ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA (ICAI)

MÁSTER EN BIG DATA: TECNOLOGÍA Y ANALÍTICA AVANZADA

Implementación y validación de un framework de automatización de liquidaciones económicas de *third parties* en entorno bancario regulado

Autor: Sofía Sebastián Sánchez

Director: Rubén Comesaña Figueiras

Madrid

Junio 2026

Declaro, bajo mi responsabilidad, que el Proyecto presentado con el título

Implementación y validación de un framework de automatización de liquidaciones económicas de *third parties* en entorno bancario regulado

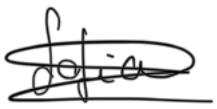
en la ETS de Ingeniería - ICAI de la Universidad Pontificia Comillas en el

curso académico 2025/26 es de mi autoría, original e inédito y

no ha sido presentado con anterioridad a otros efectos.

El Proyecto no es plagio de otro, ni total ni parcialmente y la información que ha sido

tomada de otros documentos está debidamente referenciada.



Fdo.: Nombre del alumno Fecha: 27/ junio/ 2026

Autorizada la entrega del proyecto

EL DIRECTOR DEL PROYECTO

Fdo.: Rubén Comesaña Figueiras Fecha: 27/ junio/ 2026

Firmado por MORRAS RUIZ-FALCO CARLOS - ***0907** el día 28/06/2026 con un certificado emitido por AC FNMT Usuarios

V yectos

Fdo.: Carlos Morrás Ruiz-Falcó Fecha: ……/ ……/ ……

AUTORIZACIÓN PARA LA DIGITALIZACIÓN, DEPÓSITO Y DIVULGACIÓN EN RED DE PROYECTOS FIN DE GRADO, FIN DE MÁSTER, TESINAS O MEMORIAS DE BACHILLERATO

1º. Declaración de la autoría y acreditación de la misma.

El autor D. Sofía Sebastián Sánchez

DECLARA ser el titular de los derechos de propiedad intelectual de la obra: Implementación y validación de un *framework* de automatización de liquidaciones económicas de *third parties* en entorno bancario regulado, que ésta es una obra original, y que ostenta la condición de autor en el sentido que otorga la Ley de Propiedad Intelectual.

2º. Objeto y fines de la cesión.

Con el fin de dar la máxima difusión a la obra citada a través del Repositorio institucional de la Universidad, el autor **CEDE** a la Universidad Pontificia Comillas, de forma gratuita y no exclusiva, por el máximo plazo legal y con ámbito universal, los derechos de digitalización, de archivo, de reproducción, de distribución y de comunicación pública, incluido el derecho de puesta a disposición electrónica, tal y como se describen en la Ley de Propiedad Intelectual. El derecho de transformación se cede a los únicos efectos de lo dispuesto en la letra a) del apartado siguiente.

3º. Condiciones de la cesión y acceso

Sin perjuicio de la titularidad de la obra, que sigue correspondiendo a su autor, la cesión de derechos contemplada en esta licencia habilita para:

a) Transformarla con el fin de adaptarla a cualquier tecnología que permita incorporarla a internet y hacerla accesible; incorporar metadatos para realizar el registro de la obra e incorporar "marcas de agua" o cualquier otro sistema de seguridad o de protección.
b) Reproducirla en un soporte digital para su incorporación a una base de datos electrónica, incluyendo el derecho de reproducir y almacenar la obra en servidores, a los efectos de garantizar su seguridad, conservación y preservar el formato.
c) Comunicarla, por defecto, a través de un archivo institucional abierto, accesible de modo libre y gratuito a través de internet.
d) Cualquier otra forma de acceso (restringido, embargado, cerrado) deberá solicitarse expresamente y obedecer a causas justificadas.
e) Asignar por defecto a estos trabajos una licencia Creative Commons.
f) Asignar por defecto a estos trabajos un HANDLE (URL *persistente)*.

4º. Derechos del autor.

El autor, en tanto que titular de una obra tiene derecho a:

a) Que la Universidad identifique claramente su nombre como autor de la misma
b) Comunicar y dar publicidad a la obra en la versión que ceda y en otras posteriores a través de cualquier medio.
c) Solicitar la retirada de la obra del repositorio por causa justificada.
d) Recibir notificación fehaciente de cualquier reclamación que puedan formular terceras personas en relación con la obra y, en particular, de reclamaciones relativas a los derechos de propiedad intelectual sobre ella.

5º. Deberes del autor.

- El autor se compromete a:

a) Garantizar que el compromiso que adquiere mediante el presente escrito no infringe ningún derecho de terceros, ya sean de propiedad industrial, intelectual o cualquier otro.
b) Garantizar que el contenido de las obras no atenta contra los derechos al honor, a la intimidad y a la imagen de terceros.
c) Asumir toda reclamación o responsabilidad, incluyendo las indemnizaciones por daños, que pudieran ejercitarse contra la Universidad por terceros que vieran infringidos sus derechos e intereses a causa de la cesión.

d) Asumir la responsabilidad en el caso de que las instituciones fueran condenadas por infracción de derechos derivada de las obras objeto de la cesión.

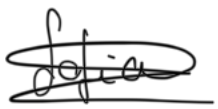
6º. Fines y funcionamiento del Repositorio Institucional.

La obra se pondrá a disposición de los usuarios para que hagan de ella un uso justo y respetuoso con los derechos del autor, según lo permitido por la legislación aplicable, y con fines de estudio, investigación, o cualquier otro fin lícito. Con dicha finalidad, la Universidad asume los siguientes deberes y se reserva las siguientes facultades:

- La Universidad informará a los usuarios del archivo sobre los usos permitidos, y no garantiza ni asume responsabilidad alguna por otras formas en que los usuarios hagan un uso posterior de las obras no conforme con la legislación vigente. El uso posterior, más allá de la copia privada, requerirá que se cite la fuente y se reconozca la autoría, que no se obtenga beneficio comercial, y que no se realicen obras derivadas.
- La Universidad no revisará el contenido de las obras, que en todo caso permanecerá bajo la responsabilidad exclusive del autor y no estará obligada a ejercitar acciones legales en nombre del autor en el supuesto de infracciones a derechos de propiedad intelectual derivados del depósito y archivo de las obras. El autor renuncia a cualquier reclamación frente a la Universidad por las formas no ajustadas a la legislación vigente en que los usuarios hagan uso de las obras.
- La Universidad adoptará las medidas necesarias para la preservación de la obra en un futuro.
- La Universidad se reserva la facultad de retirar la obra, previa notificación al autor, en supuestos suficientemente justificados, o en caso de reclamaciones de terceros.

