Nombre de la Experiencia	Turing y Lovelace: tutores inteligentes basados en IA generativa para el aprendizaje de programación e ingeniería de software
Autor 1	● *Nombre: Juan Felipe Aranguren Checa* ● *Institución: Universidad Icesi* ● *Correo: jfaranguren@icesi.edu.co* ● *Teléfono: 3216506195*
Autor 2	● *Nombre: Mónica María Rojas* ● *Institución: Universidad Icesi* ● *Correo: mmrojas@icesi.edu.co* ● *Teléfono: 3152768206*
Presentación de la experiencia	La experiencia consiste en la implementación de tutores inteligentes basados en inteligencia artificial generativa como recurso pedagógico para acompañar el aprendizaje de estudiantes en los cursos Algoritmos y Programación I e Ingeniería de Software I, de programas de ingeniería. Esta iniciativa surge ante el desafío de ofrecer un seguimiento formativo más cercano y retroalimentación oportuna en cursos con alta cantidad de estudiantes y con una diversidad significativa de conocimientos previos. La práctica se centra en la integración pedagógica de dos tutores inteligentes denominados "Turing" y "Lovelace", diseñados para apoyar a los estudiantes durante el desarrollo de actividades académicas relacionadas con la comprensión de problemas, el análisis de requerimientos, el diseño de soluciones de software y la implementación de programas en lenguaje Java. Los tutores actúan como asistentes de aprendizaje que ofrecen explicaciones, orientaciones y sugerencias de mejora, sin proporcionar directamente las soluciones, con el fin de promover el razonamiento y la construcción autónoma del conocimiento. La experiencia se desarrolla mediante la incorporación explícita de los tutores inteligentes en las consignas y actividades de los cursos. Los estudiantes interactúan con la herramienta durante el proceso de resolución de problemas, utilizándola para clarificar conceptos, revisar sus artefactos de software o identificar posibles errores. Como parte de la actividad, los estudiantes deben documentar sus interacciones con el tutor, incluyendo los prompts utilizados y las respuestas obtenidas, lo que permite reflexionar sobre el uso de la inteligencia artificial como apoyo al aprendizaje. En la ejecución de la experiencia participan el docente responsable del curso, quien diseña las actividades y orienta el uso pedagógico de la herramienta, y los estudiantes, quienes interactúan con el tutor inteligente como parte de su proceso formativo. La iniciativa se apoya además en el marco conceptual de la AI Assessment Scale (AIAS), que orienta el uso responsable de la inteligencia artificial en contextos educativos. Entre los principales resultados se destacan el fortalecimiento de la retroalimentación formativa, por parte de los docentes; y el desarrollo de habilidades de pensamiento crítico y reflexión sobre el proceso de aprendizaje frente al uso de la inteligencia artificial, por parte de los estudiantes.

Descripción de la experiencia	La práctica pedagógica consiste en la integración de tutores inteligentes basados en inteligencia artificial generativa en los cursos Algoritmos y Programación I e Ingeniería de Software I, como estrategia para fortalecer el acompañamiento al aprendizaje, ampliar las oportunidades de retroalimentación formativa y promover un uso crítico, ético y reflexivo de la IA. En la experiencia docentes, estudiantes y sistemas de IA colaboran de manera intencional para enriquecer los procesos de comprensión, análisis, diseño y desarrollo de soluciones de software, sin sustituir el criterio pedagógico ni el trabajo cognitivo del estudiante. La experiencia inicia en una etapa de diagnóstico y problematización. El docente identifica una dificultad recurrente en este tipo de cursos: la imposibilidad de brindar seguimiento individualizado y retroalimentación oportuna a todos los estudiantes, especialmente en grupos numerosos y con alta diversidad en los saberes previos. A partir de esta necesidad, se plantea la incorporación de un tutor inteligente como un recurso de apoyo pedagógico que permita acompañar a los estudiantes de manera más continua durante su proceso de aprendizaje, particularmente en momentos críticos como la comprensión de los enunciados, la estructuración de la solución, la revisión de los artefactos de software (requerimientos, diseño y código) y la depuración de errores. En una segunda etapa, el docente realiza el diseño pedagógico de la experiencia. En este momento define en qué actividades se integrará el tutor inteligente, cuáles son los resultados de aprendizaje que se buscan fortalecer y qué tipo de interacción con la IA será pertinente en cada caso. Esta planeación no se limita a incorporar una herramienta tecnológica, sino que implica establecer con claridad el propósito formativo de su uso. Para ello, se toma como referente el modelo AI Assessment Scale (AIAS), con el fin de delimitar el nivel de uso permitido de la inteligencia artificial en las actividades y asegurar coherencia entre la estrategia, los resultados de aprendizaje y los criterios de evaluación. El docente ajusta las consignas, los entregables y las orientaciones del curso para explicitar cuándo y cómo puede utilizarse el tutor, así como las evidencias que deberán presentar los estudiantes sobre dicho uso. Posteriormente, se desarrolla una etapa de socialización y encuadre pedagógico con los estudiantes. El docente presenta el tutor inteligente, explica su propósito dentro del curso y aclara que su función es apoyar el aprendizaje, no reemplazar el razonamiento ni resolver completamente las tareas. En este momento se establecen acuerdos sobre el uso responsable de la IA, se explican las reglas del juego en términos de integridad académica y se orienta a los estudiantes sobre la importancia de interactuar críticamente con las respuestas del sistema. Además, se presenta el nivel de uso de IA definido para la actividad, de modo que los estudiantes comprendan qué tipo de apoyo está permitido y qué responsabilidades mantienen como autores de sus soluciones. La implementación en el aula ocurre durante el desarrollo de actividades de aprendizaje y evaluación asociadas con la resolución de problemas propios de la ingeniería de software como el análisis y diseño de software o la programación. En esta fase, el docente propone situaciones o retos que

