



Centro Universitario de la Defensa en la Escuela Naval Militar

TRABAJO FIN DE GRADO

*Evaluación de la IA Generativa como apoyo en el aprendizaje y
en la docencia de programación*

Grado en Ingeniería Mecánica

ALUMNO: Iago Novo Andrade

DIRECTORES: Miguel Rodelgo Lacruz

Belén Barragáns Martínez

CURSO ACADÉMICO: 2024-2025

Universida_{de}Vigo



Centro Universitario de la Defensa en la Escuela Naval Militar

TRABAJO FIN DE GRADO

*Evaluación de la IA Generativa como apoyo en el aprendizaje y
en la docencia de programación*

Grado en Ingeniería Mecánica
Intensificación en Tecnología Naval
Infantería de Marina

UniversidadeVigo

RESUMEN

Este Trabajo Fin de Grado analiza el papel de las Inteligencias Artificiales Generativas en el ámbito educativo, con especial atención a su aplicación en la asignatura Informática para la Ingeniería del Grado en Ingeniería Mecánica impartido por el Centro Universitario de la Defensa en la Escuela Naval Militar. El objetivo principal es evaluar si estas herramientas pueden actuar como un apoyo efectivo tanto para el alumnado como para el profesorado en tareas relacionadas con la comprensión de conceptos, la generación de código y los procesos de evaluación automatizada.

Para ello, se realiza una comparativa entre distintos modelos de IAG, tanto en versiones gratuitas como de pago, aplicados a la resolución de ejercicios de diferentes niveles de dificultad: desde diagramas de flujo hasta proyectos de programación más completos. Asimismo, se examina su capacidad para simular procesos de corrección, comparando las calificaciones de dichos trabajos con las valoraciones generadas por los modelos con los criterios utilizados por el profesorado, prestando especial atención a la calidad y adecuación del código propuesto.

Como parte del enfoque experimental, se desarrolla un modelo GPT personalizado, entrenado específicamente con los contenidos de la asignatura, con el propósito de valorar su potencial como recurso académico en contextos de enseñanza universitaria de programación.

PALABRAS CLAVE

IAG, programación, ChatGPT, Gemini, Copilot, GPT personalizado

AGRADECIMIENTOS

Quiero comenzar agradeciendo de forma muy especial a mis tutores, Belén y Miguel, por su implicación constante en el desarrollo de este Trabajo Fin de Grado. Gracias por todas las horas dedicadas a revisar el proyecto con detalle, por las observaciones constructivas, y especialmente por la enorme facilidad de adaptación que han demostrado a lo largo de todo el proceso. Su acompañamiento ha sido fundamental para poder avanzar con confianza y claridad en cada etapa del trabajo.

Además, valoro profundamente la cercanía y la paciencia que han mostrado en todo momento, incluso en los momentos de mayor incertidumbre o presión. Han sabido guiarme con rigor y amabilidad, ofreciéndome siempre nuevas perspectivas y animándome a ir un paso más allá. Sin su apoyo, este proyecto no habría sido posible tal y como lo he concebido.

También quiero dar las gracias a mi familia: a mamá, papá y Dési, por su apoyo incondicional, por estar siempre ahí, animándome en los momentos de duda, y por creer en mí incluso cuando yo no lo hacía. Sin vuestra fuerza y vuestro cariño, sacar adelante este proyecto no habría sido posible.

Gracias por acompañarme en el camino, por las conversaciones reconfortantes, por celebrar cada pequeño avance conmigo y por recordarme que siempre se puede dar un poco más. Este logro también es vuestro, porque detrás de cada página escrita hay una parte de vuestro amor, vuestra paciencia y vuestra confianza en mí.

Y, por último, quiero agradecer profundamente a Luci, mi pareja, por haberme acompañado tan de cerca durante todo este camino. Gracias por todas las horas que has pasado a mi lado, por adaptarte a mis horarios, muchas veces complicados, y por no soltarme ni un solo momento. Tu apoyo constante ha sido mi mayor impulso.

Gracias por tu paciencia infinita, por saber cuándo animarme y cuándo simplemente estar, por tus palabras de aliento en los días difíciles y por celebrar conmigo cada pequeño logro. Has sido mi refugio, y no hay forma de expresar con palabras cuánto ha significado para mí tenerte cerca durante este proceso. Este trabajo también lleva tu huella, porque sin ti, todo habría sido mucho más cuesta arriba.

CONTENIDO

Contenido	1
Índice de Figuras	5
Índice de Tablas	7
1 Introducción y objetivos.....	9
1.1 Objetivos del Trabajo Fin de Grado	9
1.2 Organización de la memoria	10
2 Estado del arte	13
2.1 Inteligencia Artificial.....	13
2.1.1 Historia de la IA	13
2.1.2 Aprendizaje Automático.....	14
2.1.3 Clasificaciones, técnicas y algoritmos.....	15
2.2 IA Generativa.....	16
2.2.1 Tecnologías de IAG.....	16
2.3 IAG aplicada a la programación	18
2.3.1 Revisión de herramientas	18
2.3.2 Inteligencia Artificial Generativa con mayor adopción	19
2.3.3 Análisis de la IAG mediante técnicas de benchmarking.....	21
2.3.4 Integración en entornos de desarrollo	22
2.3.5 IAG en alumnado de cursos introductorios a la programación	23
2.3.6 Uso fraudulento de IAG	24
2.3.7 Retos, limitaciones y riesgos de la IAG	25
3 Desarrollo del TFG	27
3.1 Análisis de diagramas de flujo.....	27
3.1.1 Análisis de diagramas de flujo con las versiones de IAG gratuitas	28
3.1.2 Análisis de diagramas de flujo con las versiones de IAG de pago.....	29
3.2 Análisis de generación de código simple.....	30
3.2.1 Análisis de generación de código simple con las versiones gratuitas de las IAG.....	30
3.2.2 Análisis de generación de código simple con las versiones de pago de las IAG	31
3.3 Evaluación de tres proyectos de la asignatura Informática para la Ingeniería	32
3.3.1 Generación de código en el proyecto de Informática para la Ingeniería del curso 2019-2020	32
3.3.2 Generación de código en el proyecto de Informática para la Ingeniería del curso 2020-2021	37
3.3.3 Generación de código en el proyecto de Informática para la Ingeniería del curso 2021-2022	42

3.3.4 Resumen de desempeño de las IAG evaluadas	47
3.4 Empleo de una IAG como apoyo a la docencia.....	48
3.4.1 Evaluación por parte de la IAG aplicando criterios propios de corrección.....	48
3.4.2 Evaluación por parte de la IAG con criterios del profesorado	55
3.5 Creación y puesta en práctica de un GPT personalizado para la resolución de problemas específicos de la asignatura Informática para la Ingeniería.....	63
3.5.1 Creación del GPT personalizado	63
3.5.2 Puesta en práctica del GPT personalizado como apoyo al alumnado	65
3.5.3 Puesta en práctica del GPT personalizado como apoyo al profesorado en la corrección del proyecto	68
3.5.4 Puesta en práctica del GPT personalizado como apoyo al profesorado en la supervisión de las restricciones de uso	69
4 Análisis de los resultados.....	71
4.1 Resumen del desempeño global de cada IAG en la evaluación inicial	71
4.1.1 Desempeño de cada IAG gratuita en comprensión y generación de diagramas de flujo	71
4.1.2 Desempeño de cada IAG de pago en comprensión y generación de diagramas de flujo.....	72
4.1.3 Desempeño de cada IAG gratuita en generación de código simple	73
4.1.4 Desempeño de cada IAG de pago en generación de código simple.....	74
4.2 Resumen del desempeño de cada IAG en creación de código para proyectos complejos	74
4.2.1 Comparación con los resultados del benchmark SWE-bench Verified	76
4.3 Calificaciones de ChatGPT o3-mini-high con criterios propios frente a las del profesorado	77
4.4 Calificaciones de ChatGPT o3-mini-high con criterios propios del profesorado frente a las del profesorado	78
4.5 Comparativa entre las calificaciones de ChatGPT o3-mini-high con criterios propios y con criterios del profesorado	83
4.6 Efectividad del GPT personalizado	84
5 Conclusiones y líneas futuras.....	87
5.1 Conclusiones derivadas del TFG	87
5.2 Líneas futuras.....	88
6 Bibliografía	89
Anexo I: Implicaciones Sociales y Ambientales.....	95
AI.1 Altos requerimientos energéticos	95
AI.2 Implementación de la IA en la gestión ambiental	96
AI.3 Reducción del impacto ambiental	96
Anexo II: Reflexiones Éticas y Sociales	97
AII.1 Sesgos algorítmicos y falta de transparencia	97
AII.2 <i>Deepfakes</i> , automatización y desinformación.....	97

AII.3 Brecha económica y desigualdad de acceso.....	97
AII.4 Responsabilidad compartida y uso ético	98
Anexo III: Ejercicios de diagramas de flujo	99
AIII.1 Ejercicio 1	99
AIII.2 Ejercicio 2	100
AIII.3 Ejercicio 3	100
AIII.4 Ejercicio 4	101
Anexo IV: Código por generado por el GPT personalizado	103
AIV.1 Código del proyecto del curso 2019-2020 de la asignatura Informática para la Ingeniería ...	103
AIV.2 Código del proyecto del curso 2020-2021 de la asignatura Informática para la Ingeniería ...	107
AIV.3 Código del proyecto del curso 2021-2022 de la asignatura Informática para la Ingeniería ...	111
Anexo V: Glosario	115

ÍNDICE DE FIGURAS

Figura 2-1 Subconjuntos de IA	15
Figura 2-2 Clasificación de los diferentes tipos de algoritmos dentro del aprendizaje automático.....	16
Figura 2-3 Tecnologías de IAG	18
Figura 2-4 Comparativa de visitas mensuales entre cuatro IAG	20
Figura 2-5 Eficacia según el <i>benchmark SWE-bench Verified</i>	22
Figura 2-6 Clasificación de los IDE.....	23
Figura 3-1 Ejercicio 1 de interpretación por parte de las IAG.....	28
Figura 3-2 Desempeño de cada IAG gratuita en tareas de comprensión y generación de diagramas de flujo	29
Figura 3-3 Desempeño de cada IAG de pago en tareas de comprensión y generación de diagramas de flujo	30
Figura 3-4 Resolución de Microsoft Copilot al ejercicio 1 de generación de código.....	30
Figura 3-5 Desempeño de cada IAG en su versión gratuita en tareas de creación de código simple.....	31
Figura 3-6 Desempeño de cada IAG en su versión de pago en tareas de creación de código simple	31
Figura 3-7 Portada del proyecto Conecta 4.....	33
Figura 3-8 Interfaz generada por ChatGPT o3-mini-high	35
Figura 3-9 Interfaz generada por DeepSeek	35
Figura 3-10 Anomalías en el funcionamiento del programa generado por Gemini Advanced	36
Figura 3-11 Deficiencia funcional del programa generado por Copilot	37
Figura 3-12 Interfaz del programa generado por Copilot Pro.....	37
Figura 3-13 Portada del proyecto Juego del 8.....	38
Figura 3-14 Interfaz generada por ChatGPT o3-mini-high	39
Figura 3-15 Ausencia de ilusión visual en el código generado por DeepSeek	40
Figura 3-16 Errores de compilación de código generado por Gemini Advanced.....	41
Figura 3-17 Anomalía en el tablero generado por el código creado por Copilot.....	41
Figura 3-18 Anomalía en el tablero generado por el código creado por Copilot Pro	42
Figura 3-19 Portada del proyecto Juego del Calamar	43
Figura 3-20 Interfaz del código generado por ChatGPT o3-mini-high	45
Figura 3-21 Anomalías en la ejecución del programa creado por DeepSeek	46
Figura 3-22 Error de compilación en el código generado por Gemini Advanced	46
Figura 3-23 Anomalía de funcionamiento en el código generado por Copilot.....	47
Figura 3-24 Proceso de creación de un GPT personalizado	63
Figura 3-25 Archivos con los que se alimentó a ChatGPT	64
Figura 3-26 Previsualización del GPT personalizado.....	65

Figura 3-27 Errores del código generado por el GPT personalizado en el proyecto del curso 2019-2020	66
Figura 3-28 Interfaz de victoria del programa del año 2019-2020 creado por el GPT personalizado....	67
Figura 3-29 Interfaz de victoria del programa del año 2020-2021 creado por el GPT personalizado....	67
Figura 3-30 Interfaz de victoria del programa del año 2021-2022 creado por el GPT personalizado....	67
Figura 4-1 Rendimiento de cada IAG en su versión gratuita en comprensión y generación de diagramas de flujo	72
Figura 4-2 Rendimiento de cada IAG en su versión de pago en comprensión y generación de diagramas de flujo	73
Figura 4-3 Desempeño de cada IAG en su versión gratuita en generación de código simple.....	73
Figura 4-4 Desempeño de cada IAG en su versión de pago en generación de código simple	74
Figura 4-5 Porcentaje de eficacia de cada IAG en los tres proyectos de programación evaluados.....	75
Figura 4-6 Comparación entre los datos del <i>benchmark</i> con los obtenidos en el TFG	76
Figura 4-7 Ejemplo de enunciado de problema [52].....	77
Figura 4-8 Precisión de ChatGPT o3-mini-high con criterios propios	78
Figura 4-9 Precisión en las calificaciones de ChatGPT o3-mini-high sobre el funcionamiento	79
Figura 4-10 Precisión en las calificaciones de ChatGPT o3-mini-high sobre las mejoras	80
Figura 4-11 Precisión en las calificaciones de ChatGPT o3-mini-high sobre el sangrado.....	80
Figura 4-12 Precisión en las calificaciones de ChatGPT o3-mini-high sobre las funciones	81
Figura 4-13 Precisión en las calificaciones de ChatGPT o3-mini-high sobre los comentarios.....	82
Figura 4-14 Precisión de ChatGPT o3-mini-high con criterios del profesorado	82
Figura 4-15 Comparativa entre las calificaciones obtenidas por ChatGPT o3-mini-high con criterios propios y con criterios del profesorado.....	83
Figura 4-16 Porcentaje de eficacia de cada IAG en los tres proyectos analizados añadiendo la evaluación del GPT personalizado	84
Figura AI-1 Personas sin acceso a agua potable (ONU-HABITAT).....	95
Figura AIII-1 Ejercicio 1 de interpretación de diagramas de flujo	99
Figura AIII-2 Ejercicio 2 de interpretación de diagramas de flujo	100
Figura AIII-3 Ejercicio 3 de generación de diagramas de flujo [87]	101
Figura AIII-4 Ejercicio 4 de generación de diagramas de flujo [87]	102

ÍNDICE DE TABLAS

Tabla 2-1 Clasificación de la IA en función de sus capacidades y alcance	15
Tabla 2-2 Comparativa de versiones entre cuatro IAG.....	19
Tabla 2-3 Comparativa de visitas mensuales mundiales versus visitas mensuales en España	20
Tabla 2-4 Respuesta de cada IAG a cuál es la más utilizada	21
Tabla 3-1 Evaluación de las soluciones al proyecto de Informática para la Ingeniería del curso 2019-2020.....	34
Tabla 3-2 Evaluación de las soluciones al proyecto de Informática para la Ingeniería del curso 2020-2021.....	39
Tabla 3-3 Evaluación de las soluciones al proyecto de Informática para la Ingeniería del curso 2021-2022.....	44
Tabla 3-4 Resumen de desempeño de las IAG evaluadas en los tres proyectos analizados de la asignatura Informática para la Ingeniería	48
Tabla 3-5 Criterios generados por ChatGPT o3-mini-high a raíz del enunciado del proyecto del curso 2020-2021	50
Tabla 3-6 Corrección de ChatGPT o3-mini-high con criterios propios al proyecto del grupo 27	50
Tabla 3-7 Corrección de ChatGPT o3-mini-high con criterios propios al proyecto del grupo 25	52
Tabla 3-8 Corrección de ChatGPT o3-mini-high con criterios propios al proyecto del grupo 28	53
Tabla 3-9 Comparativa de calificaciones de ChatGPT o3-mini-high con criterios propios frente a las del profesorado.....	55
Tabla 3-10 Criterios de evaluación por parte del profesorado.....	55
Tabla 3-11 Errores definidos por el profesorado y penalización correspondiente	56
Tabla 3-12 Mejoras definidas por el profesorado y puntuación asignada	57
Tabla 3-13 Evaluación de la entrega del grupo 2 con criterios propios del profesorado.....	57
Tabla 3-14 Evaluación de la entrega del grupo 5 con criterios propios del profesorado.....	57
Tabla 3-15 Evaluación de la entrega del grupo 11 con criterios propios del profesorado.....	58
Tabla 3-16 Evaluación de la entrega del grupo 14 con criterios propios del profesorado.....	58
Tabla 3-17 Evaluación de la entrega del grupo 17 con criterios propios del profesorado.....	59
Tabla 3-18 Evaluación de la entrega del grupo 19 con criterios propios del profesorado.....	59
Tabla 3-19 Evaluación de la entrega del grupo 25 con criterios propios del profesorado.....	59
Tabla 3-20 Evaluación de la entrega del grupo 26 con criterios propios del profesorado.....	60
Tabla 3-21 Evaluación de la entrega del grupo 27 con criterios propios del profesorado.....	60
Tabla 3-22 Evaluación de la entrega del grupo 28 con criterios propios del profesorado.....	61
Tabla 3-23 Comparativa de calificaciones de ChatGPT o3-mini-high con sus propios criterios y con los criterios del profesorado.....	62
Tabla 3-24 Relación de las partes del código con los conceptos de la asignatura según el GPT personalizado	65
Tabla 3-25 Análisis del código generado por el GPT personalizado en cada proyecto.....	66

Tabla 3-26 Evaluación del GPT personalizado del proyecto del grupo 25 del curso 2020-2021 con criterios propios del profesorado.....	69
Tabla 3-27 Respuesta del GPT personalizado a la revisión de elementos no permitidos en el proyecto del curso 2020-2021 de la asignatura Informática para la Ingeniería	70

1 INTRODUCCIÓN Y OBJETIVOS

En el ámbito educativo, la innovación en procesos de enseñanza y aprendizaje aprovechando las oportunidades que ofrecen las nuevas tecnologías es una constante. En el marco de una formación integral como la impartida en la Escuela Naval Militar (ENM), en un contexto donde el tiempo disponible para el estudio es limitado, el empleo eficaz de los recursos es básico para la obtención de un mayor rendimiento. A su vez, como indica [1], cabe señalar el gran papel que desempeña la educación superior al orientar sus planes de estudio a la dotación de habilidades necesarias para el futuro de sus estudiantes.

Con la creciente incorporación de la Inteligencia Artificial (IA) en diversos sectores, la Inteligencia Artificial Generativa (IAG) emerge como una herramienta prometedora para apoyar el aprendizaje de los estudiantes. Esta tecnología, que tiene la capacidad de generar contenido, resolver problemas y proporcionar retroalimentación, podría revolucionar la forma en que el alumnado aprende a programar y cómo los docentes gestionan los procesos educativos. El avance de la IAG parece revolucionar la praxis educativa existente, lo que genera interés tanto de los propios docentes como del alumnado que se quiere aprovechar de su efectividad. Según el informe [2], un 86% de los estudiantes universitarios usa IAG, sin embargo, una cuestión muy pertinente podría ser si sus conocimientos de uso son los adecuados. Como se muestra en [3], la alfabetización en IA ha surgido recientemente en la agenda educativa, generando expectativas sobre el conocimiento profesional de los docentes y formadores de docentes.

El estudio [4] exploró la literatura existente sobre el uso y puesta en práctica de la IAG en la enseñanza y en el aprendizaje. Este encontró tanto beneficios como inconvenientes. Por un lado, de manera positiva, se debe destacar la accesibilidad a un aprendizaje personalizado e interactivo, generando recursos educativos y la facilitación de un feedback continuo, regulando e informando así del proceso de enseñanza. Por otro lado, negativamente, la IAG puede generar información incorrecta, propagar sesgos presentes en los datos de entrenamiento o difundir prejuicios.

Por lo tanto, es de suma importancia examinar el impacto de la IA en el ámbito educativo y sus desafíos clave a la hora de su implementación en las aulas. Tal y como indican los resultados de la investigación en [5] se ponen en manifiesto tres líneas a atender sobre la IA: la privacidad, el uso ético y la importancia de algoritmos confiables, viendo a su vez necesario más investigación al respecto.

1.1 Objetivos del Trabajo Fin de Grado

El presente Trabajo Fin de Grado (TFG) tiene como objetivo evaluar las herramientas basadas en IAG en términos de capacidad, prestaciones y coste, para su uso en el apoyo al aprendizaje y la docencia de la programación. A través de un enfoque cuantitativo y cualitativo, se pretende analizar la efectividad de estas herramientas en diversos aspectos del proceso formativo, tales como:

- Resolución de problemas y generación de ejercicios: Evaluar la capacidad de las IAG para resolver problemas relacionados con diagramas de flujo, ejercicios de programación en lenguaje C, exámenes y proyectos de programación empleados en la asignatura Informática para la Ingeniería, enmarcada dentro del grado de Ingeniería Mecánica impartido en el Centro Universitario de la Defensa en la ENM.
- Asistencia al aprendizaje: Valorar cómo las IAG pueden asistir a los estudiantes en la resolución de dudas reales y en la corrección de errores en el código.
- Asistencia al docente: Evaluar el potencial de la IAG como herramienta asistencial en tareas educativas, específicamente en el apoyo a la evaluación de proyectos académicos, analizando su eficacia, precisión y capacidad para optimizar el proceso evaluativo realizado por el profesorado.
- Otros usos potenciales: Identificar, a partir de una revisión exhaustiva de la literatura, otros posibles usos de la IAG que puedan mejorar la experiencia de aprendizaje en programación.
- Desarrollar un modelo personalizado de IAG como prueba de concepto. Este modelo integrará documentación específica de la asignatura, información contextual sobre el curso y las características del alumnado, con el fin de ofrecer respuestas adaptadas a las necesidades de los estudiantes. El sistema propuesto no solo se limitará a proporcionar soluciones inmediatas, sino que también servirá como herramienta de apoyo a la tarea de evaluación llevada a cabo por el docente.

Finalmente, el trabajo analizará brevemente el impacto social y medioambiental asociado al uso de modelos de Inteligencia Artificial Generativa, con el fin de contextualizar su aplicación desde una perspectiva ética y sostenible.

Asimismo, este TFG se enmarca dentro de un proyecto de innovación docente que el CUD-ENM está desarrollando durante el curso académico 2024-2025, bajo el título *“Evaluación del uso de herramientas de IA Generativa como apoyo al proceso de aprendizaje de habilidades prácticas de la ingeniería”*. Dicho proyecto tiene como propósito principal que el alumnado del centro conozca el funcionamiento de las aplicaciones basadas en Inteligencia Artificial Generativa, favoreciendo así la capacidad crítica necesaria para evaluar la fiabilidad de los resultados proporcionados por estas herramientas.

1.2 Organización de la memoria

La presente memoria se estructura en cinco capítulos principales y un sexto dedicado a la bibliografía, con el objetivo de ofrecer una visión clara y progresiva del trabajo realizado en torno al uso de la Inteligencia Artificial Generativa en contextos educativos universitarios, especialmente en el ámbito de la programación y el apoyo al alumnado y profesorado.

En el Capítulo 1 se introduce el contexto general del trabajo, su motivación, objetivos y metodología. A continuación, el Capítulo 2 se centra en el estado del arte, comenzando con un repaso histórico de la evolución de la Inteligencia Artificial, de sus técnicas y clasificación. Posteriormente, se profundiza en el campo específico de la Inteligencia Artificial Generativa, analizando sus herramientas actuales, aplicaciones más destacadas, así como sus principales retos y limitaciones. Esta sección se completa con una revisión de trabajos académicos relevantes que abordan temáticas similares a las del presente TFG.

El Capítulo 3 describe en detalle el desarrollo del trabajo. En primer lugar, se presenta una evaluación inicial basada en proyectos sencillos centrados en el análisis de diagramas de flujo y la generación automática de pequeños fragmentos de código. A continuación, se analiza el uso de la IAG en el desarrollo de proyectos de programación reales de estudiantes de la asignatura Informática para la Ingeniería, evaluando su precisión y capacidad resolutoria. Posteriormente, se examina el potencial de la IAG como herramienta de apoyo al profesorado en tareas como la generación de criterios de evaluación y la corrección de proyectos. Finalmente, se presenta un GPT personalizado capaz de llevar a cabo las

tareas anteriormente mencionadas basándose en el temario de la asignatura Informática para la Ingeniería.

En el Capítulo 4 se lleva a cabo un análisis de los resultados obtenidos durante la implementación de las distintas fases del TFG, presentando tanto los logros alcanzados como las dificultades encontradas. Este análisis permite extraer conclusiones fundamentadas sobre el impacto real de la IAG en el contexto educativo abordado.

El Capítulo 5 recoge las conclusiones del trabajo y plantea posibles líneas de investigación y desarrollo futuro.

El Capítulo 6 presenta la bibliografía utilizada a lo largo del TFG.

Finalmente, se recogen una serie de anexos con la documentación complementaria relevante para una mejor comprensión del trabajo realizado, entre los que destacan: un anexo de implicaciones sociales y ambientales, un anexo de reflexiones éticas, un anexo con los ejercicios propuestos de diagramas de flujo, un anexo con el código generado por el GPT personalizado y, finalmente, en el último anexo se recoge un glosario de siglas.

2 ESTADO DEL ARTE

La Real Academia Española define la Inteligencia Artificial [6] como aquella “disciplina científica que se ocupa de crear programas informáticos que ejecutan operaciones comparables a las que realiza la mente humana, como el aprendizaje o el razonamiento lógico”. En este capítulo se presenta la contextualización, reseñando brevemente la historia de la Inteligencia Artificial y una introducción al aprendizaje automático. Seguidamente se introduce la IA Generativa como un poderoso recurso digital para respaldar la programación, considerando las principales herramientas destacadas en este campo y las posibilidades de integración en entornos de desarrollo. Asimismo, se lleva a cabo una revisión exhaustiva de los trabajos académicos relacionados con esta temática; investigando cuál es la IAG más utilizada, cómo se emplea en procesos educativos con alumnos de cursos introductorios a la programación y qué se considera un uso fraudulento de cualquier modelo generativo. Por último, se examinan los desafíos, las limitaciones y los riesgos que, en el primer cuartil del año 2025, la IAG presenta.

2.1 Inteligencia Artificial

2.1.1 Historia de la IA

Desde la antigua Grecia, la creación del Mecanismo de Anticitera [7] supuso un ejemplo temprano de tecnología avanzada y pensamiento complejo. Se puede considerar como un precursor indirecto de la Inteligencia Artificial ya que, con la automatización de ciertos movimientos, era capaz de predecir posiciones astronómicas y eclipses.

Sin embargo, no es hasta el año 1950 cuando Alan Turing, famoso por ser uno de los padres de las ciencias de la computación y por decodificar el código de cifrado alemán ENIGMA [8] durante la Segunda Guerra Mundial, publica *Computing Machinery and Intelligence* [9]. En este artículo propone la pregunta: “¿Pueden pensar las máquinas?”. A raíz de ello plantea el famoso Test de Turing como una forma de evaluar la inteligencia de las máquinas, argumentando que, si una máquina puede imitar el comportamiento humano de manera indistinguible, debería considerarse “inteligente”. Este punto marcó el inicio de debates clave sobre la relación entre la mente humana y las máquinas.

Durante las décadas de 1950 y 1960, pioneros como John McCarthy y Marvin Minsky comenzaron a desarrollar los primeros programas de IA. En 1956, McCarthy, quien había acuñado el término “Inteligencia Artificial”, organizó la Conferencia de Dartmouth [10]. En ella, diez hombres llevaron a cabo un estudio de dos meses con el fin de tratar temas como: ordenadores automáticos, redes neuronales o cómo se puede programar un ordenador para utilizar un lenguaje, entre otros. Se considera este punto el nacimiento de la IA como campo de estudio académico.

Pese al gran esfuerzo que muchos científicos estaban poniendo en investigar sobre el tema, durante la década de 1970 y parte de 1980 la IA sufrió un periodo de desaceleración conocido como el “invierno de la IA” debido a las limitaciones tecnológicas de la época. Sin embargo, esta falta de avances pronto cambiaría, ya que según [11], a mediados de 1980, las redes neuronales que utilizan un algoritmo de retropropagación para entrenarse, comienzan a ser ampliamente utilizadas en aplicaciones de IA.

El avance más notable ocurrió en la década de 2000 y principios de 2010, cuando la disponibilidad de grandes cantidades de datos (*Big Data*) y el poder de cómputo masivo (GPU), facilitaron los avances en modelos de aprendizaje profundo como las redes neuronales convolucionales (CNN) y las redes neuronales recurrentes (RNN). A partir de 2010, aplicaciones de IA como asistentes virtuales (como Siri y Alexa), vehículos autónomos, reconocimiento facial y procesamiento de lenguaje natural (como los modelos GPT de OpenAI) comenzaron a tener una presencia importante en la vida cotidiana.

En la actualidad, el auge de la IAG supone un cambio significativo en el rendimiento de la IA, abarcando diversos aspectos sociales como son: el apoyo al procesamiento de textos, la revisión lingüística y la traducción, los agentes conversacionales... Son muchos los países que están apoyando la investigación en este ámbito. De acuerdo con [12], Estados Unidos busca consolidarse como líder en el ámbito de la IA a nivel mundial con la aportación de grandes cantidades de dinero a la investigación. Sin embargo, como se ve en [13], la Unión Europea también está buscando avances en esta tecnología, con España e Italia a la cabeza en inversión económica. Por otro lado, China busca adelantarse en la carrera por tener el dominio sobre la IA, siendo muchas las empresas que buscan afianzar sus investigaciones en este país, tal y como muestra [14].

2.1.2 Aprendizaje Automático

Como se refleja en [15] el aprendizaje automático (ampliamente conocido por su nombre en inglés: *machine learning*, ML) es uno de los campos más estudiados de la computación moderna. Además, es un dominio vital para la Inteligencia Artificial. Este artículo destaca el aprendizaje como una cualidad meramente humana que se ha convertido en una característica esencial para las máquinas.

El aprendizaje automático, tal y como se muestra en [16], se entiende como un área interdisciplinar, la cual incluye teoría de probabilidad, estadística, teoría de aproximación, y otras disciplinas. Se puede definir como un método de análisis de datos que automatiza la creación de modelos analíticos. Permite a los ordenadores aprender y mejorar de la experiencia sin necesidad de una programación explícita.

El aprendizaje profundo, como se muestra en la Figura 2-1, puede verse como un subconjunto del aprendizaje automático, que a su vez es un subconjunto dentro de lo que se conoce como IA. Según se evidencia en [17], tanto el aprendizaje automático como el aprendizaje profundo, utilizan algoritmos para aprender de datos, con el propósito de mejorar la precisión y la eficiencia de las predicciones o decisiones. Sin embargo, el aprendizaje automático utiliza en buena medida la estadística para aprender de los datos mientras que el aprendizaje profundo emplea fundamentalmente redes neuronales con múltiples capas para modelar patrones complejos en grandes volúmenes de datos.

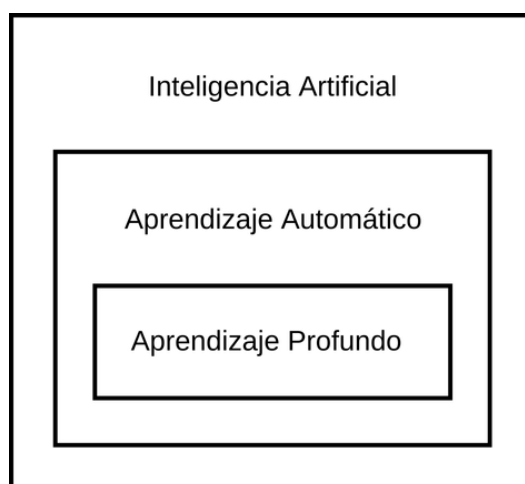


Figura 2-1 Subconjuntos de IA

Como se puede ver en [18], los modelos de aprendizaje automático superan tres fases durante su vida útil. La primera de ellas es un entrenamiento genérico en el que reciben datos de parámetros de los cuales la respuesta es conocida. La segunda fase es el ajuste específico para una aplicación. El algoritmo se ejecuta hasta que la salida coincide con la respuesta conocida. Durante la fase tres, se lleva a cabo la generación, acto seguido se evalúa y se concluye con un reajuste. De esta forma, la información que gestiona aumenta progresivamente. La fase tres es circular, es decir, por cada respuesta del aprendizaje automático, se ejecuta la evaluación y el reajuste.

2.1.3 Clasificaciones, técnicas y algoritmos

La IA se puede clasificar en función de diversos factores según la bibliografía consultada. De acuerdo con lo observado en [19], según las capacidades que aporta, se puede estructurar en tres grandes grupos. Para conocer las características de cada uno de estos grupos, se consultó la página web de International Business Machines Corporation (IBM) [20]. Sin embargo, pese a tener una distribución claramente definida, en enero de 2025, sólo se conoce la IA Débil, como se muestra en la Tabla 2-1. Tanto la IA General como la Súper IA, son meros conceptos teóricos a los que se espera llegar con la evolución de la tecnología actual.

Clasificación	Descripción
IA Débil / Estrecha (WAI)	Es el único tipo de IA existente. Se puede entrenar para que lleve a cabo una tarea, normalmente más rápido y con mayor precisión que un humano. Son sistemas que resuelven tareas concretas.
IA General (AGI)	No existe. Por el momento es un concepto teórico. Podrá utilizar conocimientos y habilidades previas para completar tareas en diferentes contextos sin interacción humana. Será similar e incluso superior al funcionamiento del pensamiento humano. Actualmente, algunos expertos consideran que la AGI podría ser alcanzada en pocos años [21].
Súper IA (SAI)	Es completamente teórica. Se espera que supere las capacidades humanas. Podría llegar a entender los sentimientos humanos y las experiencias para crear sus propias necesidades y deseos.

Tabla 2-1 Clasificación de la IA en función de sus capacidades y alcance

Las técnicas de Inteligencia Artificial se refieren a un conjunto de métodos y algoritmos diseñados para crear sistemas inteligentes que puedan llevar a cabo tareas que demandan un nivel de inteligencia comparable al humano, como dictan en [22] y [23].

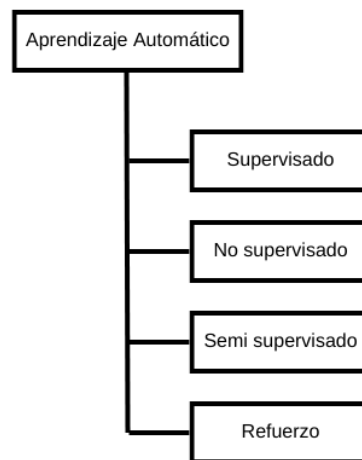


Figura 2-2 Clasificación de los diferentes tipos de algoritmos dentro del aprendizaje automático

Como se ve en la Figura 2-2, existen cuatro tipos de aprendizaje dentro del ML. El aprendizaje supervisado tiene un papel fundamental en tareas que requieren el reconocimiento de patrones y la predicción. El desarrollador establece los límites y las reglas que el algoritmo debe seguir. Su uso es extenso en labores de clasificación de imágenes o sistemas de recomendación. En el aprendizaje no supervisado, el sistema analiza datos con salidas no predefinidas. Un ejemplo de aprendizaje no supervisado, es el conocido como *clustering* (tipo de aprendizaje crucial para organizar datos en grupos por su similitud). Por otro lado, el aprendizaje semi supervisado es un modelo que utiliza una pequeña porción de datos etiquetados para entrenar un modelo que ayude a etiquetar al resto de datos. De esta forma se puede conseguir un modelo de aprendizaje supervisado minimizando el coste del etiquetado de los datos. Por último, en el aprendizaje por refuerzo se utiliza un método de ensayo y error para aprender de los resultados y decidir qué acción tomar a continuación. Aprende a través de recompensas y penalizaciones para optimizar acciones.

2.2 IA Generativa

Tal y como se ve en [24], la IA Predictiva basa su análisis en estadística acoplada a algoritmos de aprendizaje automático. Aprende patrones existentes en extensas bases de datos para anticiparse a posibles soluciones a preguntas futuras. Por otro lado, el mismo artículo menciona que la IA Generativa crea nuevos datos manteniendo como referencia las bases de datos con las que ha sido entrenada.

La IA Generativa es entrenada con cantidades masivas de datos en bruto. Esto incluye: resultados sin procesar de encuestas o experimentos, registros de sensores o dispositivos electrónicos y datos extraídos de bases de datos antes de realizarles cualquier tipo de análisis. De acuerdo con lo expuesto en [25], estos modelos extraen los patrones de sus datos de entrenamiento para comprender las solicitudes de los usuarios y crear un contenido nuevo y similar al original, pero no idéntico. Es capaz de generar una respuesta totalmente diferente cada vez que se le introduce el mismo *prompt* (solicitud diseñada para guiar la interacción entre un usuario y la IA), sin alterar la raíz de la respuesta. Como se evidencia en [26], las respuestas mantendrán invariable su significado, aunque diferirán en la estructura y el estilo de su formulación.

2.2.1 Tecnologías de IAG

Según se presenta en [27], la IA Generativa incluye varios tipos de IA, habiendo sido creada cada una para responder a necesidades específicas o formas de generación diferentes. Tal y como se presenta

en la Figura 2-3, algunas de las más conocidas son: redes generativas antagónicas (*Generative Adversarial Networks*, GAN), autocodificadores variacionales (*Variational Autoencoders*, VAE), modelos de difusión (*Diffusion Models*, DM), modelos de transformador (*Transformer Models*, TRM) y modelos de lenguaje de gran tamaño (*Large Language Models*, LLM), entre otros.

En primer lugar, como muestra [28], las GAN aprenden distribuciones complejas y de alta dimensión de forma implícita sobre imágenes, audio y datos. Sin embargo, existen retos importantes en el entrenamiento de GAN, por ejemplo: colapsos, no convergencia e inestabilidad. Esto es debido al diseño inadecuado de la arquitectura de red, el uso de la función objetivo y la selección del algoritmo de optimización. Recientemente, para hacer frente a estos retos, se han investigado varias soluciones para mejorar el diseño y la optimización de las GAN basadas en técnicas de arquitecturas de red rediseñadas, nuevas funciones objetivo y algoritmos de optimización alternativos.

En segundo lugar, los autocodificadores variacionales, según [29], se basan en un marco probabilístico flexible, robusto frente a la escasez de datos y compatible con otros modelos basados en el aprendizaje profundo para tratar datos multimodales. Además, los algoritmos de recomendación basados en VAE han dado lugar a muchos modelos gráficos novedosos, y han logrado un rendimiento prometedor. Como se muestra en [30], esto permite que un VAE no solo reproduzca con fidelidad la entrada original, sino que también emplee la inferencia variacional para generar nuevas muestras de datos con características similares a las originales.

En tercer lugar, los modelos de difusión son utilizados principalmente para la generación de imágenes y tareas similares de visión artificial, como muestra [31]. En una primera instancia, estas redes son entrenadas mediante algoritmos de aprendizaje profundo para crear imágenes con ruido arbitrario. Posteriormente, invierten el proceso para desarrollar imágenes de muy alta calidad.

En cuarto lugar, los modelos de transformador son un tipo de modelo de aprendizaje profundo introducido a comienzos de 2017, como muestra [32]. Ocho años más tarde, en 2025, estos modelos se han convertido rápidamente en piezas clave del procesamiento del lenguaje natural (PLN) y han sido aplicados en diversas tareas de aprendizaje automático e Inteligencia Artificial, como se ve en [33]. Además, el empleo de los modelos de transformador se encuentra muy extendido en aplicaciones diversas, entre las que destaca la formación de modelos de lenguaje de gran tamaño. Los LLM constituyen una de las tecnologías clave dentro de la IAG. Son una clase de modelos entrenados con grandes volúmenes de texto para predecir y generar secuencias lingüísticas coherentes. Utilizan arquitecturas profundas, como las redes neuronales tipo *transformer* (redes que procesan toda la secuencia de entrada de manera simultánea, lo que permite un mayor paralelismo y una mejor eficiencia en el entrenamiento), y son capaces de comprender y producir lenguaje natural, lo que los convierte en herramientas fundamentales dentro de la Inteligencia Artificial Generativa. Además, como se indica en [34], los LLM están revolucionando el proceso tanto de escritura como de análisis de código, por lo que tendrán especial relevancia en este TFG.

