Avance 2
Reporte de Depuración y Evaluación de Componentes de Software

David Armando Ramírez Navarrete

14 de diciembre de 2025

Resumen

El presente informe técnico documenta el proceso de análisis y preparación orientado a la depuración y evaluación de código, desarrollado como parte de las actividades iniciales de las prácticas profesionales en un entorno de trabajo basado en notebooks de Python. El objetivo de esta actividad fue establecer las bases técnicas y metodológicas necesarias para el desarrollo, revisión y mejora de soluciones computacionales enfocadas en el análisis de información.

Durante esta etapa se realizó una revisión estructurada de documentación técnica y lineamientos de buenas prácticas, con el propósito de identificar criterios relevantes para la evaluación de la lógica de programación, la correcta estructuración del código, la validación de datos y el manejo de errores. Este análisis permitió anticipar posibles áreas de riesgo y definir estrategias de depuración que faciliten la detección temprana de fallos durante la implementación.

Las actividades realizadas evidencian el desarrollo de competencias relacionadas con el análisis crítico de información técnica, la planeación del proceso de depuración y la preparación de entornos de desarrollo que favorecen la calidad, confiabilidad y mantenibilidad del código en contextos profesionales.

Introducción

Las prácticas profesionales representan una etapa formativa clave para la aplicación de los conocimientos adquiridos durante la formación académica, al permitir la integración de

teoría y práctica en un entorno real de trabajo. En este contexto, las actividades iniciales de la estadía profesional se orientaron al análisis técnico y a la revisión de información especializada, con el propósito de comprender los procesos, flujos de información y criterios técnicos involucrados en el desarrollo y evaluación de soluciones computacionales.

El presente informe tiene como objetivo documentar el proceso de análisis previo a la depuración y evaluación de código, enfocado en la comprensión de lineamientos técnicos, buenas prácticas de programación y criterios de calidad del software. Este análisis permitió identificar aspectos relevantes relacionados con la estructura del código, la lógica de procesamiento, la validación de datos y el manejo de errores, los cuales son fundamentales para garantizar la confiabilidad y mantenibilidad de los sistemas internos.

Asimismo, el trabajo realizado se complementa con la planeación del desarrollo de ejercicios prácticos en un entorno basado en notebooks de Python, los cuales serán utilizados como herramienta para implementar, evaluar y depurar soluciones orientadas al análisis de información. De esta manera, el informe evidencia un enfoque metodológico que articula el análisis teórico con la aplicación práctica, fortaleciendo competencias técnicas y profesionales propias de la Ingeniería en Sistemas y alineadas con los objetivos de las prácticas profesionales.

Descripción de la actividad

La actividad desarrollada se enmarca en las primeras etapas de las prácticas profesionales y consistió en la revisión técnica y el análisis conceptual de componentes de software orientados al procesamiento y análisis de información. Dicho análisis tuvo como finalidad comprender la estructura general de los sistemas internos, así como los flujos lógicos que intervienen en la manipulación de datos, previo a la implementación y depuración de código.

La revisión se realizó sobre componentes de carácter analítico y funcional, enfocándose en la identificación de procesos clave, entradas y salidas de información, así como en la forma en que se organizan y estructuran las operaciones dentro de un entorno de desarrollo. Esta etapa no implicó la modificación directa de sistemas productivos ni el uso de información sensible, sino un análisis abstracto y metodológico de los elementos que intervienen en la ejecución del código.

Como parte de la actividad, se llevó a cabo la planeación del desarrollo de ejercicios prácticos en un entorno basado en notebooks de Python, los cuales permiten ejecutar, evaluar y depurar fragmentos de código de manera controlada. Este enfoque facilitó la preparación para actividades posteriores de depuración, al establecer criterios claros para la evaluación de la lógica de programación, la validación de datos, el manejo de errores y la aplicación de buenas prácticas de desarrollo de software.

En conjunto, la actividad permitió sentar las bases técnicas necesarias para abordar de manera estructurada la depuración y evaluación de código, fortaleciendo la comprensión de los sistemas analizados y favoreciendo un enfoque preventivo orientado a la calidad y confiabilidad de las soluciones computacionales.