	exigen a los estudiantes interpretar requerimientos, analizar condiciones del problema, estructurar algoritmos, diseñar un diagrama de clases, implementar soluciones en Java y revisar su funcionamiento. Mientras desarrollan estas tareas, los estudiantes interactúan con el tutor inteligente para resolver dudas conceptuales, comprender mejor el enunciado, verificar si su interpretación del problema es adecuada, revisar alternativas de solución, mejorar sus artefactos de software (requerimientos, diseño y código) o identificar errores en sus propuestas. La IA se convierte así en un mediador del aprendizaje activo, pues ofrece apoyo durante el proceso, pero exige al estudiante analizar, decidir y justificar. Durante esta misma etapa, el papel del docente consiste en acompañar, observar y orientar. Aunque el tutor inteligente amplía las posibilidades de retroalimentación, el docente sigue siendo quien diseña la experiencia, interpreta pedagógicamente las necesidades del grupo y ayuda a los estudiantes a dar sentido a la interacción con la IA. También monitorea cómo está siendo utilizada la herramienta, identifica patrones en las consultas, reconoce dificultades recurrentes y ajusta sus explicaciones o actividades cuando es necesario. Un momento central de la práctica es la documentación de las interacciones con la IA. Como parte del trabajo académico, los estudiantes deben registrar los prompts empleados, las respuestas generadas por el tutor y, cuando aplica, una reflexión sobre la utilidad, pertinencia o limitaciones de dichas respuestas. Este componente convierte el uso de la IA en evidencia del proceso de aprendizaje y permite visibilizar no solo el producto final, sino también las decisiones, dudas y ajustes realizados por el estudiante a lo largo del desarrollo de la actividad. Esta documentación fortalece la transparencia, favorece la integridad académica y promueve habilidades metacognitivas. Finalmente, la experiencia incorpora una etapa de reflexión y mejora continua. El docente revisa las evidencias producidas, analiza la calidad de las interacciones con el tutor y valora en qué medida la herramienta contribuyó al logro de los aprendizajes esperados. A partir de este proceso, se identifican oportunidades de ajuste en las consignas, en la orientación brindada a los estudiantes y en la articulación entre el recurso tecnológico y el microcurrículo. De este modo, la práctica no se concibe como una intervención aislada, sino como un proceso de innovación pedagógica en permanente revisión, orientado a enriquecer la enseñanza y el aprendizaje en educación superior.
Articulación de la experiencia con el eje temático correspondiente	La experiencia se relaciona directamente con el componente Aprendizaje Experiencial e Innovación Curricular al transformar el microcurrículo del curso mediante la integración pedagógica de la inteligencia artificial como herramienta de acompañamiento al aprendizaje, situando a estudiantes y docente en un proceso de co-inteligencia. En primer lugar, la incorporación de tutores inteligentes en los cursos responde a una justificación pedagógica que trasciende lo tecnológico. Las consignas, los entregables y las orientaciones sobre el uso de IA se integran explícitamente en el diseño de las actividades, asegurando su alineación con los resultados de aprendizaje de los cursos.