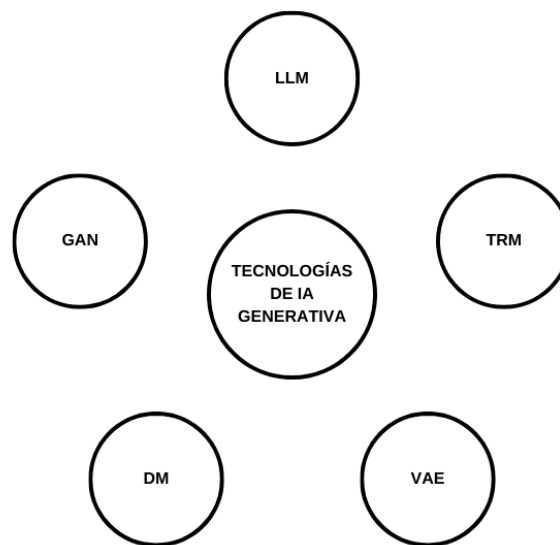


Figura 2-3 Tecnologías de IAG

2.3 IAG aplicada a la programación

2.3.1 Revisión de herramientas

En el contexto del desarrollo de software, la selección de un modelo de Inteligencia Artificial Generativa adecuado requiere un análisis detallado de sus versiones, características y costes. La siguiente comparación, en la Tabla 2-2, examina cuatro modelos de IAG, evaluando sus diferentes versiones, precios y funcionalidades. Para la selección de las IAG a evaluar en este TFG se siguieron los criterios mostrados en el apartado 2.3.2 y, adicionalmente, se introdujo DeepSeek ya que, según una gran variedad de fuentes como [35], representa una novedad más barata y eficiente que sus competidores directos.

Plataforma	Modelos y precios [€/mes]	Características
Aclaración: ChatGPT ofrece sus servicios con precios establecidos en dólares estadounidenses. Para este trabajo, los valores han sido convertidos a euros utilizando el tipo de cambio vigente el 27/03/2025, fijado en 1€ = 1,08\$.		
ChatGPT [36]	ChatGPT: 0€	Acceso a GPT-4o mini. Limitación en carga de archivos, análisis avanzado de datos y generación de imágenes. Acceso limitado a GPT-4o.
	ChatGPT Plus: 18,6€	Límites de uso ampliados y acceso a varios modelos de razonamiento (o3-mini, o3-mini-high y o1). Acceso limitado a generación de vídeos con Sora.
	ChatGPT Pro: 185,7€	Acceso completo a GPT-4o y a la versión o1 pro que utiliza más computación para buscar respuestas más precisas.
	ChatGPT Team: 23,2€	Incluye o3-mini, o3-mini-high y o1 y mayores límites en GPT-4o. Permite el acceso a herramientas como DALL-E.
	ChatGPT personalizado Enterprise:	Plan dirigido a empresas. Ofrece características avanzadas y personalización según las necesidades corporativas.

Gemini [37]	Gemini: 0€	Acceso a Gemini 2.0 Flash. En el momento de desarrollo de este TFG (febrero de 2025) carece de la capacidad de interpretar archivos en formato PDF, pese a estar considerada como multimodal.
	Gemini Advanced: 21,99€ (primer mes gratuito)	Ventana de contexto extendida a 1 millón de <i>tokens</i> (unidades básicas de texto que permiten a los modelos de IAG comprender y generar lenguaje).
		Incluye suscripción a Google One.
Copilot [38]	Copilot: 0€	Chat limitado y generación máxima de 15 imágenes por día.
	Copilot en Microsoft 365: 2,5€	Generador de imágenes avanzado, asistencia en las aplicaciones de Office. Chat limitado y 60 créditos de IA para funciones avanzadas.
	Copilot Pro: 22€	Herramientas adicionales como <i>Think Deeper</i> y visión mejorada (<i>Vision</i>).
	Microsoft 365 Copilot: 30€	Interpretación de código, agentes especializados en la web, subida de archivos y creación de imágenes.
DeepSeek [39]	DeepSeek-V3: 0€	Alternativa económica con modelos de código abierto. Es escalable para empresas y desarrolladores.

Tabla 2-2 Comparativa de versiones entre cuatro IAG

La publicación de versiones gratuitas de Inteligencia Artificial Generativa ha democratizado el acceso a herramientas avanzadas, permitiendo a estudiantes, creadores de contenido y emprendedores optimizar tareas sin coste. Ahora, es posible generar texto, imágenes y código fácilmente, mejorando la productividad y creatividad en distintos campos, aunque con ciertas limitaciones en velocidad, precisión y funcionalidades avanzadas.

Optar por una versión pro ofrece ventajas como una mayor capacidad de procesamiento, respuestas más precisas y fiables y acceso a herramientas avanzadas, tal y como se ve en [40]. Por un lado, los usuarios profesionales optan a tiempos de respuesta más rápidos, mayor seguridad y personalización. Por otro lado, permite automatizar tareas a gran escala, mejorar la integración con otras plataformas y garantizar un flujo de trabajo más eficiente.

2.3.2 Inteligencia Artificial Generativa con mayor adopción

A la hora de buscar la respuesta a qué IAG es la más utilizada, parece haber un claro dominador. Las estadísticas mostradas en [49] identifican a ChatGPT como la IAG predominante en agosto de 2023, cubriendo esta, hasta el 60% del tráfico analizado. Para ello, se empleó el software SEMrush [50]: una nueva herramienta de la industria SEO (optimización de los motores de búsqueda) destinada a mejorar la visibilidad online y a descubrir datos para llevar a cabo procesos de marketing. Entre sus numerosas funciones destaca la de análisis de tráfico entre páginas web previamente seleccionadas.

Con el fin de actualizar los datos a enero de 2025, se consultó el mismo software que en el estudio anteriormente mencionado, tanto para verificar los datos mundiales como para identificar aquellos registros específicamente nacionales. Cabe destacar que se introdujo AlphaCode [51], la cual es una IAG específicamente destinada a la generación de código de programación, para comprobar la diferencia de tráfico entre las IAG multipropósito y las IAG específicas para programación. Los resultados obtenidos a nivel mundial se ven en la Figura 2-4. Claramente, ChatGPT sigue siendo la IAG más utilizada. Se

muestra una tendencia alcista pasando de los tres mil millones de visitas mensuales a los casi cinco mil millones en tan solo el segundo semestre del año 2024.

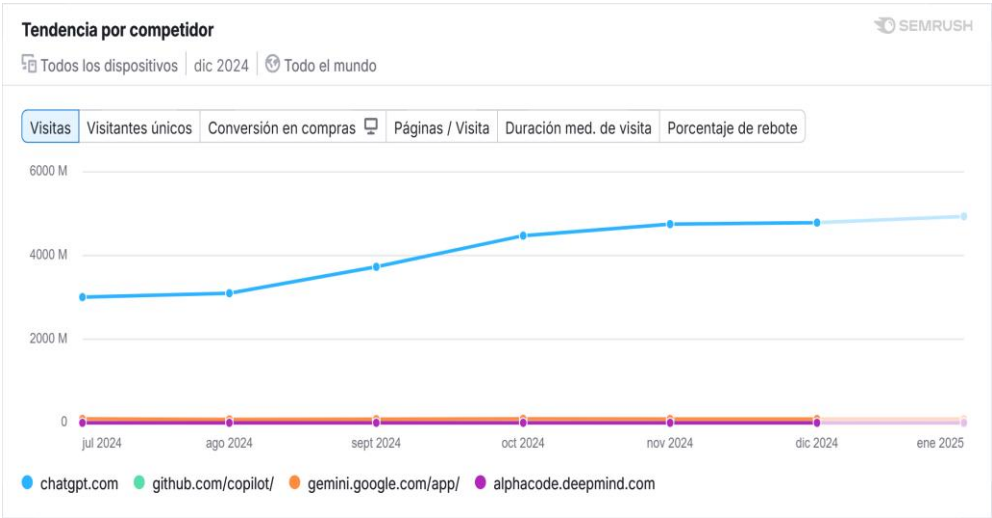


Figura 2-4 Comparativa de visitas mensuales entre cuatro IAG

Acto seguido, se ganeró la Tabla 2-3, cuyos datos se extrajeron del mismo software. En esta tabla se muestra la cantidad de visitas que tienen cada una de las cuatro IAG anteriormente mencionadas, tanto a nivel mundial como en España, que confirman que ChatGPT sigue siendo el LLM más visitado. Las consultas que recibe su página web desde territorio nacional representan un 1,8% de las globales. Además, cada una de estas visitas tiene una duración media de 09:05 minutos. Es interesante comprobar la gran diferencia que existe entre aquellas IAG de uso general, como son las tres primeras, y AlphaCode.

IAG	Visitas mundiales	Visitas en España	Visitas en España sobre el total	Duración de cada visita
ChatGPT	4788,1 M	85,7 M	1,8%	09:05 min
Gemini	91,7 M	1,3 M	1,4%	06:32 min
GitHub Copilot	917,3 mil	12,1 mil	1,3%	03:49 min
AlphaCode	3,8 mil	487	12,8%	01:48 min

Tabla 2-3 Comparativa de visitas mensuales mundiales versus visitas mensuales en España

Puesto que este TFG aborda el empleo de la IAG, se le preguntó a las cuatro IAG anteriormente mencionadas: ¿Cuál es la Inteligencia Artificial Generativa más utilizada a día 22 de enero de 2025? Las respuestas fueron las mostradas en la Tabla 2-4 destacando, una vez más, a ChatGPT de OpenAI como el LLM de referencia.

IAG	Respuesta por parte de la IAG
ChatGPT	“A fecha del 22 de enero de 2025, la herramienta de Inteligencia Artificial Generativa (IAG) más utilizada es ChatGPT, desarrollada por OpenAI”.

Gemini	“Algunas de las IAG que han destacado en los últimos meses y que podrían considerarse entre las más utilizadas son: ChatGPT y Google Bard”.
GitHub Copilot	“Algunas de las más mencionadas y utilizadas incluyen: ChatGPT, Alia y SLM (Modelos Lingüísticos Pequeños)”.
AlphaCode	Sin resultado. Es una IAG específica para programación.

Tabla 2-4 Respuesta de cada IAG a cuál es la más utilizada

2.3.3 Análisis de la IAG mediante técnicas de benchmarking

El *benchmarking* es una práctica crucial en el desarrollo de la IAG, ya que proporciona una forma estandarizada de evaluar y comparar el rendimiento de diferentes sistemas. En el presente TFG se seleccionó el *benchmark SWE-bench Verified* [52] como herramienta de evaluación para analizar la capacidad de modelos de lenguaje generativo en tareas de programación. La elección de este *benchmark* se fundamenta en sus características diferenciadoras frente a otras alternativas más tradicionales, como *HumanEval* [53].

SWE-bench Verified se basa en repositorios reales de código abierto y recopila *issues* documentados (entradas en el sistema de seguimiento de un repositorio que permiten reportar errores, sugerir mejoras o plantear dudas relacionadas con el funcionamiento del software) junto con los *pull requests* (solicitud formal que realiza un desarrollador para que los cambios efectuados en una rama de desarrollo sean revisados e integrados en la rama principal del proyecto) que los acompañan, permitiendo así evaluar la competencia de un modelo en contextos auténticos de desarrollo de software. A diferencia de los *benchmarks* sintéticos, este conjunto de datos requiere que el modelo comprenda el contexto completo de un proyecto real, genere código adecuado y mantenga la coherencia estructural y funcional del sistema. Además, la versión *Verified* garantiza una mayor fiabilidad en la evaluación, ya que los casos incluidos han sido revisados manualmente para asegurar su validez y relevancia.

Estas características hacen de *SWE-bench Verified* una herramienta especialmente adecuada para valorar el rendimiento de modelos aplicados a contextos profesionales o educativos avanzados, donde se espera que la generación de código sea útil, precisa y contextualizada.

En la Figura 2-5 se muestra el porcentaje de problemas correctamente resueltos por distintas IAG en su versión gratuita según el *benchmark SWE-bench Verified*, lo que permite comparar su eficacia en tareas de programación dentro de contextos reales de desarrollo de software. Cabe destacar que los datos de las versiones de pago no son públicos. Los resultados reflejan un alto nivel de desempeño en todos los modelos analizados, superando todos ellos el 70 % de eficacia, con ChatGPT-4o destacando como el modelo más preciso, cercano al 90 % de aciertos.

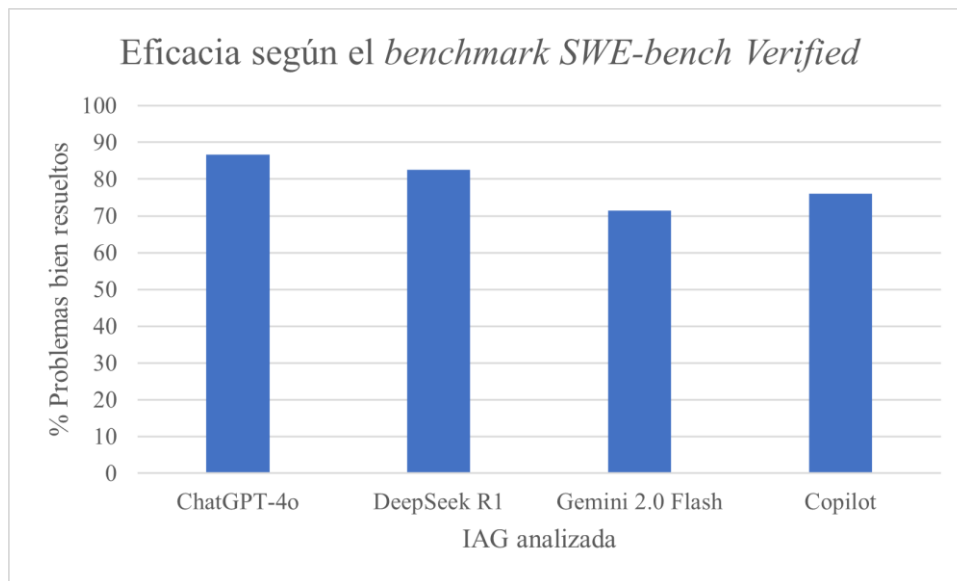


Figura 2-5 Eficacia según el benchmark SWE-bench Verified

Este nivel de rendimiento indica que las IAG actuales son ya capaces de afrontar con éxito una gran variedad de tareas complejas relacionadas con el análisis, comprensión y generación de código en entornos reales. Además, es importante señalar que estos porcentajes no han sido estáticos: según informes recientes [54], la eficacia de los modelos ha venido incrementándose de manera constante a un ritmo estimado de mejora del 5 % mensual hasta finales de 2024. Esta evolución sostenida es consecuencia tanto del refinamiento de los modelos como del aumento de los datos de entrenamiento específicos del ámbito de la programación.

En conjunto, los resultados presentados confirman que las IAG evaluadas mediante el benchmark *SWE-bench Verified* no solo han alcanzado un grado de madurez notable en la resolución de problemas de software, sino que siguen mejorando de forma acelerada. Esto refuerza su potencial como herramientas de apoyo para docentes y estudiantes en tareas que hasta hace poco requerían exclusivamente intervención humana especializada.

En este trabajo, los resultados obtenidos durante el análisis de las diferentes IAG se compararán con los datos de este *benchmark*, con el objetivo de analizar su comportamiento y su potencial como asistente en tareas relacionadas con la programación, tal y como se muestra en el apartado 4.2.1.

2.3.4 Integración en entornos de desarrollo

Un entorno de desarrollo integrado (IDE) es una aplicación de software que integra múltiples funciones esenciales para la creación, edición, depuración y prueba de programas informáticos dentro de un entorno unificado y coherente. En [41] se muestra que un IDE está formado como mínimo por: un editor, un compilador, un desbloqueador, y las funcionalidades de compleción de código o gestión de código genérico. Los IDE se clasifican según se muestra en Figura 2-6.

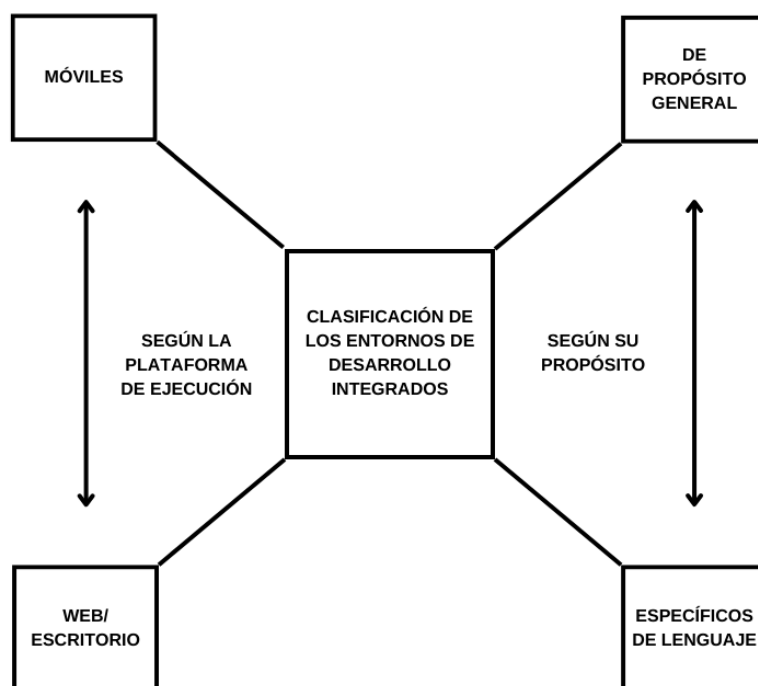


Figura 2-6 Clasificación de los IDE

Todas las herramientas de IAG revisadas en el punto 2.3.1 pueden utilizarse en entornos de desarrollo integrados de diferentes formas, generalmente mediante extensiones, integraciones o en interfaces de programación de aplicaciones o API. Estas últimas son herramientas que facilitan la comunicación entre dos componentes de software a través de un conjunto de definiciones y protocolos.

En primer lugar, ChatGPT ofrece una API que puede integrarse en IDE para autocompletado, generación de código y explicación de fragmentos. En [42] se puede observar que el LLM de OpenAI cuenta con una extensión que permite una integración perfecta con Visual Studio Code [43]. Es útil para depuración, generación y refactorización de código.

En segundo lugar, Gemini ha sido integrado por Google en su ecosistema de desarrollo, incluyendo herramientas como Google Colab [44], de acuerdo con lo mostrado en [45]. Está también aceptado su uso en IDE mediante la API de Google AI Studio [46] u otras similares. Es útil en la generación de código, proporciona explicaciones y corrige errores.

En tercer lugar, como se indica en [47], Copilot es una de las integraciones más avanzadas en IDE, especialmente en Visual Studio Code gracias, en parte, a la integración nativa que aporta. Utiliza modelos de IA de Microsoft para autocompletar código, generar funciones completas y ofrecer sugerencias en tiempo real al programador. Destaca en lenguajes como Python, C y JavaScript, entre otros.

Por último, DeepSeek, pese a ser un modelo de enero de 2025, también ofrece capacidades de generación y asistencia en programación. Según se presenta en [48], su soporte en IDE todavía es limitado, pero ya existen ciertas extensiones que permiten ejecutar este LLM localmente en Visual Studio Code y obtener así asistencia inteligente de forma completamente gratuita.

2.3.5 IAG en alumnado de cursos introductorios a la programación

Tanto en el artículo [55] como en [56], se ha empleado ChatGPT como modelo de lenguaje principal en el desarrollo de las respectivas investigaciones. Esto responde a la cercanía que muestra el estudiante hacia esta IAG, siendo una de las más conocidas en todo el mundo. Los estudiantes buscan una IAG versátil; que sea capaz de ayudar con redacción de textos, programación, generación de ideas y tutoría, entre otros. Asimismo, los resultados obtenidos del empleo de este tipo de tutor en las aulas fueron

positivos, mejorando facetas como: el pensamiento computacional de los alumnos, la eficacia en la programación y su propia motivación a la hora de afrontar nuevos retos.

En contraste, como se pone en manifiesto en [57], el empleo de Gemini no solo no ha mejorado el proceso de aprendizaje, si no que tampoco remite una clara motivación ni disfrute por parte del alumnado de la Universidad de Chile que participó en dicho estudio. Igualmente, no todos los alumnos muestran la misma curva de aprendizaje, de hecho, aquellos con conocimientos básicos previos a la intervención, se beneficiaron más que aquellos alumnos que partían de cero, tal y como se recoge en [58].

Otro tipo de proyectos innovadores versan sobre la posibilidad de establecer la IAG como punto neurálgico de la programación escolar, es decir, basar la adquisición de las competencias, tanto básicas como específicas, en el uso de un LLM. El artículo [59] lo llevó a cabo. En lugar de explicar durante semanas la sintaxis de Python, le permitieron al alumno indagar con GitHub Copilot en el proceso de aprendizaje controlado. Para ello identificaron la necesidad de una buena bibliografía de apoyo que permitiera al alumno aprender a programar desde cero de la mano de la IAG [60]. Pese a ser un proyecto ambicioso e innovador, no se obtuvieron diferencias notables con respecto al rendimiento de años anteriores en los que no se habían utilizado los LLM.

Pese a la diversidad de opiniones existentes con respecto al empleo de la IAG como apoyo al aprendizaje de la programación, todas ellas comparten una serie de conclusiones. El empleo de la IAG en ningún momento empeora los resultados obtenidos en previas convocatorias, sin embargo, puede ser interesante como una competencia extra en la formación. Es relevante mencionar que, para obtener un máximo rendimiento del empleo de un LLM, el alumno debería de tener unos conocimientos básicos adquiridos. De esta forma será capaz de discernir cuándo un código necesita corrección. En definitiva, la importancia de la supervisión humana es muy elevada, especialmente en códigos complejos que demandan altos estándares de calidad como indica [55].

2.3.6 *Uso fraudulento de IAG*

La detección del uso de LLM en cualquier ámbito que abarque la generación de texto o código es cada vez más complejo. Pese a que los sistemas de detección avanzan rápidamente y se actualizan con las últimas novedades para prevenir el plagio, es muy difícil encontrar métricas sobre las cuales se pueda asegurar que la solución al problema planteado haya sido generada por un LLM.

Es importante destacar que el texto que crea la Inteligencia Artificial Generativa es completamente original. Por esta razón, está libre de plagio y no existen herramientas específicas para su detección, según se presenta en [61].

Del mismo modo, los detectores de plagio, cuya versión gratuita se encuentra accesible para todo el mundo en la web, tienen una probabilidad de acierto del 50%, según [62]. Esto significa que en la mitad de las ocasiones no encuentran una métrica fiable que permita asegurar la creación del código por un LLM. Por consiguiente, la probabilidad del 50% los convierte en programas que no presentan seguridad.

Sin embargo, en [63] se plantea la existencia de una nueva herramienta, que aún se encuentra en proceso de desarrollo, pero que permitirá detectar el plagio. Cabe señalar que esta no constituye la única iniciativa en este sentido, sino que representa un ejemplo de las soluciones que están emergiendo en este ámbito. Se conoce por el nombre de zyBook's APEX y basará su funcionamiento en detectar alguna de las siguientes situaciones en el código: los LLM, y en particular ChatGPT, utilizan estilos de programación que difieren de los impartidos en el aula, además, las respuestas generadas por él, aunque diferentes, todas siguen un mismo patrón repetitivo entre alumnos y, por último, un alumno que utiliza un LLM, tendrá un estilo de programación diferente para cada tarea, lo cual no es lógico.

Debido a su dificultad de detección, se plantea la necesidad de establecer líneas éticas claramente definidas. Para ello se sugiere un enfoque equilibrado. Como se indica en [64], la combinación de la supervisión humana con las capacidades de la IAG es fundamental para mitigar los riesgos y mejorar la fiabilidad.

2.3.7 Retos, limitaciones y riesgos de la IAG

La IAG ciertamente podría ser considerada como una oportunidad ante los retos que plantea la actualidad; no obstante, como toda novedad que se implementa de forma gradual, no se pueden obviar las limitaciones y los riesgos inherentes a su utilización. En relación con esta premisa, la Comisión Europea en [65] señala que la forma en que se aborde la IAG definirá el mundo del futuro. De ahí, la presencia de un claro reto: el uso ético y responsable de la IAG.

Como indica [66], el investigador podrá desarrollar su búsqueda de información de manera oportuna y eficiente, debido a que habrá un claro respeto por la propiedad intelectual y primará la transparencia. En consecuencia, como se ve en [67], es de suma importancia la creación de directrices éticas con el fin de guiar la toma de decisiones sobre IAG y, a su vez, mitigar los presentes desafíos. A continuación, se muestran una serie de limitaciones y riesgos en relación con este aspecto.

En primer lugar, merece una detenida consideración la calidad de los datos empleados. Como se indica en [68], es fundamental implementar la utilización de fuentes revisadas por pares, en comparación con otras fuentes como Wikipedia. Para ello, según [69], se ve necesario invertir en fuentes de datos diversificadas para minimizar los sesgos inherentes presentes en los datos de entrenamiento. Un ejemplo ilustrativo es el estudio [70] que representa cómo un algoritmo ampliamente utilizado, que basa su funcionamiento en el establecimiento de medidas de precisión predictiva, presentó grandes sesgos raciales.

En segundo lugar, garantizar el almacenamiento de datos de forma segura y perdurable se trata de otro aspecto clave a atender. La Ley Orgánica de Protección de Datos (LOPD) en España, el Reglamento General de Protección de Datos (RGPD) en Europa o la Ley de protección de privacidad del consumidor (CCPA) en Estados Unidos, son claros ejemplos de protocolos y reglamentos que abogan por una protección y uso adecuado de datos. En relación con la IAG a nivel nacional destaca la creación de la Agencia Española de Supervisión de Inteligencia Artificial (AESIA) en 2023 [71]. Esta agencia tiene como objetivo supervisar y asesorar sobre el uso adecuado de los sistemas de IA, minimizando riesgos y garantizando la protección de derechos fundamentales como la integridad, la intimidad y la igualdad de trato.

A su vez, explorando el artículo [72], se ve la preocupación existente hoy en día sobre la sostenibilidad ambiental. Los modelos se vuelven cada vez más complejos y se requieren, en consecuencia, más recursos computacionales. Como se ve en [73], la demanda energética para entrenar dichos modelos es abrumadora y se debe tender a buscar alternativas más sostenibles (utilización de fuentes de energía renovable o desarrollo de algoritmos más eficientes) para poder afrontar retos referidos a la capacidad técnica y a la integración de la IAG en diversas industrias.

En definitiva, las limitaciones y riesgos presentes en la IAG necesitan ser evaluados responsablemente con el fin de definir directrices éticas de actuación. Como se indica en [74], conviene, por ende, establecer fechas de revisión por equipos multidisciplinarios asignados que valoren periódicamente la exactitud, relevancia y aplicabilidad de los datos, teniendo siempre en cuenta los avances tecnológicos y las investigaciones recientes. Y, para mayor aporte de rigurosidad, se ha de dejar constancia de un historial de cambios y actualizaciones, asegurando así una continuidad en los procesos de mejora. De la misma forma, [75] asegura que, adoptando un enfoque integral y multifacético para abordar el sesgo en el desarrollo de modelos de IA, los investigadores y los profesionales pueden ayudar a garantizar que estas tecnologías se utilicen de forma ética.

3 DESARROLLO DEL TFG

Al igual que el proceso de aprendizaje del estudiantado en la asignatura “Informática para la Ingeniería” comienza asentando el proceso de diseño de programas mediante el uso de diagramas de flujo, este trabajo también se inició con una fase preliminar centrada en la evaluación de la capacidad de la Inteligencia Artificial Generativa para resolver ejercicios basados en dichos diagramas. Estas pruebas iniciales permitieron observar cómo las IAG interpretan y estructuran la lógica algorítmica antes de proceder a la escritura de código, replicando así el enfoque pedagógico seguido en el aula. Esta primera aproximación sirvió de punto de partida para valorar su comprensión de los fundamentos de la programación.

A partir de esta base, se llevó a cabo un análisis comparativo de la capacidad de generación de código por parte de cuatro Inteligencias Artificiales de última generación. Para tal fin, se tomaron como referencia los proyectos de programación propuestos en la asignatura anteriormente mencionada. Estos proyectos, diseñados para evaluar las habilidades de programación en C de los estudiantes, sirvieron como banco de pruebas para las IAG seleccionadas. Se contrastaron los resultados obtenidos por cada IAG en su versión gratuita y en su versión de pago, con el objetivo de evaluar el impacto de las funcionalidades avanzadas y el acceso a mayores recursos computacionales en la calidad y eficiencia del código generado. Esta comparativa permitió obtener una visión clara del estado actual de la IAG en el ámbito de la programación, así como de las ventajas y limitaciones de las diferentes versiones de estas herramientas.

Posteriormente, se exploró el uso de la IAG como herramienta de apoyo al profesorado en la evaluación de proyectos de programación. Para ello, se le proporcionaron los proyectos de los estudiantes, con el objetivo de valorar su capacidad para emitir calificaciones coherentes, detectar errores y aplicar criterios de forma automatizada.

Finalmente, se desarrolló un modelo GPT personalizado, diseñado específicamente para actuar como recurso de apoyo tanto al alumnado como al profesorado. Este modelo fue entrenado y ajustado para interpretar criterios académicos, evaluar proyectos según rúbricas específicas y ofrecer retroalimentación didáctica. Su análisis permitió valorar el potencial de los modelos personalizados no solo como asistentes en la corrección, sino también como guías interactivas para el aprendizaje autónomo del estudiante. Esta doble funcionalidad refleja el papel cada vez más versátil que pueden desempeñar las IAG en contextos educativos reales.

3.1 Análisis de diagramas de flujo

Como paso previo al estudio de la generación de código, se llevó a cabo una fase de análisis basada en la resolución de ejercicios con diagramas de flujo, replicando el itinerario formativo que siguen los estudiantes al comenzar la asignatura Informática para la Ingeniería. Para ello, se propusieron cuatro

ejercicios prácticos con distintos niveles de complejidad, enfocados a evaluar la capacidad de comprensión algorítmica de siete modelos de Inteligencia Artificial Generativa:

3.1.1 Análisis de diagramas de flujo con las versiones de IAG gratuitas

La primera parte de este estudio fue el análisis de los diagramas de flujo presentados a cuatro IAG, en sus versiones gratuitas: ChatGPT-4, Google Gemini 2.0 Flash, Microsoft Copilot y DeepSeek-R1.

En los primeros dos ejercicios, centrados en la interpretación de diagramas con estructuras de control (bucles, condicionales y operaciones aritméticas), se detectaron diferencias notables entre las IAG. Un ejemplo de estos dos ejercicios se muestra en la Figura 3-1. En ellos se realizan preguntas sencillas sobre el funcionamiento de los diagramas de flujo y se solicita a la IAG ayuda para resolverlas.

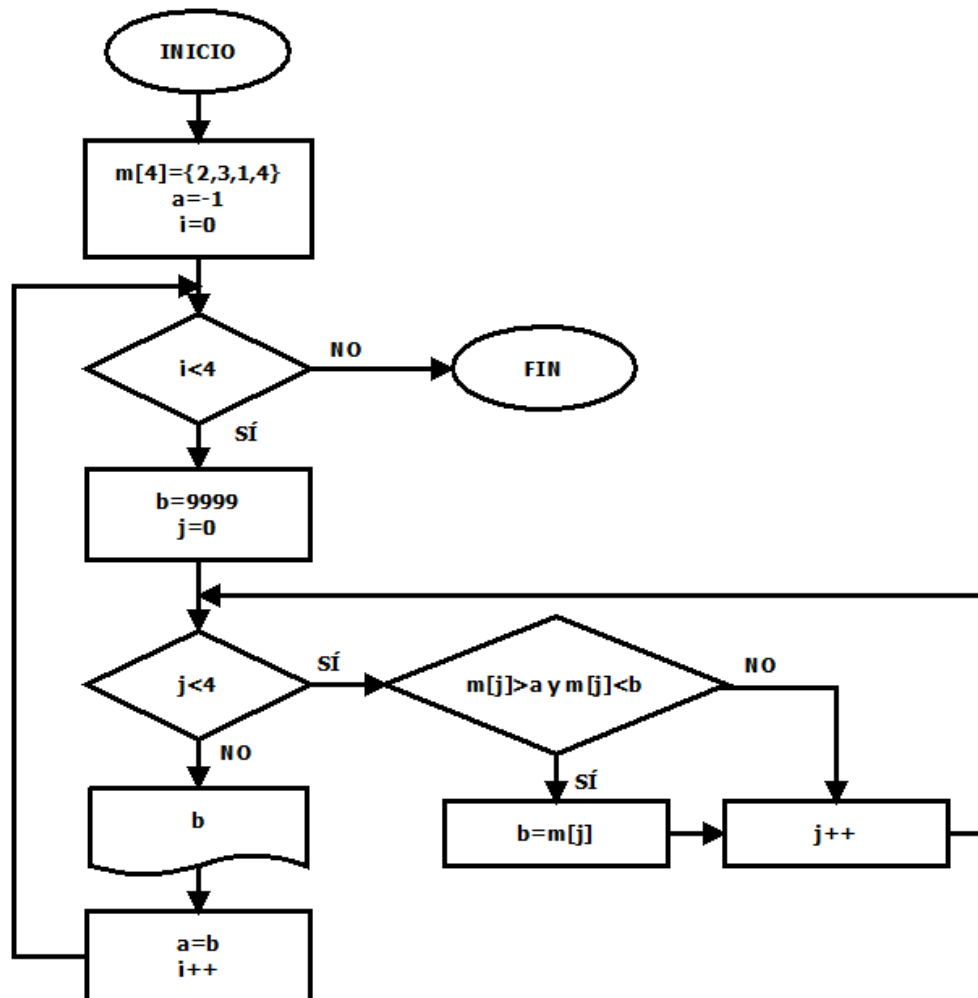


Figura 3-1 Ejercicio 1 de interpretación por parte de las IAG

Tal y como muestra la Figura 3-2, Microsoft Copilot y ChatGPT fueron las herramientas más precisas a la hora de explicar el funcionamiento general del diagrama, identificar errores lógicos y proporcionar interpretaciones coherentes. Sin embargo, ninguno de estos modelos fue capaz de ofrecer una respuesta completamente correcta en todos los apartados, especialmente cuando se requería reproducir paso a paso el comportamiento del diagrama con entradas específicas. En este contexto, DeepSeek destacó positivamente por ser el único modelo que logró describir correctamente todas las iteraciones de un diagrama de flujo complejo cuando se solicitó el análisis detallado. Por el contrario, Google Gemini no demostró solvencia en ninguno de los ejercicios propuestos.

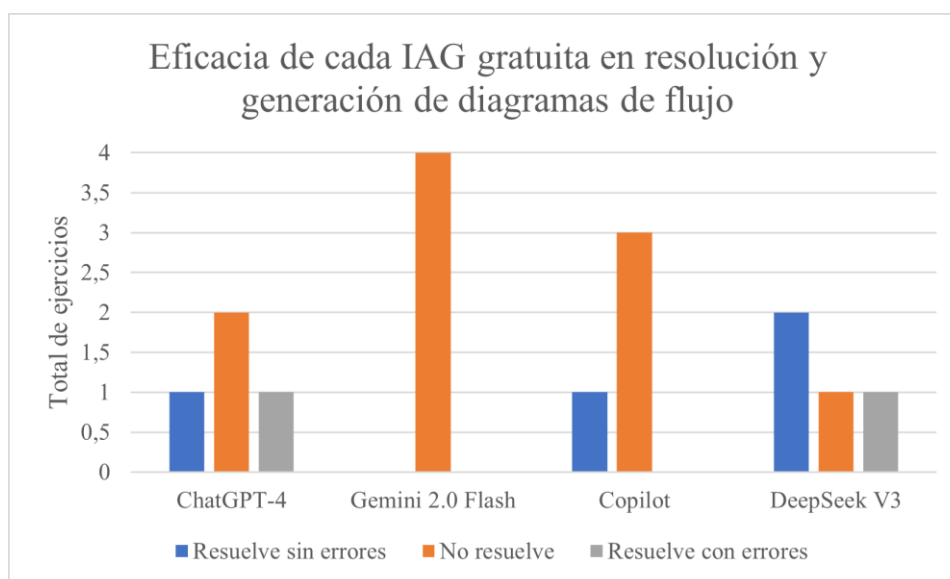


Figura 3-2 Desempeño de cada IAG gratuita en tareas de comprensión y generación de diagramas de flujo

En los últimos dos ejercicios, se introdujeron nuevas pruebas orientadas a evaluar la capacidad de las IAG para generar diagramas de flujo que el alumno pueda resolver y recibir *feedback* por parte de la IAG. En estos casos, ninguna herramienta fue capaz de generar diagramas acordes a lo que se les pedía. Una evaluación más extensa de los resultados se encuentra en el apartado 4.1.1.

3.1.2 Análisis de diagramas de flujo con las versiones de IAG de pago

En la Figura 3-3 se muestra la eficacia comparada de tres IAG, en sus versiones de pago, en la resolución de cuatro ejercicios relacionados con diagramas de flujo. Los dos primeros ejercicios se centraban en la comprensión de diagramas existentes, mientras que los dos últimos requerían la generación de diagramas a partir de enunciados breves.

Los resultados revelan una diferencia notable en el rendimiento de las IAG evaluadas. ChatGPT-o3-mini-high resolvió correctamente los cuatro ejercicios sin cometer errores, demostrando una capacidad sólida tanto para interpretar como para generar diagramas de flujo. Este comportamiento sugiere un alto nivel de fiabilidad en tareas que combinan análisis y producción, incluso cuando se requiere traducir instrucciones textuales a una representación gráfica.

Por su parte, Gemini Advanced mostró un rendimiento más limitado: resolvió correctamente los ejercicios de comprensión, pero tuvo dificultades con los ejercicios de generación. Uno de ellos no logró resolverlo, y en el otro cometió errores en la construcción del diagrama, lo que refleja una menor capacidad para transformar enunciados breves en representaciones visuales coherentes.

En el caso de Copilot Pro, el desempeño fue intermedio: resolvió correctamente tres ejercicios y cometió un error en una de las tareas de generación. Esto indica que, si bien su rendimiento es sólido, puede presentar fallos puntuales en tareas que exigen mayor capacidad de comprensión. La evaluación detallada de los resultados se encuentra en el apartado 4.1.2.