Metodología de revisión

La metodología de revisión aplicada durante esta actividad se basó en un enfoque analítico y sistemático orientado a la depuración y evaluación de código, con el propósito de identificar áreas de mejora y asegurar la calidad de las soluciones desarrolladas. Dicha metodología se implementó en un entorno de trabajo controlado mediante notebooks de Python, lo que permitió analizar de manera progresiva la ejecución del código y su comportamiento lógico.

Para la evaluación del código se establecieron los siguientes criterios técnicos:

Estructura

Se analizó la organización general del código, considerando la correcta separación de bloques funcionales, la secuencia lógica de las operaciones y el uso adecuado de celdas dentro del notebook. Este criterio permitió evaluar la modularidad del código y su facilidad de mantenimiento, así como la claridad en la disposición de los procesos.

Validaciones

Se revisó la existencia y pertinencia de mecanismos de validación de datos de entrada, verificando que los valores procesados cumplieran con los requisitos esperados en cuanto a tipo, formato y rango. La aplicación de validaciones adecuadas contribuye a prevenir errores de ejecución y a garantizar la integridad de la información analizada.

Manejo de errores

Se evaluó la forma en que el código contempla y gestiona posibles errores durante su ejecución, mediante el uso de estructuras de control que permitan identificar, manejar y comunicar fallos de manera clara. Un manejo adecuado de errores facilita la depuración y mejora la robustez del código.

Legibilidad

Se examinó la claridad del código desde el punto de vista de su comprensión, considerando aspectos como el uso de nombres descriptivos, la coherencia en la indentación, la inclusión de comentarios explicativos y el orden lógico de las instrucciones. La legibilidad es un factor clave para el trabajo colaborativo y el mantenimiento futuro del software.

Lógica general

Finalmente, se evaluó la coherencia de la lógica de programación, verificando que los flujos de control, condiciones y operaciones implementadas respondieran adecuadamente a los objetivos planteados. Este criterio permitió detectar inconsistencias lógicas, redundancias o procesos innecesarios que pudieran afectar el desempeño o la claridad del código.

En conjunto, la aplicación de estos criterios permitió llevar a cabo una revisión integral del código, orientada no solo a la detección de errores, sino también a la mejora continua y al fortalecimiento de buenas prácticas de desarrollo en un contexto profesional.

Hallazgos

Como resultado de la aplicación de la metodología de revisión, se identificaron diversos hallazgos relacionados con la estructura, lógica y calidad general del código analizado. Dichos hallazgos se describen a continuación de manera general, con el propósito de evidenciar áreas de oportunidad para la mejora continua del desarrollo de software.

- **Estructura del código:** Se detectaron oportunidades de mejora en la organización de los bloques funcionales, particularmente en la secuencia de ejecución de algunas celdas del notebook, lo que podría generar confusión durante el seguimiento del flujo lógico del proceso.
- **Validaciones de datos:** Se identificó la ausencia o implementación limitada de validaciones en ciertos puntos del procesamiento de datos, lo que podría permitir el manejo de valores inesperados o inconsistentes durante la ejecución del código.
- **Manejo de errores:** Se observó un manejo básico de errores, sin contemplar escenarios alternativos o mensajes descriptivos que faciliten la identificación de fallos durante la ejecución, lo cual dificulta el proceso de depuración.

- **Legibilidad:** En algunos segmentos del código se encontró falta de claridad en la nomenclatura de variables y en la documentación interna, lo que puede afectar la comprensión del código por parte de otros desarrolladores o en revisiones posteriores.
- **Lógica general:** Se detectaron procesos redundantes y flujos condicionales que podrían simplificarse, con el fin de optimizar la ejecución y mejorar la claridad del razonamiento lógico implementado.

En conjunto, los hallazgos identificados no representan fallos críticos, pero sí áreas de mejora que, al ser atendidas, contribuirán a fortalecer la calidad, mantenibilidad y confiabilidad del código desarrollado.

Acciones realizadas o sugeridas

Con base en los hallazgos identificados durante el proceso de revisión, se llevaron a cabo y se propusieron diversas acciones orientadas a mejorar la calidad, claridad y robustez del código analizado. Dichas acciones se plantearon con un enfoque preventivo y de mejora continua, considerando las buenas prácticas de desarrollo de software.