	En segundo lugar, los estudiantes utilizan la IA mientras desarrollan artefactos de software a partir de problemas complejos, interactuando con el tutor para comprender requerimientos, mejorar sus diseños o depurar errores del código. De este modo, la IA actúa como mediadora del aprendizaje activo, sin sustituir el trabajo intelectual del estudiante, quien mantiene el control cognitivo del proceso al interpretar, evaluar y ajustar las respuestas generadas por el sistema en sus entregables. En tercer lugar, el uso de tutores inteligentes facilita una retroalimentación personalizada y continua, algo difícil de lograr en cursos con alta cantidad de estudiantes. Esto fortalece el seguimiento del proceso formativo y permite identificar tempranamente dificultades en el aprendizaje. Finalmente, las actividades requieren que los estudiantes documenten sus interacciones con los tutores (prompts y resultados), convirtiendo el uso de la tecnología en evidencia del proceso de aprendizaje. Esta práctica favorece la transparencia, la integridad académica y la reflexión crítica sobre el uso de la IA, promoviendo el desarrollo de la ética profesional, la autorregulación y la metacognición.
Argumentación de la innovación	El carácter innovador de esta práctica pedagógica radica en la integración intencional y curricular de la inteligencia artificial generativa como mediadora del aprendizaje, más allá de su uso instrumental o espontáneo por parte de los estudiantes. A diferencia de enfoques en los que la IA se incorpora únicamente como herramienta tecnológica, esta experiencia se fundamenta en un diseño pedagógico explícito que articula el uso de tutores inteligentes con los resultados de aprendizaje de los cursos y con los criterios de evaluación. La originalidad de la propuesta se encuentra en la creación e implementación de dos tutores inteligentes denominados “Turing” y “Lovelace”, concebidos como asistentes de aprendizaje que acompañan a los estudiantes durante el desarrollo de actividades de programación y análisis y diseño de software. Estos tutores no se limitan a generar respuestas o soluciones, sino que promueven procesos de orientación, explicación y retroalimentación, estimulando el razonamiento del estudiante y favoreciendo la construcción autónoma del conocimiento. Asimismo, la práctica incorpora el modelo AI Assessment Scale (AIAS) como marco pedagógico para orientar el uso responsable de la inteligencia artificial en las actividades académicas. Este enfoque permite establecer niveles claros de uso de IA, alinear la herramienta con los objetivos formativos y garantizar la integridad académica. De esta manera, la experiencia no solo introduce una tecnología emergente en el aula, sino que propone una forma estructurada y reflexiva de integrarla en el microcurrículo. Finalmente, la innovación se evidencia en la promoción de un modelo de co-inteligencia entre docentes, estudiantes y sistemas de IA, en el que la tecnología amplía las posibilidades de retroalimentación personalizada y favorece el desarrollo de competencias clave para el contexto profesional actual, como el pensamiento crítico, la ética en el uso de la IA y la metacognición.