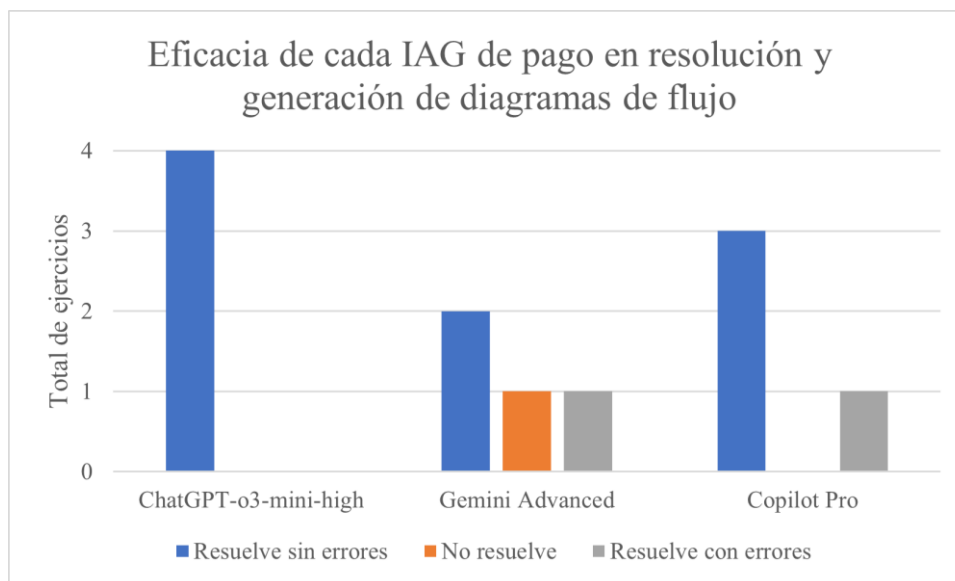


Figura 3-3 Desempeño de cada IAG de pago en tareas de comprensión y generación de diagramas de flujo

3.2 Análisis de generación de código simple

A continuación, se evaluaron estas mismas IAG con seis nuevos ejercicios orientados a la resolución de problemas clásicos mediante la generación de pequeños fragmentos de código. Un ejemplo de estos problemas es el ejercicio 1, donde se solicita a la IAG que escriba un programa en C para sumar la triangular superior de una matriz 4x4, obteniéndose códigos similares a la Figura 3-4.

```
#include <stdio.h>
#include <stdlib.h>

int main() {
    int A[4][4] = {
        {-3, 8, -5, 10},
        {2, -7, 2, 2},
        {-4, 6, 11, -3},
        {3, -8, -6, 3}
    };
    int i, j, suma = 0;

    // Sumar los elementos de la triangular superior
    for(i = 0; i < 4; i++) {
        for(j = i; j < 4; j++) {
            suma += A[i][j];
        }
    }

    printf("La suma de los elementos de la triangular superior es %d\n", suma);
    return 0;
}
```

Figura 3-4 Resolución de Microsoft Copilot al ejercicio 1 de generación de código

3.2.1 Análisis de generación de código simple con las versiones gratuitas de las IAG

Las cuatro IAG lograron proponer soluciones razonables. No obstante, ChatGPT, Copilot y DeepSeek demostraron una mejor comprensión de los conceptos fundamentales, mientras que Gemini continuó presentando dificultades a la hora de identificar correctamente la estructura lógica y los objetivos del enunciado, como se ve en la Figura 3-5, donde se representan los seis ejercicios resueltos por las IAG. Además, se valoró la capacidad de cada modelo de transformar el diagrama en código y de

sugerir mejoras o detectar errores ocultos, aspecto en el que Copilot ofreció un entorno visual más intuitivo y explicaciones mejor estructuradas.

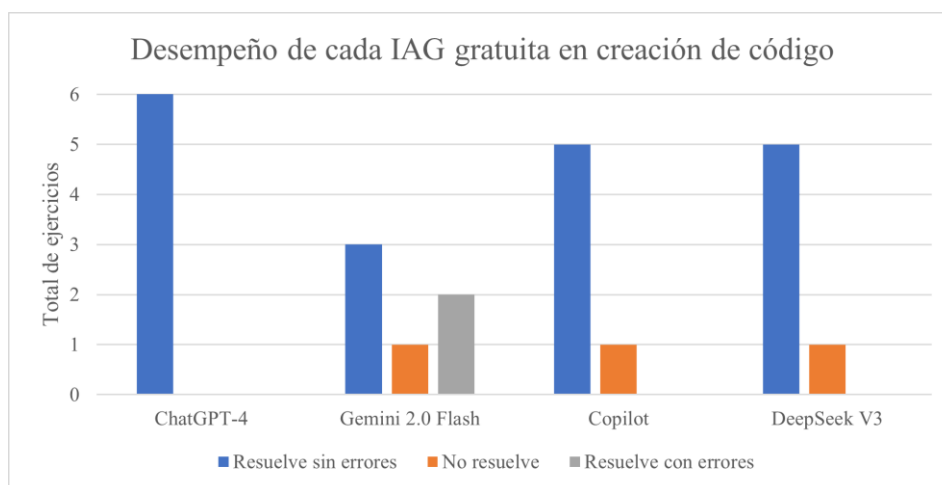


Figura 3-5 Desempeño de cada IAG en su versión gratuita en tareas de creación de código simple

En conclusión, esta primera fase de evaluación revela que las IAG, en sus versiones gratuitas, son capaces de abordar con éxito tareas de desarrollo de código, especialmente cuando el enunciado está bien definido y estructurado. No obstante, persisten limitaciones, especialmente en cuanto a interpretación contextual y análisis detallado de la ejecución paso a paso. Por ello, resulta fundamental que el profesorado mantenga un rol activo de supervisión y validación de las respuestas generadas, utilizando la IAG como una herramienta complementaria para reforzar el aprendizaje. Esta experiencia inicial marca un punto de partida sólido para el análisis posterior de generación de código. La evaluación completa de los resultados se encuentra en el apartado 4.1.3.

3.2.2 Análisis de generación de código simple con las versiones de pago de las IAG

La Figura 3-6 corresponde a la última fase de la evaluación inicial, centrada en la capacidad de las IAG de pago para resolver ejercicios sencillos de creación de código. Esta etapa permite valorar la competencia práctica de cada herramienta en la escritura de código funcional, lo que resulta clave para su aplicación en contextos educativos relacionados con la programación.

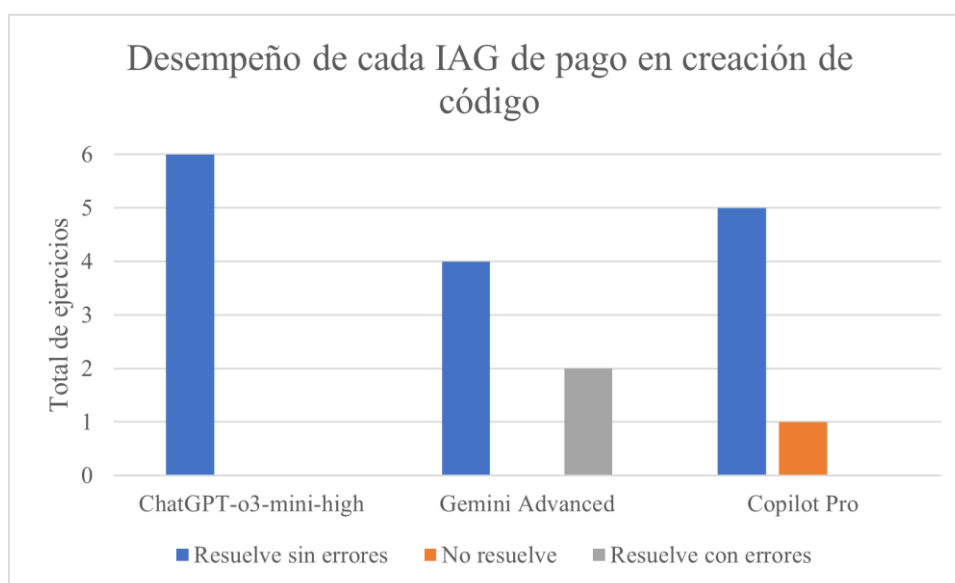


Figura 3-6 Desempeño de cada IAG en su versión de pago en tareas de creación de código simple

El modelo ChatGPT-o3-mini-high obtuvo el mejor resultado, resolviendo correctamente los seis ejercicios sin errores, lo que indica una alta precisión y consistencia en tareas básicas de programación.

Copilot Pro también mostró un buen rendimiento, con cinco ejercicios resueltos sin errores. Sin embargo, falló en uno de ellos, en el cual generó un código no compilable. Esto sugiere una leve limitación y refuerza la necesidad de supervisión por parte del usuario, ya sea el alumnado o el docente.

Por su parte, Gemini Advanced resolvió correctamente cuatro ejercicios, pero cometió errores en dos, lo que revela una mayor inestabilidad en la generación de código, incluso en tareas sencillas.

En resumen, aunque las tres IAG son capaces de generar código, la precisión varía significativamente, destacando especialmente ChatGPT-o3-mini-high como la opción más fiable en este contexto. La evaluación detallada de los resultados se encuentra en el apartado 4.1.4.

3.3 Evaluación de tres proyectos de la asignatura Informática para la Ingeniería

Finalizado el análisis preliminar basado en ejercicios sencillos de generación y comprensión de código, se procedió a la siguiente etapa del estudio. En esta fase, se expusieron distintos modelos de Inteligencia Artificial Generativa (tanto en sus versiones gratuitas como de pago) a los enunciados de los proyectos de programación correspondientes a la asignatura Informática para la Ingeniería. Dichos proyectos tienen como objetivo evaluar la competencia del estudiantado para desarrollar un proyecto de programación integral en lenguaje C, que combine y aplique los conocimientos adquiridos a lo largo del curso. Cabe destacar que esta actividad representa un 40 % de la calificación final de la asignatura, lo que le confiere un peso muy significativo en el rendimiento académico del alumnado. Todos los enunciados de los proyectos se suministraron en formato PDF a la IAG. Al final del apartado se muestra un resumen de los resultados globales en la Tabla 3-4.

3.3.1 Generación de código en el proyecto de Informática para la Ingeniería del curso 2019-2020

El proyecto [76] consiste en crear una versión en modo texto para un jugador del juego Conecta 4, tal y como muestra la Figura 3-7. La dinámica es sencilla: cada jugador cuenta con 21 fichas y el tablero tiene 42 posiciones posibles a rellenar. Además, cada jugador podrá introducir la ficha en la columna que quiera (excepto que ya esté completa) y esta caerá a la posición más baja posible.

Conecta 4 es un popular juego de mesa (de Hasbro) ideado para dos jugadores. El juego consiste en que cada jugador, para ganar, debe conseguir alinear cuatro fichas consecutivas del mismo color, introduciendo por turnos y en un tablero vertical (formado por seis filas y siete columnas) las fichas que le corresponden.



Figura 1: Conecta 4 - Edición de Hasbro

Cada jugador cuenta con 21 fichas (42 posiciones diferentes en el tablero). Por turnos, los jugadores introducen por la parte superior de una columna (siempre y cuando la misma no esté ya completa) la ficha, y esta caerá a la posición más baja libre. Gana el primer jugador que consiga alinear cuatro fichas de forma consecutiva en el tablero (puede ser de forma horizontal, vertical o diagonal). Los jugadores empatan si todas las posiciones están completas y ningún jugador ha conseguido alinear cuatro fichas.

Figura 3-7 Portada del proyecto Conecta 4

El programa debe permitir al usuario iniciar una nueva partida, cargar una partida guardada, guardar la partida actual o salir. El usuario juega contra el ordenador, y el juego continúa hasta que un jugador consigue alinear cuatro fichas o el tablero se llena. La evaluación de las soluciones proporcionadas por las diferentes IAG se ven en la Tabla 3-1.

VERSIÓN		EVALUACIÓN
ChatGPT-4o mini	Ventajas	• Ausencia de errores de compilación. • Código legible y claro.
	Inconvenientes	• No se genera la ilusión gráfica de actualización del movimiento de las fichas borrando la pantalla por cada movimiento.
ChatGPT o3-mini-high	Ventajas	• Interfaz muy clara e intuitiva. • Código legible y claro. • Emplea funciones de manera correcta. • Introduce mejoras consideradas como optativas.
	Inconvenientes	• Se identifica una advertencia menor por la declaración de una función no utilizada.
DeepSeek-V3	Ventajas	• Ausencia de errores de compilación. • Código legible y claro.
	Inconvenientes	• No se evidencian errores.
Gemini 2.0 Flash	Ventajas	• Ausencia de errores de compilación. • Código legible y claro.

	Inconvenientes	• No interpreta archivos en formato PDF. Se introdujo como entrada a la IAG el enunciado del proyecto en LaTeX. • Impone restricciones en la longitud de las respuestas generadas. Se fragmentó la solicitud del código en varias partes y posteriormente se unió de forma manual. • No se genera la ilusión gráfica de actualización del movimiento de las fichas borrando la pantalla por cada movimiento.
	Ventajas	• Incremento significativo del tamaño de la salida generada. • La representación gráfica del tablero presenta una mejora significativa con una apariencia visual mucho más conseguida.
	Gemini Advanced	Inconvenientes
		• Ausencia de un menú de inicio del juego. • No se genera la ilusión gráfica de actualización del movimiento de las fichas borrando la pantalla por cada movimiento. • No es capaz de terminar la partida cuando el jugador o la máquina gana. • La única forma de acabar la partida es forzando la finalización del programa.
	Ventajas	• Interfaz inicial muy clara y visualmente atractiva.
	Copilot	Inconvenientes
	Copilot	Inconvenientes
		• Nunca se acaba la partida: se sigue mostrando el panel de juego, aunque algún jugador gane o se complete el tablero.
	Ventajas	• Ausencia de errores de compilación. • Código legible y claro. • Interfaz inicial muy clara y visualmente atractiva.
	Copilot Pro	Inconvenientes
		• Inicialmente no generaba código debido a su adherencia a protocolos éticos. Esto se debe a que, en el enunciado del proyecto, específicamente en la sección <i>Otras consideraciones</i>, se indica que se penalizaría la copia. • Finalmente, introduciendo un archivo en formato PDF sin esas especificaciones y refinando el <i>prompt</i> en varias ocasiones, se llegó a un código completo.

Tabla 3-1 Evaluación de las soluciones al proyecto de Informática para la Ingeniería del curso 2019-2020

3.3.3.1 Evaluación del funcionamiento de ChatGPT

Al realizar una comparativa entre la versión gratuita y la versión de pago destinada a programación y lógica (o3-mini-high) de ChatGPT, se puede concluir que la versión de pago ofrece una experiencia significativamente superior en varios aspectos clave del desarrollo de software.

En primer lugar, la versión de pago destaca por la calidad de su implementación. No solo genera código funcional, sino que lo hace con un alto grado de robustez y claridad. La interfaz de usuario

resultante es intuitiva y visualmente atractiva, lo que facilita la interacción y comprensión del programa, como se ve en la Figura 3-8.

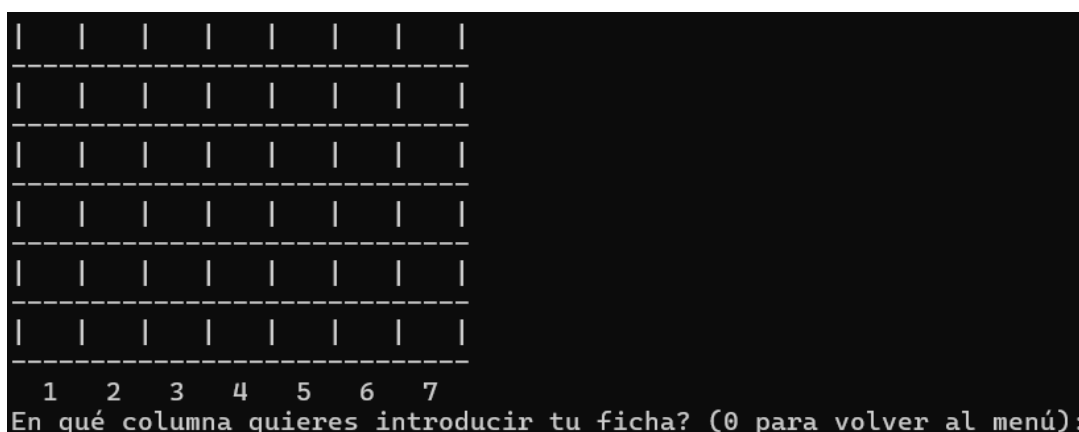


Figura 3-8 Interfaz generada por ChatGPT o3-mini-high

La generación de un código legible y sin errores de compilación contribuye de manera significativa a la facilidad de modificación y a la posibilidad de ampliar el proyecto en el futuro, lo que refleja la fiabilidad del código generado.

En segundo lugar, la versión de pago enriquece la experiencia del usuario a través de la incorporación de funcionalidades adicionales. La inclusión de un menú inicial y la capacidad de guardar y cargar partidas son ejemplos claros de cómo esta versión va más allá de la mera funcionalidad para ofrecer una experiencia más completa y satisfactoria. Si bien es cierto que se ha identificado una advertencia de compilación menor, esta no compromete la ejecución del programa, y se considera un inconveniente menor en comparación con las ventajas ofrecidas.

En contraste, la versión gratuita, aunque valiosa en términos de optimización y eficiencia, se queda corta en la riqueza de la experiencia de usuario. Su enfoque en la ausencia de errores de compilación y la fluidez de la interfaz, si bien importantes, no compensan la falta de funcionalidades adicionales y la menor claridad en la presentación.

En definitiva, la versión de pago (o3-mini-high) se posiciona como una herramienta más completa y valiosa para el desarrollo de software. Su capacidad para generar código robusto, con una interfaz de usuario clara y funcionalidades adicionales, la convierte en la opción preferible para aquellos que buscan una experiencia de desarrollo superior.

3.3.1.2 Evaluación del funcionamiento de DeepSeek

DeepSeek V3 se presenta como un LLM de notable eficiencia. La solución que genera carece de errores durante la compilación y se trata de un código claro y legible. Además, es capaz de generar una interfaz atractiva como muestra la Figura 3-9.

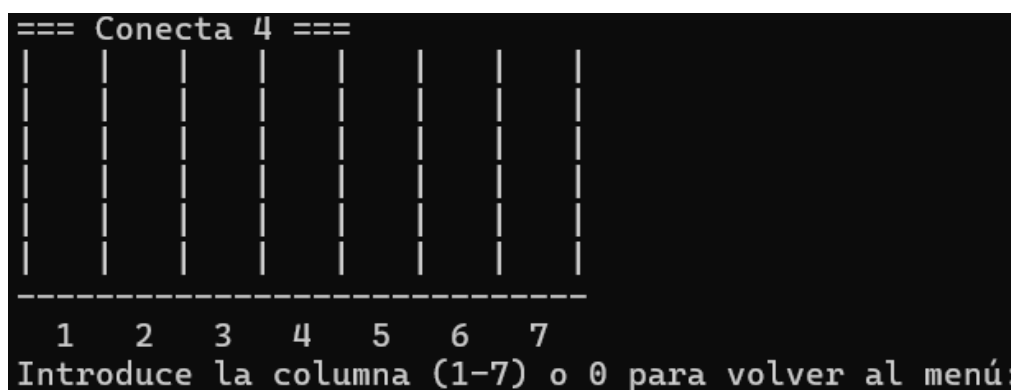


Figura 3-9 Interfaz generada por DeepSeek

3.3.1.3 Evaluación del funcionamiento de Google Gemini

El panorama comparativo entre la versión gratuita (Gemini 2.0 Flash) y la versión de pago (Gemini Advanced) de este modelo de lenguaje revela contrastes significativos, especialmente en lo que respecta a la estabilidad y funcionalidad. La versión gratuita se distingue por una implementación de software robusta, caracterizada por la ausencia de errores de compilación y la claridad del código. No obstante, esta versión no está exenta de limitaciones, como la incapacidad para interpretar archivos en formato PDF y las restricciones en la longitud de las respuestas generadas, que pueden obstaculizar ciertos flujos de trabajo.

En contraposición, Gemini Advanced promete funcionalidades mejoradas, como la interfaz de entrada compatible con el formato de documento PDF y la capacidad de generar respuestas de longitud significativamente aumentada. Además, se destaca una mejora significativa en la claridad visual de la interfaz gráfica. Sin embargo, estas ventajas se ven ensombrecidas por deficiencias funcionales críticas que comprometen la experiencia del usuario y la fiabilidad del sistema. Específicamente, la ausencia de un menú de inicio, problemas con la actualización de la visualización y fallos en la lógica de finalización de la partida, plantean serias dudas sobre la calidad y estabilidad de esta versión de pago, como muestra la Figura 3-10.

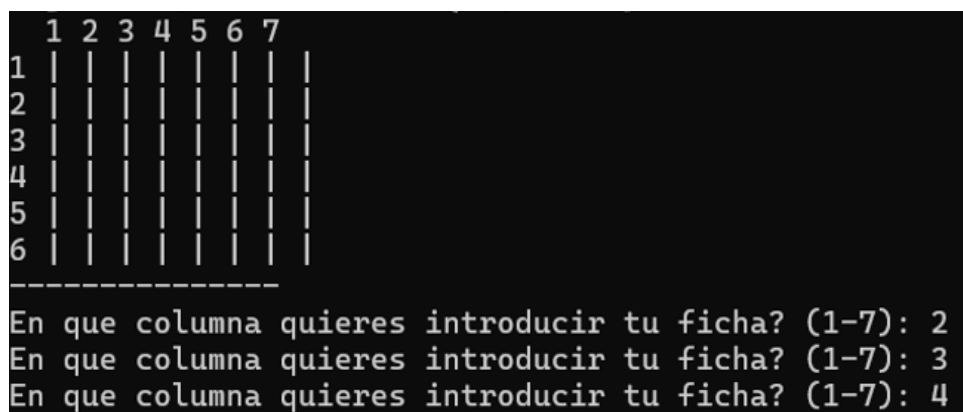


Figura 3-10 Anomalías en el funcionamiento del programa generado por Gemini Advanced

En resumen, mientras que la versión gratuita ofrece una experiencia de usuario estable y eficiente dentro de sus limitaciones, Gemini Advanced adolece de problemas fundamentales que socavan su propuesta de valor. Los fallos en la lógica del programa, la inconsistencia en la representación visual y la incapacidad para gestionar correctamente los estados del juego indican que la versión de pago aún no ha alcanzado un nivel de madurez adecuado. Por lo tanto, en su estado actual, la versión gratuita se presenta como una opción más fiable y consistente, mientras que la versión de pago no justifica el coste adicional para los usuarios que buscan estabilidad y funcionalidad.

3.3.1.4 Evaluación del funcionamiento de Microsoft Copilot

El análisis comparativo entre Copilot y Copilot Pro revela diferencias significativas en la funcionalidad y estabilidad de ambas versiones. Copilot, en su versión gratuita, se distingue por una interfaz de usuario inicial optimizada, caracterizada por su claridad y estética visualmente atractiva. Sin embargo, esta versión presenta una anomalía crítica en la lógica de finalización de la partida, donde el estado del juego persiste independientemente de las condiciones de victoria, requiriendo una terminación forzada del proceso. Como se observa en la Figura 3-11, pese a que el jugador ya había conseguido reunir cuatro fichas para ganar la partida, el juego no acaba. Esta deficiencia funcional impacta negativamente en la experiencia del usuario y la fiabilidad del software.

```

 1 2 3 4 5 6 7
|X|X|0|X| | | |
|0|X|0|0| | | |
|X|X|0|0|X| | |
|X|0|X|0|0| | |
|X|X|0|X|X|0|0|
|X|0|X|X|0|0|0|

Iago, en que columna quieres introducir tu ficha? (1-7):

```

Figura 3-11 Deficiencia funcional del programa generado por Copilot

Por otro lado, Copilot Pro, la versión de pago, muestra una mejora sustancial en la calidad de la implementación del software, evidenciada por la ausencia de errores de compilación y la claridad del código. Al igual que la versión gratuita, Copilot Pro mantiene una interfaz de usuario optimizada, como se ve en la Figura 3-12, lo que sugiere un enfoque consistente en la experiencia del usuario. No obstante, esta versión presenta limitaciones iniciales en la generación del código completo, posiblemente relacionadas con protocolos éticos y cláusulas de originalidad en las especificaciones del proyecto. Aunque estas restricciones se superaron mediante varias iteraciones y refinamiento progresivo de las consultas, señalan una diferencia clave en el comportamiento de ambas versiones.

```

| . | . | . | . | . | . | . |
| . | . | . | . | . | . | . |
| . | . | . | . | . | . | . |
| . | . | . | . | . | . | . |
| 0 | . | 0 | . | . | . | . |
| X | X | X | X | 0 | . | . |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 |
¡Has ganado!

```

Figura 3-12 Interfaz del programa generado por Copilot Pro

En definitiva, Copilot Pro ofrece ventajas significativas en términos de calidad de implementación y estabilidad, a pesar de las restricciones iniciales en la generación de código. La versión gratuita, si bien cuenta con una interfaz de usuario bien diseñada, adolece de fallos críticos en la lógica del programa. La elección entre ambas versiones dependerá de las prioridades del usuario: Copilot Pro para estabilidad, calidad de código y funcionalidad de programación, o Copilot gratuito para tareas generales y redacción de texto, teniendo en cuenta sus limitaciones en la generación de código compilable y ejecutable.

3.3.2 Generación de código en el proyecto de Informática para la Ingeniería del curso 2020-2021

Este proyecto de programación [77] se centra en el desarrollo de una versión para ordenador del juego del ocho, también conocido como "Taken", en modo texto. El juego del ocho es un rompecabezas deslizante que consiste en ordenar una serie de fichas numeradas dentro de un tablero con nueve espacios, tal y como muestra la Figura 3-13.

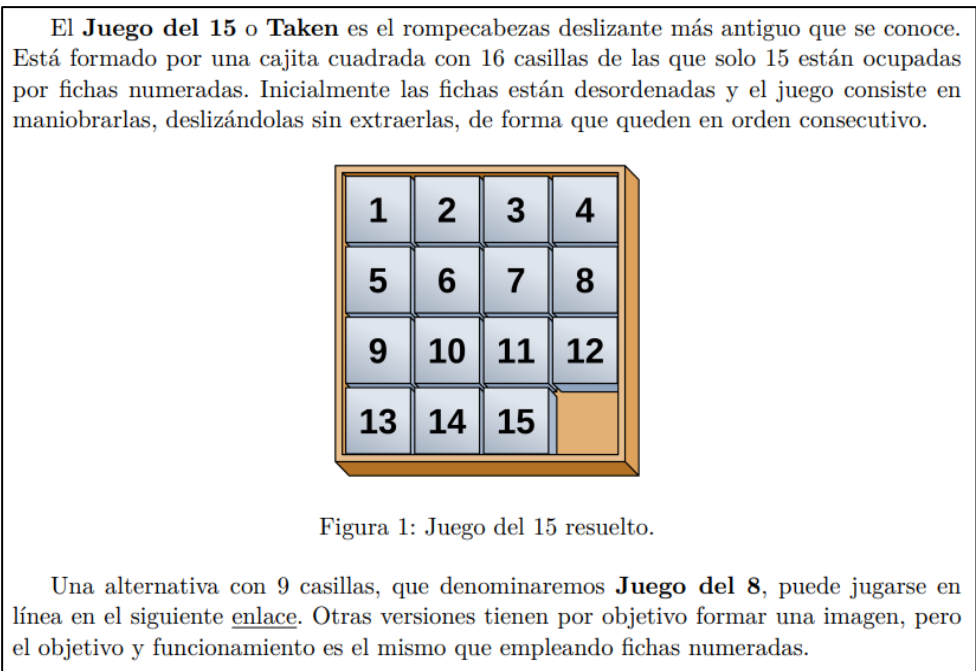


Figura 3-13 Portada del proyecto Juego del 8

Se requiere que los estudiantes implementen el juego utilizando conceptos de programación previamente explicados en la asignatura como estructuras de control, funciones, arrays y manipulación de archivos. El programa debe permitir al usuario interactuar con el juego mediante la introducción de comandos para mover las fichas, cargar y guardar partidas, y salir del juego.

Además de la funcionalidad básica, se anima a los estudiantes a incorporar mejoras al juego, como un menú inicial, la posibilidad de guardar el historial de puntuaciones y la implementación de niveles de dificultad. La evaluación de las soluciones generadas se muestra en la Tabla 3-2.

VERSIÓN		EVALUACIÓN
ChatGPT-4o mini	Ventajas	Se alcanza el nivel mínimo exigido en el enunciado del proyecto.
	Inconvenientes	No implementa ninguna funcionalidad extra como: la creación de un menú inicial o la solicitud de identificación del usuario al inicio de cada partida nueva.
ChatGPT o3-mini-high	Ventajas	- Interfaz muy clara e intuitiva. - Código legible y claro. - Emplea funciones de manera correcta. - Introduce todas las mejoras consideradas como optativas.
	Inconvenientes	No se evidencian errores.
DeepSeek-V3	Ventajas	- Ausencia de errores de compilación. - Código legible y claro.
	Inconvenientes	No se genera la ilusión gráfica de actualización del movimiento de las fichas borrando la pantalla por cada movimiento.

Gemini 2.0 Flash	Ventajas	• Ausencia de errores de compilación. • Código legible y claro. • Añade la mejora de la creación de un menú inicial.
	Inconvenientes	• No interpreta archivos en formato PDF. Se introdujo como entrada a la IAG el enunciado del proyecto en LaTeX. • Impone restricciones en la longitud de las respuestas generadas. Se fragmentó la solicitud del código en varias partes y posteriormente se unió de forma manual.
Gemini Advanced	Ventajas	• No se evidencia ninguna ventaja.
	Inconvenientes	• Imposibilidad de compilar el programa debido a varias anomalías producidas en la declaración de variables.
Copilot	Ventajas	• Ausencia de errores de compilación. • Código legible y claro.
	Inconvenientes	• En ciertos movimientos aleatorios, se descolocan las casillas del tablero.
Copilot Pro	Ventajas	• Ausencia de errores de compilación. • Código legible y claro.
	Inconvenientes	• En ciertos movimientos aleatorios, se descolocan las casillas del tablero.

Tabla 3-2 Evaluación de las soluciones al proyecto de Informática para la Ingeniería del curso 2020-2021

3.3.2.1 Evaluación del funcionamiento de ChatGPT

El análisis comparativo de ambas versiones revela diferencias sustanciales que impactan directamente en la experiencia del usuario y en la aplicabilidad de la herramienta. La versión gratuita de ChatGPT, si bien funcional para tareas básicas, presenta limitaciones inherentes a su diseño. Su capacidad de procesamiento y respuesta, aunque adecuada para consultas sencillas, se ve comprometida ante requerimientos más complejos. La omisión de funcionalidades avanzadas, tales como la personalización de la interfaz y la gestión de datos de usuario, hace que su potencial se vea mermado.

En contraste, la versión de pago (o3-mini-high) se distingue por su optimización y la inclusión de características que enriquecen significativamente la interacción. La interfaz gráfica, mostrada en la Figura 3-14, está diseñada con criterios de claridad y atractivo visual, facilitando la navegación y el acceso a las funcionalidades. Asimismo, la implementación de características adicionales, como el acceso prioritario a nuevas funciones, responde a las necesidades de usuarios que requieren un rendimiento superior y una experiencia más completa.

```

2 5
7 8 3
6 4 1
Movimientos: 0
(w) Arriba (s) Abajo (a) Izq (d) Der (c) Cargar (g) Guardar (x) Salir
Introduzca opción: |

```

Figura 3-14 Interfaz generada por ChatGPT o3-mini-high

3.3.2.2 Evaluación del funcionamiento de DeepSeek

DeepSeek es una herramienta gratuita que ofrece ventajas significativas para el desarrollo de este proyecto. Sin embargo, también presenta la desventaja de no borrar la pantalla por cada movimiento para generar la ilusión de actualización, lo que impacta la fluidez de la experiencia del usuario, como muestra la Figura 3-15. A pesar de esta limitación, la capacidad de DeepSeek para generar código de alta calidad lo convierte en una herramienta valiosa para este proyecto.

```

1 2
8 4 3
6 7 5
Movimientos: 0
(w) Arriba (s) Abajo (a) Izquierda (d) Derecha (c) Cargar (g) Guardar (x) Salir.
Introduce opcion: w

1 2
8 4 3
6 7 5
Movimientos: 1
(w) Arriba (s) Abajo (a) Izquierda (d) Derecha (c) Cargar (g) Guardar (x) Salir.
Introduce opcion: d

1 2
8 4 3
6 7 5
Movimientos: 2
(w) Arriba (s) Abajo (a) Izquierda (d) Derecha (c) Cargar (g) Guardar (x) Salir.
Introduce opcion: a

1 2
8 4 3
6 7 5
Movimientos: 3
(w) Arriba (s) Abajo (a) Izquierda (d) Derecha (c) Cargar (g) Guardar (x) Salir.
Introduce opcion: |

```

Figura 3-15 Ausencia de ilusión visual en el código generado por DeepSeek

3.3.2.3 Evaluación del funcionamiento de Google Gemini

La evaluación comparativa de Gemini 2.0 Flash (versión gratuita) y Gemini Advanced (versión de pago) en el contexto de este proyecto de programación revela diferencias notables en cuanto a la calidad del código generado y la capacidad de cumplir con las especificaciones del proyecto. Si bien ambas versiones comparten el objetivo de asistir en el desarrollo de software, su desempeño varía significativamente en áreas cruciales como la generación de código libre de errores o la claridad de este.

Gemini 2.0 Flash, a pesar de ser la versión gratuita, exhibe un alto grado de optimización en la implementación del software. Además, esta versión destaca por su capacidad de creación de código más avanzado, extendiéndose más allá de los requerimientos mínimos del proyecto e incluyendo la implementación de funcionalidades adicionales. Sin embargo, presenta limitaciones en el procesamiento de archivos PDF y en la longitud de las respuestas generadas, lo que puede requerir pasos adicionales como introducir los ficheros de entrada en formatos distintos y la posterior unión manual de las partes de código generadas por separado.

En contraste, Gemini Advanced, la versión de pago, no ha demostrado ventajas significativas en la evaluación del proyecto. De hecho, se han detectado anomalías sustanciales en la definición de variables en el código generado, lo que impide la compilación del programa, tal y como se ve en la Figura 3-16. Estas inconsistencias en la declaración de tipos de datos y la asignación de valores iniciales comprometen la funcionalidad del código y plantean serias dudas sobre la fiabilidad de esta versión de pago.

```

Proyecto20-21_Gemini_pago.c:95:24: warning: format '%s' expects argument of type 'char *', but argument 3 has type 'int' [-Wformat=]
  95 |     fprintf(archivo, "%s\n", nombre);
      |                        ^
      |                        |
      |                        int
      |
      | char *
      |
      | %d
Proyecto20-21_Gemini_pago.c: In function 'cargarPartida':
Proyecto20-21_Gemini_pago.c:114:23: warning: format '%s' expects argument of type 'char *', but argument 3 has type 'in' [-Wformat=]
  114 |     fscanf(archivo, "%s\n", nombre);
      |                      ^
      |                      |
      |                      int
      |
      | char *
Proyecto20-21_Gemini_pago.c: In function 'main':
Proyecto20-21_Gemini_pago.c:147:11: warning: passing argument 1 of 'fgets' makes pointer from integer without a cast [-Wint-conversion]
  147 |     fgets(nombre, 21, stdin);
      |           ^
      |           |
      |           char
In file included from Proyecto20-21_Gemini_pago.c:1:
/usr/include/stdio.h:654:38: note: expected 'char * restrict' but argument is of type 'char'
  654 |     extern char *fgets(char *__restrict __s, int __n, FILE *__restrict __stream)
Proyecto20-21_Gemini_pago.c:148:20: warning: passing argument 1 of 'strcspn' makes pointer from integer without a cast [-Wint-conversion]
  148 |     nombre[strcspn(nombre, "\n")] = 0; // Eliminamos el salto de línea
      |                    ^
      |                    |
      |                    char
In file included from Proyecto20-21_Gemini_pago.c:4:
/usr/include/string.h:293:36: note: expected 'const char *' but argument is of type 'char'
  293 |     extern size_t strcspn(const char *__s, const char *__reject)
Proyecto20-21_Gemini_pago.c:148:11: error: subscripted value is neither array nor pointer nor vector
  148 |     nombre[strcspn(nombre, "\n")] = 0; // Eliminamos el salto de línea
      |           ^
Proyecto20-21_Gemini_pago.c:215:35: warning: format '%s' expects argument of type 'char *', but argument 2 has type 'in' [-Wformat=]
  215 |     printf("Enhorabuena %s, has completado el juego en %d movimientos!\n", nombre, movimientos);
      |                                  ^
      |                                  |
      |                                  int
      |
      | char *

```

Figura 3-16 Errores de compilación de código generado por Gemini Advanced

En conclusión, a pesar de sus limitaciones en el procesamiento de archivos y la longitud de las respuestas, Gemini 2.0 Flash se posiciona como una herramienta más robusta y fiable para este proyecto de programación, gracias a la calidad del código generado y su capacidad para cumplir con las especificaciones del proyecto. Gemini Advanced, por otro lado, presenta deficiencias críticas que dificultan su uso efectivo en el desarrollo del proyecto, lo que plantea interrogantes sobre su valor en relación con su coste.

3.3.2.4 Evaluación del funcionamiento de Microsoft Copilot

La comparación de funcionalidades entre Copilot y Copilot Pro en el desarrollo de un proyecto de programación revela un panorama complejo donde la optimización del código y la precisión visual se entrelazan. Aunque ambas versiones muestran un buen nivel de optimización en su implementación, cada una tiene una desventaja particular que afecta la forma en que se muestra visualmente el juego.

En el caso de Copilot en su versión gratuita, la principal deficiencia radica en la desalineación de las columnas en la interfaz gráfica. Específicamente, los valores numéricos que representan las fichas del juego se posicionan de manera inconsistente, manifestándose en la intersección del espacio delimitado por dos columnas adyacentes, como se ve en la Figura 3-17. Esta anomalía visual podría afectar a la legibilidad y la precisión en la representación del juego, lo que podría impactar la experiencia del usuario.

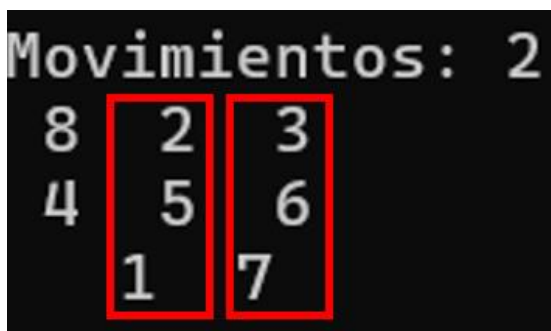


Figura 3-17 Anomalía en el tablero generado por el código creado por Copilot

Por otro lado, Copilot Pro (versión de pago), a pesar de su mayor coste, no está exento de limitaciones. En este caso, se observa una anomalía en la alineación visual de la celda que representa el hueco, según se observa en la Figura 3-18. Este comportamiento anómalo podría dificultar la identificación clara de la celda seleccionada por el usuario, lo que podría afectar la interacción fluida con el juego.

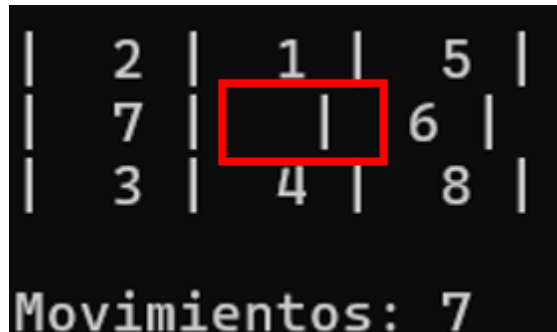


Figura 3-18 Anomalía en el tablero generado por el código creado por Copilot Pro

En resumen, la elección entre Copilot y Copilot Pro se convierte en un balance entre la optimización del código y la precisión visual. Si bien ambas versiones demuestran un buen rendimiento en la generación de código, las deficiencias específicas en la representación visual del juego deben ser consideradas al tomar una decisión. La versión gratuita presenta una desalineación en la representación de las columnas, mientras que la versión de pago muestra una anomalía en la alineación de la celda seleccionada. La decisión final dependerá de la importancia que se le dé a cada uno de estos aspectos en el contexto del proyecto de programación.

3.3.3 Generación de código en el proyecto de Informática para la Ingeniería del curso 2021-2022

El proyecto [78] requiere que los estudiantes desarrollen una aplicación en modo texto inspirada en unos de los juegos del popular drama de supervivencia "El Juego del Calamar", tal y como muestra la Figura 3-19. El juego simula una de las pruebas de la serie en la que los participantes deben cruzar un puente de cristal, con algunas placas que no soportan el peso y causan la caída del jugador. Los estudiantes deben demostrar habilidades en el empleo de programación estructurada, uso de funciones, estructuras selectivas y repetitivas, manejo de cadenas de caracteres, arrays y manipulación de ficheros.

El juego involucra a 8 participantes, incluyendo al jugador controlado por el usuario y 7 personajes no jugadores (NPCs) que juegan al azar. El objetivo es cruzar el puente con éxito, seleccionando las placas de vidrio templado correctas y aprendiendo de las trayectorias de los jugadores anteriores.

El programa debe permitir guardar y cargar partidas, mostrar la cola de participantes y el estado del puente, e indicar el número de muertos y salvados. Además, debe solicitar al jugador la dirección del salto en cada turno y mostrar mensajes indicativos sobre el progreso del juego. La evaluación de las soluciones generadas se muestra en la Tabla 3-3.

