Universidad Politécnica
de Madrid

Escuela Técnica Superior de
Ingenieros Informáticos

Grado en Ingeniería Informática

Trabajo Fin de Grado

Desarrollo de una Herramienta de Validación de Integridad de Datos entre Hojas de Cálculo

Autor: José Mario Rodríguez-Salinas Bu

Tutor: Alejandro Rodríguez González

Co-tutora: Lucía Prieto Santamaría

Madrid, Enero 2026

Este Trabajo Fin de Grado se ha depositado en la ETSI Informáticos de la Universidad Politécnica de Madrid para su defensa.

Trabajo Fin de Grado
Grado en Ingeniería Informática
Título: Desarrollo de una Herramienta de Validación de Integridad de Datos entre Hojas de Cálculo
Enero 2026

Autor: José Mario Rodríguez-Salinas Bu

Tutor:
Alejandro Rodríguez González
Lenguajes y Sistemas Informáticos e Ingeniería del Software
ETSI Informáticos
Universidad Politécnica de Madrid

Co-tutora:
Lucía Prieto Santamaría
Lenguajes y Sistemas Informáticos e Ingeniería del Software
ETSI Informáticos
Universidad Politécnica de Madrid

Resumen

Este Trabajo de Fin de Grado presenta el diseño y desarrollo de una herramienta de validación de datos en archivos Excel (.xlsx), orientada a facilitar la detección de errores e inconsistencias mediante reglas configurables por el usuario. En numerosos contextos empresariales, académicos y administrativos, Excel sigue siendo una herramienta clave para el tratamiento de datos. Sin embargo, la ausencia de mecanismos de validación estructurados y reutilizables conduce a errores manuales y pérdida de fiabilidad en los procesos. Este proyecto responde a esa necesidad, ofreciendo una solución visual, intuitiva y extensible.

La herramienta, desarrollada íntegramente en Java utilizando JavaFX para la interfaz gráfica, permite al usuario cargar múltiples archivos Excel, definir reglas de validación personalizadas (como valores obligatorios, duplicados, tipos de datos, referencias cruzadas o condiciones lógicas), ejecutar dichas reglas y obtener un informe detallado de los errores detectados, tanto en la interfaz como de forma exportable en formato Excel.

Desde el punto de vista técnico, el sistema ha sido diseñado siguiendo principios de ingeniería software como la separación en capas, el desacoplamiento funcional y la reutilización de componentes. Se ha implementado un modelo de datos unificado en memoria que permite trabajar con múltiples libros de Excel como si fuesen uno solo, lo que simplifica la validación cruzada entre hojas. La arquitectura modular permite incorporar nuevas reglas sin alterar las ya existentes.

Durante el desarrollo se han aplicado buenas prácticas como el uso de patrones, la gestión de errores con mensajes informativos y el uso de hojas de estilo CSS para mantener una apariencia visual profesional y homogénea. Se ha buscado en todo momento la facilidad de uso, permitiendo que usuarios sin conocimientos técnicos puedan aplicar reglas complejas de validación con solo unos clics.

Las pruebas realizadas han validado el correcto funcionamiento de todas las funcionalidades y han demostrado la utilidad de la herramienta en casos reales. Además de cumplir todos los objetivos definidos, el proyecto ha permitido desarrollar competencias avanzadas en diseño de software, arquitectura de sistemas y usabilidad.

En definitiva, se ha construido una herramienta funcional, flexible y visualmente cuidada, que resuelve un problema real y ofrece una base sólida para futuras ampliaciones o integraciones más complejas.

Abstract

This Bachelor's Thesis presents the design and development of a data validation tool for Excel (.xlsx) files, aimed at facilitating the detection of errors and inconsistencies through user-configurable rules. In many business, academic, and administrative contexts, Excel remains a key tool for data handling. However, the lack of structured and reusable validation mechanisms leads to manual errors and decreased reliability in processes. This project addresses that need by offering a visual, intuitive, and extensible solution.

The tool, fully developed in Java using JavaFX for the graphical interface, allows the user to load multiple Excel files, define custom validation rules (such as required values, duplicates, data types, cross-references, or logical conditions), execute those rules, and obtain a detailed report of the detected errors, both within the interface and as an exportable Excel file.

From a technical perspective, the system has been designed following software engineering principles such as layered architecture, functional decoupling, and component reuse. An in-memory unified data model has been implemented, allowing multiple Excel workbooks to be treated as one, which simplifies cross-sheet validation. The modular architecture enables the addition of new rules without altering existing ones.

Throughout the development process, best practices have been applied, including the use of design patterns, informative error handling, and CSS stylesheets to maintain a professional and consistent visual appearance. Ease of use has been a priority, enabling non-technical users to apply complex validation rules with just a few clicks.

Testing has validated the correct functioning of all features and demonstrated the tool’s usefulness in real-world scenarios. In addition to fulfilling all defined objectives, the project has fostered the development of advanced competencies in software design, system architecture, and usability.

In short, a functional, flexible, and visually refined tool has been built, solving a real-world problem and providing a solid foundation for future expansions or more complex integrations.

Tabla de contenidos

1 Introducción

1.1 Contexto y motivación del proyecto

Durante el segundo curso del grado en Ingeniería Informática, cursé una asignatura centrada en el diseño y gestión de bases de datos. Esta experiencia despertó en mí un especial interés por el tratamiento estructurado de la información, la integridad de los datos y los mecanismos que permiten garantizar su calidad y fiabilidad. A partir de entonces, he buscado profundizar en este ámbito, no solo a través de la formación académica, sino también mediante experiencias prácticas.

Esta motivación me llevó a realizar mis prácticas curriculares en una empresa del sector tecnológico, participando en un proyecto de migración de datos hacia un sistema SAP. Esta experiencia profesional me permitió observar de primera mano cómo se gestionan grandes volúmenes de datos empresariales y, especialmente, cómo se realizan las tareas de validación para asegurar la consistencia y calidad de la información antes de su integración en los nuevos sistemas.

Durante este proyecto, pude comprobar que muchas de estas validaciones se llevaban a cabo mediante métodos manuales, utilizando hojas de cálculo en Excel con fórmulas o macros personalizadas diseñadas por los propios usuarios. Estas validaciones abarcaban desde comprobaciones básicas -como verificar la presencia de ciertos valores o detectar duplicados- hasta condiciones complejas que implicaban relaciones entre múltiples hojas y columnas. A pesar de su utilidad, este enfoque presentaba importantes limitaciones: requería conocimientos técnicos específicos, era difícil de mantener o adaptar, estaba expuesto a errores humanos, y no resultaba escalable en contextos con gran volumen de datos o reglas complejas.

Fruto de esa observación surgió la idea del presente Trabajo de Fin de Grado: diseñar y desarrollar una herramienta que facilite la validación de datos en archivos Excel, permitiendo definir reglas personalizadas a través de una interfaz gráfica intuitiva, y detectando automáticamente errores o incoherencias. El objetivo es aportar una solución flexible, reutilizable y accesible que simplifique este tipo de tareas tanto en entornos empresariales como académicos o personales.

1.2 Objetivos del trabajo

El objetivo principal de este Trabajo de Fin de Grado es el diseño y desarrollo de una herramienta de validación de datos en archivos Excel, que permita detectar errores o inconsistencias mediante la aplicación personalizada de reglas típicas de validación, sin necesidad de conocimientos técnicos avanzados por parte del usuario. La herramienta está orientada a facilitar tareas habituales de verificación de datos en entornos donde Excel sigue siendo una herramienta predominante, como ocurre en numerosos contextos empresariales y académicos.

Para alcanzar dicho objetivo general, se han definido los siguientes objetivos específicos:

- Carga de múltiples archivos Excel: Permitir al usuario cargar varios archivos de forma simultánea, para poder validar datos que se encuentran distribuidos en distintas hojas y libros.

- Visualización estructurada de archivos y hojas: Mostrar de forma clara los archivos cargados y las hojas disponibles en cada uno, facilitando su selección y posterior uso en las reglas.

- Procesamiento unificado del contenido: Integrar y estructurar la información cargada para permitir una validación coherente entre distintos archivos y hojas.

- Definición de reglas de validación personalizadas: Ofrecer al usuario la posibilidad de crear reglas simples y complejas, que relacionen valores entre celdas, rangos y hojas diferentes, con operadores lógicos y comparativos.

- Gestión de reglas mediante interfaz gráfica: Implementar una interfaz visual amigable que permita definir y eliminar reglas de validación sin necesidad de escribir código.

- Ejecución de validaciones y generación de resultados: Aplicar las reglas definidas sobre los archivos cargados y mostrar los errores encontrados de forma clara.

- Visualización de errores en la interfaz: Destacar de forma visual las celdas que no cumplen las reglas establecidas, para facilitar su corrección.

- Exportación del informe de validación: Permitir al usuario guardar un informe con los errores detectados en formato Excel.

Estos objetivos pretenden proporcionar una solución completa y flexible a las necesidades comunes de validación en hojas de cálculo, facilitando su adopción por usuarios sin perfil técnico y reduciendo los errores derivados de procesos manuales.

1.3 Alcance y limitaciones

El desarrollo del proyecto ha abarcado un conjunto amplio de funcionalidades orientadas a cubrir distintas necesidades de validación de datos en hojas de cálculo, con un enfoque flexible y accesible para el usuario. Desde las primeras fases se estableció como prioridad ofrecer un sistema capaz de gestionar múltiples archivos Excel de forma simultánea, permitiendo al usuario explorar fácilmente sus hojas y contenidos.

A nivel funcional, se ha incorporado un completo sistema de definición de reglas, accesible desde una interfaz gráfica intuitiva. Estas reglas permiten establecer condiciones específicas sobre celdas, columnas o rangos, sin requerir conocimientos técnicos ni la escritura de fórmulas. Se han implementado distintos tipos de validación, entre ellos la comprobación de valores obligatorios, la detección de duplicados, la validación de tipos de datos y la verificación de referencias cruzadas entre hojas. Además, se han desarrollado mecanismos más avanzados como comparaciones entre celdas de diferentes ubicaciones y reglas condicionales del tipo “si… entonces…”, lo que amplía considerablemente la capacidad de control sobre la coherencia de los datos.

La aplicación integra un motor de validación que recorre automáticamente los datos cargados, detectando cualquier incumplimiento de las reglas definidas. Los errores se presentan de forma clara mediante una ventana de resultados que detalla el archivo, la hoja, la ubicación exacta y la naturaleza del problema. Asimismo, se ofrece al usuario la posibilidad de exportar estos informes para su revisión o conservación.

En cuanto a la gestión de reglas, se permite su creación y eliminación durante la sesión de trabajo, ofreciendo un entorno flexible y controlado para el análisis de los datos. Todo ello se realiza sin alterar en ningún momento los archivos originales, ya que la validación se efectúa de forma externa, preservando la integridad de los ficheros.

El sistema, no obstante, presenta algunas limitaciones. Actualmente, solo se garantiza la compatibilidad con archivos en formato .xlsx, quedando fuera de alcance otros formatos como .xls o documentos provenientes de otras plataformas. Además, aunque se ha diseñado para trabajar con estructuras de tabla estándar, no se contempla el tratamiento de celdas combinadas, cabeceras visuales o formatos atípicos. Tampoco se ha integrado una edición

directa sobre los archivos, por lo que cualquier corrección debe realizarse manualmente fuera de la herramienta.

Finalmente, las reglas se gestionan dentro de cada sesión de uso, si bien el sistema se ha desarrollado de forma modular y abierta, con vistas a futuras ampliaciones que puedan incluir la persistencia de configuraciones y la carga automática de conjuntos de reglas predefinidos.

1.4 Metodología y planificación

El desarrollo del proyecto se ha guiado por una metodología iterativa e incremental, que ha permitido avanzar de forma progresiva en la definición, diseño, construcción y validación de la herramienta. En lugar de optar por un enfoque rígido con fases completamente cerradas, se ha seguido un modelo de trabajo que combina una planificación inicial detallada con ciclos sucesivos de implementación, evaluación y mejora continua.

Desde las fases iniciales, se comenzó con una revisión de requisitos, tanto funcionales como técnicos, que permitió delimitar claramente los objetivos del proyecto. A partir de esta base, se definió una planificación estructurada en tareas y fases, con un cronograma orientativo reflejado en un diagrama de Gantt que se puede observar en la **Figura 1.1**.

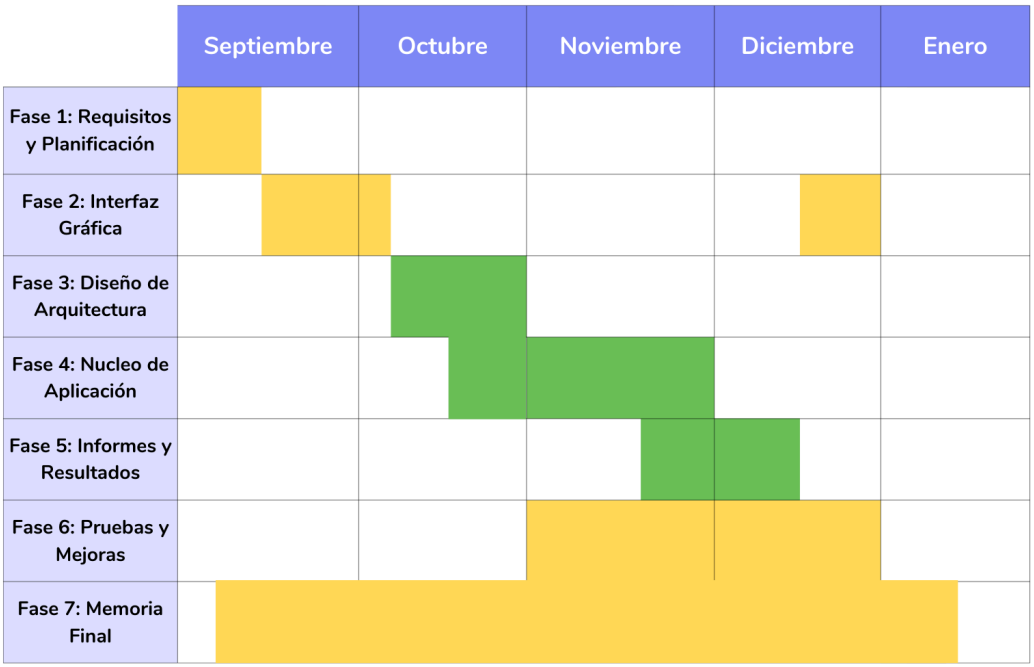


Figura 1.1. Diagrama de Gantt del desarrollo de la herramienta

Durante el proceso, cada fase ha tenido un propósito concreto. Las primeras etapas se centraron en el análisis de herramientas disponibles y en el diseño de la interfaz gráfica del sistema. Posteriormente, se trabajó en el diseño de la arquitectura y en la implementación del núcleo funcional: carga y visualización de archivos, definición y gestión de reglas, y ejecución del proceso de validación. Todo ello se ha llevado a cabo mediante una planificación flexible, que ha permitido adaptarse a las necesidades y aprendizajes surgidos en el camino.

A lo largo del desarrollo, se han ido validando los avances mediante pruebas funcionales con archivos reales. Este enfoque ha favorecido la detección temprana de errores, la incorporación de mejoras y el reajuste de tareas según las prioridades del momento.

Además, en la fase final del desarrollo, se ha realizado una revisión y mejora de la interfaz gráfica de la herramienta, con el objetivo de reforzar su apariencia profesional. Esto pone de manifiesto la importancia otorgada al aspecto visual de la aplicación como parte fundamental de la experiencia de uso.

Todo el proceso de desarrollo se ha documentado de manera progresiva a lo largo del proyecto, con el objetivo de facilitar la posterior redacción de la memoria final. Esta documentación continua ha permitido recoger de forma ordenada las decisiones de diseño adoptadas y las dificultades encontradas, contribuyendo así a una descripción más precisa y coherente del trabajo realizado.

En conjunto, esta planificación adaptativa ha permitido combinar una visión global clara con la capacidad de respuesta ante retos concretos, manteniendo siempre el foco en el objetivo principal del proyecto: desarrollar una herramienta útil, robusta y accesible para la validación de datos en ficheros Excel.

2 Estado del arte

2.1 Herramientas existentes de validación de datos

En el ámbito de la validación de datos, una de las herramientas más utilizadas a nivel empresarial es Microsoft Excel, tanto por su accesibilidad como por su versatilidad. Excel incorpora funciones básicas de validación, como listas desplegables, restricciones de tipo de datos o límites de valores, accesibles a través de su opción integrada de "validación de datos" [1]. Sin embargo, cuando se requiere validar relaciones entre distintas hojas o archivos, o realizar comprobaciones más complejas, los usuarios recurren habitualmente al uso de fórmulas avanzadas o incluso a macros en VBA (Visual Basic for Applications), diseñadas de forma personalizada para cada caso [2]. Estas soluciones, aunque potentes, presentan una serie de limitaciones: son propensas a errores humanos, difíciles de mantener, y poco escalables si se desean aplicar a conjuntos de datos grandes o a estructuras cambiantes.

En entornos más profesionales, existen plataformas especializadas de integración y calidad de datos que ofrecen funcionalidades avanzadas de validación. Entre las más destacadas se encuentran herramientas ETL (Extract, Transform, Load) como Talend, Informatica o Apache Nifi, que permiten modelar flujos de validación complejos mediante interfaces visuales o scripts personalizables [3]. Estas soluciones son ampliamente utilizadas en grandes organizaciones y en proyectos con necesidades críticas de gobernanza del dato. No obstante, su integración implica una curva de aprendizaje considerable, así como infraestructuras técnicas más complejas, lo que limita su adopción en contextos más ligeros o en validaciones centradas en ficheros Excel aislados.

Por tanto, se observa un hueco entre las validaciones manuales en Excel y las plataformas profesionales de calidad de datos, que justifica la necesidad de herramientas intermedias que permitan validar ficheros Excel con mayor flexibilidad, sin necesidad de conocimientos técnicos avanzados ni de infraestructuras complejas.

2.2 Necesidades actuales en la validación de ficheros Excel

En el contexto actual, la gestión de datos mediante hojas de cálculo como Microsoft Excel sigue siendo habitual tanto en entornos empresariales como académicos. Si bien Excel incluye funcionalidades básicas de validación para restringir la entrada de datos, dichas capacidades resultan insuficientes cuando se trata de aplicar reglas complejas o dinámicas. En particular, los procesos de validación manual mediante fórmulas o macros implican una

carga de trabajo elevada, están sujetos a errores humanos y complican el mantenimiento cuando los datos o los criterios evolucionan [4].

Esta situación genera varias necesidades clave en la validación de ficheros Excel. En primer lugar, es necesario disponer de una interfaz que permita definir reglas dinámicas, reutilizables y adaptadas a distintos rangos, hojas o incluso archivos completos, sin que el usuario deba escribir fórmulas o código. En segundo lugar, se requiere que dichas reglas puedan contemplar relaciones entre hojas diferentes, referencias cruzadas o condiciones del tipo "si... entonces...", lo cual va más allá de lo que Excel valida de forma nativa o mediante macros aisladas.

Además, la creciente cantidad de datos y la mayor complejidad de los procesos exigen un mecanismo de validación que sea escalable, automatizable y auditable. El uso intensivo de hojas de cálculo en entornos organizados sin sistemas de control adecuados conlleva riesgos elevados de errores, duplicados, pérdida de integridad o baja trazabilidad [5]. Asimismo, la tendencia actual hacia la digitalización y la analítica de datos impulsa la necesidad de herramientas que integren validación automatizada como parte del flujo de calidad del dato [6]. Finalmente, se hace necesaria la generación de informes claros que faciliten la identificación, localización y corrección de los fallos detectados.

En resumen, la combinación del uso extendido de Excel, la insuficiencia de sus mecanismos tradicionales de validación, y la demanda de mayor eficiencia y control en los procesos de calidad de datos, demuestra la necesidad de una herramienta especializada que responda a estas carencias y facilite la validación automatizada, confiable y reutilizable de ficheros Excel.

2.3 Tecnologías seleccionadas

Para el desarrollo de la herramienta de validación automatizada de ficheros Excel, se ha optado por utilizar tecnologías ampliamente consolidadas y de fácil integración, que permiten construir una aplicación robusta, extensible y mantenible.

El lenguaje principal utilizado ha sido Java, debido a su versatilidad, madurez y ecosistema de bibliotecas. Java resulta especialmente adecuado para aplicaciones de escritorio con interfaces gráficas, ya que permite un control preciso sobre la lógica del programa, el tratamiento de errores y la portabilidad entre sistemas operativos.

Para la construcción de la interfaz gráfica de usuario (GUI, en inglés) se ha seleccionado JavaFX [7], una librería moderna que facilita el desarrollo de interfaces dinámicas, con componentes visuales personalizables y capacidad

de separación entre lógica y presentación. JavaFX permite, además, una buena organización en el diseño modular de pantallas, facilitando futuras ampliaciones o mejoras visuales.

La manipulación y lectura de archivos Excel se ha realizado mediante la librería Apache POI [8], que proporciona un conjunto de herramientas para el tratamiento de documentos Microsoft Office. Esta librería permite acceder a celdas, rangos, hojas y estructuras internas de los archivos Excel con un alto grado de control, lo que resulta fundamental para implementar la validación a distintos niveles (celdas, columnas, hojas, etc.).

En lo que respecta al almacenamiento en memoria de reglas, configuraciones y resultados, se ha diseñado un sistema interno basado en estructuras de datos propias.

Finalmente, para la exportación de los resultados, al igual que para la lectura y manipulación de los archivos, se ha utilizado la biblioteca Apache POI. Esta herramienta permite generar ficheros de salida en formato Excel, facilitando así la documentación, el análisis y la revisión de los errores detectados durante el proceso de validación.

La combinación de estas tecnologías proporciona una solución sólida y escalable, centrada tanto en la usabilidad como en la fiabilidad de los resultados, al tiempo que sienta las bases para posibles ampliaciones funcionales o mejoras futuras del sistema.

2.4 Aportación personal del proyecto

La principal aportación de este proyecto reside en el diseño y desarrollo de una herramienta flexible y accesible para la validación automatizada de ficheros Excel, orientada a usuarios técnicos y no técnicos que necesitan verificar la integridad de datos en procesos administrativos, contables, de migración o análisis.

