



***Facultad  
de  
Ciencias***

**Rendimiento y Efectividad de la Inteligencia  
Artificial Generativa en el Desarrollo  
Software**

**(Performance and Effectiveness of  
Generative Artificial Intelligence in Software  
Development)**

Trabajo de Fin de Grado  
para acceder al  
**GRADO EN INGENIERIA INFORMATICA**

**Autor: Samuel Castro Rodríguez  
Director: Diego García Saiz  
Co-Director: Celestino Güemes Seoane  
Febrero - 2025**

## **Agradecimientos**

Quiero expresar mi más sincero agradecimiento a todas las personas que han formado parte de este viaje académico:

A mis tutores, Diego García y Celestino Güemes, por su guía y apoyo incondicional.

Al equipo completo de Worldgrid, por abrirme las puertas y brindarme su ayuda en cada momento necesario. A mis amigos y compañeros de profesión, quienes han hecho que el transcurso de esta carrera haya sido una experiencia amena y enriquecedora.

A mi padre, José Manuel, mi madre, María Amparo, y mi hermano, Eduardo, que han estado, están y estarán ahí siempre que los necesite. A mi abuela, América, por enseñarme con su ejemplo el verdadero significado de la resiliencia. Y a Alex, por mostrarme que cada momento, por difícil que sea, siempre vale la pena.

Os dedico este trabajo, hecho con todo el cariño y esfuerzo.

## **Resumen**

El impacto de la inteligencia artificial (IA) generativa en el desarrollo de software es un tema en auge en la investigación actual, impulsado por el creciente número de modelos de lenguaje de gran tamaño (LLMs) disponibles en el mercado, capaces de ofrecer asistencia en tareas clave de la informática. Este trabajo de fin de grado tiene como objetivo estudiar la aplicación de esta tecnología en el desarrollo software, en un entorno que refleja las condiciones habituales del sector.

El propósito de este estudio es proporcionar una visión clara y fundamentada sobre el estado actual de la IA generativa, destacando tanto sus capacidades como sus limitaciones en el entorno del desarrollo de software. Este trabajo ofrece un análisis detallado de su desempeño en escenarios prácticos, permitiendo a cualquier interesado comprender mejor su aplicación y utilidad.

Más allá de un análisis técnico, esta memoria explora cómo estas herramientas pueden transformar la forma de trabajar de los desarrolladores, agilizando diversas tareas del desarrollo software, como se ilustra mediante casos de estudio en este trabajo. No obstante, también se abordan los desafíos relacionados con la supervisión y validación de los resultados generados por la IA.

En definitiva, este estudio trata de proporcionar una comprensión profunda y práctica de cómo la IA generativa puede integrarse eficazmente en el desarrollo de software, ofreciendo valiosas recomendaciones, como es el tratamiento de prompts o las limitaciones y puntos destacables encontrado en los casos de estudio analizados, para maximizar su potencial y superar sus obstáculos en entornos empresariales.

## **Abstract**

*The impact of generative artificial intelligence (AI) on software development is a growing topic in current research, driven by the increasing number of large language models (LLMs) available in the market, capable of providing assistance in key computing tasks. This undergraduate thesis aims to study the application of this technology in software development, in an environment that reflects the usual conditions of the industry.*

*The purpose of this study is to provide a clear and well-founded view of the current state of generative AI, highlighting both its capabilities and limitations in the software development environment. This work offers a detailed analysis of its performance in practical scenarios, allowing any interested party to better understand its application and usefulness.*

*Beyond a technical analysis, this thesis explores how these tools can transform the way developers work, streamlining various software development tasks, as illustrated through case studies in this work. However, it also addresses the challenges related to the supervision and validation of AI-generated results.*

*Ultimately, this study seeks to provide a deep and practical understanding of how generative AI can be effectively integrated into software development, offering valuable recommendations, such as prompt handling or the limitations and notable points found in the analyzed case studies, to maximize its potential and overcome its obstacles in business environments.*

# Indice

1.	Introducción.....	6
1.1.	Qué es la IA Generativa.....	6
1.2.	Proyecto RIMA .....	7
1.3.	Motivación del Trabajo .....	8
1.4.	Objetivos.....	9
2.	Tecnologías Usadas .....	10
3.	Metodología.....	13
3.1.	Objetivo del Marco de Evaluación .....	13
3.2.	Proceso de Evaluación.....	13
3.3.	Sistema de Puntuación.....	15
3.4.	Base Metodológica .....	16
3.5.	Análisis de Resultados.....	17
4.	Desarrollo .....	18
4.1.	Introducción al Proyecto.....	18
4.2.	Tareas.....	18
4.3.	Optimización del Uso de la IA .....	19
4.4.	Documentación de Patrones y Resultados.....	19
4.5.	Papel y uso de MO4D en el Desarrollo .....	20
5.	Ingeniería del Prompt .....	20
6.	Resultados.....	22
6.1.	Generación de Código .....	22
6.2.	Documentación .....	29
6.3.	Pruebas unitarias.....	32
6.4.	Soporte.....	34
6.5.	Esquema de Resultados .....	37
6.5.1.	Generación de Código .....	38
6.5.2.	Documentación .....	38
6.5.3.	Pruebas unitarias.....	39
6.5.4.	Soporte.....	40
6.6.	Formulario a Empleados de Worldgrid .....	41
6.6.1.	Correlaciones y Puntos Destacables .....	44
6.7.	Estado del Arte .....	46
7.	Conclusiones.....	47

8. Líneas a Futuro .....	49
9. Referencias .....	50

**Palabras clave:** IA Generativa, Desarrollo Software, Programación Asistida por IA.

**Keywords:** *Generative AI, Software Development, AI-Assisted Programming.*

# 1. Introducción

En los últimos años, la inteligencia artificial (IA) ha transformado profundamente diversos sectores, y el desarrollo de software no ha sido la excepción. Entre las innovaciones más destacadas se encuentra la IA generativa [1] [2], una tecnología capaz de crear contenido a partir de datos existentes, con aplicaciones que van desde la generación de código hasta la documentación y las pruebas automatizadas. Estas herramientas muestran potencial para acelerar procesos, mejorar la calidad y reducir costes, pero también plantea preguntas fundamentales sobre potenciales riesgos en su uso y su impacto a largo plazo [3].

Este trabajo de fin de grado (TFG) tiene como objetivo analizar la efectividad y el rendimiento de herramientas de IA generativa en varias fases del desarrollo software, tomando como caso de estudio su utilización dentro del desarrollo de la aplicación RIMA (*Regulatory Interface Manager*), un proyecto de Worldgrid, empresa perteneciente al grupo Alten, y especializada en soluciones innovadoras para el sector eléctrico, que está comprometida con la búsqueda constante de eficiencia y optimización en sus productos, servicios y metodologías de trabajo.

Esta aplicación se centra en la gestión de ficheros normativos de empresas del sector energético, y será el escenario de aplicación de MO4D (*Machine One for Delivery*), una herramienta basada en IA generativa para apoyo a tareas de desarrollo, construida por Eviden. Durante el estudio, se evaluará cómo esta herramienta contribuye en tareas clave como la creación, refactorización y documentación de código, así como en el desarrollo de pruebas unitarias y en el soporte de tareas que lo requieran.

En cuanto al marco de trabajo técnico, la aplicación usada como escenario de las pruebas, RIMA, utiliza tecnologías modernas como módulos software de Java Spring, AWS (*Amazon Web Services*), bases de datos relacionales (MySQL y PostgreSQL), y contenedores con Docker, entre otras. Estas herramientas representan un entorno realista y robusto para medir cómo la IA generativa se integra en el flujo de trabajo de un equipo de desarrollo. La herramienta MO4D está basada en LLM (*Large Language Models*) de última generación, que consta de 3 de ellos: Azure AI, Claude y Gemini.

El trabajo está estructurado en tres partes principales. En los apartados 2 y 3 se describen las tecnologías y la metodología empleadas para realizar la evaluación. En el apartado 4 se analiza la utilización de la herramienta de IA generativa MO4D en el desarrollo de la aplicación RIMA, incluyendo en el apartado 5 un análisis detallado de la ingeniería de prompts, seguido de los resultados, en el apartado 6, obtenidos en distintos escenarios de utilización (documentación, desarrollo de pruebas, generación de código, etc.), además del análisis de los datos recopilados y las respuestas a un cuestionario realizado a un conjunto de empleados de Worldgrid. Finalmente, en el apartado 7, se presentan las conclusiones, abordando tanto los aspectos positivos como las limitaciones de estas tecnologías, y se proponen líneas futuras de investigación sobre la integración de la IA generativa en el desarrollo de software en el último apartado 8.

## 1.1. Qué es la IA Generativa

La inteligencia artificial generativa es una rama avanzada de la inteligencia artificial que se centra en la capacidad de crear contenido nuevo y original. A diferencia de las herramientas tradicionales de IA, que suelen analizar o interpretar datos existentes, los sistemas generativos van un paso más allá: son capaces de generar textos, imágenes, música, videos e incluso código de software que no existía previamente. Esto se logra a través de modelos de aprendizaje profundo, como los Transformadores (utilizados en modelos como GPT, *Generative Pre-trained Transformer*) y las

Redes Generativas Antagónicas (GAN) [4], que identifican patrones complejos en grandes volúmenes de datos para producir resultados que tratan de imitar la creatividad humana.

### **Large Language Models (LLM)**

Los LLMs son un tipo específico de modelo de IA basado en redes neuronales profundas, diseñado para procesar y generar texto natural con un alto nivel de coherencia y contexto. Estos modelos, como GPT, BERT (*Bidirectional Encoder Representations from Transformers*) y similares [5], han sido entrenados en vastos conjuntos de datos textuales para aprender las complejidades del lenguaje humano [6].

El desarrollo de los LLM se ha acelerado gracias a la disponibilidad de recursos computacionales avanzados y grandes volúmenes de datos digitales. Desde sus primeras versiones, estos modelos han evolucionado significativamente, mejorando su capacidad para comprender el contexto, responder preguntas y realizar tareas complejas como traducción automática, generación de código y creación de contenido personalizado [6].

Los LLM tienen aplicaciones en una amplia variedad de campos, incluyendo [7]:

- Desarrollo de software: Automatización de tareas repetitivas, generación de código, documentación y pruebas.
- Educación: Creación de materiales personalizados y simulaciones interactivas.
- Marketing y ventas: Generación de contenido publicitario y análisis de sentimiento en tiempo real.
- Atención al cliente: Creación de asistentes virtuales para soporte inmediato y eficaz.
- Ciencias biomédicas: Análisis de grandes conjuntos de datos genómicos y generación de hipótesis de investigación.

A pesar de su gran potencial, los LLM presentan limitaciones importantes. La calidad del contenido generado no siempre es precisa, y pueden producir resultados que carecen de coherencia o incluso contienen información incorrecta. Además, su entrenamiento requiere recursos computacionales significativos, lo que plantea preocupaciones sobre sostenibilidad y accesibilidad. Asimismo, existen riesgos éticos asociados, como la generación de desinformación o el uso indebido para fines malintencionados, como los *deepfakes* [8].

En el desarrollo de software, los LLM representan una herramienta transformadora que puede complementar el trabajo de los programadores, automatizando tareas básicas y permitiendo a los desarrolladores enfocarse en resolver problemas complejos y creativos.

## **1.2. Proyecto RIMA**

El proyecto RIMA surge de la necesidad de gestionar el ingente volumen de ficheros y datos que maneja el mercado eléctrico, requiriendo una estrategia organizativa que los haga útiles sin suponer una carga excesiva para las empresas del sector. Worldgrid, una empresa de Alten especializada en soluciones de este tipo, busca los mejores resultados para facilitar estas tareas de la manera más efectiva.

Este estudio se centra en la integración y evaluación de herramientas de inteligencia artificial generativa en el marco de este proyecto, un sistema funcional previamente desarrollado. Este proyecto destaca por su arquitectura robusta, diseñada con estructuras de clases y funciones bien definidas y consolidadas, lo que lo convierte en un entorno ideal para la implementación de mejoras y optimizaciones, aunque no para la creación de proyectos desde cero ni para el diseño de técnicas de desarrollo completamente nuevas.

La estructura de la aplicación se basa en una arquitectura modular desarrollada mediante Java Spring, encapsulando cada módulo en imágenes de Docker que posteriormente son desplegadas en AWS. Este enfoque, ampliamente adoptado en proyectos de software actuales, refleja las mejores prácticas de la industria y proporciona un entorno idóneo para llevar a cabo los casos de estudio. Esto no solo permite analizar el impacto de la IA generativa en un contexto realista y tecnológicamente relevante, sino que también asegura que los resultados obtenidos puedan ser extrapolados a una amplia variedad de proyectos similares en la actualidad.

El estado actual de este proyecto, en el momento en que se aplican herramientas de IA generativa en su desarrollo, abarca las siguientes fases del ciclo de vida del desarrollo de software [9]: implementación, pruebas, despliegue y mantenimiento. Las fases de análisis de requisitos y diseño y arquitectura quedan reservadas para futuras ampliaciones del estudio, debido a que el proyecto ya se encuentra en una etapa avanzada de desarrollo.

La importancia de este proyecto radica en que aborda cuestiones fundamentales sobre cómo integrar de manera efectiva herramientas de IA generativa en sistemas consolidados. Estos sistemas presentan retos específicos, como la necesidad de mantener la cohesión entre módulos, garantizar la compatibilidad con las dependencias existentes y preservar la calidad del código a medida que se introducen nuevas funcionalidades. Este tipo de análisis resulta especialmente relevante en un momento en el que las empresas buscan adoptar tecnologías de inteligencia artificial no solo como herramientas complementarias, sino como componentes estratégicos que transformen sus procesos de desarrollo.

En este contexto, la evolución del sistema se ha basado en la incorporación de nuevas funcionalidades, la refactorización de código para mejorar su mantenibilidad y eficiencia, la resolución de errores y la introducción de mejoras progresivas que garantizan su actualización constante. El objetivo principal de este trabajo es explorar cómo las herramientas de IA generativa pueden integrarse eficazmente en este tipo de entornos para maximizar la productividad, mejorar la calidad del código y optimizar los flujos de trabajo en equipos de desarrollo software.

### **1.3. Motivación del Trabajo**

En un entorno tan dinámico y competitivo como el actual, la inteligencia artificial ha emergido como un motor de cambio capaz de redefinir industrias. Su implementación no solo permite abordar retos complejos con mayor precisión, sino también potenciar la eficiencia operativa, algo esencial en un mercado como el eléctrico, donde la innovación marca la diferencia.

Dado el impacto transformador de estas tecnologías, Worldgrid no puede quedarse atrás. La integración de herramientas de inteligencia artificial en los procesos de desarrollo representa una oportunidad única para mejorar la calidad de sus entregables, optimizar tiempos de desarrollo y explorar nuevas capacidades tecnológicas.

Como futuro ingeniero informático, considero que adoptar soluciones basadas en IA es más que una tendencia; es un paso necesario para mantener la competitividad en un mercado global que exige rapidez, calidad y adaptabilidad. Este enfoque no solo alinea a Worldgrid con las demandas actuales del sector, sino que refuerza su posición como líder en soluciones tecnológicas avanzadas.

Mi interés en formar parte de este tipo de iniciativas en Worldgrid reside en la oportunidad de aplicar mis conocimientos en inteligencia artificial y desarrollo de software en un entorno donde la innovación tiene un impacto tangible en la sociedad. Además, colaborar con una empresa que combina experiencia tecnológica con un compromiso por liderar la transformación del sector

eléctrico es, sin duda, una motivación personal y profesional. Estoy convencido de que este proyecto no solo es una plataforma ideal para mi crecimiento, sino también una oportunidad para contribuir activamente a los objetivos de Worldgrid en su camino hacia la excelencia tecnológica.

## 1.4. Objetivos

La finalidad de este estudio es proporcionar una visión fundamentada sobre el estado actual de la inteligencia artificial generativa tanto en sus capacidades como sus limitaciones en un entorno real de desarrollo de software. A pesar de ser un campo en constante evolución, este trabajo ofrece un análisis detallado de su comportamiento en escenarios prácticos, permitiendo a cualquier persona interesada comprender mejor su aplicación y utilidad.

Más allá de un simple análisis técnico, esta memoria explora cómo estas herramientas están transformando la forma en que los desarrolladores trabajan, agilizando tareas como la generación de código, la documentación, generación de pruebas, etc., pero también planteando desafíos en cuanto a la supervisión y validación de sus resultados. Se abordan preguntas clave como: ¿Hasta qué punto la IA generativa mejora la productividad? ¿Cuáles son sus limitaciones en la toma de decisiones técnicas? ¿Cómo pueden los desarrolladores optimizar su uso para minimizar errores y alucinaciones?

En particular, este estudio resulta especialmente valioso para equipos de desarrollo que busquen ejemplos concretos y evaluaciones detalladas sobre el uso de estas herramientas en situaciones reales. A través de los casos analizados, se proporciona una guía práctica que puede servir como referencia para optimizar su implementación y aprovechar al máximo su potencial en la industria del software. Además, el estudio no solo se involucra en el impacto en la productividad, sino que también reflexiona sobre cómo el uso de estas tecnologías puede influir en la formación y evolución del rol del desarrollador en el futuro.

En el caso de desarrolladores en etapa inicial, como ingenieros informáticos junior o estudiantes de ingeniería, un uso excesivo de estas herramientas sin comprender el trasfondo de las respuestas que proporcionan puede limitar el aprendizaje técnico y convertirse en un problema a largo plazo. Esto puede tener implicaciones en los modelos docentes en estudios universitarios de ingeniería de software, para garantizar que los egresados puedan tener un dominio adecuado de estas herramientas [10] [11]. Un aspecto a tratar del análisis será abordar el debate actual sobre el impacto de estas tecnologías en el mercado laboral. Este TFG no sólo explorará las ventajas de la IA generativa, sino que también analizará su potencial para complementar el trabajo humano, así como los riesgos asociados a una dependencia excesiva.

Para enriquecer el análisis, se llevarán a cabo un formulario dirigido a desarrolladores de la empresa Worldgrid que han utilizado MO4D y otras herramientas de IA generativa, como Github Copilot, con el objetivo de recoger sus opiniones, experiencias y percepciones sobre estas herramientas. Esto permitirá identificar puntos fuertes y áreas de mejora, así como medir el impacto real de la IA generativa en la productividad, la calidad del trabajo realizado y la experiencia general del usuario.

En este contexto, el TFG abarca el estudio de estas herramientas en un entorno empresarial, evaluando tanto sus beneficios como sus desafíos. Además, pone en relieve el surgimiento de una nueva habilidad clave: la “ingeniería de prompts”. Basado en experiencias prácticas, esta memoria no solo pretende informar, sino también generar debate y reflexión sobre el papel de la inteligencia artificial generativa en el desarrollo de software, ofreciendo una base sólida para

quienes deseen profundizar en el tema e implementar estas herramientas de manera eficiente y responsable.

Los objetivos principales de este trabajo se organizan en los siguientes puntos:

**Evaluar la efectividad de la IA generativa en el desarrollo de software.**

- Se analizará su impacto en tareas como generación de código, generación de pruebas unitarias, generación de documentación y tareas referentes a soporte técnico.

**Analizar el papel de la validación humana en el uso de IA generativa.**

- Se examinará si la validación por parte de los desarrolladores es clave para corregir errores, interpretar correctamente las respuestas y garantizar su correcta aplicación en distintos entornos.

**Investigar la importancia de la formulación de prompts en la calidad de las respuestas de la IA.**

- Se explorará si un mejor contexto y una redacción más clara pueden reducir errores y alucinaciones, optimizando el uso de estas herramientas en el desarrollo de software.

**Reflexionar sobre el impacto de la IA generativa en la formación y el rol de los desarrolladores.**

- El uso creciente de estas herramientas plantea desafíos y oportunidades en la educación y el mercado laboral. Se abordará cómo pueden complementar el trabajo humano sin reemplazarlo, y qué habilidades deben desarrollar los futuros ingenieros para aprovechar su potencial sin depender excesivamente de ellas.

**Examinar el futuro de la IA en la industria del software.**

- Se tratará su posible estandarización en la industria, los desafíos que implica su integración y las mejores prácticas para su implementación efectiva en equipos de desarrollo.

## **2. Tecnologías Usadas**

Este Trabajo de Fin de Grado se centra en tecnologías clave que están presentes actualmente en el desarrollo de software, con un enfoque en la inteligencia artificial generativa y las herramientas colaborativas. Estas tecnologías son ampliamente adoptadas en una gran variedad de proyectos, lo que las hace especialmente relevantes para el estudio. A continuación, se describen las tecnologías utilizadas en el proyecto, destacando su papel en el desarrollo de este:

### **MO4D (*Machine One for Delivery*)**

MO4D es una herramienta de inteligencia artificial generativa desarrollada por Eviden, diseñada para optimizar el desarrollo de software mediante un enfoque de “programación por parejas” (*pair-programming*). Se basa en modelos avanzados de LLM como Gemini, Azure AI y Claude, permitiendo realizar tareas como la generación automatizada de código, la refactorización, la creación de documentación técnica y el diseño de pruebas unitarias, entre otras.

En este TFG, se utilizará MO4D para explorar su funcionalidad y medir su impacto en tareas de desarrollo y mantenimiento, priorizando su aplicación práctica sobre funcionalidades específicas.

## **Visual Studio Code (VS Code)**

Visual Studio Code es un editor de código fuente conocido por su versatilidad y capacidad de expansión mediante extensiones. Gracias a su diseño ligero pero potente, es ideal tanto para proyectos individuales como colaborativos. Su amplia biblioteca de herramientas permite adaptarlo a diferentes necesidades, convirtiéndolo en una plataforma popular entre los desarrolladores [12].

En este proyecto, se empleará VS Code como entorno principal de desarrollo, sirviendo también como plataforma de prueba para MO4D.

## **GitHub**

GitHub es una plataforma esencial para el desarrollo colaborativo tanto en entornos de desarrollo software empresariales como particulares, que combina la gestión de repositorios Git con herramientas avanzadas para equipos. Facilita el seguimiento de cambios, la gestión de versiones y la documentación del progreso del proyecto [13].

En este trabajo, se usará GitHub para documentar avances, versionar el código y compartir resultados.

## **Docker**

Docker es una plataforma que facilita el desarrollo, despliegue y ejecución de aplicaciones mediante el uso de contenedores. Un contenedor es una unidad autónoma que incluye la aplicación y todas sus dependencias, permitiendo que se ejecute de manera consistente en diferentes entornos, como el equipo de desarrollo, servidores de pruebas o producción [14].

La principal ventaja de Docker es la portabilidad de las aplicaciones. Al encapsular la aplicación con todo lo necesario para ejecutarse, se elimina el problema de las diferencias de configuración entre entornos. Esto asegura que la aplicación funcione de la misma manera en todos los sistemas.

Docker también es eficiente, dado que los contenedores comparten el núcleo del sistema operativo del host, lo que reduce el consumo de recursos y acelera el inicio de las aplicaciones. Esto hace que Docker sea ideal para entornos que requieren escalabilidad y flexibilidad [14].

En cuanto a desarrollo, Docker se utiliza en microservicios, permitiendo dividir aplicaciones grandes en componentes más pequeños y manejables. Cada microservicio se ejecuta en su propio contenedor, facilitando la actualización y el mantenimiento de las aplicaciones. Esta última característica coge más peso en un proyecto como en el que se basa este estudio, que utiliza microservicios.

## **Spring Boot**

Spring Boot es un framework de Java que simplifica el proceso de creación de aplicaciones basadas en el ecosistema Spring. Proporciona configuraciones predeterminadas y herramientas que permiten a los desarrolladores centrarse en la lógica de negocio en lugar de en la configuración de la infraestructura. Spring Boot es especialmente útil para el desarrollo de microservicios, ya que facilita la creación de aplicaciones modulares y escalables. Además, ofrece soporte para múltiples bases de datos, servicios web y sistemas de mensajería, lo que lo convierte en una opción versátil para el desarrollo de aplicaciones empresariales [15].

## **Amazon Web Services (AWS)**

AWS es una plataforma de servicios en la nube que ofrece una amplia gama de herramientas y servicios para el desarrollo, despliegue y gestión de aplicaciones. Entre los servicios más relevantes se encuentran [16]:

- **Colas SQS**

Este servicio permite la comunicación entre diferentes componentes de una aplicación mediante el envío, almacenamiento y recepción de mensajes. SQS es ideal para desacoplar y escalar microservicios, sistemas distribuidos y aplicaciones sin servidor, asegurando que los mensajes se entreguen de manera fiable y en el orden correcto.

- **S3**

S3 es un servicio de almacenamiento de objetos que permite almacenar y recuperar cualquier cantidad de datos en cualquier momento. Es altamente escalable y duradero, lo que lo hace ideal para copias de seguridad, almacenamiento de datos y distribución de contenido. Los datos en S3 se organizan en "buckets" y pueden ser accedidos a través de una interfaz web.

- **Cluster**

En el contexto de AWS, un clúster es un conjunto de instancias EC2 que trabajan juntas para proporcionar alta disponibilidad, escalabilidad y rendimiento. Los clústeres se utilizan en aplicaciones que requieren procesamiento paralelo o distribución de carga, como bases de datos distribuidas, aplicaciones de análisis de datos y sistemas de procesamiento de grandes volúmenes de información.

- **EC2 (*Elastic Compute Cloud*)**

EC2 proporciona capacidad de cómputo escalable en la nube. Permite lanzar y gestionar instancias de servidores virtuales, ajustando la capacidad según las necesidades de la aplicación. EC2 es flexible y ofrece una variedad de tipos de instancias optimizadas para diferentes casos de uso, desde cargas de trabajo ligeras hasta aplicaciones de alto rendimiento.

## **PostgreSQL**

PostgreSQL es un sistema de gestión de bases de datos relacional y objeto-relacional de código abierto. Es conocido por su robustez, extensibilidad y cumplimiento de estándares SQL. PostgreSQL ofrece una amplia gama de características avanzadas, como soporte para tipos de datos personalizados, índices avanzados, transacciones ACID y control de concurrencia multiversión (MVCC). Estas características lo hacen ideal para aplicaciones que requieren un alto grado de integridad de datos y consultas complejas. Además, PostgreSQL es altamente extensible, permitiendo a los desarrolladores añadir nuevas funciones, operadores y tipos de datos según sus necesidades [17].

## **MySQL**

MySQL es otro sistema de gestión de bases de datos relacional de código abierto, ampliamente utilizado en muchos sistemas que buscan un tipo de gestor de persistencia de sus datos rápida y fácil de gestionar, debido a su gran adopción y serialización. Es conocido por su rendimiento, facilidad de uso y amplia utilización en la industria. Además, MySQL es escalable y puede manejar grandes volúmenes de datos y usuarios concurrentes. Su integración con una amplia variedad de frameworks y lenguajes de programación lo convierte en una opción popular para el desarrollo de aplicaciones web y empresariales [18].

## **Java**

Java es un lenguaje de programación de propósito general, orientado a objetos y ampliamente utilizado en el desarrollo de aplicaciones empresariales, móviles y web. Una de las principales ventajas de Java es su capacidad para ser ejecutado en cualquier sistema que tenga una Máquina Virtual Java (JVM), lo que lo hace altamente portátil. Java es conocido por su robustez, seguridad

y rendimiento, y cuenta con un ecosistema rico de bibliotecas y frameworks que facilitan el desarrollo de aplicaciones complejas. Además, Java es una opción popular para el desarrollo de aplicaciones empresariales debido a su capacidad para manejar grandes volúmenes de datos y usuarios concurrentes [19].

### 3. Metodología

La evaluación del impacto de la inteligencia artificial en el desarrollo de software se realizará mediante un enfoque integral que combina la experiencia del ingeniero con un sistema estructurado de evaluación comparativa. Este proceso permitirá analizar y contrastar los resultados obtenidos en dos escenarios, si el caso de estudio lo permite: uno sin la asistencia de IA (método tradicional) y otro empleando herramientas de IA generativa.

En aquellos casos en los que no sea óptima la aplicación de la técnica de *A/B Testing* [20], se realizará un análisis cualitativo a cargo del ingeniero responsable, complementado con una evaluación basada en los criterios descritos en la tabla correspondiente a cada caso.

El análisis se basa en una guía de evaluación detallada, complementada con tablas estructuradas incluidas en el Anexo II, que definen los criterios de desempeño relevantes. Estas tablas están organizadas en categorías como generación de código, documentación, pruebas y soporte, y desglosan subcriterios específicos, tales como cohesión del código, claridad, funcionalidad y seguridad. Este enfoque asegura una evaluación sistemática y objetiva de los resultados obtenidos.

#### 3.1. Objetivo del Marco de Evaluación

El objetivo principal de este marco es proporcionar una metodología clara, estructurada y replicable para analizar el impacto de las herramientas de IA generativa en distintas fases del desarrollo de software. Además, permite identificar áreas de mejora tanto en el uso como en las capacidades de estas herramientas, asegurando que su adopción sea eficiente y segura en entornos reales de trabajo.

#### 3.2. Proceso de Evaluación

Cada caso de estudio abordado en este trabajo se analizará bajo una metodología estructurada, como se puede observar en la ilustración 1, que permite evaluar el impacto de la inteligencia artificial generativa en distintas fases del desarrollo de software. Sin embargo, es importante destacar que no todos los casos incluirán cada uno de los puntos expuestos en la metodología, sino que su aplicación dependerá de la naturaleza específica de cada caso.

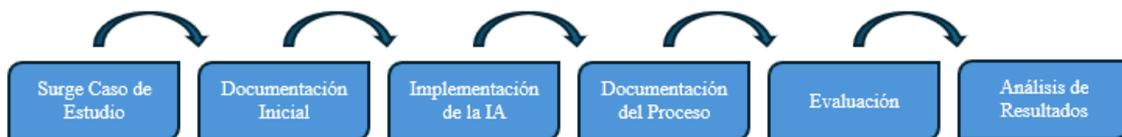


Ilustración 1. Proceso de Evaluación.

Dependiendo del contexto y del objetivo de cada análisis, se emplearán distintas combinaciones de estos criterios, asegurando que la evaluación se adapte a las particularidades de cada escenario. Algunos casos pueden centrarse en la comparación entre el desarrollo con y sin IA, mientras que en otros la perspectiva personal del investigador jugará un papel más relevante.

Esta flexibilidad metodológica permite que cada caso de estudio sea evaluado con un enfoque adecuado, asegurando que las conclusiones extraídas sean representativas de su aplicación real. Así, el lector podrá encontrar ejemplos diversos que reflejan distintos usos y alcances de la IA generativa en el desarrollo de software, proporcionando una visión amplia y fundamentada sobre su impacto en la industria.

La metodología incluye estas fases a tener en cuenta:

### **1. Evaluación de Control y Prueba:**

Para analizar el impacto de la inteligencia artificial generativa en el desarrollo de software, se empleará un enfoque experimental basado en la comparación entre un grupo de control y un grupo de prueba, similar a los estudios empíricos en otros campos (como es el A/B Testing).

En primer lugar, se llevará a cabo la tarea sin asistencia de IA generativa, registrando métricas clave como el tiempo de desarrollo, la calidad del código y la eficiencia del proceso, segmentando su análisis según las características propias de cada caso. Esta fase establece una línea base que servirá como referencia para evaluar los resultados obtenidos con la intervención de la IA.

A continuación, la misma tarea se repetirá con la ayuda de una herramienta de IA generativa. Los resultados obtenidos serán evaluados según los criterios establecidos en su implementación previa sin IA de ese caso concreto.

Este enfoque estructurado permite realizar una comparación objetiva entre ambas metodologías, proporcionando una base sólida para medir el impacto real de la inteligencia artificial en el desarrollo de software.

### **2. Análisis cualitativo personal:**

Como estudiante de ingeniería informática optando por el título, ofreceré mis propios puntos de vista y razonamientos basados en lo aprendido a lo largo de mi carrera. Además, aplicaré los nuevos conocimientos adquiridos durante este estudio, respaldados por investigaciones y experiencias recopiladas durante estos meses de trabajo.

### **3. Evaluación detallada por subcriterios específicos:**

Cada fase del desarrollo de software será evaluada utilizando las tablas del Anexo II, que contienen criterios como cohesión del código, claridad, funcionalidad, seguridad, etc. El uso de estas tablas asegura una evaluación estructurada y sistemática, que complementa tanto el análisis cuantitativo como la retroalimentación cualitativa del ingeniero. Esta metodología permite un análisis exhaustivo y objetivo del impacto de la IA en el desarrollo de software.

### **4. Evaluación cualitativa y cuantitativa mediante formularios:**

A lo largo del proceso de desarrollo, se distribuirá un formulario a los ingenieros participantes para evaluar su experiencia con el uso de IA generativa en el desarrollo de software en general. Estos formularios no están enfocados en cada caso de estudio, sino en el uso general de la IA

generativa como herramienta en el proceso de desarrollo. A través de los formularios, los participantes podrán proporcionar retroalimentación sobre aspectos como la facilidad de integración de la IA, la mejora en la productividad, la calidad del código generado y su percepción sobre el impacto general de la IA en el proceso de desarrollo. Esta retroalimentación complementará los resultados y ofrecerá una perspectiva más completa del impacto de la IA generativa en el desarrollo de software.

### 3.3. Sistema de Puntuación

Para cada caso de estudio, se asignará una puntuación basada en el cumplimiento de los criterios establecidos en las tablas de referencia (Anexo II). Este sistema de evaluación servirá como base para el análisis detallado de cada caso de estudio, cuyos resultados se encuentran recogidos en el Anexo III.

Cada subcriterio se califica según un sistema de opciones múltiples:

- No: La IA no cumple con las expectativas mínimas. (+0 puntos)
- Parcial: La IA cumple parcialmente, pero hay margen de mejora. (+1 punto)
- Sí: La IA cumple completamente, mostrando una mejora significativa. (+2 puntos)

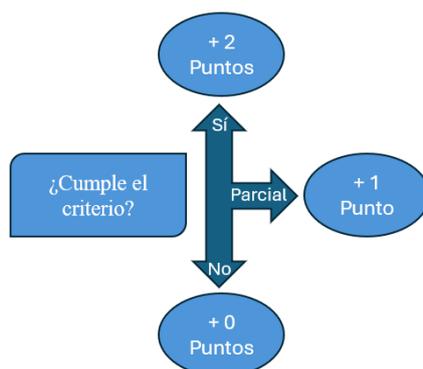


Ilustración 2. Asignación de Valor

La asignación de estas puntuaciones a cada opción se basa en la efectividad del uso de valores discretos en sistemas de evaluación. Dado que la evaluación se enfoca en determinar si un subcriterio se cumple completamente, parcialmente o no se cumple, como se puede observar en la ilustración 2, se utilizan valores discretos de 0, 1 y 2. Estos valores son los más adecuados para traducir las opciones textuales claras ("No", "Parcial", "Sí") en puntuaciones numéricas. Este método garantiza que los resultados obtenidos reflejen de manera precisa y objetiva el grado de cumplimiento de los criterios establecidos.

La ecuación que determina la puntuación de la IA en el caso de estudio que se analiza es la siguiente:

$$Puntuación = \frac{(2 \times (Puntos\ Sí) + (Puntos\ Parcial))}{(Puntos\ Posibles\ Totales)} \times 10$$

Ecuación 1. Cálculo de Puntuación de Caso de Estudio

### 3.4. Base Metodológica

El diseño de las tablas de evaluación (Anexo II) y la evaluación de cada caso de estudio (Anexo III) se alinea con principios establecidos por metodologías reconocidas en la evaluación de herramientas y procesos tecnológicos:

#### 1. Evaluación Basada en Métricas:

Siguiendo el enfoque GQM (*Goal-Question-Metric*), cada aspecto a evaluar se estructura mediante un objetivo específico, una serie de preguntas asociadas y métricas cualitativas que permiten analizar los resultados [21]. Ejemplo: en la tabla de generación de código, se incluyen criterios como "Facilita tareas tediosas" o "Cohesión en el código", cuyas respuestas ("No", "Parcial", "Sí") ofrecen una visión clara del impacto de la herramienta en el proceso.

#### 2. Evaluación por Checklists:

Este enfoque asegura que cada aspecto importante del desarrollo sea revisado de forma sistemática. Las tablas actúan como listas de verificación para evaluar desde la seguridad del código hasta la claridad de la documentación proporcionada por la IA [22]. La estructura está inspirada en prácticas empleadas en estándares como ISO/IEC 25002, que definen características clave para la calidad del software [23].

#### 3. Evaluación Comparativa (*Benchmarking*):

Además de analizar el desempeño de la IA generativa en tareas específicas, se realiza una comparación directa entre el desarrollo manual y el asistido por IA. Este enfoque resalta los beneficios y limitaciones de las herramientas generativas, identificando los escenarios donde su uso puede ser más efectivo [24].

#### 4. Experiencia del Usuario:

Se toma en cuenta la perspectiva del desarrollador, evaluando cómo la herramienta influye en su flujo de trabajo, productividad y comprensión del código.

El proceso de evaluación, combinado con una guía y la asignación de notas, proporciona una visión lo más objetiva posible, dentro de las limitaciones de este estudio, del impacto de la IA en distintas fases del desarrollo de software. La comparación entre los resultados obtenidos sin IA y con IA permite identificar áreas de mejora en el uso de herramientas de IA generativa y ajustarlas según sea necesario. Esto facilita medir la efectividad de la IA en todo lo que conlleva el desarrollo software.

Este enfoque se alinea con principios metodológicos de evaluación cuasi-experimental, como los descritos por Campbell y Stanley [25], en los cuales se busca establecer una relación causal entre la intervención (uso de IA) y el cambio observado (mejora en el desarrollo de software). En los casos de estudio donde es posible, la implementación de pruebas A/B fortalece la validez de los resultados, permitiendo tomar decisiones informadas sobre la adopción y optimización de herramientas de IA en proyectos de desarrollo de software.

Además, se han realizado casos de estudio diseñados para evaluar el desempeño de la inteligencia artificial en diferentes escenarios de uso, enfocándose específicamente en el resultado obtenido en base a distintas maneras de tratar el prompt de entrada. Este enfoque permite observar cómo varía el comportamiento de la IA según la información proporcionada en la solicitud inicial. A partir de estos análisis, se extraerán conclusiones y aspectos clave para desarrollar una buena práctica en la ingeniería de prompts, optimizando así el uso de este tipo de herramientas.

### 3.5. Análisis de Resultados

El análisis de los resultados de cada caso de estudio se ha llevado a cabo mediante una evaluación comparativa, contrastando los resultados obtenidos sin la asistencia de IA con aquellos generados mediante su uso. En los casos donde no sea viable esta comparación directa, se analizará el impacto del método empleado en el caso de estudio y su relación con los resultados obtenidos, como es el caso de la influencia de diferentes estrategias de prompting en la generación de código.

En la ilustración 3 se expone un esquema del análisis realizado en cada caso de estudio.

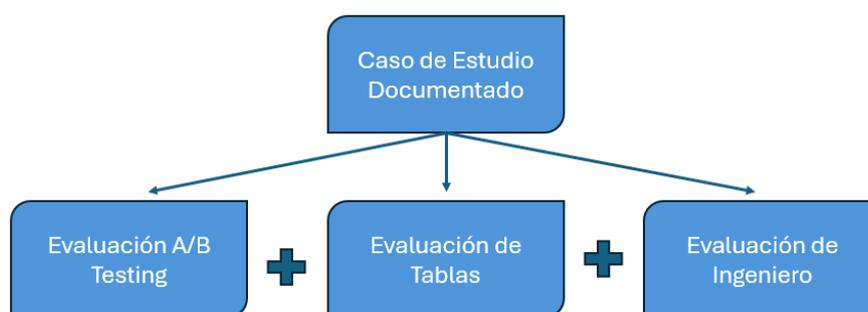


Ilustración 3. Organización Caso de Estudio

Las calificaciones asignadas en cada caso de estudio se basan en una puntuación detallada de los subcriterios evaluados. Estas puntuaciones están organizadas en tablas específicas según el tipo de caso de uso, tal como se estableció en el apartado de metodología (Sección 3.3) y se incluyen en el Anexo II, serán utilizadas para medir el impacto de la IA en aspectos clave como productividad, calidad del código, optimización del tiempo de desarrollo y facilidad de integración con herramientas existentes. Aunque este sistema cuantitativo es una parte importante del análisis, se complementa con la experiencia subjetiva del ingeniero, quien aporta una perspectiva cualitativa sobre la eficiencia, facilidad de uso e integración de las herramientas de IA.

Este enfoque metodológico garantiza que los resultados obtenidos sean tanto representativos como relevantes para evaluar el impacto real de la IA en entornos de desarrollo. El objetivo es determinar en qué medida la integración de la IA generativa contribuye a mejorar la calidad y eficiencia del proceso de desarrollo de software.

Asimismo, se realizarán análisis específicos sobre la importancia de un buen diseño de prompts y otros aspectos relevantes. Estos análisis buscan establecer pautas claras y prácticas que sirvan como guía para el uso eficaz de la inteligencia artificial generativa en el desarrollo de software. La integración de toda la información obtenida a lo largo de este estudio permitirá obtener conclusiones más enriquecidas sobre el efecto de la IA en el desarrollo de software, las cuales

contribuirán tanto a la mejora continua de las herramientas de IA como a ofrecer una perspectiva más amplia sobre su adopción en entornos reales de trabajo.

## 4. Desarrollo

A lo largo del proyecto, se han identificado varias fases clave en el análisis y la integración de la IA generativa en el desarrollo del sistema RIMA. Estas fases representan una evolución progresiva, desde el entendimiento inicial del sistema hasta la optimización avanzada de su uso, organizadas como se puede observar en la ilustración 4.