Madrid, a 27 de junio de 2026

ACEPTA



Fdo. Sofía Sebastián Sánchez

Motivos para solicitar el acceso restringido, cerrado o embargado del trabajo en el Repositorio Institucional:

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA (ICAI)

MÁSTER EN BIG DATA: TECNOLOGÍA Y ANALÍTICA AVANZADA

Implementación y validación de un framework de automatización de liquidaciones económicas de *third parties* en entorno bancario regulado

Autor: Sofía Sebastián Sánchez

Director: Rubén Comesaña Figueiras

Madrid

Junio 2026

Agradecimientos

Todos sabemos que la realización de un trabajo de fin de máster supone un largo y duro proceso, en el que tenemos que enfrentarnos a numerosos retos y dificultades, con el fin de culminar con seriedad un proyecto que demuestre las habilidades y conocimientos adquiridos durante este año. Aunque se trata de una tarea que abordamos individualmente, yo me he sentido muy acompañada en todo el trayecto. Por ello, en primer lugar, quiero agradecer a mi tutor, Rubén Comesaña, su constante escucha, su motivación en los momentos de flaqueza y su valiosa contribución académica para la elaboración del proyecto.

Mi más sincero agradecimiento a la Universidad Pontificia Comillas y a todos mis profesores por haberme brindado la posibilidad de formarme en un modelo académico de excelencia que sin duda me permitirá enfrentarme con determinación a los desafíos de la vida profesional.

Quiero también expresar mi agradecimiento a Accenture y al equipo de Banco Santander en el que he tenido la oportunidad de desarrollar este proyecto, por su confianza, su disposición a resolver mis dudas y por haberme permitido aplicar y poner a prueba mis conocimientos en un entorno profesional real.

Gracias a mis compañeros y amigos por sus buenos consejos y el apoyo que me han brindado durante este año.

En este capítulo de agradecimientos, no puedo olvidar a mi familia, y en especial a mis padres y hermanas, por quererme, por apoyarme y por recordarme que puedo enfrentarme a los retos de la vida contando siempre con su amor incondicional.

Por supuesto, a Jorge, por creer en mi cuando yo dudaba y por acompañarme en cada paso de este camino.

Por último, no puedo olvidar a mi perrita, Chloe, que tanta compañía me ha hecho en las duras horas de trabajo para culminar este proyecto.

IMPLEMENTACIÓN Y VALIDACIÓN DE UN FRAMEWORK DE AUTOMATIZACIÓN DE LIQUIDACIONES ECONÓMICAS DE *THIRD PARTIES* EN ENTORNO BANCARIO REGULADO

Autor: Sebastián Sánchez, Sofía.
Director: Comesaña Figueiras, Rubén.
Entidad Colaboradora: Accenture.

RESUMEN DEL PROYECTO

Este trabajo presenta el diseño, implementación y validación de un *framework* para automatizar el proceso de liquidación económica de agentes externos en un entorno bancario regulado. La solución, desarrollada mediante Spark Scala sobre Databricks, integra nueve fuentes de datos y aplica las reglas necesarias para calcular retenciones e impuestos. Para su validación se generó un conjunto de datos sintéticos que permitió comprobar ocho escenarios funcionales, obteniéndose resultados conformes en todos ellos. Asimismo, el pipeline procesó 103.936 contratos reales en el entorno de integración en aproximadamente 32 minutos.

Palabras clave: Apache Spark, Databricks, liquidaciones económicas, automatización, datos sintéticos, trazabilidad.

1. Introducción

La transformación digital del sector financiero ha incrementado el volumen y la complejidad de los datos gestionados por las entidades bancarias, haciendo necesaria la automatización de procesos para mejorar la eficiencia, reducir errores y facilitar su control y auditoría.

Uno de estos procesos es la liquidación económica de agentes externos, cuyo cálculo requiere integrar información contractual, personal, fiscal y territorial y aplicar diferentes reglas de negocio. El procedimiento previo, basado principalmente en consultas y operaciones manuales, presentaba limitaciones de escalabilidad, mantenibilidad y trazabilidad.

Para dar respuesta a estas necesidades, el proyecto desarrolla un *framework* automatizado mediante Apache Spark y Scala sobre Databricks. Apache Spark es un motor de procesamiento distribuido orientado al tratamiento de datos a gran escala [1], mientras que Databricks proporciona el entorno de ejecución e integración utilizado en el proyecto.

2. Definición del proyecto

El objetivo general del proyecto es diseñar e implementar un sistema que automatice el proceso de liquidación económica de agentes externos, reduciendo la intervención manual y mejorando la calidad, reproducibilidad y trazabilidad de los resultados.

Para alcanzar este objetivo se definieron las siguientes líneas de trabajo:

- Analizar y formalizar las reglas de negocio del proceso de liquidación.
- Integrar las nueve tablas necesarias para el cálculo.
- Implementar mediante Spark Scala la lógica de retención de agentes residentes y no residentes.
- Calcular los impuestos correspondientes en función de las condiciones contractuales y territoriales.
- Construir las tablas de resultados necesarias para su posterior explotación.
- Desarrollar un mecanismo de prueba que permitiera validar el sistema sin utilizar datos sensibles.
- Evaluar la corrección funcional y el rendimiento del *framework* en un entorno representativo.

El desarrollo se llevó a cabo de forma iterativa debido a la disponibilidad progresiva de accesos, fuentes de datos y permisos de infraestructura. En una primera fase se estudió la arquitectura corporativa existente y el modelo de ejecución de procesos Spark. Posteriormente se analizaron las tablas y sus relaciones, se implementó la lógica de negocio y se realizaron pruebas sobre datos reales en el entorno autorizado.

Finalmente, se desarrolló un generador de datos sintéticos que permitió reproducir los principales escenarios funcionales del proceso. Este generador se diseñó priorizando la cobertura de las reglas de negocio frente a una reproducción exacta de las distribuciones de producción. De este modo, se garantizó la presencia de casos frecuentes y casos límite, como agentes sin retención, agentes no residentes con documentación incompleta o contratos no sujetos a determinados impuestos.

3. Descripción del modelo/sistema/herramienta

La solución se compone de tres capas: fuentes de datos, procesamiento y persistencia.

