Peter B Sheridan, Harold Stern, et al. The fortran automatic coding system. In *Papers presented at the February 26-28, 1957, western joint computer conference: Techniques for reliability*, pages 188–198, 1957.

[5] Jean E Sammet. The early history of cobol. In *History of Programming Languages*, pages 199–243. 1978.

[6] Andrea Lluch Cruz. Evolución del uso de los lenguajes de programación, July 2023. URL `https://www.eniun.com/evolucion-uso-lenguajes-de-programacion/`.

[7] From Punch Cards to Python - IEEE Spectrum, . URL `https://spectrum.ieee.org/from-punch-cards-to-python`.

[8] Anthony Robins, Janet Rountree, and Nathan Rountree. Learning and teaching programming: A review and discussion. *Computer science education*, 13(2):137–172, 2003.

[9] Chin Soon Cheah. Factors contributing to the difficulties in teaching and learning of computer programming: A literature review. *Contemporary Educational Technology*, 12(2):ep272, 2020.

[10] Thomas H Park and Susan Wiedenbeck. Learning web development: Challenges at an earlier stage of computing education. In *Proceedings of the seventh international workshop on Computing education research*, pages 125–132, 2011.

[11] SRM Derus and AZ Mohamad Ali. Difficulties in learning programming: Views of students. In *1st International Conference on Current Issues in Education (ICCIE 2012)*, pages 74–79, 2012.

[12] Silvia Alonso. Entendiendo el lenguaje HTML, December 2023. URL `https://dinahosting.com/blog/entendiendo-la-base-del-desarrollo-web-html-css-y-javascript/`.

[13] David I Samudio and Thomas D LaToza. Barriers in front-end web development. In *2022 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*, pages 1–11. IEEE, 2022.

[14] Manuela Petrescu, Adrian Sterca, and Ioan Badarinza. Student's interests related to web and mobile technologies study. *arXiv preprint arXiv:2311.15293*, 2023.

[15] Wilfred J Hansen. User engineering principles for interactive systems. In *Proceedings of the November 16-18, 1971, fall joint computer conference*, pages 523–532, 1972.

[16] Romain Robbes and Michele Lanza. How program history can improve code completion. In *2008 23rd IEEE/ACM International Conference on Automated Software Engineering*, pages 317–326. IEEE, 2008.

[17] Lidia Contreras-Ochando, Cèsar Ferri, and José Hernández-Orallo. Automat [r] ix: learning simple matrix pipelines. *Machine Learning*, 110(4):779–799, 2021.

[18] Lidia Contreras Ochando. *Towards Data Wrangling Automation through Dynamically-Selected Background Knowledge*. PhD thesis, Universitat Politècnica de València, 2021.

[19] Sumit Gulwani, José Hernández-Orallo, Emanuel Kitzelmann, Stephen H Muggleton, Ute Schmid, and Benjamin Zorn. Inductive programming meets the real world. *Communications of the ACM*, 58(11):90–99, 2015.

[20] Henry Lieberman. *Your wish is my command: Programming by example*. Elsevier, 2001.

[21] Rasha Ahmad Husein, Hala Aburajouh, and Cagatay Catal. Large language models for code completion: A systematic literature review. *Computer Standards & Interfaces*, page 103917, 2024.

[22] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[23] Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

[24] Qodo Team. AI Code Generation: Revolutionizing Development and Tools, May 2024. URL `https://www.qodo.ai/blog/ai-code-generation-revolutionizing-development-and-tools/`.

[25] These are the best large language models for coding, February 2025. URL `https://dev.to/hackmamba/these-are-the-best-large-language-models-for-coding-1co2`.

[26] Measuring GitHub Copilot's Impact on Productivity – Communications of the ACM, February 2024. URL `https://cacm.acm.org/research/measuring-github-copilots-impact-on-productivity/`.

[27] Yujia Li, David Choi, Junyoung Chung, Nate Kushman, Julian Schrittwieser, Rémi Leblond, Tom Eccles, James Keeling, Felix Gimeno, Agustin Dal Lago, et al. Competition-level code generation with alphacode. *Science*, 378(6624):1092–1097, 2022.

[28] Sida Peng, Eirini Kalliamvakou, Peter Cihon, and Mert Demirer. The impact of ai on developer productivity: Evidence from github copilot. *arXiv preprint arXiv:2302.06590*, 2023.

[29] Majeed Kazemitabaar, Justin Chow, Carl Ka To Ma, Barbara J Ericson, David Weintrop, and Tovi Grossman. Studying the effect of ai code generators on supporting novice learners in introductory programming. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*, pages 1–23, 2023.

[30] Emery Berger. CoPing with CoPilot, August 2022. URL `https://itnext.io/coping-with-copilot-b2b59671e516`.

[31] Saiful Islam Salim, Rubin Yuchan Yang, Alexander Cooper, Suryashree Ray, Saumya Debray, and Sazzadur Rahaman. Impeding llm-assisted cheating in introductory programming assignments via adversarial perturbation. *arXiv preprint arXiv:2410.09318*, 2024.

[32] Florencia Bailin. Programación en la era de la IA: adaptarse o quedarse atrás, April 2025. URL `https://www.skillnest.com/blog/programacion-en-la-era-de-la-ia/`.

[33] La IA ya no es moda, es rutina, . URL https://innovacion.upv.es/es/la-ia-ya-no-es-moda-es-rutina/.

[34] GDSC UPV. "La irrupción de la IA generativa" - Salvador Más DEVFEST UPV 2023, December 2023. URL https://www.youtube.com/watch?v=vHQaCDwIJPM.

[35] Alejandro Aguilar de la Rosa. La competencia digital de los estudiantes de formación profesional: una revisión sistemática. *RiiTE Revista interuniversitaria de investigación en Tecnología Educativa*, pages 200–221, 2022.

[36] Luis Canorea Ruiz. TIC, competencia digital y educación. Una propuesta de intervención aplicada a la Formación Profesional en España. January 2023. URL https://openaccess.uoc.edu/handle/10609/147935. Accepted: 2023-06-15T12:51:36Z Publisher: Universitat Oberta de Catalunya (UOC).