A diferencia de las soluciones existentes, generalmente integradas en sistemas empresariales cerrados o bien desarrolladas de forma ad hoc mediante fórmulas y macros personalizadas, esta herramienta propone un enfoque modular y reutilizable que permite definir reglas de validación de forma gráfica, sin necesidad de conocimientos de programación, y aplicarlas sobre múltiples archivos Excel cargados simultáneamente.

A nivel técnico, el desarrollo ha sido completamente personalizado, abarcando desde el diseño de la interfaz hasta la implementación del motor de validación. Se ha puesto especial énfasis en garantizar la claridad de la interfaz gráfica, facilitando la navegación, la creación de reglas y la interpretación de los

errores detectados. Esta interfaz se apoya en una estructura lógica de pantallas que guía al usuario en un flujo intuitivo de trabajo.

Además, se ha introducido un sistema de reglas flexible, capaz de manejar validaciones comunes como la detección de valores obligatorios, duplicados o tipos de datos incorrectos, así como reglas más avanzadas que implican relaciones entre hojas o reglas condicionales. La arquitectura de validación ha sido diseñada de manera extensible, permitiendo incorporar nuevos tipos de reglas sin alterar el funcionamiento de las ya existentes.

En definitiva, la aportación personal del proyecto radica no solo en su desarrollo técnico, sino en haber identificado una necesidad real observada en un entorno profesional y proponer una solución generalizable y eficaz, orientada a mejorar la eficiencia, la precisión y la trazabilidad en la validación de datos estructurados en hojas de cálculo.

3 Análisis del sistema

3.1 Requisitos funcionales

Los requisitos funcionales han de definir de forma precisa qué debe ser capaz de hacer la herramienta desarrollada. Se han concretado los siguientes requisitos funcionales:

ID del Requisito	RF1
Nombre del Requisito	Carga de archivos Excel múltiples.
Descripción	El sistema permitirá seleccionar y cargar múltiples archivos de Excel en una misma sesión.
Prioridad	Muy alta.

ID del Requisito	RF2
Nombre del Requisito	Visualización estructurada del contenido.
Descripción	Se mostrará de manera jerárquica y organizada la lista de archivos cargados junto con sus respectivas hojas.
Prioridad	Media.

ID del Requisito	RF3
Nombre del Requisito	Procesamiento unificado de datos.
Descripción	Toda la información se estructurará internamente en un modelo de datos común para facilitar la validación entre hojas y archivos distintos.
Prioridad	Alta.

ID del Requisito	RF4
Nombre del Requisito	Definición de reglas de validación.
Descripción	El usuario podrá crear reglas de validación utilizando una interfaz gráfica, sin necesidad de escribir código.
Prioridad	Muy alta.

ID del Requisito	RF5
Nombre del Requisito	Validación de valores obligatorios.
Descripción	Se podrán definir ubicaciones que deben contener un valor no vacío, generando errores si no se cumple.
Prioridad	Alta.

ID del Requisito	RF6
Nombre del Requisito	Detección de valores duplicados.
Descripción	La herramienta detectará valores repetidos en ubicaciones donde no están permitidos.
Prioridad	Alta.

ID del Requisito	RF7
Nombre del Requisito	Validación de tipos de datos.
Descripción	Se comprobará si los datos cumplen con el tipo definido (texto, número, fecha, etc.).

Prioridad	Alta.

ID del Requisito	RF8
Nombre del Requisito	Verificación de referencias cruzadas.
Descripción	Se permitirá definir qué valores de una hoja deben existir en otra.
Prioridad	Alta.

ID del Requisito	RF9
Nombre del Requisito	Regla de comparación entre valores.
Descripción	La herramienta permitirá definir reglas de comparación directa entre valores de celdas, rangos o columnas, o bien compararlos con valores fijos, utilizando operadores como igualdad, desigualdad, mayor que o menor que.
Prioridad	Alta.

ID del Requisito	RF10
Nombre del Requisito	Regla condicional.
Descripción	El sistema permitirá definir reglas condicionales en las que, si se cumple una condición determinada sobre una ubicación, se deba cumplir una acción específica sobre otra ubicación distinta, permitiendo validar relaciones lógicas complejas entre datos.
Prioridad	Alta.

ID del Requisito	RF11
Nombre del Requisito	Visualización de reglas definidas.
Descripción	La aplicación mostrará un resumen de las reglas creadas en la sesión, permitiendo eliminar cualquiera de estas de forma sencilla.
Prioridad	Alta.

ID del Requisito	RF12
Nombre del Requisito	Ejecución de validaciones.
Descripción	El usuario podrá ejecutar todas las reglas definidas sobre los archivos cargados, obteniendo los resultados de la validación de forma inmediata.
Prioridad	Muy alta.

ID del Requisito	RF13
Nombre del Requisito	Visualización de errores.
Descripción	La aplicación mostrará un resumen de los errores encontrados.
Prioridad	Alta.

ID del Requisito	RF14
Nombre del Requisito	Exportación del informe.
Descripción	Los resultados de la validación se podrán exportar en formato Excel para su revisión externa.

Prioridad	Media.

ID del Requisito	RF15
Nombre del Requisito	Interfaz visual clara y profesional.
Descripción	La herramienta incluirá elementos gráficos visuales para facilitar su uso por parte de usuarios no técnicos.
Prioridad	Alta.

Todos estos requisitos han sido implementados completamente en la herramienta final, garantizando así una experiencia completa de validación de datos orientada a entornos reales.

3.2 Requisitos no funcionales

Los requisitos no funcionales definen criterios de calidad que deben cumplir tanto la herramienta como su desarrollo, garantizando aspectos como la usabilidad, el rendimiento, la mantenibilidad o la portabilidad. A continuación se detallan los requisitos no funcionales considerados en el proyecto:

ID del Requisito	RNF1
Nombre del Requisito	Usabilidad.
Descripción	La interfaz gráfica debe ser clara, intuitiva y accesible para usuarios sin conocimientos técnicos avanzados. La definición y ejecución de reglas debe realizarse sin necesidad de escribir código, utilizando formularios visuales guiados.
Prioridad	Muy alta.

ID del Requisito	RNF2

Nombre del Requisito	Buen rendimiento.
Descripción	El sistema debe ser capaz de cargar y procesar múltiples archivos Excel de tamaño medio (hasta unas pocas decenas de miles de celdas por archivo) en un tiempo razonable, sin bloqueos ni retardos perceptibles para el usuario.
Prioridad	Alta.

ID del Requisito	RNF3
Nombre del Requisito	Independencia del entorno.
Descripción	La herramienta debe funcionar en cualquier entorno con Java y JavaFX instalados, sin requerir bases de datos externas ni servicios adicionales. Esto facilita su uso en distintos equipos sin necesidad de configuración avanzada.
Prioridad	Media.

ID del Requisito	RNF4
Nombre del Requisito	Portabilidad.
Descripción	La aplicación debe poder distribuirse como archivo ejecutable, sin depender de rutas absolutas, librerías de sistema ni configuraciones específicas del entorno de desarrollo.
Prioridad	Media.

ID del Requisito	RNF5

Nombre del Requisito	Escalabilidad funcional.
Descripción	La arquitectura debe permitir incorporar nuevas reglas de validación en el futuro sin necesidad de reestructurar el sistema, gracias al uso de una jerarquía de clases basada en interfaces comunes.
Prioridad	Alta.

ID del Requisito	RNF6
Nombre del Requisito	Robustez frente a errores.
Descripción	La herramienta debe gestionar de forma controlada los errores más comunes, como rutas inválidas, archivos corruptos o celdas vacías inesperadas, mostrando mensajes informativos y evitando el cierre abrupto del programa.
Prioridad	Alta.

ID del Requisito	RNF7
Nombre del Requisito	Legibilidad y estructura del código fuente.
Descripción	El código debe estar organizado en paquetes según su funcionalidad (modelo, validación, interfaz, utilidades...), y seguir buenas prácticas para facilitar su comprensión por terceros.
Prioridad	Media.

ID del Requisito	RNF8

Nombre del Requisito	Generación de informes comprensibles.
Descripción	Los resultados de la validación deben mostrarse en un formato claro, tanto en la interfaz como en la exportación, agrupando los errores por regla y permitiendo su fácil interpretación.
Prioridad	Alta.

Estos requisitos no funcionales aseguran que la herramienta no solo cumpla con su funcionalidad principal, sino que además sea práctica, mantenible y usable en contextos reales, incluso por usuarios sin perfil técnico.

3.3 Actores y casos de uso

En este proyecto se identifica un único actor principal: Usuario final.

Se trata de la persona que utiliza la herramienta para validar datos en hojas de cálculo Excel. No se requiere que tenga conocimientos técnicos o de programación, pero sí se espera que conozca la estructura y significado de los datos que está validando.

Este actor interactúa con todas las funcionalidades de la herramienta a través de una interfaz gráfica amigable. Las acciones que puede realizar se resumen a través de los siguientes casos de uso principales:

Caso de uso	1
Nombre	Cargar archivos Excel.
Descripción	El usuario selecciona uno o varios archivos Excel para su análisis.
Precondición	Los archivos deben estar en formato válido (.xlsx).
Resultado esperado	Los archivos y sus hojas son cargados correctamente y se muestran en la interfaz.

Caso de uso	2

Nombre	Definir reglas de validación.
Descripción	El usuario accede a un formulario para definir una nueva regla de validación.
Precondición	Debe haber al menos un archivo cargado.
Tipos de regla disponibles	Valor obligatorio, Duplicados, Tipo de dato, Referencias cruzadas, Comparación, Condicional.
Resultado esperado	La regla se guarda en memoria y queda lista para aplicarse sobre los archivos cargados.

Caso de uso	3
Nombre	Visualizar o eliminar reglas.
Descripción	El usuario puede revisar las reglas definidas y eliminarlas si lo desea.
Precondición	Debe haber al menos una regla definida.
Resultado esperado	La lista de reglas activas se actualiza correctamente tras cada operación.

Caso de uso	4
Nombre	Ejecutar validación.
Descripción	El usuario inicia el proceso de validación. La herramienta aplica todas las reglas sobre los datos cargados.
Precondición	Debe haber al menos una regla definida.

Resultado esperado	Se muestra un mensaje confirmando que la validación se ha llevado a cabo correctamente.

Caso de uso	5
Nombre	Visualizar errores detectados.
Descripción	El usuario puede consultar los errores específicos generados por cada regla.
Precondición	Se debe haber ejecutado una validación.
Resultado esperado	Se muestra una lista con los errores de validación, incluyendo la ubicación exacta de cada uno.

Caso de uso	6
Nombre	Exportar resultados.
Descripción	El usuario puede guardar los resultados de la validación en un archivo externo.
Precondición	Se debe haber ejecutado una validación.
Formato de exportación	Excel (.xlsx).
Resultado esperado	Se genera un archivo con el resumen estructurado de errores por regla.

3.4 Flujo de trabajo típico

A continuación, se describe el flujo de trabajo estándar que seguiría un usuario al utilizar la herramienta desarrollada, desde la carga inicial de archivos hasta la exportación final de los resultados de validación. Este flujo

refleja el uso típico en un entorno real, donde se requiere validar información contenida en uno o varios archivos Excel.

- Inicio de la aplicación

El usuario ejecuta la herramienta, accediendo a una interfaz gráfica intuitiva y agradable visualmente, diseñada para facilitar su uso incluso por personas sin conocimientos técnicos.

- Carga de archivos Excel

Desde la pantalla principal, el usuario selecciona y carga uno o varios archivos Excel que contienen los datos a validar. La herramienta permite seleccionar múltiples archivos simultáneamente, integrándolos en una estructura unificada para facilitar su posterior tratamiento conjunto.

- Visualización de archivos y hojas disponibles

Tras la carga, se muestra un resumen visual de los archivos importados y las hojas que contiene cada uno. Esta visualización permite al usuario identificar fácilmente las estructuras disponibles para definir sus reglas de validación.

- Definición de reglas de validación

El usuario accede al módulo de creación de reglas, donde puede elegir entre seis distintos tipos de validaciones disponibles. Cada regla se define mediante un formulario gráfico específico para su tipo, en el que el usuario selecciona fácilmente los archivos, hojas, celdas o rangos implicados.

- Visualización y gestión de las reglas creadas

Las reglas definidas se muestran en una lista estructurada. El usuario puede revisar cada regla, y eliminarla si lo desea, antes de ejecutar la validación.

- Ejecución del proceso de validación

Una vez definidas las reglas, el usuario inicia el proceso de validación. La herramienta recorre los archivos Excel cargados y aplica todas las reglas una a una, detectando errores o inconsistencias en los datos.

- Visualización de resultados en la interfaz

Los resultados de la validación se muestran en una nueva ventana, con un resumen claro de cada regla: tipo, ubicación, estado (sin errores o número de errores encontrados) y un botón para ver detalles. Los mensajes de error están unificados bajo un mismo formato, indicando con claridad el problema y su localización exacta.

- Exportación del informe de validación

Si el usuario lo desea, puede generar un informe externo con los resultados, en formato Excel. Este informe organiza los errores por regla, respetando el mismo formato que se muestra en la interfaz, lo que permite un análisis o distribución posterior de los resultados.

Este flujo refleja la experiencia de usuario final y su interacción con el sistema, desde el primer paso hasta la obtención del informe. Gracias a su diseño modular y progresivo, la herramienta permite realizar validaciones completas de forma intuitiva, sin necesidad de conocimientos técnicos ni manipulación directa de los ficheros.

En la **Figura 3.1** se presenta un diagrama que muestra de forma más concreta el flujo de trabajo típico de la herramienta.

Figura 3.1. Diagrama de flujo del funcionamiento de la herramienta

4 Diseño del sistema

4.1 Arquitectura general de la aplicación

La arquitectura de la herramienta desarrollada se basa en un modelo modular y multicapa, siguiendo principios clásicos de arquitectura en capas (layered architecture), donde cada componente de la aplicación tiene responsabilidades claramente definidas y separadas. Esta aproximación facilita la escalabilidad, el mantenimiento del código y la comprensión del sistema en su conjunto.

4.1.1 Capas de la aplicación

La aplicación está organizada en cuatro capas principales, que colaboran entre sí a través de interfaces bien definidas:

➢ Capa de presentación (interfaz gráfica - GUI)

➢ Capa de lógica de negocio (gestión de reglas y validaciones)

➢ Capa de modelo de datos (estructura interna de los archivos y las reglas)

➢ Capa de utilidades (exportación, validaciones auxiliares, etc.)

A continuación se describe cada una en detalle.

1. Capa de presentación (interfaz gráfica)

Esta capa contiene todos los elementos visuales con los que interactúa el usuario. Se encarga de mostrar las pantallas, recolectar entradas y facilitar la navegación a lo largo de la herramienta.

Incluye las siguientes vistas principales:

- MainView: pantalla principal desde la que se cargan archivos y se gestionan reglas.
- SelectorTipoRegla: ventana intermedia para elegir el tipo de regla a crear.
- VistaReglasDefinidas: muestra los detalles de las reglas creadas en la sesión y permite eliminar cada regla de forma individual.
- VentanaResultadosValidacion: muestra los resultados detallados tras ejecutar las validaciones.
- Formularios gráficos específicos para cada tipo de regla: ReglaValorObligatorioView, ReglaDuplicadosView, etc.

Estas clases están implementadas con JavaFX, y se ha aplicado un archivo estilos.css que define un diseño visual moderno y unificado (botones redondeados, color y tipografía profesional, etc.).

2. Capa de lógica de negocio (control y validación)

Aquí se encuentra el corazón funcional de la aplicación. Esta capa contiene la lógica que gestiona:

- La creación, edición y eliminación de reglas de validación.
- La ejecución de todas las reglas definidas sobre los datos cargados.
- El almacenamiento temporal en memoria de las reglas activas durante la sesión.

Clases clave:

- GestorReglas: actúa como controlador entre la interfaz y la lógica de validación.
- Clases específicas que implementan la lógica de validación para cada tipo de regla: ValidadorValorObligatorio, ValidadorDuplicados, etc.

Esta capa se comunica con la de modelo para obtener los datos a validar y devuelve los resultados a la interfaz para su visualización.

3. Capa de modelo de datos

Define las clases que representan la estructura interna de los datos cargados y las reglas creadas. Se encarga de mantener un modelo unificado en memoria de los archivos y hojas Excel cargados.

Clases destacadas:

- LibroUnificado: representa todos los archivos y hojas cargados como una estructura lógica única.
- Regla: interfaz común que implementan todas las reglas de validación.
- Subclases de Regla: cada una representa un tipo de validación concreta (ReglaValorObligatorio, ReglaDuplicados, etc.).
- ResultadoRegla: estructura que almacena el resultado de aplicar una regla concreta, incluyendo los errores detectados.

Esta capa es esencial para garantizar que todas las reglas puedan operar sobre una representación consistente de los datos.

4. Capa de utilidades

Contiene funciones auxiliares que no pertenecen directamente a la lógica del negocio pero son necesarias para el funcionamiento de la aplicación.

Incluye:

- ExportadorResultados: genera archivos Excel con los resultados de validación.
- Clases de ayuda para validaciones y manejo de ubicaciones en Excel (ValidacionUbicacion, CargadorExcel).

Esta capa está desacoplada del resto, lo que permite su uso flexible en distintos contextos.

4.1.2 Diagrama general de arquitectura

A nivel visual, la arquitectura de la herramienta se representa en la **Figura 4.1**.

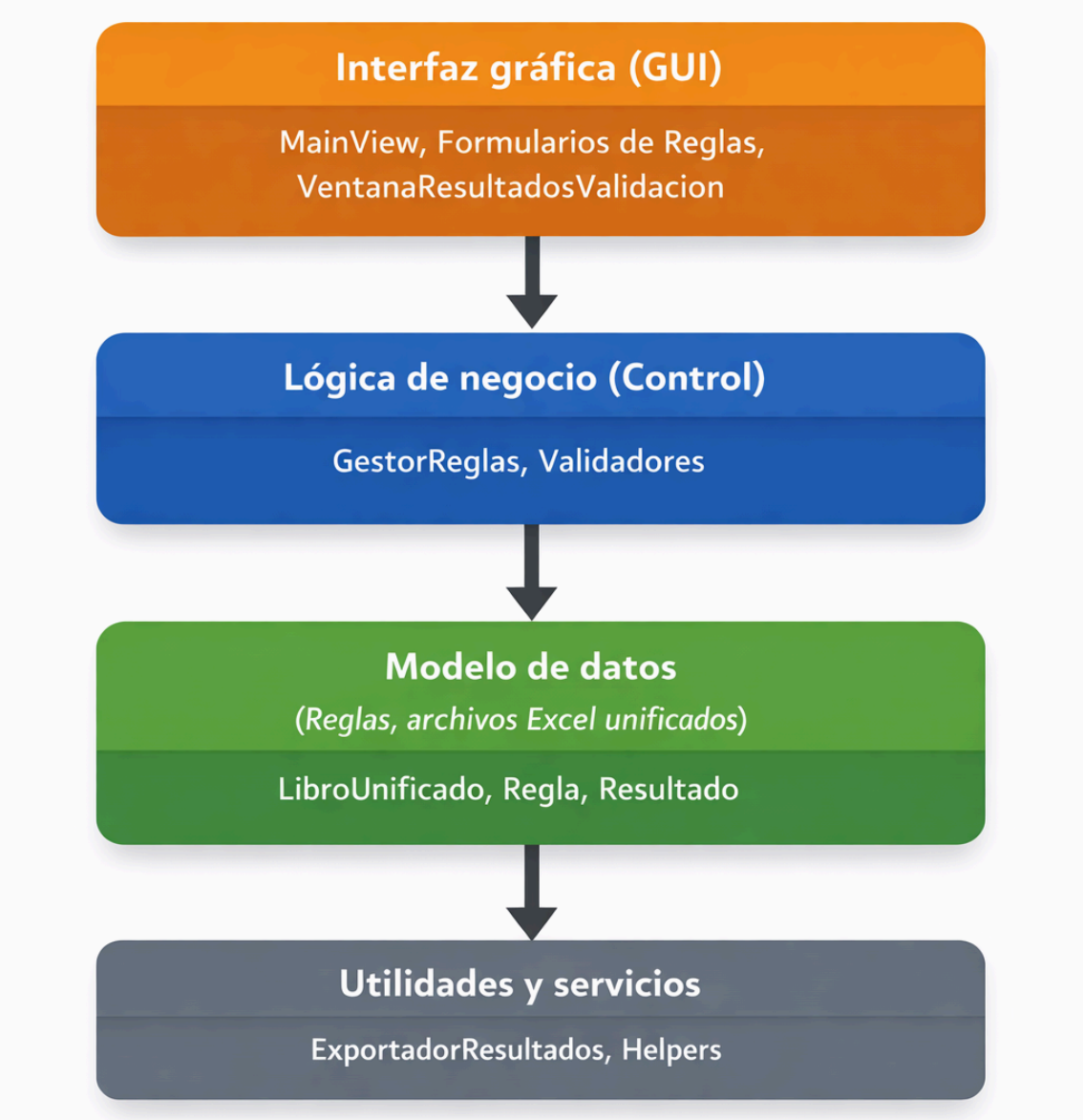


Figura 4.1. Diagrama de arquitectura de la aplicación

4.1.3 Ventajas de esta arquitectura

➢ Separación clara de responsabilidades: cada capa tiene funciones bien definidas, lo que facilita el mantenimiento y futuras ampliaciones.

➢ Escalabilidad y extensibilidad: se pueden añadir nuevos tipos de reglas o nuevos formatos de exportación sin modificar el resto del sistema.

➢ Facilidad para pruebas: la lógica de validación está desacoplada de la interfaz, lo que permite pruebas unitarias independientes.

➢ Reutilización: algunas clases como LibroUnificado o ExportadorResultados son independientes y reutilizables en otros contextos o proyectos.

4.2 Estructura de paquetes y componentes

El sistema se ha organizado de forma modular en paquetes (packages) que agrupan clases con responsabilidades afines. Esta estructura facilita el mantenimiento, mejora la legibilidad del código y permite futuras ampliaciones o sustituciones de componentes de forma aislada.

4.2.1 Entrada principal - package tfg.validación

Contiene la clase App.java, punto de entrada de la aplicación JavaFX. Desde aquí se inicializa la interfaz gráfica principal (MainView) y se aplican estilos visuales.