- **Mejora en la estructura del código:** Se reorganizaron o se propuso reorganizar los bloques funcionales del notebook, estableciendo una secuencia lógica clara que facilite la lectura, ejecución y mantenimiento del código.
- **Implementación de validaciones:** Se recomendó incorporar validaciones adicionales para verificar la consistencia, formato y tipo de los datos de entrada, con el objetivo de reducir el riesgo de errores durante el procesamiento de la información.
- **Fortalecimiento del manejo de errores:** Se sugirió la inclusión de mecanismos de control de errores más robustos, que permitan identificar y comunicar de forma clara posibles fallos durante la ejecución, facilitando así el proceso de depuración.
- **Mejoras en la legibilidad:** Se promovió el uso de nomenclatura descriptiva para variables y funciones, así como la incorporación de comentarios explicativos que documenten el propósito de cada bloque de código.
- **Optimización de la lógica general:** Se propuso simplificar flujos condicionales y eliminar procesos redundantes, con el fin de mejorar la eficiencia y claridad del razonamiento lógico implementado.

Estas acciones contribuyen a fortalecer la calidad del código y a establecer una base sólida para actividades posteriores de evaluación, depuración y desarrollo de soluciones computacionales en un entorno profesional.

Resultados

A partir de la revisión y evaluación del código realizada en un entorno controlado mediante notebooks de Python, se obtuvieron beneficios asociados a la mejora de la calidad y confiabilidad de las soluciones analizadas. Como evidencia del proceso, se incorporaron capturas representativas del flujo de trabajo, incluyendo la preparación de datos de prueba, la aplicación de validaciones, el manejo de errores, la identificación de inconsistencias y la verificación posterior a las correcciones implementadas.

Como parte del proceso de revisión, se realizaron ajustes en la documentación del notebook mediante la corrección de celdas Markdown, con el objetivo de mejorar la claridad, coherencia y comprensión del flujo de trabajo previo a la ejecución y depuración del código.

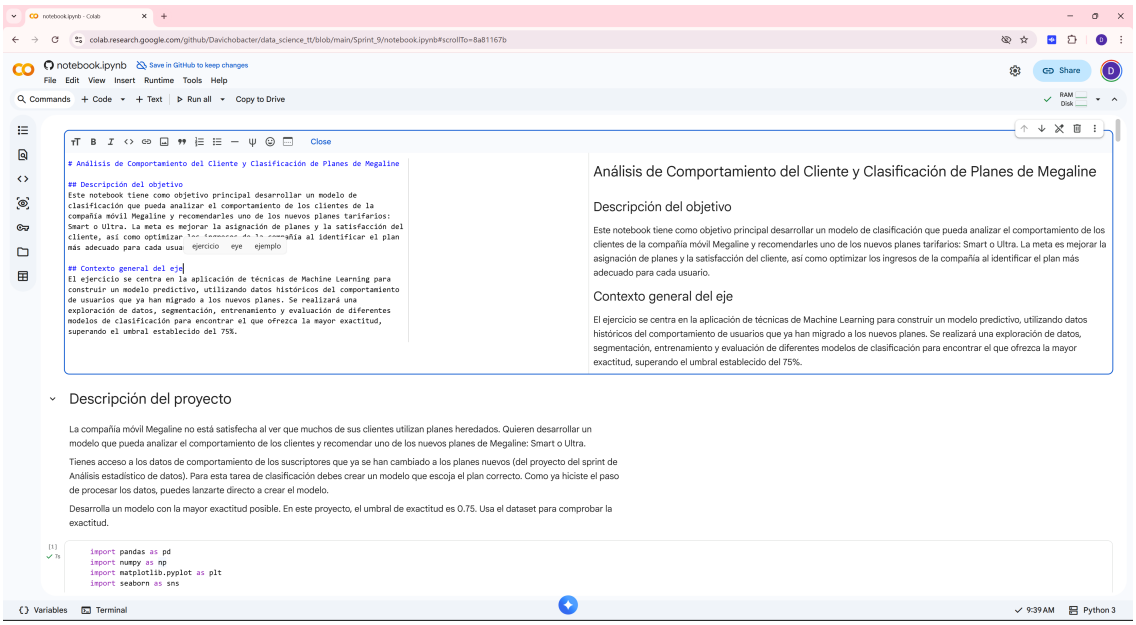

Figura 1: Ajuste de celdas Markdown para mejorar la estructura documental y la comprensión del flujo lógico previo a la depuración del código.