[37] miguel. La nueva FP: retos para el profesorado, April 2024. URL https://www.caixabankdualiza.es/la-nueva-fp-retos-para-el-profesorado/.

[38] Yongjun Xu, Xin Liu, Xin Cao, Changping Huang, Enke Liu, Sen Qian, Xingchen Liu, Yanjun Wu, Fengliang Dong, Cheng-Wei Qiu, et al. Artificial intelligence: A powerful paradigm for scientific research. *The Innovation*, 2(4), 2021.

[39] Pei Wang. On defining artificial intelligence. *Journal of Artificial General Intelligence*, 10(2):1–37, 2019.

[40] Alan M. Turing. Computing machinery and intelligence. *Mind*, 59(236):433, 1950.

[41] AI Watch. Historical evolution of artificial intelligence. *Analysis of three main paradigm shifts in AI*, 2020.

[42] Arthur L Samuel. Some studies in machine learning using the game of checkers. *IBM Journal of research and development*, 3(3):210–229, 1959.

[43] KVR Pranav and KJ Sarma. Origin, development and uses of machine learning. *International Journal For Multidisciplinary Research*, 5(1):1–18, 2023.

[44] Frank Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386, 1958.

[45] Luciano Floridi. Ai and its new winter: From myths to realities. *Philosophy & Technology*, 33:1–3, 2020.

[46] Geoffrey E Hinton, Simon Osindero, and Yee-Whye Teh. A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554, 2006.

[47] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[48] Ziming Luo, Zonglin Yang, Zexin Xu, Wei Yang, and Xinya Du. Llm4sr: A survey on large language models for scientific research. *arXiv preprint arXiv:2501.04306*, 2025.

[49] Aniruddha Shrikhande. Top AI Research Papers of 2024, December 2024. URL `https://adasci.org/top-ai-research-papers-of-2024/`.

[50] Ian J Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014.

[51] Diederik P Kingma, Max Welling, et al. Auto-encoding variational bayes, 2013.

[52] Generative AI in Education: Past, Present, and Future, . URL `https://er.educause.edu/articles/sponsored/2023/9/generative-ai-in-education-past-present-and-future`.

[53] Antonio Mastropaolo, Luca Pascarella, Emanuela Guglielmi, Matteo Ciniselli, Simone Scalabrino, Rocco Oliveto, and Gabriele Bavota. On the robustness of code generation techniques: An empirical study on github copilot. In *2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE)*, pages 2149–2160. IEEE, 2023.

[54] Michel Wermelinger. Using github copilot to solve simple programming problems. In *Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 1*, pages 172–178, 2023.

[55] La IA ha incrementado la productividad en programación: Informe de GitHub | Diario TI, . URL `https://diarioti.com/la-ia-ha-incrementado-la-productividad-en-programacion-informe-de-github/123056`.

[56] ecanorea. Microsoft Copilot vs GitHub Copilot: Differences and Benefits, October 2024. URL `https://www.plainconcepts.com/microsoft-copilot-github-copilot-differences-benefits/`.

[57] Dan Sun, Azzeddine Boudouaia, Junfeng Yang, and Jie Xu. Investigating students' programming behaviors, interaction qualities and perceptions through prompt-based learning in chatgpt. *Humanities and Social Sciences Communications*, 11(1):1–14, 2024.

[58] Amazon CodeWhisperer vs. Copilot: Which Is Right for You?, . URL `https://www.missioncloud.com/blog/github-copilot-vs-amazon-codewhisperer`.

[59] A. I. Perceiver. Complete Guide On How To Use Blackbox AI, April 2024. URL https://medium.com/@aiperceiver/howcomplete-guide-on-how-to-use-blackbox-ai-f6c454f44510.

[60] BLACKBOXAI #1 Coding Agent +10 Million Developers - Visual Studio Marketplace, . URL https://marketplace.visualstudio.com/items?itemName=Blackboxapp.blackbox.

[61] Xiao Bi, Deli Chen, Guanting Chen, Shanhuang Chen, Damai Dai, Chengqi Deng, Honghui Ding, Kai Dong, Qiushi Du, Zhe Fu, et al. Deepseek llm: Scaling open-source language models with longtermism. *arXiv preprint arXiv:2401.02954*, 2024.

[62] Qihao Zhu, Daya Guo, Zhihong Shao, Dejian Yang, Peiyi Wang, Runxin Xu, Y Wu, Yukun Li, Huazuo Gao, Shirong Ma, et al. Deepseek-coder-v2: Breaking the barrier of closed-source models in code intelligence. *arXiv preprint arXiv:2406.11931*, 2024.

[63] deepseek-ai/DeepSeek-Coder, April 2025. URL https://github.com/deepseek-ai/DeepSeek-Coder. original-date: 2023-10-20T06:38:01Z.

[64] Get coding help from Gemini Code Assist — now for free, February 2025. URL https://blog.google/technology/developers/gemini-code-assist-free/.

[65] Agnia Sergeyuk, Ilya Zakharov, Ekaterina Koshchenko, and Maliheh Izadi. Human-ai experience in integrated development environments: A systematic literature review. *arXiv preprint arXiv:2503.06195*, 2025.

[66] Beiqi Zhang, Peng Liang, Xiyu Zhou, Aakash Ahmad, and Muhammad Waseem. Practices and challenges of using github copilot: An empirical study. *arXiv preprint arXiv:2303.08733*, 2023.

[67] Ya Gao Research, GitHub Customer. Research: Quantifying GitHub Copilot's impact in the enterprise with Accenture, May 2024. URL https://github.blog/news-insights/research/research-quantifying-github-copilots-impact-in-the-enterprise-with-accenture/.

[68] Zheyuan Kevin Cui, Mert Demirer, Sonia Jaffe, Leon Musolff, Sida Peng, and Tobias Salz. The effects of generative ai on high skilled work: Evidence from three field experiments with software developers. *Available at SSRN 4945566*, 2024.

[69] Gal Bakal, Ali Dasdan, Yaniv Katz, Michael Kaufman, and Guy Levin. Experience with github copilot for developer productivity at zoominfo. *arXiv preprint arXiv:2501.13282*, 2025.