4.2.2 Capa de presentación (GUI) - package tfg.validación.interfaz

Este paquete agrupa todas las clases relacionadas con la interfaz gráfica del usuario, implementadas con JavaFX. Está diseñado para que cada vista tenga una única responsabilidad visual.

Contienes las siguientes clases:

- MainView: vista principal de la aplicación. Permite cargar archivos, definir reglas y lanzar validaciones.

- SelectorTipoRegla: muestra una ventana para elegir el tipo de regla a definir.

- VistaReglasDefinidas: panel lateral que muestra la lista de reglas definidas en la sesión.

- VentanaResultadosValidacion: ventana emergente que muestra los errores detectados tras la validación, con opción de exportar resultados.

- Vistas de formularios de reglas:
  - ReglaValorObligatorioView
  - ReglaDuplicadosView
  - ReglaTipoDatoView
  - ReglaReferenciasCruzadasView
  - ReglaComparaciónView

- ReglaCondicionalView

- GestorReglas: controlador auxiliar que gestiona la creación de nuevas ventanas de definición de reglas.

- estilos.css: hoja de estilo aplicada a toda la interfaz, diseñada con una estética moderna, limpia y profesional.

4.2.3 Modelo de datos - package tfg.validación.modelo

Contiene la clase LibroUnificado que representa la estructura interna de los datos con los que trabaja la aplicación y constituye el modelo central en memoria.

4.2.4 Definición de reglas - package tfg.validación.reglas

Este paquete contiene la interfaz base Regla y sus distintas implementaciones. Cada clase concreta de regla contiene sus atributos y lógica de configuración, pero delega la validación real en su correspondiente Validador.

Contiene las siguientes clases:

- Regla: interfaz común que deben implementar todas las reglas.

- Subclases de Regla:
  - ReglaValorObligatorio
  - ReglaDuplicados
  - ReglaTipoDato
  - ReglaReferenciasCruzadas
  - ReglaComparación
  - ReglaCondicional

- MemoriaReglas: clase de instancia única que almacena en memoria la lista de reglas definidas por el usuario durante la sesión.

- ResultadoRegla: encapsula los resultados de la validación de una regla, incluyendo los errores detectados.

4.2.5 Lógica de validación - package tfg.validación.validación

Contiene las clases que implementan la lógica real de validación de cada tipo de regla. Estas clases se encargan de aplicar la regla al LibroUnificado y generar un ResultadoRegla.

- Validadores incluidos:
    - ValidadorValorObligatorio
    - ValidadorDuplicados
    - ValidadorTipoDato
    - ValidadorReferenciasCruzadas
    - ValidadorComparación
    - ValidadorCondicional

Cada validador implementa un método principal que recorre las celdas relevantes, aplica las condiciones definidas por el usuario y construye un informe de errores detallado.

4.2.6 Funcionalidades auxiliares - package tfg.validación.utilidades

Paquete con clases de soporte que no pertenecen directamente al modelo o la lógica de negocio, pero que ofrecen servicios esenciales para el funcionamiento general.

Contiene las siguientes clases:

- CargadorExcel: encargada de leer los archivos Excel seleccionados por el usuario y convertirlos al modelo LibroUnificado.

- ExportadorResultados: genera un archivo Excel con los resultados de validación estructurados por regla.

- ValidacionUbicacion: clase utilitaria para validar y estandarizar las ubicaciones (celdas, rangos, columnas) sobre las que se aplican las reglas.

4.3 Modelo de datos unificado

El tratamiento interno de los datos en la herramienta se basa en un modelo de datos unificado que permite representar múltiples archivos Excel cargados por el usuario de forma coherente, estructurada y homogénea. Este modelo es fundamental para facilitar la definición y ejecución de reglas de validación sin importar la procedencia ni el formato específico de cada archivo.

4.3.1 LibroUnificado: representación centralizada

La clase LibroUnificado es la responsable de centralizar la información de todos los archivos Excel cargados en una única estructura de datos. Su objetivo principal es abstraer al resto del sistema de los detalles técnicos del

manejo de múltiples archivos, proporcionando una interfaz clara para acceder a hojas, celdas y rangos de forma unificada.

Cada archivo Excel cargado se representa mediante un identificador (su nombre), y dentro de él se almacenan las hojas que contiene, cada una con sus filas y celdas correspondientes. De esta forma, LibroUnificado mantiene una estructura jerárquica del tipo:

Archivo → Hoja → Fila → Celda

4.3.2 Acceso a datos desde las reglas

El modelo expone una serie de métodos de acceso y consulta que son utilizados por los distintos tipos de reglas durante su ejecución. Las validaciones pueden acceder directamente a:

- Todas las hojas de un archivo dado.
- El contenido de una celda concreta dada su ubicación (archivo, hoja, columna, fila).
- Rangos completos de celdas dentro de una hoja.
- Columnas completas para validar duplicados o tipos de datos.
- Cruces de valores entre hojas distintas.

Por ejemplo, una regla puede utilizar LibroUnificado para obtener todos los valores de la columna B de la hoja "Clientes" del archivo "Negocio.xlsx", y después compararlos con una columna similar en otra hoja o archivo. Esta capacidad de abstracción permite que las reglas se apliquen sobre múltiples archivos como si fueran una única fuente coherente de datos.

4.3.3 Conversión y normalización de datos

Durante la carga de archivos, LibroUnificado también se encarga de:

- Normalizar los valores de celdas, para convertir fechas, números y textos a un formato que pueda ser validado de forma uniforme.
- Detectar celdas vacías o nulas, para facilitar la lógica de reglas como "Valor obligatorio".
- Evitar duplicidad de nombres de hojas en caso de conflictos entre archivos, mediante una estructura de acceso basada en nombres completos.

Esta lógica de homogeneización es clave para garantizar la fiabilidad de la validación y reducir errores típicos que ocurren cuando los datos provienen de fuentes Excel diferentes.

4.3.4 Independencia del formato físico

Esta representación intermedia en LibroUnificado hace que las reglas no necesiten interactuar directamente con las APIs de lectura de archivos Excel (como Apache POI).

Esto permite una separación total entre lógica de validación y lectura física de datos, mejorando la mantenibilidad y facilitando futuras extensiones del sistema (como admitir nuevos formatos o fuentes de datos).

4.4 Interfaz gráfica de usuario (GUI)

La interfaz gráfica de usuario (GUI) de la herramienta ha sido desarrollada íntegramente en JavaFX, buscando lograr una experiencia de uso sencilla, profesional y accesible para todo tipo de usuarios, independientemente de su perfil técnico.

El diseño de la interfaz ha seguido criterios de claridad, orden visual y coherencia funcional, permitiendo que todas las acciones necesarias para el proceso de validación (carga de archivos, definición de reglas, ejecución, exportación, etc.) puedan realizarse de forma intuitiva y guiada.

4.4.1 Ventana principal

La clase principal MainView representa la ventana de inicio de la aplicación. En ella se muestran los principales botones de acción, distribuidos horizontalmente en la parte superior:

➢ Cargar archivo Excel: abre un explorador de archivos para seleccionar uno o varios ficheros Excel (.xlsx) que serán cargados y unificados.

➢ Definir reglas: abre un selector visual con los tipos de reglas disponibles, desde el cual se puede abrir el formulario correspondiente.

➢ Ver reglas definidas: muestra una ventana con todas las reglas actualmente configuradas por el usuario.

➢ Validar: ejecuta todas las reglas definidas en la sesión y muestra el resultado.

Bajo estos botones se encuentra un componente tipo TreeView, como se observa en la **Figura 4.2**, donde se listan los archivos y hojas actualmente cargados, con su estructura jerárquica. Este componente se actualiza dinámicamente conforme se cargan archivos, y permite verificar que la lectura se ha realizado correctamente, como se observa en la **Figura 4.3**.

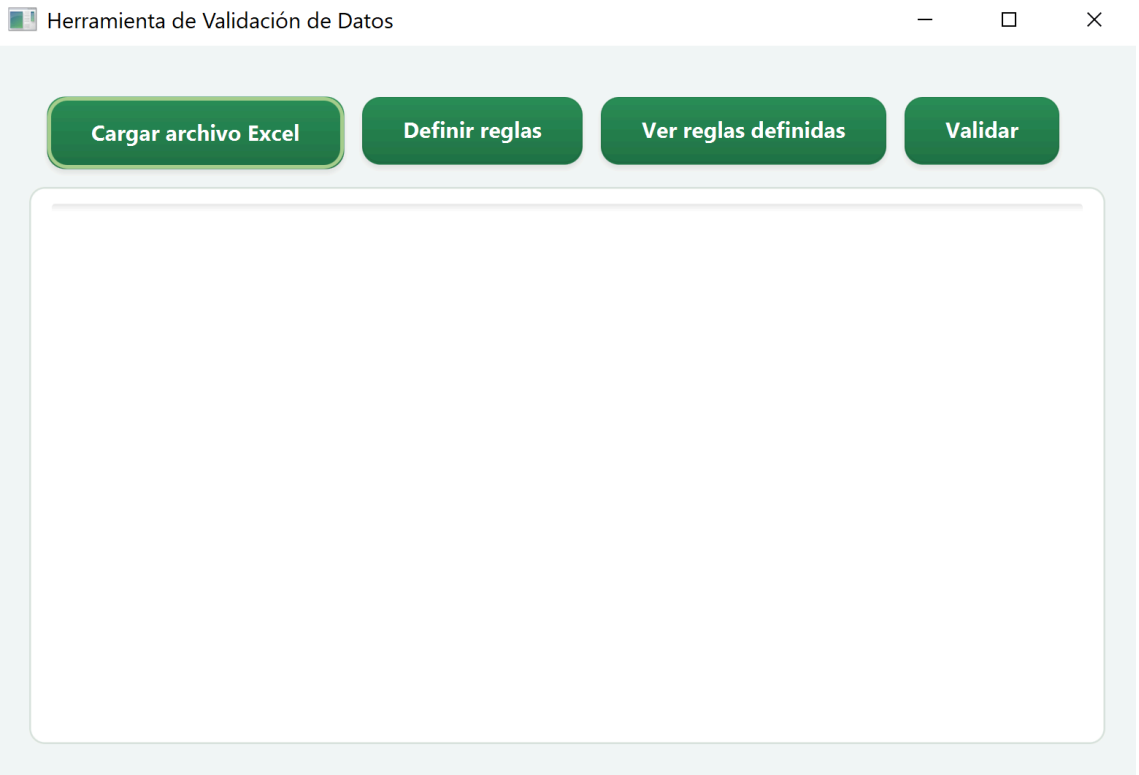


Figura 4.2. Ventana principal de la herramienta

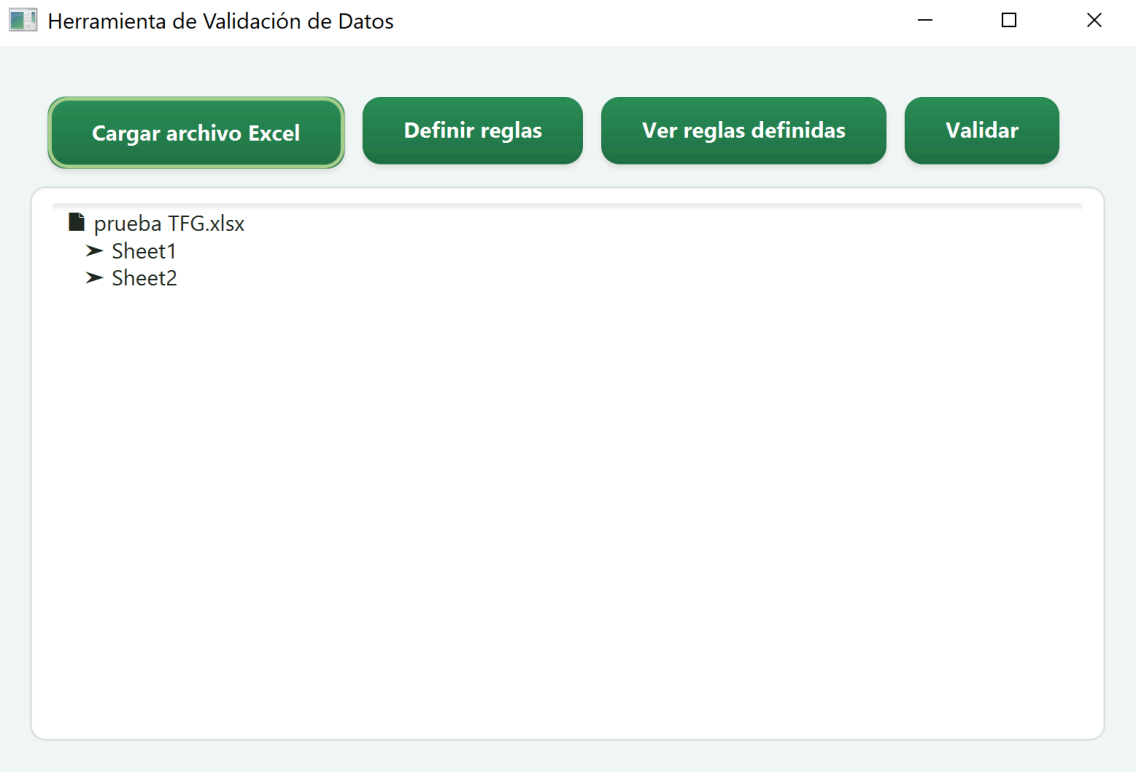


Figura 4.3. Ventana principal con un archivo de prueba cargado

4.4.2 Formularios de definición de reglas

Cada tipo de regla de validación cuenta con un formulario visual independiente que permite su configuración personalizada. Estas vistas se han implementado mediante clases que extienden directamente de layouts como VBox, aprovechando la flexibilidad de los contenedores estándar de JavaFX. Dado que VBox hereda de la clase base Region, estas vistas también pueden beneficiarse de la capacidad de aplicar estilos CSS, gestionar automáticamente el tamaño de sus elementos y adaptarse dinámicamente a diferentes resoluciones de pantalla. En la **Figura 4.4** se puede observar el formulario de selección de regla, previo al formulario específico de la regla seleccionada.

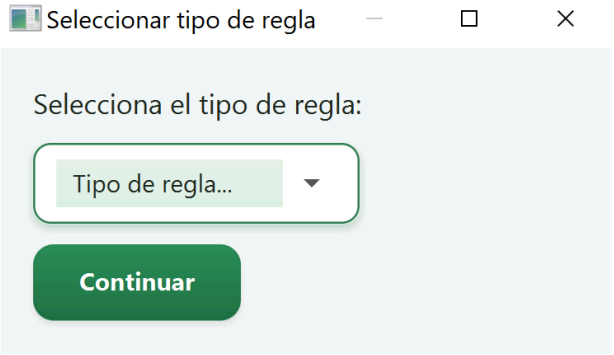


Figura 4.4. Ventana de selección de regla

Los formularios permiten al usuario seleccionar archivos, hojas, columnas, filas o celdas involucradas en la regla, y establecer los parámetros necesarios (valores esperados, condiciones, comparaciones, etc.).

Las vistas desarrolladas incluyen:

- ReglaValorObligatorioView (véase Anexo A.1)
- ReglaDuplicadosView (véase Anexo A.2)
- ReglaTipoDatoView (véase Anexo A.3)
- ReglaReferenciasCruzadasView (véase Anexo A.4)
- ReglaComparaciónView (véase Anexo A.5 y Anexo A.6)
- ReglaCondicionalView (véase Anexo A.7)

Cada una de estas vistas es cargada mediante GestorReglas, que se encarga de abrir una nueva ventana Stage con la escena y dimensiones correspondientes. Todos los formularios comparten una estructura visual coherente, con etiquetas, campos de selección, botones de confirmación y un mensaje informativo común (Label info) para mostrar errores o instrucciones.

4.4.3 Vista de reglas definidas

Como se puede ver en la **Figura 4.5**, la clase VistaReglasDefinidas muestra en una nueva ventana todas las reglas actualmente definidas, permitiendo al usuario revisar su configuración de forma ordenada. Se presenta una lista vertical con una descripción breve de cada regla, que incluye la ubicación sobre la que actúa y los parámetros definidos, facilitando así la verificación de que todas las reglas necesarias han sido correctamente configuradas antes de ejecutar la validación.

Además, cada entrada de la lista incorpora un botón de eliminación, que permite borrar una regla específica de forma individual. Esta funcionalidad mejora la experiencia del usuario, al permitir corregir o ajustar reglas concretas sin necesidad de reiniciar la sesión en la herramienta.

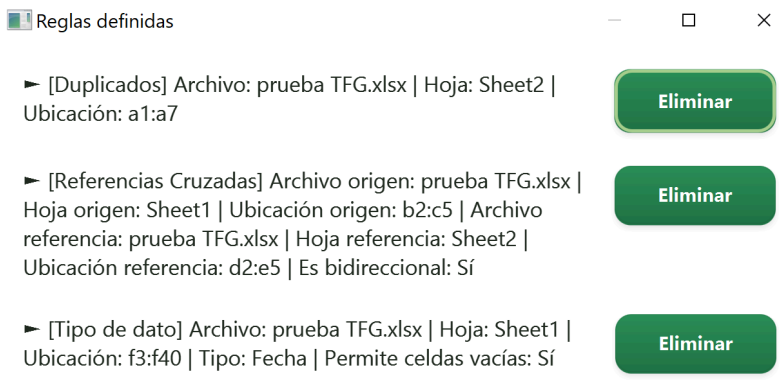


Figura 4.5. Vista de reglas definidas

4.4.4 Resultados de validación

Tras ejecutar las validaciones, la clase VentanaResultadosValidacion se encarga de mostrar en una ventana los resultados obtenidos, organizados por cada regla definida. Para cada regla, se presenta un resumen descriptivo que incluye la información básica de la regla y el número total de errores detectados, junto con un botón de “Ver detalles” que permite acceder a un resumen detallado de los errores. Dicha vista se puede observar en la **Figura 4.6**.

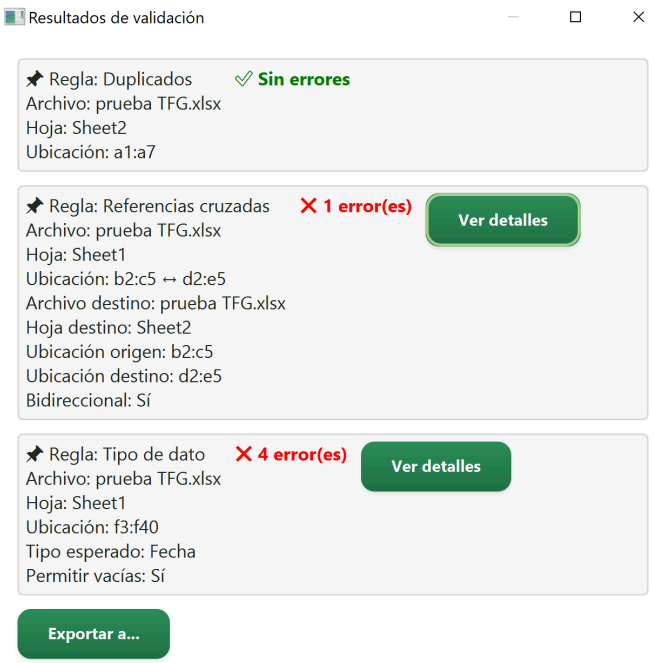


Figura 4.6. Vista de resultados de validación

En la vista de detalles, mostrada en la **Figura 4.7**, los errores se presentan de forma estructurada y homogénea, con una lista detallada de los errores individuales asociados a la regla. Asimismo, cuando una regla no produce errores, se muestra un mensaje informativo que lo indica explícitamente, reforzando la confianza del usuario en el proceso de validación.

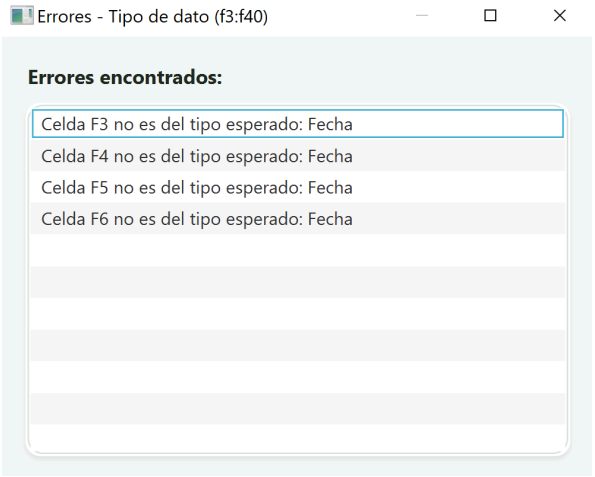


Figura 4.7. Vista de "Ver detalles"

Desde la ventana de resultado de validación se ofrece además la posibilidad de exportar los resultados en formato Excel, mediante un botón "Exportar a..." que abre una ventana de selección de formato de exportación, dicha ventana

se muestra en la **Figura 4.8**. Desde esa ventana se abre un explorador para guardar el informe generado.

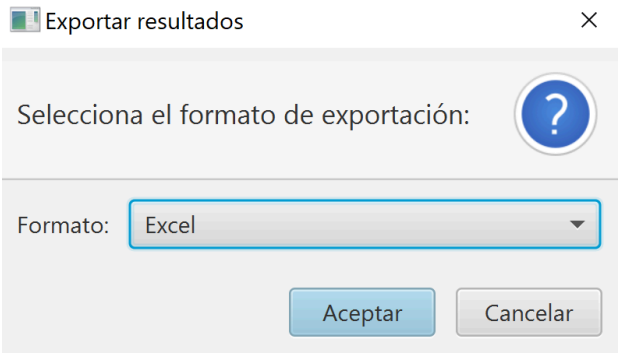


Figura 4.8. Vista de "Exportar a..."

4.4.5 Gestión de estilos

El diseño visual de la interfaz se gestiona de forma centralizada mediante una hoja de estilos externa (estilos.css), lo que permite desacoplar completamente la lógica de la aplicación de los aspectos de presentación. Esta hoja de estilos define las propiedades visuales comunes de los componentes JavaFX utilizados en la aplicación, tales como botones, etiquetas, campos de entrada, listas y contenedores.