Las fuentes están formadas por nueve tablas almacenadas que contienen información contractual, personal, fiscal y territorial. El procesamiento se implementa aplicando filtros temporales y diferenciando el cálculo según la residencia fiscal del agente.

Para los agentes residentes se obtiene directamente el porcentaje de retención correspondiente, mientras que para los no residentes se valida la documentación fiscal disponible antes de determinar la retención aplicable. Posteriormente, ambas ramas se unifican para calcular los impuestos territoriales y generar las tablas finales de liquidación.

La ejecución se integra en un *framework* corporativo basado en un DAG, lo que permite organizar las dependencias entre procesos y ejecutar tareas en paralelo cuando es posible.

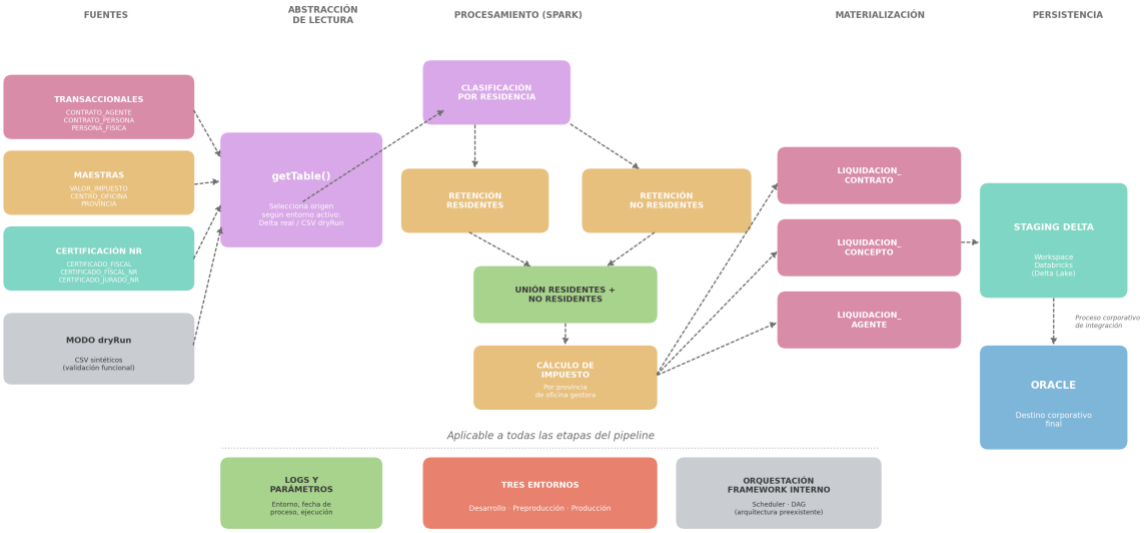


Ilustración 1 - Arquitectura general del framework de automatización de liquidaciones económicas.

Debido a las restricciones de privacidad y protección de datos propias de un entorno bancario [2], se desarrolló un generador de datos sintéticos en Python. Este componente crea conjuntos de prueba reproducibles y coherentes con las reglas de negocio del sistema, permitiendo validar la herramienta reduciendo el riesgo de exposición de información sensible [3].

4. Resultados

El *framework* se validó sobre un conjunto de 1.200 contratos sintéticos, que generaron 1.222 relaciones contrato-agente. Se comprobaron ocho escenarios funcionales relacionados con la retención de residentes y no residentes y con el cálculo del impuesto territorial, obteniéndose resultados conformes en todos los casos analizados.

Asimismo, el pipeline se ejecutó sobre 103.936 contratos reales en el entorno de integración, con un tiempo aproximado de 32 minutos. El bloque de mayor coste computacional fue el cálculo de retenciones para agentes no residentes, debido al uso de varios cruces y funciones de ventana sobre las tablas de certificación.

Ilustración 2 - Panel de indicadores clave de los resultados obtenidos.

5. Conclusiones

El proyecto ha permitido automatizar el proceso de liquidación económica de agentes externos mediante un pipeline desarrollado en Apache Spark y Scala sobre Databricks. La solución integra las fuentes necesarias, aplica de forma sistemática las reglas de retención e impuestos y mejora la reproducibilidad, mantenibilidad y trazabilidad respecto al procedimiento manual anterior. La validación sobre datos sintéticos confirmó el correcto funcionamiento de los principales escenarios definidos, mientras que la ejecución sobre datos reales permitió comprobar la capacidad del *framework* para procesar volúmenes representativos del entorno bancario. Como evolución futura, se plantea ampliar la validación en producción e incorporar mecanismos automáticos de monitorización y alertado.

6. Referencias

[1] Apache Software Foundation, “Apache Spark Documentation”. Disponible en: https://spark.apache.org/docs/

[2] Parlamento Europeo y Consejo de la Unión Europea, “Reglamento (UE) 2016/679 relativo a la protección de datos personales”, 2016.

[3] S. A. Assefa et al., “Generating synthetic data in finance: opportunities, challenges and pitfalls”, ACM International Conference on AI in Finance, 2020.

IMPLEMENTATION AND VALIDATION OF A FRAMEWORK FOR THE AUTOMATION OF THIRD-PARTY FINANCIAL SETTLEMENTS IN A REGULATED BANKING ENVIRONMENT

Author: Sebastián Sánchez, Sofía.
Supervisor: Comesaña Figueiras, Rubén.
Collaborating Entity: Accenture.

ABSTRACT

This project presents the design, implementation and validation of a framework for automating the financial settlement process for external agents in a regulated banking environment. The solution, developed using Apache Spark and Scala on Databricks, integrates nine data sources and applies the business rules required to calculate tax withholdings and taxes. For validation purposes, a synthetic dataset was generated, enabling the verification of eight functional scenarios, with compliant results obtained in all cases. In addition, the pipeline processed 103,936 real contracts in the integration environment in approximately 32 minutes.

Keywords: Apache Spark, Databricks, financial settlements, automation, synthetic data, traceability.

1. Introduction

The digital transformation of the financial sector has increased the volume and complexity of the data managed by banking institutions, making process automation necessary to improve efficiency, reduce errors and facilitate control and auditing.

One such process is the financial settlement of external agents, whose calculation requires the integration of contractual, personal, tax and territorial information, as well as the application of different business rules. The previous procedure, mainly based on manual queries and operations, presented limitations in terms of scalability, maintainability and traceability.

To address these needs, the project develops an automated framework using Apache Spark and Scala on Databricks. Apache Spark is a distributed processing engine designed for large-scale data processing [1], while Databricks provides the execution and integration environment used in the project.