[70] Eirini Kalliamvakou. Research: quantifying GitHub Copilot's impact on developer productivity and happiness, September 2022. URL `https://github.blog/news-insights/research/research-quantifying-github-copilots-impact-on-developer-productivity-and-happiness/`.

[71] Alfonso Ramon García Alcívar and Eddy Alejandro Loor Navia. Herramientas de inteligencia artificial usadas en el desarrollo de software. una revisión sistemática de la literatura.: Artifice intelligence tools used in software development. a systematic revision of literature. *Revista Científica Multidisciplinar G-nerando*, 6(1):ág–4035, 2025.

[72] Jimmy Rivera, Bolivar Mauricio Mendoza Morán, et al. La influencia de las ia generativas en la enseñanza de programación en universidades. *Estudios y Perspectivas Revista Científica y Académica*, 4(4):1257–1274, 2024.

[73] Héctor Terán. La implementación de la inteligencia artificial en la enseñanza de la programación. un estudio sobre el uso ético de chatgpt en el aula. *Encuentro Internacional de Educación en Ingeniería*, 2023.

[74] Joyce Francela Useda Medrano, Ángel Alonso Ortiz García, and Franklin Chavez Baltodano. Visión estudiantil: Ia en la transformación de la enseñanza de ingeniería en ti. *Revista Tecnología en Marcha*, pages ág–37, 2025.

[75] Francisco José García-Peñalvo. Inteligencia artificial generativa y educación: Un análisis desde múltiples perspectivas. *Education in the Knowledge Society (EKS)*, 25: e31942–e31942, 2024.

[76] Cinta Gallent-Torres, Alfredo Zapata-González, and José Luis Ortego-Hernando. El impacto de la inteligencia artificial generativa en educación superior: una mirada desde la ética y la integridad académica. *RELIEVE. Revista Electrónica de Investigación y Evaluación Educativa*, 29(2):1–21, 2023.

[77] Rocío del Amor, Adrián Colomer, and Valery Naranjo. El rol de la inteligencia artificial generativa en la educación: beneficios potenciales de chatgpt para promover el aprendizaje en tareas de programación en python. In *In-Red 2023-IX Congreso Nacional de Innovación Educativa y Docencia en Red*, pages 851–861. Editorial Universitat Politècnica de València, 2023.

[78] Leonardo Garro Mena. Optimizando el aprendizaje mediante ia: la eficacia de flujos de trabajo estructurados en la educación superior. *REDU. Revista de Docencia Universitaria*, 22(2):105–121, 2024.

[79] Pablo Dúo Terrón. Generative artificial intelligence: Educational reflections from an analysis of scientific production. 2024.

[80] IA generativa: claves, aplicación y futuro en el ámbito educativo, . URL https://www.uoc.edu/es/news/2023/192-ia-generativa-claves-aplicacion-futuro-educacion.

[81] Dennisse Alarcón-Arias, Segress García-Hevia, et al. La inteligencia artificial como aliada en la formación técnica profesional: Mejorando la calidad educativa. *MQRInvestigar*, 8(4):6877–6892, 2024.

[82] Francisco José García-Peñalvo, Faraón Llorens-Largo, and Javier Vidal. La nueva realidad de la educación ante los avances de la inteligencia artificial generativa. *RIED-Revista Iberoamericana de Educación a Distancia*, 27(1):9–39, 2024.

[83] Carina S González-González et al. El impacto de la inteligencia artificial en la educación: transformación de la forma de enseñar y de aprender. 2023.

[84] Denisse Ivonne Leon Medrano, Sorayda Petita Altamirano Cortez, Mercy Germania Reyes Espinoza, and Andrea María Sánchez García. Implementación de la inteligencia artificial como herramienta pedagógica para la formación profesional de educación inicial. *Ciencia Latina Revista Científica Multidisciplinar*, 8(6):4373–4385, 2024.

[85] Graciela María Scavone. Análisis del posible uso de la inteligencia artificial generativa en procesos formativos. 2025.

[86] La IA generativa y el futuro de la educación - UNESCO Digital Library, . URL https://unesdoc.unesco.org/ark:/48223/pf0000385877_spa.

[87] Danny Daniel Carhuaz Valdez and Elvis Joel Arcata-Maquera. Educación universitaria en la era de la inteligencia artificial generativa:¿ integridad académica en riesgo? *INGENIERÍA INVESTIGA*, 6, 2024.

[88] Jonathan Fabricio Medina Tene, Gloria Noemi Jumbo Salinas, and Juan José Astudillo Zurita. Inteligencia artificial y formación profesional de los estudiantes de derecho: Una visión desde sus actores. *Ciencia Latina Revista Científica Multidisciplinar*, 8(5): 11116–11135, 2024.

[89] Nicholas Gardella, Raymond Pettit, and Sara L. Riggs. Performance, workload, emotion, and self-efficacy of novice programmers using ai code generation. In *Proceedings of the 2024 on Innovation and Technology in Computer Science Education V. 1*, ITiCSE 2024, page 290–296, New York, NY, USA, 2024. Association for Computing Machinery. ISBN 9798400706004. doi: 10.1145/3649217.3653615. URL https://doi.org/10.1145/3649217.3653615.

[90] Doga Cambaz and Xiaoling Zhang. Use of ai-driven code generation models in teaching and learning programming: a systematic literature review. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 1*, pages 172–178, 2024.

[91] Sebastian Eilermann, Leon Wehmeier, Oliver Niggemann, and Andreas Deuter. Kiaaa: An ai assistant for teaching programming in the field of automation. In *2023 IEEE 21st International Conference on Industrial Informatics (INDIN)*, pages 1–7. IEEE, 2023.

[92] Zishan Ahmed, Shakib Sadat Shanto, and Akinul Islam Jony. Potentiality of generative ai tools in higher education: Evaluating chatgpt's viability as a teaching assistant for introductory programming courses. *STEM Education*, 4(3):165–182, 2024.

[93] Akiyoshi Wakatani, Toshiyuki Maeda, et al. Prototype advisory system for learning c programming using generative ai. *Emanate-Educational Sessions Highlights*, 2024.