*Ilustración 4. Fases del Desarrollo*

### 4.1. Introducción al Proyecto

La fase inicial del estudio se centró en la comprensión del sistema RIMA. Esto implicó un análisis exhaustivo de su arquitectura, funcionalidades y componentes principales. Dado que el sistema ya estaba consolidado, fue necesario familiarizarse con sus bases técnicas, su estructura de clases y funciones, y las interdependencias entre los distintos módulos que lo conforman.

Este proceso de inmersión resultó ser una de las etapas más desafiantes del proyecto, ya que exigió no solo un análisis detallado del código existente, sino también la comprensión de las decisiones de diseño que dieron forma al sistema. Este conocimiento profundo fue esencial para garantizar que las intervenciones posteriores fueran coherentes con la arquitectura existente y alineadas con los objetivos del proyecto.

En esta etapa, la utilización de la IA sirvió para casos de estudio en los que se analizaban tareas tales como explicación de código y la documentación (véanse los casos de estudio 3, 5, 6 y 9, en el Anexo III).

### 4.2. Tareas

Con el fin de adquirir un entendimiento más amplio del sistema, se abordaron las primeras tareas asignadas. Estas incluyeron:

- **Documentación inicial:** Proporcionando así un entendimiento sobre el funcionamiento de las distintas implementaciones y/o servicios ya presentes.
- **Explicaciones sobre clases:** Para obtener explicaciones más detalladas de las clases que en el momento requería.
- **Generación de interfaces:** Estructurando los métodos disponibles a utilizar en la aplicación.

A medida que las tareas aumentaban y me familiarizaba más con el entorno de RIMA, las siguientes actividades me permitieron obtener un entendimiento más profundo del proyecto:

- **Desarrollo de nuevas funcionalidades:** La implementación de estas funciones requería un análisis previo del impacto en el sistema existente y la integración con los módulos ya establecidos.
- **Refactorización de clases extensas y complejas:** Este proceso se centró en mejorar la legibilidad, la eficiencia y el mantenimiento del código.
- **Implementación de pruebas unitarias:** Se diseñaron pruebas para garantizar que las nuevas funcionalidades y las mejoras introducidas no afectaran negativamente a la estabilidad del sistema.
- **Migración del motor de base de datos de MySQL a PostgreSQL:** Este proceso incluyó el análisis de compatibilidad, la adaptación de consultas SQL y la configuración del nuevo entorno.
- **Pruebas de regresión:** Se llevaron a cabo para asegurar que los cambios realizados, tanto en la migración como en el despliegue, no introdujeran errores en las funcionalidades existentes. Estas pruebas requirieron un esfuerzo considerable en la automatización y el diseño de escenarios representativos. Además, se realizaron algunas modificaciones en las codificaciones previas de la aplicación para corregir los errores detectados y mitigarlos eficazmente, lo que sirvió de escenario de prueba idóneo para la IA generativa.
- **Resolución de errores:** A través de los cambios realizados en la aplicación, el surgimiento de errores causales sobre alguno de ellos supuso que tuviese que estar involucrado en la resolución y debuggeo de los mismos.

En esta etapa, se documentó minuciosamente cada actividad, con un énfasis particular en el uso de la IA generativa como herramienta de apoyo en tareas como la generación de código, la optimización de algoritmos y la documentación técnica. Se analizaron también las interacciones con la IA en un entorno colaborativo, recopilando reflexiones y comentarios del equipo de trabajo para enriquecer el proceso.

### 4.3. Optimización del Uso de la IA

A lo largo de mi participación en el proyecto RIMA, el conocimiento sobre las capacidades y limitaciones de la IA generativa creció significativamente. Este aprendizaje permitió optimizar el diseño de prompts para interactuar con la IA, mejorando la precisión de las respuestas y reduciendo la necesidad de ajustes posteriores.

Se identificaron áreas específicas en las que la IA generativa demostró ser especialmente eficaz, como la creación de plantillas de código repetitivas, la generación de documentación técnica inicial y la sugerencia de mejoras en algoritmos existentes. Paralelamente, se reconocieron limitaciones, como la tendencia de la IA a generar soluciones genéricas en contextos muy específicos, lo que requería una supervisión continua por parte del equipo.

### 4.4. Documentación de Patrones y Resultados

Conforme la complejidad de las tareas avanzaba y el número de casos de estudio aumentaba, se identificaron patrones recurrentes en el uso de estas herramientas. Algunos de estos patrones incluían errores comunes en las respuestas generadas, similitudes en las sugerencias para problemas similares y aciertos destacados según el diseño de los prompts.

Esta fase se enfocó en documentar estos patrones, estableciendo mejores prácticas y recomendaciones concretas para el uso de la IA generativa en entornos similares. Este análisis permitió no solo mejorar la eficiencia en el uso de la herramienta, sino también construir un marco de referencia útil para futuros desarrolladores interesados en integrar la IA en sus flujos de trabajo a través de un buen uso de los prompts.

#### 4.5. Papel y uso de MO4D en el Desarrollo

El uso de esta herramienta como base del estudio fue implementada mediante su herramienta web, como las normalmente conocidas de ChatGPT, de OpenAI, o Copilot, de Microsoft. Esta herramienta presenta distintas opciones de uso para determinadas tareas que se requieran realizar. Esto es, si necesitamos generar código, esta herramienta tiene un agente dedicado para la generación de código, si necesitamos generar documentación técnica, tiene otro agente dedicado a esas tareas, y así con un amplio número de posibles tareas.

Al principio del desarrollo de este estudio, donde mi experiencia con este tipo de herramientas no era muy amplia, sí que utilizaba estos agentes dedicados a cada tarea en específico, pero a medida que iba realizando más casos, comprobé que prefería utilizar el agente de asistencia de código para la mayoría de las tareas referentes al desarrollo software, ajustando el prompt de entrada según el objetivo de la tarea del momento. De esta manera los resultados obtenidos eran más fieles a lo que buscaba como objetivo, aplicando mis conocimientos de tratamiento del *prompt*.

### 5. Ingeniería del Prompt

Además de los aspectos técnicos y de evaluación, el uso de IA generativa en el desarrollo de software requiere seguir ciertas buenas prácticas para maximizar sus beneficios. Uno de los factores clave es la creación de *prompts* adecuados, que permitan a los modelos de IA generar resultados más precisos y relevantes. La formulación de *prompts* efectivos no solo implica claridad en la solicitud, sino también especificar con detalle los requisitos y limitaciones de la tarea. Por ejemplo, como se discute en el caso de estudio 20 presente en el Anexo III, especificar claramente los detalles como el tipo de base de datos, restricciones sintácticas o el estilo de código esperado puede marcar la diferencia en la calidad del código generado.

A lo largo de este estudio he ido observando como he adaptado la manera en la que enfoco las peticiones que realizo a la IA, especificando más cuando se necesitaba o haciendo uso del contexto de la conversación para ajustar el camino por el que guiar a la IA.

He podido identificar varios factores clave que determinan que la respuesta obtenida sea lo más efectiva posible y, por consiguiente, obtenga los mejores resultados ahorrando así tiempo y esfuerzo. Los siguientes puntos con ejemplos de uso determinan este razonamiento:

#### 1. Aplicación de un contexto de desarrollo:

“Eres un experto en bases de datos y me vas a ayudar a solucionar el error indicándome el porqué de este fallo y como solucionarlo para que el funcionamiento de la consulta no varíe de un motor de bases de datos a otro.” (véase Anexo III, caso de estudio 19).

La IA generativa funciona mejor cuando opera dentro de un marco definido, ya que reduce la ambigüedad en las interpretaciones. Si el contexto es claro, el modelo puede alinear sus respuestas con el propósito de la solicitud, evitando resultados irrelevantes o fuera de

alcance. Esto asegura que las respuestas estén centradas en el dominio o rol asignado, aumentando la precisión y utilidad de las mismas.

2. **Exponer situación de iniciado:**

“Soy un iniciado en migraciones de MySQL a PostgreSQL de consultas y necesito que tú, como experto en este tipo de tareas y de bases de datos me realices la correspondiente migración de estas consultas que están en MySQL a PostgreSQL.” (véase Anexo III, caso de estudio 20).

Reconocer un nivel de experiencia limitado obliga al modelo a ajustar el nivel de detalle y la complejidad de sus respuestas. Esto hace que la información proporcionada sea más accesible y fácil de aplicar. De esta manera como usuario recibirás explicaciones detalladas y guiadas que no asumen conocimiento avanzado, mejorando el aprendizaje y la comprensión, y que, de otra forma, la IA asumiría determinadas explicaciones o puntos destacables al no tener en cuenta el nivel de detalle con el que explicar su solución dada.

3. **Listar requisitos y/o pautas de respuesta:**

“Los requerimientos son los siguientes:

- El resultado a obtener de la consulta debe ser el mismo que en MySQL.
- La sintaxis debe ser 100% compatible con PostgreSQL
- Los cambios aplicados deben ser los menos posibles de tal manera que si una consulta es compatible con MySQL, se quede igual.

La lista de consultas es esta: ...” (Véase Anexo III, caso de estudio 20).

Las IAs generativas no tienen la capacidad completa de inferir en los detalles implícitos de una solicitud. Establecer restricciones claras ayuda a guiar el modelo hacia soluciones que cumplen con los requisitos específicos. De esta forma reduciremos la probabilidad de errores y aseguraremos que las respuestas cumplan con los parámetros técnicos y funcionales establecidos.

4. **Proporción de ejemplos:**

Los casos en los que se requiere realizar tareas muy marcadas o con una estructura clara de realización, lo mejor y más efectivo es, además de acompañar de una explicación que introduzca el contexto de la tarea, se proporcione uno o varios ejemplos en los que se pueda basar (véase Anexo III, caso de estudio 12).

Los ejemplos actúan como puntos de referencia que el modelo puede seguir para generar respuestas más alineadas con las expectativas. Ayudan a reducir la variabilidad en las respuestas y aclaran cómo deben aplicarse las instrucciones.

5. **Ajuste del contexto de la conversación:**

Durante el desarrollo de varias consultas relacionadas con un tema específico, la IA aprovecha el historial de la conversación para contextualizar sus respuestas. Ajustar las solicitudes en función de factores como respuestas previas, errores detectados o patrones de comportamiento identificados puede ser clave para resolver problemas de manera efectiva.

Por ejemplo, en los casos de estudio 18, 20, 21 y 22 (Anexo III), se realizaron múltiples consultas relacionadas con cambios en un módulo de software. En este escenario, los

prompts fueron ajustados progresivamente, lo que permitió refinar las respuestas obtenidas y acercarse a soluciones más precisas y útiles.

Este enfoque es especialmente útil en tareas complejas que requieren varios pasos, donde las consultas simples y aisladas no serían suficientes para alcanzar el nivel de detalle y alineación necesario con los objetivos establecidos.

En definitiva, la inteligencia artificial generativa, tal como se presenta en la actualidad, actúa como un complemento para el desarrollador. Cuando se utiliza de manera adecuada, con un *prompting* bien diseñado y ajustado a la circunstancia, junto con un conocimiento sólido del área a tratar, los resultados pueden ser sumamente alentadores, mejorando significativamente la eficiencia de los procesos de desarrollo de software.

## 6. Resultados

En este apartado trataremos los resultados obtenidos tras el análisis detallado de cada caso de estudio presente en el Anexo III, los cuales están clasificados según su índole u objetivo que pretendían atender mediante el uso de la IA.

Resulta importante señalar que las calificaciones obtenidas en cada caso de estudio podrían no ser completamente representativas para los objetivos de este trabajo. Cada caso de estudio presenta características y contextos de desarrollo únicos, lo que implica que cada uno debe evaluarse teniendo en cuenta sus parámetros y particularidades específicas. Esto es en gran medida debido a que la complejidad de las tareas realizadas iba aumentando a medida que me iba adaptando al proyecto RIMA. Este incremento gradual en la dificultad de las tareas permitió obtener aprendizajes más profundos en las fases avanzadas del análisis, lo que resultó clave para identificar tanto las fortalezas como las debilidades de las herramientas de IA generativa en entornos donde la implementación de estas tecnologías implicaba desafíos técnicos significativos. En particular, la necesidad de supervisión humana se hizo más evidente en aquellas tareas donde la precisión y el contexto resultaban determinantes para la validez de los resultados obtenidos.

### 6.1. Generación de Código

El conjunto de casos de estudio analizados en el ámbito de la generación de código abarca una variedad de temas como la refactorización, la migración de código, el desarrollo de nuevas funcionalidades y las estructuras de servicios, entre otros. Los casos específicos incluidos en este análisis son: 1, 4, 7, 10, 14, 18, 19, 20, 21 y 22.

En la siguiente tabla 1, se presentan los resultados del ámbito de la generación de código, además de los casos en los que se introdujeron también tareas de soporte, como son los casos 17, 18, 19, 20, 21 y 22. Por simplicidad, he decidido agruparlos en este apartado y, por lo tanto, no aparecen detallados en los casos de soporte, presentes en el apartado 6.4.

Los resultados obtenidos se resumen en la siguiente tabla 1, aunque posteriormente se realiza un análisis detallado de los casos:

<b>Caso de Estudio</b>	<b>Resumen</b>	<b>Resultado</b>	<b>Puntuación</b>
1	Migración de consultas SQL de MySQL a PostgreSQL.	Efectividad en la conversión de consultas SQL.	6,67
4	Generación de una interfaz Java sobre una clase extensa.	Efectividad en la generación de una interfaz para una clase extensa.	9,17
7	Modificación de métodos de tratamiento de variables temporales.	Efectividad en la adaptación de métodos de fecha en migraciones de bases de datos.	8,33
10	Codificación de una funcionalidad nueva en la aplicación de RIMA.	Efectividad en tareas complejas y tediosas, como la generación automatizada de estructuras XML.	9,17
14	Codificación de consultas SQL para PostgreSQL.	Efectividad en la comprobación de formato PostgreSQL.	4,17
17	Migración módulo backend de MySQL a PostgreSQL.	Tratamiento del prompt adecuado con resultados óptimos.	9,08
18	Depuración de errores de ejecución en consultas PostgreSQL.	Inconsistencias a la hora de interpretar el contexto de la conversación.	6,66
19	Realización de una tarea simple.	Destaca el tratamiento del prompt, que determinó el acabar encontrando la respuesta correcta.	6,9
20	Migración de consultas MySQL a PostgreSQL.	Problema de inconsistencia a la hora de determinar las necesidades de añadir el tiempo de la zona en la fecha de las consultas.	8,83
21	Codificación de módulo software con Java y Spring Boot.	Codificación eficiente y efectiva para módulos software.	8,33
22	Migración de Scripts de creación de tablas y carga inicial de datos de MySQL a PostgreSQL.	Reducción del tiempo de codificación.	8

*Tabla 1. Resumen de Casos Generación de Código*

Por consiguiente, me centraré en detallar aquellos casos de estudio que han generado conclusiones más significativas o que han aportado una experiencia más relevante en el uso de la inteligencia artificial generativa, destacando su impacto y aprendizaje en cada contexto particular.

Los casos de generación de código que no obtuvieron resultados significativos fueron los casos 1, 4, 7 y 14. Estos casos, al corresponder a tareas iniciales de la aplicación con menor complejidad, no presentaban grandes desafíos técnicos, lo que explica su menor relevancia en términos de impacto. Aunque lograron una calificación media de 7.1 sobre 10, sus resultados eran previsibles. En cuanto a los tiempos de desarrollo medidos en estos casos, para el caso 1 el tiempo de desarrollo con IA fue de 15 minutos, comparado con 10 minutos sin IA, debido a la necesidad de

ajustar una consulta y en el caso 4 hubo una reducción significativa en el tiempo de desarrollo: 10 minutos con IA frente a 40 minutos sin IA. (véase en el Anexo III).

Sin embargo, el caso 14 plantea una situación más compleja, lo que lo hace más relevante para el análisis.

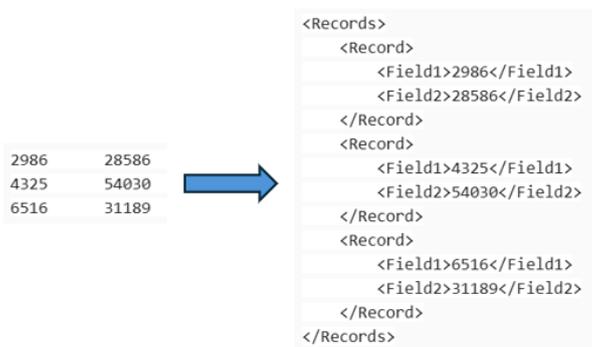
El **caso de estudio 14** evidenció que, aunque la IA generativa puede ser útil para automatizar tareas y generar código seguro, su inconsistencia en el manejo del contexto y los errores específicos, como en la compatibilidad de consultas PostgreSQL, limitaron significativamente su eficacia. Esto no solo incrementó el tiempo de desarrollo, sino que también resaltó la necesidad de una supervisión constante por parte de los desarrolladores para garantizar la calidad del código. Con una puntuación de 4,17 sobre 10, este caso resalta las limitaciones actuales de estas herramientas en escenarios donde la precisión y la cohesión son esenciales, enfatizando la importancia de su integración estratégica y supervisada en proyectos reales.

Por otra parte, los casos en los que se han obtenido resultados destacables son los casos 10, 18, 19, 20, 21 y 22.

Para estos realizaré un análisis detallado de cada uno en referencia a sus conclusiones y calificación:

- **Caso de estudio 10:**

Supuso el primer desarrollo amplio de código por mi parte en el proyecto RIMA, y marcó un punto de inflexión en mi experiencia con herramientas de inteligencia artificial generativa. En este caso, se abordó la implementación de una funcionalidad para transformar ficheros posicionales a XML, como se puede observar en la ilustración 5, creada como ejemplo visual, utilizando dos enfoques diferentes: un desarrollo inicial completamente manual y un segundo desarrollo apoyado por una herramienta de IA.



*Ilustración 5. Ejemplo Visual de Funcionalidad*

### **Desarrollo sin IA:**

El desarrollo manual implicó familiarizarse con los ficheros posicionales y las estructuras XML necesarias, tarea que resultó especialmente tediosa debido a inconsistencias en los datos disponibles y a la complejidad inherente de interpretar la documentación técnica. Además, la falta de familiaridad inicial con las librerías Java para lectura y escritura de ficheros (como Scanner y BufferedReader) alargó significativamente el proceso, requiriendo tiempo adicional para investigar y entender su uso. Este enfoque demandó un total de 3 horas para completar la

codificación, con una carga notable en la creación manual de estructuras XML y en la búsqueda de referencias técnicas.

### **Desarrollo con IA:**

El segundo desarrollo, asistido por la IA, destacó por una notable mejora en la eficiencia, reduciendo el tiempo de desarrollo a aproximadamente 1 hora y 30 minutos. Este resultado se logró gracias a las siguientes contribuciones de la herramienta:

- **Creación de estructuras XML:**

La IA permitió generar rápidamente la estructura del fichero XML solicitada al proporcionar ejemplos del formato de los datos posicionales y las especificaciones técnicas. Comparado con el enfoque manual, la IA redujo el tiempo invertido en esta tarea en más del 50%, transformando un proceso tedioso en una actividad mucho más ágil y directa.

- **Soporte en la codificación:**

La herramienta ayudó a identificar y aplicar librerías Java adecuadas para el manejo de ficheros, lo que permitió un flujo de trabajo más continuo.

- **Generación de documentación:**

La IA generó documentación detallada sobre la funcionalidad implementada, incluyendo una descripción de las principales partes del código. Sin embargo, surgieron errores en el esquema de flujos generado, debido a la falta de contexto global del proyecto, lo que limitó la utilidad de esta documentación. Este hallazgo refuerza la necesidad de explorar futuros desarrollos en herramientas de IA que permitan proporcionar un marco de trabajo más completo y contextualizado, siempre bajo estrictas políticas de seguridad y privacidad.

Con una calificación de 9.17 sobre 10, el caso de estudio 10 evidencia la efectividad de la inteligencia artificial generativa en tareas que implican la creación de estructuras complejas y tediosas, pero con una baja dependencia de otros componentes del sistema. Este alto desempeño se debe a que la IA fue utilizada en la integración de una nueva funcionalidad que no requirió cambios significativos en la codificación existente, sino que se enfocó en la ampliación de capacidades dentro del sistema. Esto sugiere que la IA es particularmente útil en escenarios donde su aplicación se limita a extender funcionalidades sin afectar de manera crítica la arquitectura del código ya establecido.

- **Caso de estudio 18:**

Este caso de estudio destaca la importancia de una correcta contextualización en los prompts al solicitar asistencia de la IA para la depuración de errores en consultas PostgreSQL. Durante la migración de consultas desde MySQL, surgieron distintos problemas que requirieron ajustes tanto en la formulación de las peticiones como en la evaluación de las respuestas generadas.

Uno de los principales aprendizajes fue que, al trabajar con JPQL (*Java Persistence Query Language*), el acceso a las tablas se realiza mediante los nombres de las variables en el código en lugar de los nombres reales de la base de datos. Esto causó un problema de reconocimiento de tablas que se resolvió tras identificar y corregir la fuente del error a la IA, contextualizándola.

Otro caso relevante fue el relacionado con el tipo de datos Timestamp. La falta de contexto en la consulta inicial llevó a que la IA generara una solución incorrecta. Sin embargo, tras ajustar el prompt para enfatizar la necesidad de tratar las fechas con Timestamp, la respuesta fue precisa y funcional.

En la tercera situación, al migrar una consulta de MySQL a PostgreSQL dentro de una clase Java con QueryFactory, la IA proporcionó una solución detallada y argumentada, pero que no generaba el resultado esperado. Tras señalar el error, la IA explicó su razonamiento y reconoció que no había reevaluado completamente el impacto de los cambios en la consulta, en particular en la cláusula HAVING.

Este caso enfatiza la importancia de:

- Redactar prompts bien contextualizados para obtener respuestas más precisas.
- No asumir que las respuestas de la IA son siempre correctas sin realizar pruebas de validación.
- Evaluar cómo los cambios en una parte de la consulta afectan a otras partes.
- Priorizar soluciones simples y funcionales sobre modificaciones innecesariamente complejas.

En términos de productividad, la asistencia de la IA redujo considerablemente el tiempo de depuración y corrección de errores, aunque sigue siendo fundamental una supervisión humana activa para validar cada solución propuesta.

Con una calificación de 6.66 sobre 10, este caso destaca las ventajas y limitaciones de la inteligencia artificial generativa en la depuración de consultas PostgreSQL. La IA resultó valiosa en términos de soporte, facilitando la identificación de errores y proporcionando soluciones contextualizadas. Sin embargo, su desempeño en la generación de código fue más discreto, con algunas respuestas incompletas o erróneas debido a la falta de contexto en los prompts iniciales.

- **Caso de estudio 19:**

Este caso de estudio pone de manifiesto cómo las herramientas de inteligencia artificial generativa pueden ser un valioso apoyo en tareas técnicas, como la migración de consultas SQL entre distintos motores de bases de datos. Sin embargo, también resalta la importancia de un enfoque colaborativo entre el desarrollador y la IA para garantizar resultados precisos y adaptados a las necesidades específicas del entorno de trabajo.

Uno de los aprendizajes más significativos radica en la necesidad de formular prompts claros y detallados. La experiencia demuestra que un prompt inicial que incluya información exhaustiva sobre las restricciones técnicas, como las reglas específicas de sintaxis del motor de base de datos destino, puede marcar la diferencia entre obtener una respuesta precisa o cometer errores. Aunque estos detalles pueden parecer obvios para un desarrollador experimentado, no deben darse por sentados al interactuar con una IA, como se comenta en el apartado 5 de esta memoria.

Asimismo, este caso refuerza la relevancia de la validación manual como un paso indispensable. Si bien la IA puede acelerar procesos y reducir el esfuerzo técnico, su comportamiento se basa en patrones estadísticos y no en una comprensión contextual profunda. Esto puede llevar a que genere respuestas aparentemente correctas pero que, en la práctica, no cumplan con todas las expectativas o requisitos. El hecho de haber identificado yo, y no la IA, un error relacionado con el uso de alias en una consulta UPDATE demuestra cómo la supervisión humana sigue siendo esencial para garantizar resultados fiables.

Un aspecto positivo a destacar es la capacidad de la IA para ajustar sus respuestas en función de la retroalimentación proporcionada. Este comportamiento evidencia el potencial de las herramientas generativas para aprender de iteraciones y perfeccionar sus propuestas. Sin embargo, este beneficio solo se materializa si el usuario está dispuesto a señalar errores y proporcionar indicaciones claras para ajustar el enfoque de la IA.

Finalmente, aunque las IAs generativas pueden facilitar enormemente tareas técnicas, su eficacia depende de cómo se utilicen. Crear prompts con ejemplos claros, detallar las expectativas y especificar las limitaciones técnicas del contexto son acciones que no solo optimizan el uso de la IA, sino que también contribuyen a minimizar errores y mejorar la calidad de las soluciones obtenidas.

En cuanto a la generación de código, la IA no logró acelerar significativamente el proceso de desarrollo, lo que se reflejó en la calificación de 5.8 puntos sobre 10. La cohesión del código fue parcial, con algunas inconsistencias en las respuestas, aunque la IA facilitó tareas repetitivas y generó código seguro y comprensible. No obstante, su funcionalidad no siempre cumplió con las expectativas debido a la falta de contexto y la necesidad de validación manual. Esto se evidenció, por ejemplo, en el error relacionado con el uso de alias en una consulta UPDATE, que fue identificado por el desarrollador y no por la IA.

Por otro lado, en el área de soporte, la IA obtuvo una calificación más alta, 8 puntos sobre 10. La asistencia fue precisa y útil, especialmente en la identificación de errores y en la contextualización de soluciones. La capacidad de la IA para ajustar sus respuestas en función de la retroalimentación proporcionada fue una de las fortalezas más destacadas, mostrando su potencial para aprender de iteraciones. Sin embargo, la falta de proactividad, la claridad parcial en las explicaciones y la necesidad de más ejemplos y referencias reflejan áreas de mejora.

Finalmente, este caso obtuvo una calificación media de 6,9 sobre 10.

- **Caso de estudio 20:**

Este caso de estudio también refuerza la importancia de un prompting detallado y estructurado al trabajar con modelos de lenguaje para la migración de consultas SQL entre MySQL y PostgreSQL. Se ha comprobado que proporcionar un contexto progresivo, detallando los requerimientos y las condiciones específicas del entorno, mejora significativamente la precisión y utilidad de las respuestas generadas por la IA.

Uno de los aspectos clave observados es que la IA puede generar soluciones técnicamente correctas pero que requieren ajustes en función del contexto de la aplicación. En este caso, el problema con la zona horaria en el uso de la cláusula INTERVAL en PostgreSQL fue un punto crítico. Aunque la solución propuesta por la IA era funcional en términos generales, fue necesario un pequeño ajuste en la parametrización de las fechas en el código Java para lograr el resultado esperado.

Otro aprendizaje relevante es que la interacción iterativa con la IA resulta clave para depurar y perfeccionar las respuestas obtenidas. A través de sucesivos prompts, se logró corregir y afinar las consultas migradas, optimizando el proceso de adaptación al nuevo motor de base de datos.

En cuanto a la generación de código, la IA logró acelerar significativamente el proceso de desarrollo, lo que se reflejó en la calificación de 9,17 puntos sobre 10. La cohesión del código fue alta, con pocas inconsistencias en las respuestas. La IA facilitó tareas repetitivas y generó código seguro y comprensible. No obstante, su funcionalidad no siempre cumplió con las expectativas debido a la necesidad de ajustes menores, como la parametrización de fechas en el código Java.

Esto se evidencia con el problema relacionado con la zona horaria en el uso de INTERVAL, que fue identificado y corregido mediante iteraciones sucesivas con la IA.

Por otro lado, en el área de soporte, la IA obtuvo una calificación alta, 8,5 puntos sobre 10. La asistencia fue precisa y útil, especialmente en la identificación de errores y en la contextualización de soluciones. La capacidad de la IA para ajustar sus respuestas en función de la retroalimentación proporcionada fue una de las fortalezas más destacadas, mostrando su potencial para aprender de iteraciones. Sin embargo, la falta de proactividad, la claridad parcial en algunas explicaciones y la necesidad de más ejemplos y referencias reflejan áreas de mejora.

La nota media obtenida en este caso de estudio es de 8,83 puntos sobre 10.

- **Caso de estudio 21:**

Este caso de estudio demuestra cómo la generación asistida de código mediante IA puede facilitar la adaptación de servicios existentes a escenarios simulados, como en este caso la consulta a un servicio web de la REE (Red Eléctrica Española) sin disponer del certificado necesario. A través de un prompting progresivo y estructurado, se ha logrado obtener un servicio funcional que reemplaza la consulta real por la lectura de un archivo JSON local, manteniendo la lógica original de la aplicación.

La interacción iterativa con la IA ha permitido ajustar la solución a las necesidades específicas del proyecto, refinando la implementación a medida que se añadía contexto adicional. Uno de los aspectos más relevantes ha sido la generación automática de la entidad para mapear el JSON, lo que ha reducido considerablemente el esfuerzo manual en la codificación y estructuración de datos. Sin embargo, se identificaron algunas imprecisiones en la representación de listas de objetos dentro del JSON, lo que requirió ajustes manuales para lograr una implementación completamente funcional.

Finalmente, este análisis reafirma que la IA puede ser una herramienta valiosa para acelerar el desarrollo y reducir la carga de trabajo en tareas repetitivas o estructurales, obteniendo una calificación total de 8,33 puntos sobre 10, aunque la supervisión y corrección humana siguen siendo indispensables para garantizar la precisión y adecuación de la solución al caso de uso específico.

- **Caso de estudio 22:**

Este caso de estudio refleja cómo la IA ha facilitado la migración de scripts SQL de MySQL a PostgreSQL, reduciendo significativamente la carga de trabajo asociada a la adaptación manual de la sintaxis y la estructura de inserciones de datos y estructura de las tablas con sus correspondientes características. A pesar de que la IA interpretó erróneamente una sintaxis no válida en PostgreSQL en la cláusula ON CONFLICT DO UPDATE, el error fue rápidamente corregido tras señalarlo en la conversación, lo que demuestra la capacidad del modelo para ajustar sus respuestas en base al *feedback* recibido.

El proceso de conversión de las inserciones fue en su mayoría correcto, requiriendo solo ajustes en el orden de carga debido a las claves foráneas y dependencias entre tablas.

En términos de productividad, el tiempo requerido para completar esta tarea se redujo drásticamente en comparación con una conversión manual, confirmando el valor de la IA como asistente en procesos de migración y adaptación de bases de datos.

Las calificaciones obtenidas fueron de 9,17 puntos de generación de código y 8 puntos de soporte, resultando en una nota final de 8,58 puntos sobre 10.

## 6.2. Documentación

La documentación es un componente esencial en el desarrollo de software, ya que facilita la comprensión del código, mejora la mantenibilidad de los proyectos y permite una colaboración más efectiva entre equipos. Con la introducción de herramientas de inteligencia artificial generativa, el proceso de documentación ha experimentado cambios significativos, optimizando la generación automática de descripciones, comentarios en código y documentación técnica estructurada.

En este apartado, se presentan los distintos casos de estudio en los que se ha evaluado el impacto de la IA generativa en la documentación del software. Se analizará cómo MO4D ha sido utilizada para asistir en la redacción de documentación técnica, desde comentarios en código hasta reportes detallados sobre la arquitectura y funcionalidad de los sistemas desarrollados.

Los casos de estudio referentes a la documentación son los siguientes: 3, 5, 6, y 9.

Los resultados obtenidos se resumen en la siguiente tabla 2, aunque posteriormente se realiza un análisis detallado de los mismos:

Caso de Estudio	Resumen	Resultado	Puntuación
3	Explicación de código sin contexto previo de su uso o funcionamiento.	Documentación inicial de un proyecto para la integración de un nuevo desarrollador. Reducción de tiempos notable.	8,33
5	Documentación de una clase Java extensa.	En proyectos grandes y con poca documentación, el uso de herramientas de automatización puede ahorrar un tiempo significativo y mejorar la claridad de la documentación.	10
6	Documentar una clase con comentarios en el código.	Documentación con necesidad de revisión debido a inconsistencias.	6,67
9	Explicación de una funcionalidad específica del código.	Entendimiento más rápido y claro de las funcionalidades a tratar, lo que resulta especialmente valioso para desarrolladores que entran a proyectos nuevos.	10

Tabla 2. Resumen de Casos de Documentación

- **Caso de estudio 3:**

Este caso de estudio se centra en la capacidad de la IA generativa para facilitar la comprensión de código en proyectos desconocidos para un desarrollador, especialmente cuando no se cuenta con documentación previa ni se ha ejecutado el entorno de desarrollo. La prueba consistió en proporcionar un módulo de software con clases extensas y estructuras con dependencias bien definidas para evaluar si la herramienta era capaz de explicar su funcionamiento de manera clara y estructurada. Además, se hicieron preguntas específicas sobre diferentes partes del código para verificar su capacidad de ofrecer respuestas precisas y detalladas.

Los resultados obtenidos muestran que la IA cumple un papel fundamental en la agilización del proceso de comprensión del código. Sin la herramienta, el análisis de las clases resultaría más laborioso y requeriría mucho más tiempo para extraer conclusiones sobre su funcionalidad. En cambio, la IA proporciona una descripción bien estructurada de los elementos del código, resumiendo la finalidad de cada clase y permitiendo que el desarrollador tenga una visión clara de su propósito sin necesidad de explorarlo manualmente en detalle.

Además de esta capacidad de resumen, la herramienta permite interactuar con ella de manera dinámica. Si se solicita información adicional sobre un método o una parte específica del código, la IA genera explicaciones más detalladas, facilitando así el proceso de aclaración de dudas. Esta posibilidad de indagar en aspectos concretos sin necesidad de recorrer manualmente todas las líneas de código representa una ventaja significativa, ya que permite que el desarrollador se enfoque en los elementos más relevantes sin perder tiempo en un análisis minucioso de cada componente.

Sin embargo, el estudio también revela ciertas limitaciones. La IA solo puede ofrecer información basada en el código proporcionado, por lo que, si el contexto del proyecto no se especifica correctamente, algunas explicaciones pueden resultar ambiguas o poco precisas. También se observa que, en algunos casos, la IA puede interpretar de forma incorrecta la funcionalidad de fragmentos de código, especialmente si estos incluyen patrones inusuales o si dependen de otros módulos que no han sido proporcionados.

A pesar de estas limitaciones, la conclusión general es que la IA generativa se muestra como una herramienta altamente útil para la documentación y comprensión de código en proyectos nuevos. Su capacidad para proporcionar explicaciones claras y responder preguntas específicas reduce considerablemente el esfuerzo requerido para entender el código y minimiza la carga cognitiva del desarrollador. No obstante, sigue siendo necesario un proceso de verificación por parte del programador para asegurar que la información proporcionada es correcta y aplicable dentro del contexto del proyecto en el que se está trabajando. En este caso la calificación ha sido de 8,33 puntos sobre 10.

- **Caso de estudio 5:**

En este caso de estudio se evaluó la capacidad de la IA generativa para documentar una clase Java extensa que desempeña un papel central dentro de su módulo. Dado que la clase en cuestión no contaba con documentación previa, ni con métodos refactorizados, ni con suficientes comentarios en su código, se buscó determinar si la herramienta era capaz de generar documentación clara y funcional de manera eficiente.

Para la prueba, se proporcionó la clase completa como entrada, y la IA respondió generando un documento que incluía la especificación detallada de cada método y de las diferentes partes de la clase, además de diagramas de flujo que representaban su estructura y funcionalidad. Este nivel de detalle permitió obtener una documentación mucho más clara y estructurada en un tiempo significativamente menor que si se hubiera hecho manualmente.

Uno de los principales beneficios observados fue la rapidez con la que se logró documentar la clase, algo especialmente relevante considerando su tamaño y la ausencia de comentarios explicativos en el código original. Sin la herramienta, este proceso habría requerido un esfuerzo considerable por parte del desarrollador, quien habría tenido que analizar línea por línea la funcionalidad de cada método y redactar la documentación de manera manual. En cambio, la IA automatizó gran parte de este proceso, facilitando un primer borrador detallado que agilizó el trabajo.

Sin embargo, el estudio también destaca un punto crucial: aunque la herramienta logra generar documentación de forma rápida y estructurada, siempre es recomendable revisar el documento resultante. Dado que la IA basa su documentación en un análisis sintáctico del código, pueden surgir imprecisiones si el código es particularmente complejo o si su funcionalidad depende de otros elementos que no fueron proporcionados como entrada, como se menciona en el anterior caso 3.

En conclusión, la experiencia muestra que las herramientas de automatización para la documentación pueden aportar un valor significativo en proyectos con clases extensas y con poca o nula documentación previa. Su uso permite ahorrar tiempo y mejorar la claridad del código, pero su eficacia depende de una supervisión manual que garantice la precisión y coherencia del contenido generado. La combinación entre la eficiencia de la IA y la verificación por parte del desarrollador resulta clave para lograr documentación de alta calidad.

La calificación final de este caso resultó en 10 puntos sobre 10.

- **Caso de estudio 6:**

En este caso de estudio se evaluó la capacidad de la IA generativa para documentar una clase Java sin comentarios previos en el código. Se proporcionaron métodos específicos a la IA para que generara comentarios pertinentes, y se comparó su efectividad con la documentación manual realizada por el desarrollador. La IA generó comentarios detallados y descriptivos, pero también añadió comentarios innecesarios que dificultaban la lectura del código. Además, la IA eliminó partes del código que, aunque no ejecutaban nada, marcaban una estructura potencialmente útil para el futuro. Obteniendo así una calificación de 6,67 puntos sobre 10.

Este caso destaca que, aunque la IA puede facilitar la documentación de clases extensas y complejas, es crucial revisar y ajustar los comentarios generados para asegurar su precisión y relevancia. La supervisión manual sigue siendo esencial para garantizar que la documentación sea útil y no introduzca confusión o errores en el código.

- **Caso de estudio 9:**

En este caso de estudio se evaluó la capacidad de la IA generativa para explicar la funcionalidad de un método específico de una clase que el desarrollador no conocía y que no estaba documentada. Se proporcionó el código de la función directamente a la IA, que respondió estructurando los distintos pasos de la ejecución del método, diferenciando cada posible caso de ejecución y resolviendo dudas específicas sobre su funcionamiento. Tras analizar la explicación proporcionada por la IA, se comprobó que la estructura y la descripción de las partes del método eran precisas y útiles. Esta introducción inicial facilitó al desarrollador la comprensión del método, permitiéndole añadir nuevas funcionalidades con mayor seguridad y eficiencia. Como consecuencia, este caso obtuvo una calificación de 10 puntos sobre 10.

Este caso destaca que, para desarrolladores que se incorporan a un proyecto sin contexto previo, la IA puede ser una herramienta valiosa para obtener rápidamente una visión general del código existente, mejorando la efectividad y reduciendo el tiempo necesario para familiarizarse con el proyecto.

### 6.3. Pruebas unitarias

Las pruebas unitarias juegan un papel clave en el desarrollo de software, asegurando que cada componente del sistema funcione de manera correcta e independiente antes de su integración. Su implementación eficiente reduce errores en etapas avanzadas del desarrollo, disminuye los costes de mantenimiento y contribuye a la estabilidad del software.

El uso de inteligencia artificial generativa en este ámbito ha permitido optimizar la creación de pruebas automatizadas, generando casos de prueba de manera más rápida y cubriendo escenarios que podrían pasar desapercibidos en un desarrollo manual. Además, la IA facilita la identificación de posibles fallos y sugiere mejoras en la cobertura del código.

En los siguientes casos de estudio, se analizará cómo MO4D ha sido empleada para asistir en la automatización de pruebas unitarias, evaluando su impacto en la detección de errores y en la mejora del proceso de validación del software.

Los casos de estudio relacionados con este apartado incluyen los siguientes: 11 y 15.

Los resultados obtenidos se resumen en la siguiente tabla 3, aunque posteriormente se realiza un análisis detallado de los casos:

Caso de Estudio	Resumen	Resultado	Puntuación
11	Pruebas unitarias de 2 clases Java, una extensa y otra pequeña.	La IA generó pruebas efectivas para una clase pequeña, pero las pruebas para una clase extensa fueron incompletas y no funcionales, requiriendo ajustes manuales.	6,25
15	Generación de pruebas unitarias con especial atención con "Mocks".	Pruebas unitarias sobre clases extensas con mucha dependencia sobre otras clases.	6,25

Tabla 3. Resumen de Casos de Pruebas Unitarias

- **Caso de estudio 11:**

En este caso de estudio, se solicitó a la IA que generara pruebas unitarias para dos clases Java: una reducida y otra extensa. La evaluación se centró en la efectividad y funcionalidad de las pruebas generadas.

Para la clase reducida, la IA logró crear una clase de prueba efectiva y funcional. Utilizó simuladores (Mocks) para evitar magic numbers y se aseguró de probar excepciones, métodos auxiliares y objetos retornados. Las pruebas no comprometieron la seguridad ni el estado del proyecto, cumpliendo con todos los estándares definidos. Tradicionalmente, este proceso tomó alrededor de 1 hora, pero con la ayuda de la IA, se redujo a solo 20 minutos.