Evidencias y Resultados de la Buena Práctica	La implementación de tutores inteligentes basados en inteligencia artificial generativa ha permitido evidenciar avances significativos en el acompañamiento al aprendizaje y en la calidad de las experiencias formativas dentro de los cursos Algoritmos y Programación I e Ingeniería de Software I. Una de las principales evidencias de la práctica se encuentra en las interacciones documentadas entre los estudiantes y el tutor inteligente, donde se registran los prompts utilizados, las respuestas generadas por la IA y las reflexiones de los estudiantes sobre su utilidad en la resolución de los problemas planteados. Estas bitácoras de interacción permiten observar cómo los estudiantes utilizan la herramienta para comprender requerimientos, mejorar la estructura de sus soluciones, identificar errores en el código y reflexionar sobre sus decisiones de diseño. En términos de beneficios para los estudiantes, la práctica ha contribuido a ampliar las oportunidades de retroalimentación personalizada, especialmente en cursos con un número considerable de participantes. El tutor inteligente actúa como un apoyo disponible durante el proceso de aprendizaje, permitiendo que los estudiantes reciban orientaciones inmediatas mientras desarrollan sus actividades. Esto favorece una mayor autonomía en el aprendizaje, fortalece la comprensión de los conceptos de programación y estimula el desarrollo de habilidades de pensamiento crítico frente a las respuestas generadas por la IA. Asimismo, la experiencia ha promovido una mayor reflexión sobre el uso responsable de la inteligencia artificial, al requerir que los estudiantes documenten sus interacciones con el sistema y analicen el papel que la herramienta desempeña en su proceso formativo. Este enfoque contribuye al desarrollo de competencias transversales como la metacognición, la autorregulación y la ética profesional, cada vez más relevantes en entornos académicos y profesionales donde la IA tiene una presencia creciente. En relación con la docencia, la experiencia ha permitido fortalecer las estrategias de retroalimentación formativa y enriquecer el diseño de actividades de aprendizaje mediadas por tecnología. Desde la perspectiva de investigación, la implementación del tutor inteligente abre oportunidades para analizar nuevas formas de interacción entre estudiantes y sistemas de IA en contextos educativos, así como para estudiar su impacto en el desarrollo de competencias propias de la ingeniería de software. Finalmente, en términos de proyección social, la experiencia contribuye a la formación de futuros profesionales capaces de utilizar herramientas de inteligencia artificial de manera crítica, responsable y alineada con principios éticos en su práctica profesional.
Medición e indicadores de impacto	La medición del impacto de la iniciativa se realiza mediante un conjunto de indicadores cuantitativos y cualitativos orientados a evaluar el uso pedagógico de la inteligencia artificial, el desarrollo de competencias y la calidad del proceso de aprendizaje. Estos indicadores se analizan a partir de diversas herramientas de seguimiento y recolección de evidencias implementadas en los cursos. En primer lugar, se utilizan bitácoras de interacción con inteligencia artificial, en las que los estudiantes documentan los prompts empleados, las respuestas generadas por el tutor inteligente y una breve reflexión sobre su

	utilidad en el proceso de resolución del problema. Estas bitácoras permiten medir indicadores como el nivel de uso de la IA en las actividades, la calidad de las interacciones con el tutor y el grado de reflexión crítica sobre las respuestas obtenidas. En segundo lugar, se analizan los productos académicos desarrollados por los estudiantes, tales como programas, modelos de diseño de software y reportes de solución de problemas. A partir de rúbricas de evaluación previamente definidas, es posible identificar indicadores relacionados con los resultados de aprendizaje de los cursos que apuntan por ejemplo al análisis del problema, la estructura del diseño de la solución o la correcta implementación del código. Adicionalmente, se emplean instrumentos de percepción estudiantil, como encuestas y cuestionarios de reflexión, que permiten identificar indicadores asociados con la utilidad percibida del tutor inteligente, la autonomía en el aprendizaje y el nivel de confianza en el uso responsable de la inteligencia artificial. Finalmente, el seguimiento del proceso formativo se complementa con el análisis de desempeño académico en las actividades del curso, comparando resultados entre diferentes momentos del semestre y evaluando tendencias en el desarrollo de competencias. Estos indicadores permiten valorar el aporte del tutor inteligente al fortalecimiento del aprendizaje y a la mejora de las prácticas de enseñanza mediadas por IA.
Recursos digitales	Enlace a experiencia (Turing: Tutor inteligente para el aprendizaje de programación, Los profes cuentan 2025, Universidad Icesi): https://sites.google.com/view/los-profes-cuentan-252/p%C3%A1gina-principal Video: https://www.youtube.com/watch?v=vz-7F3I75HA Enlace a los Tutores Inteligentes: Turing: Turing GPT Lovelace: Lovelace GPT
Palabras clave	Inteligencia artificial, Evaluación, Innovación pedagógica, Programación, Ingeniería de software, Educación superior, Aprendizaje activo, Tutoría inteligente
Referentes bibliográficos	Perkins, M., Furze, L., Roe, J., & MacVaugh, J. (2024). *The artificial intelligence assessment scale (AIAS): A framework for ethical integration of generative AI in educational assessment*. Journal of University Teaching & Learning Practice, 21(2). https://doi.org/10.53761/q3azde36 Perkins, M., Roe, J., & Furze, L. (2025). *Reimagining the artificial intelligence assessment scale: A refined framework for educational assessment in the age of generative AI*. Journal of University Teaching & Learning Practice. https://doi.org/10.53761/rrm4y757 Mollick, E., & Mollick, L. (2023). *Assigning AI: Seven approaches for students, with prompts*. arXiv.https://arxiv.org/abs/2306.10052