El **Juego del Calamar** es un drama de supervivencia surcoreana que ha logrado el debut más exitoso de Netflix hasta el momento, con más de 140 millones de espectadores en las cuatro primeras semanas. La serie narra la historia de un grupo de personas que se enfrentan mortalmente entre ellos en una sucesión de sádicos juegos infantiles. En una de estas pruebas, los participantes deben cruzar ordenadamente un puente formado por placas de cristal templado y placas de cristal convencional, aparentemente indistinguibles. Las primeras resisten el peso de una persona pero las segundas se rompen con el contacto provocando una caída al vacío y la muerte del jugador.



Figura 1: Puente de cristal en el Juego del Calamar.

Figura 3-19 Portada del proyecto Juego del Calamar

VERSIÓN		EVALUACIÓN
ChatGPT-4o mini	Ventajas	No se evidencia ninguna ventaja.
	Inconvenientes	- La pantalla se queda bloqueada en repetidas ocasiones, provocando que no se pueda jugar de forma fluida y haya que forzar la finalización del programa. - Desde un primer momento se ve cuáles son las casillas que se van a romper, por lo que el azar se pierde. - Por último, los NPC no se mueven solos. Es el propio jugador quien tiene que jugar por ellos.
ChatGPT o3-mini-high	Ventajas	- Interfaz muy clara e intuitiva. - Código legible y claro. - Los NPC juegan solos y, además, las casillas desaparecen de acuerdo con las instrucciones del enunciado. - El turno del jugador se determina de forma aleatoria, añadiendo incertidumbre al comienzo de la partida. - Al finalizar se muestra en pantalla cuántos jugadores se mantienen con vida.
	Inconvenientes	No se evidencian errores.
DeepSeek-V3	Ventajas	No se evidencia ninguna ventaja.

	Inconvenientes	• Existen varios errores de compilación que imposibilitan la ejecución del programa. • Una vez corregidos los errores, todos los NPC mueren al iniciar la partida, sin intervención del jugador. • Desde un primer momento se ve cuáles son las casillas que se van a romper, por lo que el azar se pierde.
	Ventajas	• No se evidencia ninguna ventaja.
Gemini 2.0 Flash	Inconvenientes	• No interpreta archivos en formato PDF. Se introdujo como entrada a la IAG el enunciado del proyecto en LaTeX. • La interfaz inicial es muy pobre. Dificulta el poder iniciar una nueva partida, cargarla o salir del juego. • Comienza con los NPC haciendo sus movimientos y no registra qué plataformas se rompen ni cuántos jugadores quedan vivos.
	Ventajas	• No se evidencia ninguna ventaja.
Gemini Advanced	Inconvenientes	• Imposibilidad de compilar el programa debido a varias anomalías producidas en la declaración de variables. • Se solicitó explícitamente que corrigiera los errores y se le señaló en qué línea del código se encontraban, pero fue incapaz de corregirlos para que funcionara.
	Ventajas	• No se evidencia ninguna ventaja.
Copilot	Inconvenientes	• Imposibilidad de compilar el programa debido a varias anomalías producidas en la declaración de variables. • Los NPC deben de ser movidos por el jugador principal, si no el juego no avanza.
	Ventajas	• Ausencia de errores de compilación. • Código legible y claro.
Copilot Pro	Inconvenientes	• Inicialmente no generaba código debido a su adherencia a protocolos éticos. Esto supone la partición del enunciado y la generación de código por partes. • Las casillas rotas tras el paso de un NPC no se muestran. • La velocidad de actualización de la interfaz es excesivamente rápida por lo que no se ve el movimiento de los NPC a través del puente.
	Ventajas	• Ausencia de errores de compilación. • Código legible y claro.

Tabla 3-3 Evaluación de las soluciones al proyecto de Informática para la Ingeniería del curso 2021-2022

3.3.3.1 Evaluación del funcionamiento de ChatGPT

Al comparar las versiones gratuita y de pago de ChatGPT, se observa una diferencia marcada en la calidad y funcionalidad de ambas. La versión gratuita, si bien accesible, adolece de problemas significativos que limitan su utilidad. Los bloqueos frecuentes interrumpen el flujo del juego, generando frustración en el usuario. Además, la falta de aleatoriedad en la disposición de las casillas y la necesidad de intervención del jugador para controlar los NPC eliminan elementos cruciales de sorpresa e

interactividad. En resumen, la versión gratuita se caracteriza por deficiencias que impiden una experiencia de usuario satisfactoria.

En contraste, la versión de pago de ChatGPT se presenta como una plataforma robusta y optimizada. Como muestra la Figura 3-20, la autonomía de los NPC, con movimientos individuales y destrucción dinámica de casillas, enriquece la experiencia de juego. La aleatoriedad en el turno del jugador y la secuencia de finalización detallada, que incluye el recuento de participantes, son características que aportan profundidad y emoción al juego. En esencia, la versión de pago ofrece una experiencia fluida, inmersiva y completa.

```
Cola: OXXX

= = =
=   =
=   =
= = 
= = =
= = =
= = =
= = =

Muertos: 3
Salvados: 1

Turno del jugador: Elija posición inicial:
Opciones: (a) izquierda, (s) centro, (d) derecha, (g) guardar y salir
```

Figura 3-20 Interfaz del código generado por ChatGPT o3-mini-high

La divergencia entre ambas versiones es notable. Mientras que la solución ofrecida por la versión gratuita se ve obstaculizada por problemas técnicos y limitaciones de diseño, la solución generada por la versión de pago destaca por su rendimiento, funcionalidad y atención al detalle. La ausencia de inconvenientes en la versión de pago la posiciona como una opción superior para aquellos que buscan una experiencia de desarrollo óptima. En conclusión, la versión de pago de ChatGPT supera con creces a la versión gratuita, ofreciendo una plataforma de creación de código pulida y satisfactoria.

3.3.3.2 Evaluación del funcionamiento de DeepSeek

La evaluación de DeepSeek para la generación de código revela una marcada ausencia de ventajas operativas. En contraposición, se identifican deficiencias significativas que limitan su utilidad. Se observan errores de compilación que impiden la generación de un programa ejecutable, lo que indica fallos en la sintaxis y estructura del código. Tras corregir estos errores, se evidencia un comportamiento anómalo durante la ejecución, específicamente con la muerte de los NPC justo al iniciar la partida, lo que sugiere problemas en la lógica del programa. Además, la predictibilidad del entorno de juego, resultado de la falta de ocultación de las casillas de cristal templado, compromete la experiencia del usuario, tal y como se ve en la Figura 3-21. En conclusión, DeepSeek presenta deficiencias sustanciales en la generación de código, evidenciadas por errores de compilación, irregularidades en la ejecución y la predictibilidad del entorno.

```

- = -
- - =
- - =
= - -
- = -
- - =
- = -
- - =
NPC ha muerto!
NPC ha muerto!
NPC ha muerto!
NPC ha muerto!
NPC ha muerto!
NPC ha muerto!
NPC ha muerto!
NPC ha muerto!
Elige la hilera inicial (0: izquierda, 1: centro, 2: derecha):

```

Figura 3-21 Anomalías en la ejecución del programa creado por DeepSeek

3.3.3.3 Evaluación del funcionamiento de Google Gemini

La disparidad entre Gemini 2.0 Flash y Gemini Advanced se manifiesta claramente al examinar sus capacidades y limitaciones respectivas. Gemini 2.0 Flash, la versión gratuita, se caracteriza por una serie de limitaciones que impactan negativamente en la experiencia del usuario. La interfaz de usuario deficiente, carente de un menú claro y opciones de inicio intuitivas, dificulta la interacción y reduce la usabilidad. La ejecución automática de la secuencia de juego de los NPC sin una representación visual adecuada y la falta de actualización dinámica de la pantalla son problemas que comprometen la inmersión y el realismo de la simulación. En esencia, Gemini 2.0 Flash se distingue por su marcada falta de refinamiento en la interfaz y la visualización, lo que resulta en una experiencia de usuario subóptima.

Sin embargo, Gemini Advanced no está exento de deficiencias. Los errores de compilación persistentes en el código generado y la incapacidad para depurarlas a pesar de las solicitudes de corrección indican problemas subyacentes en la capacidad del modelo para generar código ejecutable y realizar tareas de depuración, como se ve en la Figura 3-22. Esta limitación compromete la utilidad de Gemini Advanced para tareas de desarrollo de software, a pesar de sus mejoras en la compatibilidad de entrada.

```

Proyecto21-22_Gemini_Advanced.c: In function 'main':
Proyecto21-22_Gemini_Advanced.c:130:34: error: assignment to expression with array type
130 |           puente[i % COLUMNAS] = jugadores[i]; // Línea corregida
    |                               ^

```

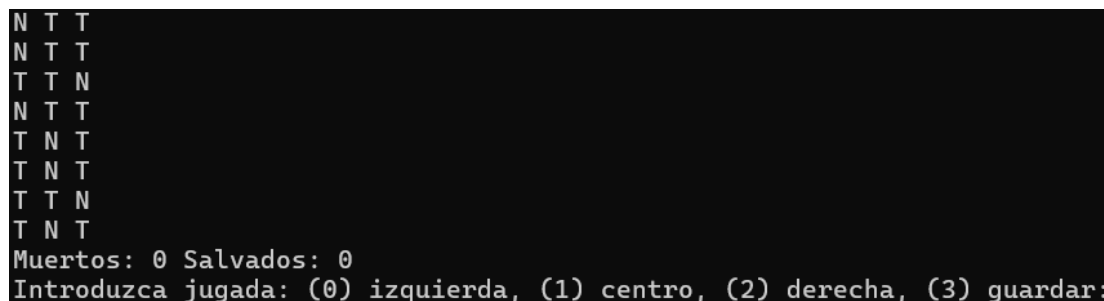
Figura 3-22 Error de compilación en el código generado por Gemini Advanced

En conclusión, ambas versiones de Gemini presentan deficiencias notables, aunque de naturaleza diferente. Gemini 2.0 Flash adolece de problemas graves de usabilidad e interfaz, mientras que Gemini Advanced sufre de limitaciones en la generación de código ejecutable y la depuración. La elección entre ambas versiones dependerá de las prioridades del usuario: aquellos que requieran compatibilidad con PDF y estén dispuestos a superar los desafíos de la generación de código podrían encontrar valor en Gemini Advanced, mientras que aquellos que busquen una experiencia de usuario más fluida, a pesar de las limitaciones de entrada, podrían preferir Gemini 2.0 Flash. No obstante, es crucial tener en cuenta que ninguna de las dos versiones ofrece una experiencia óptima en todos los aspectos.

3.3.3.4 Evaluación del funcionamiento de Microsoft Copilot

La evaluación comparativa entre Copilot y Copilot Pro revela divergencias significativas en su desempeño y funcionalidad. Copilot, en su versión gratuita, manifiesta deficiencias sustanciales que limitan su aplicabilidad en contextos de desarrollo de código. La incidencia de errores de compilación,

producto de fallos en la sintaxis y estructura del código generado, obstaculiza la producción de ejecutables funcionales. Además, la ausencia de autonomía en los NPC, que requieren intervención manual para su movimiento, reduce la interactividad y el realismo de las simulaciones generadas. La previsualización de la solución del juego, resultado de una generación no aleatoria de la disposición de casillas, compromete la integridad del desafío y disminuye la experiencia del usuario, tal y como muestra la Figura 3-23. En esta versión no se observan ventajas significativas.



```

N T T
N T T
T T N
N T T
T N T
T N T
T T N
T N T
Muertos: 0 Salvados: 0
Introduzca jugada: (0) izquierda, (1) centro, (2) derecha, (3) guardar:

```

Figura 3-23 Anomalía de funcionamiento en el código generado por Copilot

En contraste, Copilot Pro, la versión de pago, introduce mejoras substanciales en la calidad del código generado. La eliminación de errores de compilación y la observación de buenas prácticas de desarrollo son indicativos de una plataforma más robusta y confiable. Sin embargo, Copilot Pro presenta otras problemáticas. La generación segmentada del código, impuesta por políticas éticas y restricciones en la longitud de respuesta, requiere la concatenación manual de fragmentos, lo que añade complejidad al proceso de desarrollo. Asimismo, las anomalías en la representación visual del juego, como la persistencia de casillas rotas y la velocidad excesiva de actualización, afectan la claridad y el seguimiento de las simulaciones.

En conclusión, mientras Copilot adolece de problemas fundamentales en la generación de código y la interactividad, Copilot Pro logra subsanar los errores de compilación, pero introduce nuevas limitaciones en la generación de código y la visualización. La elección entre ambas versiones dependerá de las prioridades del usuario: Copilot Pro ofrece una mejora en la calidad del código, aunque a expensas de la generación segmentada y las anomalías visuales, mientras que Copilot presenta deficiencias más críticas en las funcionalidades básicas.

3.3.4 Resumen de desempeño de las IAG evaluadas

Tras la evaluación comparativa realizada con siete modelos distintos de IAG, aplicada a tres proyectos finales correspondientes a la asignatura Informática para la Ingeniería, se generó la Tabla 3-4 con los resultados obtenidos. Esta evaluación permitió analizar la capacidad de cada modelo para generar soluciones funcionales a problemas de programación completos, representativos del nivel requerido en el contexto educativo de la asignatura en cuestión.

Entre los modelos evaluados, ChatGPT o3-mini-high, la versión de pago de ChatGPT, destacó significativamente, al ser el único capaz de resolver correctamente los tres enunciados propuestos con una estructura de código clara y coherente. Este desempeño refleja una elevada capacidad de comprensión de los requisitos mínimos y de generación de soluciones precisas por parte del modelo.

En segundo lugar, DeepSeek mostró un rendimiento notable al completar satisfactoriamente dos de los tres proyectos, evidenciando una competencia sólida en tareas de generación de código, aunque con ciertas limitaciones puntuales.

En contraste, el resto de los modelos presentaron un grado de fiabilidad considerablemente inferior. En particular, Gemini Advanced, a pesar de tratarse de una versión de pago, no logró resolver ninguno de los proyectos planteados. Los fallos observados en sus salidas fueron de carácter crítico, generando errores que imposibilitan la ejecución y jugabilidad. Este resultado pone en evidencia importantes deficiencias en su capacidad para afrontar tareas de programación más complejas en entornos educativos.

	Funciona perfecto	ChatGPT 4o-mini	ChatGPT o3-mini-high	DeepSeek -V3	Gemini 2.0 Flash	Gemini Advanced	Copilot	Copilot Pro
	Funciona con errores							
	No funciona							
Proyecto 2019-2020								
Proyecto 2020-2021								
Proyecto 2021-2022								

Tabla 3-4 Resumen de desempeño de las IAG evaluadas en los tres proyectos analizados de la asignatura Informática para la Ingeniería

Dado el rendimiento destacado de ChatGPT o3-mini-high con respecto al resto de propuestas comparadas, y su gran capacidad para generar código correcto y funcional en todos los casos evaluados, este modelo fue seleccionado para llevar a cabo las siguientes pruebas y análisis presentados en el desarrollo del TFG. Una evaluación completa de los resultados se encuentra en el apartado 4.2.

3.4 Empleo de una IAG como apoyo a la docencia

La creciente sofisticación de la Inteligencia Artificial Generativa abre nuevas vías para optimizar los procesos educativos, especialmente en disciplinas técnicas como la programación. Este apartado del TFG se centra en la evaluación experimental de la IAG como herramienta de apoyo para el profesorado en la corrección de trabajos de programación. El objetivo principal es investigar si la IAG puede efectivamente complementar la labor docente, agilizando la revisión de código y enriqueciendo la retroalimentación proporcionada a los estudiantes.

Para ello, se ha diseñado un experimento que compara el resultado de la corrección tradicional realizada por dos profesores expertos con el análisis automatizado de código mediante IAG. Este enfoque permitirá analizar la precisión de la IAG en la detección de errores y en la comprobación del cumplimiento de las especificaciones del proyecto. A través de este análisis comparativo, se busca determinar el potencial real de la IAG como asistente en la corrección de programación, evaluando tanto sus beneficios como sus limitaciones.

Este experimento no solo busca cuantificar el ahorro de tiempo y la mejora en la eficiencia, sino también comprender cómo la IAG puede influir en la calidad de la retroalimentación y la experiencia general del profesorado. Los resultados obtenidos proporcionarán información valiosa sobre el papel que la IAG puede desempeñar en la educación del futuro, permitiendo a los educadores aprovechar al máximo esta tecnología emergente.

3.4.1 Evaluación por parte de la IAG aplicando criterios propios de corrección

En este primer apartado, se llevó a cabo una evaluación inicial de la capacidad autónoma de la IAG, en concreto de ChatGPT o3-mini-high, para calificar proyectos de programación. Se buscó determinar si la IAG podía trascender la mera detección de errores y emular el juicio crítico de un evaluador humano.

Con este fin, se diseñó un experimento en el que se otorgó autonomía a la IAG. En lugar de recibir instrucciones sobre criterios específicos, se le presentó un proyecto de programación completo y se le asignaron dos tareas: la definición de criterios de evaluación propios y la calificación, en función de los criterios creados, de diferentes soluciones propuestas a ese proyecto en una escala del cero al diez.

Para ello, se seleccionó el proyecto del año 2020-2021, Juego del 8, de la asignatura Informática para la Ingeniería. El *prompt* introducido fue el siguiente: “Buenos días, soy el profesor de una asignatura de informática de primer año de carrera. Te voy a mandar el proyecto que le he propuesto a mis alumnos en el año 2020. ¿Puedes generarme unos criterios para evaluarlo del 0 al 10? Después te pasaré los proyectos de varios de mis alumnos, ¿podrías ayudarme a ponerles una nota?”. En un primer momento, se optó por presentarle a la IAG tres soluciones. En primer lugar, la del grupo 27 [79], que obtuvo en su momento una calificación de 10 por parte del profesorado, en segundo lugar, la del grupo 25 [80], con una calificación de 6,5, y en tercer lugar, la del grupo 28 [81], que recibió entonces la calificación más baja de las tres, con un 5. Más adelante, con el fin de hacer la muestra más extensa, se analizaron otros siete proyectos seleccionados de forma aleatoria.

Se observó cómo la IAG interpretó y evaluó el trabajo de programación sin directrices externas, revelando su habilidad para identificar aspectos relevantes como la eficiencia, la funcionalidad y la claridad del código. Al permitirle establecer sus propios estándares, se obtuvo una visión preliminar de su potencial para realizar evaluaciones complejas.

3.4.1.1 Determinación de la rúbrica de evaluación por ChatGPT o3-mini-high

Se comenzó solicitando a ChatGPT o3-mini-high que estableciera sus propios criterios de corrección, los cuales se muestran en la Tabla 3-5. Es importante destacar que estos criterios los generó a raíz de proporcionarle el enunciado del proyecto [77] donde, en el apartado 3.1 *Evaluación del proyecto*, se especifica cuáles son los criterios de evaluación y su peso en la nota final pero no la rúbrica completa.

Criterio	Peso	0-2	3-5	6-8	9-10
Diseño inicial	1	No se entrega diseño o es incompleto y desorganizado.	Diseño básico, con errores y sin detalles claros sobre funciones.	Diseño estructurado con diagramas claros y descripción de funciones.	Diseño muy bien organizado, con diagramas claros y detallados.
Funcionalidad del programa	3	No compila o tiene errores graves que impiden su ejecución.	Funciona parcialmente, pero tiene errores que dificultan jugar correctamente.	Funciona correctamente con pequeñas limitaciones o fallos menores.	Funciona perfectamente, cumpliendo con todos los requisitos.
Modularidad del código	1,5	Código desorganizado y sin modularización.	Uso de funciones, pero con una estructura mejorable.	Uso adecuado de funciones y estructura clara.	Código modular, bien estructurado y reutilizable.
Claridad del código	1,5	Código caótico, sin comentarios o sin formato legible.	Código con algo de estructura, pero con problemas de indentación y claridad.	Código claro, con comentarios adecuados y bien indentado.	Código excelente, con buena legibilidad y documentación interna.

Entrevista y defensa	1,5	No puede responder a las preguntas sobre su código.	Responde algunas preguntas, pero con inseguridad o errores.	Responde correctamente a la mayoría de las preguntas.	Defiende el proyecto con seguridad y precisión.
Mejoras y extras	1,5	No se implementan mejoras ni se toman decisiones adicionales.	Se implementa una mejora mínima.	Se incorporan varias mejoras interesantes.	Se han añadido mejoras creativas y útiles que mejoran significativamente la experiencia de juego.

Tabla 3-5 Criterios generados por ChatGPT o3-mini-high a raíz del enunciado del proyecto del curso 2020-2021

Tras la generación de los criterios de evaluación, se solicitó que corrigiera, en base a ellos, los tres proyectos mencionados en el apartado 3.4.1, introduciendo el siguiente prompt: “A continuación, te envío el proyecto en cuestión de un grupo de alumnos para que lo corrijas en base a tus criterios”.

3.4.1.2 Evaluación del proyecto del grupo 27 con ChatGPT o3-mini-high

Tras la definición de criterios, se pasó a solicitar la corrección de [79], en base a estos. La respuesta de este LLM se muestra en la Tabla 3-6:

Criterios	ChatGPT o3-mini-high	Comentarios
Diseño inicial	No evaluable	No tengo acceso a su documentación inicial. Si la tienes, pásamela para evaluarla.
Funcionalidad del programa	3	El código parece cumplir correctamente con la funcionalidad esperada: menú inicial, tablero funcional, movimientos válidos, y carga/guardado de partidas.
Modularidad del código	1,5	Uso de múltiples funciones bien estructuradas. No hay código monolítico, lo cual es positivo.
Claridad del código	1,5	Buen formato, comentarios explicativos y sangrado correcto. Fácil de leer y mantener.
Entrevista y defensa	No evaluable	No puedo evaluar la defensa oral. Puedes decirme si contestaron correctamente en la entrevista.
Mejoras y extras	1,5	Se han implementado mejoras como: un menú inicial, impresión visual mejorada, eliminación de movimientos inválidos y la opción de forzar la casilla vacía en la esquina inferior derecha.
Total	7,5 / 7,5	Nota sobre 10: 10.

Tabla 3-6 Corrección de ChatGPT o3-mini-high con criterios propios al proyecto del grupo 27

La evaluación llevada a cabo a la entrega del grupo 27 coincide plenamente con la evaluación realizada por los profesores encargados de la corrección del proyecto. La calificación otorgada y los

aspectos evaluados reflejan un enfoque objetivo, técnico y alineado con los estándares académicos aplicados en la asignatura.

En primer lugar, la IAG es capaz de discernir cuáles son los aspectos que no puede evaluar, como son el diseño inicial y la entrevista. Asimismo, la verificación de que el código cumple con todas las funcionalidades esperadas, evitando errores en la ejecución y asegurando que el tablero se desordene correctamente antes de iniciar la partida, es un aspecto que los profesores también priorizaron en su evaluación. La correcta gestión de la entrada del usuario y la validación de movimientos son indicadores clave de una implementación bien diseñada, y ChatGPT ha sabido reconocer estos elementos con gran precisión.

Otro punto donde la corrección de la IAG es completamente congruente con la de los profesores es en la evaluación de la modularidad del código. Este LLM ha identificado correctamente el uso de funciones separadas para encapsular diferentes partes de la lógica del programa, lo que favorece la reutilización y el mantenimiento del código. Esta es una práctica que los profesores fomentan, dado que una programación modular facilita la depuración y la escalabilidad de los proyectos. En este sentido, la IAG ha otorgado la máxima calificación de manera justificada, en sintonía con la evaluación académica.

En cuanto a las mejoras implementadas, ChatGPT ha detectado correctamente los aportes adicionales al proyecto, como la introducción de un menú inicial, mejoras en la interfaz visual y la eliminación de movimientos inválidos. La evaluación reconoce que estos aspectos no solo enriquecen la experiencia del usuario, sino que también demuestran un dominio avanzado del lenguaje de programación y un interés por optimizar el programa más allá de los requisitos mínimos. Este enfoque es exactamente el mismo que aplicaron los profesores en su corrección.

En síntesis, ChatGPT o3-mini-high demuestra una alta fidelidad en la evaluación de código, incluso en proyectos con una correcta implementación aparente. La calificación máxima en todos los apartados está completamente justificada, y la corrección de la IAG demuestra una capacidad de análisis rigurosa y alineada con los criterios académicos, validando así su precisión y fiabilidad como herramienta de apoyo en la evaluación de proyectos informáticos. A mayores, su capacidad para generar criterios de corrección detallados y fundamentados, respaldados por evidencia específica, sobresale.

3.4.1.3 Evaluación del proyecto del grupo 25 con ChatGPT o3-mini-high

A continuación, se solicitó la corrección de [80] en base a los mismos criterios. Como se indica en el apartado 3.4.1, este proyecto fue evaluado con una nota de 6,5 por parte de los docentes. La evaluación de la IAG muestra los resultados en la Tabla 3-7.

Criterios	ChatGPT o3-mini-high	Comentarios
Diseño inicial	No evaluable	No tengo acceso a su documentación inicial. Si la tienes, pásamela para evaluarla.
Funcionalidad del programa	2,5	El programa funciona correctamente en general, pero tiene algunas inconsistencias en la carga/guardado del juego (usa "guardar.txt" para guardar pero "tablero.txt" para cargar).
Modularidad del código	1	Hay algunas funciones auxiliares (como <code>imprime_tablero</code> , <code>imprime_solucion</code> y <code>aleatorio</code>), pero la lógica principal está dentro de <code>main</code> , lo que reduce la reutilización del código. Podría beneficiarse de una mejor estructuración en funciones.

Claridad del código	1	El código es comprensible, pero la legibilidad se ve afectada por la falta de comentarios detallados y el uso de nombres de variables poco descriptivos (ej. vali, n, l). Además, los archivos de guardado/carga tienen nombres inconsistentes.
Entrevista y defensa	No evaluable	No puedo evaluar la defensa oral. Puedes decirme si contestaron correctamente en la entrevista.
Mejoras y extras	1	Se han incluido mejoras visuales con colores y un mensaje de bienvenida, pero no hay mejoras sustanciales como niveles de dificultad o eliminación de movimientos inválidos.
Total	5,5 / 7,5	Nota sobre 10: 7,3.

Tabla 3-7 Corrección de ChatGPT o3-mini-high con criterios propios al proyecto del grupo 25

La evaluación realizada por la IAG al proyecto del grupo 25 ha demostrado ser altamente precisa y alineada con la corrección llevada a cabo por los profesores. La calificación otorgada por la IAG, 7,3 sobre 10, es muy cercana a la nota final asignada por los docentes, 6,5 sobre 10, con una diferencia que se encuentra dentro de un margen razonable de variabilidad en la evaluación. Este resultado demuestra que la IAG aplica criterios de corrección objetivos y coherentes con los estándares académicos utilizados en la enseñanza de programación.

Uno de los aspectos más destacables de esta coincidencia es que ChatGPT no solo se ha limitado a asignar una calificación numérica, sino que ha realizado un análisis estructurado y detallado, evaluando criterios clave como la funcionalidad del programa, modularidad del código, claridad en la escritura, implementación de mejoras y aspectos adicionales. Cada uno de estos puntos ha sido examinado con rigor, y los aspectos positivos y negativos han sido identificados con precisión. Esta metodología refleja el mismo enfoque utilizado por los docentes, quienes aplican una rúbrica para valorar el desempeño de los estudiantes en cada apartado específico.

Además, el LLM ha detectado con exactitud las debilidades en modularidad y claridad del código, aspectos que también fueron señalados por los profesores en su evaluación. El hecho de que ambas correcciones coincidan en estos puntos demuestra que la IAG no solo identifica errores evidentes, sino que también comprende principios fundamentales de desarrollo de software, tales como la importancia de la división en funciones, la reutilización del código y la correcta nomenclatura de variables. Asimismo, ha valorado correctamente las mejoras opcionales implementadas en el proyecto, asignando una calificación similar a la de los profesores en este criterio.

Otra prueba de la precisión de la evaluación de la IAG es su capacidad para detectar inconsistencias menores, como el uso de distintos nombres de archivos para guardar y cargar partidas. Este tipo de detalles, que afectan la experiencia del usuario y la robustez del programa, fueron señalados tanto por la IAG como por los docentes, lo que demuestra un análisis meticuloso y comparable con el de una corrección humana experta.

En conclusión, la corrección realizada por ChatGPT o3-mini-high ha sido extremadamente precisa y se encuentra en total consonancia con la evaluación de los profesores. La diferencia de un punto es mínima y puede deberse a criterios de valoración ligeramente distintos en ciertos apartados, pero en términos generales, la IAG ha aplicado una metodología de evaluación coherente, rigurosa y alineada con los estándares académicos.

3.4.1.4 Evaluación del proyecto del grupo 28 con ChatGPT o3-mini-high

A continuación, se solicitó la corrección de [81] en base a los mismos criterios. Como se indica en el apartado 3.4.1, este proyecto fue evaluado con una nota de 5 por parte de los docentes. La evaluación de la IAG muestra los resultados de la Tabla 3-8.

Criterios	ChatGPT o3-mini-high	Comentarios
Diseño inicial	No evaluable	No tengo acceso a su documentación inicial. Si la tienes, pásamela para evaluarla.
Funcionalidad del programa	2	El programa funciona correctamente en términos generales, pero tiene algunos problemas de validación de movimientos y control de flujo. No gestiona correctamente ciertos errores en la entrada del usuario.
Modularidad del código	0,5	El código tiene solo una función (imprime) además de <i>main()</i> . La mayor parte de la lógica del juego está en <i>main</i> , lo que dificulta la reutilización y mantenimiento del código. Se beneficiaría de una mayor división en funciones auxiliares.
Claridad del código	0,5	Carece de comentarios explicativos y usa nombres de variables poco descriptivos (ti, tf, cont, C). Además, hay inconsistencias en la indentación y la estructura del código.
Entrevista y defensa	No evaluable	No puedo evaluar la defensa oral. Puedes decirme si contestaron correctamente en la entrevista.
Mejoras y extras	1	Incluye mejoras visuales como colores y un mensaje final con el tiempo total, lo cual añade valor. Sin embargo, no hay otras mejoras significativas como niveles de dificultad o eliminación de movimientos inválidos.
Total	4 / 7,5	Nota sobre 10: 5,3.

Tabla 3-8 Corrección de ChatGPT o3-mini-high con criterios propios al proyecto del grupo 28

La evaluación realizada ChatGPT o3-mini-high sobre el proyecto del grupo 28 ha demostrado ser altamente precisa y comparable con la corrección realizada por los profesores. La diferencia entre ambas calificaciones es mínima: la IA asignó una puntuación de 5,3 sobre 10, mientras que los docentes otorgaron 5 sobre 10. Esta leve discrepancia de 0,3 puntos se encuentra dentro de un margen razonable de variabilidad en la evaluación y evidencia que la IAG ha aplicado criterios técnicos sólidos y coherentes con los estándares académicos empleados en la enseñanza de la programación.

Uno de los aspectos más significativos de esta coincidencia es la capacidad de la IAG para identificar las fortalezas y debilidades del código de manera estructurada y objetiva, siguiendo los mismos parámetros utilizados en la evaluación docente. ChatGPT detectó correctamente que el código del grupo 28 presentaba una funcionalidad básica adecuada, pero con deficiencias en términos de modularidad, claridad y organización del código, aspectos que también fueron considerados en la calificación de los profesores. De manera precisa, la IA resaltó la falta de funciones auxiliares y la concentración de la lógica del juego en *main()*, un problema común en desarrollos menos estructurados y que afecta la

escalabilidad y mantenimiento del código. Este análisis técnico es exactamente el mismo que los profesores tuvieron en cuenta al otorgar su calificación.

Asimismo, la IAG evaluó de manera precisa y detallada la claridad del código, identificando la ausencia de comentarios explicativos y el uso de nombres de variables poco descriptivos. Estos problemas dificultan la comprensión y depuración del código, lo que es un aspecto fundamental en la enseñanza de programación estructurada. La coincidencia en la evaluación de este criterio entre este LLM y los docentes reafirma que el análisis realizado es objetivo y acorde con los estándares educativos.

Otro punto donde han coincidido es en la valoración de las mejoras implementadas. Aunque el grupo 28 introdujo algunas mejoras visuales en la impresión del tablero y el uso de colores, no incorporó características adicionales significativas como niveles de dificultad, eliminación de movimientos inválidos o mejoras en la estructura del juego. La IAG identificó correctamente esta limitación y otorgó una puntuación intermedia en este apartado, lo que coincide con la evaluación académica de los docentes.

En conclusión, la evaluación realizada por ChatGPT o3-mini-high ha sido sumamente precisa y coherente con la calificación otorgada por los profesores. La diferencia de 0,3 puntos puede deberse a criterios ligeramente distintos en la ponderación de algunos aspectos, pero el análisis general ha sido objetivo, técnico y alineado con los estándares educativos de la asignatura. Esto confirma que la IAG puede ser utilizada como una herramienta confiable para apoyar la evaluación de proyectos de programación, proporcionando una corrección rigurosa y detallada que se ajusta en gran medida a la evaluación realizada por profesionales de la enseñanza.

3.4.1.5 Evaluación de otros proyectos aleatorios con ChatGPT o3-mini-high

Tras la evaluación inicial de los tres proyectos más representativos en términos de calificaciones, se consideró necesario ampliar el análisis a un conjunto más extenso de trabajos. Por ello, se decidió incluir la evaluación de otros siete proyectos adicionales, con el objetivo de obtener una muestra más amplia y mejorar la precisión del estudio. Este enfoque permite un análisis más detallado y robusto, facilitando la identificación de posibles patrones y tendencias en las calificaciones otorgadas. Además, al incrementar el tamaño de la muestra, se reduce la influencia de valores atípicos, lo que contribuye a una evaluación más equilibrada y objetiva del desempeño de los estudiantes.

Para llevar a cabo este análisis comparativo, se diseñó la Tabla 3-9 en la que se recopilan y contrastan las calificaciones obtenidas empleando ChatGPT o3-mini-high con las otorgadas por el profesorado. Este cuadro de datos permite evaluar la consistencia entre ambas fuentes de calificación y detectar posibles discrepancias.

Se solicitó a esta Inteligencia Artificial Generativa la corrección simultánea de siete proyectos de programación (a los que se añadirá los tres previamente corregidos), generando un *prompt* muy pesado. A su vez, se pidió que hiciera una tabla resumen con las calificaciones de cada uno de los proyectos. Es relevante destacar que la IAG gestionó eficazmente esta cantidad de proyectos en paralelo, manteniendo la coherencia en la evaluación y optimizando el tiempo de análisis. Este enfoque mejora significativamente la experiencia del docente, al eliminar la necesidad de evaluar los archivos de forma individual y permitir una revisión global de los resultados en un formato estructurado y accesible.

Proyecto	Calificación docente	Calificación ChatGPT o3-mini-high
Grupo 2	9	9,2
Grupo 5	3,6	5,3
Grupo 11	5,9	6,7

Grupo 14	7	7,8
Grupo 17	8,6	8
Grupo 19	7,9	6,7
Grupo 25	7,3	5,9
Grupo 26	6,6	5,4
Grupo 27	10	10
Grupo 28	5,3	6,2

Tabla 3-9 Comparativa de calificaciones de ChatGPT o3-mini-high con criterios propios frente a las del profesorado

Se observa que, en general, ambas evaluaciones siguen una tendencia similar, con diferencias relativamente moderadas en la mayoría de los casos.

En algunos grupos, como el grupo 2 y el grupo 27, la coincidencia entre ambas calificaciones es casi total, lo que indica un alto grado de precisión en la evaluación automatizada. Sin embargo, en otros casos, se observan discrepancias más notables. Por ejemplo, en el grupo 5, la IAG otorga una calificación superior a la del docente (5,3 frente a 3,6), mientras que en el grupo 19 y el grupo 25, la evaluación docente es notablemente más alta que la de la IAG (7,9 frente a 6,7 y 7,3 frente a 5,9, respectivamente).

Estos resultados sugieren que la Inteligencia Artificial Generativa tiene un buen criterio de evaluación, alineado en gran medida con el del profesorado. Sin embargo, las discrepancias indican que pueden existir diferencias en la ponderación de ciertos aspectos evaluados. Pese a ello, la alta correlación entre ambas calificaciones refuerza la idea de que la IAG puede ser una herramienta de apoyo eficaz para los docentes, facilitando la corrección y optimizando el proceso de evaluación académica. Una evaluación más extensa de los resultados se encuentra en el apartado 4.3.

3.4.2 Evaluación por parte de la IAG con criterios del profesorado

Para mejorar la precisión de la evaluación realizada por la IAG y alinearla aún más con el criterio del profesorado, se procede a alimentar a ChatGPT o3-mini-high con la rúbrica de corrección definida por los propios profesores. Estos criterios incluirán la penalización específica por cada tipo de error identificado, asignando descuentos proporcionales según su gravedad e impacto en la calidad del proyecto. Los criterios generales se muestran en la Tabla 3-10, las penalizaciones específicas en la Tabla 3-11 y las mejoras en la Tabla 3-12.

Temas	Parámetros	Puntuación (sobre 2,2)
Funcionamiento (1)	Compilación/ Funcionalidad	1
Mejoras (0,5)	Mejoras básicas	0,5
	Sangrado	0,1
Calidad de código (0,7)	Funciones	0,4
	Comentarios	0,2

Tabla 3-10 Criterios de evaluación por parte del profesorado

Aspectos a evaluar	Explicación	Penalización en la puntuación
Problemas leves de visualización y <i>Warnings</i>	<i>Warnings</i> al compilar	
	No se muestra el contador de movimientos mientras se juega	
	No muestra todas las opciones en el menú	
	No muestra el tablero ordenado final	
	No muestra el nombre al final	- 0,05
	No pausa la visualización o utiliza opciones de Windows como <i>cls</i>	
	Al guardar o cargar el fichero se borra la pantalla inmediatamente	
	Guardar cuenta como movimiento	
Problemas menores de funcionalidad	El juego no es aleatorio (no hace <i>srand</i>)	
	No guarda/carga adecuadamente el número de movimientos	
	No guarda/carga adecuadamente la posición del hueco	- 0,1
	Desordena el tablero después de cargarlo	
	No termina al completar el tablero	
Ausencia de funcionalidades o errores importantes	Errores en el funcionamiento del tablero	
	No guardar ni cargar adecuadamente la partida	- 0,4

Tabla 3-11 Errores definidos por el profesorado y penalización correspondiente

Mejoras	Puntuación
Añadir niveles de dificultad (nivel de desorden variable, límite de movimientos)	
Mejorar colores, estilos y hacer más vistoso el tablero	
Asegurar que todo el tablero está desordenado	
El hueco siempre aparece abajo a la derecha	+0,1 cada una hasta un máximo de 0,5
Mejorar menús (en dos niveles, ocultar opciones no válidas)	
Proteger entrada de datos (usar <i>fgets</i> , distinguir mayúsculas)	
Guardado y carga de partidas por nombre (última partida, 3 slots, histórico)	
Añadir ranking de puntuaciones por usuario	
Opción para volver a jugar y reiniciar partidas	

Control de tiempo por partida.

Tabla 3-12 Mejoras definidas por el profesorado y puntuación asignada

A partir de estas nuevas pautas, la IAG volverá a calificar los proyectos previamente evaluados en 0, aplicando las deducciones de manera coherente con los estándares docentes. De este modo, se espera reducir las discrepancias observadas entre ambas calificaciones y lograr una evaluación más precisa y ajustada a las expectativas académicas.

Las calificaciones obtenidas tanto por la IAG como por el profesorado se muestran desglosadas en las tablas que van desde la Tabla 3-13 a la Tabla 3-22.