Sin embargo, la situación fue diferente para la clase extensa. Esta clase tenía muchos métodos a probar, que dependían de métodos auxiliares de otras clases y presentaban una estructura compleja. La IA generó una clase de pruebas unitarias, pero esta resultó ser incompleta y no funcional debido a la complejidad y extensión de la clase. Tradicionalmente, la generación de pruebas para esta clase tomó aproximadamente 3 horas, pero con la IA, el tiempo se redujo a 2 horas.

Se revisaron los resultados destacando varios puntos importantes. En proyectos grandes como RIMA, que no tienen un gran número de pruebas unitarias y cuentan con muchas clases, la generación de pruebas unitarias requiere mucho tiempo y esfuerzo. Además, la implementación de nuevas clases de prueba puede ser complicada debido a problemas con librerías y dependencias del proyecto.

También se observó que, sin un buen entendimiento del funcionamiento de las clases, la integración de las pruebas generadas por la IA puede presentar problemas, especialmente con los datos de los objetos simulados (Mocks) y las librerías necesarias. Para clases muy extensas y con estructuras complejas, los resultados pueden ser insuficientes y erróneos.

Como propuesta de mejora, se sugiere introducir una herramienta que proporcione más contexto del programa a la IA, para que pueda tratar todas las clases involucradas al generar las pruebas unitarias. Si solo se necesita una estructura inicial para las pruebas, la herramienta cumple su cometido. Sin embargo, para delegar completamente la codificación de pruebas, es necesario que el desarrollador tenga el contexto y entendimiento necesarios.

En resumen, la IA puede ser útil para generar una estructura inicial de pruebas unitarias, especialmente para clases pequeñas y tareas específicas. Pero para proyectos grandes y clases complejas, es necesario proporcionar más contexto y entender bien el funcionamiento de las pruebas para asegurar su efectividad y funcionalidad. Este caso de estudio presenta una calificación de 6,25 puntos sobre 10.

- **Caso de estudio 15:**

En este caso de estudio, se solicitó a la IA que generara pruebas unitarias para varias clases de servicios Java Spring que utilizan una API (*Application Programming Interface*) a la que aún no se tiene acceso. La idea era ver si la herramienta MO4D podía ser útil en esta situación.

La IA generó una clase de prueba casi correcta, aunque hubo un par de constructores que no coincidían exactamente con los definidos. Al ejecutar y probar todos los tests, se observó que faltaban las comprobaciones o asserts para verificar los valores de retorno de las funciones. Después de varios intentos y correcciones, la respuesta de la IA se fue acercando a la solución esperada. Sin embargo, hubo un caso en el que la IA interpretó incorrectamente que no se esperaba una excepción, cuando en realidad se esperaba una *NullPointerException*.

Cabe destacar que, aunque la IA puede generar pruebas unitarias, ajustar las respuestas a lo que busca el desarrollador puede ser complicado. Una vez realizados los ajustes en los constructores y añadidos los asserts necesarios, la ejecución de las pruebas fue satisfactoria en varios métodos. Crear las pruebas unitarias manualmente lleva más tiempo que usando la IA. Tradicionalmente, este proceso tomó un total de 4 horas, pero con la ayuda de la IA, se redujo a 3 horas. Sin embargo, hubo problemas con las declaraciones de los métodos de prueba debido a la falta de constructores en las clases utilizadas en el servicio. También surgieron errores al modificar la región del cliente de AWS, y la IA proporcionó respuestas incorrectas en algunos casos.

En resumen, aunque es necesario dedicar tiempo a revisar el resultado proporcionado por la IA, esta herramienta puede ser efectiva para agilizar las pruebas en un entorno de desarrollo empresarial real. La IA reduce el tiempo de codificación, pero aumenta el tiempo de revisión del código y las pruebas. Estas herramientas son útiles para proporcionar estructuras de desarrollo, algoritmos básicos y guías de desarrollo, sirviendo como punto de partida o apoyo. Sin embargo, delegar todo el desarrollo en la IA puede causar errores y dificultar la cohesión del proyecto a largo plazo.

Este caso de estudio obtuvo una calificación final de 6,25 puntos sobre 10.

## 6.4. Soporte

El soporte técnico y la asistencia en el desarrollo de software son elementos fundamentales para garantizar la eficiencia en la resolución de problemas, la optimización de código y la correcta aplicación de tecnologías específicas. Desde la ejecución de comandos hasta la depuración de errores, contar con una fuente confiable de ayuda puede marcar la diferencia en la productividad y calidad del desarrollo.

Con la incorporación de inteligencia artificial generativa en este ámbito, los desarrolladores han podido acceder a soluciones más rápidas y adaptadas a sus necesidades. La IA no solo responde preguntas técnicas, sino que también puede sugerir mejoras, proporcionar ejemplos prácticos y ayudar a depurar código de manera más efectiva. Sin embargo, su utilidad depende de factores como la precisión de sus respuestas, la capacidad de contextualización y la claridad en sus explicaciones.

En este apartado, se analizarán los distintos casos de estudio en los que se ha evaluado el impacto de la IA en la asistencia al desarrollo. Se examinará cómo MO4D ha sido utilizada para proporcionar soporte técnico en diversos escenarios, desde la generación de comandos hasta la resolución de incidencias complejas en el código.

Los casos de estudio relacionados con este apartado incluyen los siguientes: 2, 8, 12, 13, 16, 17, 18, 19, 20, 21, 22 y 23.

En la siguiente tabla 4, se presentan los resultados del ámbito del soporte. Los casos 17, 18, 19, 20, 21 y 22 se explican en el apartado de generación de código, como comenté en el apartado de los resultados de generación de código (Sección 6.1).

Caso de Estudio	Resumen	Resultado	Puntuación
2	Depuración de problema en el código en un proyecto de Java con Spring.	Papel de la IA como guía para determinar pasos a seguir.	7,22
8	Depuración de un error en un método en java.	Optimización del proceso en un caso con una estructura de datos poco usual, mejorando la programación de una solución aplicable.	8,5
12	Error de Región de S3 entorno AWS.	Identificación de errores de sintaxis en parámetros de ficheros de propiedades y tratamiento de regiones de la herramienta de S3 de AWS.	9,5
13	Resolución de problema de extensiones de PostgreSQL.	Pérdida de contexto conversacional, tratamiento de extensiones de PostgreSQL (pg_crypto) y caso diferenciando el tratamiento de distintos LLMs.	8
16	Asistencia con Docker.	Guía del uso de herramientas como Docker.	10

23	Asignación de Driver Oracle a conexión a base de datos.	Error de tratamiento de versiones de dependencias de Maven.	6
----	---	---	---

Tabla 4. Resumen de Casos de Soporte

A continuación, se expone un análisis más detallado de los casos de estudio involucrados en el soporte:

- **Caso de estudio 2:**

En este caso de estudio, se evaluó la capacidad de la IA generativa para asistir en la depuración de un problema en un proyecto de Java con Spring Boot. El problema se centraba en un bloqueo en una función de una librería, sobre la cual se proporcionó a la IA el código y la información sobre el bloqueo, incluyendo el texto de la excepción encontrada y detalles sobre el punto del código donde se producía el problema.

La IA ofreció sugerencias de mejora y cambios, dividiendo la información en secciones según el tipo de solución posible y aclarando los pasos a seguir. Aunque la IA proporcionó una guía inicial útil, la efectividad de sus sugerencias dependió de la especificidad del prompt inicial. Un prompt general resultó en sugerencias menos precisas, requiriendo ajustes adicionales.

La IA fue capaz de identificar posibles causas del bloqueo y ofrecer soluciones detalladas, sirviendo como una herramienta complementaria para el desarrollador. Sin embargo, la depuración tradicional sigue siendo esencial para entender a fondo el problema. Usar puntos de ruptura y comprobar los valores de las variables y llamadas a funciones permite al desarrollador obtener una visión más clara del problema. En este caso la calificación obtenida fue de 7,22 puntos sobre 10.

- **Caso de estudio 8:**

En este caso de estudio, se evaluó la capacidad de la IA generativa para facilitar la depuración de un error en un método Java que utilizaba una librería de JDBC (*Java Database Connectivity*). El problema estaba relacionado con la obtención de la clave primaria mediante el método “getKey”, que retornaba múltiples claves en lugar de una sola, que era el comportamiento esperado en anteriores implementaciones presentes.

La IA proporcionó sugerencias y posibles implementaciones, pero debido a la falta de contexto completo, fue necesario guiarla e indicar dónde se creía que estaba el error. Tras varios intentos, la IA finalmente ofreció una solución correcta, aunque inicialmente hubo que ajustar el tipo de dato a retornar.

El proceso de depuración tomó 47 minutos con la participación de dos desarrolladores, quienes encontraron la IA útil para identificar la raíz del problema y guiarla hacia una solución. Este caso demuestra la efectividad del trabajo en equipo entre la IA y los desarrolladores, agilizando la resolución de problemas complejos, obteniendo una calificación de 8,5 puntos sobre 10.

- **Caso de estudio 12:**

En este caso de estudio, se evaluó la capacidad de la IA generativa para resolver un error de configuración de región en un módulo de descarga de ficheros en un S3 de AWS. El problema era que la región del S3 no coincidía con la definida en el fichero de propiedades.

La IA proporcionó una implementación mejorada del método “@PostConstruct” para ajustar la región del cliente S3, incluyendo la creación de un nuevo cliente y mensajes de log para facilitar

la depuración. Aunque la IA fue útil para identificar errores de sintaxis y proporcionar una estructura correcta, fue necesario guiarla con el contexto específico del problema. En este caso la nota obtenida fue de 9,5 puntos sobre 10.

- **Caso de estudio 13:**

En este caso de estudio, se evaluó la capacidad de la IA generativa para resolver un problema relacionado con la migración de MySQL a PostgreSQL, que requería la introducción de extensiones como “pg\_crypto”. El problema surgió porque la extensión pg\_crypto se instalaba en el esquema público, accesible desde el gestor de bases de datos, pero no desde la aplicación Java que utilizaba un esquema específico para la aplicación de RIMA.

A lo largo del proceso, se utilizó la IA para interpretar los errores del log y ajustar los prompts según se iban descartando posibles causas. Finalmente, se descubrió que la extensión pg\_crypto no estaba instalada en el esquema específico utilizado por la aplicación Java, lo que impedía su acceso.

La IA fue útil para proporcionar hipótesis y generar consultas, pero la intervención humana fue crucial para identificar la raíz del problema. La calificación de este caso fue de 8 puntos sobre 10, donde la experiencia del ingeniero y la capacidad de ajustar los prompts basándose en los resultados obtenidos fueron determinantes para encontrar la solución final.

- **Caso de estudio 16:**

En este caso de estudio, se evaluó cómo la IA generativa puede ayudar en la “dockerización” de módulos de software en un proyecto de desarrollo. Planteé diversas cuestiones relacionadas con los procesos de utilización de Docker.

La IA proporcionó respuestas correctas y explicativas, describiendo los pasos a seguir y las distintas opciones disponibles según la consulta realizada. Esto facilitó la comprensión y ejecución de las tareas necesarias para “dockerizar” los módulos de software.

Aunque como desarrollador presentaba varios conocimientos sobre la ejecución y el entorno de Docker, encontré que la IA fue de gran ayuda. Sin la IA, habría sido necesario leer documentación y guías de uso para aprender a ejecutar las tareas relacionadas con Docker. Es por eso que el proceso fue más fluido y eficaz, logrando una dockerización exitosa de los módulos de software.

Este caso destaca que, aunque las tareas de dockerización no son extremadamente complejas, la ayuda de la IA puede ser muy útil y recomendable, especialmente para aquellos sin experiencia previa. La IA puede proporcionar una guía clara y detallada, ahorrando tiempo y esfuerzo en la ejecución de comandos y configuración de herramientas específicas. Este caso obtuvo una calificación de 10 puntos sobre 10.

- **Caso de estudio 23:**

En este caso de estudio, se evaluó la capacidad de la IA generativa para asistir en la resolución de un problema al intentar cargar el driver del JDBC Oracle en un proyecto Java. El problema surgió cuando se intentó asignar el driver y se encontró con una excepción *java.lang.ClassNotFoundException*. Para abordar este problema, se proporcionaron dos prompts a la IA, detallando la configuración del código Java y la dependencia del POM (*Project Object Model*).

Inicialmente, la IA identificó correctamente que el problema estaba relacionado con la falta del Jar de la dependencia de Oracle. Sin embargo, al proporcionar una versión incorrecta de la dependencia (12.1.0.2 en lugar de 12.2.0.1), se hizo evidente la necesidad de verificar manualmente la información en el *Maven Repository*. Además, se descubrió que la versión del *Java Compiler* era inferior a la necesaria, complicando aún más la resolución del problema.

A pesar de estos desafíos, la IA proporcionó una guía inicial útil que ayudó a identificar la causa del error. No obstante, la resolución completa del problema requirió la intervención manual para verificar y corregir la información proporcionada por la IA. Este proceso subraya la importancia de mantener un sentido crítico y realizar búsquedas manuales cuando sea necesario.

Este caso de estudio destaca varios puntos importantes. Primero, la IA generativa puede ser una herramienta valiosa para proporcionar una guía inicial en la resolución de problemas, pero su efectividad depende de la precisión de la información que maneja. Segundo, la supervisión y validación humana son cruciales para garantizar la precisión y efectividad de las soluciones propuestas por la IA. Tercero, es esencial que los desarrolladores mantengan un enfoque crítico y estén dispuestos a realizar verificaciones adicionales para asegurar la correcta implementación de las soluciones. Finalmente, aunque la IA mostró flexibilidad al adaptarse a las necesidades específicas del proyecto, su capacidad para proporcionar soluciones precisas puede verse limitada por la información disponible durante su entrenamiento. Este caso obtuvo una calificación de 6 puntos sobre 10.

## 6.5. Esquema de Resultados

Media de puntuaciones de los casos de estudio:

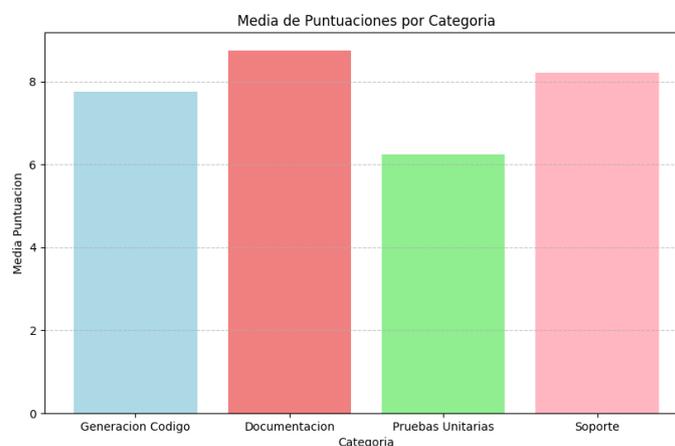


Ilustración 6. Media de Notas de los Casos de Estudio

Como se puede observar en la ilustración 6, la media de las notas de cada tipo de caso de estudio analizado presenta una tendencia notablemente positiva, reforzando el argumento de la utilidad y eficacia de estas herramientas de IA en los flujos de trabajo de la aplicación de RIMA según el marco de evaluación visto en el apartado 3.

A continuación, se detallan unos esquemas de los resultados obtenidos tras el análisis de los casos de estudio en los siguientes apartados, con el fin de proporcionar una guía esquemática de los resultados vistos en los anteriores apartados 6.1, 6.2, 6.3 y 6.4.

Los apartados presentados en cada tipo de tarea (generación de código, documentación, etc.) destacan y abordan los puntos más relevantes e interesantes de cada una.

### **6.5.1. Generación de Código**

#### **Eficiencia y Ahorro de Tiempo**

En tareas repetitivas y estructuradas, como la generación de código XML (caso 10) y la migración de bases de datos (casos 19, 20 y 22), la IA redujo significativamente el tiempo de desarrollo en comparación con el proceso manual. En el caso 10, el uso de IA disminuyó el tiempo de desarrollo de 3 horas a 1 hora y 30 minutos, destacando su utilidad en la creación de estructuras XML y soporte en la codificación. Además, en las migraciones de consultas SQL, la IA automatizó conversiones complejas, minimizando errores y mejorando la productividad.

#### **Calidad y Precisión del Código**

La IA generativa mostró limitaciones en la gestión del contexto y errores específicos, especialmente en consultas SQL y estructuras complejas (casos 14 y 18). A pesar de su capacidad para generar soluciones detalladas, en algunos casos sus respuestas contenían errores sintácticos o de compatibilidad con tecnologías específicas (caso 22). En el caso 14, la inconsistencia en consultas PostgreSQL llevó a una calificación de solo 4,17/10, evidenciando que la IA no siempre genera código funcional sin ajustes adicionales.

#### **Importancia del Prompting y la Interacción Iterativa**

La calidad de las respuestas de la IA mejoró significativamente cuando los prompts estaban bien estructurados y detallados (casos 18 y 19). Ajustar el contexto de las peticiones permitió corregir errores y mejorar la precisión de las soluciones generadas, lo que resalta la importancia de una interacción supervisada y progresiva. En casos como la adaptación de servicios web (caso 21), un prompting progresivo permitió obtener resultados más ajustados a las necesidades del proyecto.

#### **Limitaciones y Necesidad de Supervisión Humana**

Aunque la IA puede proporcionar soluciones iniciales, la validación manual sigue siendo crucial para garantizar la precisión y adecuación del código generado. En la migración de SQL, los errores en alias y uso de claves foráneas demostraron que la IA no siempre considera dependencias complejas sin ajustes manuales (casos 19 y 22). La documentación generada por la IA, aunque útil, en algunos casos presentó errores por falta de comprensión del contexto general del proyecto (caso 10).

**En resumen:** la IA demostró ser eficiente en tareas repetitivas, reduciendo significativamente el tiempo de desarrollo y mejorando la productividad. Sin embargo, mostró limitaciones en la precisión del código, especialmente en consultas SQL complejas. La calidad de las respuestas mejoró con prompts bien estructurados, destacando la importancia de la interacción iterativa. A pesar de sus ventajas, la supervisión humana sigue siendo crucial para garantizar la precisión y adecuación del código generado.

### **6.5.2. Documentación**

#### **Interacción Dinámica y Reducción de Carga Cognitiva**

La capacidad de la IA para interactuar dinámicamente con el desarrollador, respondiendo a preguntas específicas sobre partes del código, es una ventaja significativa. Esto permite aclarar

dudas sin necesidad de recorrer manualmente todas las líneas de código, lo que ahorra tiempo y esfuerzo (caso 3). Además, la IA ayuda a reducir la carga cognitiva del desarrollador al proporcionar explicaciones claras y estructuradas del código. Esto es especialmente útil en proyectos nuevos o desconocidos, donde la comprensión del código puede ser más desafiante (caso 3).

### **Generación de Diagramas y Documentación**

La capacidad de la IA para generar diagramas de flujo y otras representaciones visuales de la estructura y funcionalidad del código es un valor añadido. Estas representaciones pueden mejorar la comprensión y facilitar la comunicación entre los miembros del equipo (caso 5). Además, la IA puede proporcionar descripciones precisas y útiles de métodos específicos, lo que facilita la incorporación de nuevas funcionalidades y mejora la seguridad y eficiencia del desarrollo (caso 9). Es crucial revisar y ajustar los comentarios generados por la IA para asegurar su precisión y relevancia. La supervisión manual sigue siendo esencial para garantizar que la documentación sea útil y no introduzca confusión o errores en el código (caso 6).

### **Limitaciones y Necesidad de Ajustes Manuales**

La IA puede interpretar incorrectamente la funcionalidad de fragmentos de código si el contexto no se especifica correctamente. Esto resalta la importancia de proporcionar un contexto adecuado y de realizar ajustes manuales cuando sea necesario (caso 3).

**En resumen:** La combinación de la eficiencia de la IA y la verificación por parte del desarrollador resulta clave para lograr documentación de alta calidad y utilidad. La IA generativa es una herramienta valiosa, que resulta en una herramienta muy potente en el ámbito de la documentación.

## **6.5.3. Pruebas unitarias**

### **Eficiencia, Calidad y Precisión de las Pruebas**

La IA generativa ha demostrado ser eficaz en la creación rápida de pruebas unitarias, generando casos de prueba de manera más rápida y cubriendo escenarios que podrían pasar desapercibidos en un desarrollo manual. En proyectos grandes y complejos, la generación de pruebas unitarias requiere mucho tiempo y esfuerzo. La IA puede ser útil para generar una estructura inicial de pruebas unitarias, especialmente para clases pequeñas y tareas específicas (caso 11). Generó pruebas unitarias efectivas y funcionales para clases pequeñas, utilizando simuladores (Mocks) para evitar "magic numbers" y asegurando la prueba de excepciones, métodos auxiliares y objetos retornados (caso 11). Sin embargo, para clases extensas y con estructuras complejas, las pruebas generadas por la IA resultaron incompletas y no funcionales debido a la complejidad y extensión de la clase. Es necesario proporcionar más contexto y entender bien el funcionamiento de las pruebas para asegurar su efectividad y funcionalidad (caso 11).

### **Limitaciones, Necesidad de Supervisión Humana e Impacto en la Detección de Errores**

La integración de las pruebas generadas por la IA puede presentar problemas si no se tiene un buen entendimiento del funcionamiento de las clases, especialmente con los datos de los objetos simulados (Mocks) y las librerías necesarias (caso 11). Puede generar pruebas unitarias casi correctas, pero ajustar las respuestas a lo que busca el desarrollador puede ser complicado. Es necesario revisar y ajustar las pruebas generadas para asegurar su precisión y relevancia (caso 15). Además, la IA puede interpretar incorrectamente ciertos aspectos del código, como excepciones esperadas, lo que requiere correcciones manuales (caso 15).

Facilita la identificación de posibles fallos y sugiere mejoras en la cobertura del código, contribuyendo a la estabilidad del software. Aunque la IA reduce el tiempo de codificación, aumenta el tiempo de revisión del código y las pruebas. Es crucial revisar el resultado proporcionado por la IA para asegurar su efectividad en la detección de errores (caso 15).

**En resumen:** La IA generativa ha demostrado ser una herramienta valiosa para la automatización de pruebas unitarias, especialmente en la creación rápida de estructuras iniciales de pruebas. Su capacidad para generar casos de prueba y cubrir escenarios diversos puede ahorrar tiempo y mejorar la cobertura del código. Sin embargo, la supervisión manual sigue siendo esencial para garantizar la precisión y relevancia de las pruebas generadas. La combinación de la eficiencia de la IA y la verificación por parte del desarrollador resulta clave para lograr pruebas unitarias de alta calidad y efectividad.

#### **6.5.4. Soporte**

##### **Eficiencia, Calidad y Precisión en la Resolución de Problemas**

La IA generativa ha demostrado ser eficaz en la asistencia técnica, proporcionando soluciones rápidas y adaptadas a las necesidades de los desarrolladores. Esto incluye desde la ejecución de comandos hasta la depuración de errores, optimizando el tiempo y esfuerzo invertido en resolver problemas. En la dockerización de módulos de software, por ejemplo, la IA proporcionó respuestas claras y detalladas, facilitando la comprensión y ejecución de las tareas necesarias, especialmente para desarrolladores con conocimiento limitado sobre Docker (caso 16). En el caso de estudio 23, la IA ayuda a identificar la causa de determinadas excepciones, proporcionándola de contexto suficiente e información clave, aunque la resolución completa del problema requiere intervención manual.

La IA puede ofrecer sugerencias de mejora, ejemplos prácticos y ayuda en la depuración de código. Sin embargo, la precisión de sus respuestas depende de la especificidad del prompt inicial. Un prompt general puede resultar en sugerencias menos precisas, requiriendo ajustes adicionales (caso 2). En la resolución de problemas de configuración, como en el ajuste de la región de un cliente S3, la IA proporcionó implementaciones mejoradas y mensajes de log útiles, aunque fue necesario guiarla con el contexto específico del problema (caso 12).

##### **Limitaciones y Necesidad de Supervisión Humana**

La IA puede ser valiosa para proporcionar una guía inicial en la depuración de problemas, pero la depuración tradicional y la supervisión humana siguen siendo cruciales para garantizar soluciones precisas y adecuadas (caso 2). La colaboración entre la IA y los desarrolladores humanos es esencial para resolver problemas técnicos complejos y específicos. La experiencia del ingeniero y la capacidad de ajustar los prompts basándose en los resultados obtenidos son determinantes para encontrar la solución final (caso 13).

##### **Impacto en la Productividad y Calidad del Desarrollo**

La IA facilita la identificación de posibles causas de errores y ofrece soluciones detalladas, sirviendo como una herramienta complementaria para el desarrollador. Esto puede agilizar la resolución de problemas complejos y mejorar la calidad del desarrollo (caso 8). En la migración de bases de datos, la IA fue útil para interpretar errores del log y generar consultas, aunque la intervención humana fue crucial para identificar la raíz del problema (caso 13).

**En resumen:** La IA generativa ha demostrado ser una herramienta valiosa para el soporte técnico y la asistencia en el desarrollo de software. Su capacidad para proporcionar soluciones rápidas y

adaptadas a las necesidades de los desarrolladores puede mejorar la eficiencia y calidad del desarrollo. Sin embargo, la supervisión y el juicio del desarrollador siguen siendo esenciales para garantizar soluciones precisas y adecuadas. La combinación de la eficiencia de la IA y la verificación por parte del desarrollador resulta clave para lograr un soporte técnico de alta calidad y efectividad.

### **Conclusiones generales**

Las conclusiones generales que identifiqué tras la realización del estudio son las siguientes:

**La IA generativa es un excelente apoyo para la automatización de tareas repetitivas y la reducción de tiempos de desarrollo.**

- En la creación de estructuras de datos, generación de código y migración de bases de datos, la IA ha demostrado un impacto positivo en la productividad.

**La supervisión humana sigue siendo esencial para garantizar la calidad del código.**

- La IA no reemplaza la validación de un desarrollador experimentado, ya que sufre limitaciones en el manejo del contexto y en la precisión de consultas complejas.

**El éxito en la integración de la IA depende en gran medida de la claridad en los prompts y la retroalimentación iterativa.**

- Un prompting detallado y progresivo mejora la calidad de las respuestas y minimiza errores en la generación de código.

**La IA es más efectiva en entornos donde se puede supervisar y ajustar continuamente sus sugerencias.**

- Su integración estratégica en proyectos reales debe equilibrar automatización y control humano para maximizar su impacto.

## **6.6. Formulario a Empleados de Worldgrid**

El formulario planteado en este apartado fue creado con la finalidad de obtener información sobre el estado de la IA generativa y la experiencia de uso de cada trabajador de Worldgrid dio a estas herramientas en sus flujos de trabajo.

Durante una reunión posterior a la realización del cuestionario, se plantearon diversas preguntas y propuestas relacionadas con la incorporación de herramientas de inteligencia artificial en los flujos de trabajo de los profesionales interesados en utilizarlas. Uno de los puntos más destacados fue el interés por contar con una formación práctica basada en casos de uso reales, que permitan experimentar de primera mano las capacidades y beneficios de estas tecnologías.

Se enfatizó que, en lugar de centrarse únicamente en explicaciones teóricas sobre las funcionalidades de las herramientas o en enumerar su potencial, los profesionales valoran una capacitación enfocada en el "cómo":

- Cómo diseñar prompts efectivos para obtener resultados específicos.
- Cómo integrar estas herramientas en tareas concretas y cotidianas.
- Cómo resolver problemas reales utilizando ejemplos probados y ajustados al contexto particular de la empresa.

Esta aproximación permitiría no solo entender el alcance de la inteligencia artificial en el ámbito profesional, sino también adquirir habilidades prácticas que potencien la eficiencia y productividad en proyectos reales.

Por otra parte, entre las opiniones sobre el uso de estas herramientas de IA generativa de cada uno de los integrantes de la reunión en sus flujos de trabajo, destacaron varios casos:

1. **MO4D en ABAP (*Advanced Business Application Programming*) de SAP:** Las consultas realizadas a la IA generaban funciones inventadas sin un fundamento técnico correcto, resultando en programas que no llegaban a compilar. Esto dificultaba la integración de la IA para este tipo de tareas y/o codificaciones. Este caso resalta el potencial de un nuevo estudio para comprobar una posible correlación entre la poca información recopilada sobre estas tecnologías en el entrenamiento del LLM utilizado en cada momento y los resultados obtenidos.
2. **MO4D en PowerShell:** Sin tener un conocimiento extenso en la codificación de scripts para PowerShell, la IA proporcionó resultados funcionales y explicativos, guiando eficazmente en la creación de scripts.
3. **GitHub Copilot con Maven y Java:** En el tratamiento de dependencias de Maven, codificación en Java, automatización de tareas, optimización del código y búsqueda de comportamientos específicos en los logs, GitHub Copilot funcionó de manera satisfactoria, proporcionando asistencia útil y precisa.

Cabe destacar que, si bien los casos de estudio pueden ser revisados junto con los prompts de entrada y salida, así como la metodología utilizada con la IA, las opiniones expresadas no se basan en un rigor metodológico formal que las respalde. No obstante, representan perspectivas adicionales que pueden contribuir a futuras investigaciones y estudios en la materia.

Además, estos profesionales no utilizaron un procedimiento o técnicas de prompting marcadas que pudiesen obtener como resultado mejoras significativas en las respuestas.

En cuanto a los resultados del formulario planteado, cuyas cuestiones están presentes en el Anexo IV, los puntos destacables fueron los siguientes:

El número de personas que respondieron fue de 14 personas, obteniendo como resultado las siguientes tablas:

Rol	Porcentaje	Número de Participantes
Desarrolladores Backend	57%	8 de 14
Desarrolladores Frontend	14%	2 de 14
Fullstack	29%	4 de 14
DevOps	7%	1 de 14

Tabla 5. Formulario: Roles

Años de Experiencia	Porcentaje	Número de Participantes
1-5 años	29%	4 de 14
5-10 años	14%	2 de 14
10-20 años	29%	4 de 14
Más de 20 años	29%	4 de 14

Tabla 6. Formulario: Años de Experiencia

Lenguajes de Programación	Porcentaje	Número de Participantes
Java	86%	12 de 14
Python	14%	2 de 14
Otros lenguajes	29%	4 de 14

Tabla 7. Formulario: Lenguaje

Principales Tareas	Porcentaje	Número de Participantes
Desarrollar código	79%	11 de 14
Documentar	29%	4 de 14
Depurar	29%	4 de 14
Generación de pruebas	7%	1 de 14

Tabla 8. Formulario: Principales Tareas

Aspecto	Calificación Promedio
Funcionalidad del Código	3.64 sobre 5
Facilidad de Entendimiento y Mantenimiento	3.64 sobre 5
Cumplimiento de la Función Principal	3.64 sobre 5
Calidad del Código	3.43 sobre 5
Utilidad en Tareas	4.00 sobre 5

Tabla 9. Formulario: Aspectos Generales

→ **Calificación general:** 7,79 sobre 10.

#### Conclusiones de los resultados del formulario:

1. **Adopción y Uso:** Las herramientas de IA generativa están siendo adoptadas y utilizadas principalmente para el desarrollo de código y la resolución de tareas repetitivas.
2. **Experiencia y Satisfacción:** Los desarrolladores con diferentes niveles de experiencia, desde principiantes hasta expertos, están utilizando herramientas de IA generativa. La satisfacción general es positiva, con una calificación promedio de 7.79 sobre 10. Sin embargo, los desarrolladores con más experiencia tienden a ser más críticos con la calidad del código generado y la facilidad de integración.
3. **Calidad y Funcionalidad:** Aunque el código generado es funcional y útil en la mayoría de los casos, hay margen de mejora en términos de calidad y facilidad de mantenimiento. Las calificaciones promedio para la funcionalidad, facilidad de entendimiento y mantenimiento, y cumplimiento de la función principal son de 3.64 sobre 5, lo que indica una percepción positiva, pero con áreas para mejorar.
4. **Seguridad y Limitaciones:** No se encontraron fallos de seguridad, pero las herramientas tienen limitaciones en la interpretación de problemas complejos.
5. **Recomendaciones y Mejoras:** La mayoría de los encuestados recomendaría estas herramientas a otros desarrolladores, con un 86% de respuestas afirmativas. Sin embargo, se sugieren varias mejoras, como la capacidad de guardar un histórico de consultas (en el caso de MO4D), mejorar el orden y limpieza del código generado, y proporcionar más

ejemplos y referencias. Estas mejoras podrían aumentar la personalización y la calidad del código generado, mejorando aún más la experiencia del usuario.

### **6.6.1. Correlaciones y Puntos Destacables**

El siguiente análisis estadístico proporciona una visión representativa de las correlaciones observadas en los distintos campos evaluados en el formulario:

#### **1. Facilidad de Integración y Calificación General:**

Hay una alta correlación (0.89) entre la facilidad de integración de las herramientas de IA generativa en el flujo de trabajo y la calificación general de la experiencia con estas herramientas. Esto indica que una integración más sencilla mejora la percepción general de la herramienta. Esta correlación se puede ver representada en el apartado 6.6.2 en la figura 7.

#### **2. Calidad del Código Generado y Funcionalidad:**

Existe una fuerte correlación positiva (0.94) entre la calidad del código generado y la funcionalidad del código. Esto sugiere que cuando el código generado es de alta calidad, también tiende a ser funcional en la mayoría de los casos. Esta correlación se puede ver representada en el apartado 6.6.2 en la figura 8.

#### **3. Funcionalidad del Código y Facilidad de Entendimiento:**

La funcionalidad del código generado tiene una correlación positiva (0.78) con la facilidad de entendimiento y mantenimiento del código. Esto sugiere que el código que funciona bien también es más fácil de entender y mantener. Esta correlación se puede ver representada en el apartado 6.6.2 en la figura 9.

#### **4. Cumplimiento de la Función Principal y Utilidad en Tareas:**

Existe una correlación significativa (0.79) entre el cumplimiento de la función principal solicitada a la IA y la utilidad de las respuestas obtenidas para abordar las tareas. Esto muestra que cuando la IA cumple con la función principal, también es más útil para los usuarios en sus tareas. Esta correlación se puede ver representada en el apartado 6.6.2 en la figura 10.

#### **5. Calidad del Código y Calificación General:**

La calidad del código generado tiene una correlación positiva (0.81) con la calificación general de la experiencia con las herramientas de IA generativa. Esto indica que una mejor calidad del código contribuye a una mejor experiencia general. Esta correlación se puede ver representada en el apartado 6.6.2 en la figura 11.

#### **6. Facilidad de Integración y Facilidad de Entendimiento:**

La facilidad de integración de las herramientas tiene una correlación positiva (0.74) con la facilidad de entendimiento y mantenimiento del código generado. Esto sugiere que una integración más sencilla también facilita la comprensión y el mantenimiento del código. Esta correlación se puede ver representada en el apartado 6.6.2 en la figura 12.

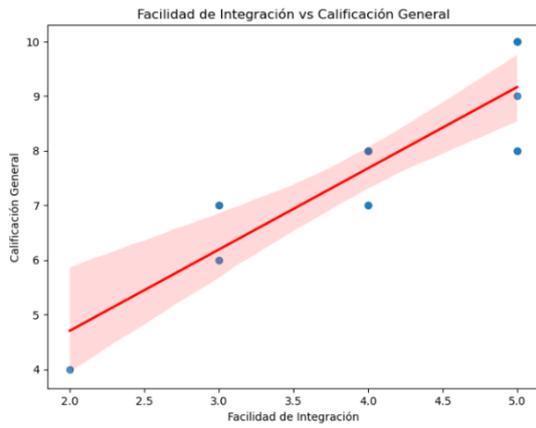


Ilustración 7. Regresión Lineal: Facilidad Integración y Calificación General

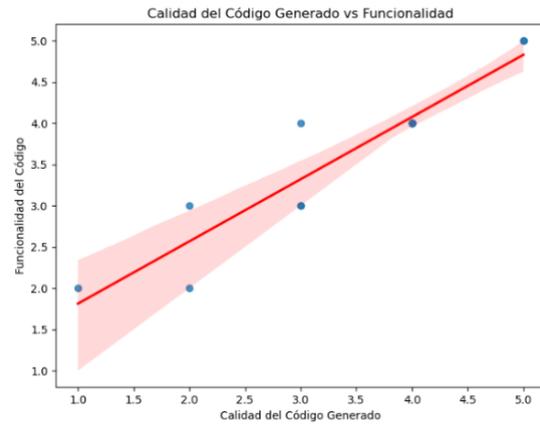


Ilustración 8. Regresión Lineal: Calidad Código y Funcionalidad

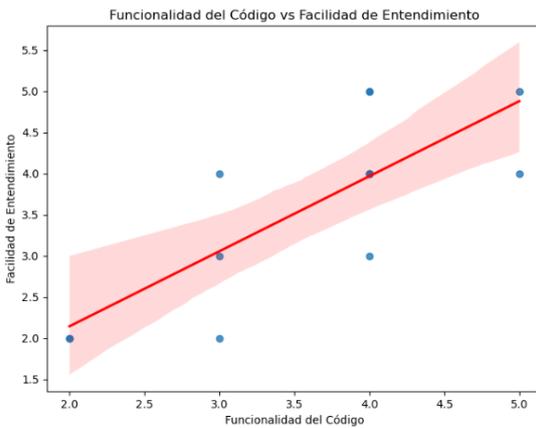


Ilustración 9. Regresión Lineal: Funcionalidad Código y Facilidad Entendimiento

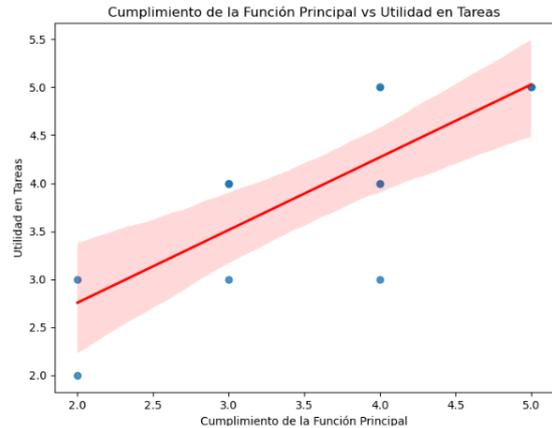


Ilustración 10. Cumplimiento Función y Utilidad de Tarea

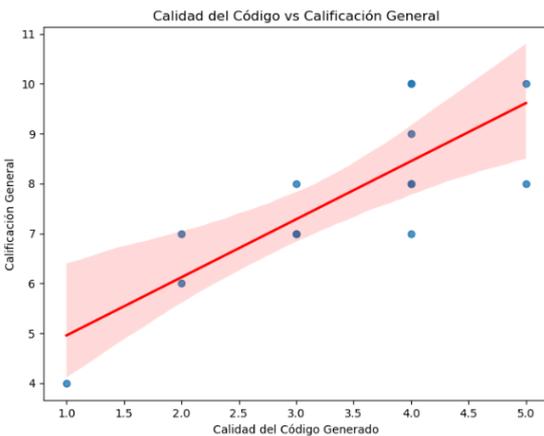


Ilustración 11. Regresión Lineal: Calidad Código y Calificación General

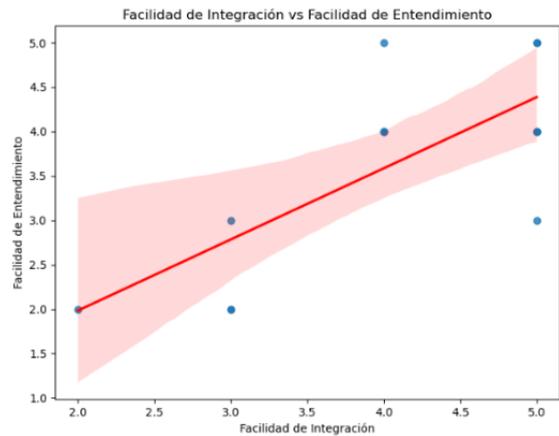


Ilustración 12. Regresión Lineal: Facilidad Integración y Facilidad Entendimiento

## 6.7. Estado del Arte

El impacto de la inteligencia artificial generativa en el desarrollo de software ya es tangible, como hemos podido observar en este estudio como entre tantos otros. Sin embargo, su verdadero potencial aún está siendo investigado, a medida que las tecnologías evolucionan y se integran de manera más profunda en los procesos productivos.

Existen otros estudios que abordan la eficiencia del uso de la IA generativa en el desarrollo de software. Por ejemplo, un estudio investiga la efectividad de la programación en pareja con GitHub Copilot en comparación con la programación en pareja humana, centrándose en la productividad y la calidad del código. Los resultados sugieren que, aunque Copilot aumenta la productividad, la calidad del código producido es inferior en comparación con la programación en pareja humana [26].

Otro estudio compara el rendimiento de estas herramientas en términos de métricas de calidad del código, como validez, corrección, seguridad, fiabilidad y mantenibilidad. Los resultados muestran que GitHub Copilot genera código correcto el 46.3% del tiempo, mientras que Amazon CodeWhisperer y ChatGPT lo hacen el 31.1% y 65.2% del tiempo, respectivamente. Además, se encontró que el tiempo promedio de deuda técnica, considerando los *code smells*, fue de 9.1 minutos para GitHub Copilot, 5.6 minutos para Amazon CodeWhisperer y 8.9 minutos para ChatGPT [27]. Estos hallazgos subrayan la necesidad de una supervisión y verificación crítica continua, aunque la fecha del estudio es de abril de 2023, y estas tecnologías es probable, debido a la creciente inversión monetaria actual en este ámbito, que realicen mejoras sustanciales en periodos muy cortos de tiempo, por lo que estos resultados en la actualidad podrían variar.