2. Project definition

The main objective of the project is to design and implement a system that automates the financial settlement process for external agents, reducing manual intervention and improving the quality, reproducibility and traceability of the results.

To achieve this objective, the following lines of work were defined:

- Analyse and formalise the business rules governing the settlement process.
- Integrate the nine tables required for the calculations.

- Implement the withholding logic for resident and non-resident agents using Spark Scala.

- Calculate the applicable taxes according to contractual and territorial conditions.

- Build the result tables required for subsequent processing and use.

- Develop a testing mechanism that enables the system to be validated without using sensitive data.

- Evaluate the functional correctness and performance of the framework in a representative environment.

The project was developed iteratively due to the progressive availability of access permissions, data sources and infrastructure authorizations. The first phase involved studying the existing corporate architecture and the Spark process execution model. The tables and their relationships were then analysed, the business logic was implemented, and tests were performed using real data within the authorised environment.

Finally, a synthetic data generator was developed to reproduce the main functional scenarios of the process. The generator was designed to prioritise business-rule coverage over the exact reproduction of production data distributions. This ensured the inclusion of both common and edge cases, such as agents with no withholding, non-resident agents with incomplete documentation and contracts not subject to certain taxes.

3. Description of the model/system/tool

The solution consists of three layers: data sources, processing and persistence.

The data sources consist of nine stored tables containing contractual, personal, tax and territorial information. Processing is implemented applying temporal filters and differentiating the calculation according to the agent's tax residence.

For resident agents, the corresponding withholding percentage is obtained directly. For non-resident agents, the available tax documentation is validated before determining the applicable withholding. Both processing branches are subsequently unified to calculate territorial taxes and generate the final settlement tables.

Execution is integrated into a corporate framework based on a directed acyclic graph, or DAG, which organises dependencies between processes and allows tasks to run in parallel whenever possible.

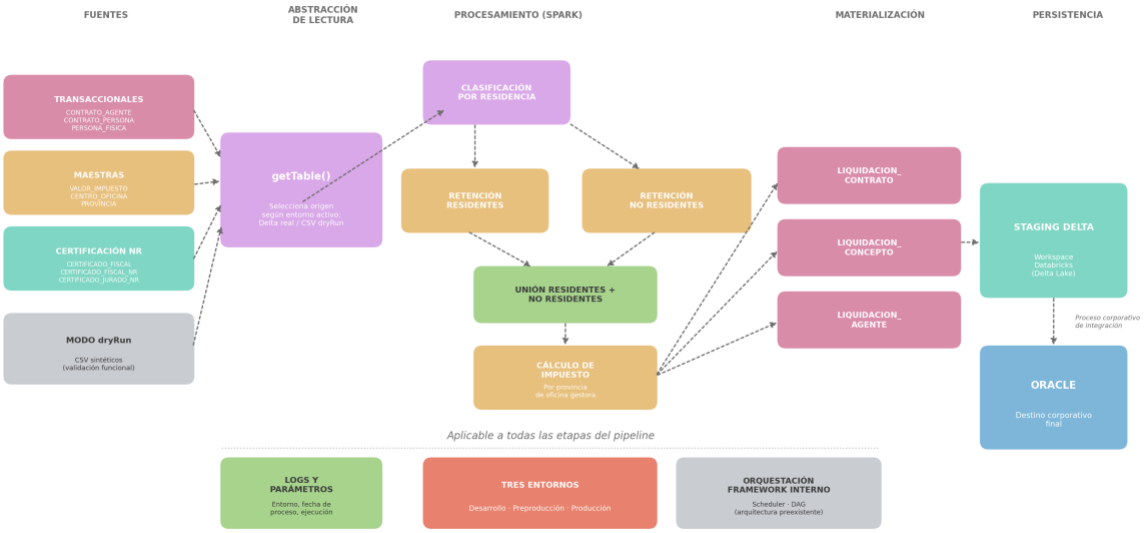


Ilustración 3 - General architecture of the financial settlement automation framework.

Due to the privacy and data-protection restrictions inherent to the banking environment [2], a synthetic data generator was developed in Python. This component creates reproducible test datasets that are consistent with the system's business rules, enabling the framework to be validated while reducing the risk of exposing sensitive information [3].

4. Results

The framework was validated using a set of 1,200 synthetic contracts, which generated 1,222 contract-agent relationships. Eight functional scenarios related to withholding calculations for resident and non-resident agents and territorial tax calculations were verified, with compliant results obtained in all the cases analysed.

The pipeline was also executed on 103,936 real contracts in the integration environment, with an approximate execution time of 32 minutes. The most computationally expensive stage was the calculation of withholdings for non-resident agents, due to the use of multiple joins and window functions on the certification tables.

Ilustración 4 - Key indicators obtained during framework validation and execution.

5. Conclusions

The project has enabled the automation of the financial settlement process for external agents through a pipeline developed using Apache Spark and Scala on Databricks. The solution integrates the required data sources, systematically applies withholding and tax rules, and improves reproducibility, maintainability and traceability compared with the previous manual procedure.

Validation using synthetic data confirmed the correct operation of the main scenarios defined, while execution using real data demonstrated the framework's capacity to process representative volumes from a banking environment. Future developments include extending production validation and incorporating automatic monitoring and alerting mechanisms.

6. References

[1] Apache Software Foundation, "Apache Spark Documentation". Disponible en: https://spark.apache.org/docs/

[2] Parlamento Europeo y Consejo de la Unión Europea, "Reglamento (UE) 2016/679 relativo a la protección de datos personales", 2016.

[3] S. A. Assefa et al., "Generating synthetic data in finance: opportunities, challenges and pitfalls", ACM International Conference on AI in Finance, 2020.

Índice de la memoria

Índice de figuras

Índice de tablas

Capítulo 1. INTRODUCCIÓN

1.1 MOTIVACIÓN DEL PROYECTO

En los últimos años, el sector bancario ha experimentado una profunda transformación impulsada por la digitalización y el creciente protagonismo del dato como activo estratégico. Las entidades financieras gestionan diariamente grandes volúmenes de información procedente de múltiples fuentes, lo que ha generado la necesidad de desarrollar sistemas capaces de procesar, integrar y validar estos datos de forma eficiente. En este contexto, la automatización de procesos se ha convertido en un elemento clave para mejorar la eficiencia operativa, reducir errores y garantizar el cumplimiento de las exigencias regulatorias.