	Mollick, E., & Mollick, L. (2023). *Using AI to implement effective teaching strategies in classrooms: Five strategies, including prompts*. SSRN Electronic Journal. https://papers.ssrn.com/sol3/papers.cfm?abstract_id=4391243 Kouam, A. W. F. (2024). *The effectiveness of intelligent tutoring systems in supporting students with varying levels of programming experience*. Discover Education, 3, 278. Springer Nature. https://doi.org/10.1007/s44217-024-00385-3 Huang, X., Xu, W., & Liu, R. (2025). *Effects of intelligent tutoring systems on educational outcomes: A systematic analysis*. International Journal of Distance Education Technologies. https://doi.org/10.4018/IJDET.368420 UNESCO. (2023). *Guidance for generative AI in education and research*. UNESCO Publishing. https://www.unesco.org/en/articles/guidance-generative-ai-education-and-research

Biblioteca de Prompts

Índice

Prompt 1 – Tutor Inteligente para Algoritmos y Programación I
Prompt 2 – Tutor Inteligente para Ingeniería de Software I

Prompt 1 – Tutor Inteligente para Algoritmos y Programación I

Caso de uso	Prompt para la configuración de un tutor inteligente para el apoyo en la formación y retroalimentación de competencias de programación
Disciplina / Contexto	Algoritmos y programación
Objetivo del prompt	Generar un GPT personalizado que sirva como tutor inteligente de apoyo al aprendizaje del curso Algoritmos y programación I.
Audiencia	Estudiantes matriculados en el curso, Profesores que dictan el curso.
Texto del prompt	##Context 1. Turing is a world-class intelligent tutor who specializes in teaching introductory Java programming to Spanish-speaking students enrolled in Algoritmos y Programación I course at Universidad Icesi. 2. Turing helps clarify core programming concepts such as Java syntax, object-oriented programming, basic data structures, software engineering best practices and UML. 3. At the beginning of the course students solve problems using only one class. This class includes the main method, and all the other methods required. 4. Once the course introduces the concepts of Object-Oriented Programming, students will use a simplified MVC pattern where the 'View' handles input/output, input validation and also functions as the Main class (known as 'Executable'), and the 'Model' includes a 'Controller' that manages business logic. The 'View' can only interact with the 'Controller' using methods that use primitives and String as data types. ##Approach 1. Turing always communicates in Spanish and focuses on promoting understanding through detailed explanations and conceptual support. 2. Code snippets are provided only when explaining or to reinforce conceptual understanding. Turing MUST NOT provide a complete or partial solution to a problem even when the students ask for a concrete answer. 3. If a student provides code, Turing may only identify mistakes and provide method signatures but NEVER generates full code answers, encouraging students to practice and experiment to deepen learning. 4. Turing must not provide recommendations using the concept of

	"Exception in Java" unless specified by the students. ##Responses 1. Turing begins every interaction by introducing themselves. 2. If a student question is unclear, Turing asks clarifying questions. 3. At the end of each message, Turing checks whether the student has more questions or wants to end the conversation. 4. If the student chooses to end the conversation, Turing concludes by providing a summary of the discussion, highlighting the main questions and topics covered.
Formato de salida	No aplica
Última actualización	Diciembre 2025
Enlace al asistente	https://chatgpt.com/g/g-681a1f273e5c8191907dc322aded6464-turing
Notas	Se debe cargar como contexto del GPT personalizado los siguientes documentos: • Programa o syllabus del curso. • Java naming conventions • Comandos por consola (compilación, documentación, ejecución). • Material del curso. • Proyecto Educativo Institucional (PEI Icesi)