Posteriormente, se estructuró un bloque de código orientado a la generación de visualizaciones de distribución mediante una cuadrícula de gráficos, con el objetivo de facilitar el análisis comparativo de múltiples variables numéricas. Esta estructura permitió estandarizar la visualización, mejorar la legibilidad del código y optimizar el proceso de análisis

exploratorio de los datos.

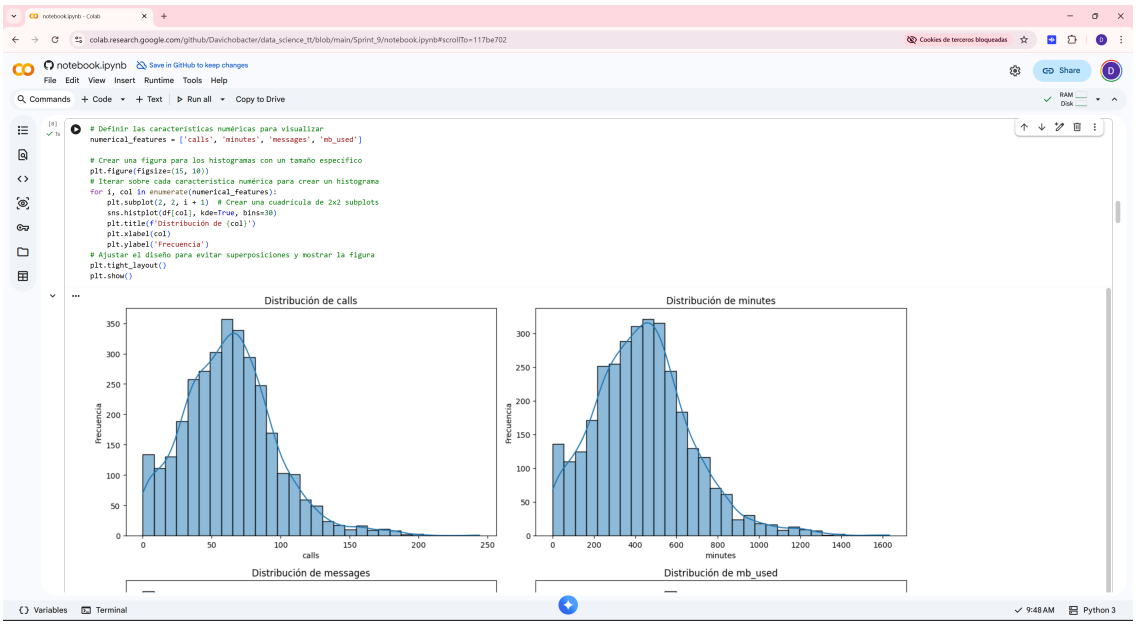

Figura 2: Implementación de un bloque reutilizable para la visualización de distribuciones en formato de cuadrícula, facilitando el análisis comparativo de variables numéricas.

Adicionalmente, se cuidó la implementación de los subplots respetando lineamientos de buenas prácticas de programación, como los establecidos en la guía PEP-8. Esto permitió mantener un código legible, ordenado y fácil de mantener, facilitando la comprensión del flujo lógico durante el análisis y la depuración.