El archivo de estilos se carga de manera global en las distintas escenas de la aplicación mediante el método getStylesheets().add(...), garantizando que todas las ventanas compartan una apariencia homogénea sin necesidad de duplicar configuraciones visuales en el código Java.

Este enfoque facilita el mantenimiento del sistema, ya que cualquier modificación estética puede realizarse de forma centralizada, y permite una evolución visual de la aplicación sin afectar a su funcionamiento interno.

4.5 Estilo visual y usabilidad

Uno de los objetivos claves del proyecto ha sido dotar a la herramienta de un estilo visual moderno, limpio y profesional, que facilite su uso y genere una experiencia agradable al usuario, sin necesidad de conocimientos técnicos.

4.5.1 Diseño unificado mediante hojas de estilo CSS

Para garantizar una apariencia coherente en todas las ventanas de la aplicación, se ha creado una hoja de estilos compartida (estilos.css) que define de forma centralizada las propiedades visuales de los componentes JavaFX.

Esta hoja de estilo se aplica a todas las escenas de la aplicación, incluyendo los formularios de definición de reglas, las ventanas emergentes y la interfaz principal.

Entre los estilos definidos destacan:

- Una tipografía elegante y uniforme, común a todas las vistas.
- Botones con esquinas redondeadas, paleta de colores suaves basada en tonos verdes, y efectos visuales de realce al pasar el ratón, recibir el foco o ser pulsados.
- Etiquetas con jerarquías visuales claras, utilizando distintos tamaños de fuente y negritas para títulos y secciones.
- Campos de texto, listas y tablas con bordes suaves, buen espaciado interno y resaltado visual al recibir el foco.
- Un fondo claro y homogéneo en todas las vistas (#f4f7f6), que mejora la legibilidad y evita distracciones visuales.

El uso de CSS permite separar claramente la lógica de presentación del resto del código de la aplicación, facilitando el mantenimiento, la evolución estética y la posible personalización futura por parte de terceros.

4.5.2 Uniformidad en la presentación de resultados

Tanto los mensajes mostrados en la interfaz como los informes generados en Excel han sido unificados bajo un formato común y coherente. Para cada regla ejecutada, los errores aparecen agrupados con:

- Un título identificando la regla y su tipo
- Una lista de errores, cada uno precedido de un símbolo visual (✅ si no hay errores, ❌ en caso de errores)
- Separación clara entre bloques de reglas

Esto proporciona una experiencia más profesional, comprensible y homogénea, tanto para la validación en pantalla como en los archivos exportados.

4.5.3 Experiencia de usuario cuidada

A lo largo del desarrollo, se han tenido en cuenta principios de usabilidad como:

- Diálogos de confirmación para evitar acciones accidentales (por ejemplo, al eliminar reglas)
- Mensajes informativos o de error claros en caso de configuraciones incorrectas, como se observa en la **Figura 4.10**.
- Interacciones intuitivas mediante menús desplegables y campos guiados
- Carga de archivos simple mediante selección múltiple

La curva de aprendizaje de la herramienta es mínima, lo que permite su uso incluso por personas sin experiencia previa en validación estructurada de datos.

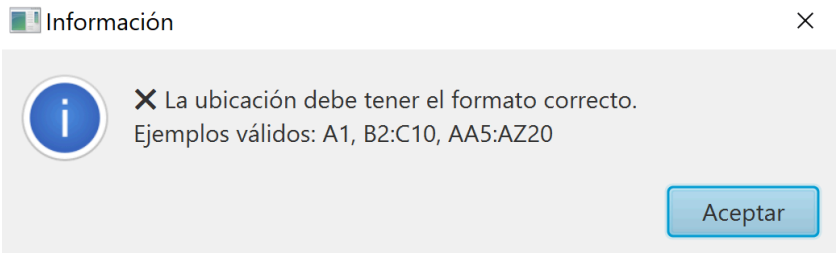


Figura 4.10. Ejemplo de mensaje informativo encontrado en la herramienta

5 Desarrollo de la aplicación

5.1 Carga y visualización de archivos Excel

Uno de los pilares fundamentales de la herramienta es la capacidad de cargar múltiples archivos Excel simultáneamente y mostrar su contenido de forma organizada y comprensible. Esta funcionalidad permite al usuario validar datos dispersos en diferentes libros y hojas de cálculo sin necesidad de combinarlos manualmente, garantizando una base sólida sobre la cual se construyen el resto de funcionalidades de validación.

5.1.1 Carga de archivos

La funcionalidad de carga se activa mediante el botón “Cargar archivo Excel” en la interfaz principal (MainView). Al pulsarlo, se abre un cuadro de diálogo para seleccionar uno o varios ficheros .xlsx.

La clase responsable del procesamiento de estos archivos es CargadorExcel, que utiliza Apache POI para:

- Abrir cada archivo Excel seleccionado.
- Recorrer todas sus hojas.
- Leer sus celdas y almacenar su contenido de forma estructurada.
- Detectar encabezados o valores vacíos en función de la configuración de validación posterior.

Todos los archivos se integran en una única instancia de la clase LibroUnificado, que actúa como modelo de datos centralizado.

5.1.2 Estructura del modelo unificado

LibroUnificado proporciona un acceso homogéneo a toda la información cargada, sin importar si proviene de un único fichero o de varios. Internamente mantiene un mapa que relaciona cada nombre de archivo con las hojas contenidas en él, permitiendo acceder fácilmente a:

- Celdas individuales por coordenadas (archivo, hoja, fila, columna).
- Rango de celdas.
- Columnas completas o encabezados.

Este diseño facilita la posterior aplicación de reglas de validación, ya que todas ellas operan sobre una estructura común y coherente.

5.1.3 Visualización en la interfaz

Una vez cargados los archivos, la interfaz muestra su contenido en un componente TreeView, donde se representan de forma jerárquica:

📄 Hoja2

📁 archivo2.xlsx

📄 Hoja1

Esta estructura se actualiza dinámicamente y permite al usuario:

- Verificar que los archivos se han leído correctamente.
- Confirmar el nombre de las hojas disponibles.
- Saber si los datos deseados están accesibles para las reglas.

En la **Figura 5.1** se muestra un ejemplo de esta representación en la herramienta.

Figura 5.1. Vista de ventana principal con tres archivos Excel cargados

5.2 Estructura de reglas de validación

La funcionalidad principal de la herramienta gira en torno a la definición y ejecución de reglas de validación. Cada regla representa una comprobación lógica que se aplica sobre una o varias celdas del modelo de datos (LibroUnificado) para detectar errores, inconsistencias o condiciones incumplidas.

Las reglas comparten una arquitectura común:

- Una clase que representa la configuración de la regla.
- Una clase que se encarga de validarla.
- Una vista específica que permite al usuario definirla mediante una interfaz visual.

Esto permite desacoplar claramente la lógica de validación, la configuración del usuario y la representación visual.

5.2.1 Valor obligatorio

La regla Valor obligatorio permite comprobar que determinadas ubicaciones dentro de un archivo Excel no estén vacías. Es una de las validaciones más habituales en cualquier proceso de verificación de datos, especialmente en campos críticos que deben ser rellenados por los usuarios.

➢ Clase de configuración: ReglaValorObligatorio

Esta clase extiende de Regla e incluye información sobre la ubicación a validar: puede ser una celda concreta, una columna completa o un rango definido por el usuario. También se almacena el identificador del archivo y hoja correspondiente.

➢ Vista asociada: ReglaValorObligatorioView

Esta vista permite al usuario seleccionar de forma visual el archivo, la hoja y la ubicación donde se desea que exista un valor obligatorio. Incluye controles desplegables y validación interactiva de la selección.

➢ Validador: ValidadorValorObligatorio

Esta clase recibe el LibroUnificado y la instancia de ReglaValorObligatorio, y recorre las celdas especificadas verificando que no estén vacías. Devuelve una lista de errores con la celda exacta que no cumple la regla.

Los errores se generan únicamente cuando hay celdas vacías en el rango evaluado.

➔ Ejemplo de uso: El usuario carga un archivo con una hoja que contiene una columna “Email”. Define una regla de valor obligatorio sobre esa columna, indicando que todas las celdas deben contener un dato. Si alguna está vacía, se registrará un error en la validación.

Esta regla proporciona una base esencial para garantizar la completitud de los datos y prevenir errores en cascada derivados de información faltante.

5.2.2 Duplicados

La regla de Duplicados permite detectar valores repetidos en una celda, columna o rango de celdas donde se espera que los datos sean únicos. Este tipo de validación es especialmente relevante en identificadores, códigos, claves primarias o cualquier campo que deba cumplir una condición de unicidad.

➢ Clase de configuración: ReglaDuplicados

Esta clase define la ubicación sobre la que se aplicará la validación de unicidad, junto con el archivo y la hoja correspondientes. La ubicación puede ser una columna completa, una celda concreta o un rango de celdas definido por el usuario.

➢ Vista asociada: ReglaDuplicadosView

La vista permite seleccionar de forma visual el archivo, la hoja y la ubicación a validar. Incluye validaciones para asegurar que la ubicación introducida tenga un formato correcto antes de guardar la regla.

➢ Validador: ValidadorDuplicados

El validador recorre las celdas especificadas y almacena los valores encontrados en una estructura auxiliar, detectando aquellos que aparecen más de una vez. Para cada repetición se genera un error detallando el valor duplicado y su localización.

Los errores se generan únicamente cuando hay valores duplicados en el rango evaluado.

➔ Ejemplo de uso: En un archivo Excel que contiene una hoja de clientes, el usuario define una regla de duplicados sobre la columna “ID_Cliente”. Durante la validación, si dos o más filas contienen el mismo identificador, la regla se considera incumplida.

La inclusión de esta regla permite garantizar la consistencia y unicidad de los datos, evitando problemas posteriores en procesos que dependan de claves únicas, como integraciones, migraciones de datos o análisis posteriores.

5.2.3 Tipo de dato

La regla de Tipo de dato permite verificar que los valores contenidos en una ubicación específica (columna, celda o rango) cumplan con un tipo esperado, como texto, número o fecha. Esta validación es esencial para asegurar la coherencia semántica de los datos, prevenir errores en cálculos y facilitar posteriores tratamientos automáticos de la información.

➢ Clase de configuración: ReglaTipoDato

Almacena la ubicación a validar, el tipo de dato esperado (texto, número o fecha) y una opción para permitir o no celdas vacías. Esta última opción amplía la flexibilidad de la regla en contextos donde hay datos opcionales.

➢ Vista asociada: ReglaTipoDatoView

El formulario permite al usuario seleccionar la ubicación (celda, columna o rango) y elegir el tipo de dato requerido mediante un desplegable. También se incluye un checkbox para indicar si las celdas vacías deben ser consideradas como error.

➢ Validador: ValidadorTipoDato

El validador recorre las celdas indicadas, identifica su contenido real y compara su tipo con el esperado. Se utilizan funciones auxiliares para detectar si una celda contiene un número, una fecha o una cadena, considerando tanto el tipo nativo como el formato visible en Excel. Si se permite el uso de celdas vacías, aquellas posiciones en las que uno de los valores esté vacío se ignoran.

Los errores se generan únicamente cuando una celda del rango seleccionado no contiene el tipo de dato esperado o está vacía, en caso de no permitirse celdas vacías.

➔ Ejemplo de uso: En una hoja con fechas de vencimiento, el usuario configura una regla de tipo fecha sobre la columna "FechaPago". El sistema validará que cada celda contenga efectivamente una fecha reconocible y no una cadena de texto o cualquier otro tipo de dato.

Esta regla permite detectar inconsistencias silenciosas que podrían pasar desapercibidas visualmente en Excel, pero que pueden provocar fallos en procesos posteriores como operaciones matemáticas, filtros o conversiones de formatos.

5.2.4 Referencias cruzadas

La regla de Referencias cruzadas permite verificar que los valores existentes en una ubicación específica de una hoja estén también presentes en otra hoja o archivo Excel cargado. Este tipo de validación es muy útil para comprobar integridad relacional entre listas, como claves primarias y foráneas en bases de datos, o referencias entre catálogos y transacciones.

➢ Clase de configuración: ReglaReferenciasCruzadas

Define dos ubicaciones: origen (donde están los valores que deben existir en la otra hoja) y referencia (la ubicación donde se espera que aparezcan esos valores). También permite configurar si la verificación será unidireccional (origen debe estar en referencia) o bidireccional (ambas ubicaciones deben contener los mismos valores).

➢ Vista asociada: ReglaReferenciasCruzadasView

Este formulario permite al usuario seleccionar las dos ubicaciones desde listas desplegables y marcar si la validación debe ser bidireccional. La interfaz guía al usuario en la selección de hojas, columnas, celdas o rangos involucrados.

➢ Validador: ValidadorReferenciasCruzadas

Compara los valores extraídos de ambas ubicaciones, detectando elementos que aparecen en una pero no en la otra. En caso de verificación bidireccional, la comparación se realiza en ambos sentidos.

Los errores se generan únicamente si los valores del origen no están contenidos en la referencia y, en caso de ser una validación bidireccional, si los valores de referencia no están contenidos en el origen.

➔ Ejemplo de uso: Si en la hoja "Pedidos" hay una columna “IDProducto” y en la hoja “Inventario” una columna con los productos disponibles, se puede aplicar una regla que asegure que todos los ID listados en pedidos existan en el inventario.

Este tipo de regla añade robustez a la validación, al permitir detectar errores de consistencia entre hojas que simulan relaciones de datos. Es especialmente útil en contextos administrativos, contables o de seguimiento de inventario, donde los errores de referencia pueden tener consecuencias importantes.

5.2.5 Comparación

La regla de Comparación permite validar relaciones directas entre valores situados en distintas celdas, columnas o rangos de una o varias hojas, utilizando operadores de comparación lógicos y relacionales como igualdad, diferencia o comparación numérica. Se trata de una regla flexible que permite expresar validaciones tanto aritméticas como de igualdad, sin necesidad de establecer dependencias condicionales entre los datos.

Esta regla resulta especialmente útil para comprobar la coherencia entre valores relacionados, como importes, cantidades o estados, cuando la validación no depende de un contexto previo, sino que debe cumplirse de forma directa.

➢ Clase de configuración: ReglaComparacion

Define la estructura de la regla y permite configurar una ubicación de origen, que será validada frente a un valor de comparación mediante un operador lógico. La comparación puede realizarse de dos formas:

> ➔ Comparación con valor fijo: Todos los valores de la ubicación de origen se comparan con un único valor constante (numérico o textual).
>
> ➔ Comparación con otra ubicación: Cada celda de la ubicación de origen se compara con la celda correspondiente de otra ubicación del libro, permitiendo validar relaciones entre columnas o rangos, incluso entre hojas distintas.

En ambos casos, el usuario puede seleccionar el operador de comparación entre los siguientes: igual (=), distinto (≠), mayor (>), menor (<), mayor o igual (≥), menor o igual (≤).

La regla incluye además la opción “permitir celdas vacías”, que define el comportamiento del validador ante valores ausentes. Si no se permiten celdas vacías, cualquier celda vacía en la comparación genera un error. Si se permiten celdas vacías, dichas celdas se ignoran y no se someten a validación.

➢ Vista asociada: ReglaComparacionView

Proporciona una interfaz gráfica que permite definir la regla de forma clara y guiada. A través de esta vista, el usuario puede seleccionar el archivo, hoja y ubicación de origen, elegir el operador de comparación, decidir si la comparación se realiza contra un valor fijo o contra otra ubicación del libro e indicar si se permiten o no celdas vacías durante la validación.

La interfaz adapta dinámicamente los campos visibles según el tipo de comparación seleccionado y realiza validaciones previas para asegurar que las ubicaciones introducidas tienen un formato correcto antes de guardar la regla.

➢ Validador: ValidadorComparacion

Es el encargado de aplicar la lógica de la regla sobre los datos del libro unificado. Obtiene todas las celdas de la ubicación de origen y sus valores asociados; si la comparación se realiza contra otra ubicación, ambas ubicaciones deben contener el mismo número de celdas. En caso contrario, la validación se detiene y se genera un único error indicando que las ubicaciones comparadas tienen diferente tamaño. La validación se realiza celda a celda, comparando cada valor de origen con su correspondiente valor de comparación. Si se permite el uso de celdas vacías, aquellas posiciones en las que uno de los valores esté vacío se ignoran.

Los errores se generan únicamente cuando la comparación definida no se cumple para una celda concreta, las ubicaciones comparadas son de diferente tamaño o se detecta una celda vacía en una posición donde no está permitida.

➔ Ejemplo de uso: Comparar si los valores de la columna “Importe pagado” de la hoja “Pagos” son menores o iguales que los valores de la columna “Importe esperado” de la hoja “Facturas”.

Esta regla proporciona una forma directa y potente de validar relaciones entre datos relacionados, sin necesidad de condiciones previas. Gracias a su capacidad para comparar rangos completos, valores fijos o ubicaciones cruzadas entre hojas, resulta especialmente útil para validar coherencias numéricas y lógicas en documentos Excel.

5.2.6 Condicional

La regla Condicional permite establecer relaciones del tipo “SI… ENTONCES…”, donde el cumplimiento de una condición en una celda o conjunto de celdas implica que otra condición debe cumplirse en una ubicación diferente. Esta regla posibilita la definición de dependencias lógicas entre datos, incluso cuando estos se encuentran en hojas o columnas distintas, lo que la convierte en una herramienta especialmente útil para validar coherencias contextuales dentro de ficheros Excel.

A diferencia de las reglas simples, la regla condicional no valida un valor de forma aislada, sino que evalúa relaciones entre datos, permitiendo expresar validaciones del tipo “solo si ocurre X, entonces debe cumplirse Y”.

➢ Clase de configuración: ReglaCondicional

Define la estructura completa de la regla y distingue claramente dos bloques independientes:

➔ Condición (SI…)

- Define una ubicación origen (celda, columna o rango), un operador de comparación y un elemento con el que comparar.
- La comparación puede realizarse contra un valor fijo (texto o número) o contra otra ubicación del libro (celda, columna o rango), permitiendo comparaciones entre hojas o columnas distintas.

➔ Acción (ENTONCES…)

- Define la validación que debe cumplirse cuando la condición anterior se satisface.
- Al igual que en la condición, la acción dispone de una ubicación a validar, un operador y un valor de comparación, que puede ser fijo o provenir de otra ubicación del libro.

Ambos bloques funcionan de manera simétrica y permiten construir reglas condicionales complejas sin limitarse a valores constantes.

Adicionalmente, la regla incluye la opción “permitir celdas vacías”, que determina el comportamiento de la validación cuando las celdas de la acción no contienen ningún valor:

- Si no se permiten celdas vacías, una celda vacía en la acción genera un error cuando la condición se cumple.
- Si se permiten celdas vacías, dichas celdas se ignoran y no se someten a validación.

➢ Vista asociada: ReglaCondicionalView

Proporciona una interfaz gráfica que permite definir la regla de forma visual e intuitiva. El usuario puede configurar de manera independiente los bloques de condición y acción, especificando para cada uno: archivo y hoja, ubicación (celda, columna o rango), operador de comparación y tipo de comparación: con valor fijo o con otra ubicación del libro.

La interfaz también permite activar o desactivar la opción de permitir celdas vacías, facilitando la adaptación de la regla a distintos contextos de validación.

Se incluyen validaciones previas para asegurar que las ubicaciones introducidas tienen un formato correcto y que se han proporcionado todos los datos necesarios antes de guardar la regla.

➢ Validador: ValidadorCondicional

Es el encargado de aplicar la lógica de la regla sobre los datos del libro unificado. Los rangos definidos en la condición, la comparación de la condición, la acción y la comparación de la acción se recorren en paralelo, alineando sus valores por índice. Para cada posición (fila lógica), se evalúa primero la condición y, si la condición se cumple, se valida la acción correspondiente en esa misma posición. Si la condición no se cumple, la acción no se evalúa y no se genera ningún error.

Es importante destacar que la cantidad de valores evaluados en cada bloque debe ser coherente, ya que la validación se realiza fila a fila, asumiendo que

los rangos de la condición y de la acción representan relaciones lógicas entre elementos de la misma posición relativa.

Los errores únicamente se generan cuando la condición se cumple y la acción no satisface la comparación definida o la condición se cumple, la celda de la acción está vacía y no se permiten celdas vacías.

➔ Ejemplo de uso: SI, en una determinada fila, el valor de la columna Estado es IGUAL a "ACTIVO", ENTONCES el valor de la columna FechaBaja debe ser MENOR QUE la fecha actual.

La regla condicional permite validar relaciones lógicas complejas que dependen del contexto. Supone un paso más allá de las validaciones directas, permitiendo adaptar reglas a necesidades del tipo "en caso de que..." o "solo si...", muy comunes en documentos Excel usados en procesos administrativos, educativos o empresariales.

5.3 Validación y gestión de errores

Una vez definidas las reglas por el usuario, la herramienta permite ejecutar la validación completa de los archivos Excel cargados, aplicando una por una todas las reglas activas. Esta fase constituye el núcleo del proceso, y ha sido diseñada para garantizar resultados fiables, comprensibles y útiles para el usuario final.

5.3.1 Ejecución de validaciones

La ejecución se inicia desde la interfaz mediante el botón “Validar”, el cual lanza un proceso que:

- Recorre todas las reglas almacenadas en MemoriaReglas.
- Para cada regla, invoca su método validar(...), pasándole como parámetro el objeto LibroUnificado que representa todos los archivos y hojas cargados.
- Cada regla devuelve un objeto ResultadoRegla, que contiene una lista de errores detectados (si los hay).
- Todos los resultados son recogidos en un mapa y enviados a la clase VentanaResultadosValidacion para su visualización.

Este proceso se realiza de forma secuencial, sin hilos paralelos, dado que el número de reglas y volumen de datos manejado en un entorno habitual no requiere paralelización.

5.3.2 Registro de errores

Los errores generados por cada regla se recogen en listas de tipo List<String> dentro del objeto ResultadoRegla. Cada mensaje de error incluye:

- El tipo de error (mediante símbolo ❌ o ✅).

- La ubicación concreta del dato inválido (nombre de archivo, hoja, celda).
- Una descripción comprensible del fallo detectado.

5.3.3 Visualización de resultados

Los resultados se muestran en una ventana específica (VentanaResultadosValidacion), donde:

- Cada regla validada aparece como un bloque separado.
- Se muestran los errores detectados en forma de lista, con buena separación visual y legibilidad.
- En caso de no haber errores en una regla, se informa claramente con un mensaje de validación superada.