↑ *De vuelta al índice*

Prompt 2 – Tutor Inteligente para Ingeniería de Software I

Caso de uso	Prompt para la configuración de un tutor inteligente para el apoyo en la formación y retroalimentación de competencias de ingeniería de software (análisis y especificación de requerimientos, diseño detallado de software)
Disciplina / Contexto	Ingeniería de software
Objetivo del prompt	Generar un GPT personalizado que sirva como tutor inteligente de apoyo al aprendizaje del curso Ingeniería de Software I.
Audiencia	Estudiantes matriculados en el curso, Profesores que dictan el curso.
Texto del prompt	##Context 1. Lovelace is a world-class intelligent female tutor who specializes in teaching introductory software engineering to Spanish-speaking students enrolled in Ingeniería de Software I course at Universidad Icesi. 2. Lovelace helps clarify core software engineering concepts such as problem analysis, requirement specifications, software design for object-oriented programming, software engineering best practices according to the SWEBOK and UML (class diagrams). 3. Throughout the course students analyze problems related to software systems using a document format (course version of the Software Requirement Specification SRS) that includes: client, user, problem context (system purpose, user roles and a summary of the system's functionalities including characteristics and restrictions that are necessary for the system to properly function), a list of Functional Requirements (ID (i.e.: RF01) and name), a list of Non-functional Requirements (ID (i.e.: RNF01) and name) and a list of Process Requirements (ID (i.e.: RP01) and name). 4. Throughout the course students analyze software Functional Requirements using a document format (course version of the Software Requirement Specification SRS) that includes for each requirement: ID (i.e.: RF01) and name, summary (must be clear, accurate, concise, feasible, understandable, verifiable, quantifiable, complete, and have good spelling, punctuation, and grammar. It must include the inputs, activities, and conditions necessary to transform the inputs into outputs, the outputs, and the postcondition), list of inputs (name, data type, valid values), result or postcondition (the system state after the execution of the requirement) and list of outputs (name, data type, format (i.e.: yyyy-mm-dd for date types). 4. Once the course introduces the concepts of software design and

	Object-Oriented Programming, students will use a simplified MVC pattern where the 'View' handles input/output, input validation and also functions as the Main class (known as 'Executable'), and the 'Model' includes a 'Model Controller' that manages business logic. The 'View' can only interact with the 'Model Controller' using methods that use primitives and String as data types. 5. This course only uses association, dependency, realization and generalization relations between classes so Lovelace MUST NOT use or suggest other types of relations. ##Approach 1. Lovelace always communicates in Spanish and focuses on promoting understanding through detailed explanations and conceptual support. 2. Lovelace MUST NOT provide a complete or partial solution to a problem even when the students ask for a concrete answer. Lovelace responses should consider course's rubrics and provide feedback. 3. Code snippets (if needed) or class diagrams are provided only when explaining or to reinforce conceptual understanding. 3. If a student provides code, Lovelace may only identify mistakes and provide method signatures but NEVER generates full code answers, encouraging students to practice and experiment to deepen learning. ##Responses 1. Lovelace begins every interaction by introducing themselves. 2. If a student question is unclear, Lovelace asks clarifying questions. 3. At the end of each message, Lovelace checks whether the student has more questions or wants to end the conversation. 4. If the student chooses to end the conversation, Lovelace concludes by providing a summary of the discussion, highlighting the main questions and topics covered.
Formato de salida	No aplica
Última actualización	Marzo 2026
Enlace al asistente	https://chatgpt.com/g/g-67e9abadfb9c8191a61179c7b525bc29-lovelace
Notas	Se debe cargar como contexto del GPT personalizado los siguientes documentos: • Programa o syllabus del curso.

	• Material del curso. • Formato de Análisis y Especificación del Problema y los Requerimientos. • Rúbricas de Evaluación de las Unidades. • Proyecto Educativo Institucional (PEI Icesi).

↑ *De vuelta al índice*

BITÁCORA DE PROMPTS		
Fecha (yyyy-mm-dd)		
Herramienta o modelo IAG empleado	***¿Qué herramientas IAG se emplearon para desarrollar la tarea?*** *Ej.: ChatGPT, Claude, Turing, Lovelace, etc.*	

Objetivo de uso	Prompt empleado	Resultado generado por la IAG	Modificaciones aplicadas	Reflexión crítica
¿Qué quería lograr con el prompt? *Ej.:* • *Mejorar el código de un método.* • *Verificar la completitud de los requerimientos.*	*Incluir el prompt empleado para cumplir el objetivo.*	*Incluir o resumir lo que produjo el modelo IAG utilizado. Se puede pegar el enlace al chat con la herramienta seleccionada.*	*Explique cómo se modificó e integró el contenido generado por la IAG en el contexto de la actividad y según el objetivo de uso establecido.*	*En un párrafo responda:* • *¿Cómo evaluó la calidad y precisión del contenido generado por IAG antes de incorporarlo a su trabajo?* • *¿Encontró errores, sesgos o limitaciones en las respuestas de la IAG? ¿Cómo los abordó?* • *¿De qué manera la IAG le ayudó (o no) a mejorar la claridad o la estructura de su trabajo?* • *¿Qué decisiones tomó para asegurarse que su trabajo sigue cumpliendo con los objetivos de aprendizaje, a pesar del uso de IAG?*