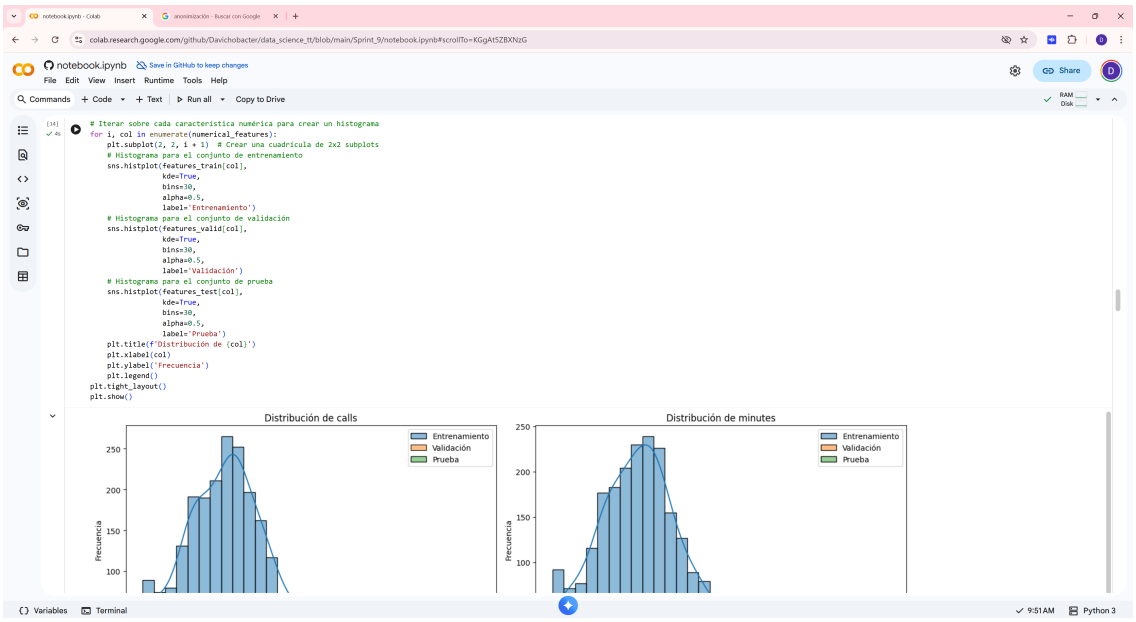

Figura 3: Estructuración de subplots para visualización de distribuciones, implementada conforme a buenas prácticas de codificación (PEP-8).

Durante la revisión del código se detectaron incumplimientos a los lineamientos de buenas prácticas establecidos en la guía PEP-8, principalmente relacionados con la longitud de las líneas y la concentración de múltiples instrucciones en un mismo bloque. A partir de esta detección, se procedió a reorganizar y ajustar el código para mejorar su legibilidad y mantenibilidad.