Esta presentación facilita una revisión rápida, permite identificar fácilmente los errores más críticos y guía al usuario hacia las celdas que necesitan ser corregidas.

5.3.4 Consistencia en el formato

Una tarea importante ha sido la uniformización del formato de presentación de errores, tanto en pantalla como en la exportación Excel. Esto garantiza que los resultados se perciban como parte de un sistema coherente, reforzando la confianza del usuario.

5.4 Exportación de resultados

Una de las funcionalidades más valoradas de la herramienta es la posibilidad de exportar los resultados de la validación a un archivo Excel, lo que permite al usuario conservar un registro detallado de los errores detectados, compartirlo con terceros o revisarlo posteriormente.

5.4.1 Mecanismo de exportación

Desde la interfaz de VentanaResultadosValidacion, una vez mostrados los resultados en pantalla, se ofrece un botón que permite al usuario exportarlos. Al hacer clic, se abre un explorador de archivos estándar donde el usuario puede seleccionar la ubicación y el nombre del archivo .xlsx que desea generar.

La exportación es gestionada por la clase ExportadorResultados, que:

- Crea un nuevo libro Excel (Workbook) utilizando la librería Apache POI.
- Añade una única hoja de resultados con un título general.
- Para cada regla validada:
    - Inserta un título de sección con el nombre y descripción de la regla.

    - Si hubo errores, los añade como una lista de celdas con formato de error.
    - Si no hubo errores, lo hace saber con un mensaje.
- Aplica estilos básicos (negrita, separadores, espaciado) para una lectura clara y profesional.

El archivo se guarda automáticamente en la ruta especificada, y puede abrirse en cualquier versión moderna de Excel.

5.4.2 Estructura del informe

El informe generado tiene un diseño limpio y uniforme, como se observa en la **Figura 5.2**.

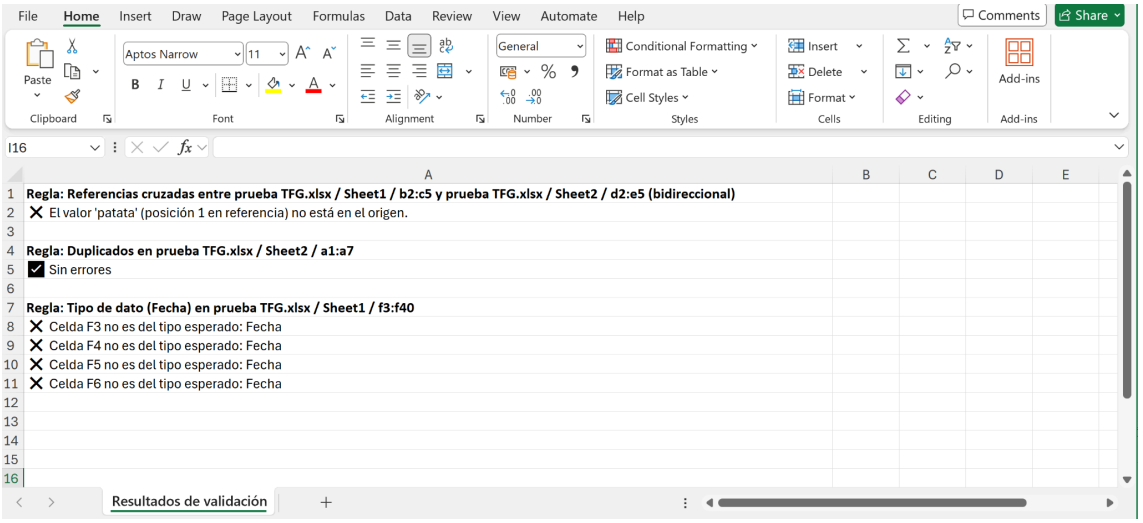


Figura 5.2. Informe de validación generado y exportado con la herramienta

Esta presentación facilita el seguimiento de errores por parte del usuario o su equipo, evitando depender exclusivamente de la interfaz en tiempo de ejecución.

5.4.3 Coherencia con la interfaz

Una de las prioridades en esta funcionalidad ha sido mantener la coherencia visual y estructural entre lo mostrado en pantalla y lo exportado. De este modo, el usuario no encuentra diferencias entre los mensajes leídos en la aplicación y los volcados al fichero.

Esto se logró reutilizando parte de la lógica de formato común y estableciendo una política clara de símbolos, encabezados por regla, y separación por bloques.

5.5 Organización del código y buenas prácticas

Durante el desarrollo de la herramienta, se ha seguido una estructura modular clara y una serie de buenas prácticas de programación que han contribuido a la escalabilidad, mantenibilidad y legibilidad del código.

5.5.1 Organización por paquetes

Durante el desarrollo del proyecto se ha optado por una organización del código basada en la separación de responsabilidades, estructurando las clases en paquetes lógicos. Esta decisión no solo responde a una cuestión de orden visual, sino a principios fundamentales de la ingeniería del software como la mantenibilidad, extensibilidad y testabilidad.

El sistema se ha diseñado siguiendo un enfoque modular donde cada paquete encapsula una función específica del sistema. A diferencia de una estructura monolítica, esta organización permite:

➢ Desacoplamiento funcional: los distintos componentes (interfaz gráfica, lógica de negocio, modelo de datos, validadores, utilidades) interactúan entre sí mediante interfaces bien definidas, lo que evita dependencias innecesarias y facilita la evolución del sistema.

➢ Facilidad de pruebas: al estar las reglas de validación y su lógica separadas de la interfaz gráfica, es posible probar los validadores de forma independiente con datos simulados.

➢ Escalabilidad: añadir una nueva regla de validación implica simplemente definir una nueva clase Regla y su correspondiente Validador, sin afectar al resto de componentes.

➢ Mantenibilidad: ante posibles errores o refactorizaciones, la localización del código afectado es inmediata gracias a la clara separación en paquetes.

Además, esta organización por paquetes refleja de forma natural el patrón arquitectónico de tres capas (presentación, lógica de negocio, modelo), complementado con un paquete adicional de utilidades reutilizables.

Por tanto, la organización interna del código no es meramente estética, sino que responde a una estrategia de diseño que mejora la calidad global del software y su preparación para futuras ampliaciones.

5.5.2 Buenas prácticas aplicadas

Durante el desarrollo se han seguido múltiples buenas prácticas de programación orientada a objetos:

➢ Encapsulamiento y modularidad: Cada componente tiene una función concreta y está desacoplado del resto. Las reglas no dependen de la interfaz, y los validadores son independientes del modelo de datos.

➢ Principio DRY ("Don't Repeat Yourself"): Se han evitado duplicaciones innecesarias mediante el uso de clases auxiliares, utilidades comunes y estructuras reutilizables.

➢ Legibilidad del código: Se han seguido convenciones de nombres claras, con clases, métodos y variables descriptivas y coherentes.

➢ Comentarios explicativos: Se han añadido comentarios breves en métodos clave para facilitar la comprensión del código.

➢ Estilo visual separado: Todos los estilos de la interfaz se gestionan mediante una hoja CSS externa, separando completamente el diseño visual del código funcional.

➢ Extensibilidad: La arquitectura permite incorporar nuevas reglas de validación con facilidad, simplemente añadiendo una nueva clase que implemente Regla, su formulario visual y su Validador.

6 Pruebas y validación

6.1 Estrategia de pruebas

Durante el desarrollo de la herramienta se ha seguido una estrategia de pruebas empírica y estructurada, basada principalmente en pruebas funcionales manuales, ya que el enfoque del proyecto es práctico y orientado a la interacción visual con el usuario final.

Dado que el sistema está centrado en la validación de datos sobre archivos Excel, y que muchas operaciones dependen directamente de la interacción gráfica, se ha optado por una estrategia en la que:

- Se han definido escenarios de prueba concretos para cada funcionalidad.
- Se han usado archivos Excel reales o de prueba, construidos específicamente para verificar comportamientos esperados.
- Se ha verificado tanto la correcta ejecución de las validaciones, como la presentación coherente de los errores en la interfaz y el informe exportado.
- Se han realizado múltiples iteraciones para ajustar las interfaces gráficas, los mensajes informativos y la estructura del código ante errores o situaciones imprevistas.

6.1.1 Tipos de pruebas realizadas

Se han aplicado principalmente las siguientes estrategias:

➢ Pruebas de caja negra: para validar que cada regla funcione correctamente sin conocer su implementación interna; se han introducido entradas conocidas y se ha verificado la salida esperada.

➢ Pruebas de integración: especialmente al cargar múltiples archivos Excel, aplicando varias reglas simultáneamente y verificando que no haya conflictos o errores cruzados.

➢ Pruebas exploratorias: durante el uso interactivo de la herramienta, detectando posibles errores de uso, comportamiento inesperado o mejoras de usabilidad.

➢ Pruebas de regresión: tras añadir nuevas funcionalidades (por ejemplo, nuevas reglas), se ha vuelto a probar el sistema completo para asegurar que el funcionamiento anterior no se ha visto afectado.

6.1.2 Cobertura de pruebas

Las pruebas han cubierto los siguientes aspectos clave del sistema:

- Carga y lectura de archivos Excel (correctos y corruptos).
- Definición individual de reglas y comprobación de su correcto almacenamiento en memoria.

- Ejemplo con múltiples reglas simultáneas, actuando sobre distintos archivos y hojas.
- Validación de errores deliberados, para asegurar su correcta detección y notificación.
- Exportación de resultados, verificando el contenido del archivo Excel generado.
- Resistencia a errores de usuario, como dejar campos vacíos, seleccionar celdas no válidas, eliminar reglas o intentar validar sin reglas definidas.

6.1.3 Herramientas utilizadas

Las pruebas se han llevado a cabo utilizando los siguientes elementos:

- Archivos Excel de prueba personalizados, con distintos patrones de datos para evaluar cada tipo de regla.
- La propia interfaz gráfica de la herramienta como entorno de pruebas, replicando acciones reales de usuarios finales.
- Mensajes de consola y alertas en pantalla para comprobar el correcto flujo de ejecución y control de errores.

6.2 Casos de prueba

Para verificar el correcto funcionamiento de la herramienta, se han utilizado ficheros Excel reales ya existentes en las carpetas del autor y, adicionalmente, se ha creado un nuevo archivo Excel de prueba con varias hojas y datos intencionalmente diseñados para activar las distintas reglas de validación (véase Anexo B.1). El nombre del archivo es “Negocio” y se ha añadido a la herramienta previo a la realización de los siguientes casos de prueba (véase Anexo B.2). A continuación se detallan los casos de prueba utilizados en dicho archivo que abarcan diferentes casos y modos de uso para cada regla:

6.2.1 Regla: Valor obligatorio

Número de prueba	1
Ubicación validada	Celdas B2 a B8 (columna Email) de la hoja “Clientes” del archivo “Negocio”.
Caso de prueba	Se han dejado celdas vacías.
Resultado esperado	Se detectan como errores las cuatro celdas sin datos.
Resultado	Correcto ✅ (véase Anexo C.1).

Número de prueba	2
Ubicación validada	Celdas A2 a A5 (columna ID_Cliente) de la hoja “Clientes” del archivo “Negocio”.
Caso de prueba	Sin errores.
Resultado esperado	No se detectan errores.
Resultado	Correcto ✅ (véase Anexo C.1).

6.2.2 Regla: Duplicados

Número de prueba	1
Ubicación validada	Celdas D2 a D5 (columnas Fecha de nacimiento y Tipo cliente) de la hoja “Clientes” del archivo “Negocio”.
Caso de prueba	Se repitió intencionadamente el valor “Empresa” en tres celdas de la columna “Tipo de cliente”.
Resultado esperado	Se detecta como error la presencia de un valor duplicado.
Resultado	Correcto ✅ (véase Anexo C.2).

Número de prueba	2
Ubicación validada	Celdas E2 a E10 (columna CIF) de la hoja “Clientes” del archivo “Negocio”.
Caso de prueba	Sin errores.

Resultado esperado	No se detectan errores.
Resultado	Correcto ✅ (véase Anexo C.2).

6.2.3 Regla: Tipo de dato

Número de prueba	1
Ubicación validada	Celdas C2 a C5 (columna Fecha de nacimiento) de la hoja “Clientes” del archivo “Negocio”.
Condición	Todos los valores deben ser de tipo fecha. No se permiten celdas vacías.
Caso de prueba	Se colocaron valores de otro tipo como “Hola” o “32/15/2023” (fecha inválida).
Resultado esperado	Se detectan como errores los dos valores no válidos como fechas.
Resultado	Correcto ✅ (véase Anexo C.3).

Número de prueba	2
Ubicación validada	Celdas D2 a D6 (columna Tipo cliente) de la hoja “Clientes” del archivo “Negocio”.
Condición	Todos los valores deben ser de tipo texto. No se permiten celdas vacías.
Caso de prueba	Se ha dejado una celda vacía.
Resultado esperado	Se detecta como error la celda vacía.
Resultado	Correcto ✅ (véase Anexo C.3).

Número de prueba	3
Ubicación validada	Celdas A2 a A11 (columna ID_Cliente) de la hoja “Clientes” del archivo “Negocio”.
Condición	Todos los valores deben ser de tipo número. Se permiten celdas vacías.
Caso de prueba	Sin errores.
Resultado esperado	No se detectan errores.
Resultado	Correcto ✅ (véase Anexo C.3).

6.2.4 Regla: Referencias cruzadas

Número de prueba	1
Ubicación origen	Celdas B2 a B5 (columna ID_Producto) de la hoja “Facturas” del archivo “Negocio”.
Ubicación referencia	Celdas A2 a A3 (columna ID_Producto) de la hoja “Inventario” del archivo “Negocio”.
Condición	Validación no bidireccional.
Caso de prueba	Se ha añadido un producto con ID inexistente en la hoja “Facturas”.
Resultado esperado	Se detecta como error el valor no referenciado.
Resultado	Correcto ✅ (véase Anexo C.4).

Número de prueba	2

Ubicación origen	Celdas A2 a A3 (columna ID_Producto) de la hoja “Inventario” del archivo “Negocio”.
Ubicación referencia	Celdas B2 a B5 (columna ID_Producto) de la hoja “Facturas” del archivo “Negocio”.
Condición	Validación no bidireccional.
Caso de prueba	Sin errores.
Resultado esperado	No se detectan errores.
Resultado	Correcto ✅ (véase Anexo C.4).

Número de prueba	3
Ubicación origen	Celdas A2 a A3 (columna ID_Producto) de la hoja “Inventario” del archivo “Negocio”.
Ubicación referencia	Celdas B2 a B5 (columna ID_Producto) de la hoja “Facturas” del archivo “Negocio”.
Condición	Validación bidireccional.
Caso de prueba	Al ser validación bidireccional se comprueba que todos los valores de referencia también se encuentren en el origen; no es el caso para el producto con ID inexistente en el inventario.
Resultado esperado	Se detecta como error el valor no referenciado.
Resultado	Correcto ✅ (véase Anexo C.4).

6.2.5 Regla: Comparación

Número de prueba	1
Ubicación origen	Celdas C2 a C5 (columna Importe) de la hoja “Facturas” del archivo “Negocio”.
Ubicación destino	500 (Valor fijo).
Condición	Ubicación origen MAYOR (>) que 500. No se permiten celdas vacías.
Caso de prueba	Se han introducido importes iguales e inferiores a 500.
Resultado esperado	Se detectan como errores los dos valores no superiores a 500.
Resultado	Correcto ✅ (véase Anexo C.5).

Número de prueba	2
Ubicación origen	Celdas C2 a C3 (columna Stock) de la hoja “Inventario” del archivo “Negocio”.
Ubicación destino	Celdas C2 a C5 (columna Importe) de la hoja “Facturas” del archivo “Negocio”.
Condición	Ubicación origen MENOR (<) que Ubicación destino. No se permiten celdas vacías.
Caso de prueba	Se han comparado dos ubicaciones con distinta cantidad de celdas.
Resultado esperado	Se detecta como error la comparación entre ubicaciones con diferente número de celdas.
Resultado	Correcto ✅ (véase Anexo C.5).

Número de prueba	3
Ubicación origen	Celdas A2 a A3 (columna ID_Producto) de la hoja “Inventario” del archivo “Negocio”.
Ubicación destino	Celdas B2 a B3 (columna ID_Producto) de la hoja “Facturas” del archivo “Negocio”.
Condición	Ubicación origen IGUAL (=) que Ubicación destino. No se permiten celdas vacías.
Caso de prueba	Sin errores.
Resultado esperado	No se detectan errores.
Resultado	Correcto ✅ (véase Anexo C.5).

6.2.6 Regla: Condicional

Número de prueba	1
Condición (SI…)	Celdas D2 a D5 (columna Tipo cliente) de la hoja “Clientes” del archivo “Negocio”.
Comparador condición	Empresa (Valor fijo).
Acción (ENTONCES…)	Celdas E2 a E5 (columna CIF) de la hoja “Clientes” del archivo “Negocio”.
Comparador acción	B12345678 (Valor fijo).
Condición	SI el valor de la columna Tipo cliente es IGUAL (=) a "Empresa", ENTONCES el valor de la columna CIF debe ser DIFERENTE (≠) de "B12345678". Se permiten celdas vacías.

Caso de prueba	Se ha definido una fila con Tipo cliente = "Empresa" y el campo CIF B12345678.
Resultado esperado	Se detecta error en la fila en la que, con la condición cumplida, la comparación de la acción no se satisface.
Resultado	Resultado: Correcto ✅ (véase Anexo C.6).

Número de prueba	2
Condición (SI…)	Celdas D2 a D5 (columna Tipo cliente) de la hoja “Clientes” del archivo “Negocio”.
Comparador condición	Celdas C2 a C5 (columna Fecha de nacimiento) de la hoja “Clientes” del archivo “Negocio”.
Acción (ENTONCES…)	Celdas E2 a E5 (columna CIF) de la hoja “Clientes” del archivo “Negocio”.
Comparador acción	Celdas C2 a C5 (columna Fecha de nacimiento) de la hoja “Clientes” del archivo “Negocio”.
Condición	SI los valores de la columna Tipo cliente son DIFERENTES (≠) a los valores de la columna Fecha de nacimiento, ENTONCES los valores de la columna CIF deben ser DIFERENTES (≠) de los valores de la columna Fecha de nacimiento. Se permiten celdas vacías.
Caso de prueba	Sin errores.
Resultado esperado	No se detectan errores.
Resultado	Correcto ✅ (véase Anexo C.6).

Número de prueba	3
Condición (SI…)	Celdas C3 a C4 (columna Importe) de la hoja “Facturas” del archivo “Negocio”.
Comparador condición	Celdas C2 a C3 (columna Stock) de la hoja “Inventario” del archivo “Negocio”.
Acción (ENTONCES…)	Celdas B2 a B6 (columna Email) de la hoja “Clientes” del archivo “Negocio”.
Comparador acción	@mail.com (Valor fijo).
Condición	SI los valores de la columna Importe son MAYORES (>) que los valores de la columna Stock, ENTONCES los valores de la columna Email deben CONTENER (⊃) a "@mail.com". No se permiten celdas vacías.
Caso de prueba	Se han dejado dos celdas vacías en el rango seleccionado de la columna Email.
Resultado esperado	Se detecta como error una fila vacía en la acción. La condición se cumple pero valida únicamente dos filas, por lo cual, en la acción únicamente serán validadas las dos primeras filas del rango de celdas ingresado.
Resultado	Correcto ✅ (véase Anexo C.6).

6.3 Pruebas de interfaz y usabilidad

Además de las pruebas funcionales, se han realizado pruebas centradas en la experiencia de usuario para evaluar la usabilidad, claridad visual y fluidez del flujo de trabajo de la herramienta.

6.3.1 Pruebas informales con usuarios

Para este propósito, se ha contado con la colaboración de varias personas (algunas con perfil técnico y otras sin experiencia en validación de datos), a quienes se les ha pedido que utilicen la herramienta con un archivo Excel de ejemplo y que realicen tareas como:

➔ Cargar varios archivos Excel simultáneamente.

➔ Definir una o varias reglas (valor obligatorio, tipo de dato, condicional, etc.).

➔ Ejecutar la validación.

➔ Revisar los errores detectados.

➔ Exportar los resultados.

Estas pruebas permitieron identificar aspectos clave sobre la experiencia de uso:

➢ Comprensión inmediata del funcionamiento: todos los participantes fueron capaces de comprender rápidamente qué hacía cada botón y cómo avanzar en el flujo de trabajo sin necesidad de explicación adicional.

➢ Interfaz clara y coherente: la distribución de botones, formularios y vistas fue considerada intuitiva, y el diseño visual agradable; cada vista tiene una única responsabilidad visual, lo que reduce la carga cognitiva del usuario y minimiza el riesgo de errores durante su uso.

➢ Mensajes útiles: los textos de ayuda incluidos en los formularios (por ejemplo, el Label info) fueron percibidos como muy útiles para saber si se había completado correctamente una regla o si faltaba algún parámetro.

➢ Eliminación de reglas sencilla: la posibilidad de eliminar reglas individuales desde la vista de reglas definidas fue bien valorada por su facilidad.

➢ Exportación comprensible: el archivo Excel de resultados fue fácilmente comprensible incluso para personas no técnicas, gracias a la estructura común y uso de iconos visuales.

6.3.2 Curva de aprendizaje y autonomía

Un aspecto clave validado fue que la curva de aprendizaje es mínima. En general, los usuarios sin experiencia previa en validación de datos pudieron completar el flujo completo (cargar, definir reglas, validar y exportar) sin ayuda directa.

Esto demuestra que la herramienta cumple con uno de sus objetivos más importantes: ser accesible y útil para usuarios sin conocimientos técnicos, facilitando su adopción en entornos académicos, administrativos o profesionales.

7 Resultados y conclusiones

7.1 Valoración de los objetivos alcanzados

Al finalizar el desarrollo de la herramienta, se puede realizar una valoración muy positiva del grado de cumplimiento de los objetivos definidos al inicio del proyecto. Todos los objetivos funcionales han sido alcanzados con éxito, y la aplicación resultante cumple plenamente con los requisitos establecidos.