[94] Mario Sandoval Hernández, Griselda J Morales Alarcón, Héctor Vázquez Leal, Jesús Huerta Chua, Uriel A Filobello Niño, et al. El uso del prompt de chatgpt como asistente en la educación. *RIDE. Revista Iberoamericana para la Investigación y el Desarrollo Educativo*, 14(28), 2024.

[95] Nestor Antonio Gallegos-Ramos. Chatgpt en el aprendizaje de lenguajes de programación en estudiantes universitarios. *Revista Amazonía Digital*, 2(2):e250–e250, 2023.

[96] Joe Llerena-Izquierdo, Johan Mendez-Reyes, Raquel Ayala-Carabajo, and Cesar Andrade-Martinez. Innovations in introductory programming education: The role of ai with google colab and gemini. *Education Sciences*, 14(12), December 2024. ISSN 2227-7102. doi: 10.3390/educsci14121330. Publisher Copyright: © 2024 by the authors.

[97] Leo Porter and Daniel Zingaro. *Learn AI-Assisted Python Programming: With Github Copilot and ChatGPT*. Simon and Schuster, 2024.

[98] Rina Zviel-Girshin. The good and bad of ai tools in novice programming education. *Education Sciences*, 14(10):1089, 2024.

[99] Farman Ali Pirzado, Awais Ahmed, Román A Mendoza-Urdiales, and Hugo Terashima-Marin. Navigating the pitfalls: Analyzing the behavior of llms as a coding assistant for computer science students-a systematic review of the literature. *IEEE Access*, 2024.

[100] Gökhan Akçapınar and Elif Sidan. Ai chatbots in programming education: guiding success or encouraging plagiarism. *Discover Artificial Intelligence*, 4(1):87, 2024.

[101] Savithree Senanayake, Kasun Karunanayaka, and KVJP Ekanayake. Review on ai assistant systems for programming language learning in learning environments. In *2024 8th SLAAI International Conference on Artificial Intelligence (SLAAI-ICAI)*, pages 1–6. IEEE, 2024.

Apéndice A

Cuestionario Post-intervención

1. **Grupo**
   - 1º ASIR
   - 2º ASIR
   - 2º SMR
2. **Herramientas de IA usadas (puedes marcar más de una)**
   - ChatGPT
   - GitHub Copilot
   - Blackbox
   - Gemini
   - Claude
   - Otras .....
3. **Antes de usar IA, ¿cómo resolvías dudas de programación? (marca hasta 2)**
   - Buscando en internet
   - Documentación / libros
   - Preguntando al profesor/a
   - Preguntando a compañeros
   - Otras .....
4. **¿Con qué frecuencia usas IA para programar?**
   - Siempre
   - Frecuentemente

- Ocasionalmente
- Nunca

5. **¿Crees que has mejorado tu manera de escribir *prompts* estructurados desde que empezó el curso?**
   - Sí, ahora sé cómo estructurarlo
   - No, los escribo igual
   - No estoy seguro/a
6. **¿Qué dificultad crees que tiene escribir un buen *prompt*?**
   - Muy fácil
   - Fácil
   - Neutral
   - Difícil
   - Muy difícil
7. **¿La IA te ha ayudado a entender cómo programar?**
   - Sí, mucho
   - Sí, algo
   - No, no ha influido
   - No, ha dificultado
8. **¿Cómo de seguro/a te sientes escribiendo código sin IA?**
   - Nada seguro/a
   - Algo inseguro/a
   - Bastante seguro/a
   - Muy seguro/a
9. **¿Con qué frecuencia tienes que corregir el código que te genera la IA?**
   - Siempre
   - Frecuentemente
   - Ocasionalmente
   - Casi nunca
10. **Si la IA genera código incorrecto, ¿cómo lo corriges? (marca hasta 2)**

- Lo modifico yo mismo
- Pido a la IA que lo ajuste
- Consulto al profesor/a
- Pido ayuda a compañeros
- Otras .....

11. **Comparado con métodos tradicionales (aprender sin IA, memorizar código, documentación...), ¿prefieres el uso de IA?**

    - Sí, prefiero IA
    - No, prefiero tradicional
    - No estoy seguro/a

12. **¿La combinación de IA + métodos tradicionales es efectiva?**

    - Muy efectiva
    - Efectiva
    - Poco efectiva
    - Nada efectiva

13. **Ordena los siguientes elementos según el porcentaje de tiempo que usas (1 = más tiempo, 3 = menos tiempo).**

    a) Escribir código manualmente (sin IA)

    b) Usar IA para generar código

    c) Modificar código que ha generado la IA

14. **¿Recomendarías integrar IA en la enseñanza?**

    - Sí
    - No
    - Otras .....

15. **¿Cómo te ayuda la IA a generar páginas web?**
    Respuesta abierta.

16. **¿Qué habilidades crees que has mejorado gracias al uso de IA en programación?**
    Respuesta abierta.

17. **¿Qué desafíos has enfrentado al integrar IA en tu proceso de aprendizaje?**
    Respuesta abierta.

18. **¿Crees que haber aprendido a usar la IA te puede ayudar en tu futuro profesional?**

    - Sí
    - No
    - No estoy seguro/a

19. **¿Cómo crees que la IA cambiará la forma de programar y de enseñar a programar?**
    Respuesta abierta.

20. **Sugerencias para mejorar el uso de IA en clase.**
    Respuesta abierta.

Apéndice B

Respuestas a las preguntas abiertas del cuestionario

Pregunta 15: ¿Cómo te ayuda la IA a crear páginas web?