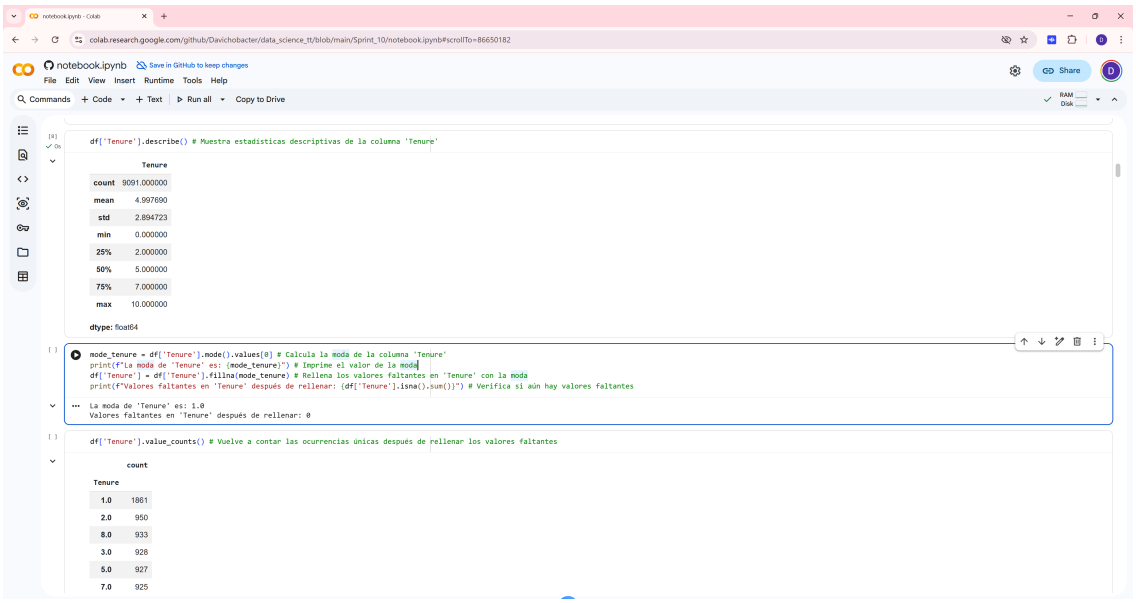

Figura 4: Detección de incumplimientos a los lineamientos PEP-8 durante la revisión del código, previo a la aplicación de correcciones.

Como parte de las acciones correctivas, se propuso y aplicó el uso de f-strings multilínea como alternativa para mejorar la legibilidad del código y cumplir con los lineamientos de PEP-8. Esta solución permitió estructurar mensajes extensos de forma clara, evitando líneas demasiado largas y facilitando la comprensión del código sin afectar su funcionalidad.

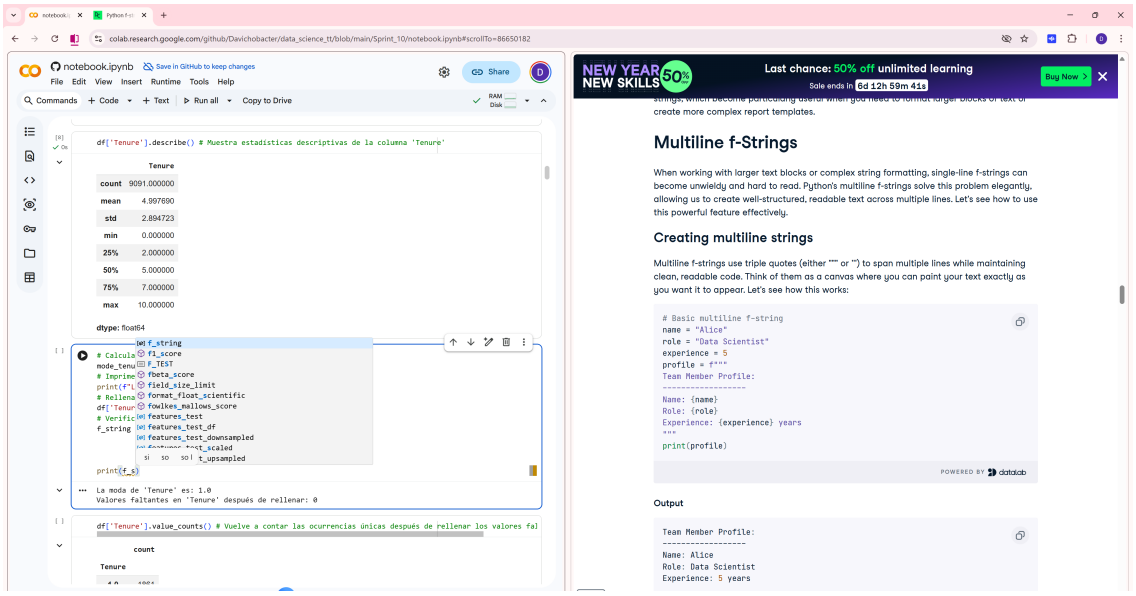

Figura 5: Aplicación de f-strings multilínea como solución para mejorar la legibilidad del código y cumplir con los lineamientos PEP-8.

Como parte del proceso de depuración, se identificó un error en tiempo de ejecución durante la carga de datos desde una URL externa. En respuesta, se investigaron prácticas recomendadas para el manejo de excepciones en Python y se incorporó un mecanismo de control mediante `try-except`, permitiendo gestionar errores de forma segura y prevenir fallos posteriores en la ejecución del notebook.