☑ Funcionalidad completa de validación

La herramienta permite cargar múltiples archivos Excel y visualizar su estructura interna (libros y hojas) de forma clara y organizada. El sistema permite definir reglas de validación que abarcan los aspectos más comunes de calidad de datos: existencia de valores, unicidad, formato, relaciones entre hojas, comparaciones y condiciones lógicas. Todas estas reglas pueden ser definidas sin necesidad de escribir código, lo que permite el uso de la herramienta por parte de usuarios no técnicos.

☑ Interfaz gráfica amigable y efectiva

Se ha logrado construir una interfaz gráfica coherente, limpia y profesional, en la que cada componente tiene una función clara. Las vistas de configuración de reglas han sido cuidadosamente diseñadas para guiar al usuario paso a paso, minimizando errores y reduciendo la curva de aprendizaje.

Además, los resultados de validación se presentan de forma clara, tanto en pantalla como en los archivos exportados, siguiendo un formato homogéneo que facilita la revisión y toma de decisiones.

☑ Diseño técnico robusto y extensible

Desde el punto de vista técnico, la aplicación presenta una arquitectura modular, basada en principios sólidos de diseño de software. Gracias a la separación en paquetes y la independencia entre modelo, vista y lógica de validación, el sistema está preparado para evolucionar en el futuro.

Aunque finalmente se descartó la integración de una base de datos por decisión estratégica, el sistema se ha construido con un nivel de desacoplamiento que facilita su incorporación futura si fuera necesario.

☑ Cumplimiento de los objetivos específicos

A continuación se presenta una lista resumen de los objetivos específicos alcanzados:

- Carga de múltiples archivos Excel
- Visualización estructurada de archivos y hojas
- Procesamiento unificado del contenido
- Definición de reglas de validación personalizadas
- Validación de existencia de valores obligatorios
- Detección de valores duplicados

- Validación de tipos de datos y formatos
- Verificación de referencias cruzadas entre hojas
- Definición de reglas de comparación directa
- Definición de reglas condicionales
- Gestión de reglas mediante interfaz gráfica
- Ejecución de validaciones y generación de resultados
- Visualización de errores en la interfaz
- Exportación del informe de validación en formato Excel

El sistema ha cumplido plenamente con su propósito inicial: ofrecer una herramienta accesible, profesional y eficaz para validar datos en hojas Excel mediante reglas configurables. La valoración de los resultados alcanzados es por tanto muy satisfactoria, tanto desde el punto de vista funcional como técnico y visual.

7.2 Limitaciones detectadas y soluciones aplicadas

Durante el proceso de desarrollo y prueba de la herramienta, se identificaron diversas limitaciones técnicas y funcionales que requirieron ajustes en el diseño o implementación. A continuación, se detallan las principales incidencias detectadas y las soluciones aplicadas en cada caso:

7.2.1 Lectura de múltiples archivos Excel con estructuras dispares

▸ Problema:

Cuando los archivos cargados tenían estructuras muy diferentes (por ejemplo, distintas hojas, formatos o encabezados), resultaba complejo unificarlos y permitir una validación coherente.

▹ Solución:

Se desarrolló una clase central (LibroUnificado) que estructura internamente todos los archivos y hojas como un único modelo de datos accesible, sin requerir que tengan una estructura idéntica. Se implementaron métodos robustos de acceso por nombre de hoja, celda o rango, permitiendo validar archivos heterogéneos sin conflicto.

7.2.2 Validación cruzada entre hojas de distintos archivos

▸ Problema:

Al aplicar reglas que requerían comparar datos entre hojas de distintos archivos, surgían errores si los nombres eran ambiguos o se repetían.

▹ Solución:

Se añadió una lógica de identificación única basada en el nombre del archivo y de la hoja, garantizando referencias inequívocas incluso cuando los nombres se repiten. Además, se mejoró la interfaz de selección para mostrar claramente qué archivo y hoja está seleccionando el usuario.

7.2.3 Diferencias visuales en componentes JavaFX

▸ Problema:

El estilo visual por defecto de JavaFX resultaba demasiado básico y poco atractivo, afectando negativamente la percepción profesional de la herramienta.

▹ Solución:

Se diseñó e integró una hoja de estilos CSS personalizada (estilos.css) con botones estilizados, tipografías elegantes, colores suaves, esquinas redondeadas y formatos visuales unificados, mejorando la experiencia estética del usuario.

7.2.4 Detección inconsistente de errores en celdas vacías

▸ Problema:

En reglas como "valor obligatorio" o "tipo de dato", las celdas vacías no se trataban siempre de forma coherente, causando falsos negativos.

▹ Solución:

Se añadió un parámetro adicional en estas reglas para permitir o no valores vacíos, y se ajustaron los validadores para distinguir correctamente entre “vacío permitido” y “vacío como error”.

7.2.5 Mensajes de error poco explicativos

▸ Problema:

Los errores mostrados al usuario al configurar reglas incorrectamente (por ejemplo, sin seleccionar ubicación) eran poco informativos o no se mostraban claramente.

▹ Solución:

Se unificó la gestión de errores visuales con una etiqueta (Label info) común en todos los formularios, destacando en rojo los mensajes de validación y explicando claramente el motivo del error y cómo corregirlo.

7.2.6 Inconsistencias en la estructura de resultados

▸ Problema:

Los errores mostrados en la interfaz y los exportados a Excel no seguían siempre el mismo formato, lo que dificultaba su interpretación.

▹ Solución:

Se definió un formato común de resultados para todas las reglas, tanto en la visualización en pantalla como en la exportación. Este formato incluye: título de la regla, símbolo visual, lista clara de errores y separación entre bloques.

7.2.7 Exportación en formato PDF

▸ Problema:

Se evaluó la posibilidad de exportar los resultados en formato PDF, pero las bibliotecas de generación de PDF como iText presentaban restricciones de licencia (AGPL), lo cual podía ser incompatible con futuras aplicaciones del proyecto.

▹ Solución:

Se descartó la integración de exportación en PDF para evitar problemas legales. En su lugar, se potenció la exportación en Excel, ampliamente compatible, editable y más útil en entornos ofimáticos.

7.3 Posibles mejoras futuras

A pesar de que la herramienta cumple satisfactoriamente con todos los objetivos funcionales definidos, existen varias líneas de mejora que podrían implementarse en futuras versiones para ampliar su funcionalidad, mejorar la experiencia del usuario o adaptarse a contextos de uso más avanzados:

➢ Persistencia de reglas y configuraciones

Actualmente, las reglas definidas por el usuario existen únicamente durante la sesión activa. Sería interesante permitir guardar y cargar conjuntos de reglas previamente configuradas, para facilitar la reutilización en futuras validaciones o su aplicación sistemática sobre archivos similares.

➢ Historial de validaciones

Otra mejora útil sería almacenar un historial de archivos validados, junto con las reglas aplicadas y los resultados obtenidos. Esto permitiría al usuario retomar sesiones anteriores, comparar validaciones entre versiones de un mismo archivo o documentar el proceso de control de calidad.

➢ Edición de reglas sobre archivos validados previamente

Junto con el historial, se podría implementar una función para seleccionar un archivo previamente validado y modificar las reglas aplicadas (añadir, eliminar o cambiar parámetros), sin necesidad de reiniciar completamente la sesión.

➢ Mejoras visuales adicionales

Aunque la herramienta ya cuenta con una estética cuidada y profesional, podrían incorporarse elementos como iconos contextuales, animaciones sutiles o modo oscuro, para modernizar aún más la experiencia y adaptarse a las preferencias del usuario.

➢ Soporte para nuevos formatos de salida

Actualmente, los resultados pueden exportarse en formato Excel, pero podrían habilitarse nuevos formatos como PDF, CSV o incluso integraciones con servicios online (correo electrónico, almacenamiento en la nube, etc.).

➢ Integración con bases de datos

Aunque se decidió no integrar finalmente una base de datos por simplicidad, esta funcionalidad podría ser de utilidad en entornos empresariales donde sea necesario centralizar reglas, historiales o usuarios.

➢ Validaciones adicionales

Se podrían incorporar nuevas reglas de validación específicas.

➢ Internacionalización

Actualmente la interfaz y los mensajes están en español. Sería interesante permitir cambiar el idioma de la herramienta para hacerla accesible a una audiencia internacional.

8 Análisis de impacto

8.1 Impacto académico y personal

El desarrollo de esta herramienta ha tenido un impacto significativo tanto en el ámbito académico como en el crecimiento personal del autor.

Desde el punto de vista académico, el proyecto ha permitido consolidar y aplicar de forma práctica numerosos conocimientos adquiridos a lo largo del Grado en Ingeniería Informática. Conceptos clave como el diseño modular, la programación orientada a objetos, el manejo de estructuras de datos, el uso de librerías externas (como Apache POI), el desarrollo de interfaces gráficas con JavaFX o la separación de capas en arquitecturas software han sido puestos en práctica de manera integrada, evidenciando la capacidad de transformar conocimientos teóricos en una solución funcional y real.

Además, se han adquirido competencias nuevas no abordadas directamente en asignaturas, como el diseño visual con CSS adaptado a JavaFX, la validación estructurada de datos en contextos empresariales y la generación de informes dinámicos en hojas de cálculo. Todo ello ha contribuido a ampliar el perfil técnico del autor y su preparación para entornos laborales reales.

A nivel personal, el proyecto ha supuesto un reto que ha requerido constancia, planificación, capacidad de aprendizaje autónomo y toma de decisiones informadas. Se han afrontado obstáculos técnicos, cambios de alcance y reconsideración de ideas iniciales, lo que ha fortalecido la capacidad crítica y la flexibilidad ante imprevistos.

El hecho de haber diseñado y desarrollado una herramienta completa desde cero, con interfaz gráfica personalizada, reglas configurables, validación automatizada y exportación de resultados, ha sido una fuente de satisfacción personal y una demostración tangible de las habilidades adquiridas durante la carrera.

En resumen, este Trabajo de Fin de Grado no solo ha cumplido una función académica como ejercicio de integración final, sino que también ha reforzado la confianza del autor en su capacidad para abordar proyectos técnicos complejos, sentando una base sólida para su futura trayectoria profesional.

8.2 Aplicabilidad en entornos reales

La herramienta desarrollada en este proyecto está plenamente alineada con las necesidades reales de validación de datos en entornos donde el uso de archivos Excel sigue siendo predominante. A pesar de la creciente automatización de procesos en muchas organizaciones, la realidad es que Excel continúa siendo una herramienta de uso masivo en departamentos de administración, finanzas, contabilidad, recursos humanos, investigación y docencia, entre otros. [9]

En estos contextos, la validación de datos suele realizarse de forma manual o mediante fórmulas distribuidas en distintas hojas, lo que incrementa la probabilidad de errores, duplicidades u omisiones. [10] La herramienta

propuesta aporta valor de manera inmediata en este tipo de entornos, gracias a las siguientes características:

➢ No requiere conocimientos técnicos: cualquier usuario con manejo básico de Excel puede utilizarla sin necesidad de programación ni conocimientos en bases de datos o lenguajes de scripting.

➢ Versatilidad en reglas de validación: permite definir reglas complejas y condicionales entre diferentes hojas o libros, lo cual resulta difícil de implementar únicamente con fórmulas de Excel.

➢ Revisión estructurada: centraliza el análisis de múltiples archivos y genera informes claros y exportables, lo que facilita la trazabilidad de errores y el cumplimiento de estándares de calidad en los datos.

➢ Aplicabilidad directa en tareas cotidianas: validación de informes financieros, revisión de listados de personal, control de inventarios, verificación de encuestas, análisis de notas académicas, entre otros.

➢ Sin necesidad de instalación compleja: al tratarse de una aplicación de escritorio, puede ejecutarse en cualquier máquina con Java sin requerir infraestructura adicional o conexión a Internet.

Estas características la convierten en una herramienta especialmente útil para pequeñas y medianas empresas (PYMEs) que no disponen de sistemas avanzados de validación de datos, departamentos dentro de grandes organizaciones donde se siguen utilizando hojas de cálculo como soporte operativo, centros educativos o investigadores que trabajan con datos recopilados manualmente o sin un sistema de validación automático, consultores de datos o auditores que necesiten revisar ficheros de clientes antes de integrarlos en sus propios sistemas, etc.

En resumen, la herramienta desarrollada trasciende el ámbito académico y presenta un alto grado de aplicabilidad en entornos reales, sirviendo como solución eficaz y accesible a una problemática común en numerosos sectores. Esta capacidad de transferencia y aplicabilidad práctica es uno de los puntos fuertes del proyecto.

8.3 Contribución al proceso de validación en Excel

La herramienta desarrollada en este Trabajo de Fin de Grado representa una contribución relevante y concreta al proceso de validación de datos en archivos Excel, un área donde, a pesar del uso masivo de esta tecnología, siguen existiendo carencias importantes en términos de control de calidad, automatización y prevención de errores.

En primer lugar, la herramienta formaliza y estructura el proceso de validación, algo que habitualmente se realiza de forma manual, dispersa y propensa a errores. Estudios sobre la calidad de hojas de cálculo han identificado que la validación y el aseguramiento de calidad suelen necesitar enfoques más sistemáticos que los controles manuales tradicionales. Mediante un enfoque declarativo basado en reglas personalizadas, el usuario puede establecer criterios claros de integridad, coherencia y formato, lo que transforma la validación en un proceso replicable, trazable y más profesional.

A diferencia de soluciones ad-hoc basadas en fórmulas o macros, este sistema permite centralizar múltiples comprobaciones en una única plataforma visual. Esto es especialmente útil cuando se trabaja con varios archivos simultáneamente, o con hojas de cálculo extensas donde validar manualmente cada aspecto es inviable. [11]

Además, la herramienta introduce una lógica de separación de responsabilidades, donde la validación no está incrustada en los propios documentos, sino definida en una capa externa y flexible. Este enfoque facilita el mantenimiento de reglas, su revisión, y su reutilización en diferentes conjuntos de datos.

La contribución también se manifiesta en términos de accesibilidad: cualquier usuario puede aplicar validaciones complejas sin necesidad de programar ni conocer expresiones avanzadas de Excel. [12] Esto democratiza el control de calidad en hojas de cálculo y lo hace accesible para perfiles no técnicos en contextos empresariales, administrativos o académicos.

Por último, la herramienta promueve buenas prácticas de trabajo con Excel, al impulsar una mayor conciencia sobre la necesidad de validar los datos y ofrecer una vía eficaz para hacerlo. En este sentido, el proyecto no solo ofrece una solución técnica, sino que también contribuye a una cultura más rigurosa en el manejo y análisis de datos.

8.4 Relación del trabajo con los Objetivos de Desarrollo Sostenible

La Agenda 2030 para el Desarrollo Sostenible de las Naciones Unidas establece 17 Objetivos de Desarrollo Sostenible (ODS) que constituyen un plan de acción para promover la prosperidad y proteger el planeta, abordando además desigualdades sociales y económicas a nivel global [13]. Cada objetivo está acompañado de metas e indicadores que permiten monitorizar el progreso hacia su cumplimiento. La disponibilidad de datos de alta calidad, accesibles y fiables es fundamental para medir ese avance, diseñar políticas eficazmente y tomar decisiones basadas en evidencia.

El desarrollo de una herramienta de validación de datos en Excel, como la presentada en este Trabajo de Fin de Grado, contribuye de manera tangible a fortalecer las capacidades estadísticas y de gestión de datos que son clave para el logro de varios ODS, especialmente aquellos que dependen de indicadores fiables y completos.

En particular, se puede vincular este trabajo con los siguientes objetivos:

➢ *ODS 9: Industria, innovación e infraestructuras*

El ODS 9 promueve la construcción de infraestructuras resilientes, la industrialización inclusiva y la innovación tecnológica. La herramienta desarrollada, al ofrecer una plataforma automatizada para validar calidad de datos en hojas de cálculo (una de las formas dominantes de gestión de datos en empresas y administraciones), impulsa la innovación en prácticas de gestión de datos y respalda procesos internos más eficientes y robustos.

➢ *ODS 17: Alianzas para lograr los objetivos*

El ODS 17 destaca la importancia de fortalecer la cooperación, los sistemas de datos y la disponibilidad de estadísticas desagregadas como base para monitorizar y alcanzar los ODS. El desarrollo de herramientas que mejoran la calidad, consistencia y accesibilidad de los datos contribuye directamente a mejorar la capacidad de generar estadísticas confiables que pueden ser utilizadas por distintos actores (empresas, administraciones públicas, investigadores) para informar indicadores y facilitar seguimiento de metas de la Agenda 2030. Además, al habilitar a usuarios no técnicos para aplicar validaciones complejas sin programación, se fomenta la participación de un mayor número de organizaciones y personas en los procesos de control de calidad de datos, fortaleciendo así la cultura de datos confiables e inclusivos que requiere la comunidad global para avanzar en los ODS.

De este modo, aunque el enfoque principal de esta herramienta sea técnico y orientado al ámbito empresarial o administrativo, su impacto trasciende ese ámbito al facilitar mejores prácticas de gestión de información y contribuir a procesos decisionales basados en evidencia, condición indispensable para medir progresos en los Objetivos de Desarrollo Sostenible.

9 Bibliografía

[1] Microsoft, “Aplicar la validación de datos a celdas”, *Microsoft Support*. [En línea].
https://support.microsoft.com/es-es/office/aplicar-la-validaci%C3%B3n-de-datos-a-celdas-29fecbcc-d1b9-42c1-9d76-eff3ce5f7249

[2] M. Alexander y D. Kusleika, *Excel 2019 Power Programming with VBA*. Hoboken, NJ, USA: John Wiley & Sons, 2019.

[3] R. S. Koppula, “Comparative analysis of ETL tools: Talend, Informatica, and more”, *Journal of Scientific and Engineering Research*, vol.8, no.6, pp.45-52, 2021.

[4] G. Maksimenko, “Lack of automation in Excel: challenges in modern logistics operations”, *Adexin Blog*, 2024. [En línea].
https://adexin.com/blog/excel-limitations/

[5] N. Gagliordi, “10 common spreadsheet risks and solutions for businesses”, *Oracle Analytics*, 2023. [En línea].
https://www.oracle.com/analytics/spreadsheet-risks/

[6] I. Taleb, M. A. Serhani, C. Bouhaddioui y R. Dssouli, “Big data quality framework: a holistic approach to continuous quality management”, *Journal of Big Data*, vol. 8, art. no. 12, 2021, doi: 10.1186/s40537-021-00409-7

[7] OpenJFX, “JavaFX API documentation,” 2025. [En línea].
https://openjfx.io/javadoc/25/

[8] Apache Software Foundation, “Apache POI API documentation”, 2020. [En línea].
https://poi.apache.org/apidocs/4.1/

[9] P.-L. Poon, M. F. Lau, Y. T. Yu y S.-F. Tang, “Spreadsheet quality assurance: a literature review”, *Frontiers of Computer Science*, vol. 18, no. 2, art. no. 182203, Jan. 2024, doi: 10.1007/s11704-023-2384-6.

[10] S. G. Powell, K. R. Baker y B. Lawson, “Impact of errors in operational spreadsheets”, *Decision Support Systems*, vol. 47, no. 3, pp. 123–132, 2009, doi: 10.1016/j.dss.2009.02.002.

[11] IDBS, “El problema de Excel: elegir la herramienta adecuada para la gestión de datos de productos y procesos”, 2023. [En línea].

Elegir la herramienta adecuada para la gestión de datos de productos y procesos

[12] L. Chou, “How to use data validation in Excel effectively”, *FanRuan Blog*, 2025. [En línea].

How to Use Data Validation in Excel Effectively

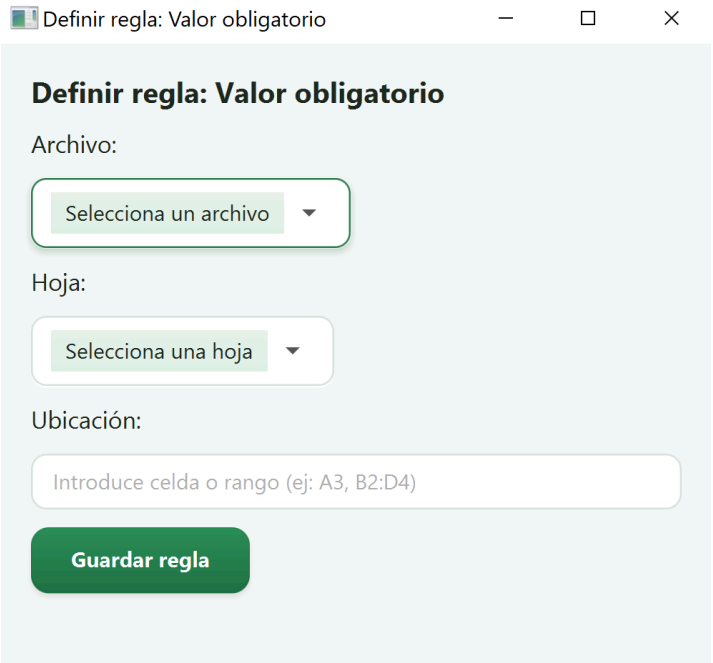
[13] Naciones Unidas, “Objetivos y metas de desarrollo sostenible”, *Agenda 2030*, 2015. [En línea].