Además, otro estudio presenta los resultados de un experimento controlado donde los desarrolladores completaron tareas de programación un 55.8% más rápido con GitHub Copilot en comparación con el grupo de control sin acceso a la herramienta. Sin embargo, también se ha observado que, aunque Copilot puede aumentar la productividad, la calidad del código generado puede ser inferior en comparación con la programación en pareja humana. Estos resultados, teniendo en cuenta la fecha de realización de junio de 2022, ponen de manifiesto que el punto de partida de mejora de estos modelos cada vez más potentes presentan un potencial notable [28].

El impacto de la IA generativa no se limita solo al desarrollo de software, sino que también está reconfigurando el mercado laboral en general. Según Brynjolfsson et al. [29], la automatización basada en IA está promoviendo un cambio en los roles laborales, desplazando ciertas tareas repetitivas y generando nuevas oportunidades en la supervisión y optimización de modelos de IA. Sin embargo, Acemoglu y Restrepo [30] advierten que, si bien estas herramientas pueden incrementar la productividad, también pueden provocar la reducción de ciertos puestos de trabajo, haciendo esencial la capacitación en habilidades complementarias para la adaptación de la fuerza laboral. En cuanto a la oferta laboral del mercado de IT (*Information Technology*), se observa un aumento en la tasa de desempleo en el sector, alcanzando el 5.7% en enero de 2025, según datos del Departamento de Trabajo de EE.UU. Esto se debe, en parte, a la adopción de IA generativa, que está automatizando tareas rutinarias como la administración de datos y ciertos roles de programación. Aunque algunas empresas aún buscan talento en áreas especializadas, la tendencia sugiere una reestructuración del mercado, donde la optimización de costes mediante IA puede estar limitando la creación de nuevos puestos de trabajo en IT [31].

Para los profesionales, esto implica un reescalamiento de roles: dejar atrás tareas repetitivas para enfocarse en supervisión estratégica, ética de la IA y toma de decisiones complejas. Paralelamente, surgen nuevas especializaciones, como la ingeniería de prompts o la auditoría de código generado, que redefinirán las trayectorias laborales. En el caso de los estudiantes y recién

graduados, la educación debe evolucionar hacia habilidades complementarias (pensamiento crítico, gestión de proyectos) y la integración de la IA generativa como colaboradora en el diseño de soluciones innovadoras, no como sustituta del razonamiento humano.

La adopción a gran escala de estas herramientas supone un cambio en el paradigma del desarrollo, demandando a los profesionales no solo habilidades técnicas tradicionales, sino también la capacidad de trabajar de forma colaborativa con sistemas de IA. Este cambio, sin embargo, no está exento de riesgos: la posible obsolescencia de perfiles que no se adapten y la sobreconfianza en respuestas automatizadas sin verificación crítica son desafíos urgentes. Aquellos que logren adaptarse a este nuevo entorno serán los más preparados para liderar la innovación en un futuro próximo.

Mientras tanto, es crucial mantener un enfoque crítico y estratégico en su implementación, asegurando que las respuestas generadas sean precisas, aplicables y alineadas con los objetivos del proyecto. Con el tiempo, y a medida que se superen las limitaciones actuales, estas herramientas se podrán convertir en un pilar fundamental para impulsar la eficiencia y creatividad en el ámbito del desarrollo de software.

## **7. Conclusiones**

La inteligencia artificial generativa ha demostrado ser una herramienta poderosa y valiosa en el desarrollo de software, aunque no es omnipotente. Su principal tarea es proporcionar respuestas lo más precisas posibles basadas en los datos y contextos proporcionados y su aprendizaje en la fase de entrenamiento del modelo LLM [32].

Es evidente que dotar a la IA de un contexto de trabajo claro y específico mejora significativamente la precisión de las respuestas y reduce la probabilidad de errores o "alucinaciones", como hemos podido observar en los últimos casos de estudio realizados, donde el tratamiento de los prompts es más cuidadoso con el contexto y la información proporcionada.

La IA debe ser vista como una herramienta complementaria que potencia las capacidades del desarrollador humano, no como un sustituto. La habilidad de los desarrolladores para formular prompts efectivos, además de interpretar y ajustar las respuestas generadas por la IA, se ha vuelto una competencia fundamental. Estas habilidades, como se ha podido observar a lo largo del desarrollo de los casos de estudio, permiten aprovechar al máximo el potencial de estas tecnologías, incrementando tanto la eficiencia como la precisión en el proceso de desarrollo de software.

Aunque la automatización que proporciona la IA puede suponer un beneficio, aún estamos lejos de reemplazar completamente la labor humana. Esta afirmación está fundamentada tras aquellos casos, cuyo número no es escaso, en los que la interpretación por el desarrollador y su acción para dar cohesión y utilidad a las respuestas de la IA han sido clave (véase caso de estudio 23, Anexo III).

Las herramientas basadas en IA generativa son un recurso valioso en la actualidad y tienen el potencial de convertirse en un pilar esencial para el desarrollo de software en el futuro. Considero que estas herramientas están cambiando el paradigma tradicional de nuestra profesión, haciendo que la colaboración entre humanos y máquinas sea cada vez más común y, eventualmente, inevitable. El 60% de los trabajadores afirma que utiliza la IA en su día a día, mientras que en el mercado global el 42% de las empresas ya han integrado la IA en sus procesos operativos y un 40% están explorando activamente su posible aplicación en sus flujos de trabajo [33]. Es probable

que, en un futuro no muy lejano, el uso de la IA generativa sea un estándar en la industria, convirtiéndose en un componente clave que definirá el rumbo del desarrollo de software [34].

Sin embargo, este avance depende de cómo se integren estas tecnologías con el conocimiento y la supervisión crítica de los desarrolladores. Entender las limitaciones actuales de la IA generativa es igualmente importante. Estos modelos han mostrado ser efectivos en tareas repetitivas, como estructuras muy marcadas tales como la codificación de módulos muy estructurados (véase el caso de estudio 10, Anexo III) y en la automatización de procesos específicos, pero enfrentan desafíos cuando se trata de comprender contextos profundos o resolver problemas altamente complejos (véase el caso de estudio 15, Anexo III). Para mitigar estos desafíos, es esencial establecer un contexto claro en las consultas que se formulan a los modelos y realizar revisiones críticas de los resultados generados. Sin esta supervisión, se corre el riesgo de cometer errores o aceptar resultados engañosos debido a la falta de precisión en la información proporcionada (véanse los casos de estudio 6, 14 o 22, Anexo III).

Además, la interacción humana en el desarrollo de software no se limita solo a la ejecución de tareas técnicas. La colaboración, la comunicación efectiva y la capacidad de trabajar en equipo son aspectos fundamentales que la IA aún no puede replicar completamente. Los profesionales aportan una perspectiva práctica y contextual que es esencial para desarrollar soluciones efectivas y adecuadas a las necesidades reales.

Otro aspecto a considerar es la adaptabilidad y la creatividad humana. Los desarrolladores pueden pensar fuera de los límites establecidos por los algoritmos y encontrar soluciones innovadoras a problemas inesperados. La capacidad de improvisar y adaptarse a nuevas situaciones es una cualidad intrínseca del ser humano que la IA, por muy avanzada que sea, aún no puede igualar. Cuando surgen anomalías en una codificación e implementación que, en teoría, deberían ser correctas, es el juicio humano y la capacidad de hilar conceptos del trabajo realizado por el equipo en un proyecto determinado lo que toma la partida principal. Esto se observa claramente en todos los casos de estudio, donde la intervención principal siempre ha sido la del desarrollador. La experiencia y el conocimiento acumulado permiten a los profesionales identificar y resolver problemas que pueden no ser evidentes para una IA, representando la responsabilidad última de asegurar que el software funcione correctamente y de manera segura en ellos.

Este cambio en la forma de trabajar nos invita a reconsiderar el papel de la inteligencia artificial en nuestra profesión. Si bien la IA está diseñada para mejorar la eficiencia y la precisión, es crucial que no perdamos de vista nuestra capacidad como desarrolladores de tomar decisiones fundamentadas en el conocimiento adquirido a lo largo de nuestros estudios y experiencias en el área de la informática. Al final, la IA debe servir como un aliado que nos permita enfocarnos en lo que realmente requiere creatividad, innovación y juicio humano, y no como un reemplazo de esas cualidades que nos hacen únicos en nuestra labor.

Finalmente, este trabajo ha supuesto una valiosa oportunidad para profundizar mis conocimientos en el funcionamiento de la IA generativa en las distintas fases del desarrollo de software abordadas en este estudio. En mi formación como ingeniero informático, esto ha supuesto un aumento en mis conocimientos prácticos sobre el funcionamiento de empresas del sector IT, como Worldgrid, donde se implementan técnicas tales como la integración continua y metodologías ágiles de trabajo, reforzando los conocimientos adquiridos a lo largo del grado.

Por otro lado, el uso de tecnologías avanzadas y ampliamente adoptadas en la actualidad, como AWS, programación modular, imágenes Docker, bases de datos relacionales, entre otras, me ha proporcionado una mayor seguridad en estos ámbitos, posicionándome para estar cada vez más involucrado en equipos de trabajo con altas responsabilidades.

## 8. Líneas a Futuro

Antes de abordar las proyecciones a futuro, considero importante destacar una limitación fundamental de este trabajo: a diferencia de estudios como el realizado por el MIT [35], donde se compara el rendimiento entre grupos de trabajo que utilizan herramientas de IA generativa y aquellos que no las emplean, no he contado con la posibilidad de implementar un enfoque experimental con grupos de control y prueba. Dichos estudios permiten obtener resultados más precisos y aplicar técnicas de análisis estadístico descriptivo que refuercen la validez de las conclusiones. En este caso, el análisis presentado se basa en observaciones y casos prácticos que, si bien aporta valor al estudio, limita su alcance y generalización. Es esencial tener en cuenta esta restricción para interpretar los hallazgos dentro del marco de posibilidades del proyecto.

No obstante, considero que este estudio proporciona detalles valiosos y un análisis óptimo y útil del estado y la utilidad de la IA generativa en el desarrollo de software, ya que los casos prácticos y las opiniones de los profesionales con los que hemos contado para abordar cuestiones relacionadas con el uso de la IA proporcionan una visión clara del estado actual de estas herramientas en el ámbito de la informática.

Como consiguiente a todo lo que ha englobado este TFG, una clara y prometedora continuación sería realizar pruebas con grupos de control y grupos de prueba, donde ahí sí podamos aplicar estadísticas aplicadas en base a datos más precisos. Además, se podrían considerar las siguientes líneas de investigación y desarrollo:

### 1. **Evaluación Comparativa de Herramientas de IA Generativa:**

Realizar estudios comparativos entre diferentes herramientas de IA generativa permite identificar sus fortalezas y debilidades en distintos contextos de desarrollo de software. Además, es fundamental analizar cómo cada herramienta se adapta a diferentes lenguajes de programación, frameworks y entornos de desarrollo. De esta manera, se podrán comparar las respuestas de los modelos en base a sus datos de entrenamiento, destacando las particularidades de cada uno.

### 2. **Formación y Capacitación en el Uso de IA:**

Desarrollar programas de formación y capacitación es esencial para que los desarrolladores adquieran las habilidades necesarias para utilizar herramientas de IA de manera efectiva. Además, es importante crear recursos educativos que ayuden a los desarrolladores a comprender mejor las capacidades y limitaciones de la IA generativa. Por último, estudiar cómo mejorar la interacción entre los desarrolladores y las herramientas de IA puede hacer que la colaboración sea más intuitiva y eficiente.

### 3. **Evaluación del Impacto en el Mercado Laboral:**

Investigar cómo la adopción de herramientas de IA generativa está afectando el mercado laboral en el ámbito del desarrollo de software es crucial. Además, es importante analizar cómo están cambiando los roles y las habilidades requeridas para los desarrolladores y cómo pueden adaptarse a estos cambios.

### 4. **Aplicación de las herramientas de IA generativa en las fases de análisis de requisitos y diseño del sistema y arquitectura.**

Analizar el comportamiento en las distintas tareas que conllevan estas dos fases del desarrollo software para implementaciones nuevas de software en distintos proyectos.

## 5. Repaso de casos de estudio tras la realización de este trabajo.

Las respuestas obtenidas en los primeros casos de estudio podrían haber sido más satisfactorias si mis conocimientos sobre el tratamiento de estas tareas con IA hubieran sido tan avanzados como los que poseo actualmente. De esta manera, es probable que hubiera encontrado muchos menos errores simples. Por ejemplo, en el caso de estudio 3, si hubiera formulado un prompt que listara requisitos que, como se ha mencionado, pueden parecer obvios, tales como no eliminar nada del código proporcionado o no modificar el código, la respuesta habría sido óptima. Esto me lleva a la conclusión de que algunos de los errores observados en este informe podrían haberse mitigado o evitado mediante buenas prácticas en el uso de la IA generativa.

Estas líneas de investigación y desarrollo no solo contribuirán a mejorar la efectividad y eficiencia de las herramientas de IA generativa, sino que también ayudarán a integrar estas tecnologías de manera más profunda y segura en el desarrollo de software, potenciando la colaboración entre humanos y máquinas y preparando a los desarrolladores para los desafíos del futuro.

## 9. Referencias

- [1] "Copilot Boosts Developer Productivity," InfoQ, 2024. [Online]. Available: [https://www.infoq.com/news/2024/09/copilot-developer-productivity/?utm\\_source=email&utm\\_medium=ai-ml-data-eng&utm\\_campaign=newsletter&utm\\_content=10012024](https://www.infoq.com/news/2024/09/copilot-developer-productivity/?utm_source=email&utm_medium=ai-ml-data-eng&utm_campaign=newsletter&utm_content=10012024). [Accessed: Dec. 10, 2024].
- [2] "El auge de la IA," GetManfred, 2024. [Online]. Available: <https://www.getmanfred.com/blog/el-auge-de-la-ia>. [Accessed: Dec. 10, 2024].
- [3] "Risks and Unknowns of Generative AI," OECD.AI, 2024. [Online]. Available: <https://oecd.ai/en/genai/issues/risks-and-unknowns>. [Accessed: Feb. 7, 2025].
- [4] I. Goodfellow et al., "Generative Adversarial Nets," in Advances in Neural Information Processing Systems (NIPS), vol. 27, 2014.
- [5] "GPT-3 vs BERT," InvGate Blog, 2024. [Online]. Available: <https://blog.invgate.com/es/gpt-3-vs-bert>. [Accessed: Feb. 7, 2025].
- [6] Gobierno, "Inteligencia artificial generativa: introducción a los LLMs," Gobierno Blog, 2024. [Online]. Available: <https://www.gobierno.es/blog/inteligencia-artificial-generativa-introduccion-a-los-llms/>. [Accessed: Jan. 25, 2025].
- [7] Novita AI, "Un estudio exhaustivo de los modelos de lenguaje de gran tamaño (LLM)" Novita AI Blog, 2024. [Online]. Available: <https://blogs.novita.ai/es/a-comprehensive-survey-of-large-language-models-llms/> [Accessed: Jan. 25, 2025].
- [8] Europa Press. (2025, enero 28). 'Deepfakes' en llamadas, mensajes personalizados con IA y anuncios clonados, entre las amenazas emergentes para 2025. PortalTIC. <https://www.europapress.es/portaltic/ciberseguridad/noticia-deepfakes-llamadas-mensajes-personalizados-ia-anuncios-clonados-amenazas-emergentes-2025-20250128141115.html>. [Accessed: Feb. 8, 2025].
- [9] ISO/IEC/IEEE 12207:2017, "Systems and software engineering — Software life cycle processes," ISO/IEC/IEEE, 2017.

- [10] "The impact of generative AI on higher education learning and teaching: A study of educators' perspectives," *Computers and Education: Artificial Intelligence*, vol. 2024, 2024. [Online]. Available: <https://doi.org/10.1016/j.caeai.2024.100221>. [Accessed: Feb. 7, 2025].
- [11] Gergely Orosz, "How AI Will Change Software Engineering," *The Pragmatic Engineer*, 2024. [Online]. Available: [https://newsletter.pragmaticengineer.com/p/how-ai-will-change-software-engineering?utm\\_source=substack&utm\\_medium=email](https://newsletter.pragmaticengineer.com/p/how-ai-will-change-software-engineering?utm_source=substack&utm_medium=email). [Accessed: Dec. 10, 2024].
- [12] Ebis Education, "Visual Studio Code: Qué es, Cómo funciona y Ventajas," [Online]. Available: <https://www.ebiseducation.com/visual-studio-code-que-es-como-funciona-y-ventajas> [Accessed: Dec. 24, 2024].
- [13] GitHub, "Acerca de GitHub," [Online]. Available: <https://github.com/about> [Accessed: Dec. 24, 2024].
- [14] Docker, "Use containers to Build, Share and Run your applications" [Online]. Available: <https://www.docker.com/resources/what-container/> [Accessed: Dec. 24, 2024].
- [15] Spring, "Spring Boot" [Online]. Available: <https://spring.io/projects/spring-boot> [Accessed: Dec. 24, 2024].
- [16] AWS, "AWS Documentation" [Online]. Available: [https://docs.aws.amazon.com/?nc2=h\\_ql\\_doc\\_do](https://docs.aws.amazon.com/?nc2=h_ql_doc_do) [Accessed: Dec. 24, 2024].
- [17] PostgreSQL, "PostgreSQL: The World's Most Advanced Open Source Relational Database" [Online]. Available: <https://www.postgresql.org/> [Accessed: Dec. 24, 2024].
- [18] MySQL, "MySQL TM" [Online]. Available: <https://www.mysql.com/> [Accessed: Dec. 24, 2024].
- [19] Java, "¿Qué es Java?" [Online]. Available: <https://www.ibm.com/mx-es/topics/java> [Accessed: Dec. 24, 2024].
- [20] R. Kohavi et al., "Online A/B Testing: A Practical Guide," *Journal of Machine Learning Research*, vol. 10, pp. 1-30, 2009.
- [21] V. Basili, "The Goal Question Metric Approach," *Encyclopedia of Software Engineering*, Wiley, 1994.
- [22] F. Lonchamp, "A structured conceptual and terminological framework for software process engineering," *Proceedings of the 2nd International Conference on the Software Process*, pp. 41-53, 1993.
- [23] International Organization for Standardization, "ISO/IEC 25002:2024: Systems and software engineering—Systems and software Quality Requirements and Evaluation (SQuaRE)—Quality model overview and usage" ISO, 2024.
- [24] R. Camp, *Benchmarking: The Search for Industry Best Practices That Lead to Superior Performance*, ASQC Quality Press, 1989.
- [25] D. T. Campbell and J. C. Stanley, *Experimental and Quasi-Experimental Designs for Research*. Boston, MA, USA: Houghton Mifflin, 1966.
- [26] S. Imai, "Is GitHub Copilot a Substitute for Human Pair-programming? An Empirical Study," 2022 IEEE/ACM 44th International Conference on Software Engineering: Companion Proceedings (ICSE-Companion), 2022, pp. 319-321, doi: 10.1109/ICSE-Companion55297.2022.9793778.
- [27] B. Yetiştiren, I. Özsoy, M. Ayerdem, y E. Tüzün, "Evaluating the Code Quality of AI-Assisted Code Generation Tools: An Empirical Study on GitHub Copilot, Amazon CodeWhisperer, and ChatGPT," *arXiv preprint arXiv:2304.10778*, 2023. [Online]. Available: [https://www.researchgate.net/publication/370213526\\_Evaluating\\_the\\_Code\\_Quality\\_of\\_AI-](https://www.researchgate.net/publication/370213526_Evaluating_the_Code_Quality_of_AI-)

- [28] S. Peng, E. Kalliamvakou, P. Cihon, y M. Demirer, "The Impact of AI on Developer Productivity: Evidence from GitHub Copilot," arXiv preprint arXiv:2302.06590, 2023. [Online]. Available: <https://arxiv.org/pdf/2302.06590>
- [29] E. Brynjolfsson, D. Rock, y P. S. Tambe, "Artificial Intelligence and the Modern Productivity Paradox: A Clash of Expectations and Statistics," NBER Working Paper No. 24001, 2017.
- [30] D. Acemoglu y P. Restrepo, "Automation and New Tasks: How Technology Displaces and Reinstates Labor," *Journal of Economic Perspectives*, vol. 33, no. 2, pp. 3-30, 2019.
- [31] B. Lin, "IT Unemployment Rises to 5.7% as AI Hits Tech Jobs," *The Wall Street Journal*, vol. –, no. –, Feb. 7, 2025. [Online]. Available: <https://www.wsj.com/articles/it-unemployment-rises-to-5-7-as-ai-hits-tech-jobs-7726bb1b>. [Accessed: Feb. 10, 2025].
- [32] Pinecone, "What is a Vector Database?" *Pinecone.io*, 2023. [Online]. Available: <https://www.pinecone.io/learn/vector-database/>. [Accessed: Jan. 15, 2025].
- [33] "Estadísticas esenciales de inteligencia artificial para 2025: quién la usa y para qué," IEBSchool, 2025. [Online]. Available: <https://www.iebschool.com/blog/estadisticas-esenciales-de-inteligencia-artificial-para-2025-quien-la-usa-y-para-que-tecnologia/>. [Accessed: Feb. 12, 2025].
- [34] G. H. Gergely, "AI Tooling 2024," *The Pragmatic Engineer*, Nov. 22, 2024. [Online]. Available: <https://newsletter.pragmaticengineer.com/p/ai-tooling-2024>. [Accessed: Feb. 12, 2025].
- [35] S. Noy and W. Zhang, "The Effects of Generative AI on High Skilled Work: Evidence from Three Field Experiments with Software Developers," MIT, 2023. [Online]. Available: [https://papers.ssrn.com/sol3/papers.cfm?abstract\\_id=4945566](https://papers.ssrn.com/sol3/papers.cfm?abstract_id=4945566)

## **Anexo**

### **Anexo I. Comentario de la Empresa (por Celestino Güemes Seoane, responsable de Soluciones e Innovación en Worldgrid España).**

Como es patente en cualquier sector económico, la IA Generativa está suponiendo un cambio profundo en los procesos de las empresas. En el caso del sector TIC en el que opera Worldgrid, nos supone una visión dual. Por un lado, integrando la IA en las soluciones para nuestros clientes. Y por otro, aplicando herramientas basadas en ella en sus procesos productivos, que tienen un amplio espectro, pero en el que predomina el desarrollo de software. Algunos equipos de la organización habían hecho algunas pruebas con distintas herramientas, sin una metodología bien establecida, y con resultados no concluyentes. Entendimos necesario hacer un esfuerzo para hacer un planteamiento más organizado, que explorase no solo los aspectos técnicos, sino también cómo las herramientas influyen en los procesos de desarrollo (cualitativamente y cuantitativamente), cómo evitar posibles problemas derivados de limitaciones de la IA actual (como las denominadas “alucinaciones”, o los referidos a la propiedad intelectual), y también cómo abordar el cambio organizativo y cultural necesario para que los empleados puedan hacer un aprovechamiento óptimo de las herramientas, aplicando conceptos colaborativos humano-IA como lo que algunos autores denominan “co-inteligencia”.

Con esta necesidad, no podíamos dejar pasar la oportunidad que nos brindaba la colaboración con la Universidad de Cantabria para que un alumno especializado en Ingeniería de Software hiciera su Trabajo Fin de Grado sobre la aplicación práctica, y de una manera controlada, de las capacidades de las herramientas de IA Generativa de código en el escenario real de una aplicación de negocio compleja, basada en tecnologías modernas. El trabajo realizado por el autor de este TFG, Samuel Castro, ha sido exhaustivo, combinando aspectos teóricos (relacionados con los LLM que conforman las herramientas) como empíricos, con la aplicación real de la herramienta y la observación cuantitativa y cualitativa en un gran número de escenarios esenciales del desarrollo de software que debe abordar cualquier ingeniero de software hoy día. Otro aspecto a destacar del trabajo ha sido la interacción con otros equipos de la empresa, para entender sus experiencias (positivas y negativas) con las herramientas, y sus necesidades para una mejor adopción por el resto de equipos.

Ya desde sus etapas intermedias, el trabajo realizado para el TFG nos está sirviendo para dar forma a un plan de adopción de las herramientas de IA generativa a nivel corporativo, con una estrategia claramente definida, un plan de actuación detallado y con acciones específicas de formación, comunicación y de comunidad de practica que lo apoyen. Esto, en un campo que sigue avanzando a un ritmo vertiginoso, tanto en el aspecto técnico (con LLMs cada vez más potentes, modelos de “razonamiento” o modelos multi-agente) como en el campo de aplicación, donde desde el núcleo de “desarrollo de código” analizado en el TFG se puede expandir al resto del ciclo de vida de soluciones, como el análisis, la arquitectura, o el prototipado de soluciones, o a otros ámbitos, como la ciencia de datos. Campos de actuación que esperamos poder proseguir explorando con el autor del TFG.

En conclusión, el uso de las herramientas de IA Generativa aplicadas al ciclo de vida de desarrollo en empresas del sector TIC como Worldgrid se ha convertido en un imperativo para garantizar su competitividad en mercados cada vez más desafiantes. Pero también para sus empleados, donde creo que representa, más que una amenaza, una oportunidad

de mejora profesional. Pero en ambos casos, corporativo o individual, debe adoptarse de una forma planificada, orientada a objetivos claros, y siempre buscando aprovechar su potencial y limitando sus riesgos.

## Anexo II. Tablas de evaluación de la IA

### Generación de Código

<b>Criterio</b>	<b>Descripción</b>	<b>Evaluación Usando IA</b>	<b>Notas/Comentarios</b>
<b>1.1 Quita tiempo de desarrollo</b>	Evalúa si la IA ayuda a acelerar el proceso de desarrollo.	<input type="checkbox"/> No, <input type="checkbox"/> Parcial, <input type="checkbox"/> Sí	
<b>1.2 Cohesión en el código</b>	Las partes de código generadas funcionan bien juntas, manteniendo el contexto.	<input type="checkbox"/> No, <input type="checkbox"/> Parcial, <input type="checkbox"/> Sí	
<b>1.3 Facilita tareas tediosas</b>	Evalúa si la IA reduce la necesidad de realizar tareas manuales repetitivas.	<input type="checkbox"/> No, <input type="checkbox"/> Parcial, <input type="checkbox"/> Sí	
<b>2. Código seguro</b>	El código generado sigue buenas prácticas de seguridad.	<input type="checkbox"/> No, <input type="checkbox"/> Parcial, <input type="checkbox"/> Sí	
<b>3. Funcionalidad recibida</b>	La funcionalidad solicitada por la IA se cumple correctamente.	<input type="checkbox"/> No, <input type="checkbox"/> Parcial, <input type="checkbox"/> Sí	
<b>4. Código claro y entendible</b>	El código generado es legible, fácil de entender y mantener.	<input type="checkbox"/> No, <input type="checkbox"/> Parcial, <input type="checkbox"/> Sí	

### Documentación

<b>Criterio</b>	<b>Descripción</b>	<b>Evaluación Usando IA</b>	<b>Notas/Comentarios</b>
<b>1. Consistencia terminológica</b>	Uso consistente de términos técnicos y nombres de variables en toda la documentación.	<input type="checkbox"/> No, <input type="checkbox"/> Parcial, <input type="checkbox"/> Sí	
<b>2. Cobertura de elementos clave</b>	La documentación incluye todos los elementos necesarios: descripción del módulo/clase, parámetros, retorno, excepciones, ejemplos de uso, etc.	<input type="checkbox"/> No, <input type="checkbox"/> Parcial, <input type="checkbox"/> Sí	
<b>3. Claridad y precisión</b>	Las explicaciones son claras, precisas y adecuadas al nivel técnico esperado de los desarrolladores objetivo.	<input type="checkbox"/> No, <input type="checkbox"/> Parcial, <input type="checkbox"/> Sí	
<b>4. Adaptabilidad al contexto</b>	La documentación refleja adecuadamente el propósito y las particularidades del	<input type="checkbox"/> No, <input type="checkbox"/> Parcial, <input type="checkbox"/> Sí	

	proyecto o dominio en el que se usa el código.		
<b>5. No introduce ambigüedad</b>	La documentación no genera confusión ni contradice el funcionamiento real del código.	<input type="checkbox"/> No, <input type="checkbox"/> Parcial, <input type="checkbox"/> Sí	
<b>6. Impacto en la capacidad cognitiva</b>	La documentación mejora o al menos no dificulta la comprensión del código por parte de desarrolladores experimentados y novatos.	<input type="checkbox"/> No, <input type="checkbox"/> Parcial, <input type="checkbox"/> Sí	

### Pruebas Unitarias

<b>Criterio</b>	<b>Descripción</b>	<b>Evaluación Usando IA</b>	<b>Notas/Comentarios</b>
<b>1. Efectividad</b>	Las pruebas unitarias son útiles y cubren los casos importantes.	<input type="checkbox"/> No, <input type="checkbox"/> Parcial, <input type="checkbox"/> Sí	
<b>2. Completitud y Cobertura</b>	Las pruebas cubren todos los posibles casos, incluyendo edge cases.	<input type="checkbox"/> No, <input type="checkbox"/> Parcial, <input type="checkbox"/> Sí	
<b>3. Capacidad cognitiva y simplicidad</b>	Las pruebas son fáciles de leer y entender sin perder eficacia.	<input type="checkbox"/> No, <input type="checkbox"/> Parcial, <input type="checkbox"/> Sí	
<b>4. Seguridad</b>	Las pruebas son independientes y consistentes, sin afectar el resto del sistema.	<input type="checkbox"/> No, <input type="checkbox"/> Parcial, <input type="checkbox"/> Sí	

### Soporte

<b>Criterio</b>	<b>Descripción</b>	<b>Evaluación Usando IA</b>	<b>Notas/Comentarios</b>
<b>1. Precisión en la asistencia</b>	La IA ofrece respuestas correctas y comandos precisos (e.g., comandos de Docker, configuraciones).	<input type="checkbox"/> No, <input type="checkbox"/> Parcial, <input type="checkbox"/> Sí	
<b>2. Facilidad para debuggear</b>	La IA guía de manera clara y efectiva en la identificación y solución de errores en el código, si es necesario.	<input type="checkbox"/> No, <input type="checkbox"/> Parcial, <input type="checkbox"/> Sí	
<b>3. Soluciones contextualizadas</b>	La IA adapta sus respuestas al contexto y	<input type="checkbox"/> No, <input type="checkbox"/> Parcial,	

	las necesidades específicas del problema planteado.	<input type="checkbox"/> Sí	
<b>4. Proactividad</b>	La IA propone mejoras, optimizaciones o pasos adicionales útiles sin necesidad de ser solicitada.	<input type="checkbox"/> No, <input type="checkbox"/> Parcial, <input type="checkbox"/> Sí	
<b>5. Tiempo de respuesta</b>	La IA responde rápidamente, reduciendo el tiempo necesario para avanzar en las tareas.	<input type="checkbox"/> No, <input type="checkbox"/> Parcial, <input type="checkbox"/> Sí	
<b>6. Claridad en las explicaciones</b>	Las explicaciones y guías ofrecidas son claras y fáciles de entender.	<input type="checkbox"/> No, <input type="checkbox"/> Parcial, <input type="checkbox"/> Sí	
<b>7. Consistencia en las respuestas</b>	Las respuestas de la IA son consistentes y no contradictorias entre sí.	<input type="checkbox"/> No, <input type="checkbox"/> Parcial, <input type="checkbox"/> Sí	
<b>8. Referencias y ejemplos útiles</b>	La IA proporciona ejemplos relevantes y enlaces útiles para ampliar el entendimiento.	<input type="checkbox"/> No, <input type="checkbox"/> Parcial, <input type="checkbox"/> Sí	
<b>9. Flexibilidad</b>	La IA se adapta bien a diferentes lenguajes de programación, frameworks, y herramientas.	<input type="checkbox"/> No, <input type="checkbox"/> Parcial, <input type="checkbox"/> Sí	
<b>10. Reducción del esfuerzo cognitivo</b>	La IA simplifica tareas complejas, haciéndolas más manejables para el desarrollador.	<input type="checkbox"/> No, <input type="checkbox"/> Parcial, <input type="checkbox"/> Sí	

### Puntos de Calificación de la IA

Análisis más detallado y estructurado de los elementos a evaluar en las tablas.

#### Generación de Código

1. Calificación principal → Quita tiempo de desarrollo.
  - a. El desarrollador no pierde el hilo conductor del desarrollo que se va realizando con apoyo de la IA.
  - b. Las distintas partes de código solicitadas trabajan bien conjuntamente.
    - i. Cohesión en base al contexto de la conversación.
  - c. Facilita el desarrollo de tareas tediosas en caso de tener que ser realizadas manualmente.
2. Código seguro.
  - a. Los inputs (si llega a haber) están bien sanitizados y filtrados para garantizar la seguridad del sistema.

- b. El código generado no presenta fallos de seguridad.
  - c. Las conexiones entre módulos son seguras y se comprueban los I/O.
  - d. Usa librerías, estructuras de datos y algoritmos actualizados.
3. La funcionalidad que se pide es la recibida.
- a. Evaluamos en base a 5 casos:
    - i. Respuesta completamente no relacionada.
    - ii. Tema relacionado, pero no responde a la consulta.
    - iii. Intentos de respuesta, pero contiene información incorrecta.
    - iv. Respuesta parcialmente correcta.
    - v. Respuesta completa y correcta.
4. El código es claro y entendible.

### **Documentación**

1. Los comentarios son claros, simples (si se precisa) y concisos.
2. No empeora la capacidad cognitiva del método o la clase.
3. Ayuda a tener un entendimiento general o facilitado del funcionamiento de la clase o método.
4. Explica correctamente el funcionamiento de esa clase o método.
5. Comprende todos los elementos presentes a documentar.
6. No introduce elementos que produzcan ambigüedad.

### **Pruebas unitarias:**

1. Efectividad
  - Que sean pruebas útiles.
2. Completitud y Cobertura
  - Las pruebas son completas y abarcan todo lo necesario para probar ese método.
  - Prueba los posibles Edge cases
    - Casos que ocurren en los límites de los datos de entrada y, a menudo, son los más críticos de probar.
  - Utiliza librerías de comprobación (Mockito o AssertJ)
3. Capacidad cognitiva y simplicidad
  - La prueba deberá de ser legible y entendible por partes iguales, sin perder la eficacia de esta.
  - Mantiene una convención de nombres para cada prueba.
  - Crea bloques diferenciales utilizando la sintaxis Gherkin a la hora de crear las pruebas:
    - Given: bloque donde se establece el caso de uso a probar
    - When: ejecución del caso de uso a probar
    - Then: comprobación del resultado
  - Evita los magic numbers y magic strings.
4. Seguridad

- La prueba deberá ser consistente e independiente, asegurando que su ejecución no suponga ningún retardo en las demás tareas o procesos del programa total y que su ejecución no depende de ningún otro proceso o prueba.

### **Soporte:**

#### **1. Precisión en la asistencia**

La IA ofrece respuestas correctas y comandos precisos (e.g., comandos de Docker, configuraciones).

#### **2. Facilidad para debuggear**

La IA guía de manera clara y efectiva en la identificación y solución de errores en el código, si es necesario.

#### **3. Soluciones contextualizadas**

La IA adapta sus respuestas al contexto y las necesidades específicas del problema planteado.

#### **4. Proactividad**

La IA propone mejoras, optimizaciones o pasos adicionales útiles sin necesidad de ser solicitada.

#### **5. Tiempo de respuesta**

La IA responde rápidamente, reduciendo el tiempo necesario para avanzar en las tareas.

#### **6. Claridad en las explicaciones**

Las explicaciones y guías ofrecidas son claras y fáciles de entender.

#### **7. Consistencia en las respuestas**

Las respuestas de la IA son consistentes y no contradictorias entre sí.

#### **8. Referencias y ejemplos útiles**

La IA proporciona ejemplos relevantes y enlaces útiles para ampliar el entendimiento.

#### **9. Flexibilidad**

La IA se adapta bien a diferentes lenguajes de programación, frameworks, y herramientas.

#### **10. Reducción del esfuerzo cognitivo**

La IA simplifica tareas complejas, haciéndolas más manejables para el desarrollador.

## Anexo III. Casos de estudio

### 1 Asunto: Migración de MySQL a PostgreSQL.

#### Caso de estudio: (Generación de Código)

Se requiere realizar una migración del motor de BBDD de MySQL a PostgreSQL para la que se necesita realizar cambios en las consultas SQL ya que la sintaxis esperada no es la misma.

#### Prompt de Entrada:

Se preguntará a un experto en BBDD sobre el problema y por otra parte a la herramienta MO4D en el modo de desarrollo de código. Realizaremos la comparación correspondiente de ambas respuestas recibidas.

El prompt contará con el código a migrar, haciendo previamente una explicación de la tarea que se ha de realizar, que es la migración de las consultas proporcionadas a PostgreSQL.

#### Respuesta de MO4D:

Consultas adaptadas a PostgreSQL.

#### Respuesta de experto:

La solución que propone la IA la he estado probando y no da un resultado correcto.

La solución de la IA para sustituir la expresión TIMEDIFF tampoco acaba de funcionar, la versión original en MySQL calcula el número de minutos entre 2 fechas, he utilizado la tabla tm\_alertas\_ficheros porque la tabla tm\_alertas\_especiales no tiene registros. He restado 3 días a la fecha para que se vea clara la diferencia con la solución de la IA.

Por lo que parece la solución de la IA extrae la parte de minutos de la diferencia de fechas, no el número de minutos entre 2 fechas.

Lo curioso es que en otra consulta la solución que propone la IA coincide con la que proponía yo.

El resto de las consultas las recomendaciones que realiza las veo validas.

#### Análisis tras Resultados:

Aunque en primera instancia podemos creer que la IA entiende el resultado que esperamos obtener al intentar solucionar el error que nos surge, alguna de las consultas obtenidas no resultaba para lo que estaban codificadas en un inicio, otro caso que confirma la premisa de apoyarse en la IA, pero con cuidado, ya que al darte algunas respuestas correctas te puedes confiar y pasar por alto algún error que te hace ver que está bien.

Criterio	Descripción	Evaluación Usando IA	Notas/Comentarios
----------	-------------	----------------------	-------------------

<b>1.1 Quita tiempo de desarrollo</b>	Evalúa si la IA ayuda a acelerar el proceso de desarrollo.	<input type="checkbox"/> No, <input checked="" type="checkbox"/> Parcial, <input type="checkbox"/> Sí	La IA acelera el proceso, pero las consultas generadas necesitan revisión para asegurar su precisión.
<b>1.2 Cohesión en el código</b>	Las partes de código generadas funcionan bien juntas, manteniendo el contexto.	<input type="checkbox"/> No, <input checked="" type="checkbox"/> Parcial, <input type="checkbox"/> Sí	Aunque algunas funciones propuestas por la IA eran correctas, otras no representaban correctamente su propósito inicial.
<b>1.3 Facilita tareas tediosas</b>	Evalúa si la IA reduce la necesidad de realizar tareas manuales repetitivas.	<input type="checkbox"/> No, <input type="checkbox"/> Parcial, <input checked="" type="checkbox"/> Sí	La IA reduce la necesidad de realizar tareas repetitivas manualmente, en este caso la conversión de consultas SQL.
<b>2. Código seguro</b>	El código generado sigue buenas prácticas de seguridad.	<input type="checkbox"/> No, <input type="checkbox"/> Parcial, <input checked="" type="checkbox"/> Sí	
<b>3. Funcionalidad recibida</b>	La funcionalidad solicitada por la IA se cumple correctamente.	<input checked="" type="checkbox"/> No, <input type="checkbox"/> Parcial, <input type="checkbox"/> Sí	El comportamiento de una consulta era distinto al que realizaba en MySQL, lo que resultó en más tiempo de codificación y revisión.
<b>4. Código claro y entendible</b>	El código generado es legible, fácil de entender y mantener.	<input type="checkbox"/> No, <input type="checkbox"/> Parcial, <input checked="" type="checkbox"/> Sí	El código generado por la IA es legible y fácil de entender, lo cual facilita su mantenimiento.

**Nota caso de estudio:**

1.1 **Quita tiempo de desarrollo:** Parcial (1 punto)

1.2 **Cohesión en el código:** Parcial (1 punto)

1.3 **Facilita tareas tediosas:** Sí (2 puntos)

2. **Código seguro:** Sí (2 puntos)

3. **Funcionalidad recibida:** Parcial (0 puntos)

4. **Código claro y entendible:** Sí (2 puntos)

**Suma de puntos:** 1 + 1 + 2 + 2 + 0 + 2 = 8 puntos

**Puntos posibles:** 12

**Puntuación final:**  $(8 / 12) * 10 = 6,67$  puntos sobre 10

Tiempo de desarrollo tradicional: 10 minutos.

Tiempo de desarrollo con la IA: 15 minutos.

→El tiempo de desarrollo con la IA aumentó dado el cambio necesario de una consulta tras comprobar que el comportamiento tras su migración de MySQL a PostgreSQL era distinto.

## 2 Asunto: Depuración de problema en el código en un proyecto de Java con Spring

### Caso de estudio: (Soporte)

Se presenta un bloqueo en una parte de una función java, específicamente en una función de una librería de spring boot. Para su depuración se pregunta a la herramienta pasándole el código y preguntando a qué se puede deber.

### Prompt de Entrada:

Se proporcionará la información recopilada sobre el bloqueo presente, además del punto del código donde se queda parado.

El prompt constará del texto recibido de la excepción encontrada, seguido de detalles sobre el código donde puede estar presente el bloqueo.