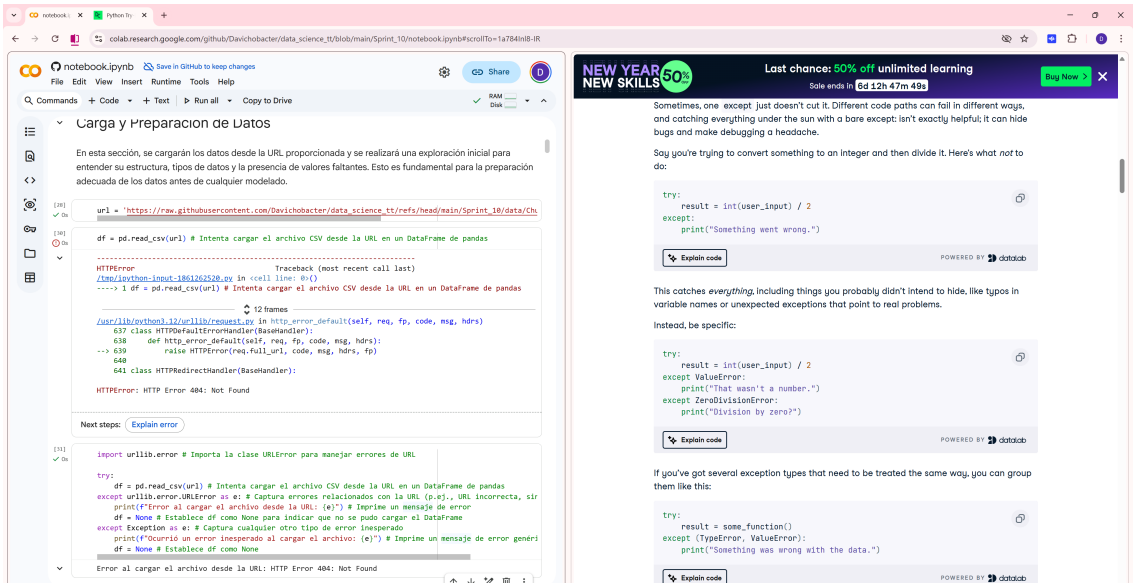

Figura 6: Aplicación de un bloque `try-except` al inicio del notebook para gestionar errores de carga de datos y mejorar la robustez del flujo de ejecución.

La revisión permitió mejorar la claridad estructural del código y fortalecer controles preventivos mediante validaciones, reduciendo el riesgo de fallos durante la ejecución. Asimismo, se observaron mejoras en la robustez del manejo de errores y en la trazabilidad del proceso de depuración, al documentar de forma verificable los escenarios de fallo y su corrección. En conjunto, los resultados respaldan un enfoque de mejora continua orientado a la mantenibilidad y estabilidad del código.

Finalmente, los resultados obtenidos durante el proceso de revisión y depuración fueron documentados en un reporte técnico elaborado en LaTeX, integrando de forma estructurada las evidencias gráficas, los hallazgos identificados y las conclusiones derivadas del análisis. Esta etapa permitió formalizar el trabajo realizado y asegurar una presentación clara, coherente y alineada con criterios académicos.