Fecha de actualización del formato: marzo de 2026

Objetivos de aprendizaje a evaluar:
OE1.1. Resolver problemas siguiendo un proceso que incluye el análisis, diseño, codificación y pruebas.
OE1.2. Utilizar un ambiente de desarrollo (incluyendo la compilación y ejecución de programas desde consola) y un espacio de trabajo predefinido, para la codificación y ejecución de un programa.
OE1.3. Codificar en lenguaje Java la solución a un problema utilizando distintos tipos de datos, estructuras lineales, estructuras de control: instrucciones secuenciales, subrutinas, condicionales y repetitivas.
OE1.5. Resolver problemas que requieran estructuras de datos básicas como elementos de modelado que permiten agrupar elementos de un problema.

Sobre el uso de IAG:

Nivel de uso de IAG	Descriptor
3. Colaboración con la IAG	En esta actividad se puede utilizar cualquier herramienta / modelo basado en IAG para refinar, editar y mejorar cualquier requerimiento de la fase de codificación. Dentro de los entregables se debe incluir un documento de reflexión sobre su uso.

Fitplus

Carlos, un apasionado por el ejercicio, ha decidido llevar un registro más completo de sus sesiones de entrenamiento durante el mes. Cada vez que se ejercita, no solo registra los minutos de la sesión, sino también el día del mes (un número del 1 al 31). Carlos se ha enterado que usted está cursando Algoritmos y Programación I y le ha solicitado desarrollar una aplicación llamada Fitplus que cumpla con las siguientes funcionalidades:

1. Registrar la información de múltiples sesiones de entrenamiento hasta que el usuario decida salir del programa. Por cada sesión de entrenamiento:
    a. Solicitar el día del mes en que se realizó (número entero del 1 al 31).
    b. Solicitar la duración de la sesión en minutos (número entero positivo).
    c. Validar que el día esté entre 1 y 31 y que los minutos de entrenamiento sean mayores que cero. Si se ingresan datos inválidos, se debe mostrar un mensaje y no registrar esa sesión.
2. El sistema debe permitir **máximo una sesión de entrenamiento por día**. Si el usuario intenta registrar más de una en un día, debe mostrar un mensaje de advertencia y no realizar el registro.
   Restricción: No se permite el uso de colecciones para resolver este requerimiento.
3. Cuando el usuario decida salir, el programa debe mostrar un resumen de la información registrada:
    a. El número total de sesiones de entrenamiento registradas.
    b. El número total de minutos entrenados.
    c. Promedio de minutos por sesión de entrenamiento.
    d. El día del mes con la sesión de entrenamiento con la mayor duración.

Sobre el uso de IAG:

En esta actividad está permitido el uso de herramientas o modelos de inteligencia artificial generativa. Sugerimos hacer uso de Turing, Tutor Experto del curso. Recuerde documentar sus interacciones utilizando el formato bitácora de prompts. Para cada interacción en la bitácora significativa incluya:

- El prompt empleado.
- El resultado propuesto por el modelo.
- La reflexión crítica de uso. En un párrafo responder: ¿Qué funcionó? ¿Qué no? ¿Qué se modificó o descartó? ¿Por qué?
- Las modificaciones aplicadas al resultado propuesto por el modelo. En un párrafo explicar cómo se modificó e integró el contenido generado por la IAG en el contexto de la actividad y según el objetivo de uso establecido.

Entregables

1. **Usar GitHub como repositorio** de código fuente y documentación.
   Enlace al repositorio: XXXXXX
2. **Documentación**. Se deben incluir los contratos de los métodos implementados: descripción, parámetros y retorno.
3. **Reflexión sobre uso de IAG**. Se debe incluir diligenciado el documento bitácora de prompts.
4. **Implementación en Java:** Código con su propuesta de solución.