### Respuesta de MO4D:

Ofrece sugerencias de mejora y/o cambios que posiblemente se acerquen al origen del error. La información proporcionada la divide en secciones según el posible tipo de solución a la excepción encontrada.

Aclara lo que hay que realizar en cada punto e indica las pautas a seguir en caso de querer aplicar alguno de los cambios que expresa.

### Respuesta de experto:

La respuesta obtenida es la recibida sobre un primer prompt muy general en el cual se detalla la excepción y el código donde está presente el error.

### Análisis tras resultados:

En este caso, aunque la IA sí que te puede servir como guía inicial de pasos a seguir, como desarrollador software lo óptimo es realizar una depuración tradicional, usando puntos de ruptura y comprobando los valores de las variables y llamadas a funciones para entender más a fondo el problema, obteniendo así una idea más desarrollada de la excepción a resolver generando un prompt más específico que sirviese a la IA para ser más precisa en su respuesta.

Criterio	Descripción	Evaluación Usando IA	Notas/Comentarios
1. Precisión en la asistencia	La IA ofrece respuestas correctas y comandos precisos (e.g., comandos de Docker, configuraciones).	<input type="checkbox"/> No, <input checked="" type="checkbox"/> Parcial, <input type="checkbox"/> Sí	La IA proporciona sugerencias útiles, pero no siempre acierta en la solución específica sin una mayor contextualización.

<b>2. Facilidad para debuggear</b>	La IA guía de manera clara y efectiva en la identificación y solución de errores en el código, si es necesario.	<input type="checkbox"/> No, <input checked="" type="checkbox"/> Parcial, <input type="checkbox"/> Sí	La IA ofrece una guía general para la depuración, pero una depuración manual tradicional sigue siendo necesaria para una comprensión más profunda.
<b>3. Soluciones contextualizadas</b>	La IA adapta sus respuestas al contexto y las necesidades específicas del problema planteado.	<input type="checkbox"/> No, <input checked="" type="checkbox"/> Parcial, <input type="checkbox"/> Sí	Las respuestas son útiles, pero a veces requieren prompts más específicos para ser completamente relevantes.
<b>4. Proactividad</b>	La IA propone mejoras, optimizaciones o pasos adicionales útiles sin necesidad de ser solicitada.	<input type="checkbox"/> No, <input type="checkbox"/> Parcial, <input checked="" type="checkbox"/> Sí	
<b>5. Tiempo de respuesta</b>	La IA responde rápidamente, reduciendo el tiempo necesario para avanzar en las tareas.	<input type="checkbox"/> No, <input type="checkbox"/> Parcial, <input checked="" type="checkbox"/> Sí	
<b>6. Claridad en las explicaciones</b>	Las explicaciones y guías ofrecidas son claras y fáciles de entender.	<input type="checkbox"/> No, <input type="checkbox"/> Parcial, <input checked="" type="checkbox"/> Sí	
<b>7. Consistencia en las respuestas</b>	Las respuestas de la IA son consistentes y no contradictorias entre sí.	<input type="checkbox"/> No, <input type="checkbox"/> Parcial, <input checked="" type="checkbox"/> Sí	
<b>8. Referencias y ejemplos útiles</b>	La IA proporciona ejemplos relevantes y enlaces útiles para ampliar el entendimiento.	<input type="checkbox"/> No, <input checked="" type="checkbox"/> Parcial, <input type="checkbox"/> Sí	La IA proporciona ejemplos útiles, pero a veces estos requieren verificación adicional.
<b>9. Flexibilidad</b>	La IA se adapta bien a diferentes lenguajes de programación, frameworks, y herramientas.	<input type="checkbox"/> No, <input type="checkbox"/> Parcial, <input type="checkbox"/> Sí	<No se contempla en este caso de estudio>
<b>10. Reducción del esfuerzo cognitivo</b>	La IA simplifica tareas complejas, haciéndolas más	<input type="checkbox"/> No, <input checked="" type="checkbox"/> Parcial, <input type="checkbox"/> Sí	La IA simplifica las tareas complejas hasta cierto punto, pero el desarrollador aún necesita

	manejables para el desarrollador.		intervenir en momentos críticos.
--	-----------------------------------	--	----------------------------------

Nota caso de estudio: Al no contemplar el punto 9 en este caso de estudio debida la naturaleza del problema y del entorno, la puntuación posible total es de 18, siendo el total obtenido de:

1. **Precisión en la asistencia:** Parcial (1 punto)
2. **Facilidad para debuggear:** Parcial (1 punto)
3. **Soluciones contextualizadas:** Parcial (1 punto)
4. **Proactividad:** Sí (2 puntos)
5. **Tiempo de respuesta:** Sí (2 puntos)
6. **Claridad en las explicaciones:** Sí (2 puntos)
7. **Consistencia en las respuestas:** Sí (2 puntos)
8. **Referencias y ejemplos útiles:** Parcial (1 punto)
9. **Flexibilidad:** <No se contempla en este caso de estudio>
10. **Reducción del esfuerzo cognitivo:** Parcial (1 punto)

**Suma de puntos:**  $1 + 1 + 1 + 2 + 2 + 2 + 2 + 1 + 0 + 1 = 13$  puntos

**Puntos posibles:** 18

**Puntuación final:**  $(13 / 18) * 10 = 7.22$  puntos sobre 10

### **3 Asunto: Explicación del código sin contexto previo de su uso o funcionamiento.**

#### **Caso de estudio: (Documentación)**

Introduciéndose en un proyecto nuevo para el ingeniero, sin previa ejecución del entorno de desarrollo y con las clases sin documentación. Se le pasará a la herramienta las clases que sean necesarias para probar si facilita el entendimiento del funcionamiento de los elementos software presentes en el proyecto.

#### **Prompt de Entrada:**

Se proporcionará un módulo software con clases extensas y dependencias muy estructuradas. Además de las dudas o sugerencias encontradas a lo largo del proceso.

#### **Respuesta de MO4D:**

Realiza una explicación detallada y bien estructurada de las partes presentes en las clases que hemos proporcionado para que nos las explique, además de resumir la finalidad de cada una. También responde a preguntas específicas sobre determinadas partes una vez te ha proporcionado la respuesta a la primera consulta a analizar en este caso de estudio, permitiendo al desarrollador indagar en ciertos aspectos que quiera clarificar más o simplemente validar la información. Es decir, si preguntas que explique más en detalle un método, por ejemplo, te expone más información relevante sobre él.

### Respuesta de experto:

Realizar la misma tarea de comprensión de otras clases sin la asistencia de la herramienta de IA resulta un proceso más laborioso y menos eficiente, aunque finalmente se logre entender su finalidad.

### Análisis tras resultados:

Permite comprender de manera más ágil proyectos nuevos para el desarrollador, evitando que el proceso sea tedioso e ineficiente como podría suceder al analizar el código sin estas herramientas. De esta manera se permite a los desarrolladores concentrarse en aspectos más críticos del análisis.

<b>Criterio</b>	<b>Descripción</b>	<b>Evaluación Usando IA</b>	<b>Notas/Comentarios</b>
<b>1. Consistencia terminológica</b>	Uso consistente de términos técnicos y nombres de variables en toda la documentación.	<input type="checkbox"/> No, <input checked="" type="checkbox"/> Parcial, <input type="checkbox"/> Sí	La IA utiliza términos técnicos de manera consistente, pero puede requerir ajustes menores para mantener una terminología uniforme.
<b>2. Cobertura de elementos clave</b>	La documentación incluye todos los elementos necesarios: descripción del módulo/clase, parámetros, retorno, excepciones, ejemplos de uso, etc.	<input type="checkbox"/> No, <input checked="" type="checkbox"/> Parcial, <input type="checkbox"/> Sí	La documentación incluye la mayoría de los elementos clave, aunque algunos detalles pueden necesitar ser añadidos manualmente.
<b>3. Claridad y precisión</b>	Las explicaciones son claras, precisas y adecuadas al nivel técnico esperado de los desarrolladores objetivo.	<input type="checkbox"/> No, <input type="checkbox"/> Parcial, <input checked="" type="checkbox"/> Sí	El nivel de detalle proporcionado es lo suficientemente técnico como para ser aplicado en un equipo de trabajo de cualquier proyecto software real.
<b>4. Adaptabilidad al contexto</b>	La documentación refleja adecuadamente el propósito y las particularidades del proyecto o dominio en el que se usa el código.	<input type="checkbox"/> No, <input type="checkbox"/> Parcial, <input checked="" type="checkbox"/> Sí	En primera instancia la respuesta obtenida es adecuada para disponer de un entendimiento claro del entorno de las clases proporcionadas.
<b>5. No introduce ambigüedad</b>	La documentación no genera confusión ni contradice el funcionamiento real del código.	<input type="checkbox"/> No, <input type="checkbox"/> Parcial, <input checked="" type="checkbox"/> Sí	La documentación no genera confusión ni contradice el funcionamiento real del código

<b>6. Impacto en la capacidad cognitiva</b>	La documentación mejora o al menos no dificulta la comprensión del código por parte de desarrolladores experimentados y novatos.	[ ] No, [ ] Parcial, [X] Sí	La documentación mejora la comprensión del código, quitando tiempo de trabajo.
---	--	-----------------------------------	--

Nota caso de estudio:

1. **Consistencia terminológica:** Parcial (1 punto)
2. **Cobertura de elementos clave:** Parcial (1 punto)
3. **Claridad y precisión:** Sí (2 puntos)
4. **Adaptabilidad al contexto:** Sí (2 puntos)
5. **No introduce ambigüedad:** Sí (2 puntos)
6. **Impacto en la capacidad cognitiva:** Sí (2 puntos)

**Suma de puntos:** 1 + 1 + 2 + 2 + 2 + 2 = 10 puntos

**Puntos posibles:** 12

**Puntuación final:** (10 / 12) \* 10 = 8.33 puntos sobre 10

#### 4 Asunto: Generación de una interfaz java en base a una clase que no dispone de una.

##### Caso de estudio: (Generación de Código)

Se solicita generar una interfaz java de la clase X.java, la cual es bastante extensa y carece de interfaz que la apoye.

##### Prompt de Entrada:

En base a la clase X.java, génrame una interfaz java que contemple todos sus métodos.

##### Respuesta de MO4D:

Interfaz java con todos los métodos declarados de la clase proporcionada en el prompt de entrada inicial. La interfaz estaría lista para ser implementada en el código.

##### Respuesta de experto:

Al revisar la respuesta con la clase, se ha observado que faltan métodos por declarar en la interfaz. Esto sugiere que, en casos tan extensos como este, es necesario realizar una revisión antes de continuar con el siguiente desarrollo, aunque, en realidad, debería ser una práctica habitual, ya que ningún proceso es infalible.

### Análisis tras resultados:

Dado que se trata de una clase extensa (5500 líneas de código), sin documentación ni refactorización, la tarea de crear manualmente la interfaz Java para recoger todos los métodos presentes en la clase sería tan tediosa que resultaría ineficiente. En cambio, con el uso de la herramienta, se puede obtener una respuesta en menos de un minuto, habiendo adaptado las dependencias según las necesidades de la clase, lo cual ha tomado muy poco tiempo.

Como resultado, se obtiene una interfaz para una clase muy extensa que, inicialmente, carecía de elementos esenciales en términos de calidad de desarrollo. Ahora, en un tiempo reducido, se dispone de una interfaz que ofrece una visión más intuitiva y detallada de los métodos presentes en la clase.

<b>Criterio</b>	<b>Descripción</b>	<b>Evaluación Usando IA</b>	<b>Notas/Comentarios</b>
<b>1.1 Quita tiempo de desarrollo</b>	Evalúa si la IA ayuda a acelerar el proceso de desarrollo.	<input type="checkbox"/> No, <input type="checkbox"/> Parcial, <input checked="" type="checkbox"/> Sí	La IA acelera significativamente el proceso de desarrollo, generando la interfaz en menos de un minuto.
<b>1.2 Cohesión en el código</b>	Las partes de código generadas funcionan bien juntas, manteniendo el contexto.	<input type="checkbox"/> No, <input type="checkbox"/> Parcial, <input checked="" type="checkbox"/> Sí	
<b>1.3 Facilita tareas tediosas</b>	Evalúa si la IA reduce la necesidad de realizar tareas manuales repetitivas.	<input type="checkbox"/> No, <input type="checkbox"/> Parcial, <input checked="" type="checkbox"/> Sí	La IA reduce significativamente en tiempos de desarrollo la creación de una interfaz muy extensa como es en este caso.
<b>2. Código seguro</b>	El código generado sigue buenas prácticas de seguridad.	<input type="checkbox"/> No, <input type="checkbox"/> Parcial, <input checked="" type="checkbox"/> Sí	
<b>3. Funcionalidad recibida</b>	La funcionalidad solicitada por la IA se cumple correctamente.	<input type="checkbox"/> No, <input checked="" type="checkbox"/> Parcial, <input type="checkbox"/> Sí	Aunque el resultado obtenido fue óptimo, la falta de determinados métodos concluyó en un error de conceptos o de resultado de la IA.
<b>4. Código claro y entendible</b>	El código generado es legible, fácil de entender y mantener.	<input type="checkbox"/> No, <input type="checkbox"/> Parcial, <input checked="" type="checkbox"/> Sí	

Nota caso de estudio:

- 1.1. Quita tiempo de desarrollo: Parcial (2 puntos)
- 1.2. Cohesión en el código: Parcial (2 puntos)
- 1.3. Facilita tareas tediosas: Sí (2 puntos)
2. Código seguro: Sí (2 puntos)
3. Funcionalidad recibida: Parcial (1 punto)
4. Código claro y entendible: Sí (2 puntos)

**Suma de puntos:**  $2 + 2 + 2 + 2 + 1 + 2 = 11$  puntos

**Puntos posibles:** 12

**Puntuación final:**  $(11 / 12) * 10 = 9,17$  puntos sobre 10

Tiempo de desarrollo tradicional: 40 minutos.

Tiempo de desarrollo con la IA: 10 minutos.

## **5 Asunto: Documentación de una clase Java extensa**

### **Caso de estudio: (Documentación)**

Se solicitará la documentación de la clase X.java, la cual es la principal clase del módulo, además de ser muy extensa.

### **Prompt de Entrada:**

Clase a documentar.

### **Respuesta de MO4D:**

Documento Word con la especificación de cada método y parte contemplada en la clase proporcionada en el prompt de entrada. Además de incluir diagramas de flujo de la clase.

### **Respuesta de experto:**

He tardado muy poco en documentar la clase utilizando la herramienta. Considerando el tamaño considerable de la clase, la falta de documentación previa, la ausencia de métodos refactorizados y la escasez de comentarios en los métodos, se llega a la conclusión de que estas herramientas resultan muy efectivas en cuanto a la eficiencia en el desarrollo de documentación clara y funcional.

### **Análisis tras resultados:**

Aunque la herramienta permitió documentar la clase de manera rápida y eficiente, es importante señalar que siempre es recomendable revisar el documento generado para asegurarse de que la información presentada sea precisa y veraz en relación con la clase documentada. Este análisis muestra que, en proyectos grandes y con poca documentación, el uso de herramientas de automatización puede ahorrar un tiempo significativo y mejorar la claridad de la documentación. La combinación de la eficiencia de la herramienta con la supervisión manual es clave para obtener los mejores resultados en cuanto a calidad y precisión.

<b>Criterio</b>	<b>Descripción</b>	<b>Evaluación Usando IA</b>	<b>Notas/Comentarios</b>
<b>1. Consistencia terminológica</b>	Uso consistente de términos técnicos y nombres de variables en toda la documentación.	<input type="checkbox"/> No, <input type="checkbox"/> Parcial, <input checked="" type="checkbox"/> Sí	
<b>2. Cobertura de elementos clave</b>	La documentación incluye todos los elementos necesarios: descripción del módulo/clase, parámetros, retorno, excepciones, ejemplos de uso, etc.	<input type="checkbox"/> No, <input type="checkbox"/> Parcial, <input checked="" type="checkbox"/> Sí	Describe las distintas partes, sus funciones, elementos que abarcan y diagrama de flujo.
<b>3. Claridad y precisión</b>	Las explicaciones son claras, precisas y adecuadas al nivel técnico esperado de los desarrolladores objetivo.	<input type="checkbox"/> No, <input type="checkbox"/> Parcial, <input checked="" type="checkbox"/> Sí	
<b>4. Adaptabilidad al contexto</b>	La documentación refleja adecuadamente el propósito y las particularidades del proyecto o dominio en el que se usa el código.	<input type="checkbox"/> No, <input type="checkbox"/> Parcial, <input checked="" type="checkbox"/> Sí	
<b>5. No introduce ambigüedad</b>	La documentación no genera confusión ni contradice el funcionamiento real del código.	<input type="checkbox"/> No, <input type="checkbox"/> Parcial, <input checked="" type="checkbox"/> Sí	No hay contradicciones presentes en el documento devuelto por la IA al solicitar la documentación de la clase.
<b>6. Impacto en la capacidad cognitiva</b>	La documentación mejora o al menos no dificulta la comprensión del código por parte de desarrolladores experimentados y novatos.	<input type="checkbox"/> No, <input type="checkbox"/> Parcial, <input checked="" type="checkbox"/> Sí	La documentación recibida por la IA explica todos los puntos necesarios para o bien poder tener un entendimiento general si es lo que buscas o para obtener toda la información sobre el funcionamiento de la clase.

Nota caso de estudio:

- 1. Consistencia terminológica:** Sí (2 puntos)

2. **Cobertura de elementos clave:** Sí (2 puntos)
3. **Claridad y precisión:** Sí (2 puntos)
4. **Adaptabilidad al contexto:** Sí (2 puntos)
5. **No introduce ambigüedad:** Sí (2 puntos)
6. **Impacto en la capacidad cognitiva:** Sí (2 puntos)

**Suma de puntos:**  $2 + 2 + 2 + 2 + 2 + 2 = 12$  puntos

**Puntos posibles:** 12

**Puntuación final:**  $(12 / 12) * 10 = 10$  puntos sobre 10

## **6 Asunto: Documentar una clase con comentarios en el código**

### **Caso de estudio: (Documentación)**

Se realizarán los comentarios pertinentes a una clase sin documentación alguna con ayuda de la herramienta de IA generativa y posteriormente se realizará parte de esa clase manualmente sin su ayuda. En ambos casos se medirá la efectividad para determinar la usabilidad de MO4D en este tipo de funciones o tareas.

### **Prompt de Entrada:**

Se irán proporcionando métodos a comentar, añadiendo información relativa si se precisa.

### **Respuesta de MO4D:**

Código bien comentado, describiendo las funciones presentes en cada método que se proporciona.

Por otra parte, realiza comentarios en algunos puntos que son innecesarios por su obviedad, añadiendo así un gran overhead de comentarios del código que más que ayudar dificultan la lectura y el entendimiento. Aunque he intentado que no incluya los comentarios que son innecesarios, la respuesta sigue resultando de la misma manera.

Tras haber revisado detalladamente cada comentario y parte de la respuesta obtenida, he observado que retira partes del código que, si bien no ejecutaban nada, sí que estaban ahí escritos marcando una estructura que podría ser utilizada a futuro.

### **Respuesta de experto:**

Comentar métodos que desconoces es tarea compleja por lo que implica su estudio y entendimiento previo.

Por otra parte, comentar funcionalidades que ha desarrollado uno mismo es más fácil y no requiere grandes cantidades de tiempo para su fin.

### **Análisis tras resultados:**

La generación de comentarios en métodos de clases extensas y complejas, como la propuesta en este caso, que consta de aproximadamente 5500 líneas de código y métodos de gran longitud, se ve favorecida por la utilización de IA generativa. Esta tecnología

permite obtener explicaciones detalladas sobre la función que cada parte del código desempeña, facilitando la comprensión global del mismo.

Sin embargo, la tarea de comentar código, especialmente si no ha sido escrito por el propio desarrollador, puede ser compleja, ya que requiere un análisis exhaustivo del comportamiento de cada sección del código. Esto implica un considerable gasto de tiempo para entender las fases y funciones ejecutadas. En este sentido, es necesario tener precaución al utilizar herramientas de IA generativa, ya que estas podrían proporcionar explicaciones incorrectas o imprecisas sobre lo que realmente está sucediendo en el código.

Por último, tras una observación más en detalle de la respuesta obtenida por la herramienta de IA generativa en la tarea de comentar un método que proporcionamos, he observado que retira partes del código que, si bien no realizan ninguna ejecución en el código, ni eran útiles o utilizados en el conjunto global de la ejecución, siempre debe ser decisión del desarrollador eliminar esas partes, y más si la tarea que se le precisa a la IA es comentar y no refactorizar o modificar específicamente esos campos. Esto nos hace ver que siempre hay que revisar las respuestas y nunca hay que confiar ciegamente, ya que podría resultar en un gran problema a futuro. Solo en caso de querer el código comentado sin estar presentes las partes del código que quizás no sean críticas y así tener las partes importantes comentadas puede que sea útil, aunque no es del todo recomendable.

<b>Criterio</b>	<b>Descripción</b>	<b>Evaluación Usando IA</b>	<b>Notas/Comentarios</b>
<b>1. Consistencia terminológica</b>	Uso consistente de términos técnicos y nombres de variables en toda la documentación.	<input type="checkbox"/> No, <input type="checkbox"/> Parcial, <input checked="" type="checkbox"/> Sí	
<b>2. Cobertura de elementos clave</b>	La documentación incluye todos los elementos necesarios: descripción del módulo/clase, parámetros, retorno, excepciones, ejemplos de uso, etc.	<input checked="" type="checkbox"/> No, <input type="checkbox"/> Parcial, <input type="checkbox"/> Sí	La IA retira partes del código que no debería tocar.
<b>3. Claridad y precisión</b>	Las explicaciones son claras, precisas y adecuadas al nivel técnico esperado de los desarrolladores objetivo.	<input type="checkbox"/> No, <input type="checkbox"/> Parcial, <input checked="" type="checkbox"/> Sí	

<b>4. Adaptabilidad al contexto</b>	La documentación refleja adecuadamente el propósito y las particularidades del proyecto o dominio en el que se usa el código.	<input type="checkbox"/> No, <input type="checkbox"/> Parcial, <input checked="" type="checkbox"/> Sí	
<b>5. No introduce ambigüedad</b>	La documentación no genera confusión ni contradice el funcionamiento real del código.	<input checked="" type="checkbox"/> No, <input type="checkbox"/> Parcial, <input type="checkbox"/> Sí	Si bien realiza la documentación al completo, la gran cantidad de comentarios evidentes introduce mucho overhead al código que dificulta su lectura más que facilitarla.
<b>6. Impacto en la capacidad cognitiva</b>	La documentación mejora o al menos no dificulta la comprensión del código por parte de desarrolladores experimentados y novatos.	<input type="checkbox"/> No, <input type="checkbox"/> Parcial, <input checked="" type="checkbox"/> Sí	

Nota caso de estudio:

1. **Consistencia terminológica:** Sí (2 puntos)
2. **Cobertura de elementos clave:** No (0 puntos)
3. **Claridad y precisión:** Sí (2 puntos)
4. **Adaptabilidad al contexto:** Sí (2 puntos)
5. **No introduce ambigüedad:** No (0 puntos)
6. **Impacto en la capacidad cognitiva:** Sí (2 puntos)

**Suma de puntos:**  $2 + 0 + 2 + 2 + 0 + 2 = 8$  puntos

**Puntos posibles:** 12

**Puntuación final:**  $(8 / 12) * 10 = 6.67$  puntos sobre 10

## 7 Asunto: Cambio de formato o uso de librerías en una clase java

### Caso de estudio: (Generación de Código)

Se solicita realizar un cambio de uso de métodos de formato de salida de fecha de “DateTimeFormatter” al uso de “TimeStamp”, debido a problemas en la compatibilidad con bases de datos de PostgreSQL al intentar migrar desde MySQL.

### **Prompt de Entrada:**

Se especifica el problema de conversión de tipos de datos fecha de PostgreSQL y la implementación dada en la clase (DateTimeFormatter), la cual no es compatible en la nueva implementación de PostgreSQL (Siendo necesario TimeStamp).

### **Respuesta de MO4D:**

Proporciona la clase adaptada a las nuevas necesidades del proyecto indicadas, pudiendo implementar la solución directamente al código.

### **Respuesta de experto:**

Dudamos si el resultado esperado es correcto, ya que al cambiar la clase adaptándola al nuevo requerimiento, se quita de en medio unos formatos de fecha que había previamente. Este comportamiento de la IA no supimos determinar si era consciente o producto de una alucinación, por lo que nos llevó un poco de tiempo determinar si el resultado de la IA era correcto y el deseado.

### **Análisis tras resultados:**

Tras analizar la respuesta final ante las dudas de si eran o no alucinaciones, finalmente comprobamos que no lo eran del todo. ¿Qué quiere decir esto? Lo que está claro es que hay que entender el contexto donde estamos aplicando la IA generativa, que es en un proyecto ya avanzado donde las dependencias entre clases son muy arraigadas y cualquier cambio realizado causará en muchos casos el efecto dominó. Al solicitar el cambio mencionado anteriormente podemos observar que al cambiar los tipos de datos devueltos por los métodos sus correspondientes llamadas en otras clases producen errores de los cuales tampoco nos libraríamos sin la herramienta de IA. Por lo que el trabajo a realizar es prácticamente el mismo, solo que la IA nos ha facilitado en primera instancia los cambios solicitados en la clase TimeFactory.java. Para mitigar este “error” habría que contextualizar más a la IA, buscando en primer lugar las llamadas a esos métodos modificados, y pasando las clases afectadas por ese cambio a la IA, de tal manera que nos devolviese también los demás cambios a realizar necesariamente para que la nueva funcionalidad fuese efectiva y correcta.

<b>Criterio</b>	<b>Descripción</b>	<b>Evaluación Usando IA</b>	<b>Notas/Comentarios</b>
<b>1.1 Quita tiempo de desarrollo</b>	Evalúa si la IA ayuda a acelerar el proceso de desarrollo.	<input type="checkbox"/> No, <input checked="" type="checkbox"/> Parcial, <input type="checkbox"/> Sí	Aunque la respuesta final sí era la buscada, ha sido necesario ir ajustando el prompt varias veces.
<b>1.2 Cohesión en el código</b>	Las partes de código generadas funcionan bien juntas, manteniendo el contexto.	<input type="checkbox"/> No, <input type="checkbox"/> Parcial, <input checked="" type="checkbox"/> Sí	El código generado por la IA era capaz de ser introducido directamente al proyecto.
<b>1.3 Facilita tareas tediosas</b>	Evalúa si la IA reduce la necesidad de realizar tareas	<input type="checkbox"/> No, <input type="checkbox"/> Parcial, <input checked="" type="checkbox"/> Sí	En cuanto a la codificación de la solución sí que quita tiempo de desarrollo.

	manuales repetitivas.		
<b>2. Código seguro</b>	El código generado sigue buenas prácticas de seguridad.	[ ] No, [ ] Parcial, [X] Sí	
<b>3. Funcionalidad recibida</b>	La funcionalidad solicitada por la IA se cumple correctamente.	[ ] No, [X] Parcial, [ ] Sí	Al no tener el contexto global del proyecto, fue necesario un ajuste manual de la implementación del código generado en cuanto a sus dependencias.
<b>4. Código claro y entendible</b>	El código generado es legible, fácil de entender y mantener.	[ ] No, [ ] Parcial, [X] Sí	

Nota caso de estudio:

**1.1. Quita tiempo de desarrollo:** Parcial (1 punto)

**1.2. Cohesión en el código:** Parcial (2 puntos)

**1.3. Facilita tareas tediosas:** Sí (2 puntos)

**2. Código seguro:** Sí (2 puntos)

**3. Funcionalidad recibida:** Parcial (1 punto)

**4. Código claro y entendible:** Sí (2 puntos)

**Suma de puntos:** 1 + 2 + 2 + 2 + 1 + 2 = 10 puntos

**Puntos posibles:** 12

**Puntuación final:**  $(10 / 12) * 10 = 8.33$  puntos sobre 10

## 8 Asunto: Depuración de un error en un método en java

### Caso de estudio: (Soporte)

Se solicita facilitar el debugeo del uso de una función de una librería de jdbc.

### Prompt de Entrada:

Error de obtención de la clave primaria sobre el método “getKey” de una librería de jdbc.

Además de conversación con la herramienta en base a las respuestas obtenidas y los resultados que iban produciendo y de las pruebas hechas por nuestra parte.

### Respuesta de MO4D:

Al no contar con todo el contexto de llamadas y código del proyecto ha hecho falta guiar a la herramienta e incluso indicarla donde creíamos que estaba el error para que diese con una posible implementación a probar. Tras varios intentos finalmente nos dio la respuesta correcta.

### Respuesta de experto:

Tras estar 47 minutos 2 personas consultando el error a resolver, la IA ha sido de notable ayuda guiándonos por donde iba el error, ya que era un problema referente a la obtención de la primary key de una tabla del MySQL la cual devolvía varias keys, cuando solo queríamos la primera y el código estaba implementado para solo recibir una únicamente. Tras un rato conversando con la herramienta e indicándola donde íbamos viendo que estaba la raíz del problema finalmente nos dio una solución a implementar, no correcta del todo ya que el valor a devolver era long y el método utilizado devolvía Integer, lo cual le indicamos y nos corrigió, entregando finalmente el código esperado y funcional.

Mi compañero y yo sacamos la conclusión de que sin esta herramienta quizás habríamos tenido que estar un rato más, ya que ya estuve un rato indagando en la posible respuesta o solución y manualmente era bastante tediosa al tratarse de listas de mapas en java.

### Análisis tras resultados:

Solamente la herramienta de IA no habría sido capaz de encontrar el problema y nosotros solos sí solo que invirtiendo algo más de tiempo. La participación de la persona ha sido crucial para determinar donde estaba la raíz del problema e indicárselo a la IA para poder encontrar una posible solución. Podemos tomar este caso de estudio como ejemplo de trabajo en equipo con una inteligencia artificial y otro compañero. (Doble pair-programming). En este caso concreto al estar utilizando una estructura de datos poco usual en proyectos, como es el acceso de datos en una lista de mapas que apuntan a tablas de una base de datos, la ayuda contextual de la IA en cuanto al tratamiento de estos casos ayudó a agilizar el proceso de la programación de una solución aplicable.

<b>Criterio</b>	<b>Descripción</b>	<b>Evaluación Usando IA</b>	<b>Notas/Comentarios</b>
<b>1. Precisión en la asistencia</b>	La IA ofrece respuestas correctas y comandos precisos (e.g., comandos de Docker, configuraciones).	<input type="checkbox"/> No, <input checked="" type="checkbox"/> Parcial, <input type="checkbox"/> Sí	La IA ofreció respuestas correctas, pero fue necesario ajustar el prompt varias veces.
<b>2. Facilidad para debuggear</b>	La IA guía de manera clara y efectiva en la identificación y solución de errores en el código, si es necesario.	<input type="checkbox"/> No, <input checked="" type="checkbox"/> Parcial, <input type="checkbox"/> Sí	Fue necesario guiar a la IA en aspectos que identificábamos los desarrolladores.
<b>3. Soluciones contextualizadas</b>	La IA adapta sus respuestas al contexto y las necesidades específicas del problema planteado.	<input type="checkbox"/> No, <input type="checkbox"/> Parcial, <input checked="" type="checkbox"/> Sí	
<b>4. Proactividad</b>	La IA propone mejoras, optimizaciones o pasos adicionales	<input type="checkbox"/> No, <input checked="" type="checkbox"/> Parcial, <input type="checkbox"/> Sí	La IA propuso algunas mejoras, pero no de manera constante.

	útiles sin necesidad de ser solicitada.		
<b>5. Tiempo de respuesta</b>	La IA responde rápidamente, reduciendo el tiempo necesario para avanzar en las tareas.	<input type="checkbox"/> No, <input type="checkbox"/> Parcial, <input checked="" type="checkbox"/> Sí	
<b>6. Claridad en las explicaciones</b>	Las explicaciones y guías ofrecidas son claras y fáciles de entender.	<input type="checkbox"/> No, <input type="checkbox"/> Parcial, <input checked="" type="checkbox"/> Sí	
<b>7. Consistencia en las respuestas</b>	Las respuestas de la IA son consistentes y no contradictorias entre sí.	<input type="checkbox"/> No, <input type="checkbox"/> Parcial, <input checked="" type="checkbox"/> Sí	La IA tuvo en cuenta correctamente el contexto de la conversación del momento y problema a solucionar.
<b>8. Referencias y ejemplos útiles</b>	La IA proporciona ejemplos relevantes y enlaces útiles para ampliar el entendimiento.	<input type="checkbox"/> No, <input type="checkbox"/> Parcial, <input checked="" type="checkbox"/> Sí	La IA proporcionó ejemplos aplicativos y posibles opciones de desarrollo disponibles en cada caso que se presentaba.
<b>9. Flexibilidad</b>	La IA se adapta bien a diferentes lenguajes de programación, frameworks, y herramientas.	<input type="checkbox"/> No, <input type="checkbox"/> Parcial, <input checked="" type="checkbox"/> Sí	La IA entendió bien el problema de los tipos de datos a cambiar debido a la migración a PostgreSQL y lo que eso suponía.
<b>10. Reducción del esfuerzo cognitivo</b>	La IA simplifica tareas complejas, haciéndolas más manejables para el desarrollador.	<input type="checkbox"/> No, <input type="checkbox"/> Parcial, <input checked="" type="checkbox"/> Sí	

Nota caso de estudio:

1. **Precisión en la asistencia:** Parcial (1 punto)
2. **Facilidad para debuggear:** Parcial (1 punto)
3. **Soluciones contextualizadas:** Sí (2 puntos)
4. **Proactividad:** Parcial (1 punto)
5. **Tiempo de respuesta:** Sí (2 puntos)
6. **Claridad en las explicaciones:** Sí (2 puntos)
7. **Consistencia en las respuestas:** Sí (2 puntos)
8. **Referencias y ejemplos útiles:** Sí (2 puntos)
9. **Flexibilidad:** Sí (2 puntos)
10. **Reducción del esfuerzo cognitivo:** Sí (2 puntos)

**Suma de puntos:**  $1 + 1 + 2 + 1 + 2 + 2 + 2 + 2 + 2 + 2 = 17$  puntos

**Puntos posibles:** 20

**Puntuación final:**  $(17 / 20) * 10 = 8.5$  puntos sobre 10

## 9 Asunto: Explicación de código

### Caso de estudio: (Documentación)

Se requiere que la IA explique la funcionalidad de un método específico de una clase que el desarrollador no conoce y que no está documentada.

### Prompt de Entrada:

Se proporciona el código de la función directamente.

### Respuesta de MO4D:

Estructura los distintos pasos que conforma la ejecución del método a explicar, sabiendo diferenciar cada posible caso de ejecución posible, además de resolver dudas más específicas de su funcionamiento.

### Respuesta de experto:

Tras haber analizado posteriormente la función que he solicitado que me explique, he comprobado que estructura bien las distintas partes presentes y su tarea en el proceso dado.

Como la situación del momento en el que se realiza la prueba es que como desarrollador voy a estar añadiendo una nueva funcionalidad sobre ese código, esta primera introducción del método me ha ayudado a entrar a trabajar sobre él con una visión general más cercana y dándome así más seguridad a la hora de probar la nueva implementación.

### Análisis tras resultados:

Para desarrolladores que entran a un proyecto nuevo sin tener ningún contexto previo de lo que realiza ese código a analizar y que, sobre todo, a la hora de ponerse a desarrollar sobre él necesite obtener la mayor cantidad de información de lo ya presente en el proyecto, es de gran ayuda y resulta ser bastante efectivo en este tipo de situaciones que en el ámbito empresarial resultan ser bastante cotidianas.

Criterio	Descripción	Evaluación Usando IA	Notas/Comentarios
<b>1. Consistencia terminológica</b>	Uso consistente de términos técnicos y nombres de variables en toda la documentación.	<input type="checkbox"/> No, <input type="checkbox"/> Parcial, <input checked="" type="checkbox"/> Sí	
<b>2. Cobertura de elementos clave</b>	La documentación incluye todos los elementos necesarios: descripción del módulo/clase, parámetros, retorno,	<input type="checkbox"/> No, <input type="checkbox"/> Parcial, <input checked="" type="checkbox"/> Sí	Las especificaciones retornadas son las idóneas para cumplir con el cometido principal de esta caso de estudio.

	excepciones, ejemplos de uso, etc.		
<b>3. Claridad y precisión</b>	Las explicaciones son claras, precisas y adecuadas al nivel técnico esperado de los desarrolladores objetivo.	<input type="checkbox"/> No, <input type="checkbox"/> Parcial, <input checked="" type="checkbox"/> Sí	
<b>4. Adaptabilidad al contexto</b>	La documentación refleja adecuadamente el propósito y las particularidades del proyecto o dominio en el que se usa el código.	<input type="checkbox"/> No, <input type="checkbox"/> Parcial, <input checked="" type="checkbox"/> Sí	
<b>5. No introduce ambigüedad</b>	La documentación no genera confusión ni contradice el funcionamiento real del código.	<input type="checkbox"/> No, <input type="checkbox"/> Parcial, <input checked="" type="checkbox"/> Sí	
<b>6. Impacto en la capacidad cognitiva</b>	La documentación mejora o al menos no dificulta la comprensión del código por parte de desarrolladores experimentados y novatos.	<input type="checkbox"/> No, <input type="checkbox"/> Parcial, <input checked="" type="checkbox"/> Sí	

Nota caso de estudio:

1. **Consistencia terminológica:** Sí (2 puntos)
2. **Cobertura de elementos clave:** Sí (2 puntos)
3. **Claridad y precisión:** Sí (2 puntos)
4. **Adaptabilidad al contexto:** Sí (2 puntos)
5. **No introduce ambigüedad:** Sí (2 puntos)
6. **Impacto en la capacidad cognitiva:** Sí (2 puntos)

**Suma de puntos:**  $2 + 2 + 2 + 2 + 2 + 2 = 12$  puntos

**Puntos posibles:** 12

**Puntuación final:**  $(12 / 12) * 10 = 10$  puntos sobre 10

## **10 Asunto: Desarrollo de funcionalidad nueva**

### **Caso de estudio: (Generación de Código)**

Se desarrollará la funcionalidad de transformación de ficheros posicionales a XML sin la ayuda de la herramienta de IA. Durante este proceso anotaré dificultades encontradas y resultados obtenidos.

Posteriormente, se solicitará a la herramienta que apoye en el desarrollo de la misma implementación y se realizará su correspondiente análisis de resultados y del transcurso del desarrollo.

### **Prompt de Entrada:**

Cuestiones recurrentes en base a la necesidad de cada momento a lo largo del desarrollo.

### **Respuesta de MO4D:**

#### Segundo desarrollo con la herramienta de IA:

##### 1. Estructura de documentos:

Para realizar la estructura del XML que se creará en la transformación de los datos presentes en el fichero posicional he pasado a la IA un documento que explica cada campo presente en el fichero posicional, además de haber proporcionado el contenido de un ejemplo de los ficheros posicionales para que me realizase la estructura del fichero XML solicitado.

El resultado ha sido sorprendente ya que haciendo esta tarea a mano me ha requerido más del doble del tiempo invertido en explicar a la IA lo que precisaba.

##### 2. Codificación:

El desconocimiento inicial de las librerías java que tratan ficheros y su codificación a XML hizo del uso de la herramienta una potente fuente de apoyo para poder tener un flujo de trabajo mucho más continuo y exitoso que teniendo que leer la documentación de dichas librerías.

Por otra parte, considero que sigue siendo esencial la validación y observación de un profesional que guíe a la IA hacia el resultado esperado con las condiciones requeridas sobre la funcionalidad que se desarrolla, además de poner especial atención de que siga buenas prácticas de programación que no comprometan la seguridad del proyecto.

El tiempo de desarrollo ronda la hora y media de codificación, adaptación y revisión del código producido por la IA.

##### 3. Documentación:

Al pedir a la herramienta de IA que genere la documentación pertinente al método de transformación de ficheros posicionales a XML, realiza una descripción detallada de las partes presentes, aunque al no tener el contexto global de la aplicación, tiene fallos en un esquema de flujos que me proporciona ya que la asignación de valores entre las variables que presenta es errónea.

### **Respuesta de experto:**

#### Primer desarrollo sin la herramienta:

### 1. Estructura de documentos:

El primer paso a tomar es familiarizarse con los ficheros posicionales que queremos transformar a XML. Esto al contar con documentación al respecto nos facilita la tarea bastante, pero, aun así, los resultados de los ficheros que observo en el servidor donde se tratan estos documentos no me cuadra a primera vista, por lo que me está tomando más tiempo del necesario. Además de esto, la creación de la estructura del fichero XML lleva bastante tiempo a para codificar.

### 2. Codificación:

Al no estar familiarizado con las librerías de tratamiento de ficheros, Scanner o BufferedReader principalmente, la tarea de leer la documentación y determinar cuál de ellas utilizar en el contexto de la transformación ha requerido algo de tiempo.

Tras saber que clases utilizar, la codificación es medianamente fluida y parecida a la resultante con la herramienta.

El tiempo de desarrollo ronda las 3 horas de codificación.

### 3. Documentación:

Tras haber realizado las dos implementaciones, con y sin IA, la documentación del método no lleva mucho trabajo.

### **Análisis tras resultados:**

El ahorro de tiempo visto en la creación de la estructura del fichero XML a generar refuerza la idea de que las herramientas de IA generativa son altamente eficientes y de gran ayuda en tareas donde el resultado a obtener es fácil de revisar, pero el proceso que conlleva es muy tedioso.