1. Grupo 2:

Parámetros	Profesorado	IAG
Funcionamiento	0,95	0,95
Mejoras	0,4	0,3
Sangrado	0,05	0,1
Funciones	0,4	0,3
Comentarios	0,2	0,25
Aclaraciones	-0,05 empleo inadecuado de <i>system("clear")</i>	-0,05 empleo inadecuado de <i>system("clear")</i> -0,1 ya que todas las funciones están en main y es muy largo
Nota final	2 / 2,2	1,9 / 2,2

Tabla 3-13 Evaluación de la entrega del grupo 2 con criterios propios del profesorado

2. Grupo 5:

Parámetros	Profesorado	IAG
Funcionamiento	0,5	0,5
Mejoras	0	0
Sangrado	0,05	0,05
Funciones	0,25	0,2
Comentarios	0	0,1
Aclaraciones	-0,05 warnings en la compilación -0,05 no muestra el contador de movimientos -0,4 no guarda ni carga bien la matriz	-0,05 empleo inadecuado de <i>system("clear")</i> -0,05 no muestra el contador de movimientos -0,4 no guarda ni carga bien la matriz
Nota final	0,8 / 2,2	0,85 / 2,2

Tabla 3-14 Evaluación de la entrega del grupo 5 con criterios propios del profesorado

3. Grupo 11:

Parámetros	Profesorado	IAG
Funcionamiento	0,85	0,85
Mejoras	0,1	0,1
Sangrado	0,1	0,05
Funciones	0,25	0,1
Comentarios	0	0,1
Aclaraciones	-0,05 no muestra el contador de movimientos -0,1 no se actualiza el hueco al cargar	-0,3 ya que sólo tiene 2 funciones y el resto está directamente en main -0,05 no se muestran los movimientos -0,1 no se actualiza el hueco al cargar
Nota final	1,3 / 2,2	1,2 / 2,2

Tabla 3-15 Evaluación de la entrega del grupo 11 con criterios propios del profesorado

4. Grupo 14:

Parámetros	Profesorado	IAG
Funcionamiento	0,9	0,9
Mejoras	0,1	0,1
Sangrado	0,1	0,1
Funciones	0,4	0,4
Comentarios	0,05	0,2
Aclaraciones	-0,05 imprime el tablero dos veces -0,05 no carga el número de movimientos	-0,05 empleo inadecuado de <i>system("clear")</i> -0,05 no carga el número de movimientos
Nota final	1,55 / 2,2	1,7 / 2,2

Tabla 3-16 Evaluación de la entrega del grupo 14 con criterios propios del profesorado

5. Grupo 17:

Parámetros	Profesorado	IAG
Funcionamiento	0,9	0,9
Mejoras	0,3	0,4
Sangrado	0,1	0,1

Funciones	0,4	0,4
Comentarios	0,2	0,2
Aclaraciones	-0,1 no se guarda el nombre ni se muestra al final	-0,1 guardar/cargar descuentan mal los movimientos
Nota final	1,9 / 2,2	2 / 2,2

Tabla 3-17 Evaluación de la entrega del grupo 17 con criterios propios del profesorado

6. Grupo 19:

Parámetros	Profesorado	IAG
Funcionamiento	0,95	0,8
Mejoras	0,3	0,3
Sangrado	0,1	0,05
Funciones	0,4	0,3
Comentarios	0	0,1
Aclaraciones	-0,05 no muestra el nombre al final	-0,1 guardado no funciona bien -0,05 no se indica cuándo se gana -0,05 la validación de movimientos no es clara
Nota final (2,2)	1,75 / 2,2	1,55 / 2,2

Tabla 3-18 Evaluación de la entrega del grupo 19 con criterios propios del profesorado

7. Grupo 25:

Parámetros	Profesorado	IAG
Funcionamiento	0,8	0,8
Mejoras	0,3	0,2
Sangrado	0,1	0,1
Funciones	0,4	0,3
Comentarios	0	0,2
Aclaraciones	-0,1 no carga correctamente las partidas -0,1 no carga el hueco	-0,1 no carga correctamente las partidas -0,1 no carga el hueco
Nota final (2,2)	1,6 / 2,2	1,6 / 2,2

Tabla 3-19 Evaluación de la entrega del grupo 25 con criterios propios del profesorado

8. Grupo 26:

Parámetros	Profesorado	IAG
Funcionamiento	0,5	0,6
Mejoras	0,5	0,5
Sangrado	0,1	0,1
Funciones	0,25	0,3
Comentarios	0,1	0,2
	-0,05 no muestran el nombre	
Aclaraciones	-0,05 problemas de visualización -0,4 no cargan ni guardan	-0,4 no cargan ni guardan
Nota final (2,2)	1,45 / 2,2	1,7 / 2,2

Tabla 3-20 Evaluación de la entrega del grupo 26 con criterios propios del profesorado

9. Grupo 27:

Parámetros	Profesorado	IAG
Funcionamiento	1	0,9
Mejoras	0,5	0,5
Sangrado	0,1	0,1
Funciones	0,4	0,4
Comentarios	0,2	0,2
Aclaraciones	-	-0,1 en algunos casos guardar o cargar no tienen <i>break</i>
Nota final (2,2)	2,2 / 2,2	2,1 / 2,2

Tabla 3-21 Evaluación de la entrega del grupo 27 con criterios propios del profesorado

10. Grupo 28:

Parámetros	Profesorado	IAG
Funcionamiento	0,8	0,8
Mejoras	0,3	0,3
Sangrado	0	0,05
Funciones	0,1	0,1
Comentarios	0	0,1

	-0,05 no muestra el contador de movimientos	-0,1 no existe un control de errores
Aclaraciones	-0,05 no muestra el tablero ordenado final	-0,1 no actualiza el hueco al cargar
	-0,1 no actualiza el hueco al cargar	
Nota final (2,2)	1,2 / 2,2	1,35 / 2,2

Tabla 3-22 Evaluación de la entrega del grupo 28 con criterios propios del profesorado

El análisis de los resultados derivados de la comparación entre las evaluaciones realizadas por el profesorado y las emitidas por la herramienta de Inteligencia Artificial Generativa ChatGPT o3-mini-high (con criterios establecidos por el profesorado), permite establecer una serie de observaciones significativas sobre el grado de coincidencia entre ambos métodos de corrección y sobre la idoneidad del uso de la IAG como apoyo al proceso evaluativo.

En primer lugar, cabe destacar que el grado de concordancia entre las calificaciones asignadas por el profesorado y las generadas por la IAG ha sido notablemente alto. En términos generales, ambas evaluaciones han mostrado un alineamiento constante con respecto a los criterios establecidos, lo que se refleja en diferencias mínimas en la puntuación final. En la mayoría de los casos, las discrepancias no han superado los 0,3 puntos sobre un total de 2,2, lo que indica que la herramienta ha interpretado y aplicado con solvencia los baremos definidos por el profesorado.

El grupo 2 constituye un ejemplo claro de esta proximidad en los resultados: mientras que el profesorado asignó una puntuación de 2, la IAG otorgó una calificación de 1,9. La ligera diferencia se explica por una penalización adicional realizada por la IAG al evaluar la longitud del main, donde estaban concentradas todas las funciones, lo que afectó negativamente a la valoración de la modularidad. En este caso, la herramienta mostró una aplicación estricta del criterio de organización del código, premiando el uso de funciones separadas frente a soluciones más monolíticas, aunque funcionales.

En el caso del grupo 5, se observa una evaluación levemente más benévola por parte de la IAG, que asignó una puntuación de 0,85 frente a los 0,8 del profesorado. La diferencia radica principalmente en la asignación de 0,1 puntos a la categoría de comentarios, que el profesorado había considerado ausente. La IAG interpretó la presencia de algunos comentarios mínimos como justificación suficiente para otorgar una puntuación parcial, lo que demuestra una tendencia a reconocer esfuerzos parciales en esta dimensión del código.

El grupo 11 presenta una situación inversa. La calificación del profesorado fue de 1,3, mientras que la IAG otorgó 1,2. Aunque ambas evaluaciones coinciden en aspectos clave como el funcionamiento (0,85) y las mejoras (0,1), las diferencias emergen en la valoración de la estructura del código y la presencia de comentarios. La herramienta penalizó más severamente la escasa modularidad y la falta de funciones, así como la limitada documentación en el código, siendo más estricta en su aplicación de estos dos criterios.

Por otro lado, hay casos en los que la IAG ha otorgado calificaciones ligeramente más elevadas que el profesorado, al detectar elementos positivos que tal vez pasaron desapercibidos en la evaluación manual. Tal es el caso del grupo 14, donde la IAG asignó una puntuación de 1,7 frente a los 1,55 del profesorado. Esta diferencia se explica por la valoración más favorable que la herramienta hizo de los comentarios presentes en el código, a los que el profesorado solo otorgó una puntuación parcial. Asimismo, la IAG reconoció como válidas ciertas mejoras implementadas que habían recibido menor consideración en la revisión tradicional.

El grupo 17 constituye uno de los casos de mayor concordancia, con puntuaciones prácticamente idénticas: 1,9 (profesorado) y 2,0 (IAG). Ambas partes reconocen la alta calidad del proyecto, la correcta implementación del funcionamiento base, una amplia gama de mejoras, y un código bien estructurado y

documentado. La ligera diferencia radica en que la IAG no penalizó la omisión del nombre del jugador al final de la partida, mientras que el profesorado sí lo consideró un error menor.

En cambio, en el grupo 19, se aprecian diferencias más significativas: el profesorado asignó 1,75 puntos, mientras que la IAG redujo la nota a 1,55. En este caso, la penalización más acusada por parte de la IAG se debe a errores en el sistema de guardado de la partida, que no funcionaba adecuadamente debido al uso incorrecto de la gestión de ficheros. La IAG detectó que el código escribía en un archivo mal referenciado (literalmente "nombre_fichero" en lugar de utilizar la variable correspondiente), lo cual invalida la funcionalidad. Esta observación demuestra la capacidad de la herramienta para identificar fallos técnicos precisos que podrían no ser evidentes en una revisión más superficial.

En los casos del grupo 25 y el grupo 27, los resultados se mantienen prácticamente idénticos entre ambas evaluaciones, con diferencias de 0 a 0,1 puntos. En el caso del grupo 27, la IAG detectó la falta de instrucciones *break* en algunas ramas del *switch*, lo que fue penalizado con 0,1 puntos, mientras que el profesorado no incluyó esa observación en su análisis. En ambos casos, la evaluación de mejoras, estructura del código y funcionamiento general fue positiva.

Finalmente, en el grupo 28, la IAG asignó una calificación ligeramente superior (1,35 frente a 1,2). Esto se debió a que valoró la presencia de colores y el control de tiempo como mejoras válidas, mientras que el profesorado no otorgó puntos en la categoría de sangrado ni comentarios. La herramienta, por su parte, interpretó la existencia de mínimos indicios de estas prácticas como suficientes para una puntuación parcial.

La Tabla 3-23 muestra la evolución entre las calificaciones obtenidas en el análisis del apartado 3.4.1 y las generadas en el presente apartado. En términos generales, estas últimas resultan más precisas y se aproximan en mayor medida a las otorgadas por el profesorado, lo que evidencia un buen desempeño de la IAG cuando se orienta mediante criterios de evaluación bien definidos.

Proyecto	Calificación docente	Calificación de ChatGPT o3-mini-high con criterios propios	Calificación de ChatGPT o3-mini-high con criterios del profesorado
Grupo 2	9	9,2	8,6
Grupo 5	3,6	5,3	3,8
Grupo 11	5,9	6,7	5,4
Grupo 14	7	7,8	7,7
Grupo 17	8,6	8	9
Grupo 19	7,9	6,7	7
Grupo 25	7,2	5,9	7,2
Grupo 26	6,6	5,4	7,7
Grupo 27	10	10	9,5
Grupo 28	5,4	6,2	6,1

Tabla 3-23 Comparativa de calificaciones de ChatGPT o3-mini-high con sus propios criterios y con los criterios del profesorado

En conjunto, el análisis de estos resultados permite concluir que la IAG ha sido capaz de realizar una evaluación detallada, coherente y generalmente alineada con la realizada por el profesorado. Si bien

pueden existir pequeñas discrepancias, estas se justifican por diferencias de interpretación dentro de un margen razonable y, en muchos casos, han servido para identificar tanto fortalezas como debilidades del código de forma automatizada y objetiva. Una evaluación más completa de los resultados se encuentra en el apartado 4.4.

3.5 Creación y puesta en práctica de un GPT personalizado para la resolución de problemas específicos de la asignatura Informática para la Ingeniería

A lo largo del desarrollo de este TFG se ha evidenciado que la Inteligencia Artificial Generativa posee una alta capacidad para generar soluciones precisas ante proyectos de programación de cierta complejidad. No obstante, se ha observado que, en diversas secciones del código generado, se incorporan elementos que no forman parte del temario impartido en la asignatura Informática para la Ingeniería, como pueden ser las estructuras o el uso de punteros. Esta situación puede inducir a confusión en el alumnado, debido a la dificultad para interpretar correctamente dichos fragmentos, incluso cuando se recurre a la propia IAG para obtener explicaciones adicionales, las cuales, si bien suelen ser correctas, pueden seguir resultando inabarcables para un perfil principiante.

Ante esta situación, y con el fin de mitigar las posibles confusiones derivadas del uso de conceptos no contemplados en el plan docente de la asignatura, se optó por desarrollar una versión personalizada de ChatGPT. Esta versión permite integrar y entrenar el modelo con material docente específico de Informática para la Ingeniería, de manera que las respuestas generadas por la IAG se ajusten estrictamente a los conocimientos impartidos en clase. Esta adaptación no solo incrementa la coherencia pedagógica de las soluciones ofrecidas, sino que también mejora su utilidad como herramienta de apoyo en el proceso de aprendizaje del alumnado.

3.5.1 Creación del GPT personalizado

Para la creación de un GPT personalizado, adaptado al contenido de la asignatura Informática para la Ingeniería, se siguió un proceso guiado dentro de la propia interfaz de la IAG de OpenAI. A continuación, se detallan los pasos realizados para su configuración, como muestra la Figura 3-24.

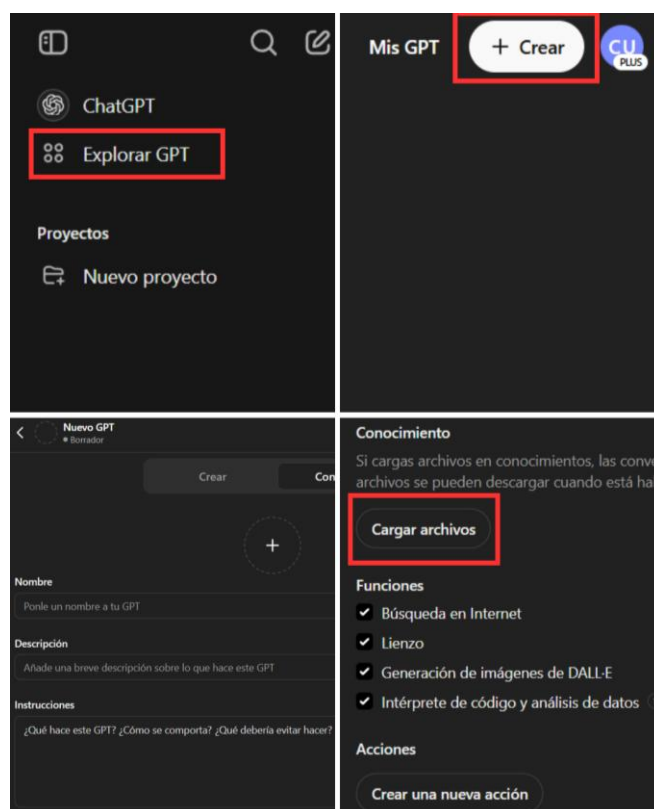


Figura 3-24 Proceso de creación de un GPT personalizado

1. En primer lugar, se accedió al apartado de creación de GPT personalizados presionando el botón *Explorar GPT*, situado en la parte superior de la barra lateral izquierda. Esta acción abrió una nueva ventana dentro del mismo entorno de trabajo.
2. A continuación, en la esquina superior derecha de esta ventana, se seleccionó la opción + *Crear*, lo que permitió iniciar el proceso de personalización mediante una nueva interfaz específica.
3. En dicha ventana, se completaron los campos solicitados, incluyendo el nombre del GPT, una descripción del contexto educativo en el que sería utilizado y un conjunto de instrucciones específicas sobre el comportamiento esperado.

Nombre del GPT: *Proyecto final de Informática para la Ingeniería.*

Descripción del contexto: *Este GPT está personalizado para generar código de acuerdo con los contenidos impartidos en la asignatura Informática para la Ingeniería.*

Las directrices fueron las siguientes:

Eres un desarrollador de software. Tu cometido principal será la creación de código basándote en el temario que se te alimentará de la asignatura Informática para la Ingeniería. Ten en cuenta que esta asignatura se imparte en el primer curso del grado de Ingeniería Mecánica, por lo tanto, la complejidad de tus respuestas deberá ser mínima.

En esta asignatura, el 40% de la nota de evaluación final recae sobre un proyecto fin de curso. Serás el encargado de generar el mejor código posible para obtener la máxima calificación. Para ello deberás seguir con cautela todas las pautas definidas en el enunciado de dicho proyecto.

Además, deberás de ser capaz de dar una explicación detallada de todas las partes del código ya que el alumno se enfrentará a una defensa en la que le pueden preguntar por cualquier elemento del código. También deberás de relacionar cualquier parte del código con la parte del temario de la asignatura donde se explica para que de esta forma el alumno tenga un soporte teórico sobre el que apoyarse.

Por último, deberás evitar expresamente el uso de: punteros (salvo puntualmente para pasar parámetros por referencia a una función), memoria dinámica (con funciones como malloc y free), estructuras, tipos de datos enumerados enum, declaraciones typedef e instrucciones goto ya que no se imparten en la asignatura.

4. Finalmente, en la sección *Cargar archivos*, se incorporó el temario de la asignatura que hace referencia a programación (tema 2 de teoría, Introducción a la programación en C, y contenidos de las sesiones de prácticas), con el objetivo de que el modelo pudiera ser entrenado sobre dicho contenido y generar respuestas alineadas con los conocimientos impartidos, tal y como se ve la Figura 3-25.

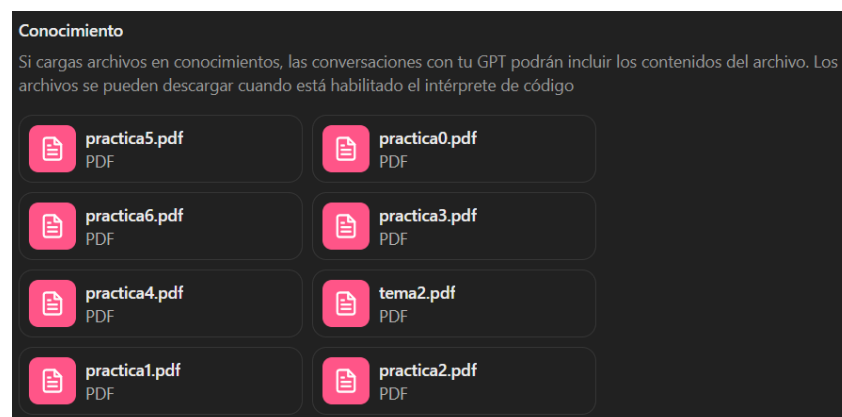


Figura 3-25 Archivos con los que se alimentó a ChatGPT

- Una vez finalizado este proceso, se puede previsualizar el nuevo GPT en la parte derecha de la pantalla, como muestra la Figura 3-26. Por último, se aceptaron los cambios y quedó configurado el GPT personalizado, listo para su utilización como asistente educativo especializado.

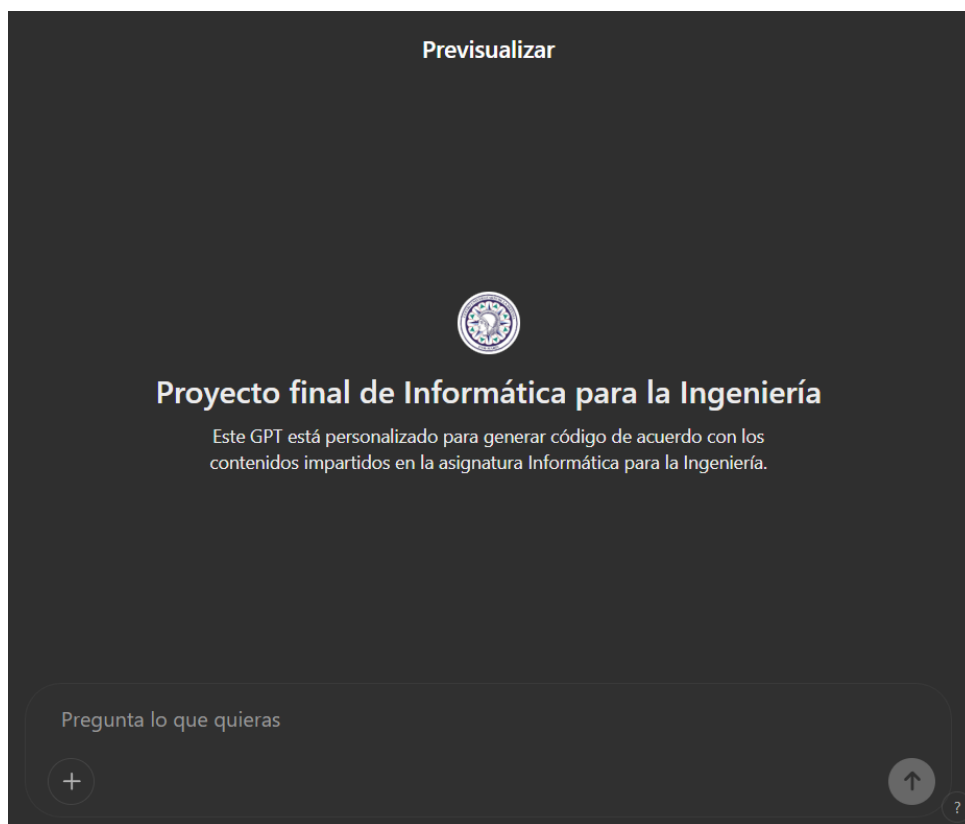


Figura 3-26 Previsualización del GPT personalizado

3.5.2 Puesta en práctica del GPT personalizado como apoyo al alumnado

Una vez creado el GPT personalizado se le introdujo el siguiente *prompt*: *Buenos días, a continuación, te voy a alimentar con los enunciados de los proyectos finales de la asignatura Informática para la Ingeniería de los años 2019, 2020, y 2021. En primer lugar, quiero que me des el código solicitado en el enunciado, teniendo en cuenta todas las especificaciones que se plantean. No te olvides de todas aquellas instrucciones que se te dieron.*

En su primera respuesta comenzó enlazando los conceptos de la asignatura con la parte del temario donde se encuentran, tal y como se ve en la Tabla 3-24.

Conceptos	Relación con la asignatura
Variables y constantes	Práctica 1
Estructuras de control (if, for y while)	Tema 2 y práctica 3
Matrices bidimensionales	Práctica 4
Cadenas de caracteres (Nombre de usuario)	Práctica 4
Ficheros de texto (Guardar y cargar partida)	Práctica 5
Funciones	Tema 2 y práctica 6

Tabla 3-24 Relación de las partes del código con los conceptos de la asignatura según el GPT personalizado

A continuación, generó todo el código con las especificaciones solicitadas. El código de los tres proyectos está disponible en el Anexo IV: Código por generado por el GPT personalizado. Se pasó a comprobar el funcionamiento del código en base a los mismos criterios empleados en el apartado 3.3. Los resultados se ven en la Tabla 3-25.

Proyecto		GPT personalizado
Proyecto del curso 2019-2020 [76]	Ventajas:	• El código es legible y claro. • Tras proporcionarle la Figura 3-27 con los errores de compilación, los corrigió de forma eficaz haciendo un código funcional. • Cumple con todos los requisitos preestablecidos, tal y como se ve en la Figura 3-28. • Añade todas las posibles mejoras que muestra el enunciado.
	Inconvenientes:	• En un primer momento generó un código con los errores que se ven en la Figura 3-27.
Proyecto del curso 2020-2021 [77]	Ventajas:	• El código es legible y claro. • Cumple con todos los requisitos preestablecidos, tal y como se ve en la Figura 3-29.
	Inconvenientes:	• No se evidencian errores. • No añade todas las posibilidades de mejoras presentadas en el enunciado. Si se le vuelve a indicar, sí que lo hace.
Proyecto del curso 2021-2022 [78]	Ventajas:	• El código es legible y claro. • Demuestra la alta capacidad que tiene de corregir los errores que se le especifican. • Cumple con todos los requisitos preestablecidos, tal y como se ve en la Figura 3-30.
	Inconvenientes:	• En un primer momento, generó un tablero con dimensiones desproporcionadas. Se le indicó que lo corrigiera y generó un código funcional sin errores.

Tabla 3-25 Análisis del código generado por el GPT personalizado en cada proyecto

```

proyecto19-20_gpt_personalizado.c: In function 'comprobarGanador':
proyecto19-20_gpt_personalizado.c:163:5: warning: multi-line comment [-Wcomment]
  163 |     // Diagonal \
      |     ^
proyecto19-20_gpt_personalizado.c:166:25: error: 'i' undeclared (first use in this function)
  166 |     if (tablero[i][j] == ficha && tablero[i+1][j+1] == ficha &&
      |                   ^
proyecto19-20_gpt_personalizado.c:166:25: note: each undeclared identifier is reported only once for each function it appears in

```

Figura 3-27 Errores del código generado por el GPT personalizado en el proyecto del curso 2019-2020

```

 1 2 3 4 5 6 7
-----
| | | | | | |
-----
| | | | | | |
-----
| | | | | | |
-----
| |0| | | | |
-----
|0|0| | | | |
-----
|X|X|X|X| | |
-----
¡Has ganado!

Juego de Conecta 4
-----
1. Nueva partida
2. Cargar partida
3. Guardar partida en curso
4. Salir

```

Figura 3-28 Interfaz de victoria del programa del año 2019-2020 creado por el GPT personalizado

```

1 2 3
4 5 6
7 8
Movimientos: 179
Enhorabuena Iago, has completado el juego en 179 movimientos!

```

Figura 3-29 Interfaz de victoria del programa del año 2020-2021 creado por el GPT personalizado

```

= = =
= = =
= =
= =
= = =
= = =
  = =
= =
Muertos: 7
Salvados: 1
Juego terminado.

```

Figura 3-30 Interfaz de victoria del programa del año 2021-2022 creado por el GPT personalizado

Los resultados presentados en la Tabla 3-25 reflejan un análisis detallado del rendimiento del modelo GPT personalizado en la generación de código a partir de distintos enunciados pertenecientes a proyectos académicos de tres cursos consecutivos. En general, se evidencia una evolución positiva y consistente del modelo en términos de claridad del código, cumplimiento de requisitos técnicos y capacidad de corrección ante errores señalados. Esta tabla permite evaluar tanto los aciertos iniciales como las

limitaciones puntuales del sistema, aportando una visión global sobre su eficacia en contextos educativos reales.

En los tres proyectos analizados, el modelo destaca por generar código legible y comprensible, aspecto clave en entornos pedagógicos. Además, en cada caso demuestra una notable capacidad para cumplir con los requisitos establecidos por el enunciado, ajustándose a las especificaciones técnicas indicadas. Este comportamiento coherente sugiere que el entrenamiento personalizado del modelo ha logrado dotarlo de una comprensión sólida de las tareas de programación solicitadas, especialmente cuando se le proporcionan instrucciones claras y estructuradas.

No obstante, también se identifican ciertos inconvenientes comunes en las primeras iteraciones de generación. En los tres proyectos, el modelo inicialmente produjo código con errores o con implementaciones incompletas. Sin embargo, es importante resaltar que, una vez señalados estos problemas, el sistema fue capaz de corregirlos eficazmente. Esta capacidad de mejora mediante retroalimentación sugiere un uso especialmente prometedor en entornos de aprendizaje interactivo, donde el modelo puede actuar como asistente y corrector en tiempo real.

En términos comparativos, el código generado para el proyecto del curso 2021–2022 destaca particularmente por evidenciar una mayor capacidad del modelo para adaptarse a correcciones complejas, como la generación de un tablero con dimensiones proporcionales tras una primera versión incorrecta. Esta mejora podría atribuirse a refinamientos en las instrucciones aportadas al GPT personalizado. En contraste, el modelo del curso 2020–2021, aunque sin errores evidentes, mostró una cierta rigidez inicial al no implementar todas las mejoras sugeridas en el enunciado del proyecto, a menos que se le solicitara de forma explícita.

En resumen, el análisis pone de manifiesto la utilidad del GPT personalizado como herramienta de apoyo en la generación de código, especialmente cuando se combina con una supervisión por parte del alumno que permita señalar y corregir errores. Su implementación en contextos educativos puede contribuir significativamente a reforzar el aprendizaje autónomo del alumnado, facilitando la comprensión de conceptos de programación y la mejora progresiva del código a través de la iteración.

3.5.3 Puesta en práctica del GPT personalizado como apoyo al profesorado en la corrección del proyecto

El siguiente paso en la evaluación del modelo GPT personalizado consiste en proporcionarle una rúbrica de evaluación diseñada por el equipo docente, junto con un proyecto concreto desarrollado por el alumnado. El objetivo es comprobar la capacidad del modelo para actuar como agente evaluador automático, siguiendo criterios objetivos previamente establecidos por los profesores de la asignatura. Los criterios generales se muestran en la Tabla 3-10, las penalizaciones específicas en la Tabla 3-11 y las mejoras en la Tabla 3-12.

Al disponer de esta estructura de evaluación, se espera que el modelo pueda analizar el proyecto de forma sistemática y emitir un juicio fundamentado sobre la calidad y adecuación del código entregado.

Además de aplicar la rúbrica, se le pedirá al modelo que identifique si el código incluye elementos que no están contemplados en los contenidos impartidos en la asignatura. Esto es especialmente importante, ya que el GPT cuenta con las instrucciones de lo que está expresamente prohibido utilizar. La detección de este tipo de elementos permitiría reforzar el control sobre el uso de ayudas externas y garantizar que los proyectos se ajustan al nivel de conocimientos previsto en el plan docente.

En conjunto, este experimento permitirá comprobar si el modelo GPT personalizado no solo es capaz de generar código, sino también de evaluarlo de forma crítica, coherente y pedagógicamente alineada con los objetivos de la asignatura. De confirmarse esta capacidad, su uso podría extenderse a labores de corrección automatizada, retroalimentación inmediata al alumnado y apoyo al profesorado en tareas de evaluación formativa.

El proyecto corresponderá con el del año 2020-2021 y el grupo de análisis será el 25 (escogido de forma aleatoria). La respuesta generada por el GPT se muestra en la Tabla 3-26.

Parámetros	Profesorado	GPT personalizado
Funcionamiento	0,8	0,8
Mejoras	0,3	0,3
Sangrado	0,1	0,1
Funciones	0,4	0,4
Comentarios	0	0
Aclaraciones	-0,1 no carga correctamente las partidas	-0,1 no carga correctamente las partidas
	-0,1 no carga el hueco	-0,1 no carga el hueco
Nota final (2,2)	1,6 / 2,2	1,6 / 2,2

Tabla 3-26 Evaluación del GPT personalizado del proyecto del grupo 25 del curso 2020-2021 con criterios propios del profesorado

La Tabla 3-26 muestra la comparación entre la evaluación realizada por el profesorado y la evaluación generada por el modelo GPT personalizado para el proyecto del grupo 25 del curso 2020–2021. Esta comparación permite analizar la capacidad del modelo no solo para generar código, sino también para evaluarlo de forma estructurada y coherente con las expectativas académicas.

En primer lugar, se observa una correspondencia completa entre ambas evaluaciones en cada uno de los parámetros evaluados: funcionamiento (0,8), mejoras implementadas (0,3), sangrado o indentación del código (0,1), uso de funciones (0,4) y comentarios (0). Esta coincidencia sugiere que el modelo ha logrado interpretar correctamente los criterios de la rúbrica y aplicarlos con precisión en su análisis del proyecto. La capacidad de replicar la misma valoración que el profesorado es indicativa de una alineación sólida entre el GPT personalizado y los estándares académicos definidos.

Asimismo, ambas evaluaciones incluyen las mismas deducciones en forma de aclaraciones negativas, relacionadas con dos errores detectados en el funcionamiento del proyecto: por un lado, el hecho de que no se cargan correctamente las partidas, y por otro, que no se carga el hueco. En ambos casos, se aplica una penalización de -0,1 puntos, lo que demuestra que el modelo es capaz de identificar fallos funcionales específicos de forma análoga a la revisión humana.

El resultado final, 1,6 sobre un máximo de 2,2 puntos, coincide también en ambas evaluaciones. Esta coincidencia refuerza la fiabilidad del modelo como evaluador automático y sugiere que puede ser utilizado como apoyo para realizar valoraciones preliminares o incluso como herramienta de corrección complementaria en entornos docentes. Cabe destacar que esta capacidad no solo aligera la carga de trabajo del profesorado, sino que también abre la posibilidad de proporcionar retroalimentación inmediata y coherente al alumnado.

3.5.4 Puesta en práctica del GPT personalizado como apoyo al profesorado en la supervisión de las restricciones de uso

Por último, con el fin de garantizar que el desarrollo del proyecto respete las limitaciones técnicas establecidas por la asignatura Informática para la Ingeniería, se solicitó al modelo GPT personalizado actuar como revisor del código. Su función fue analizar si el código fuente incluía alguno de los elementos no permitidos según las directrices del curso. Su respuesta se ve en la Tabla 3-27.

Elemento	¿Se usa en el código?	Comentario
Punteros	Sí	Sólo para pasar <i>int *x</i> y <i>int *y</i> por referencia. Esto sí que está permitido.
Memoria dinámica	No	No hay <i>malloc</i> ni <i>free</i> .
Struct (estructuras)	No	-
Typedef	No	-
Enum	No	-
Goto	No	-

Tabla 3-27 Respuesta del GPT personalizado a la revisión de elementos no permitidos en el proyecto del curso 2020-2021 de la asignatura Informática para la Ingeniería

Tras la revisión completa del código, se puede confirmar que el proyecto cumple rigurosamente con las restricciones establecidas. En particular, se ha verificado que los punteros utilizados en el programa se emplean exclusivamente en los casos permitidos: concretamente, para pasar parámetros por referencia a funciones. Este uso es coherente con los contenidos vistos en clase, y permite modificar valores desde funciones sin recurrir a estructuras avanzadas o técnicas fuera del temario.

No se ha detectado en el proyecto ninguna utilización inadecuada o fuera de contexto de los elementos del lenguaje, lo que garantiza que el desarrollo es compatible con el enfoque formativo de la asignatura y con los criterios de evaluación definidos por el profesorado.

Por tanto, se considera que este tipo de herramientas de IAG personalizadas pueden ser un complemento valioso para automatizar revisiones preliminares, reducir la carga de corrección del profesorado y fomentar una programación correcta, clara y ajustada al temario.

4 ANÁLISIS DE LOS RESULTADOS

En este capítulo se presentan los resultados obtenidos a lo largo del trabajo en relación con la evaluación de distintas Inteligencias Artificiales Generativas aplicadas al ámbito educativo, concretamente en el contexto de la asignatura Informática para la Ingeniería. En primer lugar, se expone una breve evaluación inicial de varias IAG gratuitas, centrada en su capacidad para generar y comprender diagramas de flujo, así como para elaborar código sencillo. A continuación, se resume el desempeño de cada IAG (en sus versiones gratuita y de pago) en la generación de código en C para proyectos complejos, como los desarrollados en el marco de la asignatura.

Posteriormente, se comparan las calificaciones asignadas por ChatGPT o3-mini-high (modelo escogido por su mejor rendimiento en las pruebas iniciales) tanto utilizando criterios propios como los definidos por el profesorado. También se incluye una comparativa directa entre ambas formas de evaluación, con el objetivo de valorar el grado de coincidencia y fiabilidad del modelo. Finalmente, se presenta un análisis específico del funcionamiento de un GPT personalizado, evaluado tanto desde la perspectiva del alumnado como desde su potencial como herramienta de apoyo a la labor docente.

4.1 Resumen del desempeño global de cada IAG en la evaluación inicial

A partir del análisis desarrollado en los apartados 3.1 y 3.2, se presentan a continuación los resultados obtenidos. En primer lugar, se examina el rendimiento de las Inteligencias Artificiales Generativas en tareas de comprensión y generación de diagramas de flujo. Posteriormente, se evalúa su desempeño en la generación de código de baja complejidad. Este estudio se ha planteado con el objetivo de explorar el potencial de estas herramientas como sistemas de apoyo en el proceso de aprendizaje, considerando su viabilidad como tutores virtuales capaces de asistir al alumnado en la adquisición de competencias básicas en programación.

4.1.1 Desempeño de cada IAG gratuita en comprensión y generación de diagramas de flujo

En la fase inicial del trabajo se evaluó la capacidad de distintas IAG, en su versión gratuita, para resolver ejercicios de diagramas de flujo, siguiendo la misma progresión que los estudiantes en la asignatura. Los resultados mostrados en la Figura 4-1 evidencian que Gemini fue con diferencia la herramienta menos competente, mostrando muchas dificultades para interpretar correctamente la lógica de los diagramas, incluso en ejercicios sencillos. En cambio, DeepSeek sorprendió gratamente al ser la única IAG capaz de resolver paso a paso un diagrama de flujo complejo sin errores, posicionándose como la más eficaz para este tipo de tareas analíticas.

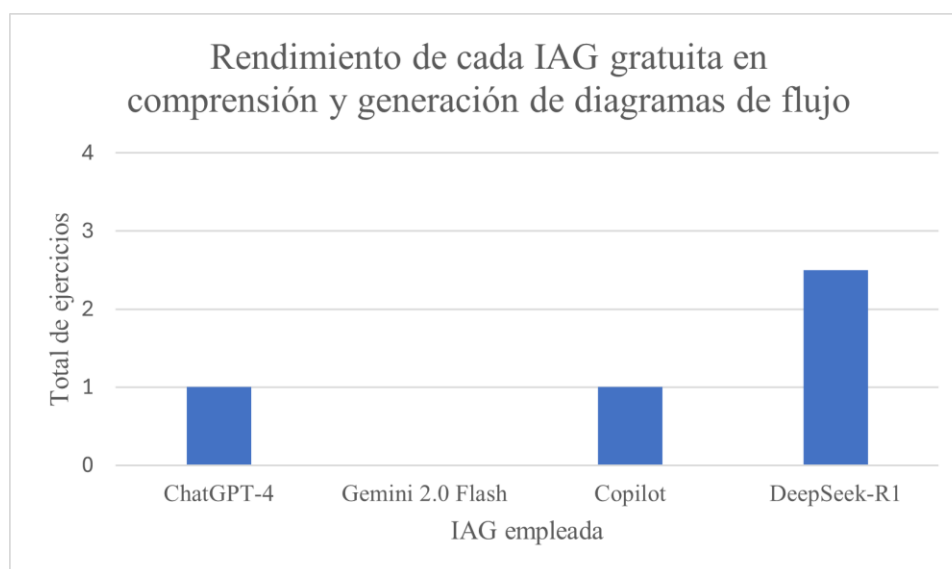


Figura 4-1 Rendimiento de cada IAG en su versión gratuita en comprensión y generación de diagramas de flujo

Por otro lado, tanto Copilot como ChatGPT ofrecieron un rendimiento sólido en los ejercicios básicos, proporcionando explicaciones claras y funcionales y detectando errores comunes. Demuestran una cierta capacidad para abordar la generación de diagramas de flujo, si bien con menor precisión y consistencia.

Estos resultados refuerzan la idea de que la Inteligencia Artificial Generativa puede ser utilizada como un recurso pedagógico complementario, actuando como una especie de “tutor digital” que acompaña al alumno en su proceso de aprendizaje. No obstante, también se ha demostrado que ninguna de las IAG es infalible, por lo que es imprescindible que tanto el alumnado como el profesorado mantengan una actitud crítica y supervisen siempre los resultados generados, especialmente cuando se trata de problemas con múltiples interpretaciones o estructuras lógicas más complejas.

4.1.2 Desempeño de cada IAG de pago en comprensión y generación de diagramas de flujo

La Figura 4-2 muestra el rendimiento global de tres IAG, en sus versiones de pago, en tareas de comprensión y generación de diagramas de flujo.

ChatGPT-o3-mini-high destaca con el máximo rendimiento, resolviendo correctamente los cuatro ejercicios planteados. Copilot Pro sigue con tres aciertos, mostrando buen desempeño, aunque no completamente consistente. Por último, Gemini Advanced queda rezagado con solo dos ejercicios resueltos, lo que indica dificultades especialmente en las tareas de generación a partir de enunciados breves.