Uno de los ámbitos donde esta necesidad resulta especialmente relevante es en la gestión de las liquidaciones económicas asociadas a agentes externos (*third parties*). Estos procesos implican el cálculo de importes a percibir por diferentes entidades o colaboradores, teniendo en cuenta múltiples factores como contratos, condiciones específicas, retenciones fiscales e impuestos aplicables. Tradicionalmente, estos cálculos han estado soportados por sistemas *legacy* o procesos parcialmente manuales, lo que introduce limitaciones en términos de escalabilidad, trazabilidad y fiabilidad.

La complejidad inherente a estos procesos radica en la necesidad de integrar información procedente de diversas tablas y sistemas, así como en la aplicación de reglas de negocio no triviales, especialmente en lo relativo al cálculo de impuestos y retenciones. Esta situación incrementa el riesgo de errores en las liquidaciones, lo que puede derivar en impactos económicos, problemas operativos y posibles incumplimientos normativos en un entorno altamente regulado como el bancario.

En este contexto, surge la necesidad de diseñar e implementar soluciones que permitan automatizar estos procesos, garantizando al mismo tiempo la calidad y consistencia de los resultados.

El presente trabajo se enmarca en este escenario y tiene como objetivo abordar estas problemáticas mediante el desarrollo de un *framework* de automatización y validación de liquidaciones económicas en un entorno bancario regulado. Para ello, se hace uso de tecnologías propias del paradigma *Big Data*, concretamente Apache Spark como motor de procesamiento distribuido y Databricks como plataforma de ejecución, integradas con bases de datos Oracle para la persistencia de los resultados. La solución propuesta busca no solo mejorar la eficiencia del proceso, sino también proporcionar trazabilidad, robustez y capacidad de adaptación a futuras necesidades del negocio.

1.2 DESCRIPCIÓN DE LAS TECNOLOGÍAS

1.2.1 BIG DATA Y PROCESAMIENTO DISTRIBUIDO

Desde un punto de vista tecnológico, el tratamiento de grandes volúmenes de información ha dado lugar al desarrollo del paradigma *Big Data*, que engloba un conjunto de técnicas y herramientas orientadas a la gestión, almacenamiento y procesamiento eficiente de datos a gran escala [1]. Este paradigma se caracteriza por la necesidad de manejar datos con alto volumen, variedad y velocidad, lo que requiere enfoques distintos a los sistemas tradicionales.

Para dar respuesta a estos desafíos, se han desarrollado arquitecturas de procesamiento distribuido, que permiten dividir las tareas en múltiples unidades de trabajo y ejecutarlas de forma simultánea en distintos nodos de un sistema. Este enfoque se basa en el principio de paralelismo, mediante el cual se optimiza el uso de los recursos computacionales disponibles, reduciendo significativamente los tiempos de procesamiento [2].

El procesamiento distribuido constituye, por tanto, la base de los sistemas modernos de análisis de datos, permitiendo abordar problemas que implican el tratamiento de grandes conjuntos de información y la aplicación de transformaciones complejas. Este tipo de arquitecturas no solo mejora la eficiencia del procesamiento, sino que también aporta

escalabilidad, permitiendo adaptar el sistema a diferentes volúmenes de datos sin necesidad de rediseñar la solución.

En el contexto del presente trabajo, estos enfoques resultan especialmente relevantes, ya que el procesamiento de la información necesaria para el cálculo de las liquidaciones implica la integración y transformación de múltiples fuentes de datos. El uso de técnicas de procesamiento distribuido permite abordar esta complejidad de forma eficiente, proporcionando una base sólida para la implementación de sistemas automatizados y escalables.

En este sentido, la adopción de arquitecturas de datos adecuadas resulta fundamental para organizar, almacenar y explotar eficientemente la información procesada, lo que ha dado lugar a la evolución de distintos modelos arquitectónicos en los sistemas de gestión de datos.

1.2.2 ARQUITECTURAS MODERNAS DE DATOS

Las arquitecturas de datos han evolucionado significativamente en los últimos años para adaptarse al crecimiento del volumen, variedad y complejidad de la información generada en entornos empresariales. En sus primeras etapas, los sistemas de información se basaban principalmente en *data warehouses*, diseñados para almacenar datos estructurados y optimizados para la realización de consultas analíticas. Estos sistemas permitían consolidar información procedente de distintas fuentes, garantizando consistencia y calidad del dato, aunque presentaban limitaciones en términos de flexibilidad y escalabilidad ante grandes volúmenes de datos heterogéneos [3].

Con la aparición del paradigma *Big Data*, surgieron nuevas soluciones como los *data lakes* que permiten almacenar grandes cantidades de datos en su formato original, sin necesidad de una estructura previa definida. Este enfoque facilita la integración de datos estructurados y no estructurados, proporcionando mayor flexibilidad y capacidad de almacenamiento, aunque introduce nuevos retos relacionados con la gobernanza y la calidad de los datos, ya que una gestión inadecuada puede provocar que el repositorio se convierta en un *data swamp* [4].

Más recientemente, han emergido arquitecturas híbridas como el *data lakehouse*, que combinan las ventajas de los *data lakes* y los *data warehouses*. Este modelo permite mantener la flexibilidad del almacenamiento masivo de datos, al tiempo que incorpora mecanismos de control, consistencia y optimización de consultas propios de los sistemas tradicionales [5]. De este modo, el enfoque *lakehouse* se plantea como una solución adecuada para entornos que requieren tanto procesamiento a gran escala como capacidades analíticas avanzadas.