### **Estructura de Documentos**

En el primer desarrollo, sin la ayuda de la IA, tuve que familiarizarme con los ficheros posicionales y su documentación. Aunque la documentación facilitó la tarea, los resultados de los ficheros en el servidor no coincidían a primera vista, lo que me llevó más tiempo del necesario. Crear la estructura del XML manualmente fue una tarea laboriosa y consumió bastante tiempo.

En el segundo desarrollo, con la ayuda de la IA, proporcioné a la herramienta un documento explicando cada campo del fichero posicional y un ejemplo del fichero. La IA generó la estructura del XML rápidamente, ahorrándome más de la mitad del tiempo que hubiera tomado hacerlo manualmente. Fue sorprendente ver cómo la IA podía acelerar este proceso tedioso.

### **Codificación**

Sin la IA, la falta de familiaridad con las librerías Java para el tratamiento de ficheros, como Scanner y BufferedReader, me obligó a leer documentación y decidir cuál usar. Esto tomó tiempo adicional. Una vez identificadas las clases necesarias, la codificación fue fluida, pero el proceso completo tomó alrededor de tres horas.

Con la IA, el proceso fue mucho más eficiente. La herramienta sugirió las librerías y métodos adecuados, lo que redujo la necesidad de consultar documentación. El tiempo total de desarrollo, incluyendo codificación, adaptación y revisión del código generado por la IA, fue de aproximadamente una hora y media. Sin embargo, sigue siendo esencial la supervisión de un ingeniero para asegurar que el código cumpla con las buenas prácticas y no comprometa la seguridad del proyecto.

**Documentación:**

En el desarrollo sin IA, la documentación del método no presentó grandes dificultades y se completó sin problemas.

Con la IA, pedí a la herramienta que generara la documentación pertinente. La IA proporcionó una descripción detallada de las partes presentes, pero cometió errores en el esquema de flujos debido a la falta de contexto global de la aplicación. Esto demuestra que, aunque la IA puede ser útil para generar documentación, es necesario revisar y corregir los errores que puedan surgir.

En definitiva, el uso de herramientas de IA generativa en el desarrollo de software puede ofrecer grandes beneficios en términos de eficiencia y ahorro de tiempo, especialmente en tareas repetitivas y tediosas. La IA demostró ser muy eficiente en la creación de estructuras XML y en la codificación, reduciendo significativamente el tiempo de desarrollo. Sin embargo, la supervisión y validación por parte de un ingeniero siguen siendo esenciales para garantizar la calidad y seguridad del código. Aunque la IA puede ayudar a generar documentación, es necesario revisar y corregir los errores que puedan surgir debido a la falta de comprensión del contexto completo.

<b>Criterio</b>	<b>Descripción</b>	<b>Evaluación Usando IA</b>	<b>Evaluación Sin IA</b>	<b>Notas/Comentarios</b>
<b>1.1 Quita tiempo de desarrollo</b>	Evalúa si la IA ayuda a acelerar el proceso de desarrollo.	<input type="checkbox"/> No, <input type="checkbox"/> Parcial, <input checked="" type="checkbox"/> Sí	<input type="checkbox"/> No, <input type="checkbox"/> Parcial, <input type="checkbox"/> Sí	
<b>1.2 Cohesión en el código</b>	Las partes de código generadas funcionan bien juntas, manteniendo el contexto.	<input type="checkbox"/> No, <input type="checkbox"/> Parcial, <input checked="" type="checkbox"/> Sí	<input type="checkbox"/> No, <input type="checkbox"/> Parcial, <input checked="" type="checkbox"/> Sí	La ejecución del nuevo método resultó ser exitosa.
<b>1.3 Facilita tareas tediosas</b>	Evalúa si la IA reduce la necesidad de realizar tareas manuales repetitivas.	<input type="checkbox"/> No, <input type="checkbox"/> Parcial, <input checked="" type="checkbox"/> Sí	<input type="checkbox"/> No, <input type="checkbox"/> Parcial, <input checked="" type="checkbox"/> Sí	La IA aceleró significativamente el proceso de desarrollo, especialmente en la creación de la

				estructura del fichero XML.
<b>2. Código seguro</b>	El código generado sigue buenas prácticas de seguridad.	<input type="checkbox"/> No, <input type="checkbox"/> Parcial, <input checked="" type="checkbox"/> Sí	<input type="checkbox"/> No, <input type="checkbox"/> Parcial, <input checked="" type="checkbox"/> Sí	Aunque el código seguía buenas prácticas de seguridad, fue necesaria la supervisión del ingeniero para asegurar su corrección.
<b>3. Funcionalidad recibida</b>	La funcionalidad solicitada por la IA se cumple correctamente.	<input type="checkbox"/> No, <input checked="" type="checkbox"/> Parcial, <input type="checkbox"/> Sí		La funcionalidad solicitada se cumplió en su mayoría, pero algunos detalles requerían ajustes manuales.
<b>4. Código claro y entendible</b>	El código generado es legible, fácil de entender y mantener.	<input type="checkbox"/> No, <input type="checkbox"/> Parcial, <input checked="" type="checkbox"/> Sí		

Nota de caso de estudio:

1.3.1 **Quita tiempo de desarrollo:** Sí (2 puntos)

1.3.2 **Cohesión en el código:** Parcial (2 puntos)

1.3.3 **Facilita tareas tediosas:** Sí (2 puntos)

5. **Código seguro:** Sí (2 puntos)

6. **Funcionalidad recibida:** Parcial (1 punto)

7. **Código claro y entendible:** Sí (2 puntos)

**Suma de puntos:** 2 + 2 + 2 + 2 + 1 + 2 = 11 puntos

**Puntos posibles:** 12

**Puntuación final:**  $(11 / 12) * 10 = 9.17$  puntos sobre 10

<b>Criterio</b>	<b>Tradicional</b>	<b>IA Generativa</b>	<b>Diferencia (Mejora)</b>	<b>Porcentaje de Eficiencia</b>
<b>1. Tiempo de desarrollo</b>	3 horas	1,5 horas	-1,5 horas	50% menos tiempo invertido

Tiempo de desarrollo tradicional: 3 horas

Tiempo de desarrollo con la IA: 1 hora y 30 minutos.

→En este caso el tiempo de desarrollo ha sido medido en cuanto a la codificación. Los demás ámbitos que envuelven la adaptación y cohesión de las distintas partes para que la funcionalidad sea efectiva no ha sido medida debido a su dificultad o naturaleza.

## **11 Asunto: Generación de Pruebas unitarias**

### **Caso de estudio: (Pruebas Unitarias)**

Se solicitará a la IA que genere las pruebas unitarias en base a dos clases proporcionadas.

#### **Prompt de Entrada:**

Clase java pequeña a generar pruebas unitarias.

Clase java extensa a generar pruebas unitarias.

#### **Respuesta de MO4D:**

##### Para la clase pequeña:

Clase test de la solicitada, con los nombres, métodos y estructuras de pruebas siguiendo los estándares definidos para su creación.

Utilización de simuladores o *Mocks*, evita los “magic numbers”, prueba las excepciones, métodos auxiliares y objetos retornados.

Las pruebas no comprometen la seguridad ni el estado del proyecto.

El resultado final es una clase de pruebas unitarias efectiva y funcional.

En resumen, cumple con todos los campos descritos en la guía de evaluación que comprenden las pruebas unitarias.

##### Para la clase extensa:

Al contar con una gran cantidad de métodos a probar, los cuales a su vez hacen un amplio uso de métodos auxiliares de otras clases y que tienen una estructura muy arraigada, ha desembocado en la generación de una clase de pruebas unitarias incompleta y no funcional.

#### **Respuesta de experto:**

---

#### **Análisis tras resultados:**

La codificación de las pruebas unitarias de cada clase presente en proyectos tan extensos como RIMA, el cual no presenta ninguna prueba unitaria y el tamaño y número de clases es muy extenso, requeriría mucho tiempo y trabajo que de esta manera he visto ahorrado.

Aun así, la implementación de esta nueva clase test ha sido complicada por problemas de implementación de librerías y dependencias del proyecto, proceso que era inevitable que ocurriera.

Dado este último imprevisto, he observado que para la generación de pruebas unitarias en un proyecto avanzado y sin tener mucho contexto o entendimiento del funcionamiento de la clase sobre la que generamos los test, aunque la IA crea bien lo solicitado, en la mayoría de los casos la integración de estas pruebas en el proyecto da muchos problemas en cuanto a los datos que hay

que introducir en los objetos simulados (*Mocks*), o también con las librerías permitidas/necesitadas para su ejecución en el entorno.

Además de esto, si la clase sobre la que queremos trabajar es muy extensa y presenta una estructura muy compleja, como es el caso de la segunda clase que proporcionamos a la IA, el resultado obtenido ha sido insuficiente y erróneo, dejando partes incompletas y lo generado no funcional, añadiendo más peso al argumento de que cuanto más específica la tarea, y menos extensa o cambios puede presentar, mejor será la respuesta. Como propuesta de mejora en estos casos también aplico la introducción de la herramienta que permita dotar del contexto del programa a la IA y que de esta manera trate todas las clases involucradas a la hora de generar las pruebas unitarias, así como las demás solicitudes que se precisen.

Si solo se requiere una estructura de codificación para iniciar las pruebas, la herramienta sí cumple su cometido. En cambio, para delegar completamente su codificación no es recomendable su uso, ya que como desarrollador no tendrás el foco de atención ni el contexto de desarrollo de las pruebas necesario para entender el funcionamiento de las pruebas en cada caso específico.

<b>Criterio</b>	<b>Descripción</b>	<b>Evaluación Usando IA</b>	<b>Notas/Comentarios</b>
<b>1. Efectividad</b>	Las pruebas unitarias son útiles y cubren los casos importantes.	<input type="checkbox"/> No, <input checked="" type="checkbox"/> Parcial, <input type="checkbox"/> Sí	Las pruebas unitarias generadas para la clase extensa fueron incompletas y no funcionales, aunque las de la clase pequeña fueron efectivas.
<b>2. Completitud y Cobertura</b>	Las pruebas cubren todos los posibles casos, incluyendo edge cases.	<input type="checkbox"/> No, <input checked="" type="checkbox"/> Parcial, <input type="checkbox"/> Sí	Las pruebas para la clase extensa no cubrieron todos los casos posibles, mientras que las de la clase pequeña sí.
<b>3. Capacidad cognitiva y simplicidad</b>	Las pruebas son fáciles de leer y entender sin perder eficacia.	<input type="checkbox"/> No, <input type="checkbox"/> Parcial, <input checked="" type="checkbox"/> Sí	
<b>4. Seguridad</b>	Las pruebas son independientes y consistentes, sin afectar el resto del sistema.	<input type="checkbox"/> No, <input checked="" type="checkbox"/> Parcial, <input type="checkbox"/> Sí	Las pruebas unitarias generadas no siempre fueron independientes y consistentes, especialmente en el caso de la clase extensa.

Nota caso de estudio:

1. **Efectividad:** Parcial (1 punto)
2. **Completitud y cobertura:** Parcial (1 punto)
3. **Capacidad cognitiva y simplicidad:** Sí (2 puntos)
4. **Seguridad:** Parcial (1 punto)

**Suma de puntos:**  $1 + 1 + 2 + 1 = 5$  puntos

**Puntos posibles:** 8

**Puntuación final:**  $(5 / 8) * 10 = 6,25$  puntos sobre 10

Tiempo de desarrollo tradicional:

- Clase extensa: 3 horas.
- Clase pequeña: 1 hora.

Tiempo de desarrollo con la IA:

- Clase extensa: 2 horas.
- Clase pequeña: 20 minutos.

## 12 Asunto: Error de Región de S3 entorno AWS

### Caso de estudio: (Soporte)

Se presenta un error de ejecución en un módulo de descarga de ficheros en un S3 de AWS, donde la región del S3 no está bien configurada según un error en el log, pero en el fichero de propiedades donde se definen las regiones está bien definida.

El error es el siguiente:

```
[2025-01-30 13:45:16.616] [omie - DESKTOP-HF3JHH1]
[simpleMessageListenerContainer-2] ERROR
com.awg.rima.omie.sqs.Consumer@processMessage:43 - The bucket is in this region:
us-east-1. Please use this region to retry the request (Service: Amazon S3; Status Code:
301; Error Code: PermanentRedirect; Request ID: <ID ejemplo>; S3 Extended Request
ID: <ID ejemplo> Proxy: null)
```

### Prompt de Entrada:

#### Primer prompt (Inicio contexto + Proporción de ejemplo de estructura):

Eres un experto en arquitecturas de AWS con amplia experiencia en el sector del desarrollo software. En este caso requiero que realices un análisis de un error que aparece por el log tras intentar realizar la persistencia de un objeto en un S3, el cual es el siguiente:

```
[2025-01-30 13:45:16.616] [omie - DESKTOP-HF3JHH1]
[simpleMessageListenerContainer-2] ERROR
com.awg.rima.omie.sqs.Consumer@processMessage:43 - The bucket is in this region:
us-east-1. Please use this region to retry the request (Service: Amazon S3; Status Code:
301; Error Code: PermanentRedirect; Request ID: <ID ejemplo>; S3 Extended Request
ID: <ID ejemplo> Proxy: null)
```

Quiero implementar en mi código un método postconstruct que, si la región del cliente del s3 no es igual a la que está definida en el properties del módulo java, la cambie a esa misma. Tengo implementado algo inicial como esto:

```
@Value("${aws.sns.region}")
String regionProperties;

@PostConstruct
public void compruebaRegionS3() {
    log.info("REGION S3: " + s3Client.getRegionName());
}
```

```

        if (!s3Client.getRegionName().equals(regionProperties)) {
            s3Client.setRegion(new RegionUtils());
        }
    }
}

```

¿Cómo lo acabarías?

### **Respuesta de MO4D:**

#### **Respuesta al primer prompt:**

Para abordar el problema que estás enfrentando con el cliente S3 y la región, es importante asegurarse de que el cliente esté configurado correctamente antes de realizar cualquier operación. El error que mencionas indica que el bucket S3 está en una región diferente a la que el cliente está utilizando. Así que, efectivamente, se debe ajustar la región del cliente S3.

Aquí tienes una implementación mejorada para tu método `@PostConstruct`, asegurando que el cliente S3 se configure correctamente según la región definida en tus propiedades. También es importante manejar adecuadamente la creación del cliente S3, ya que no se puede cambiar la región de un cliente ya creado. En lugar de cambiar la región, deberías crear un nuevo cliente con la región correcta.

Aquí tienes un código ejemplo:

```

``java
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.stereotype.Component;
import javax.annotation.PostConstruct;

@Component
public class S3ClientConfig {
    @Value("${aws.sns.region}")
    private String regionProperties;
    private AmazonS3 s3Client;

    @PostConstruct
    public void init() {
        // Log la región actual del cliente S3

```

```

    log.info("REGION S3 actual: " + (s3Client != null ? s3Client.getRegionName() : "No
configurado"));

    // Verifica si el cliente S3 ya está configurado
    if (s3Client == null || !s3Client.getRegionName().equals(regionProperties)) {
        // Crea un nuevo cliente S3 con la región especificada en las propiedades
        s3Client = AmazonS3ClientBuilder.standard()
            .withRegion(regionProperties) // Establece la región desde las propiedades
            .build();

        log.info("Cliente S3 configurado a la región: " + regionProperties);
    }
}

public AmazonS3 getS3Client() {
    return s3Client;
}
}
...

```

#### ### Explicación del Código:

##### 1. **\*\*Inyección de Dependencias\*\*:**

- Utilizamos `@Value` para inyectar la región de AWS desde el archivo de propiedades.

##### 2. **\*\*Método `@PostConstruct`\*\*:**

- El método `init()` se llama automáticamente después de que se haya construido el bean. Aquí se verifica si el cliente S3 ya está configurado.

- Si el cliente S3 no está configurado o la región del cliente no coincide con la región de las propiedades, se crea un nuevo cliente S3 utilizando `AmazonS3ClientBuilder`, configurando la región adecuada.

##### 3. **\*\*Manejo de Clientes\*\*:**

- Se asegura que el cliente S3 se cree correctamente según la región especificada en las propiedades, evitando el problema de la región incorrecta.

##### 4. **\*\*Logging\*\*:**

- Se añaden mensajes de log para ayudar a depurar y verificar la configuración de la región.

#### ### Consideraciones Adicionales:

- Asegúrate de que el cliente S3 se inicialice antes de cualquier operación que lo requiera.

- Este enfoque permite una fácil configuración y cambio de región sin alterar el estado del cliente existente.

### Respuesta de experto:

Como este problema no solo implica la correcta codificación de determinadas secciones del código, sino que requiere un conocimiento de la arquitectura del programa y su supervisión por el desarrollador, en estos casos la IA será útil para poder llegar a identificar errores de sintaxis en parámetros de los ficheros de propiedades, supervisando si falta algún campo de configuración y también revisando el código que hace uso de los objetos relacionados con la conexión con la arquitectura del servicio, en este caso el S3 de AWS.

Tras haber estado pensando cómo abordar el problema, intento reparar el error forzando la región del cliente a la predefinida en el properties del módulo, a lo que creo un esquema del método que quiero codificar y se lo proporciono a MO4D, ya que no conozco la estructura ni la documentación de los métodos disponibles de “amazonaws.services.s3” y así consigo obtener un ejemplo de cómo podría crear o modificar la región del cliente S3.

### Análisis tras resultados:

Otro ejemplo de la importancia de seguir aplicando la razón humana en primer lugar y tras analizar el error o reto por delante, abordarlo con todas las herramientas disponibles, como en este caso la IA.

La proporción de un ejemplo de codificación o un inicio de por donde quería que la IA me diese la solución ha sido clave en la obtención del resultado final que, si bien fui yo como desarrollador el que finalmente abordó el problema, sí que me apoyé en la IA para obtener estructura correcta para establecer la región del cliente del S3 antes que tener que leerme la documentación completa o buscar en internet.

<b>Criterio</b>	<b>Descripción</b>	<b>Evaluación Usando IA</b>	<b>Notas/Comentarios</b>
<b>1. Precisión en la asistencia</b>	La IA ofrece respuestas correctas y comandos precisos (e.g., comandos de Docker, configuraciones).	<input type="checkbox"/> No, <input type="checkbox"/> Parcial, <input checked="" type="checkbox"/> Sí	En base a la codificación inicial que proporciono, el resultado es el esperado.
<b>2. Facilidad para debuggear</b>	La IA guía de manera clara y efectiva en la identificación y solución de errores en el código, si es necesario.	<input type="checkbox"/> No, <input type="checkbox"/> Parcial, <input checked="" type="checkbox"/> Sí	
<b>3. Soluciones contextualizadas</b>	La IA adapta sus respuestas al contexto y las necesidades	<input type="checkbox"/> No, <input type="checkbox"/> Parcial, <input checked="" type="checkbox"/> Sí	Trata bien el contexto aplicativo y su rol.

	específicas del problema planteado.		
<b>4. Proactividad</b>	La IA propone mejoras, optimizaciones o pasos adicionales útiles sin necesidad de ser solicitada.	<input type="checkbox"/> No, <input checked="" type="checkbox"/> Parcial, <input type="checkbox"/> Sí	Aunque el problema se aborda correctamente, las sugerencias de donde podría estar relacionado el problema no son del todo claras.
<b>5. Tiempo de respuesta</b>	La IA responde rápidamente, reduciendo el tiempo necesario para avanzar en las tareas.	<input type="checkbox"/> No, <input type="checkbox"/> Parcial, <input checked="" type="checkbox"/> Sí	
<b>6. Claridad en las explicaciones</b>	Las explicaciones y guías ofrecidas son claras y fáciles de entender.	<input type="checkbox"/> No, <input type="checkbox"/> Parcial, <input checked="" type="checkbox"/> Sí	
<b>7. Consistencia en las respuestas</b>	Las respuestas de la IA son consistentes y no contradictorias entre sí.	<input type="checkbox"/> No, <input type="checkbox"/> Parcial, <input checked="" type="checkbox"/> Sí	
<b>8. Referencias y ejemplos útiles</b>	La IA proporciona ejemplos relevantes y enlaces útiles para ampliar el entendimiento.	<input type="checkbox"/> No, <input type="checkbox"/> Parcial, <input checked="" type="checkbox"/> Sí	
<b>9. Flexibilidad</b>	La IA se adapta bien a diferentes lenguajes de programación, frameworks, y herramientas.	<input type="checkbox"/> No, <input type="checkbox"/> Parcial, <input checked="" type="checkbox"/> Sí	
<b>10. Reducción del esfuerzo cognitivo</b>	La IA simplifica tareas complejas, haciéndolas más manejables para el desarrollador.	<input type="checkbox"/> No, <input type="checkbox"/> Parcial, <input checked="" type="checkbox"/> Sí	

Nota caso de estudio:

1. **Precisión en la asistencia:** Sí (2 puntos)
2. **Facilidad para debuggear:** Sí (2 puntos)
3. **Soluciones contextualizadas:** Sí (2 puntos)
4. **Proactividad:** Parcial (1 punto)
5. **Tiempo de respuesta:** Sí (2 puntos)
6. **Claridad en las explicaciones:** Sí (2 puntos)
7. **Consistencia en las respuestas:** Sí (2 puntos)
8. **Referencias y ejemplos útiles:** Sí (2 puntos)

9. **Flexibilidad:** Sí (2 puntos)

10. **Reducción del esfuerzo cognitivo:** Sí (2 puntos)

**Suma de puntos:**  $2 + 2 + 2 + 1 + 2 + 2 + 2 + 2 + 2 + 2 = 19$  puntos

**Puntos posibles:** 20

**Puntuación final:**  $(19 / 20) * 10 = 9,5$  puntos sobre 10

### **13 Asunto: Resolución de problema de extensiones de PostgreSQL.**

#### **Caso de estudio: (Soporte)**

Se presenta un problema referente a la migración de MySQL a PostgreSQL, la cual precisa de la introducción de extensiones tales como pg\_crypto. La imposibilidad de acceso desde el esquema de la base de datos donde estamos trabajando nos hizo tener que indagar en base a una consulta específica, cuya ejecución en el gestor de bases de datos sí funcionaba, pero al realizar la consulta mediante jpa exponiendo la query directamente en el código no reconocía la función mencionada anteriormente.

#### **Prompt de Entrada:**

A lo largo de este proceso se ha ido haciendo un uso de la IA dándole en primer lugar los errores del log para ver qué interpretaba a primera vista. Posteriormente se ha ido ajustando el prompt de entrada según íbamos viendo donde podía estar la raíz del problema.

A medida que descartábamos posibles casos y hacíamos cada vez el prompt más preciso con sugerencias y pruebas ya realizadas, añadiendo hipótesis sobre el posible problema, finalmente dimos con la solución tras varios días de trabajo (no todos dedicados exclusivamente a este problema), resultando en que la instalación de la extensión pg\_crypto se estaba realizando en el esquema público, cuya ejecución en el gestor de base de datos sí que puede acceder al esquema "public" y por lo tanto acceder a la librería, pero en la aplicación java en el fichero "properties" determinamos un esquema específico de trabajo en la base de datos, lo que imposibilitaba el acceso a esa librería de encriptación desde el módulo software ya que no estaba instalada en ese esquema.

#### **Respuesta de MO4D:**

A medida que íbamos avanzando en la introspección del problema, las respuestas eran cada vez más acertadas y dándonos lo que buscábamos.

#### **Respuesta de experto:**

La respuesta final y el que determinó qué era el problema, fue el humano. Claramente haciendo uso de estas herramientas de IA, pero al ser un problema tan específico y poco usual la determinación del ingeniero en la unión de conceptos probados y posibilidades del problema hizo que los prompts que íbamos creando fuesen más certeros y, por consiguiente, nos fuese de gran ayuda a la hora de requerir consultas para comprobar qué librerías estaban instaladas en la base de datos o, por ejemplo, qué esquema era el que presentaba la extensión de pg\_crypto en nuestro gestor de base de datos, lo cual fue determinante a la hora de encontrar la solución final.

### Análisis tras resultados:

Como este caso de estudio se ha prolongado un par de días, realicé la prueba de cambiar el modelo LLM base sobre el que funciona la herramienta MO4D. Hasta el momento estuve utilizando el motor de AI Azure, pero realicé el cambio al de Gemini, de Google, con la finalidad de encontrar cambios de comportamiento tales como efectividad en las respuestas o cantidad de información útil devuelta.

Como punto a destacar en base a este cambio está el uso del contexto de las conversaciones que mantienes con la IA. Esto es, los LLM tienen una función la cual te permite guardar el contexto de la conversación que estás manteniendo con ella, de tal manera que así si ya has tratado un tema previamente con la herramienta, el LLM va a poder acceder a esa conversación o punto y hacer uso de ello.

En este caso, he observado que la inteligencia artificial de Google, Gemini, al realizar consultas relacionadas con PostgreSQL, inicialmente proporcionaba respuestas adecuadas para las consultas específicas de este motor de base de datos. Sin embargo, después de unas cuantas interacciones, parecía perder el contexto del entorno en el que estábamos trabajando, olvidando que se trataba de desarrollo de consultas para PostgreSQL. Esto resultaba en una pérdida de tiempo y la necesidad de realizar solicitudes repetidas para obtener consultas compatibles.

Este caso refleja la importancia de la intervención humana en la resolución de problemas técnicos complejos, a pesar de la asistencia que proporcionan las herramientas de inteligencia artificial. Aunque la IA puede ser útil para proporcionar hipótesis y generar consultas, la experiencia y el juicio del ingeniero fueron clave para combinar diferentes conceptos y realizar pruebas que finalmente llevaron a identificar la raíz del problema. La capacidad de afinar los prompts de IA, basándose en los resultados obtenidos y en la comprensión del contexto, es un proceso que resalta la necesidad de la colaboración entre la máquina y el ser humano para abordar desafíos técnicos específicos y poco comunes.

<b>Criterio</b>	<b>Descripción</b>	<b>Evaluación Usando IA</b>	<b>Notas/Comentarios</b>
<b>1. Precisión en la asistencia</b>	La IA ofrece respuestas correctas y comandos precisos (e.g., comandos de Docker, configuraciones).	<input type="checkbox"/> No, <input checked="" type="checkbox"/> Parcial, <input type="checkbox"/> Sí	La IA ofreció respuestas correctas, pero fue necesario ajustar el prompt varias veces.
<b>2. Facilidad para debuggear</b>	La IA guía de manera clara y efectiva en la identificación y solución de errores en el código, si es necesario.	<input type="checkbox"/> No, <input checked="" type="checkbox"/> Parcial, <input type="checkbox"/> Sí	Fue necesario guiar a la IA en aspectos que identificábamos los desarrolladores.
<b>3. Soluciones contextualizadas</b>	La IA adapta sus respuestas al contexto y las necesidades	<input type="checkbox"/> No, <input type="checkbox"/> Parcial, <input checked="" type="checkbox"/> Sí	

	específicas del problema planteado.		
<b>4. Proactividad</b>	La IA propone mejoras, optimizaciones o pasos adicionales útiles sin necesidad de ser solicitada.	<input type="checkbox"/> No, <input checked="" type="checkbox"/> Parcial, <input type="checkbox"/> Sí	La IA propuso algunas mejoras, pero no de manera constante.
<b>5. Tiempo de respuesta</b>	La IA responde rápidamente, reduciendo el tiempo necesario para avanzar en las tareas.	<input type="checkbox"/> No, <input type="checkbox"/> Parcial, <input checked="" type="checkbox"/> Sí	
<b>6. Claridad en las explicaciones</b>	Las explicaciones y guías ofrecidas son claras y fáciles de entender.	<input type="checkbox"/> No, <input type="checkbox"/> Parcial, <input checked="" type="checkbox"/> Sí	
<b>7. Consistencia en las respuestas</b>	Las respuestas de la IA son consistentes y no contradictorias entre sí.	<input type="checkbox"/> No, <input type="checkbox"/> Parcial, <input checked="" type="checkbox"/> Sí	
<b>8. Referencias y ejemplos útiles</b>	La IA proporciona ejemplos relevantes y enlaces útiles para ampliar el entendimiento.	<input type="checkbox"/> No, <input type="checkbox"/> Parcial, <input checked="" type="checkbox"/> Sí	
<b>9. Flexibilidad</b>	La IA se adapta bien a diferentes lenguajes de programación, frameworks, y herramientas.	<input type="checkbox"/> No, <input checked="" type="checkbox"/> Parcial, <input type="checkbox"/> Sí	La IA inicialmente proporcionó respuestas adecuadas, pero perdió el contexto después de varias interacciones.
<b>10. Reducción del esfuerzo cognitivo</b>	La IA simplifica tareas complejas, haciéndolas más manejables para el desarrollador.	<input type="checkbox"/> No, <input type="checkbox"/> Parcial, <input checked="" type="checkbox"/> Sí	

Nota caso de estudio:

1. **Precisión en la asistencia:** Parcial (1 punto)
2. **Facilidad para debuggear:** Parcial (1 punto)
3. **Soluciones contextualizadas:** Sí (2 puntos)
4. **Proactividad:** Parcial (1 punto)
5. **Tiempo de respuesta:** Sí (2 puntos)

- 6. **Claridad en las explicaciones:** Sí (2 puntos)
- 7. **Consistencia en las respuestas:** Sí (2 puntos)
- 8. **Referencias y ejemplos útiles:** Sí (2 puntos)
- 9. **Flexibilidad:** Parcial (1 punto)
- 10. **Reducción del esfuerzo cognitivo:** Sí (2 puntos)

**Suma de puntos:** 1 + 1 + 2 + 1 + 2 + 2 + 2 + 2 + 1 + 2 = 16 puntos

**Puntos posibles:** 20 puntos

**Puntuación final:** (16 / 20) \* 10 = 8 puntos sobre 10

#### 14 Asunto: Consultas comprobación de formato PostgreSQL

##### Caso de estudio: (Generación de Código)

Se realizarán cuestiones a la IA en relación con unas consultas codificadas para un motor de base de datos como PostgreSQL, de tal manera que analice su compatibilidad y su codificación.

##### Prompt de Entrada:

Consultas de las clases presentes en los repositorios.

##### Respuesta de MO4D:

Confunde determinados conceptos que previamente dejaba como claros y concisos.

##### Respuesta de experto:

Durante varios días se realizaron numerosas consultas a la IA, obteniendo en general respuestas bastante precisas tras ser revisadas y probadas. Sin embargo, en una ocasión, al consultarle por primera vez sobre la compatibilidad de una consulta específica, indicó que el paso de parámetros en las consultas utilizando el formato “:variable” no era compatible con PostgreSQL. Esto es incorrecto, ya que en días anteriores la IA había proporcionado la respuesta adecuada respecto a este mismo tema, lo que lleva a pensar que puede tener puntos contradictorios en conversaciones extensas perdiendo así el contexto y la veracidad en sus respuestas.

##### Análisis tras resultados:

Este último incidente pone de manifiesto que, aunque en la mayoría de los casos la herramienta ofrece respuestas correctas, no siempre se puede confiar plenamente en que mantendrá la consistencia en sus resultados. Por tanto, es esencial verificar las respuestas y no asumir que su precisión será constante.

<b>Criterio</b>	<b>Descripción</b>	<b>Evaluación Usando IA</b>	<b>Notas/Comentarios</b>
<b>1.1 Quita tiempo de desarrollo</b>	Evalúa si la IA ayuda a acelerar el	[ ] No, [X] Parcial, [ ] Sí	Fue necesario ajustar los prompts más de lo normal debido a la contradicción de

	proceso de desarrollo.		los tipos de paso de parámetros en las consultas.
<b>1.2 Cohesión en el código</b>	Las partes de código generadas funcionan bien juntas, manteniendo el contexto.	<input checked="" type="checkbox"/> No, <input type="checkbox"/> Parcial, <input type="checkbox"/> Sí	En este caso la pérdida de contexto supuso un gran problema en cuanto al requerimiento de las consultas para este caso de estudio, por lo que habría sido más efectivo el no usar a la IA debido a este error.
<b>1.3 Facilita tareas tediosas</b>	Evalúa si la IA reduce la necesidad de realizar tareas manuales repetitivas.	<input type="checkbox"/> No, <input checked="" type="checkbox"/> Parcial, <input type="checkbox"/> Sí	
<b>2. Código seguro</b>	El código generado sigue buenas prácticas de seguridad.	<input type="checkbox"/> No, <input type="checkbox"/> Parcial, <input checked="" type="checkbox"/> Sí	
<b>3. Funcionalidad recibida</b>	La funcionalidad solicitada por la IA se cumple correctamente.	<input checked="" type="checkbox"/> No, <input type="checkbox"/> Parcial, <input type="checkbox"/> Sí	La IA no cumplió con su tarea principal y en algunos casos puede provocar más problemas que soluciones.
<b>4. Código claro y entendible</b>	El código generado es legible, fácil de entender y mantener.	<input type="checkbox"/> No, <input checked="" type="checkbox"/> Parcial, <input type="checkbox"/> Sí	La inconsistencia de los pasos de parámetro provocó una revisión necesaria más exhaustiva de lo normal.

Nota caso de estudio:

1.1. **Quita tiempo de desarrollo:** Parcial (1 punto)

1.2. **Cohesión en el código:** No (0 puntos)

1.3. **Facilita tareas tediosas:** Parcial (1 punto)

2. **Código seguro:** Sí (2 puntos)

3. **Funcionalidad recibida:** No (0 puntos)

4. **Código claro y entendible:** Parcial (1 punto)

**Suma de puntos:**  $1 + 0 + 1 + 2 + 0 + 1 = 5$

**Puntos posibles:** 12

**Puntuación final:**  $(5 / 12) * 10 = 4,17$  puntos sobre 10

## **15 Asunto: Generación de pruebas unitarias con especial atención con “Mocks”**

### **Caso de estudio: (Pruebas Unitarias)**

Se solicitará a la IA que genere las pruebas unitarias correspondientes a unas clases de varios servicios java spring que hacen uso de una API sobre la cual no se tiene acceso todavía para realizar las pruebas pertinentes, por lo que se realizará la prueba de usar MO4D para comprobar si nos es de ayuda.

### **Prompt de Entrada:**

Servicios java, consultas e información necesaria para ir ajustando el prompt al lugar que precisamos.

### **Respuesta de MO4D:**

Genera la clase de test unitarios prácticamente correcta, salvo por dos constructores que no eran exactamente como los definidos.

Tras la ejecución y prueba de todos los test se comprobó que la parte de ejecución de los métodos a probar tras haber preparado los datos mockeados faltaba la comprobación o los “asserts” que indicasen o comprobasen los valores de retorno de las llamadas de esas funciones.

Varios intentos después con varias correcciones a la IA, la respuesta se iba acercando a la solución esperada. Un caso a destacar es en un método a probar sobre el cual se esperaba que devolviese una NullPointerException, pero la IA interpretó que lo que se esperaba era que no se devolviese ninguna excepción, lo cual es incorrecto y tuve que rectificar.

En cuanto a la realización de pruebas unitarias y como MO4D está desarrollado, lo que conlleva el desarrollo de las mismas exige la realización de cuestiones para ir ajustando las respuestas obtenidas a lo que busca el desarrollador, esto es que, el modo de MO4D de unit test no está muy bien configurado para responder preguntas o ajustar errores, para esto es mejor utilizar el agente LLM de desarrollo de código general.

### **Respuesta de experto:**

Una vez realizados esos pequeños ajustes en los constructores, que damos uso con los mocks, y añadiendo los asserts necesarios, la ejecución de las pruebas es satisfactoria en varios métodos.

Realizando la tarea de crear las pruebas unitarias de esta clase se tarda un tiempo mucho más elevado que con el uso de la IA.

Un caso destacable que ha ocurrido en dos clases de servicios que he proporcionado a MO4D es que, al no tener todos los constructores de las clases utilizadas en el servicio, sus respectivas declaraciones en los métodos test no eran correctas del todo, teniendo que haber ajustado el prompt enseñando esas clases para que me hiciese sus llamadas de manera correcta.

Tras unos cuantos días, quise revisar la ejecución de los test y realizar algunas modificaciones debido al uso de la región del cliente de AWS. Esto ocasionó errores en

el corto plazo que, si bien estaba haciendo un uso de la IA, las respuestas obtenidas en base a determinadas preguntas, tales como los métodos necesarios para establecer la región del AWS, me daba resultados erróneos que tras indicárselo reconocía.

**Análisis tras resultados:**

Si bien es cierto que hay que dedicar algo de tiempo en revisar el resultado que te proporciona la herramienta, para agilizar las pruebas es bastante efectivo y aplicable en un entorno de desarrollo empresarial real.

Queda claro que el uso de la IA quita tiempo de codificación, pero lo añade en revisión del código proporcionado y en sus correspondientes pruebas.

Este tipo de herramientas en el desarrollo software son útiles para proporcionarte estructuras de desarrollo, algoritmos básicos y guías de desarrollo en determinados temas que te servirán como apoyo o más bien como posible punto de partida de lo que quieres desarrollar si estás en un punto en el que no sabes por dónde empezar o donde simplemente quieres tener otro punto de vista sobre lo realizado hasta el momento en ciertos aspectos. Por el contrario, si delegas todo el desarrollo en la IA, la cohesión entre las distintas partes que supone un proyecto software puede ocasionar errores tanto en el corto como en el largo plazo, dificultando y deslegitimizando para lo que en principio está desarrollada esta herramienta de IA.

<b>Criterio</b>	<b>Descripción</b>	<b>Evaluación Usando IA</b>	<b>Notas/Comentarios</b>
<b>1. Efectividad</b>	Las pruebas unitarias son útiles y cubren los casos importantes.	<input type="checkbox"/> No, <input checked="" type="checkbox"/> Parcial, <input type="checkbox"/> Sí	Las pruebas unitarias generadas fueron útiles, pero faltaban algunos "asserts" y ajustes en los constructores.
<b>2. Completitud y Cobertura</b>	Las pruebas cubren todos los posibles casos, incluyendo edge cases.	<input type="checkbox"/> No, <input checked="" type="checkbox"/> Parcial, <input type="checkbox"/> Sí	Las pruebas cubrieron la mayoría de los casos, pero algunos edge cases no fueron considerados y tuvieron que ser incluidos.
<b>3. Capacidad cognitiva y simplicidad</b>	Las pruebas son fáciles de leer y entender sin perder eficacia.	<input type="checkbox"/> No, <input type="checkbox"/> Parcial, <input checked="" type="checkbox"/> Sí	
<b>4. Seguridad</b>	Las pruebas son independientes y consistentes, sin afectar el resto del sistema.	<input type="checkbox"/> No, <input checked="" type="checkbox"/> Parcial, <input type="checkbox"/> Sí	Las pruebas no siempre fueron independientes y consistentes, especialmente en el manejo de mocks y excepciones.

Nota caso de estudio:

1. **Efectividad:** Parcial (1 punto)
2. **Completitud y Cobertura:** Parcial (1 punto)
3. **Capacidad cognitiva y simplicidad:** Sí (2 puntos)

4. **Seguridad:** Parcial (1 punto)

**Suma de puntos:** 1 + 1 + 2 + 1 = 5 puntos

**Puntos posibles:** 8

**Puntuación final:**  $(5 / 8) * 10 = 6.25$  puntos sobre 10

Tiempo de desarrollo tradicional: 4 horas.

Tiempo de desarrollo con la IA: 3 horas.

## 16 Asunto: Docker

### Caso de estudio: (Soporte)

Se solicitará a la IA las cuestiones que surjan en base a la dockerización de los módulos software presentes en el proyecto software de desarrollo.

### Prompt de Entrada:

Cuestiones referentes a los distintos procesos que conlleva la utilización del programa Docker en un proyecto software.

### Respuesta de MO4D:

Las respuestas sobre distintas cuestiones son correctas y explicativas, describiendo los pasos a realizar y las distintas opciones posibles que se presentan dependiendo de qué consulta realices.

### Respuesta de experto:

En mi caso personal, el desconocimiento de la ejecución y el entorno de Docker (aunque si conocía su uso y función) hizo que, sin el uso de la IA, tuviese que leer la documentación y las distintas guías de uso para conocer y saber cómo ejecutar las distintas tareas referentes al uso de Docker en el proyecto.

Por otra parte, mediante MO4D, su uso y aplicación fue muy fluida y eficaz, haciendo que la tarea de dockerización de los módulos software fuese exitosa en todos los sentidos.

### Análisis tras resultados:

Aunque las tareas que conlleva la dockerización de módulos software no son muy complejas, el hecho de tener la ayuda de la herramienta de IA y que su resultado fuese el esperado hace ver que para tareas de ejecución de comandos para determinadas herramientas las cuales tienen una librería de uso definida, sea muy útil y recomendable cuando no se tiene un conocimiento o experiencia de uso previa.

<b>Criterio</b>	<b>Descripción</b>	<b>Evaluación Usando IA</b>	<b>Notas/Comentarios</b>
<b>1. Precisión en la asistencia</b>	La IA ofrece respuestas correctas y comandos precisos (e.g., comandos de Docker, configuraciones).	<input type="checkbox"/> No, <input type="checkbox"/> Parcial, <input checked="" type="checkbox"/> Sí	La IA proporcionó los comandos exactos para lo requerido en cada momento.