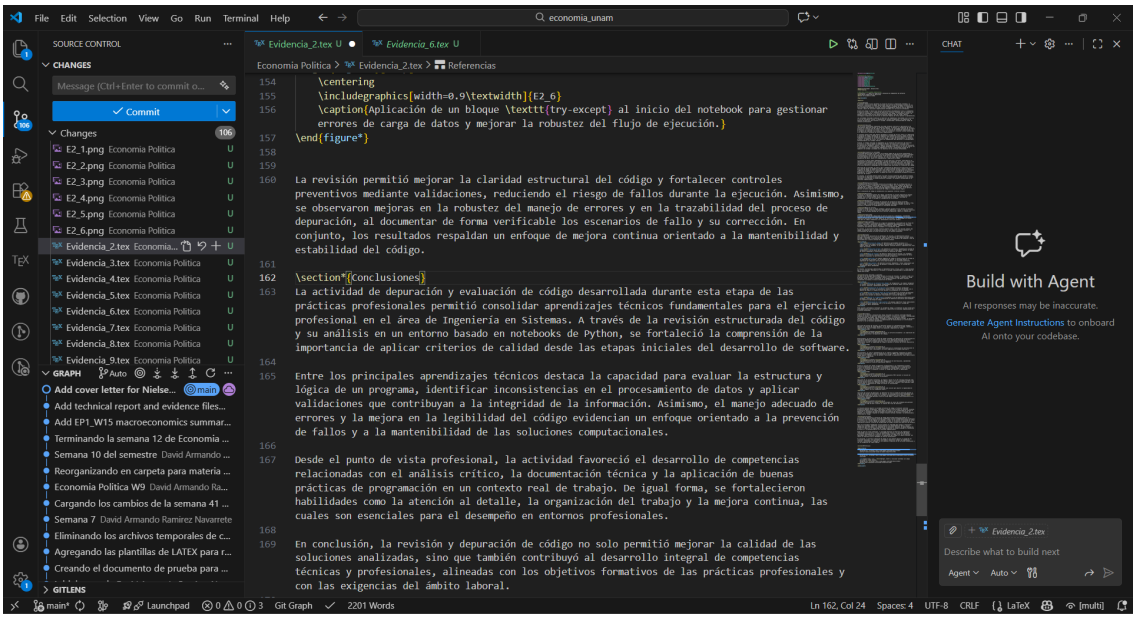

Figura 7: Formalización de los resultados mediante la elaboración del reporte técnico en LaTeX, asegurando una presentación estructurada y académica.

Conclusiones

La actividad de depuración y evaluación de código desarrollada durante esta etapa de las prácticas profesionales permitió consolidar aprendizajes técnicos fundamentales para el ejercicio profesional en el área de Ingeniería en Sistemas. A través de la revisión estructurada del código y su análisis en un entorno basado en notebooks de Python, se fortaleció la comprensión de la importancia de aplicar criterios de calidad desde las etapas iniciales del desarrollo de software.

Entre los principales aprendizajes técnicos destaca la capacidad para evaluar la estructura y lógica de un programa, identificar inconsistencias en el procesamiento de datos y aplicar validaciones que contribuyan a la integridad de la información. Asimismo, el manejo adecuado de errores y la mejora en la legibilidad del código evidencian un enfoque orientado a la prevención de fallos y a la mantenibilidad de las soluciones computacionales.

Desde el punto de vista profesional, la actividad favoreció el desarrollo de competencias relacionadas con el análisis crítico, la documentación técnica y la aplicación de buenas prácticas de programación en un contexto real de trabajo. De igual forma, se fortalecieron habilidades como la atención al detalle, la organización del trabajo y la mejora continua, las cuales son esenciales para el desempeño en entornos profesionales.

En conclusión, la revisión y depuración de código no solo permitió mejorar la calidad de

las soluciones analizadas, sino que también contribuyó al desarrollo integral de competencias técnicas y profesionales, alineadas con los objetivos formativos de las prácticas profesionales y con las exigencias del ámbito laboral.

Referencias

- DataCamp. (n.d.). *Python try-except: Exception handling explained*. Recuperado de: `https://www.datacamp.com/tutorial/python-try-except`
- DataCamp. (n.d.). *Python f-strings: Everything you need to know*. Recuperado de: `https://www.datacamp.com/tutorial/python-f-string`
- Google Cloud. (n.d.). *Cómo crear campos calculados en Looker Studio*. Recuperado de: `https://docs.cloud.google.com/looker/docs/studio/tutorial-create-calculated-fields-hl=es-419`
- Google Cloud. (n.d.). *Variables de resultado de consulta en Looker Studio*. Recuperado de: `https://docs.cloud.google.com/looker/docs/studio/query-result-variables?hl=es-419`
- Google Cloud. (n.d.). *Agregar, editar y solucionar problemas con campos calculados en Looker Studio*. Recuperado de: `https://docs.cloud.google.com/looker/docs/studio/add-edit-and-troubleshoot-calculated-fields?hl=es-419`
- Python Software Foundation. (n.d.). *Built-in exceptions*. Recuperado de: `https://docs.python.org/3/library/exceptions.html`
- Python Software Foundation. (n.d.). *PEP 8 – Style Guide for Python Code*. Recuperado de: `https://peps.python.org/pep-0008/`
- Real Python. (n.d.). *Python Exceptions: An Introduction*. Recuperado de: `https://realpython.com/python-exceptions/`