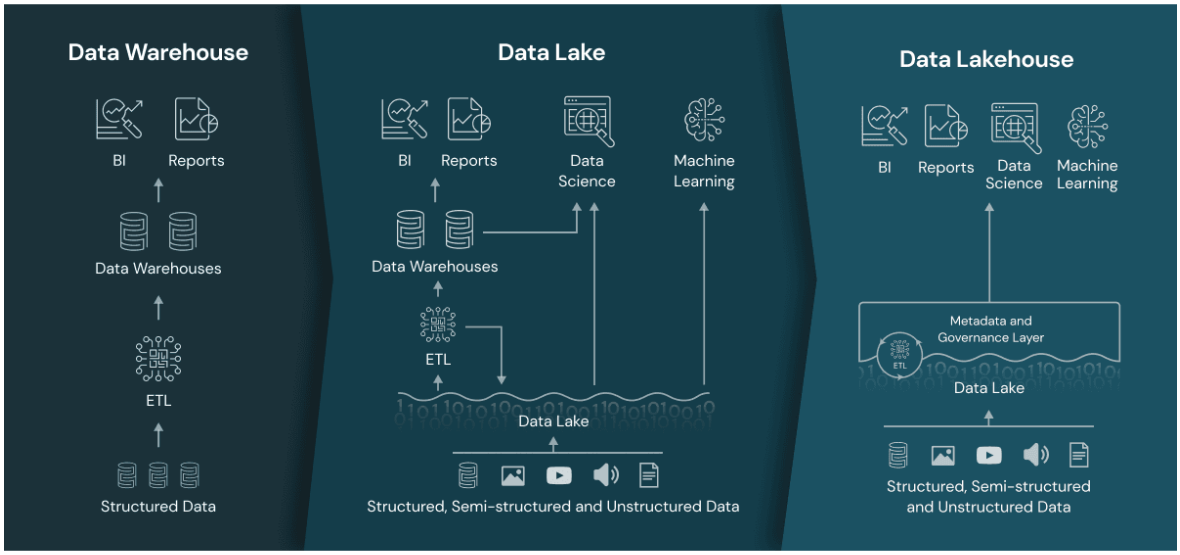


Figura 1. Evolución de las arquitecturas de datos: Data Warehouse, Data Lake y Lakehouse [13]

En el marco de esta evolución, la plataforma Databricks desempeña un papel fundamental al ser la impulsora del concepto *Lakehouse* mediante el uso de *Delta Lake*. Esta tecnología añade una capa de almacenamiento optimizada que permite dotar al *data lake* de capacidades de transacciones ACID (Atomicidad, Consistencia, Aislamiento y Durabilidad). Para un sistema de liquidaciones financieras, esta característica es crítica, ya que garantiza la integridad de los datos ante fallos en los procesos de escritura y permite el control de versiones de los archivos (*time travel*), asegurando que los cálculos de impuestos y retenciones se realicen siempre sobre estados de datos consistentes y auditables.

Estas arquitecturas resultan fundamentales en el presente trabajo, ya que permiten gestionar grandes volúmenes de datos procedentes de múltiples fuentes, proporcionando una base sólida sobre la que construir sistemas de procesamiento y análisis orientados a la automatización de procesos complejos.

1.2.3 PIPELINES DE DATOS Y PARADIGMAS ETL/ELT

El procesamiento de datos en entornos *Big Data* se articula mediante *pipelines* de datos, que representan el conjunto de procesos encargados de transportar, transformar y preparar la información desde sus fuentes originales hasta su destino final. Estos *pipelines* constituyen un elemento fundamental en las arquitecturas modernas de datos, ya que permiten estructurar el flujo de información de forma organizada, reproducible y escalable.

Tradicionalmente, los procesos de integración de datos se han implementado mediante el paradigma ETL (*Extract, Transform, Load*), en el cual los datos se extraen de las fuentes de origen, se transforman de acuerdo con las reglas de negocio y los requisitos de calidad y, posteriormente, se cargan en un sistema de almacenamiento de destino, como un *data warehouse*. Sin embargo, en escenarios caracterizados por grandes volúmenes de información, la transformación previa a la carga puede convertirse en un cuello de botella y limitar la flexibilidad del proceso [6].

Con la aparición de tecnologías *Big Data* y sistemas de almacenamiento más flexibles, ha cobrado relevancia el paradigma *ELT (Extract, Load, Transform)*, en el que los datos se cargan primero en su formato original y las transformaciones se realizan posteriormente utilizando motores de procesamiento distribuido. Este enfoque permite aprovechar la capacidad de cómputo de plataformas como Apache Spark, facilitando el tratamiento de grandes volúmenes de datos y la aplicación de transformaciones complejas de forma eficiente [7].

En este contexto, los *pipelines* de datos se estructuran habitualmente en varias etapas: ingesta de datos desde diferentes fuentes, almacenamiento en bruto, procesamiento y transformación, y finalmente carga en sistemas de destino preparados para su explotación.

Este modelo permite desacoplar las distintas fases del procesamiento, mejorando la mantenibilidad, escalabilidad y trazabilidad del sistema.

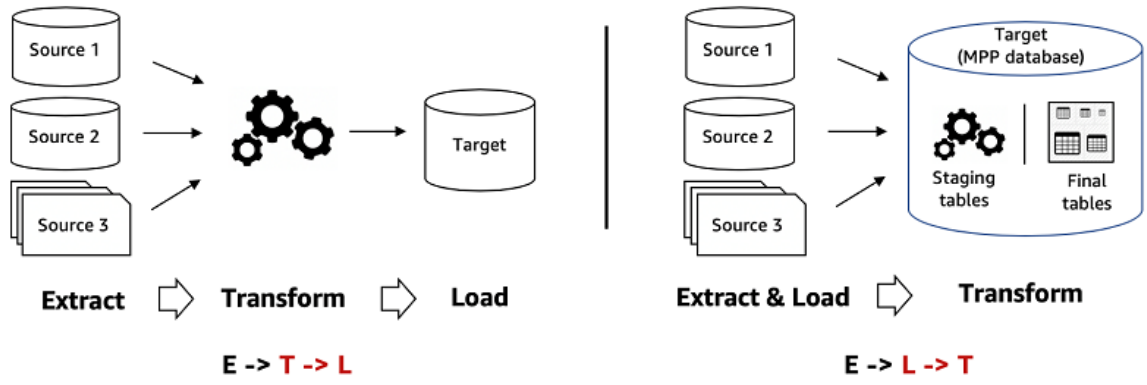


Figura 2. Comparación entre los paradigmas ETL y ELT [8]

Para este sistema se adopta un enfoque ELT, en el que los datos se ingieren y transforman en Databricks mediante Apache Spark, cargándose finalmente en Oracle para su explotación.

1.2.4 AUTOMATIZACIÓN DE PROCESOS FINANCIEROS

Los *pipelines* de datos no solo permiten estructurar y procesar la información de forma eficiente, sino que también constituyen el soporte sobre el cual se implementa la lógica de negocio necesaria para transformar los datos en resultados útiles. En este sentido, las etapas de transformación dentro de un *pipeline* representan el punto en el que se aplican reglas y criterios que determinan el comportamiento del sistema.