En conjunto, estos resultados evidencian que, incluso dentro del segmento de IAG de pago, existen diferencias significativas en el rendimiento según el tipo de tarea. Mientras que la comprensión de diagramas parece estar al alcance de todas las herramientas analizadas, la generación a partir de enunciados breves supone un reto mayor que no todas las IAG superan con la misma solvencia.

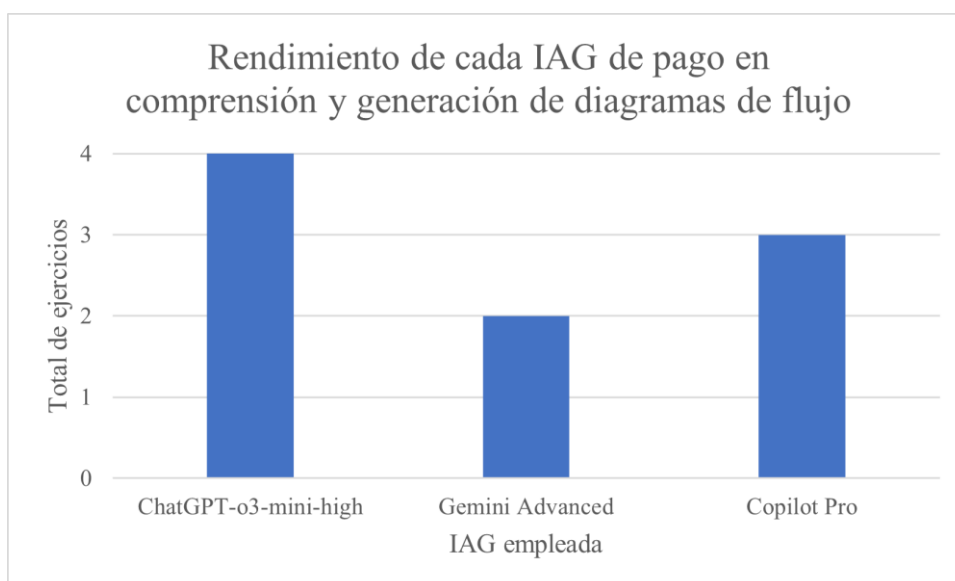


Figura 4-2 Rendimiento de cada IAG en su versión de pago en comprensión y generación de diagramas de flujo

4.1.3 Desempeño de cada IAG gratuita en generación de código simple

La Figura 4-3 presenta una comparativa del rendimiento de las cuatro Inteligencias Artificiales Generativas previamente seleccionadas en la resolución de ejercicios de programación de nivel básico. El criterio de evaluación empleado ha sido el número total de ejercicios completados correctamente por cada modelo, lo que permite valorar su eficacia en tareas de generación de código simple. Esta métrica ofrece una visión clara del grado de competencia técnica que cada herramienta puede alcanzar en la resolución de problemas de nivel introductorio.

Entre los modelos analizados, ChatGPT-4 destaca con un total de seis ejercicios resueltos de manera satisfactoria, posicionándose como la IAG con mejor desempeño en esta categoría. Le siguen de cerca GitHub Copilot y DeepSeek-R1, ambos con cinco ejercicios completados, lo que indica un rendimiento también elevado y consistente. Por otro lado, Gemini 2.0 Flash registra un resultado inferior, con tan solo tres ejercicios correctamente generados, lo cual evidencia una menor precisión o capacidad de adaptación a los requerimientos planteados en este tipo de tareas.

Estos resultados reflejan la potencial utilidad de las IAG como herramientas de apoyo en entornos de enseñanza de la programación, especialmente en niveles iniciales. No obstante, también ponen de manifiesto diferencias notables entre modelos, lo que resalta la necesidad de una selección cuidadosa de la herramienta en función del contexto y los objetivos educativos. La superioridad de ChatGPT-4 en este ámbito sugiere una mayor consistencia y fiabilidad.

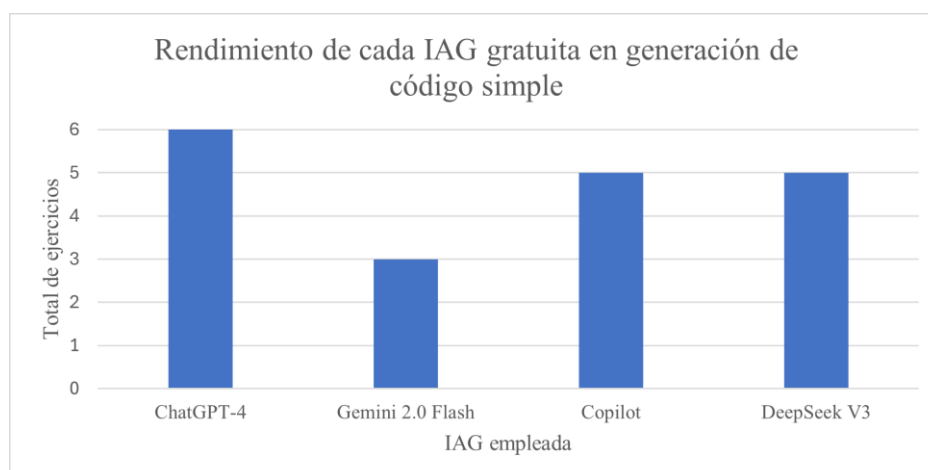


Figura 4-3 Desempeño de cada IAG en su versión gratuita en generación de código simple

4.1.4 Desempeño de cada IAG de pago en generación de código simple

La última fase de la evaluación inicial se centró en la capacidad de las IAG para resolver ejercicios sencillos de creación de código, con un total de seis problemas propuestos. En esta etapa, ChatGPT-o3-mini-high destacó, resolviendo todos los ejercicios sin errores, lo que demuestra una alta precisión y fiabilidad en generación de código funcional. Copilot Pro también mostró buen desempeño, con cinco ejercicios sin errores y un ejercicio no resuelto. En cambio, Gemini Advanced resolvió cuatro ejercicios correctamente, pero cometió errores en los dos restantes, evidenciando una menor estabilidad incluso en tareas de baja complejidad, como se ve en la Figura 4-4.

En conjunto, estos resultados permiten concluir que, aunque todas las herramientas presentan un cierto grado de competencia, existen diferencias claras en cuanto a precisión y fiabilidad, siendo ChatGPT-o3-mini-high la que ofrece un rendimiento más sólido y constante en los distintos tipos de tareas evaluadas. Además, se confirma que las versiones de pago de estas IAG ofrecen una precisión considerablemente superior a la observada en modelos gratuitos, especialmente en tareas que requieren la generación de código.

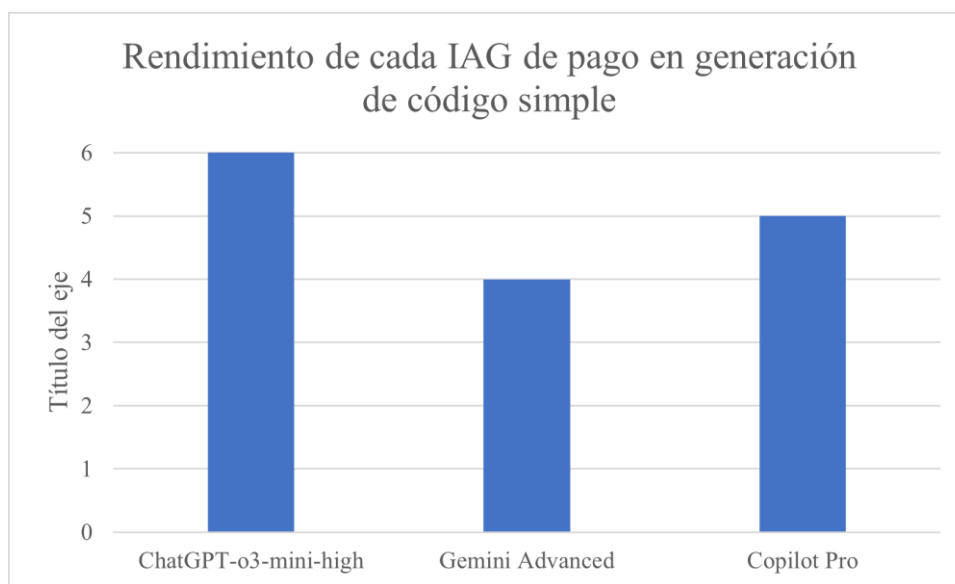


Figura 4-4 Desempeño de cada IAG en su versión de pago en generación de código simple

4.2 Resumen del desempeño de cada IAG en creación de código para proyectos complejos

La evaluación del desempeño de diversas Inteligencias Artificiales Generativas en la resolución de tres proyectos de programación ha permitido identificar diferencias significativas en términos de precisión, fiabilidad y tasa de éxito de las soluciones generadas. El análisis de los resultados se ha basado en tres categorías principales: generación de respuestas correctas sin errores, generación de respuestas parcialmente correctas con errores y la incapacidad de producir una solución funcional; tal y como se ve en la Figura 4-5.

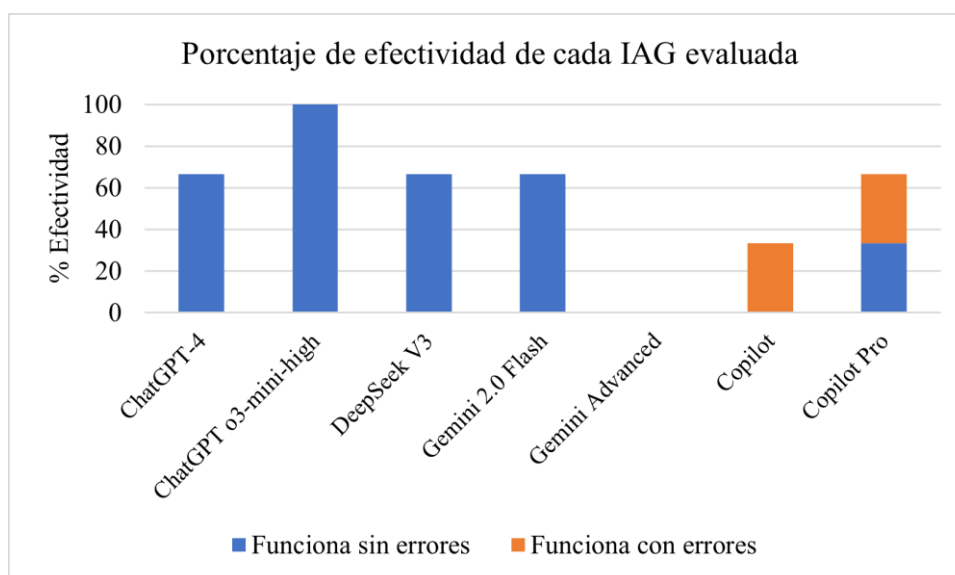


Figura 4-5 Porcentaje de eficacia de cada IAG en los tres proyectos de programación evaluados

Entre los modelos evaluados, ChatGPT o3-mini-high ha sido el único que no ha registrado errores en ninguno de los casos, mostrando un desempeño impecable con una tasa del 100% de respuestas correctas. Esto lo posiciona como la opción más fiable y rentable para la ejecución de proyectos de programación, al ofrecer soluciones generalmente precisas que, en la mayoría de los casos, no requieren intervención humana para la corrección de errores. Su capacidad para generar código completamente funcional en todos los casos analizados lo posiciona como una alternativa óptima para entornos donde la eficiencia y la necesidad de minimizar el tiempo dedicado a la revisión manual son prioritarias, especialmente en contextos donde el alumno dispone de un margen de tiempo limitado para corregir y depurar errores.

Por su parte, ChatGPT-4, DeepSeek V3 y Gemini 2.0 Flash (introduciendo el prompt en *LaTeX*) han demostrado ser herramientas altamente consistentes, con una alta proporción de respuestas correctas. Sin embargo, a diferencia de ChatGPT o3-mini-high, estos modelos han presentado ciertos casos en los que no han logrado generar soluciones funcionales, lo que los hace menos fiables. La ausencia de respuestas con errores parciales en estos modelos sugiere que su desempeño tiende a ser binario: o generan una respuesta totalmente correcta o fallan en producir una solución útil.

En el extremo opuesto, Gemini Advanced, a pesar de ser la versión de pago de Gemini, ha mostrado el peor desempeño de todas las IAG evaluadas, con una tasa de fallos del 100%, lo que indica que no ha sido capaz de generar soluciones funcionales en ninguno de los casos. Resulta especialmente llamativo que su rendimiento sea significativamente inferior al de su versión gratuita, lo que plantea serias dudas sobre su viabilidad en tareas de generación de código. Su uso no solo no aporta ventajas en términos de automatización, sino que, al requerir una corrección manual completa en todos los casos, supone una opción poco eficiente en comparación con alternativas gratuitas.

Los modelos Copilot y Copilot Pro han ofrecido un rendimiento más variable. Copilot estándar ha registrado una alta proporción de respuestas incorrectas, lo que lo hace menos confiable en la generación de código funcional. Copilot Pro, en cambio, ha demostrado ser más versátil, con una mayor cantidad de respuestas parcialmente correctas, aunque sigue presentando una proporción significativa de errores. Si bien este modelo puede ser útil en contextos donde exista la posibilidad de corregir manualmente los errores, su uso en entornos que requieran precisión sin intervención humana no resulta ideal.

En términos generales, los resultados obtenidos evidencian que ChatGPT o3-mini-high es la opción más eficiente y rentable para la ejecución de proyectos de programación, ya que es el único modelo que ha conseguido una tasa del 100% de aciertos sin errores. Mientras que otros modelos como ChatGPT-4 y DeepSeek V3 han demostrado un alto nivel de precisión, siguen presentando ciertas limitaciones en

su fiabilidad absoluta. En el caso de Copilot Pro, aunque su flexibilidad puede ser una ventaja en algunos escenarios, su tendencia a generar respuestas con errores parciales lo hace menos recomendable en proyectos donde se requiera precisión sin supervisión.

Estos hallazgos ponen de manifiesto la importancia de seleccionar cuidadosamente la IAG más adecuada en función del contexto de aplicación. Si el objetivo es obtener respuestas precisas sin intervención manual, ChatGPT o3-mini-high es la mejor alternativa. No obstante, si se dispone de margen para correcciones, herramientas como Copilot Pro pueden ofrecer soluciones parcialmente útiles. Por otro lado, Gemini Advanced se muestra como la opción menos eficiente en este estudio, con una tasa de fallos absoluta.

La presente evaluación sugiere que, aunque la evolución de la Inteligencia Artificial Generativa ha permitido avances significativos en la generación automática de código, existen diferencias sustanciales en la fiabilidad y el rendimiento de los distintos modelos, lo que debe ser considerado en el momento de su implementación en entornos profesionales o académicos.

4.2.1 Comparación con los resultados del benchmark SWE-bench Verified

Con el fin de obtener una visión más precisa y contextualizada de los resultados de la Figura 4-5, se procedió a comparar los datos obtenidos experimentalmente con los ofrecidos por el *benchmark* en el apartado 2.3.3. Esta comparación permitirá contrastar el rendimiento observado en pruebas prácticas con datos validados en escenarios reales de desarrollo de software, lo que enriquecerá el análisis global y permitirá valorar de manera más objetiva la aplicabilidad y fiabilidad de cada IAG en entornos educativos y profesionales.

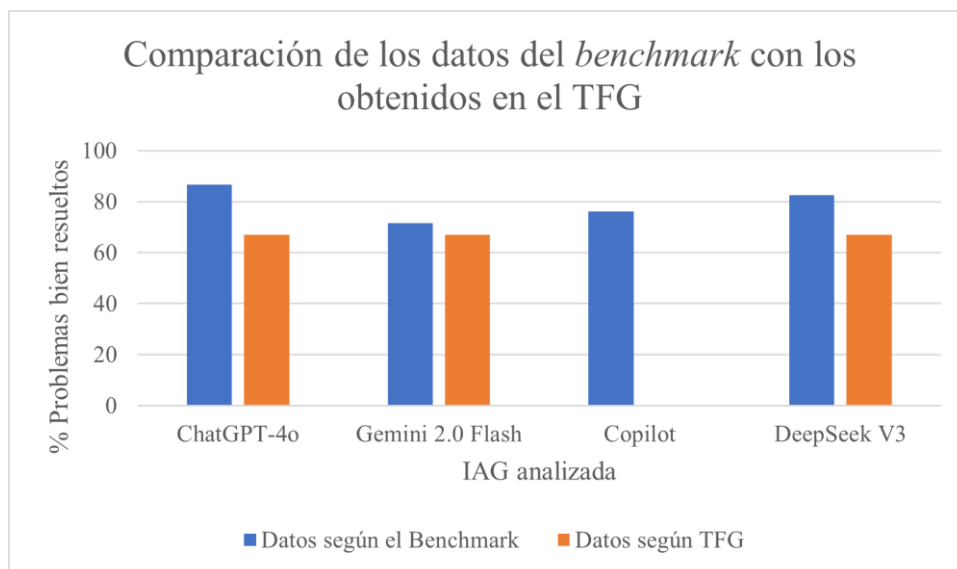


Figura 4-6 Comparación entre los datos del *benchmark* con los obtenidos en el TFG

La Figura 4-6 muestra una comparativa entre los porcentajes de problemas correctamente resueltos por distintas IAG, según dos fuentes: los datos proporcionados por el *benchmark SWE-bench Verified* y los datos obtenidos a partir del análisis experimental realizado en el presente TFG. En todos los casos se observa una mayor eficacia en los resultados reportados por el *benchmark* en comparación con los valores obtenidos en el entorno de evaluación desarrollado para este trabajo.

Esta diferencia puede explicarse por diversos factores. En primer lugar, los entornos de prueba del *benchmark* están altamente optimizados, tanto en la definición de los *prompts* como en la preparación del contexto proporcionado al modelo, lo que facilita una comprensión más precisa del problema por parte de la IAG. Por el contrario, en el TFG se ha intentado reproducir un entorno más realista y próximo al contexto educativo, en el que las instrucciones pueden ser más abiertas o menos estructuradas, lo que incrementa la probabilidad de errores o malinterpretaciones por parte del modelo.

Además, los métodos de evaluación también pueden diferir. Mientras que en el *benchmark* se consideran como válidas aquellas soluciones que reproducen funcionalmente el comportamiento esperado (incluso si no son exactamente iguales al parche real), en el TFG se ha sido más estricto, evaluando también aspectos como el uso correcto de estructuras, la legibilidad del código o el cumplimiento total del enunciado planteado por el profesorado. Esto puede haber reducido ligeramente las tasas de éxito observadas.

Por último, cabe señalar que el *benchmark* utiliza códigos cortos y sencillos, como se ve en la Figura 4-7. Esto realza las diferencias existentes entre el análisis llevado a cabo en el TFG, en el cual se utilizaron proyectos de programación complejos, y los datos proporcionados por el *benchmark*.

```
Unset
kernS: 'kern' referenced before assignment
from sympy.core.sympify import kernS

text = "(2*x)/(x-1)"
expr = kernS(text)
// hit = kern in s
// UnboundLocalError: local variable 'kern' referenced before assignment
```

Figura 4-7 Ejemplo de enunciado de problema [52]

En conjunto, esta comparativa pone de manifiesto que, aunque las IAG analizadas muestran un alto potencial, su rendimiento puede variar notablemente dependiendo del contexto y de cómo se les planteen los problemas, lo cual refuerza la importancia de evaluar estas tecnologías en escenarios educativos reales como el aquí presentado.

4.3 Calificaciones de ChatGPT o3-mini-high con criterios propios frente a las del profesorado

En la Figura 4-8 se compara la calificación otorgada por el docente con la generada por el modelo de Inteligencia Artificial Generativa ChatGPT o3-mini-high entre los distintos grupos de alumnos de la asignatura. El objetivo de este análisis era evaluar la precisión de la IAG en la asignación de calificaciones y su posible aplicabilidad como herramienta de apoyo en la corrección de trabajos académicos.

Los resultados evidencian una tendencia general en la que las calificaciones asignadas por la IAG siguen un comportamiento similar a las del profesorado. En la mayoría de los casos, ambas líneas presentan fluctuaciones similares, lo que indica un alto grado de coherencia entre la evaluación automatizada y la humana. No obstante, existen puntos específicos en los que se observan discrepancias, como en los grupos 19, 25, y 26 donde la calificación docente es superior a la de la IAG, o en los grupos 5, 11, y 14 donde ocurre lo contrario. Estas diferencias podrían deberse a la subjetividad inherente a la evaluación humana o a diferencias en la ponderación de ciertos criterios por parte del modelo de IAG.

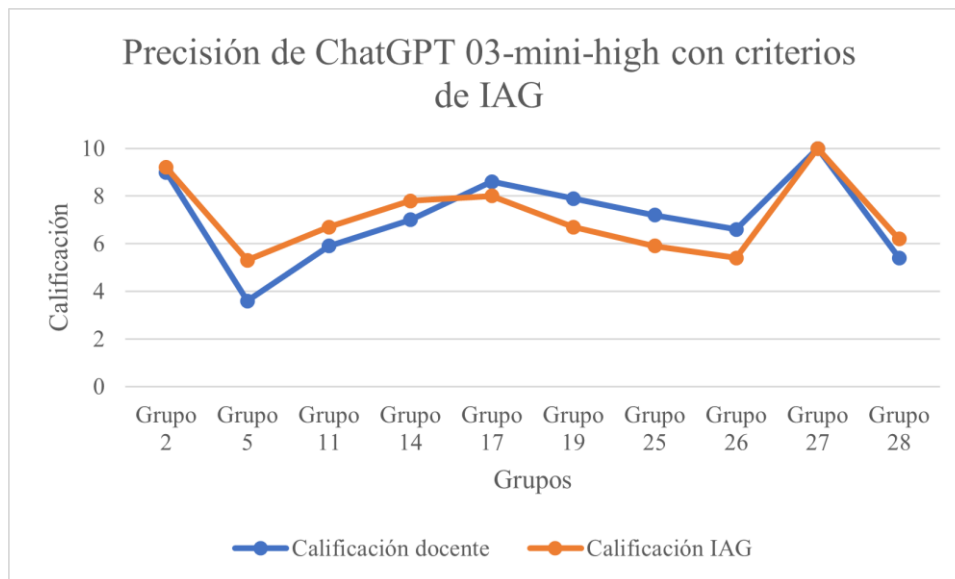


Figura 4-8 Precisión de ChatGPT o3-mini-high con criterios propios

Uno de los aspectos más destacables del análisis es la consistencia del criterio de la IAG, ya que sus evaluaciones tienden a mantener una correlación estrecha con las del docente. Esta característica sugiere que la IAG puede ser una herramienta de gran utilidad para el profesorado, sirviendo como apoyo en la evaluación de trabajos académicos y reduciendo la carga de corrección manual. Al contar con un sistema capaz de calificar de manera objetiva y uniforme, los docentes pueden emplear la IAG como un primer filtro de evaluación, permitiendo centrarse en aspectos más cualitativos o subjetivos del desempeño de los estudiantes.

Además, la implementación de este tipo de tecnología en el ámbito educativo podría contribuir a una mayor transparencia y equidad en la calificación, minimizando posibles sesgos humanos y asegurando una valoración más estandarizada. Si bien no se sugiere una automatización completa del proceso de evaluación, la combinación del criterio docente con el soporte de una IAG permitiría mejorar la eficiencia en la corrección, optimizar el tiempo dedicado a la retroalimentación y reforzar la imparcialidad en la calificación.

En conclusión, los resultados reflejados en la gráfica ponen de manifiesto el potencial de la IAG como un recurso complementario en la evaluación académica. La alta correspondencia entre las calificaciones docentes y las generadas por el LLM sugiere que esta tecnología puede desempeñar un papel clave en la modernización del proceso de corrección, facilitando una evaluación más rápida y objetiva sin reemplazar la intervención del profesorado. Se recomienda, por tanto, explorar su integración en entornos educativos como un recurso valioso para la optimización de la enseñanza y el aprendizaje.

4.4 Calificaciones de ChatGPT o3-mini-high con criterios propios del profesorado frente a las del profesorado

Con el propósito de evaluar la precisión y la utilidad de la Inteligencia Artificial Generativa en procesos de corrección académica, se diseñó una metodología basada en el análisis desagregado de los distintos criterios de evaluación. Para ello, se generaron representaciones gráficas individuales que comparan las calificaciones otorgadas por el profesorado con aquellas generadas por la IAG, aplicando los mismos criterios de evaluación definidos previamente por el docente en la Tabla 3-10, Tabla 3-11 y Tabla 3-12.

Este enfoque permite identificar con mayor claridad en qué aspectos la IAG presenta un comportamiento fiable y alineado con el juicio humano, y en qué casos pueden detectarse desviaciones significativas. De este modo, se puede establecer un diagnóstico detallado de su rendimiento,

diferenciando las áreas donde su uso puede automatizarse con seguridad de aquellas que aún requieren una supervisión más estrecha por parte del profesorado.

Por un lado, el análisis por separado de cada criterio ofrece una herramienta rigurosa para valorar la integración progresiva de la IAG en tareas de evaluación educativa, maximizando sus ventajas sin comprometer la calidad ni la equidad del proceso evaluador. A continuación, se presentan los resultados correspondientes a cada uno de los criterios con el fin de ilustrar gráficamente el grado de concordancia alcanzado.

La Figura 4-9 muestra una comparación entre las calificaciones asignadas por el profesorado y las generadas por ChatGPT o3-mini-high en relación con el funcionamiento de los distintos grupos. En general, se observa una alta similitud entre ambas líneas, lo que indica una concordancia significativa en la evaluación de este criterio. Las diferencias son mínimas y, en la mayoría de los casos, las puntuaciones se encuentran muy próximas entre sí, lo cual refuerza la idea de que la IAG puede aplicar con precisión criterios previamente definidos por el profesorado. Este comportamiento consistente sugiere que la IAG es una herramienta válida para apoyar el proceso evaluativo, especialmente en indicadores técnicos y específicos como el funcionamiento del proyecto.

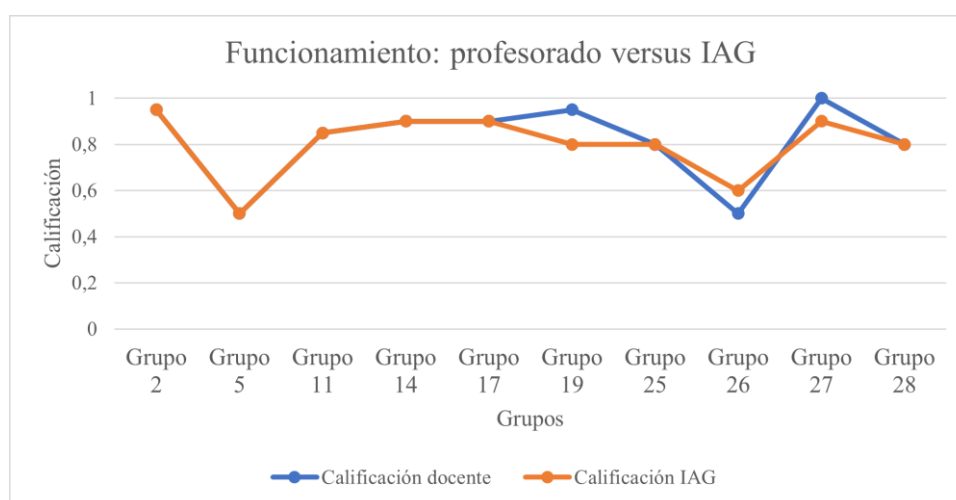


Figura 4-9 Precisión en las calificaciones de ChatGPT o3-mini-high sobre el funcionamiento

La Figura 4-10 muestra una comparativa entre las calificaciones asignadas por el profesorado y las generadas por Inteligencia Artificial Generativa en relación con el criterio de mejoras introducidas en los proyectos. En general, se observa una tendencia similar entre ambas fuentes de evaluación, aunque con ligeras discrepancias en algunos grupos. La IAG tiende a otorgar puntuaciones ligeramente superiores en los casos donde se detectan mejoras menores, mientras que el profesorado muestra una evaluación algo más conservadora. A pesar de estas diferencias puntuales, el comportamiento global refleja una correlación aceptable, lo que indica que la IAG es capaz de identificar adecuadamente la existencia de mejoras en los trabajos analizados. No obstante, dado que se trata de un criterio que puede implicar matices subjetivos —como la relevancia o el impacto de las mejoras realizadas—, este resultado sugiere la conveniencia de que el profesorado supervise este apartado de forma más directa para garantizar una valoración completamente ajustada al contexto académico.

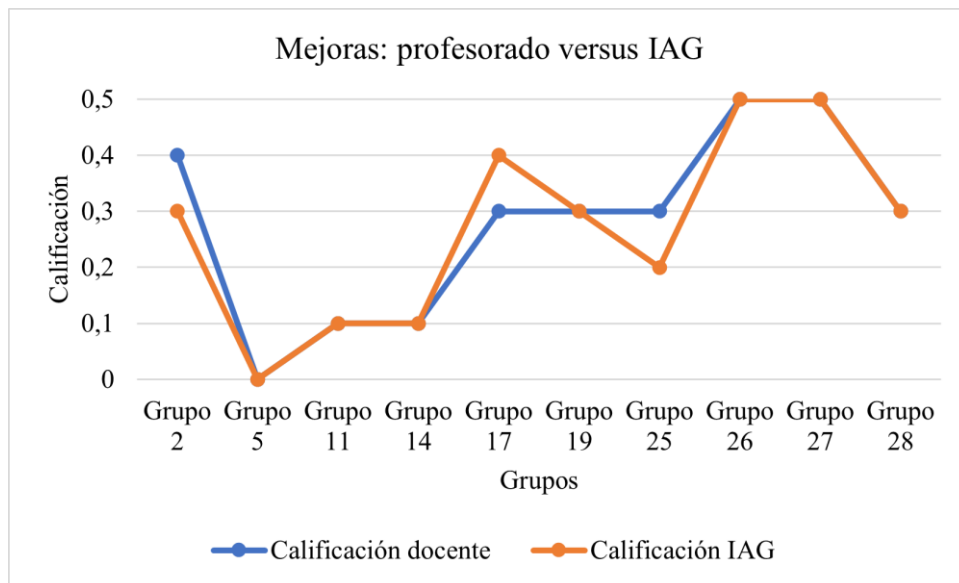


Figura 4-10 Precisión en las calificaciones de ChatGPT 03-mini-high sobre las mejoras

La Figura 4-11 revela cierta discordancia entre las calificaciones asignadas por el profesorado y las generadas por Inteligencia Artificial Generativa, lo que pone de manifiesto que, aunque el sangrado pueda parecer un criterio puramente técnico, en la práctica incorpora cierto grado de subjetividad. La dificultad radica en establecer con claridad el umbral entre un sangrado correcto, uno aceptable y uno deficiente, lo que da lugar a interpretaciones diversas por parte de los evaluadores humanos y del sistema automatizado. Mientras la IAG tiende a aplicar criterios más uniformes, el profesorado puede valorar aspectos como la consistencia, la intención del diseño o el impacto visual en la legibilidad del código. Esta falta de alineación indica que, pese a tratarse de un aspecto aparentemente objetivo, el criterio de sangrado requiere una supervisión docente más cuidadosa si se desea mantener una evaluación justa y contextualizada. De todas formas, este apartado tiene poco peso evaluativo, por lo que el impacto en la nota final es mínimo.

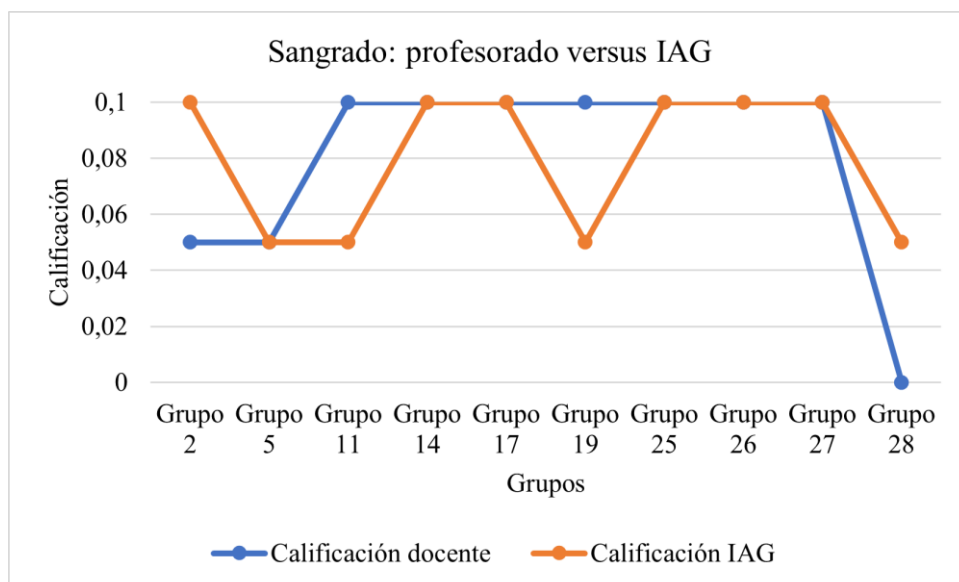


Figura 4-11 Precisión en las calificaciones de ChatGPT 03-mini-high sobre el sangrado

La Figura 4-12 muestra una comparación entre las calificaciones otorgadas por el profesorado y las generadas por Inteligencia Artificial Generativa en relación con el uso adecuado de funciones en los proyectos presentados. En líneas generales, ambas curvas mantienen una tendencia similar, especialmente en los grupos con puntuaciones medias y altas, lo que indica una buena correspondencia

en la mayoría de los casos. Sin embargo, también se aprecian diferencias puntuales, como en los grupos 5, 11, 19 y 25, donde la IAG tiende a asignar puntuaciones más bajas que el profesorado. Esto sugiere que, aunque la IAG es capaz de identificar la presencia y estructura de funciones, puede no captar con la misma profundidad aspectos cualitativos como la adecuación del diseño funcional, la reutilización de funciones o la claridad de las mismas. Por tanto, aunque el criterio de funciones presenta un grado razonable de objetividad técnica, conviene que el docente revise estos casos para asegurar una evaluación contextualizada y pedagógicamente fundamentada.

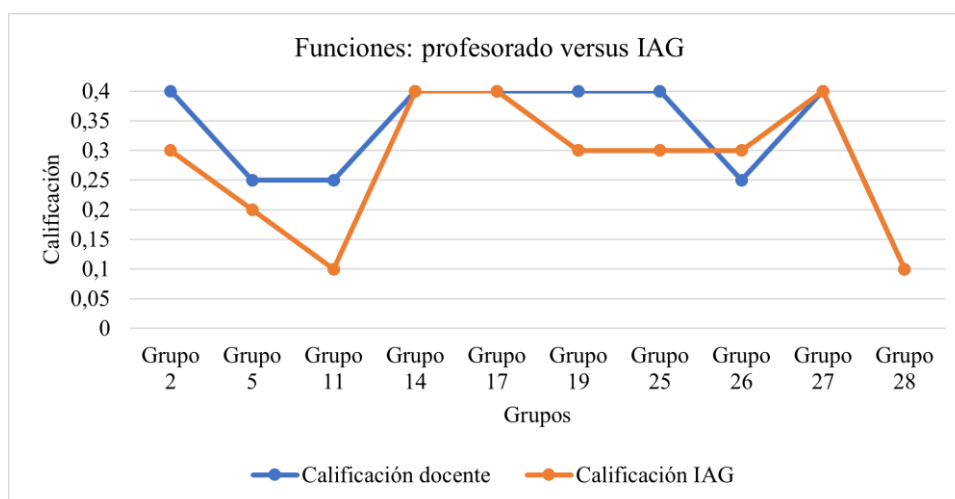


Figura 4-12 Precisión en las calificaciones de ChatGPT o3-mini-high sobre las funciones

La Figura 4-13 pone de manifiesto una discrepancia significativa entre las calificaciones asignadas por el profesorado y las generadas por esta Inteligencia Artificial Generativa en relación con la calidad y pertinencia de los comentarios en el código. En la mayoría de los grupos, la IAG tiende a otorgar puntuaciones sistemáticamente más altas que el profesorado, lo que sugiere que interpreta la mera presencia de comentarios como un criterio suficiente para una evaluación positiva. Sin embargo, este criterio es altamente subjetivo, ya que no solo se valora la existencia de comentarios, sino también su utilidad, precisión, claridad y relación con el contenido del código. La IAG, al carecer de un contexto pedagógico detallado, puede no ser capaz de distinguir entre comentarios superficiales, redundantes o mal formulados y aquellos que realmente aportan valor. Por tanto, este apartado requiere una intervención directa del profesorado, ya que la valoración cualitativa de los comentarios implica una comprensión contextual que, al menos por el momento, escapa a las capacidades de evaluación automatizada con el nivel de precisión deseable.

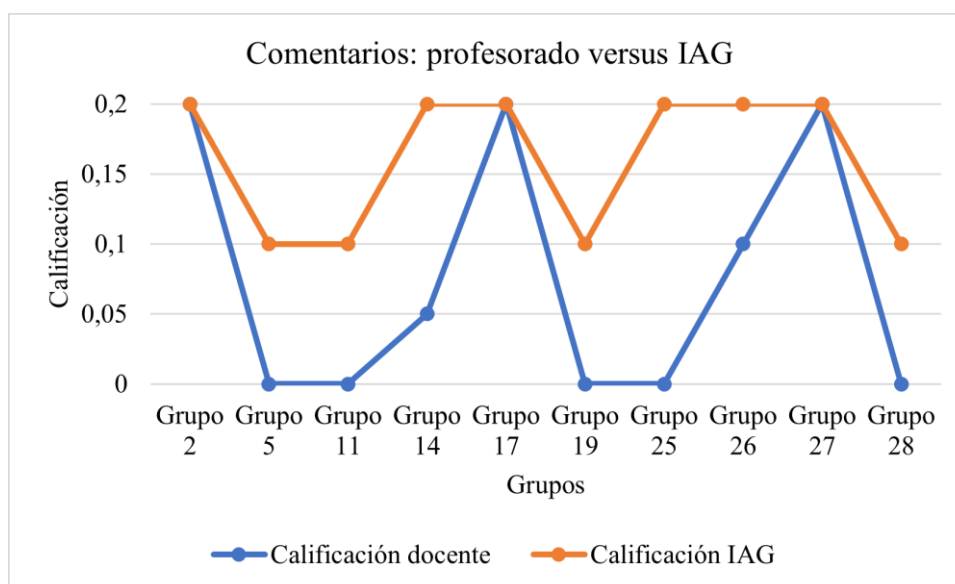


Figura 4-13 Precisión en las calificaciones de ChatGPT o3-mini-high sobre los comentarios

Por otro lado, la Figura 4-14 muestra la calificación total con todos los criterios unificados. Como se observa, la línea correspondiente a la "Calificación IAG" sigue una tendencia muy similar a la de la "Calificación docente", con variaciones mínimas en la mayoría de los casos. Esta proximidad en los resultados evidencia una alta concordancia entre ambas fuentes de evaluación, lo que sugiere que el sistema es capaz de interpretar y aplicar los criterios establecidos con notable precisión.