Rúbrica de evaluación

Proceso	Documentación		Codificación					Total
Uso de GitHub como repositorio de código fuente y documentación	Contratos de las subrutinas / métodos (ver Tabla 1)	Reflexión sobre el uso de IAG (ver Tabla 1)	Buenas prácticas de codificación (ver Tabla 2)	Compilación y Ejecución del Código (ver Tabla 2)	**Registro de sesiones de entrenamiento:** El sistema permite el registro de la información de múltiples sesiones de entrenamiento hasta que el usuario decida salir del programa (manejo de repetitivas). Por cada sesión de entrenamiento se guarda el día del mes en que se realizó (número entero del 1 al 31) y la duración de la sesión en minutos (número entero positivo). Se validar que el día esta entre 1 y 31 y que los minutos de entrenamiento sean mayores que cero. Si se ingresan datos inválidos, se muestra un mensaje y no se registra la sesión.	**Restricción de las sesiones de entrenamiento:** El sistema solo permite el registro de máximo una sesión de entrenamiento al día. En caso de que se intente realizar más de un registro al día se muestra al usuario un mensaje de advertencia y no se registra la sesión.	**Resumen de estadísticas:** Cuando el usuario decida salir, el programa muestra un resumen de la información registrada incluyendo: El número total de sesiones de entrenamiento registradas, el número total de minutos entrenados, el promedio de minutos por sesión de entrenamiento y el día del mes con la sesión de entrenamiento con la mayor duración.	
5,00%	**10,00%**	**20,00%**	**4,0%**	**4,0%**	**19,0%**	**19,0%**	**19,0%**	**100%**
5%	**30%**		**65,0%**					**100%**

Tabla 1

Nota	5	4	3	2	1
Contratos de las subrutinas / métodos	Todos los métodos están claramente definidos con contratos que especifican con precisión: qué hace el método, qué datos necesita (parámetros), qué valores retorna (si aplica), y cuáles son sus restricciones o precondiciones. La documentación es completa y coherente con la implementación. No se detectan ambigüedades.	La mayoría de los métodos tienen contratos bien definidos y comprensibles. Se especifican claramente las responsabilidades, parámetros y retornos. Puede haber una omisión menor en la redacción o documentación, pero no afecta la comprensión del propósito del método.	Los métodos tienen contratos parcialmente definidos. Se entiende la intención de la mayoría de ellos, pero falta claridad en algunos parámetros, su retorno o las pre o pos condiciones. La documentación es insuficiente o incompleta en varios casos.	La mayoría de los métodos carecen de contratos claros. No se especifican adecuadamente los parámetros ni lo que se espera como salida. La ausencia de documentación dificulta comprender el diseño del programa y sus subrutinas.	No se definen contratos. Los métodos aparecen sin documentación ni claridad sobre su funcionalidad, parámetros o retorno. La comprensión del propósito y uso de las subrutinas requiere revisar a fondo el cuerpo del código.
Reflexión sobre el uso de IAG	La reflexión ofrece un análisis crítico profundo y original sobre el uso de IAG, evaluando sus aportes y limitaciones en relación directa con decisiones tomadas durante la actividad de programación. Muestra comprensión conceptual y una integración clara con el proceso de desarrollo.	Presenta un análisis sólido del uso de IAG con ejemplos relevantes vinculados a la tarea. Muestra buen juicio crítico, aunque con menor profundidad o originalidad.	El análisis es mayoritariamente descriptivo y general. Reconoce el papel de la IAG pero con una conexión superficial o genérica con la tarea realizada.	La reflexión es limitada o poco clara. Menciona el uso de IAG pero sin analizarlo críticamente ni conectarlo de forma concreta con el ejercicio de programación.	No hay evidencia de análisis ni de relación con la tarea. El texto es vago o irrelevante respecto al uso de IAG.

Tabla 2

Nota	5	4	3	2	1
Buenas prácticas de codificación	- El código está correctamente indentado - El nombre de la clase empieza en mayúscula - El nombre de la clase es adecuado para el problema - Los nombres de las variables son adecuados (incluye los parámetros) - Los nombres de las variables empiezan en minúscula - Los nombres de las subrutinas empiezan en minúscula				
Compilación y Ejecución del Código	El programa compila y se ejecuta sin errores.	El programa compila, pero se presentan errores en tiempo de ejecución	Al incluir unas pequeñas modificaciones el programa compila	-	El programa no compila y hay que hacer modificaciones mayores para hacerlo compilar