Objetivos y metas de desarrollo sostenible - Desarrollo Sostenible

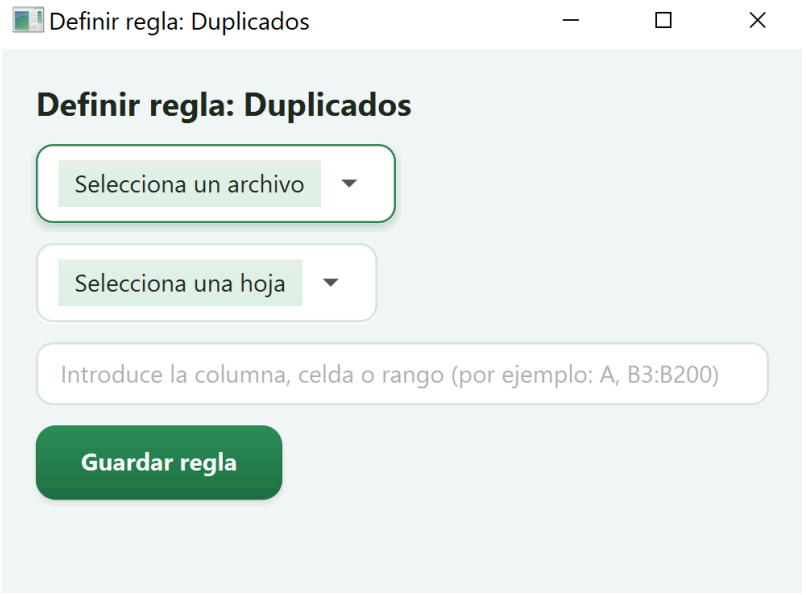
10 Anexos

Anexo A: Vistas de definición de reglas

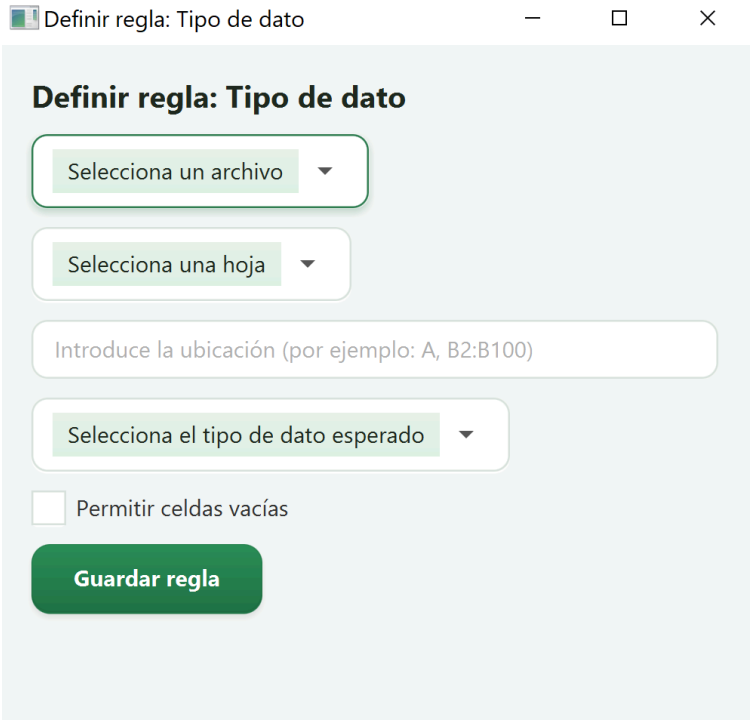
Anexo A.1: Valor Obligatorio



Anexo A.2: Duplicados

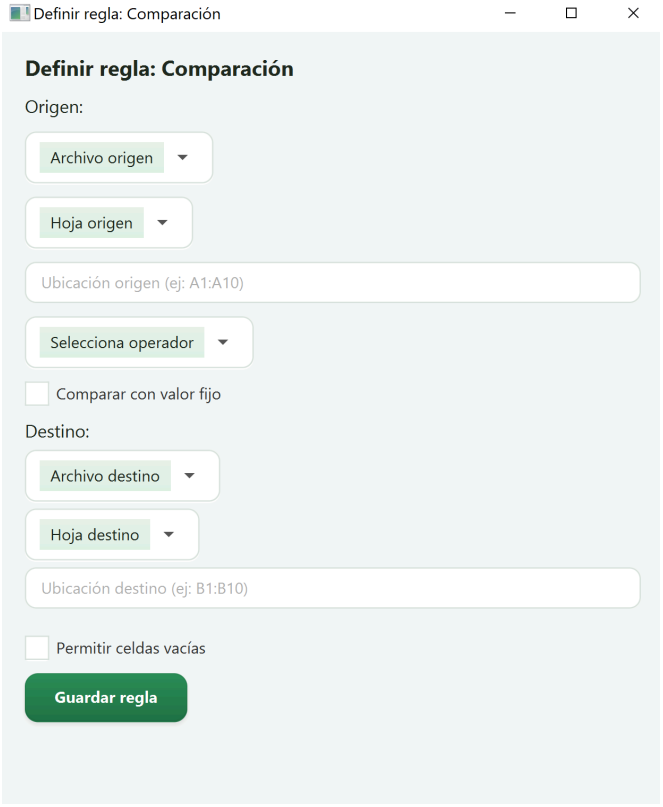


Anexo A.3: Tipo de dato

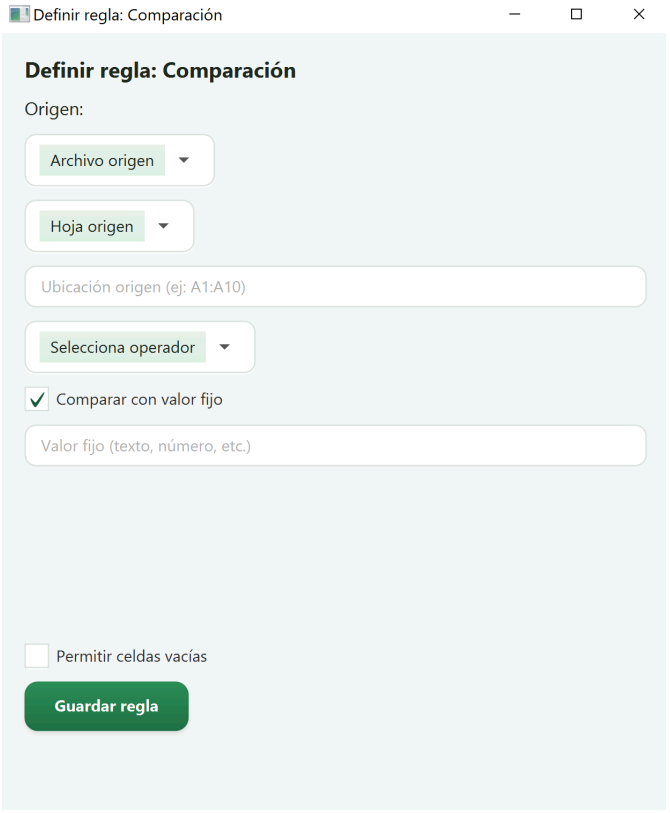


Anexo A.4: Referencias cruzadas

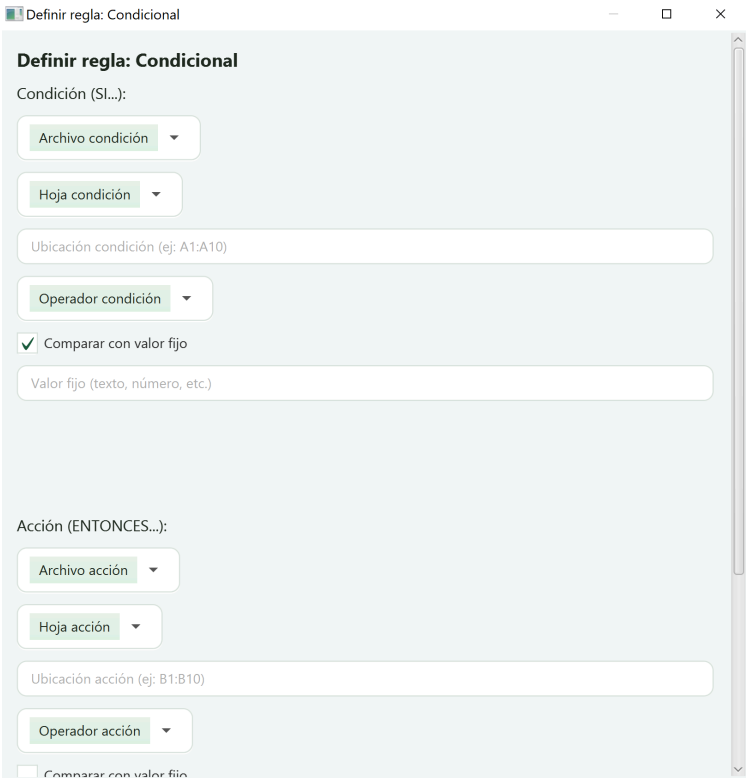
Anexo A.5: Comparación con destino

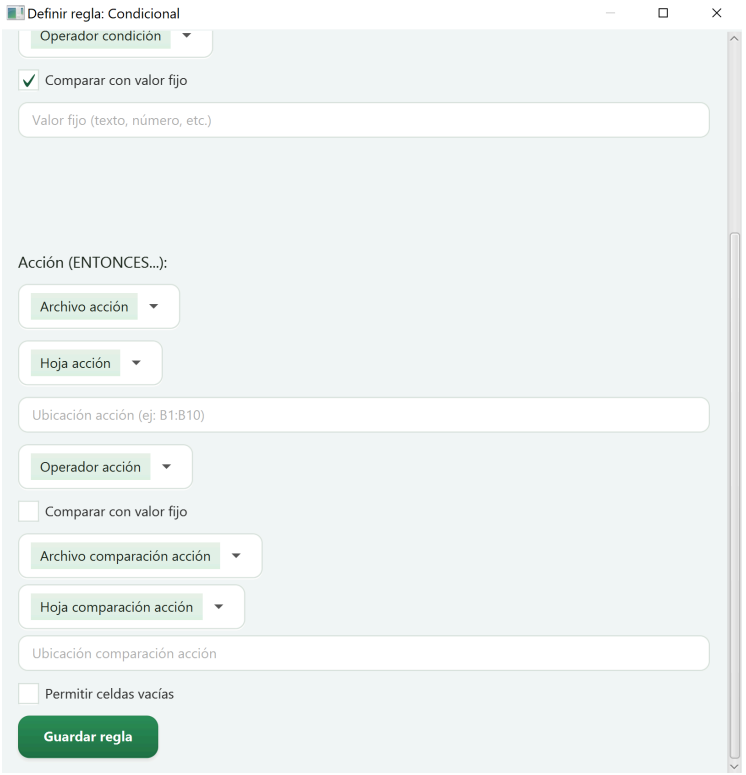


Anexo A.6: Comparación con valor fijo



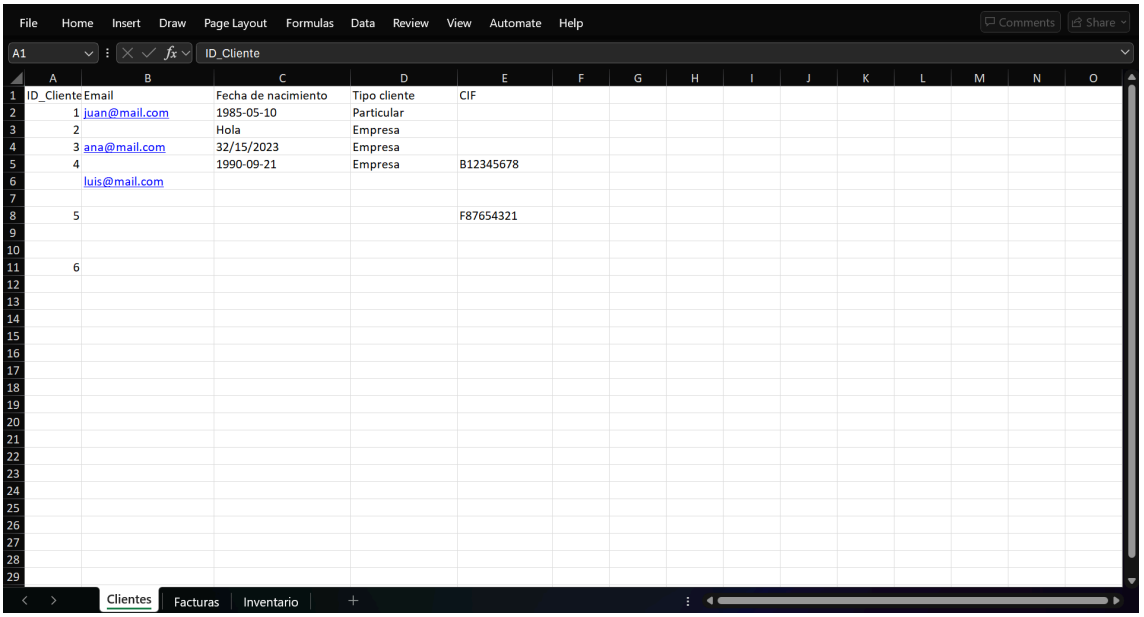
Anexo A.7: Condicional



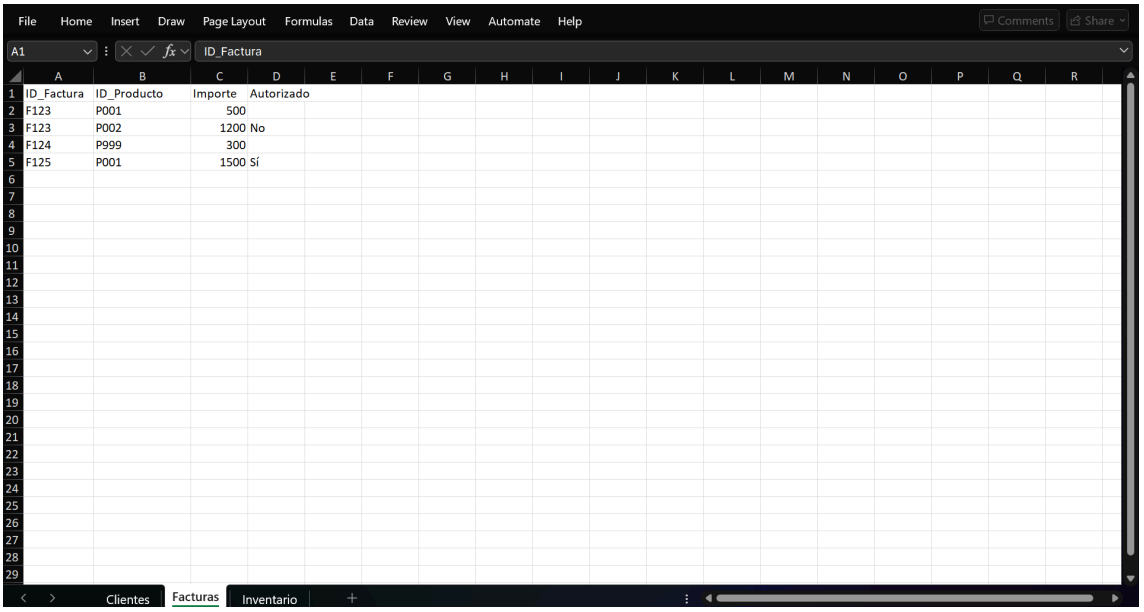


Anexo B: Archivo creado para pruebas

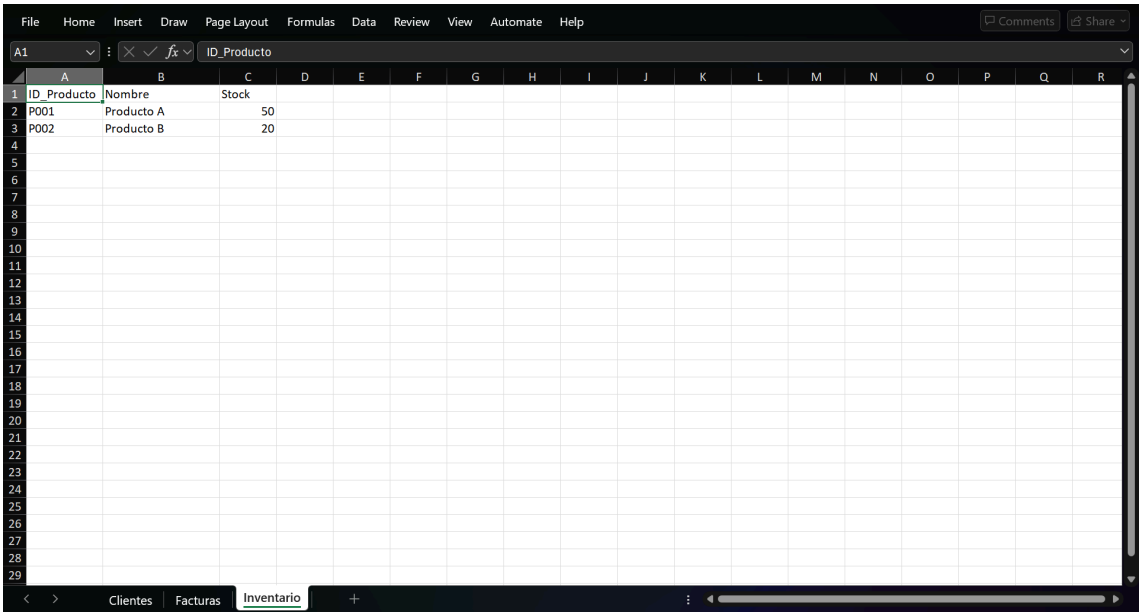
Anexo B.1: Hojas del archivo



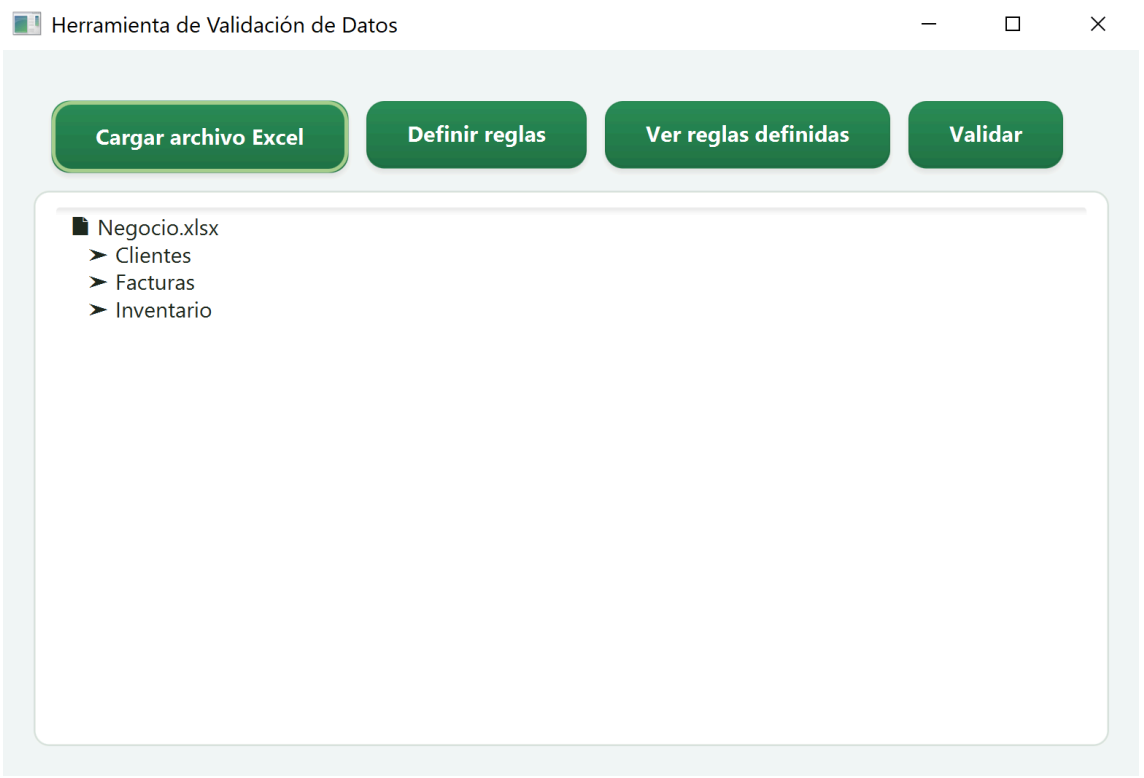
ID_Cliente	Email	Fecha de nacimiento	Tipo cliente	CIF
1	juan@mail.com	1985-05-10	Particular	
2		Hola	Empresa	
3	ana@mail.com	32/15/2023	Empresa	
4		1990-09-21	Empresa	B12345678
	luis@mail.com			
5				F87654321
6				



ID_Factura	ID_Producto	Importe	Autorizado
F123	P001	500	
F123	P002	1200	No
F124	P999	300	
F125	P001	1500	Sí

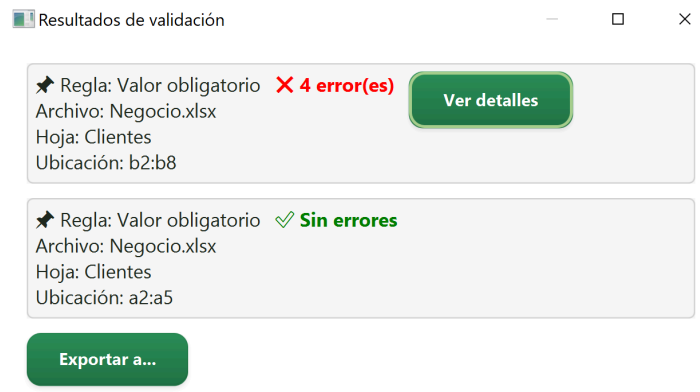


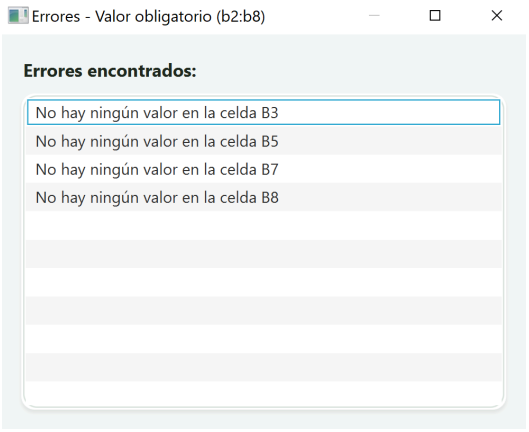
Anexo B.2: Archivo añadido a la herramienta