[1]

ID	Respuesta
1	En estructurar mis ideas, haciendo que sea más sencillo centrarme en lo que se pide y ultimar detalles.
2	Me ayuda especialmente a la parte más difícil: empezar. Pedirle que estructure los apartados agiliza dar forma a la web.
3	Me ayuda a crear una estructura para luego personalizarla.
4	La IA lo hace más rápido que cualquier persona; es eficiente.
5	Agregar estilos modernos de forma rápida e integrar PHP a la página.
6	Corrige mi código, señala fallos y explica cómo arreglarlos; así aprendo de los errores y mis proyectos mejoran.
7	Me permite comprender y generar de forma personalizada según mis necesidades.
8	Más o menos te da lo que pides.
9	Genera mejores ideas y mejora las mías.
10	Ahorra tiempo; no necesito tener cientos de fragmentos en la cabeza.
11	Aprendo nuevas funciones y otras formas de crear cosas con otro código.
12	Aporta ideas y reduce el tiempo de escritura de código.
13	Me suele hacer el CSS y el JavaScript.
14	Me da la estructura y yo la modifico hasta que queda como quiero.
15	Genera HTML, CSS y JS; corrige errores, da ideas de diseño y optimiza el código.
16	Genera código automáticamente, optimiza, corrige errores y sugiere mejoras de diseño y accesibilidad.
17	Me da la vida.

[1]Nota: los nombres se han omitido por privacidad; los ID son solo un identificador numérico.

18	Aporta ideas cuando me quedo en blanco y ahorra tiempo; usa lenguaje más avanzado y produce mejores páginas.
19	Ayuda en tareas más complejas y corrige errores propios o de la IA.
20	Resuelve elementos que no sé abordar.
21	Con muchas discusiones.
22	Facilita generar la estructura y sólo hay que corregir pequeños errores.
23	Permite crear diseños y esqueletos rápidos, aumentando la eficiencia.
24	Le doy un prompt de lo que quiero y lo hace, aunque no siempre perfecto.
25	Pido una estructura general (menú, banner, contenido, footer...). Luego personalizo estilos y añado funciones; ahorro código repetitivo.
26	Tiene más creatividad que yo.
27	Me ayuda en casi todo.
28	Crea el header, footer y la paleta; ayuda con la compatibilidad móvil.
29	Puedo pedir la página entera o trozos; la IA secciona y explica.
30	Es muy útil para casi todo; a veces falla, pero ayuda.
31	Me ayuda a crear páginas más profesionales y llamativas.
32	Genera una base rápidamente.
33	Proporciona una base que luego modifico, evitando empezar de cero.
34	Me ahorra el trabajo en algo que no me gusta: programar páginas web.

Pregunta 16: ¿Qué habilidades crees que has mejorado con la IA?

ID	Respuesta
1	Comprensión lectora y eficiencia.
2	He mejorado mucho en CSS: modificar el código generado me ha enseñado nuevas etiquetas y ajustes de estilo.
3	Rapidez al escribir código y más creatividad al ofrecer posibilidades que superan mi base.
4	Detección de errores: al entender el código noto enseguida si la IA se equivoca.
5	Mayor eficiencia: tareas que antes requerían más tiempo ahora se resuelven en minutos.
6	Ahora comprendo para qué sirve cada sección del código de una página web.
7	Nuevos conocimientos y formas de entender cómo funcionan ciertas cosas; me motiva a crear.
8	Cometo menos errores.
9	Mayor soltura para entender partes del código.
10	Más velocidad de reacción y capacidad para corregir algunos códigos.
11	Mejor creación de *prompts* y comprensión del código.

12	He mejorado en corregir código.
13	HTML.
14	Escribo código más rápido y con menos errores; encuentro fallos más deprisa.
15	Eficiencia, depuración, optimización y aprendizaje de nuevas tecnologías.
16	Muchos avances.
17	No sé.
18	Sé explicar mejor lo que necesito a la IA y diferenciar resultados buenos de erróneos.
19	Uso de *prompts* más avanzados.
20	Sin saber programar del todo puedo ver qué está mal y qué no.
21	Corrección y entendimiento del código pidiendo a la IA que lo corrija.
22	Optimización de tiempo y recursos al generar código.
23	Rapidez para finalizar la tarea.
24	Comprensión y resolución de problemas gracias a explicaciones adaptadas.
25	Entender y visualizar mejor las cosas para poder hacerlo a mano si fuera necesario.
26	Principalmente productividad.
27	Velocidad para acabar el proyecto.
28	Mejora en los *prompts*.
29	Desarrollo y mejora de habilidades en HTML y CSS.
30	Uso de JavaScript.
31	Optimización de tareas.
32	Entender mejor el código y depurarlo.

Pregunta 17: ¿Qué desafíos te ha supuesto usar la IA?

ID	Respuesta
1	Tener que pedir muchas veces que arregle lo que se le pide o darle mucho contexto para obtener la respuesta buscada.
2	La IA en sí no representa desafíos; el problema es la mala praxis.
3	Intentar comprender código generado sin tener aún mucha base y usarla para que me lo enseñe.
4	Dependencia si no tienes conocimientos básicos; corregir errores cuesta más sin consultarla.
5	A veces la IA devuelve información errónea o lejos de lo buscado; hay que ser muy claro al pedir.
6	Cada día surgen cosas nuevas y eso me obliga a mejorar mis conocimientos constantemente.
7	Aprender con mayor facilidad.
8	Casi ninguno; lo difícil te lo explica.
9	A veces nos volvemos un poco más perezosos.
10	Conseguir que la IA me entienda.

11	Sólo corregir algunos errores que comete.
12	Los múltiples errores que genera.
13	Tareas escolares.
14	Debo revisar todo porque a veces parece correcto pero tiene fallos; confío demasiado y no pienso por mí misma.
15	Dependencia excesiva, validar la información, comprender el código generado y adaptarse a herramientas en evolución.
16	Saber decirle cómo quiero las cosas.
17	Ninguno.
18	Diferenciar los errores cuando aún debo aprender el tema.
19	A veces no hace lo que le pides.
20	Madrugadas de discusiones.
21	Mejorar la calidad de los *prompts*.
22	Entender cómo estructurar *prompts* para obtener respuestas correctas.
23	Cuando ChatGPT llega al límite diario y hay que buscar alternativas.
24	Respuestas con conocimientos más avanzados que los míos; necesito ir despacio y entender.
25	Ninguno, salvo corregir cuando se equivoca.
26	Entender qué hace lo que me entrega.
27	Cuando se acababa el ChatGPT-4 gratuito.
28	Ninguno.
29	A veces no ofrece una respuesta sólida y tengo que cambiar cosas.
30	Ninguno.
31	Costó implementarla, pero una vez aprendida facilita tareas y estudio.
32	Ninguno.