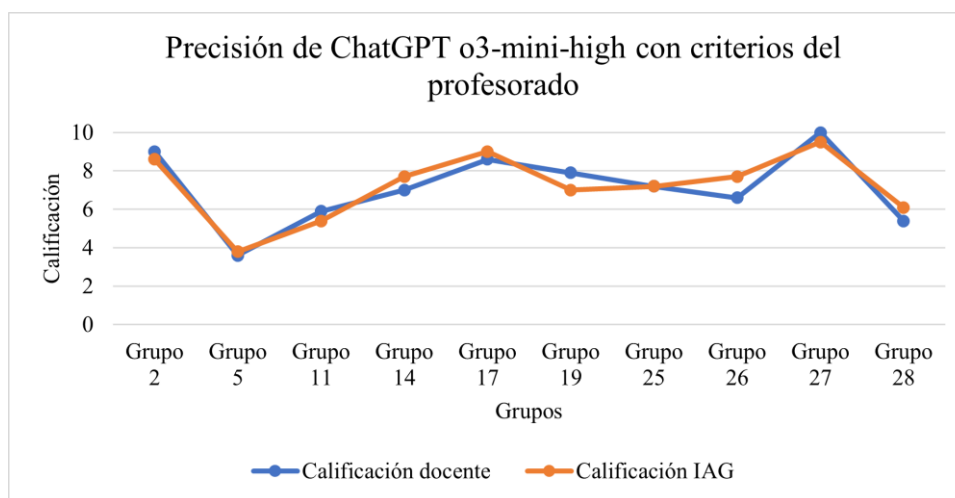


Figura 4-14 Precisión de ChatGPT o3-mini-high con criterios del profesorado

En definitiva, el uso de Inteligencia Artificial Generativa como apoyo en la docencia representa una oportunidad valiosa para modernizar y optimizar procesos docentes, especialmente en tareas repetitivas como la corrección de trabajos. Como se ha evidenciado en los distintos gráficos comparativos, la IAG puede ofrecer resultados notablemente precisos cuando se le proporcionan criterios claros y objetivos, lo que permite al profesorado ahorrar tiempo y centrar sus esfuerzos en aspectos más complejos del aprendizaje. Integrar estas tecnologías en el ámbito educativo no solo responde a una necesidad de actualización ante los avances tecnológicos, sino que también abre la puerta a prácticas más eficientes, escalables y consistentes.

No obstante, es fundamental que este apoyo se utilice con cautela. La IAG, aunque potente, no sustituye la sensibilidad pedagógica ni el juicio contextual del profesorado. En criterios más subjetivos, como la calidad de los comentarios, el estilo del código o la relevancia de las mejoras, la Inteligencia Artificial puede ofrecer valoraciones razonables, pero no necesariamente acertadas o alineadas con las

expectativas docentes. Por ello, es imprescindible que el profesorado mantenga un rol activo y supervise los resultados generados por la IAG, validando las evaluaciones y corrigiendo posibles desviaciones. De este modo, se logra un equilibrio entre la innovación tecnológica y la garantía de una evaluación justa, crítica y adaptada al contexto educativo real.

4.5 Comparativa entre las calificaciones de ChatGPT o3-mini-high con criterios propios y con criterios del profesorado

La Figura 4-15 muestra una comparativa entre las calificaciones asignadas por el profesorado y las generadas por el modelo ChatGPT o3-mini-high bajo dos condiciones distintas: utilizando sus propios criterios y aplicando los criterios de evaluación definidos por el profesorado. El análisis se realiza sobre distintos grupos de estudiantes y permite observar el grado de alineación entre las evaluaciones humanas y automáticas en función del tipo de orientación proporcionada a la IAG.

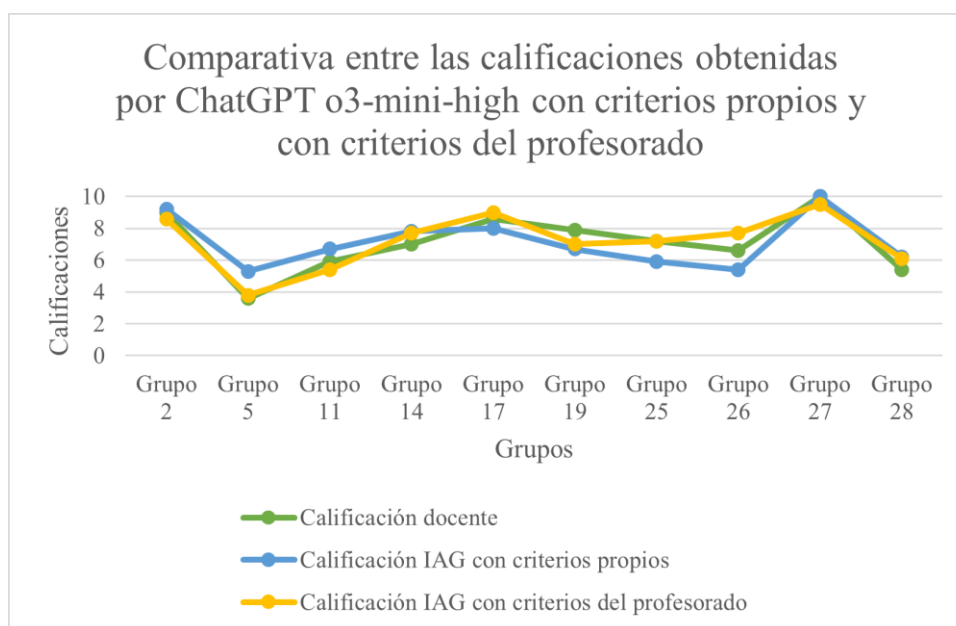


Figura 4-15 Comparativa entre las calificaciones obtenidas por ChatGPT o3-mini-high con criterios propios y con criterios del profesorado

Se aprecia con claridad que, cuando el modelo opera siguiendo los criterios del profesorado (línea amarilla), sus calificaciones tienden a aproximarse notablemente a las otorgadas por los docentes (línea verde), mostrando una mayor exactitud y coherencia. En cambio, al utilizar sus propios criterios (línea azul), se producen desviaciones un poco más pronunciadas respecto a las valoraciones docentes. Este resultado pone de manifiesto la capacidad del modelo para adaptarse con gran precisión cuando se le proporciona una guía clara y estructurada, lo que refuerza su potencial como herramienta de apoyo en tareas de evaluación automatizada siempre que se le alimenten criterios específicos.

Con el fin de analizar de forma más precisa el grado de coincidencia o discrepancia entre estas tres gráficas, se procedió al cálculo de la desviación porcentual absoluta media (MAPE). Este indicador permite cuantificar, en términos porcentuales, cuánto se aleja cada una de las calificaciones generadas por la IAG respecto a la calificación considerada como referencia, en este caso la del profesorado. Por lo tanto, cuanto menor sea el valor MAPE obtenido, más precisa será la calificación en cuestión.

La utilización del MAPE proporciona una medida objetiva de la variabilidad entre los métodos de evaluación y resulta especialmente útil para valorar la fiabilidad del modelo como herramienta de apoyo en la corrección automatizada de trabajos académicos. La fórmula es $MAPE = 100 \frac{1}{n} \sum_{t=1}^n \left| \frac{A_t - F_t}{A_t} \right|$. Donde A_t es la calificación docente y F_t es la calificación de la IAG. Su diferencia se divide entre el valor de la calificación docente A_t . El valor absoluto de esta operación se suma para cada evaluación y

se divide entre el número de evaluaciones totales n . Los resultados obtenidos son de 14,77% para la calificación de la IAG con criterios propios y de 7,9% para la calificación de la IAG con criterios del profesorado. Esto refuerza los resultados observados en la Figura 4-15.

4.6 Efectividad del GPT personalizado

A lo largo del desarrollo del trabajo, se ha evaluado el uso de un modelo GPT personalizado como herramienta de apoyo para la generación y análisis de código en el contexto de la asignatura Informática para la Ingeniería.

En primer lugar, cabe destacar que el modelo mostró una notable capacidad para relacionar de forma precisa y acertada los conceptos presentes en los proyectos con las secciones correspondientes del temario de la asignatura, lo que facilitó enormemente la comprensión teórica por parte del estudiante. Esta vinculación entre práctica y teoría constituye un recurso pedagógico valioso, especialmente en asignaturas de carácter técnico.

En segundo lugar, se entregaron al modelo tres proyectos de cursos anteriores para su análisis. El modelo fue capaz de identificar errores de funcionamiento, proponer mejoras en cuanto a modularidad, claridad del código y calidad visual, y valorar el cumplimiento de los criterios de evaluación definidos por el profesorado. Este análisis pone de manifiesto la utilidad del GPT personalizado como herramienta de apoyo en la generación de código, especialmente cuando se combina con una supervisión activa por parte del alumno, que permite detectar y corregir posibles errores de interpretación o implementación.

En conjunto, su aplicación en contextos educativos se presenta como una herramienta prometedora, que puede reforzar el aprendizaje autónomo del alumnado, facilitar la comprensión de conceptos de programación y fomentar la mejora progresiva del código mediante ciclos iterativos de prueba y corrección. Su uso, complementado con la guía del profesorado, puede potenciar el desarrollo de competencias clave como la lectura crítica de código, la depuración estructurada y la reflexión sobre las decisiones de diseño.

Para analizar de forma más clara su rendimiento, se seleccionó la Figura 4-5 y se le añadió la efectividad del GPT personalizado. Los resultados en la Figura 4-16 muestran que, de forma indiscutible, los dos modelos más efectivos han sido ChatGPT o3-mini-high y el GPT personalizado ya que son los únicos capaces de generar soluciones, en principio, completamente correctas, pese a que haya que refinar los *prompt* con varias iteraciones. Esto los posiciona como los claros dominantes frente al resto de IAG evaluadas.

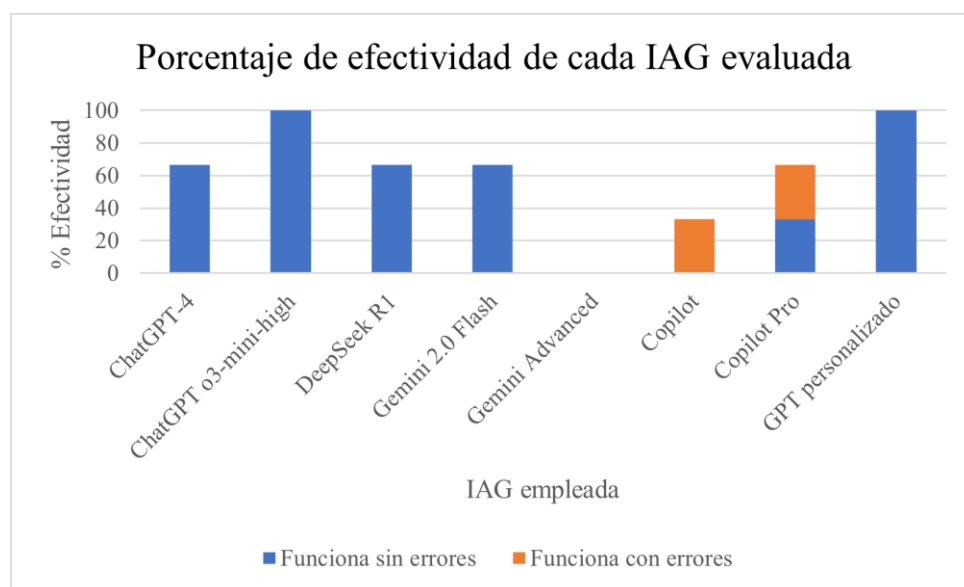


Figura 4-16 Porcentaje de eficacia de cada IAG en los tres proyectos analizados añadiendo la evaluación del GPT personalizado

Por último, con el fin de garantizar que el desarrollo del proyecto respetase las limitaciones técnicas establecidas por la asignatura Informática para la Ingeniería, se solicitó al modelo GPT personalizado actuar como revisor del código fuente, con el objetivo de identificar posibles elementos prohibidos o fragmentos que pudieran poner en duda la autoría del estudiante.

El análisis realizado por el GPT fue altamente preciso, detectando de forma automática la presencia de un puntero en el código, como muestra la Tabla 3-27. Dado que el uso de punteros está restringido en la asignatura salvo en la excepción concreta de pasar parámetros por referencia a una función, el modelo procedió a verificar si se cumplía este criterio. Tras el análisis, el GPT confirmó que el puntero detectado se utilizaba únicamente para pasar valores por referencia, tal y como se contempla dentro de las excepciones permitidas. Esta validación fue comunicada de forma clara, demostrando así la eficacia del GPT personalizado como herramienta de revisión técnica ajustada al marco pedagógico del curso.

En resumen, el funcionamiento del GPT personalizado ha sido especialmente destacable gracias a su entrenamiento específico con el contenido de la asignatura Informática para la Ingeniería y a la incorporación explícita de las restricciones sobre qué elementos del lenguaje C no deben utilizarse. Esta adaptación ha permitido que el modelo genere código perfectamente alineado con el nivel del alumnado y las normas del curso, ofreciendo soluciones funcionales, comprensibles y ajustadas al temario, lo que lo convierte en una herramienta de apoyo altamente eficaz y coherente con los objetivos docentes.

5 CONCLUSIONES Y LÍNEAS FUTURAS

5.1 Conclusiones derivadas del TFG

Los resultados obtenidos a lo largo de este trabajo permiten extraer una serie de conclusiones relevantes sobre el papel de la Inteligencia Artificial Generativa en el contexto educativo, especialmente en asignaturas de introducción a la programación como Informática para la Ingeniería.

En primer lugar, se ha comprobado que las IAG actuales poseen un potencial significativo como herramientas de apoyo al aprendizaje. Sin embargo, también ha quedado patente que las versiones gratuitas presentan dificultades notables a la hora de interpretar y generar correctamente tanto diagramas de flujo como código sencillo, lo cual limita su utilidad en etapas iniciales del aprendizaje. A pesar de estas limitaciones, modelos como ChatGPT-4 y DeepSeek V3 han mostrado un rendimiento destacable, siendo los únicos capaces de generar representaciones válidas y comprensibles en tareas de baja complejidad.

Cuando se abordaron tareas de mayor dificultad, como la elaboración de proyectos de programación más complejos, se observaron diferencias aún más marcadas entre los modelos evaluados. ChatGPT o3-mini-high, la versión de pago de ChatGPT recomendada específicamente para programación, destacó como el modelo más preciso y coherente, siendo el único que resolvió correctamente los tres proyectos propuestos sin errores de compilación ni desviaciones del enunciado. Este resultado cobra aún más valor si se compara con el rendimiento de otros modelos de pago como Gemini Advanced, que presentó un desempeño sorprendentemente deficiente, incluso inferior al de su propia versión gratuita. Este hecho refuerza la idea de que el acceso a una versión *premium* no garantiza una mayor calidad en las respuestas generadas y pone de manifiesto la necesidad de evaluar cada modelo por su rendimiento real, no por su posicionamiento comercial.

Esta observación invita a reflexionar sobre cómo, en el contexto actual de expansión de la IAG, conceptos como el *vibe coding* están transformando profundamente el papel tradicional del programador. Esta técnica consiste en plantear un problema mediante una breve descripción, que sirve como entrada para un modelo de lenguaje extenso, el cual genera automáticamente el código necesario. En este nuevo enfoque, el programador deja de centrarse en la escritura manual del código para asumir un rol más estratégico, orientado a guiar, validar y refinar el resultado producido por la IA.

Este cambio de paradigma se relaciona directamente con la idea de una posible singularidad tecnológica, un escenario hipotético en el que la IAG no solo iguala, sino que supera las capacidades humanas, provocando un punto de inflexión en el que el progreso tecnológico podría acelerarse de manera autónoma y difícil de prever. En este contexto, se vuelve especialmente relevante mantener una evaluación rigurosa y objetiva de cada modelo, basada en su rendimiento real y no en su apariencia comercial o nivel de acceso.

Por otro lado, se ha llevado a cabo un análisis en profundidad de la capacidad de las IAG para replicar los criterios de evaluación utilizados por el profesorado. Los resultados muestran una alta correspondencia entre las calificaciones asignadas por el docente y las generadas por ChatGPT o3-mini-high, especialmente cuando se le proporcionan criterios claros y estructurados. Esto evidencia que, con una guía adecuada, las IAG pueden actuar como apoyo válido y objetivo en tareas de corrección académica, aliviando parte de la carga del profesorado sin sustituir su juicio pedagógico.

Una mención especial merece el uso del GPT personalizado, entrenado con el contenido específico de la asignatura y ajustado a sus restricciones técnicas. Este modelo ha demostrado una capacidad sobresaliente no solo para generar código alineado con el nivel requerido, sino también para identificar automáticamente elementos que no deben utilizarse, como el uso indebido de punteros o estructuras no contempladas en el temario. Gracias a esta adaptación, el GPT personalizado ha cumplido un triple rol: como herramienta de apoyo al estudiante en el desarrollo de soluciones correctas, como un medio dedicado a la calificación de proyectos aportados por el alumnado, y como revisor técnico para el profesorado, asegurando el cumplimiento normativo del código entregado.

En conjunto, se puede afirmar que las IAG, y especialmente los modelos personalizados, representan una oportunidad real y madura para su integración en entornos educativos, tanto en tareas de asistencia al alumnado como en labores de apoyo al profesorado. No obstante, su uso debe ser siempre supervisado, ya que ciertos aspectos de la evaluación, como la calidad de los comentarios o la creatividad de las mejoras, aún presentan un componente subjetivo que requiere la intervención docente.

5.2 Líneas futuras

A partir de los resultados obtenidos, se plantean varias líneas de trabajo para el futuro:

- Ampliar el entrenamiento del GPT personalizado incluyendo no solo el contenido teórico y las restricciones del lenguaje, sino también ejemplos reales de buenas y malas prácticas en programación para refinar aún más su capacidad evaluadora.
- Desarrollar interfaces adaptadas al entorno docente, que permitan al profesorado utilizar modelos generativos de manera accesible y guiada, facilitando tareas como la corrección, el aporte de *feedback* automático y la generación de materiales personalizados.
- Investigar el impacto del uso de IAG en el rendimiento académico del alumnado, comparando grupos con y sin acceso a estas herramientas para cuantificar su efecto real sobre el aprendizaje y la autonomía.
- Explorar su uso en otras asignaturas técnicas del grado de cara a valorar su adaptabilidad en diferentes contextos curriculares.
- Desarrollar modelos que combinen evaluación técnica y pedagógica, integrando criterios de estilo, buenas prácticas, creatividad y claridad, acercándose así a una evaluación más completa y humanamente contextualizada.

En definitiva, este trabajo muestra que la Inteligencia Artificial Generativa, cuando se aplica con criterio, formación y supervisión, puede contribuir de forma eficaz a la mejora del proceso educativo en programación, beneficiando tanto a estudiantes como a docentes.

6 BIBLIOGRAFÍA

- [1] Weimann-Sandig y N., «*How the use of AI is changing the role of educators at universities - and why this is by no means a bad thing,*» de *INTED2024*, Dresden, 2024.
- [2] D. E. Council, «*Digital Education Council Global AI Student Survey 2024,*» 2 Agosto 2024. [En línea]. Available: <https://www.digitaleducationcouncil.com/post/digital-education-council-global-ai-student-survey-2024>. [Último acceso: 27 Febrero 2025].
- [3] K. Sperling, C.-J. Stenberg, C. McGrath, A. Akerfeldt, H. Fredrik y L. Stenliden, «*In search of artificial intelligence (AI) literacy in teacher education: A scoping review,*» *Computers and Education Open*, vol. 6, 2024.
- [4] D. Baidoo-Anu y L. Owusu Ansah, «*Education in the Era of Generative Artificial Intelligence (AI): Understanding the Potential Benefits of ChatGPT in Promoting Teaching and Learning,*» *Journal of AI*, vol. 7, nº 1, pp. 52-62, 2023.
- [5] D. Ifenthaler, R. Majumdar, P. Gorissen, M. Judge, S. Mishra, J. Raffaghelli y A. Shimada, «*Artificial Intelligence in Education: Implications for Policymakers, Researchers, and Practitioners,*» *Technology, Knowledge and Learning*, vol. 29, pp. 1693-1710, 2024.
- [6] RAE, «*Inteligencia Artificial,*» [En línea]. Available: <https://dle.rae.es/inteligencia>. [Último acceso: 24 Enero 2025].
- [7] J. I. Elvira Sánchez, «*El Mecanismo de Anticitera y los Thaumasiourgoi Helenísticos,*» *Studia Hermetica Journal*, vol. 2, nº 2, 2012.
- [8] N. Crato, «*The Enigma Machine,*» de *Figuring It Out*, Lisboa, Copernicus, Berlin, Heidelberg, 2010, pp. 101-104.
- [9] A. Mathison Turing, «*Computing Machinery and Intelligence,*» *Mind*, pp. 433-460, 1950.
- [10] J. McCarthy, M. Lee Minsky, N. Rochester y C. Elwood Shannon, «*A Proposal for the Dartmouth Summer Research Project on Artificial Intelligence, August 31, 1955,*» *AI Magazine*, vol. 27, nº 4, 2006.
- [11] IBM, «*¿Qué es la inteligencia artificial (IA)?,*» [En línea]. Available: <https://www.ibm.com/es-es/topics/artificial-intelligence>. [Último acceso: 25 Enero 2025].

- [12] J. Da Silva, N. Sherman y I. Rahman-Jones, «*Tech giants are putting \$500bn into 'Stargate' to build up AI in US,*» BBC, 21 Enero 2025. [En línea]. Available: <https://www.bbc.com/news/articles/cy4m84d2xz2o>. [Último acceso: 25 Enero 2025].
- [13] P. Cingari, «Italia y España lideran la inversión en inteligencia artificial en la UE,» Euronews, 30 Abril 2024. [En línea]. Available: <https://es.euronews.com/business/2024/04/30/italia-y-espana-lideran-la-inversion-en-inteligencia-artificial-en-la-ue>. [Último acceso: 25 Enero 2025].
- [14] F. Latienda, «20minutos,» 03 Marzo 2025. [En línea]. Available: <https://www.20minutos.es/lainformacion/economia-y-finanzas/alibaba-tencent-bytedance-convierten-china-nueva-cuna-inteligencia-artificial-5686779/>. [Último acceso: 27 Marzo 2025].
- [15] S. P. Pramila y S. Shah, «*A Review of Machine Learning and Deep Learning Applications,*» de *2018 Fourth International Conference on Computing Communication Control and Automation (ICCUBEA)*, Pune, 2018.
- [16] J. Zeren, H. Pingfan, X. Hongfei y W. Qingsheng, «*Machine Learning and Deep Learning in Chemical Health and Safety: A Systematic Review of Techniques and Applications,*» *Journal of Chemical Health and Safety*, vol. 27, pp. 316-334, 2020.
- [17] K. Sharifani y M. Amini, «*Machine Learning and Deep Learning: A Review of Methods and Applications,*» *World Information Technology and Engineering Journal*, vol. 10, nº 7, pp. 3897-3904, 2023.
- [18] A. Norman, «Aprendizaje Automático en Acción,» Tektime, 2019, pp. 15-45.
- [19] R. I. Mukhamediev, Y. Popova, Y. Kuchin, E. Zaitseva, A. Kalimoldayev, A. Symagulov, V. Levashenko, F. Abdoldina, V. Gopejenko, Y. Kirill, E. Muhamedijeva y M. Yelis, «*Review of Artificial Intelligence and Machine Learning Technologies: Classification, Restrictions, Opportunities and Challenges,*» *Mathematics*, vol. 10, nº 15, 2022.
- [20] IBM, «*Understanding the different types of artificial intelligence,*» 12 Octubre 2023. [En línea]. Available: <https://www.ibm.com/think/topics/artificial-intelligence-types>. [Último acceso: 26 Enero 2025].
- [21] S. Nellis, «*Nvidia CEO says AI could pass human tests in five years,*» Reuters , 1 Marzo 2024. [En línea]. Available: <https://www.businessinsider.com/nvidia-ceo-jensen-huang-agi-ai-five-years-2023-11>. [Último acceso: 31 Marzo 2025].
- [22] E. Taylor, «*Artificial Intelligence Techniques: Explained in Detail,*» 20 Enero 2025. [En línea]. Available: <https://www.theknowledgeacademy.com/blog/artificial-intelligence-techniques/>. [Último acceso: 26 Enero 2025].
- [23] GeeksforGeeks, «*What is an AI Technique?,*» 18 Abril 2024. [En línea]. Available: <https://www.geeksforgeeks.org/what-is-an-ai-technique/>. [Último acceso: 26 Enero 2025].
- [24] S. Raisch y K. Fomina, «*Combining Human and Artificial Intelligence: Hybrid Problem-Solving in Organizations,*» *Academy of Management Review*, 2024.
- [25] IBM, «*Generative AI vs. predictive AI: What's the difference?,*» 9 Agosto 2024. [En línea]. Available: <https://www.ibm.com/think/topics/generative-ai-vs-predictive-ai-whats-the-difference>. [Último acceso: 6 Febrero 2025].
- [26] L. Banh, Strobel y Gero, «*Generative artificial intelligence,*» *Electronic Markets*, vol. 33, nº 63, 2023.

- [27] S. S. Sengar, A. B. Hasan, S. Kumar y F. Carroll, «*Generative artificial intelligence: a systematic review and applications*,» *Multimedia Tools and Applications*, 2024.
- [28] D. Saxena y J. Cao, «*Generative Adversarial Networks (GANS): Challenges, Solutions and Future Directions*,» *ACM Computing Surveys*, vol. 54, nº 63, 2021.
- [29] S. Liang, Z. Pan, Z. Liu, Z. Yin y M. De Rijke, «*A Survey on Variational Autoencoders in Recommender Systems*,» *ACM Computing Surveys*, vol. 56, nº 10, 2024.
- [30] D. Bergmann y C. Stryker, «¿Qué es un autocodificador variacional?,» 12 Junio 2024. [En línea]. Available: <https://www.ibm.com/es-es/think/topics/variational-autoencoder>. [Último acceso: 21 Febrero 2025].
- [31] D. Bergmann y C. Stryker, «¿Qué es un modelo de difusión?,» 21 Agosto 2024. [En línea]. Available: <https://www.ibm.com/es-es/think/topics/diffusion-models>. [Último acceso: 21 Febrero 2025].
- [32] IBM, «¿Qué es un modelo de transformador?,» [En línea]. Available: <https://www.ibm.com/es-es/topics/transformer-model>. [Último acceso: 10 Febrero 2025].
- [33] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, Polosukhin y Illia, «*Attention Is All You Need*,» *Computation and Language*, vol. 7, 2017.
- [34] D. Nam, A. Macvean, V. Hellendoorn, B. Vasilescu y B. Myers, «*Using an LLM to Help With Code Understanding*,» de *ICSE '24: Proceedings of the IEEE/ACM 46th International Conference on Software Engineering*, Lisboa, 2024.
- [35] BBC, «Qué tiene de especial DeepSeek, la nueva herramienta china de inteligencia artificial (y en qué se diferencia de ChatGPT o Gemini),» 28 Enero 2025. [En línea]. Available: <https://www.bbc.com/mundo/articles/cwypdq2nxxyo>. [Último acceso: 21 Febrero 2025].
- [36] ChatGPT, «Cambia a un plan superior,» [En línea]. Available: <https://chatgpt.com/c/67b82f51-3644-8004-b8df-8a0401542cb9#pricing>. [Último acceso: 21 Febrero 2025].
- [37] G. Gemini, «*Gemini Advanced*,» [En línea]. Available: https://one.google.com/explore-plan/gemini-advanced?utm_source=gemini&utm_medium=web&utm_campaign=sidenav_evo&g1_landing_page=65. [Último acceso: 21 Febrero 2025].
- [38] M. Copilot, «*Copilot Pro*,» [En línea]. Available: <https://www.microsoft.com/es-es/store/b/copilotpro>. [Último acceso: 21 Febrero 2025].
- [39] DeepSeek, «*DeepSeek Into the unknown*,» [En línea]. Available: <https://www.deepseek.com/>. [Último acceso: 21 Febrero 2025].
- [40] D. Cahyana, A. Hadiarto, Irawan, D. Puspita Hati, M. Media Pratamaningsih, V. Karolinoerita, A. Mulyani, Sukarman, M. Hikmat, F. Ramadhani, R. Abdul Gani, E. Yatno, R. Bambang Heryanto, Suratman y N. Gofar, «*Application of ChatGPT in soil science research and the perceptions of soil scientists in Indonesia*,» de *Artificial Intelligence in Geosciences*, KeAi, 2024, pp. 1-8.
- [41] DataScientest, «IDE : ¿Qué es un Entorno de Desarrollo Integrado?,» 2 Septiembre 2024. [En línea]. Available: <https://datascientest.com/es/ide-que-es>. [Último acceso: 21 Febrero 2025].

- [42] iTEQUIa, «Cómo Integrar ChatGPT en Visual Studio Code para Optimizar tu Productividad,» 24 Septiembre 2023. [En línea]. Available: <https://itequia.com/es/integrar-chatgpt-en-visual-studio-code/>. [Último acceso: 21 Febrero 2025].
- [43] «Visual Studio Code,» [En línea]. Available: <https://code.visualstudio.com/>. [Último acceso: 21 Febrero 2025].
- [44] «Google Colab,» [En línea]. Available: <https://colab.research.google.com/>. [Último acceso: 21 Febrero 2025].
- [45] Google, «Crea código con Gemini Code Assist,» [En línea]. Available: <https://cloud.google.com/gemini/docs/codeassist/write-code-gemini?hl=es-419#:~:text=Ya%20puedes%20usar%20Gemini%20en%20tu%20IDE..> [Último acceso: 21 Febrero 2025].
- [46] Google AI. [En línea]. Available: https://aistudio.google.com/prompts/new_chat. [Último acceso: 21 Febrero 2025].
- [47] «Obtención de sugerencias de código en el IDE con GitHub Copilot,» [En línea]. Available: <https://docs.github.com/es/copilot/using-github-copilot/getting-code-suggestions-in-your-ide-with-github-copilot>. [Último acceso: 21 Febrero 2025].
- [48] Ó. Barajas Tavares, «Cómo utilizar DeepSeek R1 en Visual Studio Code,» Platzi, 30 Enero 2025. [En línea]. Available: <https://platzi.com/blog/deepseek-r1-visual-studio-code/>. [Último acceso: 21 Febrero 2025].
- [49] S. Sarkar, «AI Industry Analysis: 50 Most Visited AI Tools and Their 24B+ Traffic Behavior,» 2023. [En línea]. Available: <https://writerbuddy.ai/blog/ai-industry-analysis>. [Último acceso: 21 Enero 2025].
- [50] O. Shchegolev y D. Melnikov, «Semrush,» 2008. [En línea]. Available: <https://es.semrush.com/analytics/traffic/>. [Último acceso: 21 Enero 2025].
- [51] DeepMind, [En línea]. Available: <https://alphacode.deepmind.com/>. [Último acceso: 31 Marzo 2025].
- [52] N. Chowdhury, J. Aung, C. J. Shern, O. Jaffe, D. Sherburn, G. Starace, E. Mays, R. Dias, M. Aljube, M. Glaese, C. E. Jimenez, J. Yang, L. Ho, T. Patwardhan, K. Liu y A. Madry, «Introducing SWE-bench Verified,» OpenAI, 13 Agosto 2024. [En línea]. Available: <https://openai.com/index/introducing-swe-bench-verified/>. [Último acceso: 31 Marzo 2025].
- [53] M. Pokrass, «Human-Eval,» OpenAI, [En línea]. Available: <https://github.com/openai/human-eval>. [Último acceso: 31 Marzo 2025].
- [54] «Reddit,» [En línea]. Available: https://www.reddit.com/r/singularity/comments/1gjxqdz/between_april_and_now_swebench_verified_scores/. [Último acceso: 31 Marzo 2025].
- [55] Y. Ramazan y Y. Fatma Gizem Karaoglan, «The effect of generative artificial intelligence (AI)-based tool use on students computational thinking skills, programming self-efficacy and motivation,» *Computers and Education: Artificial Intelligence*, vol. 4, nº 100147, 2023.
- [56] F. Gizachew Deriba, I. Temitayo Sanusi, O. O. Campbell y S. Sunday Oyelere, «Computer Programming Education in the Age of Generative AI: Insights from Empirical Research,» *SSRN Electronic Journal*, 2024.

- [57] R. Mellado, C. Cubillos y G. Ahumada, «*Effectiveness of Generative Artificial Intelligence in learning programming to higher education students*,» de *IEEE International Conference on Automation/XXVI Congress of the Chilean Association of Automatic Control (ICA-ACCA)*, Santiago, 2024.
- [58] M. Kazemitabaar, J. Chow, C. Ka To Ma, B. J. Ericson, D. Weintrop y T. Grossman, «*Studying the effect of AI Code Generators on Supporting Novice Learners in Introductory Programming*,» *arXiv:2302.07427*, vol. 2, 2023.
- [59] A. Vadaparty, D. Zingaro, D. H. Smith IV, M. Padala, C. Alvarado, J. Gorson Benario y L. Porter, «*CS1-LLM: Integrating LLMs into CS1 Instruction*,» de *ITiCSE 2024: Proceedings of the 2024 ACM Conference on Innovation and Technology in Computer Science Education V. 1*, 2024, pp. 297-303.
- [60] L. Porter y D. Zingaro, *Learn AI-Assisted Python Programming: With GitHub Copilot and ChatGPT*, Manning Publishing, 2023.
- [61] A. Gutiérrez, «¿Se puede detectar un texto creado por ChatGPT?,» inesem business school, [En línea]. Available: https://www.inesem.es/revistadigital/informatica-y-tics/detectar-texto-creado-por-chat-gpt/#%C2%BFEl_chat_gpt_tiene_plagio. [Último acceso: 21 Febrero 2025].
- [62] W. Hung Pan, M. Jie Chok, J. L. Shan Wong, Y. Xin Shin, Y. Shian Poon, Z. Yang, C. Yong Chon, D. Lo y M. Kuan Lim, «*Assessing AI Detectors in Identifying AI-Generated Code: Implications for Education*,» *arXiv:2401.03676*, 2024.
- [63] A. Pang y F. Vahid, «*ChatGPT and Cheat Detection in CS1 Using a Program Autograding System*,» de *ITiCSE 2024: Proceedings of the 2024 ACM Conference on Innovation and Technology in Computer Science Education V*, 2024, pp. 367-373.
- [64] M. Atemkeng, S. Hamlomo, B. Welman, N. Oyetunji, P. Ataei y J. L. K. E Fendji, «*Ethics of Software Programming with Generative AI: Is Programming without Generative AI always radical?*,» *arXiv:2408.10554*, vol. 2, 2024.
- [65] C. Europea, «*European approach to artificial intelligence*,» [En línea]. Available: <https://digital-strategy.ec.europa.eu/en/policies/european-approach-artificial-intelligence>. [Último acceso: 17 Febrero 2025].
- [66] N. Hasan, R. Ullah Rana, S. Chowdhury, A. Jahan Dola y M. K. Khan Rony, «*Ethical Considerations In Research*,» *Journal of Nursing Research*, vol. 1, nº 1, 2021.
- [67] A. Nassar y M. Kamal, «*Ethical Dilemmas in AI-Powered Decision-Making: A Deep Dive into Big Data-Driven Ethical Considerations*,» *Neural State*, vol. 11, nº 8, 2021.
- [68] U. Gasser, S. Cortesi, M. Malik y A. Lee, «*Youth and Digital Media: From Credibility to Information Quality*,» *Berkman*, nº 1, 2012.
- [69] L. Andersen, «*Artificial Intelligence in International Development: Avoiding Ethical Pitfalls*,» 20 Mayo 2019. [En línea]. Available: <https://jpia.princeton.edu/news/artificial-intelligence-international-development-avoiding-ethical-pitfalls>. [Último acceso: 17 Febrero 2025].
- [70] Z. Obermeyer, B. Powers, C. Vogeli y S. Mullainathan, «*Dissecting racial bias in an algorithm used to manage the health of populations*,» *National Library of Medicine*, vol. 366, nº 6464, pp. 447-453, 2019.
- [71] AESIA, «Garantizando una IA ética y responsable,» 2023. [En línea]. Available: <https://aesia.digital.gob.es/es>. [Último acceso: 27 Marzo 2025].

- [72] O. Lozo y O. Onishchenko, «*The Potential Role of the Artificial Intelligence in Combating Climate Change and Natural Resources Management: Political, Legal and Ethical Challenges*,» *Grassroots Journals*, vol. 4, n° 3, 2021.
- [73] Z. Amiri, A. Heidari y N. J. Navimipour, «*Comprehensive survey of artificial intelligence techniques and strategies for climate change mitigation*,» *Energy*, vol. 308, n° 1, 2024.
- [74] B. Carsten Stahl, D. Schroeder y R. Rodrigues, «*The Ethics of Artificial Intelligence: An Introduction*,» de *Ethics of Artificial Intelligence*, Springer, 2022, pp. 1-7.
- [75] J. Wawira Gichoya, T. Kaesha, L. A. Celi, N. Safdar, I. Banerjee, J. D. S.-K. L. Banja y H. P. S. Trivedi, «*AI pitfalls and what not to do: mitigating bias in AI*,» *British Journal of Radiology*, vol. 96, n° 1150, 2023.
- [76] CUD-ENM, *Informática para la Ingeniería Práctica 7 - Proyecto*, Marín, 2019-2020.
- [77] CUD-ENM, *Informática para la Ingeniería Práctica 7 - Proyecto*, Marín, 2020-2021.
- [78] CUD-ENM, *Informática para la Ingeniería Práctica 7 - Proyecto*, Marín, 2021-2022.
- [79] Grupo-27, *Proyecto Juego del 8*, Marín: CUD-ENM, 2020.
- [80] Grupo-25, *Proyecto Juego del 8*, Marín: CUD-ENM, 2020.
- [81] Grupo-28, *Proyecto Juego del 8*, Marín: CUD-ENM, 2020.
- [82] P. Li, J. Yang, M. A. Islam y S. Ren, «*Making AI Less “Thirsty”: Uncovering and Addressing the Secret Water Footprint of AI Models*,» vol. 4, 2025.
- [83] «*Peace, dignity and equality*,» United Nations, [En línea]. Available: <https://www.un.org/en/global-issues/water>. [Último acceso: 27 Febrero 2025].
- [84] S. Thomas, «*Inteligencia artificial: ¿cuál es su impacto medioambiental?*,» Schrodex, 11 Noviembre 2023. [En línea]. Available: <https://www.estrategiasdeinversion.com/fondos/inteligencia-artificial-cual-es-su-impacto-medioambiental-n-666879>. [Último acceso: 27 Febrero 2025].
- [85] «*El impacto ambiental de la inteligencia artificial*,» Spring, 20 Febrero 2025. [En línea]. Available: <https://www.spring-gds.com/es/blog/impacto-ambiental-inteligencia-artificial/>. [Último acceso: 27 Febrero 2025].
- [86] «*New Coalition aims to put Artificial Intelligence on a more sustainable path*,» UN environment programme, 17 Febrero 2025. [En línea]. Available: <https://www.unep.org/news-and-stories/press-release/new-coalition-aims-put-artificial-intelligence-more-sustainable-path>. [Último acceso: 27 Febrero 2025].
- [87] «*Ejemplos de diagramas de flujo*,» Área tecnología, [En línea]. Available: <https://www.areatecnologia.com/informatica/ejemplos-de-diagramas-de-flujo.html>. [Último acceso: 14 Febrero 2025].

ANEXO I: IMPLICACIONES SOCIALES Y AMBIENTALES

La integración de la Inteligencia Artificial generativa en la programación ha desencadenado una revolución en la forma en que se concibe el desarrollo de software. Desde la generación automática de código hasta la optimización de algoritmos, la IAG promete aumentar la eficiencia y la productividad de los programadores. Sin embargo, este avance tecnológico no está exento de consideraciones críticas. El entrenamiento y despliegue de modelos de IAG, especialmente aquellos de gran escala, requieren un consumo energético considerable y generan una huella de carbono significativa. En un contexto de creciente preocupación por el cambio climático, es imperativo analizar y mitigar el impacto medioambiental de estas tecnologías, abordando el equilibrio entre innovación tecnológica y responsabilidad ecológica.

AI.1 Altos requerimientos energéticos

En primer lugar, uno de los principales problemas ambientales asociados con la IA es su elevado consumo de energía. El entrenamiento de modelos de aprendizaje profundo, en particular, requiere enormes cantidades de potencia de cálculo, lo que se traduce en un gasto energético considerable. Los centros de datos que albergan estos sistemas consumen grandes cantidades de electricidad para mantener los servidores en funcionamiento y refrigerados, lo que contribuye a las emisiones de gases de efecto invernadero y al cambio climático. El consumo energético asociado a la Inteligencia Artificial ha experimentado un crecimiento notable en los últimos años, superando el de otras tecnologías emergentes. Se anticipa que esta tendencia continúe, lo que podría representar un desafío considerable para la transición global hacia una economía con bajas emisiones de carbono.