En el ámbito financiero, la automatización de procesos ha evolucionado mediante la incorporación de distintos enfoques tecnológicos. Entre ellos se encuentra la Automatización Robótica de Procesos (*Robotic Process Automation, RPA*), que permite automatizar tareas repetitivas y basadas en reglas mediante robots de software que interactúan con las interfaces de los sistemas existentes, imitando determinadas acciones realizadas por los usuarios. Si bien el RPA ha supuesto una mejora significativa respecto a la ejecución manual de estas tareas, presenta limitaciones cuando los procesos incorporan una lógica de negocio

compleja, excepciones frecuentes o cambios en las aplicaciones con las que interactúa. Su funcionamiento se basa principalmente en reglas y secuencias previamente definidas, por lo que su capacidad de adaptación a escenarios variables es limitada [9].

Para gestionar procesos cuya lógica de negocio es compleja y está sujeta a cambios, han cobrado relevancia los sistemas basados en motores de reglas de negocio (*business rules engines*). Estas soluciones permiten definir, administrar y ejecutar de forma sistemática las reglas que determinan las decisiones y el comportamiento de un proceso. La externalización de dichas reglas respecto al código principal de la aplicación favorece la separación de responsabilidades y facilita su mantenimiento y modificación ante cambios en los requisitos [10].

No obstante, las reglas de negocio también pueden implementarse directamente en las etapas de transformación de un pipeline de datos, sin necesidad de utilizar un motor de reglas independiente. En estos casos, motores de procesamiento distribuido como Apache Spark permiten aplicar condiciones, cruces y transformaciones sobre grandes conjuntos de datos mediante su ejecución paralela en los distintos nodos del clúster.

Asimismo, la automatización contribuye a mejorar la consistencia y reproducibilidad de los resultados, al reducir la variabilidad y el riesgo de error asociados a la intervención manual. También facilita la adaptación del sistema ante incrementos en el volumen de datos o en la complejidad de las reglas aplicadas. Estas características resultan especialmente relevantes en entornos regulados como el financiero, donde la trazabilidad y la posibilidad de auditar los cálculos constituyen requisitos fundamentales.

1.2.5 CALIDAD DEL DATO Y VALIDACIÓN

La automatización de procesos basada en reglas de negocio requiere garantizar la calidad y fiabilidad de la información procesada. La calidad del dato (*data quality*) se define como el conjunto de propiedades que determinan si los datos son adecuados para su uso, incluyendo aspectos como la precisión, consistencia, integridad y completitud [11].

En entornos donde se ejecutan procesos automatizados, la calidad del dato adquiere una especial relevancia ya que cualquier error o inconsistencia en los datos de entrada puede propagarse a lo largo del sistema, afectando directamente a los resultados obtenidos. Por este motivo, resulta necesario incorporar mecanismos que permitan detectar, corregir y prevenir posibles anomalías en las distintas etapas del procesamiento.

La validación de datos constituye uno de los principales enfoques para garantizar esta calidad, y puede llevarse a cabo mediante diferentes estrategias, como la verificación de integridad, la comprobación de reglas de negocio o la comparación de resultados con referencias previamente conocidas.

La validación no solo permite asegurar la corrección de los resultados, sino que también contribuye a mejorar la trazabilidad del sistema, facilitando la identificación del origen de posibles errores y permitiendo auditar el comportamiento del proceso. De este modo, la combinación de automatización y validación constituye un elemento clave en el desarrollo de sistemas robustos y fiables.

1.2.6 ÉTICA DE DATOS Y ENTORNOS DE PRUEBA CONTROLADOS

La implementación de sistemas basados en datos en entornos financieros requiere considerar aspectos relacionados con la privacidad, la seguridad de la información y el cumplimiento normativo. En particular, regulaciones como el Reglamento General de Protección de Datos (GDPR) establecen restricciones estrictas sobre el uso de datos personales, lo que condiciona el desarrollo y validación de soluciones en estos contextos [12].

En este sentido, el uso de datos sintéticos se ha consolidado como una práctica habitual en entornos de prueba controlados, permitiendo trabajar con conjuntos de datos que reproducen las características estadísticas y las relaciones lógicas de los datos reales, reduciendo el riesgo de exposición de información sensible o identificable, si bien su idoneidad depende del diseño concreto del proceso de generación. Desde un punto de vista técnico, la generación de datos sintéticos puede abordarse mediante diferentes enfoques, que van desde métodos estadísticos clásicos, como el muestreo basado en distribuciones de probabilidad,

hasta técnicas más avanzadas basadas en modelos generativos. En cualquier caso, el objetivo común es preservar la estructura y las propiedades del dato original, garantizando que los sistemas desarrollados puedan ser evaluados en condiciones representativas reduciendo el riesgo de exposición de información sensible.

El empleo de datos sintéticos resulta especialmente útil en procesos de validación, ya que permite generar escenarios específicos, incluyendo casos límite o situaciones poco frecuentes en los datos históricos. Esto contribuye a evaluar de forma más exhaustiva el comportamiento del sistema, mejorando su robustez y fiabilidad.

De este modo, la combinación de un marco normativo claro y el uso de datos sintéticos como herramienta técnica permite abordar el desarrollo de soluciones complejas en entornos regulados, sin necesidad de operar sobre información real identificable.

1.3 REVISIÓN DE SOLUCIONES TECNOLÓGICAS

La automatización de procesos de liquidación económica en entornos financieros no es un problema nuevo, y existen en el mercado diversas soluciones tecnológicas orientadas a dar respuesta a esta necesidad. Sin embargo, el grado de adecuación de estas soluciones varía significativamente en función de los requisitos específicos de cada entidad, especialmente cuando el proceso involucra reglas de negocio complejas, integración con sistemas propios y grandes volúmenes de datos. En este apartado se revisan las principales alternativas existentes y se analiza su idoneidad para el contexto del presente trabajo.

1.3.1 PLATAFORMAS COMERCIALES DE GESTIÓN FINANCIERA

Entre las soluciones comerciales implantadas en el sector financiero para la gestión de operaciones de tesorería, mercados de capitales, riesgo y pagos se encuentran plataformas como Murex, Calypso y Finastra. Estas herramientas ofrecen funcionalidades especializadas para gestionar instrumentos financieros, posiciones, operaciones, liquidaciones y requisitos de control y cumplimiento normativo.