Pregunta 19: ¿Cómo cambiará la IA la forma de aprender/programar?

ID	Respuesta
1	Nos centraremos en entender la sintaxis básica y la lógica; lo demás lo hará la IA.
2	El lenguaje natural será el nuevo lenguaje de programación; dominar los *prompts* será más importante que programar directamente.
3	La IA se convertirá en una herramienta común, como VS Code, y habrá especialización tanto de la IA como del usuario.
4	Mucha gente la usará sin conocimientos previos, pero la curiosidad permitirá aprender cómo funciona el código.
5	La IA ya cambia la forma de programar; debe integrarse como complemento, pero es imprescindible una base para validar los resultados.
6	Bien usada, ayudará a aprender más rápido y a conseguir la información necesaria.

7	Reduce el tiempo de programación y amplía la visión creativa, generando nuevos conocimientos en el usuario.
8	La gente dejará de programar manualmente y lo hará directamente con IA.
9	Cambiará bastante porque enseña bien a hacer las cosas.
10	Hará la programación más fácil y la explicará mejor.
11	Los alumnos se volverán muy dependientes de la IA.
12	Facilitará el aprendizaje con explicaciones claras y personalizadas, pero puede generar copia sin reflexión.
13	Hará la programación más accesible, automatizando tareas repetitivas y fomentando la creatividad.
14	Mejorará, obviamente.
15	El futuro pertenece a quienes sepan utilizar la IA; ahorra tiempo y aporta ideas de mejora.
16	Dará facilidad y permitirá ser más concreto.
17	No lo sé con exactitud.
18	En vez de 6 h programando, con 2 h de conversación con la IA se terminan las tareas.
19	Se enfocará más en corregir a la IA y entender fundamentos que en programar desde cero.
20	Hará la programación mucho más accesible al público general.
21	La hará mucho más amena.
22	Como la maquinaria en construcción: ahorra tiempo y permite avanzar más rápido.
23	La IA hará casi todo.
24	Herramienta muy potente que ya optimiza procesos en empresas (ej. Mojang).
25	Dentro de 10 años todo será *prompts*; quizá en 20 serán pensamientos directos al PC.
26	Positivo para el desarrollo, aunque podría eliminar trabajos si se lleva muy lejos.
27	Ayudará mucho en programación y enseñanza en general.
28	No se necesitarán programadores dedicados; bastará con saber armar *prompts*.
29	Sí.
30	Acabará con los programadores de bajo y medio nivel.

Pregunta 20: ¿Qué sugerencias tienes para mejorar el uso de IA en clase?

ID	Respuesta
1	Dar libertad total para proyectos con IA una vez vista la teoría, para explorar hasta dónde se puede llegar.
2	Profundizar en la ingeniería de *prompts*.
3	Ayudar con los *prompts* y sus posibilidades.
4	Enseñar cómo hacer un buen *prompt* para obtener el resultado necesario.

5	De momento ninguna sugerencia; uso la IA para cuerpo de páginas, correcciones y comandos.
6	Enseñar bien la IA y luego plantear retos para comprobar la utilidad aprendida.
7	Usarla como una fuente más, sin llegar a aborrecerla.
8	Saber escribir *prompts* y entender lo que devuelve.
9	No usarla para todo y pedirle que explique cómo se hace.
10	No usarla siempre.
11	Verificar la información de la IA; ejercicios para corregir sus errores; intentar resolver antes de preguntar.
12	Fomentar pensamiento crítico, integrar IA en proyectos reales, usar asistentes para dudas, tratar ética y creatividad.
13	Ofrecer un listado de IA según la necesidad.
14	Plantear tareas con IA (ej. web que sume números) y luego analizar el código generado.
15	Sin sugerencias; debería plantearlas directamente a la IA.
16	Ninguna; le gusta cómo lo hace la profesora.
17	Que se pueda usar siempre.
18	Está bien así; acceso a muchas IA.
19	Combinar explicación paso a paso del profesor con profundización en IA cuando sea necesario.
20	Explicar la IA y utilizarla.
21	Mantener la secuencia: bases primero, IA después para leer y mejorar código.
22	Dedicar tiempo a explicar qué hay fuera y cómo usarlo en beneficio propio.
23	Usarla para aprender, no sólo para copiar.
24	Implementar IA también en asignaturas más teóricas para facilitar la comprensión.
25	El uso actual le parece correcto.
26	Hacer exposiciones explicando los tipos de IA que existen.
27	Usarla para tareas repetitivas y para aprender y entender conceptos.

Apéndice C

Boletín de ejercicios de ingeniería de prompts

Ejercicios de Ingeniería de Prompts

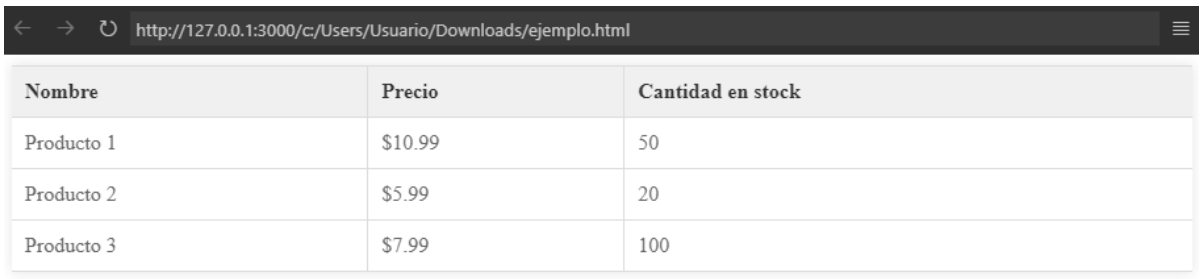
Redacta el prompt más adecuado para cada uno de los ejercicios de modo que se obtenga exactamente lo que se espera en cada uno de ellos.