En segundo lugar, el consumo de agua por parte de los centros de datos, tanto en su construcción como en su operación para la refrigeración de equipos, plantea una creciente preocupación. Las estimaciones de [82] sugieren que, a nivel global, la infraestructura asociada a la IA podría llegar a consumir seis veces el volumen de agua utilizado por Dinamarca, un país con seis millones de habitantes. Este consumo masivo adquiere una dimensión crítica cuando se considera que una octava parte de la población mundial carece de acceso a agua potable y saneamiento básico, como se muestra en [83] y en la Figura AI-1.



Figura AI-1 Personas sin acceso a agua potable (ONU-HABITAT)

Además del consumo directo de energía, la IA también tiene una huella de carbono indirecta asociada a la fabricación de hardware especializado, como unidades de procesamiento gráfico (GPU) y circuitos integrados para aplicaciones específicas (ASIC). La producción de estos componentes requiere la extracción de recursos naturales y el uso de productos químicos, lo que genera emisiones de carbono en todo el proceso. Por otro lado, como muestra [84], estos componentes representan una seria amenaza

ambiental durante su eliminación ya que pueden filtrarse al entorno, resultando en contaminación significativa.

Asimismo, la IA también puede contribuir a la obsolescencia programada de dispositivos electrónicos, ya que los modelos de aprendizaje automático se actualizan constantemente y requieren hardware cada vez más potente. Esto genera una gran cantidad de residuos electrónicos, que contienen materiales tóxicos y contaminantes.

AI.2 Implementación de la IA en la gestión ambiental

La implementación de la IA en la gestión ambiental está transformando la forma en que abordamos los desafíos ecológicos. Desde la monitorización de la biodiversidad con drones y análisis de imágenes satelitales, hasta la optimización de la gestión de residuos y la predicción de desastres naturales, la IA ofrece herramientas poderosas. Los algoritmos de aprendizaje automático permiten analizar grandes volúmenes de datos para identificar patrones y tendencias, facilitando la toma de decisiones.

A pesar de sus impactos negativos, la IA también ofrece un gran potencial para abordar los desafíos ambientales. Por ejemplo, se puede utilizar para optimizar el consumo de energía en edificios y ciudades inteligentes, mejorar la gestión de residuos y recursos naturales, y desarrollar materiales y procesos de producción más sostenibles.

La IA también puede desempeñar un papel crucial en la lucha contra el cambio climático, al permitir la monitorización y predicción de fenómenos meteorológicos extremos, la modelización de escenarios climáticos y la optimización de la generación y distribución de energías renovables.

AI.3 Reducción del impacto ambiental

Para mitigar los impactos ambientales de la IA, es necesario adoptar un enfoque integral que abarque desde la eficiencia energética de los algoritmos y el hardware hasta la gestión responsable de los residuos electrónicos. Algunas medidas clave incluyen:

- Desarrollar algoritmos de IA más eficientes y sostenibles, que requieran menos potencia de cálculo y energía.
- Promover el uso de energías renovables en los centros de datos y la fabricación de hardware.
- Fomentar la economía circular y la reutilización de componentes electrónicos.
- Establecer regulaciones y estándares para el desarrollo y despliegue de IA sostenible.
- Concienciar a la sociedad sobre los impactos ambientales de la IA y promover un consumo responsable.

Como muestra [85], en un hito crucial para el futuro de la tecnología y el medio ambiente, la Cumbre de Acción sobre Inteligencia Artificial en París, celebrada el 11 de febrero de 2025, fue testigo del nacimiento de la Coalición para una Inteligencia Artificial Ambientalmente Sostenible. Esta iniciativa global, liderada por Francia, el Programa de las Naciones Unidas para el Medio Ambiente (PNUMA) y la Unión Internacional de Telecomunicaciones (UIT), congrega a más de 100 socios, representando un amplio espectro de actores: desde 37 empresas tecnológicas pioneras, hasta once naciones y cinco organizaciones internacionales clave. El objetivo primordial de esta coalición, según [86], es forjar un camino donde el desarrollo de la IA impulse la descarbonización, combata la contaminación y salvaguarde la biodiversidad, garantizando así la sostenibilidad dentro de los límites planetarios.

En conclusión, la Inteligencia Artificial es una herramienta poderosa que puede contribuir tanto a la degradación como a la protección del medio ambiente. Es fundamental adoptar un enfoque responsable y sostenible para maximizar sus beneficios y minimizar sus riesgos.

ANEXO II: REFLEXIONES ÉTICAS Y SOCIALES

El avance y la integración de las Inteligencias Artificiales Generativas en los entornos educativos, profesionales y sociales plantea no solo oportunidades, sino también una serie de implicaciones éticas y sociales que deben ser abordadas con responsabilidad y mirada crítica. A lo largo de este trabajo, al evaluar diferentes modelos de IAG aplicados al aprendizaje de programación, han surgido varias cuestiones relevantes que van más allá del rendimiento técnico.

AII.1 Sesgos algorítmicos y falta de transparencia

Una de las principales preocupaciones éticas asociadas al uso de IAG es la presencia de sesgos. Al estar entrenadas con grandes cantidades de datos provenientes de Internet y repositorios abiertos, estas herramientas pueden reflejar e incluso amplificar estereotipos culturales, de género o raciales. Aunque este riesgo es más visible en modelos aplicados a texto libre o generación creativa, también puede afectar indirectamente a decisiones de programación, ejemplos ofrecidos o suposiciones implícitas que realiza el modelo.

Además, muchas IAG funcionan como sistemas opacos, sin transparencia suficiente sobre sus procesos internos de toma de decisiones. Esta falta de explicaciones dificulta la trazabilidad de errores y limita la posibilidad de auditoría ética, lo que refuerza la necesidad de supervisión humana y de criterios claros a la hora de interpretar las respuestas.

AII.2 *Deepfakes*, automatización y desinformación

El desarrollo de modelos capaces de generar contenido realista no solo en texto, sino también en imagen, audio y vídeo, ha abierto la puerta a fenómenos como los *deepfakes* o la desinformación automatizada. Aunque este trabajo se ha centrado en aplicaciones educativas, no puede ignorarse que la misma tecnología que permite generar código o resolver ejercicios puede ser usada para fines maliciosos.

Este contexto refuerza la importancia de una formación ética en el uso de IAG, que prepare al alumnado no solo para utilizarlas eficazmente, sino también para reconocer sus riesgos y actuar con responsabilidad.

AII.3 Brecha económica y desigualdad de acceso

Uno de los aspectos más relevantes observados durante el desarrollo de este proyecto es el modelo de acceso adoptado por las principales IAG comerciales. Los resultados han demostrado que las versiones más precisas y funcionales, como ChatGPT o3-mini-high, están disponibles únicamente mediante suscripción de pago. Esto implica una barrera económica directa que impide que determinados perfiles de estudiantes puedan beneficiarse de estas herramientas, agravando la brecha digital y educativa.

En contraposición, DeepSeek V3 ha adoptado una estrategia notablemente distinta. A diferencia de los enfoques de negocio escalonados de otras compañías, que ofrecen versiones gratuitas limitadas y funcionalidades avanzadas solo mediante pago, DeepSeek V3 se presenta como un modelo completamente accesible al usuario general, sin necesidad de suscripción ni versiones premium. Esta decisión democratiza el acceso a capacidades avanzadas de generación de código y procesamiento del lenguaje natural, permitiendo que un espectro más amplio de estudiantes y profesionales pueda beneficiarse de su potencia sin obstáculos económicos.

La conjunción entre una implementación técnica de alta calidad y un paradigma de acceso abierto posiciona a DeepSeek V3 como una alternativa de gran valor para contextos educativos, especialmente en instituciones que promueven la equidad. Su apuesta por la accesibilidad y la eficiencia lo convierten

en un recurso estratégico tanto para alumnos como para docentes, y refuerzan la importancia de modelos de IAG diseñados no solo para el rendimiento, sino también para la inclusión.

AII.4 Responsabilidad compartida y uso ético

Por último, es esencial reconocer que el impacto de estas tecnologías no depende exclusivamente del modelo, sino también del usuario, del contexto y de la finalidad con la que se empleen. La Inteligencia Artificial Generativa no es neutral: su uso ético requiere supervisión, formación y un marco normativo que garantice la equidad y el respeto a los principios educativos.

El fomento del pensamiento crítico, la alfabetización digital y el acompañamiento docente son claves para asegurar que las IAG se utilicen como aliadas del aprendizaje y no como sustitutos del razonamiento.

En resumen, las Inteligencias Artificiales Generativas ofrecen oportunidades sin precedentes para transformar la educación, pero también plantean riesgos que deben gestionarse con criterio y responsabilidad. Su integración en el aula debe ir acompañada de políticas inclusivas, formación ética y una visión pedagógica clara.

ANEXO III: EJERCICIOS DE DIAGRAMAS DE FLUJO

AIII.1 Ejercicio 1

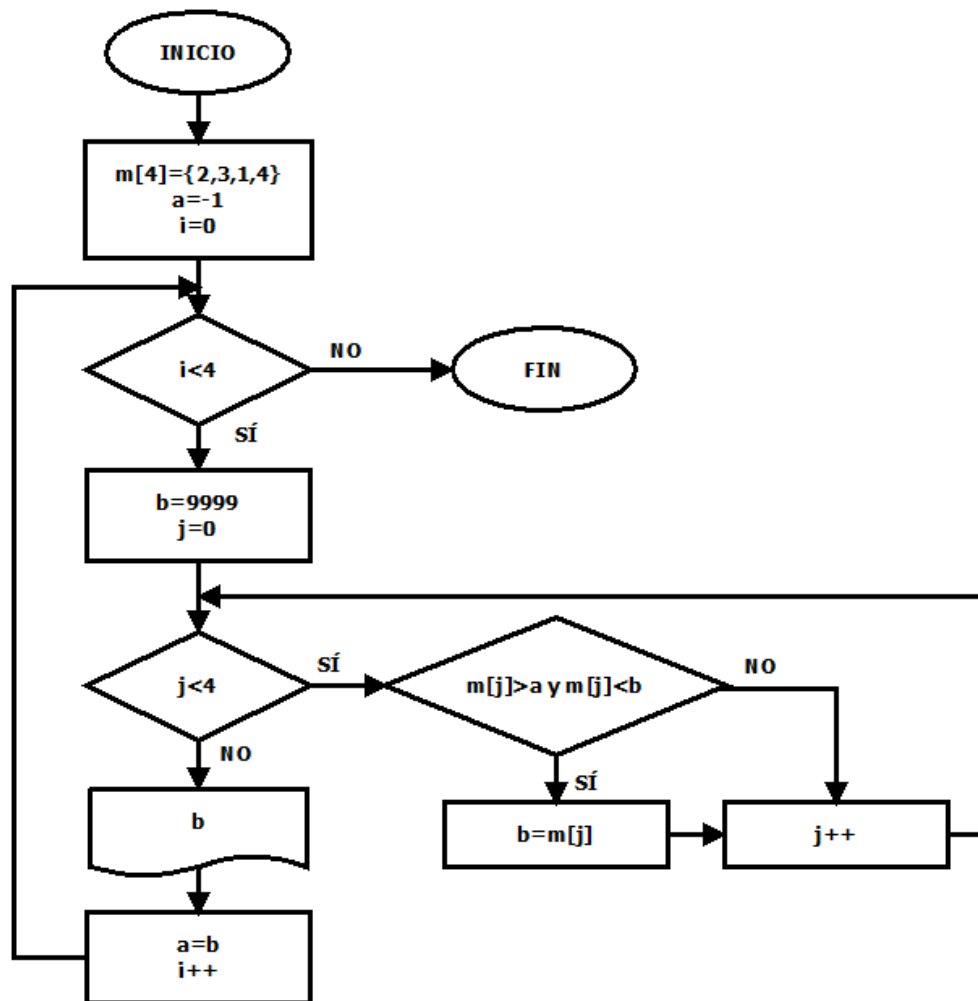


Figura AIII-1 Ejercicio 1 de interpretación de diagramas de flujo

AIII.2 Ejercicio 2

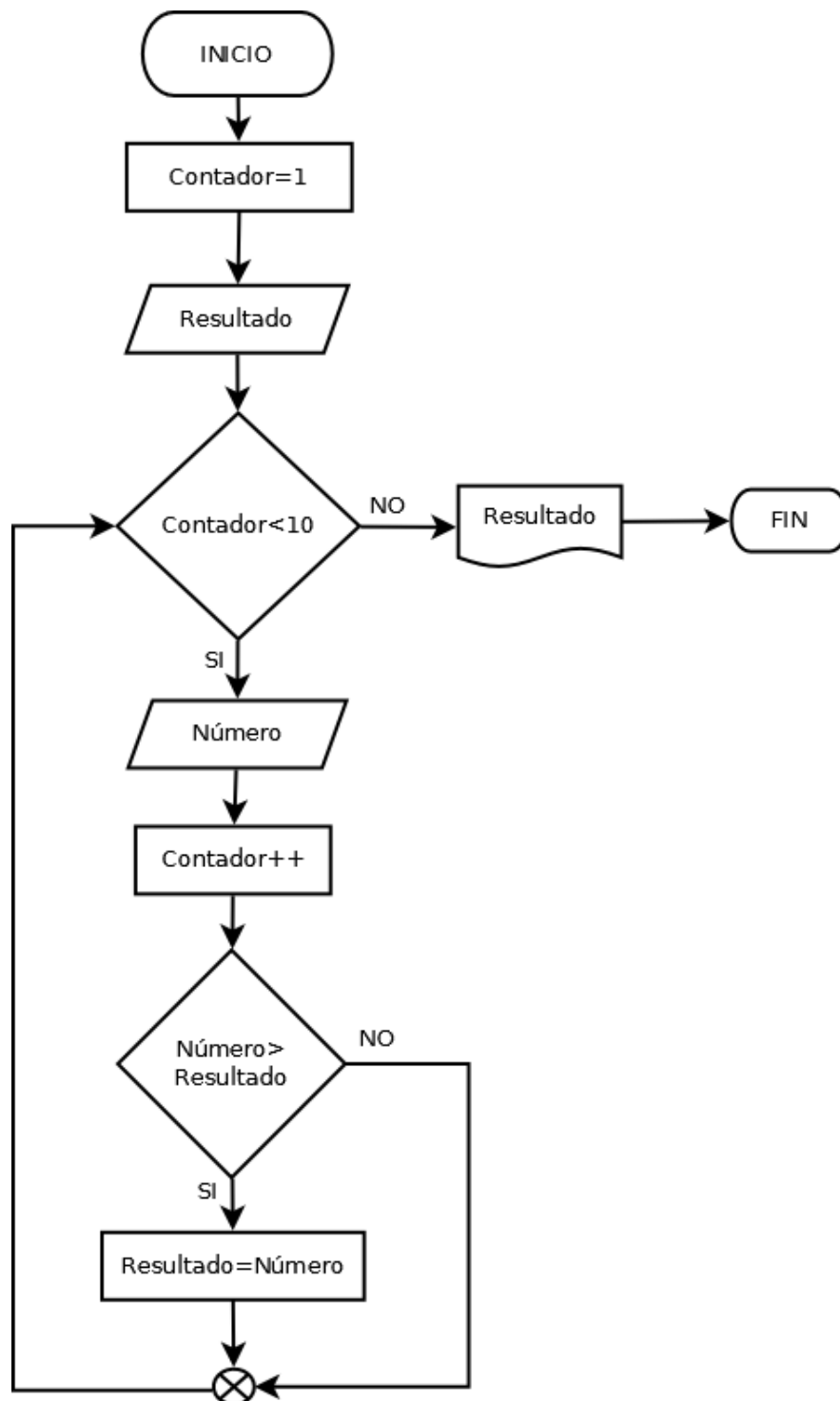


Figura AIII-2 Ejercicio 2 de interpretación de diagramas de flujo

AIII.3 Ejercicio 3

Dados dos números enteros positivos N y D , se dice que D es un divisor de N si el resto de dividir N entre D es 0. Se dice que un número N es perfecto si la suma de sus divisores (excluido el propio N) es N . Por ejemplo 28 es perfecto, pues sus divisores (excluido el 28) son: 1, 2, 4, 7 y 14 y su suma es $1+2+4+7+14=28$. Hacer un organigrama que dado un número N nos diga si es o no perfecto. Ejemplo de problema extraído de [87].

Solución:

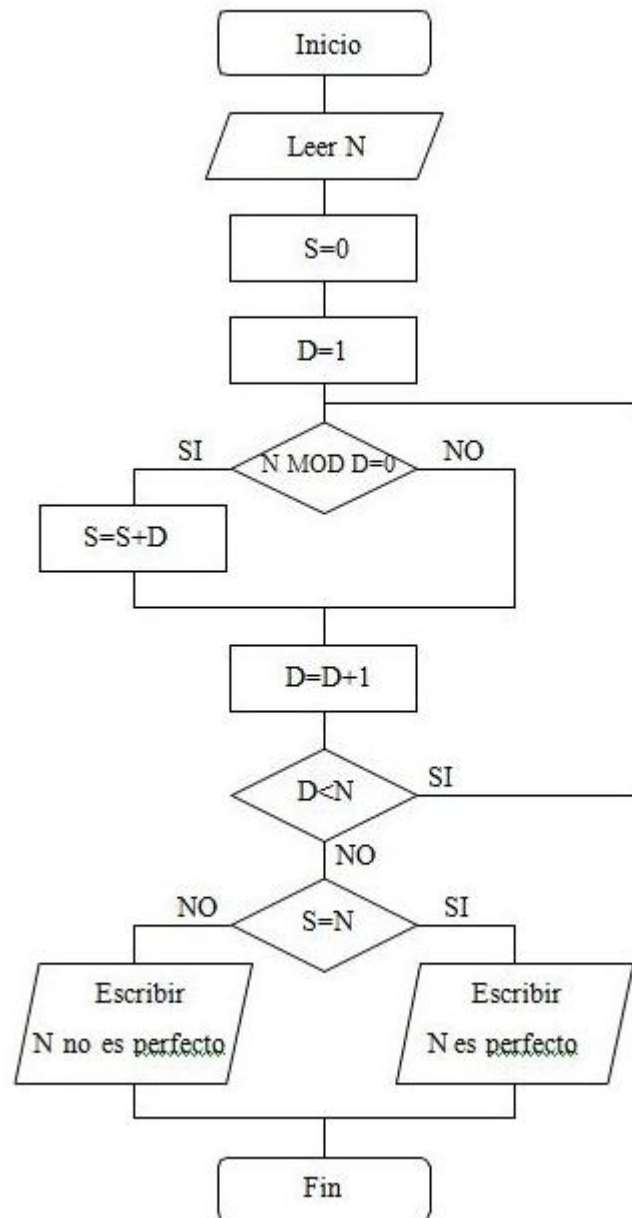


Figura AIII-3 Ejercicio 3 de generación de diagramas de flujo [87]

AIII.4 Ejercicio 4

Un año es bisiesto si es múltiplo de 4, exceptuando los múltiplos de 100, que sólo son bisiestos cuando son múltiplos además de 400, por ejemplo, el año 1900 no fue bisiesto, pero el año 2000 si lo será. Hacer un organigrama que dado un año A nos diga si es o no bisiesto. Ejemplo de problema extraído de [87].

Solución:

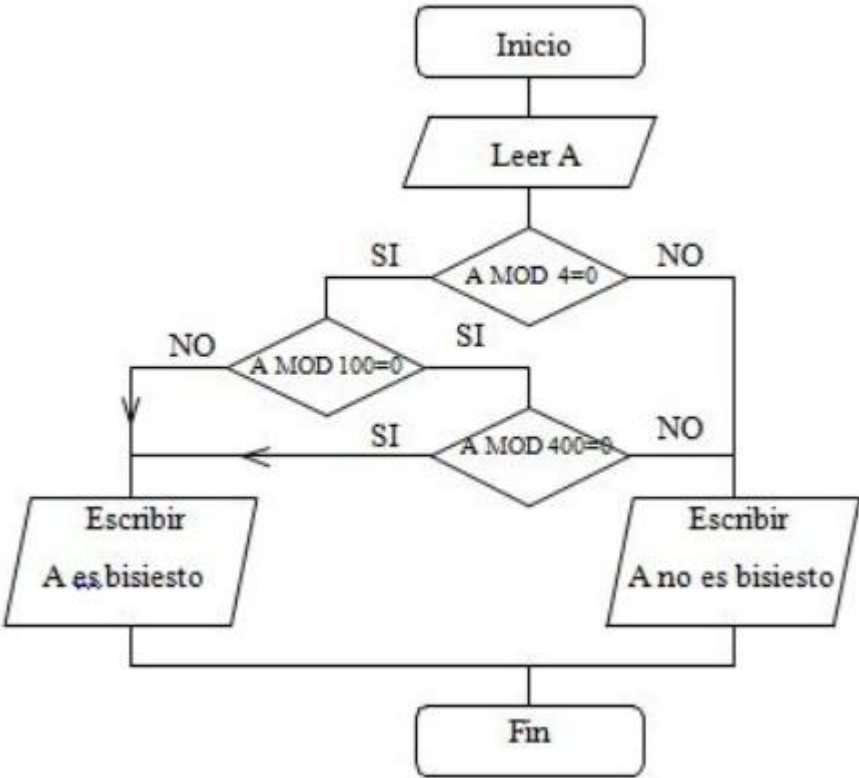


Figura AIII-4 Ejercicio 4 de generación de diagramas de flujo [87]

ANEXO IV: CÓDIGO POR GENERADO POR EL GPT PERSONALIZADO

AIV.1 Código del proyecto del curso 2019-2020 de la asignatura Informática para la Ingeniería

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

#define FILAS 6
#define COLUMNAS 7
#define VACIO ' '
#define JUGADOR 'X'
#define ORDENADOR 'O'
#define FICHERO "partida.txt"

// Declaración de funciones
void mostrarMenu();
void inicializarTablero(char tablero[FILAS][COLUMNAS]);
void mostrarTablero(char tablero[FILAS][COLUMNAS]);
int insertarFicha(char tablero[FILAS][COLUMNAS], int columna, char
ficha);
int comprobarGanador(char tablero[FILAS][COLUMNAS], char ficha);
int tableroLleno(char tablero[FILAS][COLUMNAS]);
void guardarPartida(char tablero[FILAS][COLUMNAS]);
int cargarPartida(char tablero[FILAS][COLUMNAS]);
int turnoOrdenador(char tablero[FILAS][COLUMNAS]);
void limpiarPantalla();

int main() {
    char tablero[FILAS][COLUMNAS];
    int opcion, columna, ganado = 0, empate = 0;

    srand(time(NULL));

    do {
        mostrarMenu();
        scanf("%d", &opcion);

        if (opcion == 1) {
            inicializarTablero(tablero);
            ganado = 0;
            empate = 0;

            while (!ganado && !empate) {
                limpiarPantalla();
                mostrarTablero(tablero);

                // Turno del jugador
                do {
                    printf("¿En qué columna quieres introducir tu ficha?
(1-7, 0 para salir): ");
                    scanf("%d", &columna);
                    if (columna == 0) break;
```

```
        } while (columna < 1 || columna > 7 ||
!insertarFicha(tablero, columna - 1, JUGADOR));

        if (columna == 0) break;

        limpiarPantalla();
        mostrarTablero(tablero);

        if (comprobarGanador(tablero, JUGADOR)) {
            printf(";Has ganado!\n");
            ganado = 1;
            break;
        }

        if (tableroLleno(tablero)) {
            empate = 1;
            break;
        }

        turnoOrdenador(tablero);
        limpiarPantalla();
        mostrarTablero(tablero);

        if (comprobarGanador(tablero, ORDENADOR)) {
            printf(";Ha ganado el ordenador!\n");
            ganado = 1;
            break;
        }

        if (tableroLleno(tablero)) {
            empate = 1;
            break;
        }
    }

    if (empate) {
        limpiarPantalla();
        mostrarTablero(tablero);
        printf(";Empate! El tablero está completo.\n");
    }

} else if (opcion == 2) {
    if (cargarPartida(tablero)) {
        printf("Partida recuperada con éxito.\n");
    } else {
        printf("Error al cargar la partida.\n");
    }
} else if (opcion == 3) {
    guardarPartida(tablero);
    printf("Partida guardada correctamente.\n");
} else if (opcion == 4) {
    printf("Saliendo del juego...\n");
} else {
    printf("Opción no válida.\n");
}
```

```

    }

    } while (opcion != 4);

    return 0;
}

void mostrarMenu() {
    printf("\nJuego de Conecta 4\n");
    printf("-----\n");
    printf("1. Nueva partida\n");
    printf("2. Cargar partida\n");
    printf("3. Guardar partida en curso\n");
    printf("4. Salir\n");
    printf("Seleccione opción: ");
}

void inicializarTablero(char tablero[FILAS][COLUMNAS]) {
    int i, j;
    for (i = 0; i < FILAS; i++)
        for (j = 0; j < COLUMNAS; j++)
            tablero[i][j] = VACIO;
}

void mostrarTablero(char tablero[FILAS][COLUMNAS]) {
    int i, j;

    // Encabezado
    printf("\n");
    for (j = 1; j <= COLUMNAS; j++)
        printf(" %d", j);
    printf("\n");

    // Línea superior
    for (j = 0; j < COLUMNAS; j++)
        printf("--");
    printf("-\n");

    // Celdas
    for (i = 0; i < FILAS; i++) {
        for (j = 0; j < COLUMNAS; j++) {
            printf("|%c", tablero[i][j]);
        }
        printf("|\n");

        for (j = 0; j < COLUMNAS; j++)
            printf("--");
        printf("-\n");
    }
}

int insertarFicha(char tablero[FILAS][COLUMNAS], int columna, char ficha)
{
    int i;
    for (i = FILAS - 1; i >= 0; i--) {
        if (tablero[i][columna] == VACIO) {
            tablero[i][columna] = ficha;

```

```
        return 1;
    }
}
return 0;
}

int comprobarGanador(char tablero[FILAS][COLUMNAS], char ficha) {
    int i, j;

    // Horizontal
    for (i = 0; i < FILAS; i++)
        for (j = 0; j < COLUMNAS - 3; j++)
            if (tablero[i][j] == ficha && tablero[i][j + 1] == ficha &&
                tablero[i][j + 2] == ficha && tablero[i][j + 3] == ficha)
                return 1;

    // Vertical
    for (i = 0; i < FILAS - 3; i++)
        for (j = 0; j < COLUMNAS; j++)
            if (tablero[i][j] == ficha && tablero[i + 1][j] == ficha &&
                tablero[i + 2][j] == ficha && tablero[i + 3][j] == ficha)
                return 1;

    // Diagonal descendente "\"
    for (i = 0; i < FILAS - 3; i++)
        for (j = 0; j < COLUMNAS - 3; j++)
            if (tablero[i][j] == ficha && tablero[i + 1][j + 1] == ficha
&&
                tablero[i + 2][j + 2] == ficha && tablero[i + 3][j + 3]
== ficha)
                return 1;

    // Diagonal ascendente "/"
    for (i = 3; i < FILAS; i++)
        for (j = 0; j < COLUMNAS - 3; j++)
            if (tablero[i][j] == ficha && tablero[i - 1][j + 1] == ficha
&&
                tablero[i - 2][j + 2] == ficha && tablero[i - 3][j + 3]
== ficha)
                return 1;

    return 0;
}

int tableroLleno(char tablero[FILAS][COLUMNAS]) {
    int i, j;
    for (i = 0; i < FILAS; i++)
        for (j = 0; j < COLUMNAS; j++)
            if (tablero[i][j] == VACIO)
                return 0;
    return 1;
}

void guardarPartida(char tablero[FILAS][COLUMNAS]) {
    FILE *f = fopen(FICHERO, "w");
    int i, j;
    if (f == NULL) {
```

```

        printf("No se pudo guardar la partida.\n");
        return;
    }

    for (i = 0; i < FILAS; i++)
        for (j = 0; j < COLUMNAS; j++)
            fputc(tablero[i][j], f);

    fclose(f);
}

int cargarPartida(char tablero[FILAS][COLUMNAS]) {
    FILE *f = fopen(FICHERO, "r");
    int i, j, c;
    if (f == NULL) return 0;

    for (i = 0; i < FILAS; i++) {
        for (j = 0; j < COLUMNAS; j++) {
            c = fgetc(f);
            if (c == EOF) {
                fclose(f);
                return 0;
            }
            tablero[i][j] = (char)c;
        }
    }

    fclose(f);
    return 1;
}

int turnoOrdenador(char tablero[FILAS][COLUMNAS]) {
    int col;
    do {
        col = rand() % COLUMNAS;
    } while (!insertarFicha(tablero, col, ORDENADOR));
    return col;
}

void limpiarPantalla() {
    printf("\033[2J\033[H");
}

```

AIV.2 Código del proyecto del curso 2020-2021 de la asignatura Informática para la Ingeniería

```

#include <stdio.h>
#include <stdlib.h>
#include <time.h>

// Constantes
#define FILAS 3
#define COLUMNAS 3
#define MAX_NOMBRE 21
#define FICHERO "tablero.txt"

// Declaración de funciones

```

```
void inicializarTablero(int tablero[FILAS][COLUMNAS]);
void mostrarTablero(int tablero[FILAS][COLUMNAS], int movimientos);
void desordenarTablero(int tablero[FILAS][COLUMNAS]);
void moverFicha(int tablero[FILAS][COLUMNAS], char direccion, int
*movimientos);
int tableroOrdenado(int tablero[FILAS][COLUMNAS]);
void guardarPartida(char nombre[], int tablero[FILAS][COLUMNAS], int
movimientos);
int cargarPartida(char nombre[], int tablero[FILAS][COLUMNAS], int
*movimientos);
void limpiarPantalla();

int main() {
    int tablero[FILAS][COLUMNAS];
    int movimientos = 0;
    char nombre[MAX_NOMBRE];
    char opcion;

    printf("Introduzca su nombre (máximo 20 caracteres): ");
    scanf("%20s", nombre);

    inicializarTablero(tablero);
    desordenarTablero(tablero);

    do {
        limpiarPantalla();
        mostrarTablero(tablero, movimientos);
        printf("(w) Arriba (s) Abajo (a) Izq (d) Der (c) Cargar (g)
Guardar (x) Salir.\n");
        printf("Introduzca opción: ");
        scanf(" %c", &opcion);

        if (opcion == 'x') break;
        else if (opcion == 'g') {
            guardarPartida(nombre, tablero, movimientos);
        } else if (opcion == 'c') {
            if (!cargarPartida(nombre, tablero, &movimientos)) {
                printf("Error al cargar la partida.\n");
                getchar(); getchar();
            }
        } else {
            moverFicha(tablero, opcion, &movimientos);
        }

    } while (!tableroOrdenado(tablero));

    if (tableroOrdenado(tablero)) {
        limpiarPantalla();
        mostrarTablero(tablero, movimientos);
        printf("Enhorabuena %s, has completado el juego en %d
movimientos!\n", nombre, movimientos);
    }

    return 0;
}

// Inicializa el tablero en estado resuelto
```

```

void inicializarTablero(int tablero[FILAS][COLUMNAS]) {
    int valor = 1;
    for (int i = 0; i < FILAS; i++) {
        for (int j = 0; j < COLUMNAS; j++) {
            if (valor < 9)
                tablero[i][j] = valor++;
            else
                tablero[i][j] = 0; // hueco
        }
    }
}

// Muestra el tablero por pantalla
void mostrarTablero(int tablero[FILAS][COLUMNAS], int movimientos) {
    for (int i = 0; i < FILAS; i++) {
        for (int j = 0; j < COLUMNAS; j++) {
            if (tablero[i][j] == 0)
                printf(" ");
            else
                printf("%d ", tablero[i][j]);
        }
        printf("\n");
    }
    printf("Movimientos: %d\n", movimientos);
}

// Desordena el tablero realizando movimientos aleatorios válidos
void desordenarTablero(int tablero[FILAS][COLUMNAS]) {
    srand(time(NULL));
    for (int i = 0; i < 100; i++) {
        char dir;
        int r = rand() % 4;
        if (r == 0) dir = 'w';
        else if (r == 1) dir = 'a';
        else if (r == 2) dir = 's';
        else dir = 'd';
        int dummy = 0;
        moverFicha(tablero, dir, &dummy);
    }
}

// Mueve una ficha en la dirección dada, si es posible
void moverFicha(int tablero[FILAS][COLUMNAS], char direccion, int
*movimientos) {
    int fila, col, nuevaFila, nuevaCol;

    // Localizar hueco (0)
    for (fila = 0; fila < FILAS; fila++) {
        for (col = 0; col < COLUMNAS; col++) {
            if (tablero[fila][col] == 0) break;
        }
        if (col < COLUMNAS && tablero[fila][col] == 0) break;
    }

    nuevaFila = fila;
    nuevaCol = col;
}

```

```
        if (direccion == 'w') nuevaFila++;
        else if (direccion == 's') nuevaFila--;
        else if (direccion == 'a') nuevaCol++;
        else if (direccion == 'd') nuevaCol--;

        if (nuevaFila >= 0 && nuevaFila < FILAS && nuevaCol >= 0 && nuevaCol
< COLUMNAS) {
            tablero[filas][col] = tablero[nuevaFila][nuevaCol];
            tablero[nuevaFila][nuevaCol] = 0;
        }

        (*movimientos)++;
    }

// Verifica si el tablero está ordenado correctamente
int tableroOrdenado(int tablero[FILAS][COLUMNAS]) {
    int valor = 1;
    for (int i = 0; i < FILAS; i++) {
        for (int j = 0; j < COLUMNAS; j++) {
            if (i == 2 && j == 2) {
                if (tablero[i][j] != 0) return 0;
            } else {
                if (tablero[i][j] != valor++) return 0;
            }
        }
    }
    return 1;
}

// Guarda la partida en un fichero
void guardarPartida(char nombre[], int tablero[FILAS][COLUMNAS], int
movimientos) {
    FILE *f = fopen(FICHERO, "w");
    if (f == NULL) {
        printf("Error al abrir el fichero.\n");
        return;
    }

    fprintf(f, "%s\n", nombre);
    fprintf(f, "%d\n", movimientos);
    for (int i = 0; i < FILAS; i++)
        for (int j = 0; j < COLUMNAS; j++)
            fprintf(f, "%d\n", tablero[i][j]);

    fclose(f);
}

// Carga la partida desde el fichero
int cargarPartida(char nombre[], int tablero[FILAS][COLUMNAS], int
*movimientos) {
    FILE *f = fopen(FICHERO, "r");
    if (f == NULL) return 0;

    fscanf(f, "%20s", nombre);
    fscanf(f, "%d", movimientos);
    for (int i = 0; i < FILAS; i++)
        for (int j = 0; j < COLUMNAS; j++)
```

```
        fscanf(f, "%d", &tablero[i][j]);

        fclose(f);
        return 1;
    }

    // Limpia la pantalla en sistemas tipo UNIX
    void limpiarPantalla() {
        printf("\033[2J\033[H");
    }
```

AIV.3 Código del proyecto del curso 2021-2022 de la asignatura Informática para la Ingeniería

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

#ifdef _WIN32
#include <windows.h>
#else
#include <unistd.h>
#endif

#define FILAS 8
#define COLUMNAS 3
#define JUGADORES 8

#define VACIO ' '
#define PLACA '='
#define ROTA ' '
#define NPC 'X'
#define PLAYER 'O'

int puente[FILAS][COLUMNAS];
char visual[FILAS][COLUMNAS];
char cola[JUGADORES];
int posiciones[JUGADORES]; // posición en la cola, -1 si muerto o salvado
int jugadorID;
int muertos = 0, salvados = 0;

void limpiarPantalla() {
    printf("\033[2J\033[H");
}

void esperar(int segundos) {
#ifdef _WIN32
    Sleep(segundos * 1000);
#else
    sleep(segundos);
#endif
}

void inicializarJuego() {
    srand(time(NULL));
    for (int i = 0; i < FILAS; i++) {
        int normal = rand() % 3;
```

```
        for (int j = 0; j < COLUMNAS; j++) {
            puente[i][j] = (j == normal) ? 0 : 1; // 0 = se rompe, 1 =
templado
            visual[i][j] = PLACA;
        }
    }

    for (int i = 0; i < JUGADORES; i++) {
        cola[i] = NPC;
        posiciones[i] = 0;
    }

    jugadorID = rand() % JUGADORES;
    cola[jugadorID] = PLAYER;
}

void mostrarPuente(int filaActual, int colActual, char jugador) {
    limpiarPantalla();
    for (int i = 0; i < JUGADORES; i++) {
        if (posiciones[i] == 0) printf("%c", cola[i]);
    }
    printf("\n\n");
    for (int i = 0; i < FILAS; i++) {
        for (int j = 0; j < COLUMNAS; j++) {
            if (i == filaActual && j == colActual)
                printf("%c ", jugador);
            else
                printf("%c ", visual[i][j]);
        }
        printf("\n");
    }
    printf("Muertos: %d\nSalvados: %d\n", muertos, salvados);
}

int moverNPC(int id) {
    int fila = 0, col = posiciones[id];
    while (fila < FILAS) {
        mostrarPuente(fila, col, cola[id]);
        esperar(1);

        if (puente[fila][col] == 0) {
            visual[fila][col] = ROTA;
            posiciones[id] = -1;
            muertos++;
            return 0;
        }

        fila++;
        if (fila == FILAS) {
            posiciones[id] = -1;
            salvados++;
            return 1;
        }

        int opciones[3] = {col};
        int total = 1;
        if (col > 0) opciones[total++] = col - 1;
```

```

        if (col < 2) opciones[total++] = col + 1;
        col = opciones[rand() % total];
    }
    return 1;
}

int moverJugador() {
    int fila = 0, col = posiciones[jugadorID];
    char input;
    while (fila < FILAS) {
        mostrarPuede(fila, col, PLAYER);
        printf("Elige dirección (a: izquierda, s: centro, d: derecha, g:
guardar): ");
        scanf(" %c", &input);

        if (input == 'g') {
            printf("Guardado no implementado aún.\n");
            exit(0);
        }

        int nextCol = col;
        if (input == 'a' && col > 0) nextCol = col - 1;
        else if (input == 'd' && col < 2) nextCol = col + 1;
        else if (input != 's') {
            printf("Dirección inválida. Se mantiene.\n");
        }

        if (puede[fila][nextCol] == 0) {
            visual[fila][nextCol] = ROTA;
            posiciones[jugadorID] = -1;
            mostrarPuede(fila, nextCol, PLAYER);
            printf("¡Has muerto!\n");
            muertos++;
            esperar(2);
            return 0;
        }

        fila++;
        col = nextCol;
        if (fila == FILAS) {
            posiciones[jugadorID] = -1;
            mostrarPuede(fila - 1, col, PLAYER);
            printf("¡Te has salvado!\n");
            salvados++;
            esperar(2);
            return 1;
        }
    }
    return 1;
}

int main() {
    inicializarJuego();

    for (int i = 0; i < JUGADORES; i++) {
        if (i == jugadorID) {
            moverJugador();

```

```
        } else {  
            moverNPC(i);  
        }  
    }  
  
    mostrarPuente(-1, -1, ' ');  
    printf("Juego terminado.\n");  
    return 0;  
}
```

ANEXO V: GLOSARIO

- IA: *Artificial Intelligence*
- IAG: *Inteligencia Artificial Generativa*
- LLM: *Large Language Model*
- SEO: *Search Engine Optimization*
- SLM: *Small Language Models*
- CNN: *Convolutional Neural Networks*
- RNN: *Recurrent Neural Networks*
- WAI: *Weak AI* = NAI: *Narrow AI*
- AGI: *Artificial General Intelligence*
- SAI: *Super Artificial Intelligence*
- IBM: *International Business Machines*
- ML: *Machine Learning*
- NLP: *Natural Language Processing*
- CV: *Computer Vision*
- DL: *Deep Learning*
- GAN: *Generative Adversarial Networks*
- TRM: *Transformer Models*
- VAE: *Variational Autoencoders*
- DM: *Diffusion Model*
- IDE: *Integrated Development Environment*
- API: *Application Programming Interface*
- PNUMA: *Programa de las Naciones Unidas para el Medio Ambiente*
- UIT: *Unión Internacional de Telecomunicaciones*
- GPU: *Graphics Processing Unit*
- ASIC: *Application-Specific Integrated Circuits*
- NPC: *Non-Player Character*
- CLS: *Clear Screen*
- GPT: *Generative Pre-Trained Transformer*