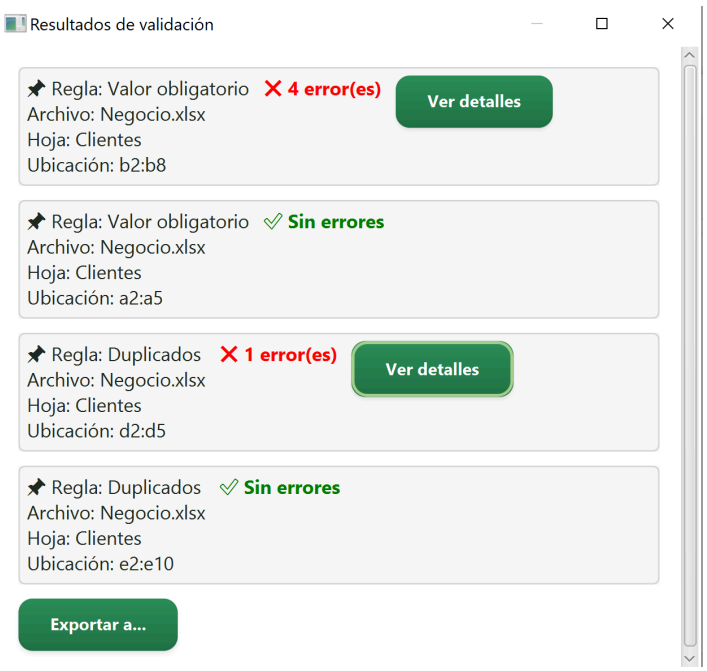
Anexo C: Pruebas Realizadas

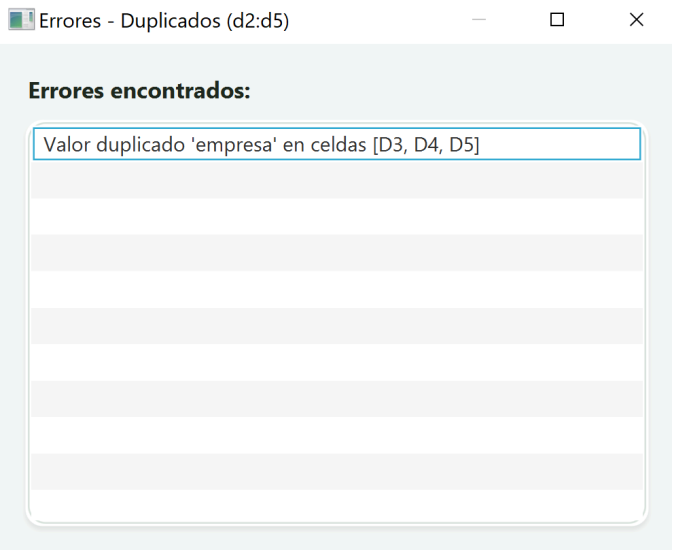
Anexo C.1: Valor Obligatorio



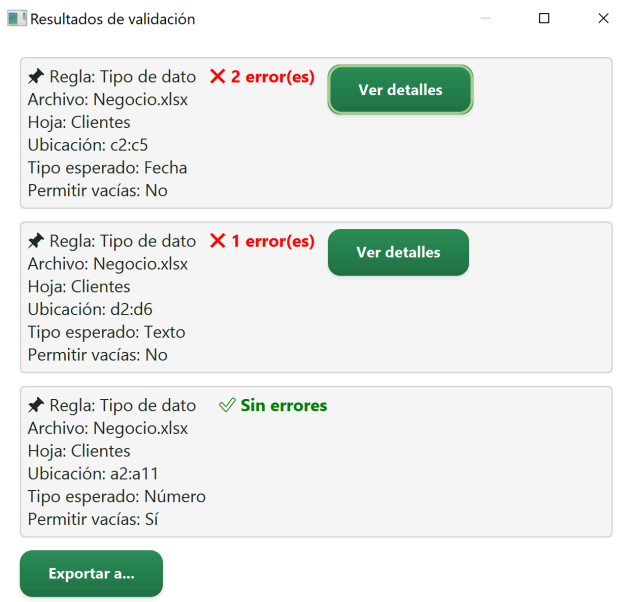


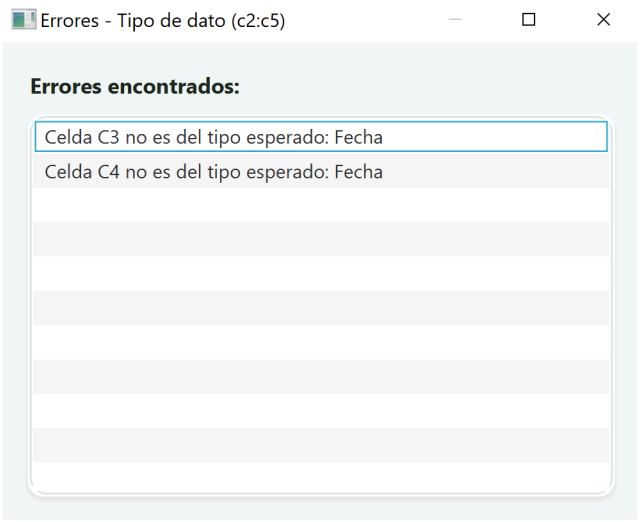
Anexo C.2: Duplicados





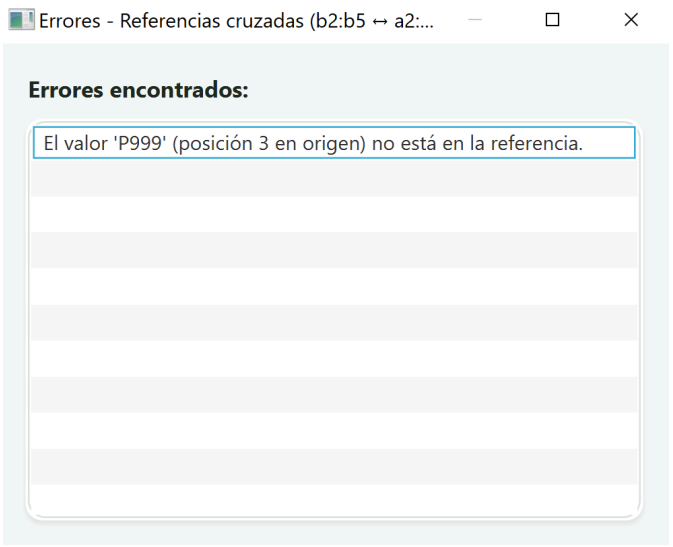
Anexo C.3: Tipo de dato

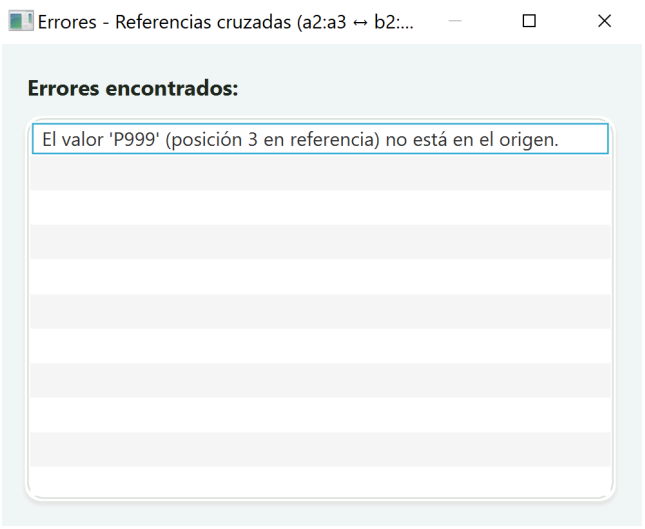




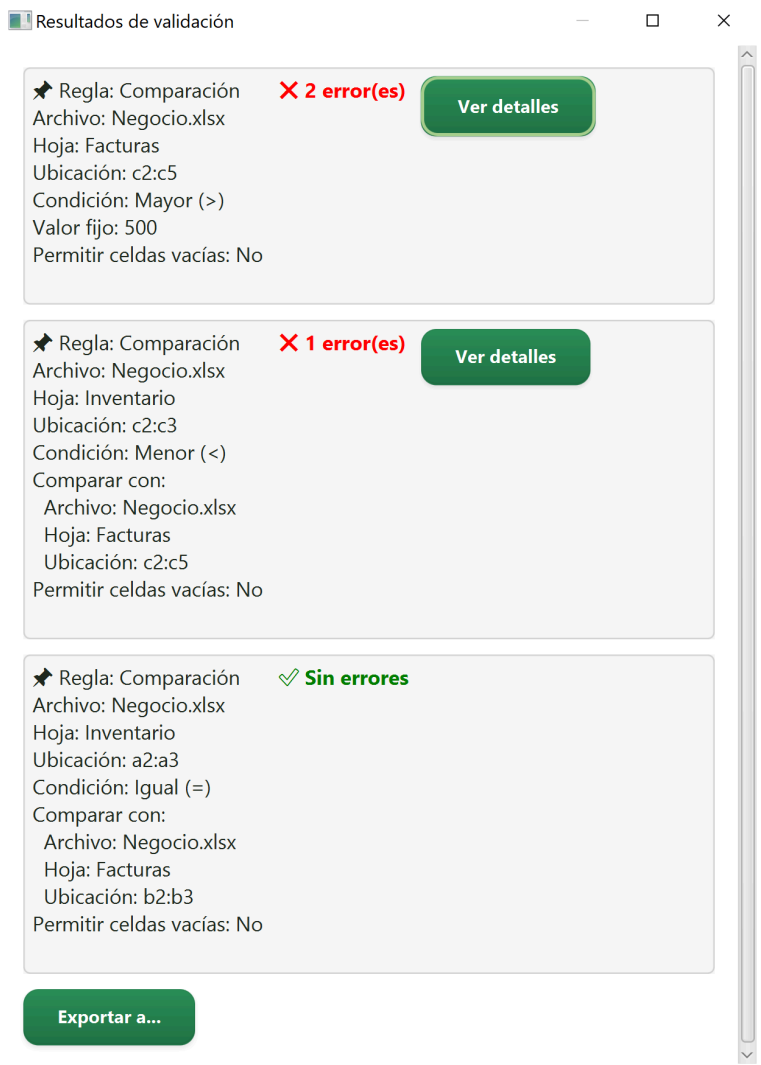
Anexo C.4: Referencias cruzadas

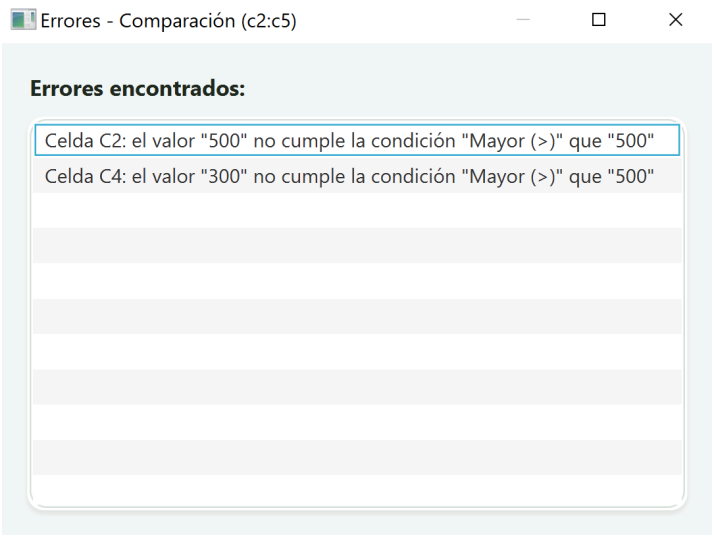

Resultados de validación
Regla: Referencias cruzadas
1 error(es)
Ver detalles
Archivo: Negocio.xlsx
Hoja: Facturas
Ubicación: b2:b5 ↔ a2:a3
Archivo destino: Negocio.xlsx
Hoja destino: Inventario
Ubicación origen: b2:b5
Ubicación destino: a2:a3
Bidireccional: No
Regla: Referencias cruzadas
Sin errores
Archivo: Negocio.xlsx
Hoja: Inventario
Ubicación: a2:a3 ↔ b2:b5
Archivo destino: Negocio.xlsx
Hoja destino: Facturas
Ubicación origen: a2:a3
Ubicación destino: b2:b5
Bidireccional: No
Regla: Referencias cruzadas
1 error(es)
Ver detalles
Archivo: Negocio.xlsx
Hoja: Inventario
Ubicación: a2:a3 ↔ b2:b5
Archivo destino: Negocio.xlsx
Hoja destino: Facturas
Ubicación origen: a2:a3
Ubicación destino: b2:b5
Bidireccional: Sí
Exportar a...

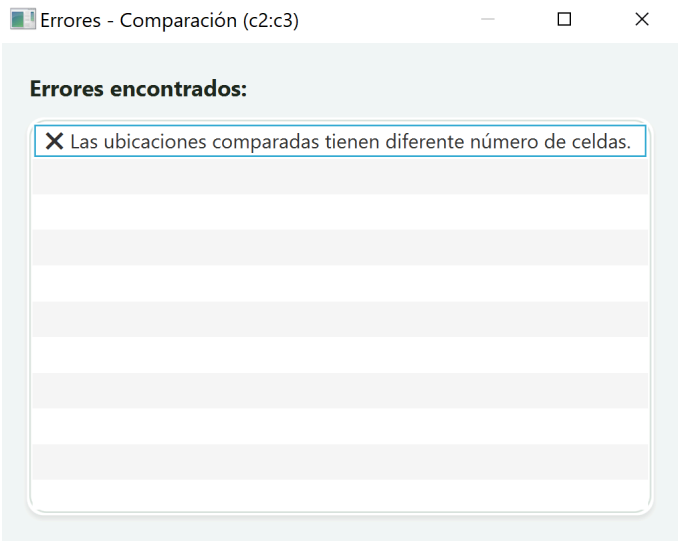

Errores - Referencias cruzadas (b2:b5 ↔ a2:...
Errores encontrados:
El valor 'P999' (posición 3 en origen) no está en la referencia.



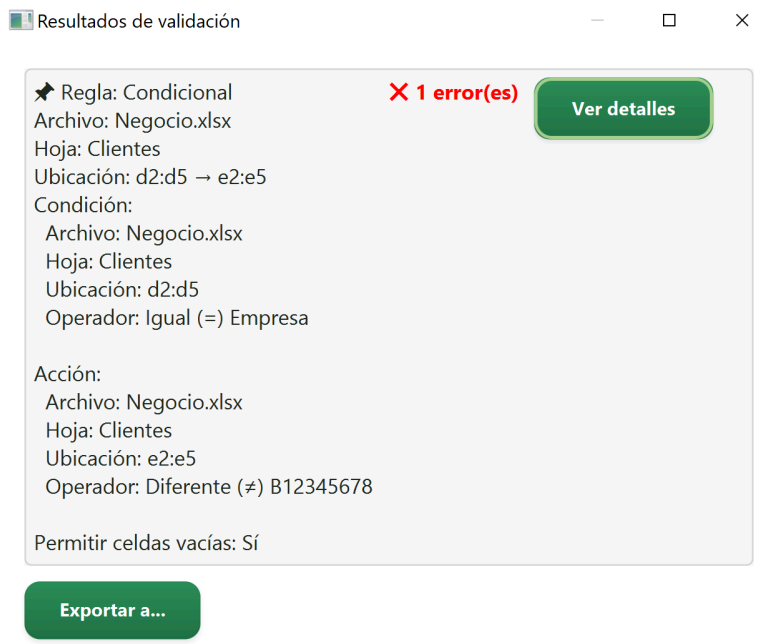
Anexo C.5: Comparación



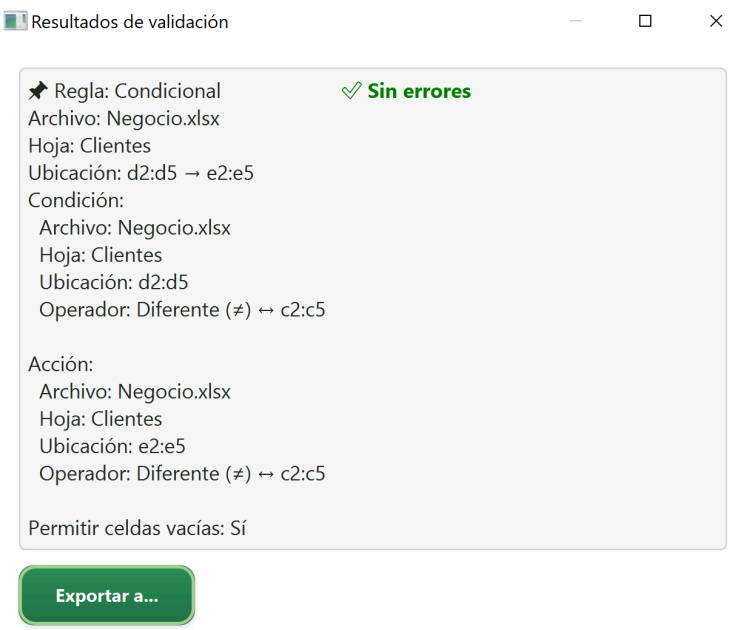


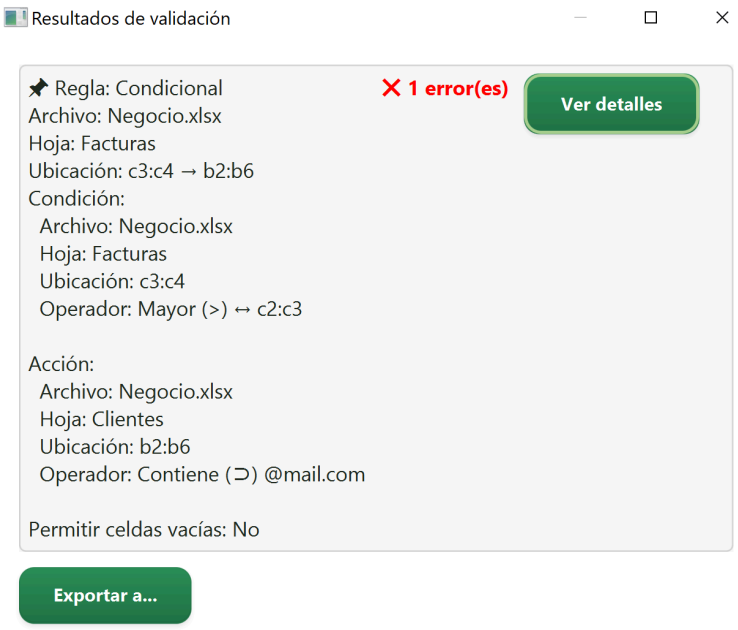


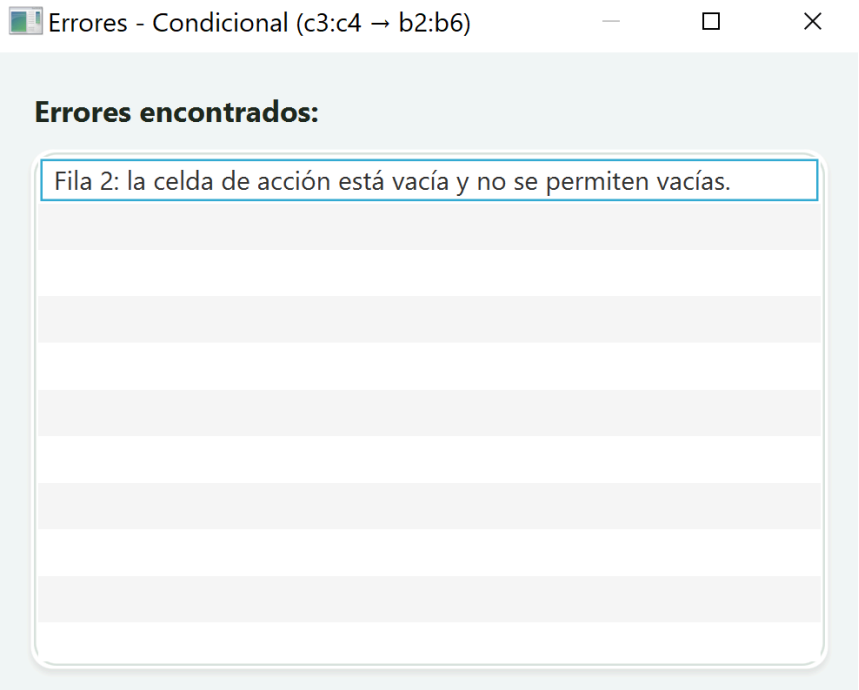
Anexo C.6: Condicional


Resultados de validación
📌 Regla: Condicional
❌ 1 error(es)
Ver detalles
Archivo: Negocio.xlsx
Hoja: Clientes
Ubicación: d2:d5 → e2:e5
Condición:
Archivo: Negocio.xlsx
Hoja: Clientes
Ubicación: d2:d5
Operador: Igual (=) Empresa
Acción:
Archivo: Negocio.xlsx
Hoja: Clientes
Ubicación: e2:e5
Operador: Diferente (≠) B12345678
Permitir celdas vacías: Sí
Exportar a...


Errores - Condicional (d2:d5 → e2:e5)
Errores encontrados:
Fila 4: la acción no se cumple. Valor actual: 'B12345678', esperado: 'B12345678'.


Resultados de validación
📌 Regla: Condicional
✓ Sin errores
Archivo: Negocio.xlsx
Hoja: Clientes
Ubicación: d2:d5 → e2:e5
Condición:
Archivo: Negocio.xlsx
Hoja: Clientes
Ubicación: d2:d5
Operador: Diferente (≠) ↔ c2:c5
Acción:
Archivo: Negocio.xlsx
Hoja: Clientes
Ubicación: e2:e5
Operador: Diferente (≠) ↔ c2:c5
Permitir celdas vacías: Sí
Exportar a...


Resultados de validación
📌 Regla: Condicional
✕ 1 error(es)
Ver detalles
Archivo: Negocio.xlsx
Hoja: Facturas
Ubicación: c3:c4 → b2:b6
Condición:
Archivo: Negocio.xlsx
Hoja: Facturas
Ubicación: c3:c4
Operador: Mayor (>) ↔ c2:c3
Acción:
Archivo: Negocio.xlsx
Hoja: Clientes
Ubicación: b2:b6
Operador: Contiene (⊃) @mail.com
Permitir celdas vacías: No
Exportar a...



Anexo D: Repositorio con código de la herramienta

El código fuente completo de la herramienta desarrollada se encuentra disponible en el repositorio GitHub:

https://github.com/jmario37/herramienta-validacion-excel

Este documento esta firmado por

	**Firmante**	CN=tfgm.fi.upm.es, OU=CCFI, O=ETS Ingenieros Informaticos - UPM, C=ES
	Fecha/Hora	Tue Jan 20 23:05:23 CET 2026
	Emisor del Certificado	EMAILADDRESS=camanager@etsiinf.upm.es, CN=CA ETS Ingenieros Informaticos, O=ETS Ingenieros Informaticos - UPM, C=ES
	Numero de Serie	561
	Metodo	urn:adobe.com:Adobe.PPKLite:adbe.pkcs7.sha1 (Adobe Signature)