Genera un documento explicando cada **prompt** y mostrando **capturas de su ejecución** y **del código generado**. Si has iterado (preguntado a la IA más de una vez) o has tenido que mejorar un prompt o reescribirlo, explícalo también en la documentación. **Todos los pasos seguidos para conseguir los resultados deben estar explicados**.

Comprueba que los prompts generan el código necesario para cumplir con cada una de las tareas y que no tienen errores. Si tienes que modificarlos porque no cumplen lo que necesitas o tienen errores, **descríbelo en la documentación**.

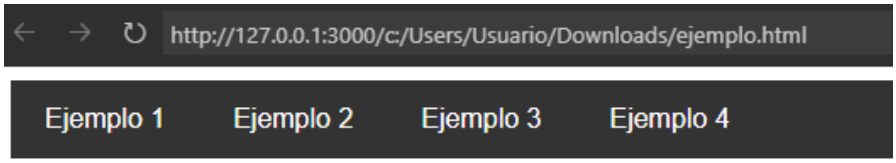
Ejercicio 1

Escribe uno o varios prompts para generar la siguiente web (HTML + CSS):

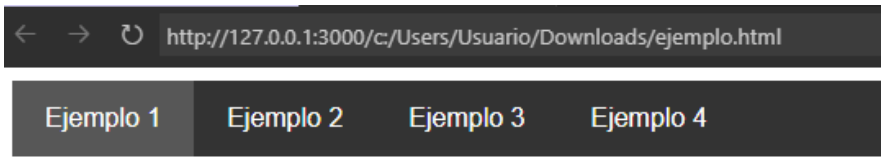


Ejercicio 2

Escribe uno o varios prompts para generar una web (HTML + CSS) con un menú como el siguiente:



Cada elemento del menú es un enlace que se abre en una ventana nueva y al pasar el ratón por encima deben cambiar de color:



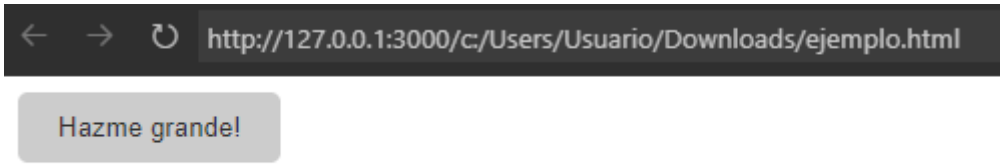
Ejercicio 3

Escribe uno o varios prompts de modo que consigas una web con un formulario como el que se muestra a continuación (HTML + CSS):

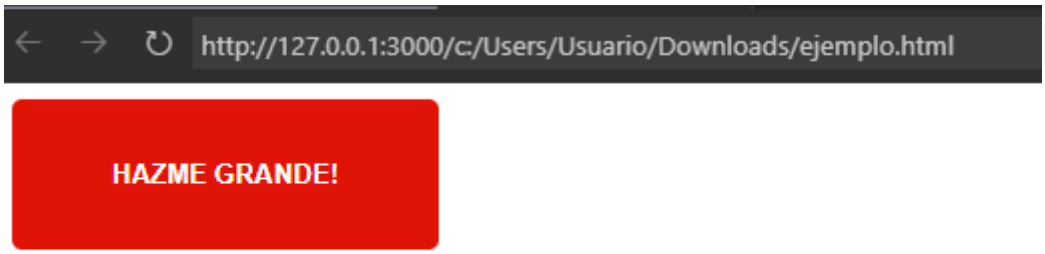
Los campos de los formularios se llaman input. Tienes información sobre formularios en: https://www.w3schools.com/html/html_forms.asp

Ejercicio 4

Escribe un prompt que genere una web (HTML) con un botón como el siguiente:



Ahora escribe un prompt para que genere un CSS para esta web de modo que, al pasar por encima del botón pase esto:



La transformación del botón debe tardar 2segundos.

La propiedad CSS que necesitas es Transitions:
https://www.w3schools.com/css/css3_transitions.asp

Ejercicio 5

Escribe los prompts necesarios para generar una web con una imagen en la que, infinitamente, tiene que pasar esto:

Las propiedades que necesitas son animation (rotate) y keyframes:

https://www.w3schools.com/css/css3_animations.asp

Apéndice D

Boletín de ejercicios de depuración de código

Ejercicios de generación y depuración de código con Blackbox

Instalación de la extensión:

1. Abre Visual Studio Code.
2. Ve al menú de Extensiones (Ctrl + Shift + X) y busca Blackbox AI.
3. Instala la extensión y asegúrate de que el ícono de Blackbox aparezca en la barra inferior.

Ejercicio 1: Uso de Blackbox desde el chat mediante prompts

1. Generar código HTML a través del chat:
    - Abre un archivo en Visual Studio Code.
    - Accede a la opción "Code Chat" desde la barra lateral de Blackbox.
    - Usando el chat pide a Black box que genere el código HTML de una web con una cabecera, una sección principal con un texto, y un pie de página.
    - Copia el código generado en el archivo HTML, ejecútalo y visualiza el resultado.
2. Modificar el código para añadir CSS:
    - Pídele a Blackbox que te del el CSS de la web de modo que la cabecera tenga un fondo azul y el texto de la sección principal sea de color gris oscuro con un tamaño de letra de 18px.
    - Observa el código CSS generado y ajusta el diseño según las indicaciones.

Entrega: Adjunta una captura de pantalla del proceso y otra mostrando el resultado. ¿Te parece útil usar una extensión de Visual Studio como Blackbox o prefieres las herramientas externas en navegador, como ChatGPT?

Ejercicio 2: Cambiar de modelo en Blackbox

1. En el chat de Blackbox, selecciona la opción para cambiar el modelo.
2. Utiliza los diferentes modelos que haya disponibles y ejecuta el siguiente prompt:
    - *"Genera un diseño en HTML + CSS con un menú de navegación horizontal que tenga tres enlaces: Inicio, Servicios y Contacto."*
    - Compara los resultados generados por cada modelo. ¿Notas alguna diferencia en los estilos generados?

Entrega: Escribe una breve comparación de los estilos generados por los diferentes modelos y adjunta una captura de cada resultado.