<b>2. Facilidad para debuggear</b>	La IA guía de manera clara y efectiva en la identificación y solución de errores en el código, si es necesario.	<input type="checkbox"/> No, <input type="checkbox"/> Parcial, <input checked="" type="checkbox"/> Sí	
<b>3. Soluciones contextualizadas</b>	La IA adapta sus respuestas al contexto y las necesidades específicas del problema planteado.	<input type="checkbox"/> No, <input type="checkbox"/> Parcial, <input checked="" type="checkbox"/> Sí	
<b>4. Proactividad</b>	La IA propone mejoras, optimizaciones o pasos adicionales útiles sin necesidad de ser solicitada.	<input type="checkbox"/> No, <input type="checkbox"/> Parcial, <input checked="" type="checkbox"/> Sí	
<b>5. Tiempo de respuesta</b>	La IA responde rápidamente, reduciendo el tiempo necesario para avanzar en las tareas.	<input type="checkbox"/> No, <input type="checkbox"/> Parcial, <input checked="" type="checkbox"/> Sí	
<b>6. Claridad en las explicaciones</b>	Las explicaciones y guías ofrecidas son claras y fáciles de entender.	<input type="checkbox"/> No, <input type="checkbox"/> Parcial, <input checked="" type="checkbox"/> Sí	
<b>7. Consistencia en las respuestas</b>	Las respuestas de la IA son consistentes y no contradictorias entre sí.	<input type="checkbox"/> No, <input type="checkbox"/> Parcial, <input checked="" type="checkbox"/> Sí	
<b>8. Referencias y ejemplos útiles</b>	La IA proporciona ejemplos relevantes y enlaces útiles para ampliar el entendimiento.	<input type="checkbox"/> No, <input type="checkbox"/> Parcial, <input checked="" type="checkbox"/> Sí	
<b>9. Flexibilidad</b>	La IA se adapta bien a diferentes lenguajes de programación, frameworks, y herramientas.	<input type="checkbox"/> No, <input type="checkbox"/> Parcial, <input checked="" type="checkbox"/> Sí	
<b>10. Reducción del esfuerzo cognitivo</b>	La IA simplifica tareas complejas, haciéndolas más manejables para el desarrollador.	<input type="checkbox"/> No, <input type="checkbox"/> Parcial, <input checked="" type="checkbox"/> Sí	

Nota caso de estudio:

1. **Precisión en la asistencia:** Sí (2 puntos)
2. **Facilidad para debuggear:** Sí (2 puntos)
3. **Soluciones contextualizadas:** Sí (2 puntos)
4. **Proactividad:** Sí (2 puntos)
5. **Tiempo de respuesta:** Sí (2 puntos)
6. **Claridad en las explicaciones:** Sí (2 puntos)
7. **Consistencia en las respuestas:** Sí (2 puntos)
8. **Referencias y ejemplos útiles:** Sí (2 puntos)
9. **Flexibilidad:** Sí (2 puntos)
10. **Reducción del esfuerzo cognitivo:** Sí (2 puntos)

**Suma de puntos:**  $2 + 2 + 2 + 2 + 2 + 2 + 2 + 2 + 2 + 2 = 20$  puntos

**Puntos posibles:** 20

**Puntuación final:**  $(20 / 20) * 10 = 10$  puntos sobre 10

## **17 Asunto: Migración módulo backend de MySQL a PostgreSQL**

### **Caso de estudio: (Prompting / Soporte)**

A medida que vamos avanzando en la migración del proyecto desde MySQL a PostgreSQL nos vamos topando con errores de conversión y consultas que precisan algún cambio.

En este caso, al desplegar el frontend con el backend nos encontramos con unas consultas pertenecientes al dashboard principal de la interfaz de usuario que tratan los datos con especificaciones de MySQL que no soporta PostgreSQL.

### **Prompt de Entrada:**

En primer lugar, como en el código encuentro un QueryBuilder que genera una clase implícitamente en la consulta y no estoy familiarizado con este tipo de procedimientos, señalo a MO4D que me explique ese fragmento de código para entender su funcionamiento.

Como en la primera respuesta obtenida da por hecho que no puedo utilizar el constructor expresión de JPA con PostgreSQL y no me cuadra, realizo en otro prompt la pregunta de por qué no podría dar uso a esa técnica de construcción de objetos dinámica de JPA en la consulta con PostgreSQL.

### **Respuesta de MO4D:**

- 1) Respuesta a la primera consulta:

Realiza una explicación detallada del funcionamiento del constructor expresión de JPA presente en ese fragmento del código.

Expone la sintaxis, los parámetros y, finalmente y sin pedirlo explícitamente, me proporciona los pasos a seguir para realizar la migración a PostgreSQL, aunque no muy detallado.

- 2) Respuesta a la segunda consulta:

Tras preguntar sobre la utilización del constructor expression de JPA con PostgreSQL, me confirma que sí se puede realizar y me proporciona la solución dada.

### Respuesta de experto:

El tiempo de desarrollo ha sido de 2 minutos de desarrollo, los cuales sin haber tenido a mano la herramienta se hubiese alargado el tiempo al tener que documentarme en base a esa característica propia de JPA y probar posteriormente su uso a la nueva implementación para la migración.

El uso de la IA, tras saber predecir su comportamiento en base a qué se solicita en el prompt, hace que la herramienta sea cada vez más efectiva en manos de quien la sepa utilizar correctamente.

Es decir, si por ejemplo hubiese dado por hecho que, como en primera instancia la IA no me dijo que podía seguir utilizando el constructor expression de JPA para la migración y en el prompt inicial no indiqué explícitamente (que mantuviese al máximo el formato de creación de la consulta tal y como estaba originalmente), no me lo tuvo en cuenta. Una vez indicada la pregunta concreta sobre este asunto, obtuve la consulta migrada utilizando esta característica de JPA sin cambiar grandes rasgos de la consulta original.

### Análisis tras resultados:

La indicación de la máxima cantidad de información disponible sobre la tarea que queremos delegar en la IA es crucial para obtener resultados buenos y útiles.

### Generación de código:

<b>Criterio</b>	<b>Descripción</b>	<b>Evaluación Usando IA</b>	<b>Notas/Comentarios</b>
<b>1.1 Quita tiempo de desarrollo</b>	Evalúa si la IA ayuda a acelerar el proceso de desarrollo.	[ ] No, [ ] Parcial, [X] Sí	
<b>1.2 Cohesión en el código</b>	Las partes de código generadas funcionan bien juntas, manteniendo el contexto.	[ ] No, [X] Parcial, [ ] Sí	Hasta que no indiqué en el prompt que utilizase la característica de JPA de los “constructor expression” no lo tuvo en cuenta.
<b>1.3 Facilita tareas tediosas</b>	Evalúa si la IA reduce la necesidad de realizar tareas manuales repetitivas.	[ ] No, [ ] Parcial, [X] Sí	
<b>2. Código seguro</b>	El código generado sigue buenas prácticas de seguridad.	[ ] No, [ ] Parcial, [X] Sí	
<b>3. Funcionalidad recibida</b>	La funcionalidad solicitada por la IA	[ ] No, [ ] Parcial, [X] Sí	

	se cumple correctamente.		
<b>4. Código claro y entendible</b>	El código generado es legible, fácil de entender y mantener.	[ ] No, [ ] Parcial, [X] Sí	

**Soporte:**

<b>Criterio</b>	<b>Descripción</b>	<b>Evaluación Usando IA</b>	<b>Notas/Comentarios</b>
<b>1. Precisión en la asistencia</b>	La IA ofrece respuestas correctas y comandos precisos (e.g., comandos de Docker, configuraciones).	[ ] No, [ ] Parcial, [X] Sí	
<b>2. Facilidad para debuggear</b>	La IA guía de manera clara y efectiva en la identificación y solución de errores en el código, si es necesario.	[ ] No, [ ] Parcial, [X] Sí	
<b>3. Soluciones contextualizadas</b>	La IA adapta sus respuestas al contexto y las necesidades específicas del problema planteado.	[ ] No, [ ] Parcial, [X] Sí	
<b>4. Proactividad</b>	La IA propone mejoras, optimizaciones o pasos adicionales útiles sin necesidad de ser solicitada.	[ ] No, [X] Parcial, [ ] Sí	
<b>5. Tiempo de respuesta</b>	La IA responde rápidamente, reduciendo el tiempo necesario para avanzar en las tareas.	[ ] No, [ ] Parcial, [X] Sí	
<b>6. Claridad en las explicaciones</b>	Las explicaciones y guías ofrecidas son claras y fáciles de entender.	[ ] No, [ ] Parcial, [X] Sí	Además de proporcionar potenciales soluciones, explica su acometido, aunque no significa que esté correcto.
<b>7. Consistencia en las respuestas</b>	Las respuestas de la IA son consistentes y	[ ] No, [X] Parcial, [ ] Sí	El no haber indicado o haberse cerciorado de que podía utilizar el

	no contradictorias entre sí.		constructor expression de JPA hizo que la solución no fuese del todo certera.
<b>8. Referencias y ejemplos útiles</b>	La IA proporciona ejemplos relevantes y enlaces útiles para ampliar el entendimiento.	<input type="checkbox"/> No, <input type="checkbox"/> Parcial, <input checked="" type="checkbox"/> Sí	
<b>9. Flexibilidad</b>	La IA se adapta bien a diferentes lenguajes de programación, frameworks, y herramientas.	<input type="checkbox"/> No, <input type="checkbox"/> Parcial, <input checked="" type="checkbox"/> Sí	
<b>10. Reducción del esfuerzo cognitivo</b>	La IA simplifica tareas complejas, haciéndolas más manejables para el desarrollador.	<input type="checkbox"/> No, <input type="checkbox"/> Parcial, <input checked="" type="checkbox"/> Sí	

Nota caso de estudio:

Generación de código:

1. **Quita tiempo de desarrollo:** Sí (2 puntos)
2. **Cohesión en el código:** Parcial (1 punto)
3. **Facilita tareas tediosas:** Sí (2 puntos)
4. **Código seguro:** Sí (2 puntos)
5. **Funcionalidad recibida:** Sí (2 puntos)
6. **Código claro y entendible:** Sí (2 puntos)

Soporte

1. **Precisión en la asistencia:** Sí (2 puntos)
2. **Facilidad para debuggear:** Sí (2 puntos)
3. **Soluciones contextualizadas:** Sí (2 puntos)
4. **Proactividad:** Parcial (1 punto)
5. **Tiempo de respuesta:** Sí (2 puntos)
6. **Claridad en las explicaciones:** Sí (2 puntos)
7. **Consistencia en las respuestas:** Parcial (1 punto)
8. **Referencias y ejemplos útiles:** Sí (2 puntos)
9. **Flexibilidad:** Sí (2 puntos)
10. **Reducción del esfuerzo cognitivo:** Sí (2 puntos)

Total de puntos:

$(2 + 1 + 2 + 2 + 2 + 2) = 11$  puntos

$(11 \text{ puntos} / 12 \text{ posibles}) * 10 = 9,16$  puntos de generación de código.

$(2 + 2 + 2 + 1 + 2 + 2 + 1 + 2 + 2 + 2) = 18$  puntos

$(18 \text{ puntos} / 20 \text{ posibles}) * 10 = 9$  puntos sobre 10 de soporte.

**Puntuación final:**  $(9 + 9,16) / 2 = 9,08$  puntos.

## **18 Asunto: Depuración de errores de ejecución en consultas PostgreSQL**

### **Caso de estudio: (Prompting / Soporte / Generación de código)**

Tras realizar la migración de todas las consultas del módulo de backend, surgen errores de ejecución al realizar las pruebas de regresión.

En este caso se solicitará a la IA que me ayude en los casos que se vayan precisando en el proceso de depuración y corrección de errores.

#### **Prompt de Entrada:**

##### **Primer error encontrado:**

En primer lugar, al intentar hacer uso de una funcionalidad del dashboard donde se muestran determinados ficheros filtrados según su tipo, salta una excepción, la cual se proporciona seguido de una explicación del problema dado a la IA.

Proporciono la excepción obtenida seguido de la contextualización a la IA de las clases involucradas en el error.

##### **Segundo prompt en base a otro error (Aumento de contexto a la conversación):**

Proporciono la excepción obtenida y el código referente al error.

##### **Tercer prompt de otro error: (Contextualiza tarea a realizar y rol de la IA):**

Necesito migrar una consulta de MySQL a PostgreSQL, la cual está codificada en una clase java con un "QueryFactory", por eso vi necesario pasar la clase entera en la que estaba presente.

Tras proporcionar la clase completa, incluí en el prompt esto que cito textualmente: "Ten en cuenta que estoy realizando una migración a postgresQL y que esto funcionaba antes en MySQL.

Eres un experto en bases de datos y me vas a ayudar a solucionar el error indicándome el porqué de este fallo y como solucionarlo para que el funcionamiento de la consulta no varíe de un motor de bases de datos a otro."

#### **Respuesta de MO4D:**

##### **Respuesta al primer error encontrado:**

Indica que la codificación de la consulta está hecha con JPQL y que existe un error de nombres en ella por el cual falla la ejecución.

### **Respuesta al segundo error encontrado:**

La respuesta al segundo prompt en base al otro error no es errónea, pero no tiene en cuenta que mi base de datos de PostgreSQL está utilizando los tipos de fecha Timestamp y esto genera errores de ejecución. Por lo que deja en claro la necesidad de ajustar el prompt a las características del proyecto.

### **Respuesta al segundo error encontrado, pero ajustando el prompt:**

Código correcto y funcional.

### **Respuesta al tercer prompt:**

Retorna la consulta compatible con el motor de PostgreSQL. Explica las distintas partes y el porqué de cada cambio.

A simple vista puede parecer que está todo correcto y los cambios parecen tener sentido.

### **Respuesta tras señalamiento de error en la respuesta al tercer prompt:**

Confirma el error que he señalado, señala que lo tendrá en cuenta y explica por qué llegó a ese error en base a sus conclusiones.

“

Tienes razón, y agradezco tu persistencia en señalar esto. Mi error fue no reevaluar completamente la situación después de considerar los cambios en el GROUP BY. Debería haber seguido este razonamiento:

1. Al añadir todos los campos necesarios al GROUP BY, cada grupo se vuelve efectivamente único.
2. Con grupos únicos, la cláusula HAVING original se vuelve válida y funcional, ya que ahora está operando sobre grupos bien definidos.
3. La función `min()` en este contexto simplemente retorna el único valor en cada grupo, que es por definición el mínimo (y el máximo) para ese grupo.
4. Por lo tanto, la comparación en el HAVING se vuelve una tautología (siempre verdadera) para cada grupo, pero cumple con los requisitos sintácticos de PostgreSQL.

Este razonamiento debería haberme llevado a la conclusión de que la cláusula HAVING original funcionaría correctamente después de ajustar el GROUP BY, sin necesidad de modificaciones adicionales.

Mi fallo estuvo en:

1. No reconsiderar el impacto completo de los cambios en el GROUP BY.
2. Asumir prematuramente que la cláusula HAVING necesitaría una modificación más compleja.

3. No evaluar paso a paso cómo los cambios en una parte de la consulta (*GROUP BY*) afectarían a otras partes (*HAVING*).

Esta es una lección valiosa sobre la importancia de:

- Reevaluar constantemente todas las partes de una consulta cuando se hacen cambios.
- No hacer suposiciones sin verificar el impacto completo de los cambios.
- Considerar siempre la solución más simple y directa antes de proponer cambios más complejos.

Agradezco que hayas insistido en este punto, ya que me ayuda a mejorar mi proceso de análisis y a ser más cuidadoso en futuras evaluaciones de consultas SQL.

”

### **Respuesta de experto:**

Después de un tiempo interactuando con la IA y ajustando el prompt de entrada, concluyo que, al estar la consulta codificada en JPQL, el acceso a las tablas se realiza a través de los nombres de las variables del código en lugar de los nombres nativos de la base de datos. Esto generó un problema cuando, debido a un error en la definición del nombre de una clase relacionada con “tabla\_ejemplo” en el código, la tabla correspondiente no se reconoció, resultando en el error descrito.

En cuanto al segundo problema, la falta de contexto suficiente en las indicaciones iniciales afectó el resultado obtenido. En la primera consulta relacionada con este error, no especificué de manera explícita la necesidad de tratar las fechas utilizando “Timestamp”, lo que ocasionó errores de ejecución. Sin embargo, tras ajustar el prompt para resaltar la obligatoriedad del tratamiento de fechas con “Timestamp”, la respuesta fue correcta y funcional.

En el tercer caso, asigné a la IA un rol específico como experta en bases de datos, indicando claramente que el comportamiento esperado de la consulta debía permanecer inmutable tras la migración de MySQL a PostgreSQL. Aunque el resultado no presentó mejoras significativas respecto a otros casos analizados, esto era de esperarse, dado que la herramienta de IA utilizada está configurada con un rol por debajo.

Es importante subrayar que la instrucción de mantener el resultado final inalterado fue clave para este análisis, el cual detallaremos más adelante. La IA no solo pretendió asegurar que el comportamiento esperado se conservara tras la migración, sino que también explicó los cambios realizados en la consulta para adaptarla a las particularidades de PostgreSQL.

Durante las pruebas de esta consulta modificada, noté que no obtenía la lista de ficheros esperada utilizando la consulta retornada por la IA. Tras analizar las cláusulas de la consulta generada, llegué a la conclusión de que, en la consulta original funcional en MySQL, solo era necesario añadir dos campos adicionales al *GROUP BY* y dejar la cláusula *HAVING* intacta. Este ajuste permitió obtener el resultado deseado.

Posteriormente, compartí con la IA el procedimiento seguido, explicando las razones detrás de cada decisión y los posibles errores en su respuesta inicial. La IA explicó que, al no haber considerado desde el principio los campos adicionales que yo incorporé al *GROUP BY*, la cláusula *HAVING* no funcionaba correctamente, lo que derivó en una modificación más compleja de lo necesario.

Curiosamente, la solución propuesta por MO4D incluía los campos necesarios en el *GROUP BY*, pero también introducía un cambio innecesario en la cláusula *HAVING*, provocando que la consulta no funcionara como se esperaba.

En este último caso, podría haber sido más efectivo especificar en el prompt que requería una solución lo más simple y funcional posible, además de incluir todas las precondiciones mencionadas previamente.

### **Análisis tras resultados:**

Caso de estudio que realiza la importancia de un buen prompt que contextualice al máximo los parámetros y necesidades de la respuesta a obtener.

Un factor importante que destacar es que sobre la respuesta al tercer prompt, si das por buena la solución obtenida y no realizas las pruebas correspondientes para validar el resultado, puede desembocar en errores de funcionamiento que provoquen un efecto dominó a largo plazo.

Debido a la longitud de este caso, creo correspondiente realizar la evaluación en base a las tablas de generación de código y soporte, ya que abarca ambos campos.

Generación de código:

<b>Criterio</b>	<b>Descripción</b>	<b>Evaluación Usando IA</b>	<b>Notas/Comentarios</b>
<b>1.1 Quita tiempo de desarrollo</b>	Evalúa si la IA ayuda a acelerar el proceso de desarrollo.	<input checked="" type="checkbox"/> No, <input type="checkbox"/> Parcial, <input type="checkbox"/> Sí	En este caso las soluciones dadas me hicieron perder tiempo ya que la verdadera solución era más simple que la proporcionada por la IA.
<b>1.2 Cohesión en el código</b>	Las partes de código generadas funcionan bien juntas, manteniendo el contexto.	<input type="checkbox"/> No, <input checked="" type="checkbox"/> Parcial, <input type="checkbox"/> Sí	Fueron necesarios ajustes.
<b>1.3 Facilita tareas tediosas</b>	Evalúa si la IA reduce la necesidad de realizar tareas manuales repetitivas.	<input type="checkbox"/> No, <input type="checkbox"/> Parcial, <input checked="" type="checkbox"/> Sí	
<b>2. Código seguro</b>	El código generado sigue buenas	<input type="checkbox"/> No, <input type="checkbox"/> Parcial, <input checked="" type="checkbox"/> Sí	

	prácticas de seguridad.		
<b>3. Funcionalidad recibida</b>	La funcionalidad solicitada por la IA se cumple correctamente.	[X] No, [ ] Parcial, [ ] Sí	Debido al error que tuvo en cuanto al tratamiento de la cláusula HAVING.
<b>4. Código claro y entendible</b>	El código generado es legible, fácil de entender y mantener.	[ ] No, [ ] Parcial, [X] Sí	

Soporte:

<b>Criterio</b>	<b>Descripción</b>	<b>Evaluación Usando IA</b>	<b>Notas/Comentarios</b>
<b>1. Precisión en la asistencia</b>	La IA ofrece respuestas correctas y comandos precisos (e.g., comandos de Docker, configuraciones).	[ ] No, [ ] Parcial, [X] Sí	Identificó correctamente los errores y proporcionó soluciones detalladas.
<b>2. Facilidad para debuggear</b>	La IA guía de manera clara y efectiva en la identificación y solución de errores en el código, si es necesario.	[ ] No, [ ] Parcial, [X] Sí	
<b>3. Soluciones contextualizadas</b>	La IA adapta sus respuestas al contexto y las necesidades específicas del problema planteado.	[ ] No, [ ] Parcial, [X] Sí	
<b>4. Proactividad</b>	La IA propone mejoras, optimizaciones o pasos adicionales útiles sin necesidad de ser solicitada.	[ ] No, [X] Parcial, [ ] Sí	
<b>5. Tiempo de respuesta</b>	La IA responde rápidamente, reduciendo el tiempo necesario para avanzar en las tareas.	[ ] No, [X] Parcial, [ ] Sí	
<b>6. Claridad en las explicaciones</b>	Las explicaciones y guías ofrecidas son claras y fáciles de entender.	[ ] No, [X] Parcial, [ ] Sí	En cuanto a la solución errónea presentaba inconsistencias.

<b>7. Consistencia en las respuestas</b>	Las respuestas de la IA son consistentes y no contradictorias entre sí.	<input type="checkbox"/> No, <input checked="" type="checkbox"/> Parcial, <input type="checkbox"/> Sí	
<b>8. Referencias y ejemplos útiles</b>	La IA proporciona ejemplos relevantes y enlaces útiles para ampliar el entendimiento.	<input type="checkbox"/> No, <input checked="" type="checkbox"/> Parcial, <input type="checkbox"/> Sí	
<b>9. Flexibilidad</b>	La IA se adapta bien a diferentes lenguajes de programación, frameworks, y herramientas.	<input type="checkbox"/> No, <input type="checkbox"/> Parcial, <input checked="" type="checkbox"/> Sí	
<b>10. Reducción del esfuerzo cognitivo</b>	La IA simplifica tareas complejas, haciéndolas más manejables para el desarrollador.	<input type="checkbox"/> No, <input type="checkbox"/> Parcial, <input checked="" type="checkbox"/> Sí	

Nota caso de estudio:

Generación de código:

- 1.1. **Quita tiempo de desarrollo:** No (0 puntos)
- 1.2. **Cohesión en el código:** Parcial (1 puntos)
- 1.3. **Facilita tareas tediosas:** Sí (2 puntos)
- 2. **Código seguro:** Sí (2 puntos)
- 3. **Funcionalidad recibida:** No (0 puntos)
- 4. **Código claro y entendible:** Sí (2 puntos)

Soporte:

- 1. **Precisión en la asistencia:** Sí (2 puntos)
- 2. **Facilidad para debuggear:** Sí (2 puntos)
- 3. **Soluciones contextualizadas:** Sí (2 puntos)
- 4. **Proactividad:** Parcial (1 punto)
- 5. **Tiempo de respuesta:** Parcial (1 punto)
- 6. **Claridad en las explicaciones:** Parcial (1 punto)
- 7. **Consistencia en las respuestas:** Parcial (1 punto)
- 8. **Referencias y ejemplos útiles:** Parcial (1 punto)
- 9. **Flexibilidad:** Sí (2 puntos)
- 10. **Reducción del esfuerzo cognitivo:** Sí (2 puntos)

**Suma de puntos:**

$(0 + 1 + 2 + 2 + 0 + 2) = 7$  puntos

$(7 \text{ puntos} / 12 \text{ posibles}) * 10 = 5,83$  puntos de generación de código.

$$(2 + 2 + 2 + 1 + 1 + 1 + 1 + 1 + 2 + 2) = 15 \text{ puntos}$$

$$(15 \text{ puntos} / 20 \text{ posibles}) * 10 = 7,5 \text{ puntos de soporte.}$$

**Puntuación final:**  $(5,83 + 7,5) / 2 = 6,66 \text{ puntos.}$

## 19 Asunto: Migración simple de consulta UPDATE de MySQL a PostgreSQL

### Caso de estudio: (Prompting / Soporte / Generación de código)

Se requiere la migración de una consulta que actualiza el valor de un campo de una determinada tabla de la base de datos de PostgreSQL.

Este caso pretende ser lo más simple posible en cuanto al requerimiento solicitado, de tal manera que se puedan observar resultados hasta en los detalles más simples posibles de una tarea usual para un desarrollador como es la siguiente.

#### **Prompt de Entrada:**

##### **Prompt inicial (Contexto inicial):**

Estoy realizando una migración de MySQL a PostgreSQL, quiero que me realices la migración de esta consulta, ten en cuenta que quiero que el resultado a obtener de la consulta sea el mismo independientemente del motor de bases de datos en el que se ejecute:

```
UPDATE tabla_ejemplo t
    SET t.fecha_baja = DATE_FORMAT(DATE(:fecha), '%Y-%m-%d')
    WHERE t.id_tabla = :id
```

##### **Segundo prompt tras error encontrado (descrito en el apartado “Respuesta de experto”):**

El acceso a las tablas mediante acrónimos no está permitido en PostgreSQL, aun así, en tu solución tomaste por bueno este recurso.

#### **Respuesta de MO4D:**

##### **Primera respuesta obtenida:**

Para migrar esta consulta de MySQL a PostgreSQL, manteniendo el mismo resultado, necesitamos hacer algunos ajustes en la función de formateo de fecha. Aquí está la versión equivalente para PostgreSQL:

```
``sql
UPDATE tabla_ejemplo t
SET t.fecha_baja = TO_CHAR(CAST(:fecha AS DATE), 'YYYY-MM-DD')
WHERE t.id_tabla = :id;
``
```

Explicación de los cambios:

1. `DATE\_FORMAT` en MySQL se reemplaza por `TO\_CHAR` en PostgreSQL.
2. `DATE(:fecha)` se cambia a `CAST(:fecha AS DATE)` para asegurar que el valor se trate como una fecha.

3. El formato de fecha '%Y-%m-%d' en MySQL se cambia a 'YYYY-MM-DD' en PostgreSQL.

Esta consulta en PostgreSQL producirá el mismo resultado que la consulta original en MySQL, formateando la fecha en el formato 'YYYY-MM-DD'.

Notas adicionales:

- Asegúrate de que el parámetro `:fecha` se pase en un formato que PostgreSQL pueda interpretar correctamente como fecha.

- Si `:fecha` ya es de tipo DATE en tu aplicación, puedes simplificar la consulta aún más:

```
```sql
UPDATE tabla_ejemplo t
SET t.fecha_baja = :fecha::DATE
WHERE t.id_tabla = :id;
```
```

Esto asume que `:fecha` ya es una fecha válida y simplemente la convierte al tipo DATE de PostgreSQL.

- Si necesitas mantener exactamente el mismo formato de consulta para ambos sistemas de gestión de bases de datos, podrías crear una función personalizada en PostgreSQL que imite el comportamiento de `DATE\_FORMAT` de MySQL, pero en la mayoría de los casos, usar las funciones nativas de PostgreSQL como se muestra arriba es la mejor práctica.

#### **Respuesta al error identificado del segundo prompt:**

Entiendo tu problema. El error que estás experimentando se debe a que PostgreSQL maneja los alias de tabla de manera diferente en las cláusulas UPDATE comparado con MySQL. En PostgreSQL, no puedes usar el alias de tabla en la parte SET de una declaración UPDATE.

Para corregir esto, simplemente elimina el alias 't' en la cláusula SET. Aquí tienes la versión corregida que funcionará en PostgreSQL:

```
```sql
UPDATE tabla_ejemplo
SET fecha_baja = :fecha::DATE
WHERE id_tabla = :id;
```
```

Esta versión debería funcionar correctamente en PostgreSQL. Aquí están los cambios y explicaciones:

1. Eliminamos el alias 't' después de 'tabla\_ejemplo en la línea UPDATE.
2. En la cláusula SET, usamos directamente 'fecha\_baja' sin el prefijo 't'.
3. En la cláusula WHERE, también usamos directamente 'id\_tabla sin el prefijo 't'.

Esta sintaxis es compatible con PostgreSQL y realizará la misma operación que la consulta original de MySQL. Asegúrate de que :fecha se pase como un valor que PostgreSQL pueda interpretar como fecha, y :id como el tipo de dato correcto para id\_tabla.

### **Respuesta de experto:**

En cuanto a la primera respuesta obtenida de MO4D, al no haber indicado que el campo de la fecha presente en la tabla a modificar es del tipo Date, como en la consulta original de MySQL el tratamiento de esa fecha en específico se hace con String, para la solución de PostgreSQL también lo trata como si la fecha en este caso fuese del tipo String.

Una vez aclarado esto tras haberme dado cuenta de no haber considerado ese detalle, MO4D me retorna la consulta deseada, pero con un error que, aunque no es muy grave, sigue siendo algo a tener en cuenta. Esto es que, en MySQL las operaciones de “UPDATE” pueden utilizar los acrónimos de tablas (“tabla\_ejemplo t”, por ejemplo), pero en PostgreSQL esto no está permitido.

Cuando identifiqué ese error tras intentar ejecutar la consulta, se lo indicé a MO4D, la cual me da la razón sobre el error mencionado y corrige la respuesta indicando el porqué del fallo encontrado.

Esto último es interesante ya que pone en claro que la IA sí que precisaba de la respuesta correcta pero el problema fue que al no ajustar el prompt lo suficiente en cuanto al requerimiento de una sintaxis 100% compatible con PostgreSQL y estricta, el error en la respuesta era más probable y es lo que ha sucedido.

### **Análisis tras resultados:**

En base a lo último mencionado en mi análisis, la conclusión es que, aunque el indicar en el prompt que preciso de una migración de MySQL a PostgreSQL donde la sintaxis sea 100% estricta según lo permitido en PostgreSQL es algo que como desarrollador es algo obvio, debemos tener en cuenta que la IA es estadística aplicada a un entrenamiento previo que, como hemos visto, sí que tenía acceso a la información de que esa aplicación de acrónimos en consultas “UPDATE” de PostgreSQL no están permitidas, pero el problema fue que en este caso se dio la casualidad de que el error obtenido fue debido a un ajuste algo pobre del prompt inicial.

Un ejemplo de lo que podría haber sido un buen prompt para este caso sería el siguiente:

“

Por favor, realiza la migración precisa de la siguiente consulta JPA/MySQL a PostgreSQL, asegurando la compatibilidad total y explicando los cambios necesarios:

```
@Query("""
    UPDATE tabla_ejemplo t
    SET t.fecha_baja = DATE_FORMAT(DATE(:fecha), '%Y-%m-%d')
    WHERE t.id_tabla = :id
    """)
```

@Modifying

```
void updateFechaBaja(LocalDate fecha, int id);
```

Considera las siguientes especificaciones:

1. La base de datos destino es PostgreSQL.
2. La columna fecha\_baja es de tipo DATE en PostgreSQL.
3. El parámetro fecha es de tipo LocalDate.
4. Asegúrate de que la sintaxis sea 100% compatible con PostgreSQL, incluyendo el manejo de alias y funciones de fecha.
5. Explica cada cambio realizado y por qué es necesario.
6. Proporciona la consulta final en formato JPA con las anotaciones correspondientes.

”

Este caso pone en relieve otra vez la importancia de la creación de un buen prompt con la máxima información posible.

**Generación de código:**

| <b>Criterio</b>                       | <b>Descripción</b>   | <b>Evaluación Usando IA</b>   | <b>Notas/Comentarios</b>  |
|---------------------------------------|--|---|---|
| <b>1.1 Quita tiempo de desarrollo</b> | Evalúa si la IA ayuda a acelerar el proceso de desarrollo.                     | <input checked="" type="checkbox"/> No<br><input type="checkbox"/> Parcial,<br><input type="checkbox"/> Si  | En este caso, las soluciones dadas me hicieron perder tiempo ya que la verdadera solución era más simple que la proporcionada por la IA.              |
| <b>1.2 Cohesión en el código</b>      | Las partes de código generadas funcionan bien juntas, manteniendo el contexto. | <input type="checkbox"/> No,<br><input checked="" type="checkbox"/> Parcial,<br><input type="checkbox"/> Sí | Fueron necesarios ajustes. Aunque la IA proporcionó una solución funcional, no consideró todas las particularidades de PostgreSQL desde el principio, |

|                                     |  |   |   |
|-------------------------------------|--|---|---|
|                                     |  |   | lo que afectó la cohesión del código generado.  |
| <b>1.3 Facilita tareas tediosas</b> | Evalúa si la IA reduce la necesidad de realizar tareas manuales repetitivas. | <input type="checkbox"/> No,<br><input type="checkbox"/> Parcial,<br><input checked="" type="checkbox"/> Sí |   |
| <b>2. Código seguro</b>             | El código generado sigue buenas prácticas de seguridad.                      | <input type="checkbox"/> No,<br><input type="checkbox"/> Parcial,<br><input checked="" type="checkbox"/> Sí |   |
| <b>3. Funcionalidad recibida</b>    | La funcionalidad solicitada por la IA se cumple correctamente.               | <input checked="" type="checkbox"/> No,<br><input type="checkbox"/> Parcial,<br><input type="checkbox"/> Sí | Debido al error en el tratamiento de la cláusula HAVING y el uso incorrecto de alias en la consulta UPDATE, la funcionalidad no se cumplió correctamente en la primera respuesta. |
| <b>4. Código claro y entendible</b> | El código generado es legible, fácil de entender y mantener.                 | <input type="checkbox"/> No,<br><input type="checkbox"/> Parcial,<br><input checked="" type="checkbox"/> Sí |   |

Soporte:

| <b>Criterio</b>                      | <b>Descripción</b>  | <b>Evaluación Usando IA</b>   | <b>Notas/Comentarios</b> |
|--------------------------------------|---|---|--------------------------|
| <b>1. Precisión en la asistencia</b> | La IA ofrece respuestas correctas y comandos precisos (e.g., comandos de Docker, configuraciones).              | <input type="checkbox"/> No,<br><input type="checkbox"/> Parcial,<br><input checked="" type="checkbox"/> Sí |                          |
| <b>2. Facilidad para debuggear</b>   | La IA guía de manera clara y efectiva en la identificación y solución de errores en el código, si es necesario. | <input type="checkbox"/> No,<br><input type="checkbox"/> Parcial,<br><input checked="" type="checkbox"/> Sí |                          |

|   |   |   |  |
|---|---|---|--|
| <b>3. Soluciones contextualizadas</b>       | La IA adapta sus respuestas al contexto y las necesidades específicas del problema planteado.     | <input type="checkbox"/> No,<br><input type="checkbox"/> Parcial,<br><input checked="" type="checkbox"/> Sí |  |
| <b>4. Proactividad</b>                      | La IA propone mejoras, optimizaciones o pasos adicionales útiles sin necesidad de ser solicitada. | <input type="checkbox"/> No,<br><input checked="" type="checkbox"/> Parcial,<br><input type="checkbox"/> Sí |  |
| <b>5. Tiempo de respuesta</b>               | La IA responde rápidamente, reduciendo el tiempo necesario para avanzar en las tareas.            | <input type="checkbox"/> No,<br><input checked="" type="checkbox"/> Parcial,<br><input type="checkbox"/> Sí | Las respuestas fueron rápidas, pero la necesidad de ajustes adicionales debido a errores iniciales afectó el tiempo total de desarrollo. |
| <b>6. Claridad en las explicaciones</b>     | Las explicaciones y guías ofrecidas son claras y fáciles de entender.                             | <input type="checkbox"/> No,<br><input checked="" type="checkbox"/> Parcial,<br><input type="checkbox"/> Sí |  |
| <b>7. Consistencia en las respuestas</b>    | Las respuestas de la IA son consistentes y no contradictorias entre sí.                           | <input type="checkbox"/> No,<br><input checked="" type="checkbox"/> Parcial,<br><input type="checkbox"/> Sí | Las respuestas fueron coherentes en su mayoría, pero la necesidad de correcciones indicó una falta de consistencia inicial.              |
| <b>8. Referencias y ejemplos útiles</b>     | La IA proporciona ejemplos relevantes y enlaces útiles para ampliar el entendimiento.             | <input type="checkbox"/> No,<br><input type="checkbox"/> Parcial,<br><input checked="" type="checkbox"/> Sí |  |
| <b>9. Flexibilidad</b>                      | La IA se adapta bien a diferentes lenguajes de programación, frameworks, y herramientas.          | <input type="checkbox"/> No,<br><input type="checkbox"/> Parcial,<br><input checked="" type="checkbox"/> Sí | La IA se adaptó bien a las necesidades específicas del proyecto, manejando tanto MySQL como PostgreSQL.                                  |
| <b>10. Reducción del esfuerzo cognitivo</b> | La IA simplifica tareas complejas, haciéndolas más  | <input type="checkbox"/> No,<br><input type="checkbox"/> Parcial,   | La IA simplificó tareas complejas, facilitando el trabajo del desarrollador  |

|  |                                   |        |   |
|--|-----------------------------------|--------|---|
|  | manejables para el desarrollador. | [X] Sí | y reduciendo el esfuerzo cognitivo necesario. |
|--|-----------------------------------|--------|---|

Generación de código:

- 1.1. **Quita tiempo de desarrollo:** No (0 puntos)
- 1.2. **Cohesión en el código:** Parcial (1 punto)
- 1.3. **Facilita tareas tediosas:** Sí (2 puntos)
- 2. **Código seguro:** Sí (2 puntos)
- 3. **Funcionalidad recibida:** No (0 puntos)
- 4. **Código claro y entendible:** Sí (2 puntos)

Soporte:

- 1. **Precisión en la asistencia:** Sí (2 puntos)
- 2. **Facilidad para debuggear:** Sí (2 puntos)
- 3. **Soluciones contextualizadas:** Sí (2 puntos)
- 4. **Proactividad:** Parcial (1 punto)
- 5. **Tiempo de respuesta:** Parcial (1 punto)
- 6. **Claridad en las explicaciones:** Parcial (1 punto)
- 7. **Consistencia en las respuestas:** Parcial (1 punto)
- 8. **Referencias y ejemplos útiles:** Parcial (2 punto)
- 9. **Flexibilidad:** Sí (2 puntos)
- 10. **Reducción del esfuerzo cognitivo:** Sí (2 puntos)

Total de puntos:

$$(0 + 1 + 2 + 2 + 0 + 2) = 7 \text{ puntos}$$

$$(7 \text{ puntos} / 12 \text{ posibles}) * 10 = 5,8 \text{ puntos de generación de código.}$$

$$(2 + 2 + 2 + 1 + 1 + 1 + 1 + 2 + 2 + 2) = 16 \text{ puntos}$$

$$(16 \text{ puntos} / 20 \text{ posibles}) * 10 = 8 \text{ puntos de soporte.}$$

**Puntuación final:**  $(5,8 + 8) / 2 = 6,9$  puntos.

## 20 Asunto: Migración de MySQL a PostgreSQL

### Caso de estudio: (Prompting / Generación de código / Soporte)

Se estudiará otro caso de estudio referente a un prompting más específico y detallado siguiendo una estructura marcada que sabemos que funciona con los LLM.

### Prompt de Entrada:

**Primer Prompt: (Técnica de “Consultor iniciado” + Ampliar contexto + Listar requerimientos)**

Soy un iniciado en migraciones de MySQL a PostgreSQL de consultas y necesito que tú, como experto en este tipo de tareas y de bases de datos me realices la correspondiente migración de estas consultas que están en MySQL a PostgreSQL.

Los requerimientos son los siguientes:

- El resultado a obtener de la consulta debe ser el mismo que en MySQL.
- La sintaxis debe ser 100% compatible con PostgreSQL
- Los cambios aplicados deben ser los menos posibles de tal manera que si una consulta es compatible con MySQL, se quede igual.

La lista de consultas es esta:

<Introduzco la lista de las consultas identificadas por un número cada una>

### **Segundo prompt (Aumento de contexto de la conversación + ejemplo):**

Al realizar los cambios pertinentes sobre la primera consulta que me has proporcionado como solución, me aparece en el log del módulo esta excepción: <Inserto excepción>.

Para ponerte más en contexto, este servicio es el que trata estas consultas y las ejecuta:

<Inserto el servicio Java pertinente>

### **Tercer prompt (Aumento del contexto de la conversación):**

Indico que la zona horaria de las fechas debe estar presente para poder hacer uso de *INTERVAL*.

### **Respuesta de MO4D:**

#### **Respuesta a primer prompt:**

Obtengo una lista de las consultas a migrar según los requerimientos que he marcado. Al final de todo añade unas notas sobre los cambios:

Aquí tienes la migración de las consultas de MySQL a PostgreSQL, teniendo en cuenta los requerimientos que mencionaste:

<Lista de consultas migradas>

### Notas sobre los cambios:

- En PostgreSQL, la función `ADDDATE()` no existe. Se utiliza la resta o suma de intervalos directamente (por ejemplo, `:fecha - INTERVAL '1 DAY')`).
- Se mantiene la estructura de las consultas lo más similar posible a la original para facilitar la migración y comprensión.
- Asegúrate de que los tipos de datos y otras configuraciones en tus tablas sean compatibles entre MySQL y PostgreSQL, ya que esto podría afectar los resultados.

### Respuesta a segundo prompt:

Señala que ha entendido el contexto proporcionado sobre ese error (En este caso yo era conocedor del problema que había presente, pero quería probar como trataría la IA este aumento de contexto con el siguiente prompt que proporcione).