Ejercicio 3: Crear y usar un agente en Blackbox

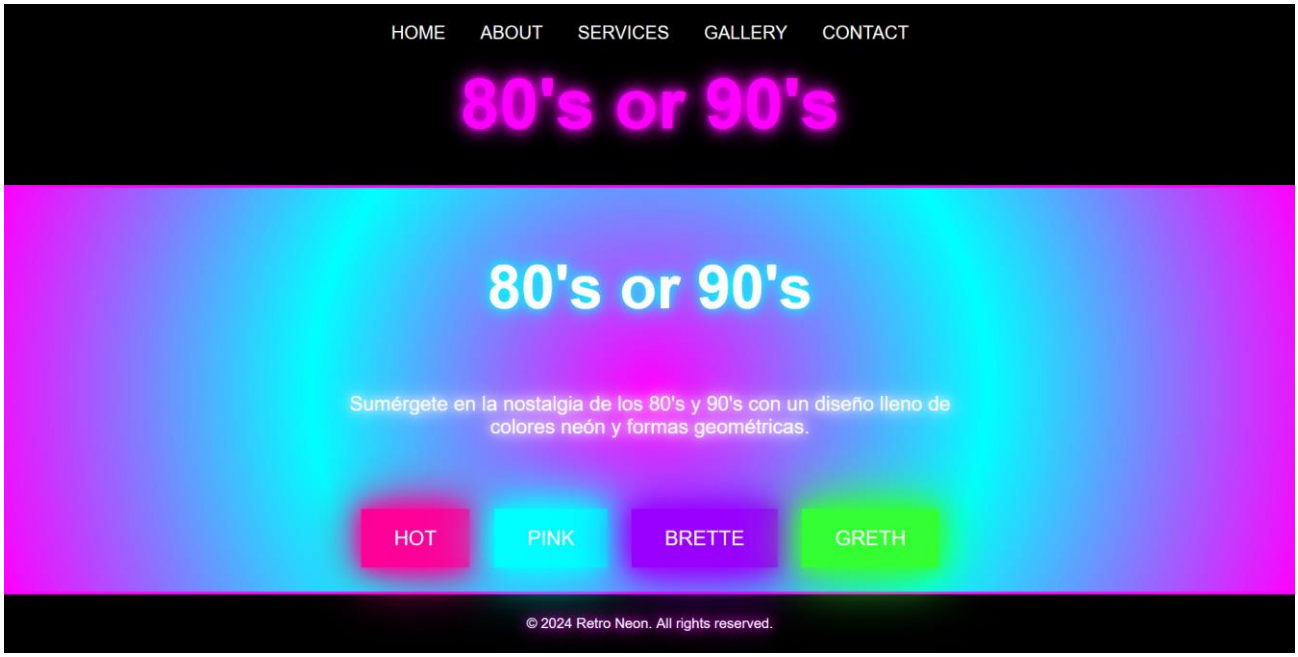
1. Creación de un agente personalizado:
    - Crea un agente personalizado dentro de Blackbox orientado a la generación de código web responsive.
        - Las páginas web responsive se adaptan al tamaño de la pantalla (PC, smartphone, Tablet, etc.).
    - Define un nombre para el agente y escribe una descripción clara de lo que hará, por ejemplo:
        - *"Este agente generará estilos CSS optimizados y sugerirá buenas prácticas de diseño responsive. Además, comentará todo el código para que sea más comprensible."*
    - Una vez creado, selecciona el agente y utiliza un prompt como el siguiente:
        - *"Crea un diseño CSS responsive para una tarjeta de presentación con imagen, título y descripción."*

Entrega: Adjunta capturas de la creación del agente, así como del código generado usando su chat. Escribe una breve reflexión sobre cómo un agente nos puede ayudar en la generación de código.

Ejercicio 4: Generación de código a partir de una imagen

- Haz una captura de la imagen de la siguiente página o usa la imagen adjunta en la tarea.
- Sube la imagen a Blackbox usando la opción “Image to App”.
- Observa cómo Blackbox genera el código HTML/CSS a partir de la imagen.

Tarea: Verifica si el código generado coincide con lo que se muestra en la imagen y realiza ajustes si es necesario.

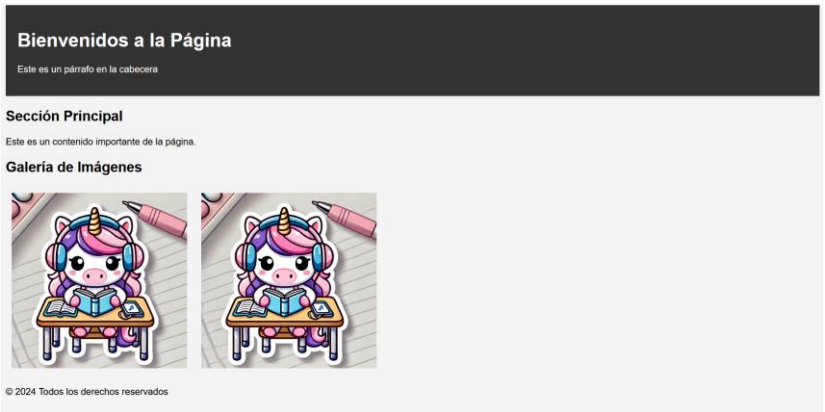


Entrega: Adjunta una captura de pantalla del resultado inicial y otra del resultado final, si has tenido que hacer cambios. Explica los cambios que hayas tenido que realizar para conseguir el resultado final.

Ejercicio 5: Depuración de código con Blackbox

1. Abre el archivo zip adjunto en la tarea con Visual Studio. Este archivo contiene un archivo HTML y otro CSS con errores, tanto en el código HTML como en el CSS.
2. Utiliza Blackbox para revisar y corregir los errores.

Tarea: Realiza las correcciones sugeridas por Blackbox y comprueba que el código funcione correctamente en el navegador. La siguiente captura muestra cómo debe quedar la página.



Entrega: Describe los errores corregidos y adjunta capturas de los prompts y las respuestas. Sube los archivos HTML/CSS depurados comprimidos en un zip.