### Respuesta a tercer prompt:

Comprende el problema dado y tras razonar el porqué, proporciona la codificación de la clase adaptada y los cambios a las consultas que he de realizar para poder hacer uso de la operación INTERVAL de PostgreSQL.

Explícitamente:

“Sí, tienes razón. Si la base de datos utiliza timestamp with time zone o necesita trabajar con zonas horarias específicas, debes asegurarte de que las fechas que pasas como parámetros incluyan la información de zona horaria adecuada.

PostgreSQL diferencia entre:

1. **timestamp without time zone:** Almacena una fecha y hora sin información de zona horaria. Las operaciones con INTERVAL se realizan directamente.
2. **timestamp with time zone:** Almacena una fecha y hora con información de zona horaria. Las operaciones consideran la zona horaria.

Si tu columna fe\_reinicio es de tipo timestamp with time zone, necesitas incluir explícitamente una zona horaria para evitar errores en las comparaciones o cálculos.

<Inserta modificaciones en el servicio Java seguido de la explicación>

1. **Verifica la zona horaria en tu base de datos:** Consulta la zona horaria configurada en PostgreSQL con:
2. SHOW TIME ZONE;
3. **Ajusta el tipo de dato de la columna:**  
Asegúrate de que la columna fe\_reinicio sea compatible con las operaciones de zona horaria (timestamp with time zone).
4. **Prueba la consulta directamente en la base de datos:**
5. SELECT \*
6. FROM tabla\_ejemplo
7. WHERE fe\_reinicio >= (CAST('2025-01-23 00:00:00' AS TIMESTAMPTZ) AT TIME ZONE 'Europe/Madrid' - INTERVAL '7 DAY')
8. AND fe\_reinicio < (CAST('2025-01-23 00:00:00' AS TIMESTAMPTZ) AT TIME ZONE 'Europe/Madrid' + INTERVAL '1 DAY');

Este enfoque asegura que las comparaciones funcionen correctamente, especialmente si trabajas en un entorno con múltiples zonas horarias.

”

### Respuesta de experto:

La lista de consultas migradas no se ejecutaba correctamente ya que el uso de la función *INTERVAL* requiere del time zone en la fecha a comparar y esta no disponía de ella. Aun así, esto no era un error de la IA ya que la fecha se pasa por parámetro y la solución dada era en una parte del código en la parametrización de estas fechas.

En cuanto a la solución dada sobre el tercer prompt, no es del todo aplicable a lo que requería como objetivo final, pero solamente necesité realizar un pequeño ajuste en cómo se obtenían las fechas parametrizadas del método java del servicio con la consulta dada y funcionó perfectamente.

### Análisis tras resultados:

El tiempo de codificación y la proporción de posibles soluciones en base al problema encontrado de las zonas de tiempo de las fechas Timestamp ha sido de gran ayuda solo habiendo tenido que invertir tiempo en el *debuggeo* y el cambio de las consultas.

De todas formas, en este caso, como en la mayoría de ellos, el papel del ingeniero sigue siendo fundamental a la hora de integrar las posibles soluciones dadas de la IA y su ajuste al contexto de la aplicación con su consiguiente depuración.

### Generación de código:

| Criterio                              | Descripción  | Evaluación Usando IA   | Notas/Comentarios   |
|---------------------------------------|--|--|---|
| <b>1.1 Quita tiempo de desarrollo</b> | Evalúa si la IA ayuda a acelerar el proceso de desarrollo.                     | <input type="checkbox"/> No,<br><input type="checkbox"/> Parcial,<br><input checked="" type="checkbox"/> Sí            |   |
| <b>1.2 Cohesión en el código</b>      | Las partes de código generadas funcionan bien juntas, manteniendo el contexto. | <input type="checkbox"/> No,<br><input checked="" type="checkbox"/> Parcial,<br><input type="checkbox"/> Sí            | El tratamiento de la cláusula <i>INTERVAL</i> fue algo conflictiva. |
| <b>1.3 Facilita tareas tediosas</b>   | Evalúa si la IA reduce la necesidad de realizar tareas manuales repetitivas.   | <input type="checkbox"/> No,<br><input type="checkbox"/> Parcial,<br><input checked="" type="checkbox"/> Sí            |   |
| <b>2. Código seguro</b>               | El código generado sigue buenas prácticas de seguridad.                        | <input type="checkbox"/> No,<br><input type="checkbox"/> Parcial,<br><input checked="" type="checkbox"/> Sí            |   |
| <b>3. Funcionalidad recibida</b>      | La funcionalidad solicitada por la IA se cumple correctamente.                 | <input type="checkbox"/> No,<br><input checked="" type="checkbox"/> Parcial,<br><input checked="" type="checkbox"/> Sí | Necesidad de implementar el tiempo de la zona en la fecha.          |
| <b>4. Código claro y entendible</b>   | El código generado es legible, fácil de entender y mantener.                   | <input type="checkbox"/> No,<br><input type="checkbox"/> Parcial,<br><input checked="" type="checkbox"/> Sí            |   |

### Soporte:

| <b>Criterio</b>                   | <b>Descripción</b>   | <b>Evaluación Usando IA</b>  | <b>Notas/Comentarios</b>                          |
|-----------------------------------|--|--|---|
| 1. Precisión en la asistencia     | La IA ofrece respuestas correctas y comandos precisos (e.g., comandos de Docker, configuraciones). | <input type="checkbox"/> No<br><input type="checkbox"/> Parcial,<br><input checked="" type="checkbox"/> Si |   |
| 2. Facilidad para debuggear       | La IA guía de manera clara y efectiva en la identificación y solución de errores en el código.     | <input type="checkbox"/> No<br><input type="checkbox"/> Parcial,<br><input checked="" type="checkbox"/> Si |   |
| 3. Soluciones contextualizadas    | La IA adapta sus respuestas al contexto y las necesidades específicas del problema planteado.      | <input type="checkbox"/> No<br><input type="checkbox"/> Parcial,<br><input checked="" type="checkbox"/> Si |   |
| 4. Proactividad                   | La IA propone mejoras, optimizaciones o pasos adicionales útiles sin necesidad de ser solicitada.  | <input type="checkbox"/> No<br><input checked="" type="checkbox"/> Parcial,<br><input type="checkbox"/> Si | No sugirió soluciones antes de aumentar contexto. |
| 5. Tiempo de respuesta            | La IA responde rápidamente, reduciendo el tiempo necesario para avanzar en las tareas.             | <input type="checkbox"/> No<br><input checked="" type="checkbox"/> Parcial,<br><input type="checkbox"/> Si | Se necesitó hacer ajustes manuales adicionales.   |
| 6. Claridad en las explicaciones  | Las explicaciones y guías ofrecidas son claras y fáciles de entender.                              | <input type="checkbox"/> No<br><input type="checkbox"/> Parcial,<br><input checked="" type="checkbox"/> Si |   |
| 7. Consistencia en las respuestas | Las respuestas de la IA son consistentes y no contradictorias entre sí.                            | <input type="checkbox"/> No<br><input checked="" type="checkbox"/> Parcial,<br><input type="checkbox"/> Si | Hubo correcciones necesarias.                     |
| 8. Referencias y ejemplos útiles  | La IA proporciona ejemplos relevantes y enlaces útiles para ampliar el entendimiento.              | <input type="checkbox"/> No<br><input type="checkbox"/> Parcial,<br><input checked="" type="checkbox"/> Si |   |
| 9. Flexibilidad                   | La IA se adapta bien a diferentes lenguajes de programación, frameworks y herramientas.            | <input type="checkbox"/> No<br><input type="checkbox"/> Parcial,<br><input checked="" type="checkbox"/> Si | Se adaptó bien a MySQL y PostgreSQL.              |

|                                      |  |  |  |
|--------------------------------------|--|--|--|
| 10. Reducción del esfuerzo cognitivo | La IA simplifica tareas complejas, haciéndolas más manejables para el desarrollador. | <input type="checkbox"/> No<br><input type="checkbox"/> Parcial,<br><input checked="" type="checkbox"/> Sí |  |
|--------------------------------------|--|--|--|

### Cálculo de Puntuación Final

Generación de código:

1. **Quita tiempo de desarrollo:** Sí (2 puntos)
2. **Cohesión en el código:** Parcial (1 punto)
3. **Facilita tareas tediosas:** Sí (2 puntos)
4. **Código seguro:** Sí (2 puntos)
5. **Funcionalidad recibida:** Sí (2 puntos)
6. **Código claro y entendible:** Sí (2 puntos)

Soporte:

1. **Precisión en la asistencia:** Sí (2 puntos)
2. **Facilidad para debuggear:** Sí (2 puntos)
3. **Soluciones contextualizadas:** Sí (2 puntos)
4. **Proactividad:** Parcial (1 punto)
5. **Tiempo de respuesta:** Parcial (1 punto)
6. **Claridad en las explicaciones:** Sí (2 puntos)
7. **Consistencia en las respuestas:** Parcial (1 punto)
8. **Referencias y ejemplos útiles:** Sí (2 puntos)
9. **Flexibilidad:** Sí (2 puntos)
10. **Reducción del esfuerzo cognitivo:** Sí (2 puntos)

$(2 + 1 + 2 + 2 + 2 + 2) = 11$  puntos

$(11 \text{ puntos} / 12 \text{ posibles}) * 10 = 9,17$  puntos de generación de código.

$(2 + 2 + 2 + 1 + 1 + 2 + 1 + 2 + 2 + 2) = 17$  puntos

$(17 \text{ puntos} / 20 \text{ posibles}) * 10 = 8,5$  puntos sobre 10 de soporte.

**Puntuación final:**  $(9,17 + 8,5) / 2 = 8,83$  puntos.

### 21 Asunto: Codificación de módulo de consultas simulado

#### Caso de estudio: (Prompting / Generación de código)

Se requiere realizar la simulación de un web service de la REE (red eléctrica española), ya que no se dispone de su certificado y necesitamos realizar pruebas de regresión de un determinado módulo del proyecto.

Para ello necesito codificar la consulta de estos ficheros que proporciona el web service, para que posteriormente en el módulo de descargas de la REE se puedan realizar las pruebas de regresión sin necesidad de poseer el certificado.

### **Prompt de Entrada:**

#### **Primer prompt (Técnica de “Consultor iniciado”):**

Como soy un iniciado en servicios para módulos de Java Spring, requiero que me codifiques una clase service de java Spring que realice los procesos de esta clase de tal manera que en vez de consultar el web service de REE, se consulte un directorio de resources donde hay un fichero json que simulará ese web service, de tal manera que se realicen todos los procesos vistos y tratados en esta clase simulando su comportamiento:

...Servicio consultaReeService.java...

(Clase de 325 líneas de código que consulta si existen nuevos ficheros a procesar del web service de REE y el cual hace uso de las colas SQS de AWS para la comunicación entre módulos.)

#### **Segundo prompt: (Aumento de contexto de la conversación)**

Proporcione un fichero json que será tratado como ejemplo de un fichero que recibimos por parte del web service de REE.

Codifícame la lectura de ese fichero json en el servicio simulado.

#### **Tercer prompt: (Aumento de contexto de la conversación)**

En base a este apartado de la clase que ya te he proporcionado antes, quiero que como aparece aquí, obtengas el tipo de fichero de la base de datos con esa variable autowired de tabla \_ejemplo, esta clase es la siguiente:

<inserto clase Java>

Y en este servicio se hace uso del método de listarDescargaFicherosOrigenTipoDescargaNull

¿Haría falta cambiar algo del servicio que ya me has proporcionado para cumplir lo que te acabo de exponer?

#### **Cuarto prompt: (Aumento de contexto de la conversación)**

Necesito que en base a este fichero json me codifiques la entidad en java con las anotaciones necesarias para que sea mapeada con Jackson.

...Fichero JSON...

### **Respuesta de MO4D:**

#### **Respuesta a primer prompt:**

Para simular el comportamiento de la clase `ConsultaReeService` y hacer que consulte un archivo JSON en lugar de un servicio web, puedes crear un nuevo servicio que lea el archivo JSON y realice los mismos procesos que la clase original. Aquí tienes un ejemplo de cómo podrías implementarlo:

### Paso 1: Crear una clase de modelo para el contenido del JSON

Primero, necesitas definir las clases de modelo que representarán la estructura de tu JSON. A continuación, se muestra un ejemplo básico. Asegúrate de adaptar los campos a la estructura real de tu JSON.

<Clase Java para mapeo del JSON>

### Paso 2: Crear un servicio que lea el archivo JSON

Luego, crea un nuevo servicio llamado `ConsultaReeMockService` que leerá los datos de un archivo JSON y simulará la lógica de la clase original.

<Clase Java del servicio simulado>

### Resumen

Con esta implementación, el servicio `ConsultaReeMockService` simula el comportamiento del servicio original, pero en lugar de consultar un servicio web, lee los datos de un archivo JSON. Asegúrate de adaptar los campos y la lógica según tus necesidades específicas y la estructura de tu archivo JSON.

### **Respuesta a tercer prompt:**

Para adaptar el servicio `ConsultaReeMockService` de manera que obtenga los tipos de fichero desde la base de datos usando el método `listarDescargaFicherosOrigenTipoDescargaNull` de `Tabla\_ejemplo`, no necesitas realizar una deserialización del JSON, sino simplemente leer el archivo desde el directorio de recursos y luego consultar la base de datos.

A continuación, te muestro cómo puedes modificar el servicio:

### Modificaciones en el Servicio

1. **Inyectar el `tabla\_ejemplo`.**
2. **Obtener los tipos de fichero desde la base de datos.**
3. **Leer el archivo JSON desde el directorio de recursos (sin procesar).**

Aquí tienes un ejemplo de cómo podría quedar el servicio adaptado:

<Servicio Java adaptado a los cambios>

### Detalles de la Implementación

1. **Lectura del Archivo JSON:**
  - El método `readJsonFile` lee el contenido del archivo JSON y lo devuelve como una cadena. No se deserializa, ya que mencionaste que no es necesario.
2. **Consulta a la Base de Datos:**

- Utiliza el método `listarDescargaFicherosOrigenTipoDescargaNull` del `tabla_ejemplo` para obtener la lista de tipos de fichero. Asegúrate de pasar el `idAgente` correspondiente (tendrás que definir cómo obtendrás este valor).

### 3. **Log de Resultados**:

- Se registra el contenido del archivo JSON y los tipos de fichero obtenidos para que puedas ver lo que se está procesando.

#### ### Notas Adicionales

- Asegúrate de que la propiedad `ruta.json.fichero` esté configurada correctamente en tu archivo `application.properties`.

- Si `idAgente` es necesario para realizar la consulta, tendrás que definir cómo obtendrás en tu lógica, ya sea a través de una configuración, un parámetro, o un servicio adicional.

- La implementación asume que no hay necesidad de procesar el contenido del JSON más allá de obtenerlo, pero puedes agregar lógica adicional según sea necesario.

MO4D va ajustando las respuestas según voy ampliando el contexto y el objetivo al que quiero llegar.

### **Respuesta al cuarto prompt:**

Si deseas utilizar Jackson junto con JPA (Java Persistence API) para mapear entidades a una base de datos, puedes combinar las características de ambas bibliotecas. A continuación, te muestro cómo hacerlo utilizando las anotaciones de JPA y Jackson en la misma clase.

Aquí tienes un ejemplo de cómo se vería una entidad Java que usa ambas bibliotecas:

<Entidad Java adaptada al mapeo del JSON>

#### ### Explicación

##### 1. **Anotaciones JPA**:

- `@Entity`: Marca la clase como una entidad JPA.

- `@Table`: Especifica el nombre de la tabla en la base de datos.

- `@Id`: Marca el campo como la clave primaria de la entidad.

- `@GeneratedValue`: Indica que la clave primaria debe ser generada automáticamente.

- `@Column`: Especifica el nombre de la columna en la tabla.

- `@OneToMany` y `@ManyToOne`: Representan las relaciones entre las entidades.

##### 2. **Anotaciones Jackson**:

- `@JsonProperty`: Se utiliza para mapear los nombres de los campos en JSON a los nombres de las propiedades en Java.

- `@JsonInclude`: Indica que las propiedades que son `null` no deben incluirse en la serialización.

#### ### Integración

Asegúrate de tener las siguientes dependencias en tu archivo `pom.xml` (si usas Maven):

```
``xml
```

```
<dependency>
```

```
  <groupId>com.fasterxml.jackson.core</groupId>
```

```
  <artifactId>jackson-databind</artifactId>
```

```

    <version>X</version> <!-- O la versión que necesites -->
</dependency>
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-data-jpa</artifactId>
</dependency>
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-web</artifactId>
</dependency>
<dependency>
    <groupId>com.h2database</groupId> <!-- O tu base de datos preferida -->
    <artifactId>h2</artifactId>
    <scope>runtime</scope>
</dependency>
...

```

### ### Conclusión

Este es un ejemplo básico de cómo utilizar Jackson junto con JPA en una entidad Java. Puedes adaptar los tipos de datos y las relaciones según las necesidades de tu aplicación.

### Respuesta de experto:

Tras unos pocos ajustes según la lógica que quiero darle, la solución obtenida es sin duda bastante efectiva en cuanto a lo que se buscaba en este caso de estudio, que era obtener de la IA una estructura de un servicio que simule la consulta a un web service sobre el que no disponemos de certificado para realizar las peticiones correspondientes.

Por otra parte, la entidad del fichero JSON no es del todo correcta según lo que buscaba así que he tenido que realizar algunos cambios manualmente en cuanto a una lista de objetos que no la mapeaba correctamente.

### Análisis tras resultados:

La solución final obtenida tras todos los reajustes de la conversación me ha ahorrado mucho tiempo ya que la tarea de adaptación de un servicio ya existente a una funcionalidad simulada sobre un web service es muy complejo y tedioso.

Sobre todo, destaco la codificación de las clases necesarias para la deserialización del fichero JSON de prueba, que de no disponer de la ayuda de la IA su codificación habría sido muy tediosa.

### Generación de código:

Criterio	Descripción	Evaluación Usando IA	Notas/Comentarios
<b>1.1 Quita tiempo de desarrollo</b>	Evalúa si la IA ayuda a acelerar el proceso de desarrollo.	[ ] No, [ ] Parcial, [X] Sí	La IA permitió ahorrar tiempo en la adaptación del servicio existente para simular el web service.

<b>1.2 Cohesión en el código</b>	Las partes de código generadas funcionan bien juntas, manteniendo el contexto.	<input type="checkbox"/> No, <input checked="" type="checkbox"/> Parcial, <input type="checkbox"/> Sí	Aunque la estructura general fue útil, se necesitaron ajustes manuales en la entidad del JSON.
<b>1.3 Facilita tareas tediosas</b>	Evalúa si la IA reduce la necesidad de realizar tareas manuales repetitivas.	<input type="checkbox"/> No, <input type="checkbox"/> Parcial, <input checked="" type="checkbox"/> Sí	La generación automática de las clases necesarias para la deserialización del JSON simplificó un proceso tedioso.
<b>2. Código seguro</b>	El código generado sigue buenas prácticas de seguridad.	<input type="checkbox"/> No, <input type="checkbox"/> Parcial, <input checked="" type="checkbox"/> Sí	
<b>3. Funcionalidad recibida</b>	La funcionalidad solicitada por la IA se cumple correctamente.	<input type="checkbox"/> No, <input checked="" type="checkbox"/> Parcial, <input type="checkbox"/> Sí	Hubo que realizar varios cambios en el JSON
<b>4. Código claro y entendible</b>	El código generado es legible, fácil de entender y mantener.	<input type="checkbox"/> No, <input type="checkbox"/> Parcial, <input checked="" type="checkbox"/> Sí	

Nota caso de estudio:

Generación de código:

1. **Quita tiempo de desarrollo:** Sí (2 puntos)
2. **Cohesión en el código:** Parcial (1 punto)
3. **Facilita tareas tediosas:** Sí (2 puntos)
4. **Código seguro:** Sí (2 puntos)
5. **Funcionalidad recibida:** Parcial (1 punto)
6. **Código claro y entendible:** Sí (2 puntos)

Suma de puntos:  $2 + 1 + 2 + 2 + 1 + 2 = 10$

Puntos posibles: 12

**Puntuación final:**  $(10 / 12) * 10 = 8,33$  puntos sobre 10

## 22 Asunto: Migración de Scripts (tablas y datos) de MySQL a PostgreSQL

### Caso de estudio: (Prompting / Soporte / Generación de código)

Se requiere realizar la migración de unos scripts SQL de creación de la base de datos del proyecto, además de la carga inicial de los datos para su funcionamiento y pruebas.

#### Prompt de Entrada:

##### Primer prompt: (Contextualización inicial)

Necesito que tú, como experto en gestión de bases de datos y más en específico de MySQL y PostgreSQL, me realices la migración de estas inserciones a una base de datos de MySQL para que pueda ser ejecutada en mi base de datos que es de PostgreSQL:

- La sintaxis debe ser 100% compatible con PostgreSQL
- Si encuentras algo que no te cuadra y debería saber sobre las inserciones que te paso, házmelo saber.
- Quiero que la carga de datos sea completa y la misma que se daría en el caso de la base de datos de MySQL.

El fichero de inserciones a adaptar a PostgreSQL es el siguiente:

<Inserciones de MySQL>

Debido a la longitud de las inserciones y el marco de tokens de entrada que permitía la herramienta en ese momento, tuve que separar en 2 el fichero.

### **Segundo prompt sobre error encontrado (Aumento de contexto de la conversación):**

En base a esta inserción:

```
INSERT INTO tm_ejemplo (id_ejemplo, cod_ ejemplo, des_ ejemplo, fecha_alta, fl_activo)
VALUES
```

```
(1, x, x x, '2024-07-01'::date, 1); -- Ejemplo, ajusta valores según tus datos
```

```
ON CONFLICT (id_ ejemplo) DO UPDATE SET (cod_ ejemplo, des_ ejemplo, fecha_alta,
fl_activo) = (excluded.cod_ ejemplo, excluded.des_ ejemplo, excluded.fecha_alta,
excluded.fl_activo);
```

me da este error:

```
ERROR: error de sintaxis en o cerca de «ON»
```

```
LINE 15: ON CONFLICT (id_ ejemplo) DO UPDATE SET (cod_ ejemplo,...
```

```
^
```

```
SQL state: 42601
```

```
Character: 1082
```

### **Tercer prompt (Aumento de contexto de la conversación):**

Realízame la adaptación de estas inserciones de datos:

<Inserciones de datos iniciales SQL>

### **Respuesta de MO4D:**

#### **Respuesta a primer prompt:**

<Script SQL adecuado a PostgreSQL>

En este caso presentaba una inconsistencia en una inserción específica que se trata en el segundo prompt.

### **Respuesta a segundo prompt:**

El error que estás viendo se debe a la sintaxis incorrecta en la cláusula ON CONFLICT DO UPDATE. En PostgreSQL, la cláusula SET no acepta una lista de columnas entre paréntesis. En su lugar, debes especificar cada columna y su valor de actualización por separado.

Aquí tienes la corrección:

```
INSERT INTO tm_ejemplo (id_ejemplo, cod_ejemplo, des_ejemplo, fecha_alta, fl_activo)
VALUES
```

```
(1, x, x x, '2024-07-01'::date, 1)
```

```
ON CONFLICT (id_ejemplo) DO UPDATE SET
```

```
cod_ejemplo = EXCLUDED.cod_ejemplo,
```

```
des_ejemplo = EXCLUDED.des_ejemplo,
```

```
fecha_alta = EXCLUDED.fecha_alta,
```

```
fl_activo = EXCLUDED.fl_activo;
```

Prueba con esta versión y debería funcionar sin problemas. Si encuentras algún otro problema o necesitas más ayuda, ¡no dudes en decírmelo!

### **Respuesta a tercer prompt:**

<Inserciones adaptadas a PostgreSQL>

### **Respuesta de experto:**

La codificación de las inserciones de las tablas y a su vez de los datos iniciales habría supuesto una carga de trabajo demasiado tediosa que de esta manera he visto increíblemente reducida mediante el uso de la IA.

Aun así, la primera respuesta obtenida no fue del todo perfecta ya que encontré un error en una inserción específica que trataba de una operación que la IA interpretó que se podía realizar en PostgreSQL, como es utilizar paréntesis para determinar varios campos en un SET UPDATE tras la condición ON CONFLICT. Esto no es aplicable en PostgreSQL y al indicárselo a la IA reconoció el error.

Por mi parte no tuve que codificar nada para solucionar el error dado, simplemente se lo señalé a la IA y esta me dio la respuesta al problema.

Por otro lado, en cuanto a la carga de datos iniciales, me devolvió la adaptación correcta en su totalidad, la única tarea que tuve que realizar fue la de adaptar las inserciones según su orden establecido debido a las claves foráneas de las tablas y sus correspondientes dependencias.

### Análisis tras resultados:

Este caso es curioso debido al error encontrado de la migración de la inserción que señalo en el segundo prompt, ya que, aunque contextualizo que la sintaxis debe ser 100% compatible con PostgreSQL, que es un experto en bases de datos, etc. Comete ese error de utilizar una sentencia no compatible en PostgreSQL.

No obstante, el tiempo de codificación se redujo significativamente, y la tarea se completó de manera mucho más rápida en comparación con el enfoque tradicional.

### Generación de código:

| <b>Criterio</b>                       | <b>Descripción</b>   | <b>Evaluación Usando IA</b>   | <b>Notas/Comentarios</b>   |
|---------------------------------------|--|---|--|
| <b>1.1 Quita tiempo de desarrollo</b> | Evalúa si la IA ayuda a acelerar el proceso de desarrollo.                     | <input type="checkbox"/> No<br><input type="checkbox"/> Parcial,<br><input checked="" type="checkbox"/> Sí  | Aunque la solución requirió algunos ajustes que llevaron tiempo, el cómputo global del tiempo ahorrado es tan grande que merece la pena el uso de la IA en estos casos.        |
| <b>1.2 Cohesión en el código</b>      | Las partes de código generadas funcionan bien juntas, manteniendo el contexto. | <input type="checkbox"/> No,<br><input checked="" type="checkbox"/> Parcial,<br><input type="checkbox"/> Sí |  |
| <b>1.3 Facilita tareas tediosas</b>   | Evalúa si la IA reduce la necesidad de realizar tareas manuales repetitivas.   | <input type="checkbox"/> No,<br><input type="checkbox"/> Parcial,<br><input checked="" type="checkbox"/> Sí | La codificación de la creación de tablas y carga inicial de datos de un motor de bases de datos a otros manualmente supone una gran carga de trabajo que se ha visto reducida. |
| <b>2. Código seguro</b>               | El código generado sigue buenas prácticas de seguridad.                        | <input type="checkbox"/> No,<br><input type="checkbox"/> Parcial,<br><input checked="" type="checkbox"/> Sí |  |
| <b>3. Funcionalidad recibida</b>      | La funcionalidad solicitada por la IA se cumple correctamente.                 | <input type="checkbox"/> No,<br><input type="checkbox"/> Parcial,<br><input checked="" type="checkbox"/> Sí |  |
| <b>4. Código claro y entendible</b>   | El código generado es legible, fácil de entender y mantener.                   | <input type="checkbox"/> No,<br><input type="checkbox"/> Parcial,<br><input checked="" type="checkbox"/> Sí |  |

**Soporte:**

| <b>Criterio</b>                          | <b>Descripción</b>  | <b>Evaluación Usando IA</b>   | <b>Notas/Comentarios</b>  |
|--|---|---|---|
| <b>1. Precisión en la asistencia</b>     | La IA ofrece respuestas correctas y comandos precisos (e.g., comandos de Docker, configuraciones).              | <input type="checkbox"/> No,<br><input type="checkbox"/> Parcial,<br><input checked="" type="checkbox"/> Sí |   |
| <b>2. Facilidad para debuggear</b>       | La IA guía de manera clara y efectiva en la identificación y solución de errores en el código, si es necesario. | <input type="checkbox"/> No,<br><input type="checkbox"/> Parcial,<br><input checked="" type="checkbox"/> Sí | Tras dar el aviso del error encontrado supo darme la solución correcta. |
| <b>3. Soluciones contextualizadas</b>    | La IA adapta sus respuestas al contexto y las necesidades específicas del problema planteado.                   | <input type="checkbox"/> No,<br><input type="checkbox"/> Parcial,<br><input checked="" type="checkbox"/> Sí |   |
| <b>4. Proactividad</b>                   | La IA propone mejoras, optimizaciones o pasos adicionales útiles sin necesidad de ser solicitada.               | <input type="checkbox"/> No,<br><input checked="" type="checkbox"/> Parcial,<br><input type="checkbox"/> Sí |   |
| <b>5. Tiempo de respuesta</b>            | La IA responde rápidamente, reduciendo el tiempo necesario para avanzar en las tareas.                          | <input type="checkbox"/> No,<br><input type="checkbox"/> Parcial,<br><input checked="" type="checkbox"/> Sí |   |
| <b>6. Claridad en las explicaciones</b>  | Las explicaciones y guías ofrecidas son claras y fáciles de entender.   | <input type="checkbox"/> No,<br><input checked="" type="checkbox"/> Parcial,<br><input type="checkbox"/> Sí |   |
| <b>7. Consistencia en las respuestas</b> | Las respuestas de la IA son consistentes y no contradictorias entre sí.   | <input type="checkbox"/> No,<br><input checked="" type="checkbox"/> Parcial,<br><input type="checkbox"/> Sí | Hubo necesidad de correcciones que indicó una falta de consistencia     |
| <b>8. Referencias y ejemplos útiles</b>  | La IA proporciona ejemplos relevantes y enlaces útiles para ampliar el entendimiento.                           | <input type="checkbox"/> No,<br><input checked="" type="checkbox"/> Parcial,<br><input type="checkbox"/> Sí |   |
| <b>9. Flexibilidad</b>                   | La IA se adapta bien a diferentes lenguajes de programación,  | <input type="checkbox"/> No,  |   |

|   |  |   |  |
|---|--|---|--|
|   | frameworks, y herramientas.  | <input type="checkbox"/> Parcial,<br><input checked="" type="checkbox"/> Sí                                 |  |
| <b>10. Reducción del esfuerzo cognitivo</b> | La IA simplifica tareas complejas, haciéndolas más manejables para el desarrollador. | <input type="checkbox"/> No,<br><input type="checkbox"/> Parcial,<br><input checked="" type="checkbox"/> Sí |  |

Generación de código:

1. **Quita tiempo de desarrollo:** Sí (2 puntos)
2. **Cohesión en el código:** Parcial (1 punto)
3. **Facilita tareas tediosas:** Sí (2 puntos)
4. **Código seguro:** Sí (2 puntos)
5. **Funcionalidad recibida:** Sí (2 puntos)
6. **Código claro y entendible:** Sí (2 puntos)

Soporte:

1. **Precisión en la asistencia:** Sí (2 puntos)
2. **Facilidad para debuggear:** Sí (2 puntos)
3. **Soluciones contextualizadas:** Sí (2 puntos)
4. **Proactividad:** Parcial (1 punto)
5. **Tiempo de respuesta:** Sí (2 puntos)
6. **Claridad en las explicaciones:** Parcial (1 punto)
7. **Consistencia en las respuestas:** Parcial (1 punto)
8. **Referencias y ejemplos útiles:** Parcial (1 punto)
9. **Flexibilidad:** Sí (2 puntos)
10. **Reducción del esfuerzo cognitivo:** Sí (2 puntos)

Total de puntos:

$(2 + 1 + 2 + 2 + 2 + 2) = 11$  puntos

$(11 \text{ puntos} / 12 \text{ posibles}) * 10 = 9,17$  puntos de generación de código.

$(2 + 2 + 2 + 1 + 2 + 1 + 1 + 1 + 2 + 2) = 16$  puntos

$(16 \text{ puntos} / 20 \text{ posibles}) * 10 = 8$  puntos de soporte.

**Puntuación final:**  $(9,17 + 8) / 2 = 8,58$  puntos.

## 23 Asunto: Asignación de Driver Oracle a conexión a BBDD

### Caso de estudio: (Soporte)

Se solicita a la IA que ayude a encontrar el posible error que encontramos al intentar cargar el driver del JDBC Oracle.

### Prompt de Entrada:

#### Primer prompt (Contextualización):

Al ser un prototipo de un desarrollo en sus inicios, pretendemos realizar una conexión inicial simple a la base de datos de Oracle, la cual es de la versión 12.

El prompt inicial para contextualizar a la IA es el siguiente:

“Estamos intentando realizar una conexión a una base de datos de Oracle mediante esta configuración presente que tenemos en el código Java: <Código Java>

Al intentar asignar el driver nos da esta excepción:

```
java.lang.ClassNotFoundException: oracle.jdbc.driver.OracleDriver      at
java.net.URLClassLoader.findClass(URLClassLoader.java:381)           at
java.lang.ClassLoader.loadClass(ClassLoader.java:424)                at
sun.misc.LauncherAppClassLoader.loadClass(Launcher.java:349)atjava.lang.ClassLoad
er.loadClass(ClassLoader.java:357)...
```

### **Segundo prompt (Contextualización):**

Como tenemos una versión de Oracle 12 y la IA no tiene constancia hasta el momento de las dependencias presentes en el pom, contextualizo proporcionandose lo:

“

```
tengo          esta          dependencia          en          el
pom: <dependency> <groupId>com.oracle.database.jdbc</groupId> <artifactId>ojdbc7
</artifactId> <version>12.1.0.2</version> <scope>runtime</scope> </dependency>
```

”

### **Respuesta de MO4D:**

#### **Respuesta a primer prompt:**

Indica que el problema puede estar relacionado con que no se encuentra el jar referente a la dependencia de Oracle.

#### **Respuesta a segundo prompt:**

Indica que la dependencia es correcta y que debería funcionar, a lo que responde con otro tipo de implementación.

### **Respuesta de experto:**

La segunda respuesta sobre la dependencia del POM que indicamos es incorrecta. La dependencia del POM no se encontraba en el artefacto de esa versión, por lo que busqué en el Maven Repository para consultar la disponibilidad completa de esta librería, específicamente de la versión 12. Como resultado, encontré que la versión 12 disponible era la 12.2.0.1 y no la que previamente nos indicó la IA (12.1.0.2).

Por otra parte, al reinstalar las dependencias con “mvn install” desde Eclipse, la versión del Java Compiler era inferior a la necesaria, ya que el código contenía funciones lambda

no soportadas por la versión presente. Sin los conocimientos adecuados, esto habría resultado en una tarea compleja para un inexperto en ingeniería informática.

### Análisis tras resultados:

Para investigar más sobre este caso, después de haber solucionado el problema, pregunté a la IA qué versiones de esta librería estaban disponibles. Utilicé el LLM de Claude disponible en MO4D, que me respondió con un gran número de versiones, incluyendo la no disponible 12.1.0.2. Esto sugiere que, cuando se realizó el entrenamiento de este modelo, la versión errónea estaba disponible, pero ya no lo está en la actualidad. Esto pone de manifiesto la importancia del sentido crítico y la búsqueda manual cuando la situación y tus conocimientos lo requieren.

Soporte

| <b>Criterio</b>                       | <b>Descripción</b>  | <b>Evaluación Usando IA</b>   | <b>Notas/Comentarios</b>  |
|---------------------------------------|---|---|---|
| <b>1. Precisión en la asistencia</b>  | La IA ofrece respuestas correctas y comandos precisos (e.g., comandos de Docker, configuraciones).              | <input type="checkbox"/> No,<br><input checked="" type="checkbox"/> Parcial,<br><input type="checkbox"/> Sí | El error en las versiones de la dependencia supone que no ha asistido correctamente en su tarea del todo, aunque acertó en que la excepción era referente a la falta del jar. |
| <b>2. Facilidad para debuggear</b>    | La IA guía de manera clara y efectiva en la identificación y solución de errores en el código, si es necesario. | <input type="checkbox"/> No,<br><input checked="" type="checkbox"/> Parcial,<br><input type="checkbox"/> Sí | Proporcionó una guía inicial útil, pero no resolvió completamente el problema.  |
| <b>3. Soluciones contextualizadas</b> | La IA adapta sus respuestas al contexto y las necesidades específicas del problema planteado.                   | <input type="checkbox"/> No,<br><input type="checkbox"/> Parcial,<br><input checked="" type="checkbox"/> Sí |   |
| <b>4. Proactividad</b>                | La IA propone mejoras, optimizaciones o pasos adicionales útiles sin necesidad de ser solicitada.               | <input type="checkbox"/> No,<br><input checked="" type="checkbox"/> Parcial,<br><input type="checkbox"/> Sí |   |

|   |  |   |   |
|---|--|---|---|
| <b>5. Tiempo de respuesta</b>               | La IA responde rápidamente, reduciendo el tiempo necesario para avanzar en las tareas.   | <input type="checkbox"/> No,<br><input type="checkbox"/> Parcial,<br><input checked="" type="checkbox"/> Sí |   |
| <b>6. Claridad en las explicaciones</b>     | Las explicaciones y guías ofrecidas son claras y fáciles de entender.                    | <input checked="" type="checkbox"/> No,<br><input type="checkbox"/> Parcial,<br><input type="checkbox"/> Sí | Las explicaciones no fueron suficientemente claras, lo que llevó a confusión sobre la versión correcta de la dependencia.   |
| <b>7. Consistencia en las respuestas</b>    | Las respuestas de la IA son consistentes y no contradictorias entre sí.                  | <input type="checkbox"/> No,<br><input checked="" type="checkbox"/> Parcial,<br><input type="checkbox"/> Sí | Las respuestas fueron coherentes en su mayoría, pero la necesidad de correcciones indicó una falta de consistencia inicial. |
| <b>8. Referencias y ejemplos útiles</b>     | La IA proporciona ejemplos relevantes y enlaces útiles para ampliar el entendimiento.    | <input type="checkbox"/> No,<br><input type="checkbox"/> Parcial,<br><input checked="" type="checkbox"/> Sí |   |
| <b>9. Flexibilidad</b>                      | La IA se adapta bien a diferentes lenguajes de programación, frameworks, y herramientas. | <input type="checkbox"/> No,<br><input type="checkbox"/> Parcial,<br><input checked="" type="checkbox"/> Sí |   |
| <b>10. Reducción del esfuerzo cognitivo</b> | La IA simplifica tareas complejas, haciéndolas más manejables para el desarrollador.     | <input checked="" type="checkbox"/> No,<br><input type="checkbox"/> Parcial,<br><input type="checkbox"/> Sí | Sin mi experiencia en entornos Java y resolución de dependencias con el POM este caso habría sido más complejo              |

Nota caso de estudio:

1. **Precisión en la asistencia:** Parcial (1 punto)
2. **Facilidad para debuggear:** Parcial (1 punto)
3. **Soluciones contextualizadas:** Sí (2 puntos)
4. **Proactividad:** Parcial (1 punto)
5. **Tiempo de respuesta:** Sí (2 puntos)
6. **Claridad en las explicaciones:** No (0 puntos)

7. **Consistencia en las respuestas:** Parcial (1 punto)
8. **Referencias y ejemplos útiles:** Sí (2 puntos)
9. **Flexibilidad:** Sí (2 puntos)
10. **Reducción del esfuerzo cognitivo:** No (0 puntos)

**Suma de puntos:**  $1 + 1 + 2 + 1 + 2 + 0 + 1 + 2 + 2 + 0 = 12$  puntos

**Puntos posibles:** 20

**Puntuación final:**  $(12 / 20) * 10 = 6$  puntos sobre 10

## **Anexo IV. Cuestiones de los formularios**

1. Nombre del proyecto o proyectos involucrados.
2. Persona que contesta el formulario.
3. ¿Cuál es tu rol principal en el desarrollo software?
4. ¿Cuántos años de experiencia tienes en el desarrollo software?
5. ¿Qué lenguajes o tecnologías utilizas con mayor frecuencia?
6. ¿Qué herramientas de IA generativa has utilizado en el desarrollo software?
7. ¿Qué tipos de tareas has realizado principalmente con las herramientas de IA generativa?
8. ¿En qué entornos o herramientas integraste la IA generativa?
9. La herramienta generó código funcional en la mayoría de los casos.
10. El código generado era fácil de entender y mantener.
11. Las respuestas generadas cumplían la función principal que se le solicitó a la IA.
12. El código generado era de calidad.
13. Las respuestas obtenidas te ayudaron a abordar la tarea a cumplimentar.
14. ¿Has encontrado fallos de seguridad en alguna respuesta obtenida por la IA generativa?
15. Si encontraste algún fallo de seguridad, ¿Podrías determinar algunos de ellos?
16. En qué tareas o ámbitos te ha resultado más útil en cuanto a rendimiento la IA generativa.
17. ¿Qué tan fácil fue integrar estas herramientas en tu flujo de trabajo?
18. ¿Requeriste capacitación o documentación para usar estas herramientas?
19. ¿En qué aspectos has observado que estas herramientas son fáciles de usar?
20. Si por el contrario la usabilidad de las herramientas de IA generativa que has utilizado te han resultado difíciles de usar, indica en qué aspectos destacarías este razonamiento.
21. ¿Has notado alguna dependencia excesiva en la herramienta en tus procesos de desarrollo?
22. Si notaste alguna dependencia, explica en qué ámbito/tareas.
23. ¿Qué limitaciones encontraste al usar la herramienta de IA generativa?
24. ¿Qué mejoras agregarías a esta herramienta o herramientas de IA generativa que has utilizado?
25. ¿Recomendarías estas herramientas a otros desarrolladores?
26. En una escala del 1 al 10, ¿Cómo calificarías tu experiencia general con estas herramientas?
27. Comentarios adicionales.