No obstante, se trata de plataformas empresariales de amplio alcance, diseñadas para cubrir procesos financieros completos y altamente estandarizados. Por ello, su implantación para automatizar un proceso concreto y basado en una lógica interna propia de la entidad puede requerir tareas adicionales de configuración, personalización e integración con los sistemas existentes. En el caso analizado en este trabajo, donde los datos se procesan dentro de un ecosistema basado en Databricks y Apache Spark, la implementación directa de la lógica de negocio en los pipelines de transformación permite mantener el procesamiento próximo al dato y aprovechar de forma inmediata la infraestructura distribuida disponible.

1.3.2 SOLUCIONES DE AUTOMATIZACIÓN ROBÓTICA DE PROCESOS (RPA)

Como se mencionó en el apartado 1.2.4, el RPA ha sido ampliamente adoptado en el sector financiero como mecanismo para automatizar tareas repetitivas basadas en la interacción con sistemas existentes. No obstante, su aplicabilidad en el contexto de las liquidaciones económicas es limitada: el RPA opera a nivel de interfaz y puede presentar dificultades cuando el proceso exige integrar múltiples fuentes de datos, manejar volúmenes elevados de información o aplicar reglas de negocio complejas y variables. En este tipo de escenarios, el RPA puede resultar útil para automatizar tareas auxiliares o interacciones concretas con aplicaciones, pero no constituye la alternativa más adecuada como núcleo del procesamiento.

1.3.3 DESARROLLO A MEDIDA SOBRE SOLUCIONES BIG DATA

Ante las limitaciones de las soluciones comerciales y del RPA para el caso de uso concreto, el desarrollo de una solución a medida sobre tecnologías *Big Data* se presenta como una alternativa adecuada. Este enfoque permite adaptar completamente la lógica de negocio a los requisitos específicos del proceso, aprovechar la infraestructura de procesamiento distribuido ya existente en la entidad y facilitar la integración nativa con los sistemas de origen y destino de los datos.

En este sentido, Apache Spark constituye una opción apropiada como motor de procesamiento, debido a su capacidad para ejecutar transformaciones distribuidas sobre grandes volúmenes de información [7]. Por su parte, Databricks proporciona un entorno de

ejecución sobre Spark que incorpora funcionalidades para el desarrollo, la gestión y la orquestación de pipelines de datos, así como para el trabajo colaborativo en entornos empresariales [14].

La siguiente tabla resume de forma comparativa las principales características de las alternativas analizadas en relación con los requisitos del presente trabajo:

	Plataformas comerciales	*RPA*	*Desarrollo a medida*
Adaptabilidad a reglas propias	Baja	Media	Alta
Integración con ecosistemas Big Data	Baja	Baja	Alta
Escalabilidad	Alta	Baja	Alta
Coste de implementación	Alto	Medio	Medio
Trazabilidad del dato	Media	Baja	Alta

Tabla 1. Comparativa de soluciones tecnológicas para la automatización de liquidaciones económicas.

De acuerdo con esta comparación, el desarrollo de un *framework* a medida sobre Apache Spark y Databricks resulta la alternativa que mejor se ajusta a los requisitos del presente trabajo. Esta opción ofrece una elevada flexibilidad para implementar las reglas de negocio y permite aprovechar la infraestructura tecnológica ya existente en la entidad, reduciendo la necesidad de incorporar una plataforma adicional.

Capítulo 2. DEFINICIÓN DEL TRABAJO

2.1 OBJETIVOS

2.1.1 OBJETIVO GENERAL

El objetivo principal del presente trabajo es diseñar e implementar un *framework* que automatice el proceso de liquidación económica de agentes externos en un entorno bancario regulado, reduciendo la intervención manual, mejorando la eficiencia operativa y favoreciendo la calidad y trazabilidad del dato a lo largo de todo el proceso.

2.1.2 OBJETIVOS ESPECÍFICOS

2.1.2.1 Análisis y formalización del proceso de liquidación económica

Analizar el proceso actual de liquidación económica de terceros para identificar sus reglas de negocio, dependencias de datos y limitaciones operativas. Este análisis constituye la base sobre la que se define la lógica de negocio a implementar, incluyendo el tratamiento de contratos, agentes, retenciones fiscales e impuestos aplicables.

2.1.2.2 Implementación del pipeline sobre arquitectura existente

Implementar el pipeline de procesamiento del flujo de liquidación económica sobre la infraestructura *Big Data* existente en la entidad, integrando las etapas de lectura de fuentes, transformación de datos mediante Apache Spark y la construcción y persistencia de las tablas de resultados. El desarrollo se orienta a construir una solución modular y mantenible que pueda evolucionar hacia una arquitectura dedicada en una fase estratégica posterior.

2.1.2.3 Validación y trazabilidad del proceso

Incorporar mecanismos que permitan verificar y justificar los cálculos realizados por el *framework*, mediante el registro de los parámetros de ejecución, la identificación de las fuentes utilizadas y la conservación de los resultados generados para cada fecha de proceso.

Estos mecanismos deben mejorar la trazabilidad operacional y facilitar la revisión y auditoría del proceso dentro del entorno bancario.

2.1.2.4 Verificación del framework mediante pruebas controladas

Diseñar y ejecutar pruebas sobre el *framework* para comprobar su correcto funcionamiento en distintos escenarios, incluyendo casos representativos y casos límite. Para ello se utilizarán datos sintéticos que reproduzcan la estructura y lógica de los datos reales, permitiendo validar la solución y reducir el riesgo de exposición de información sensible.

2.1.2.5 Evaluación de la aplicabilidad del framework

Evaluar la capacidad del sistema para procesar un volumen de datos representativo del entorno bancario y determinar su adecuación como base reutilizable para la automatización de procesos de liquidación similares. Asimismo, se pretende justificar la idoneidad de Apache Spark y Databricks para implementar una solución distribuida, modular y adaptable a futuras necesidades del negocio.

2.2 METODOLOGÍA

El desarrollo del trabajo se ha llevado a cabo de forma iterativa e incremental, condicionado en gran medida por la disponibilidad progresiva de accesos, datos e información por parte del equipo del banco. Este enfoque, habitual en proyectos desarrollados en entornos corporativos reales, implica que las distintas fases del trabajo no siempre siguen una secuencia estrictamente lineal, sino que avanzan en función de las dependencias externas y del conocimiento adquirido en cada etapa.

A continuación, se describen las principales fases que han estructurado el desarrollo del proyecto:

2.2.1 FASE 1: COMPRENSIÓN DEL ENTORNO Y LA ARQUITECTURA EXISTENTE

La primera fase consistió en la toma de contacto con el entorno tecnológico del proyecto. Tras obtener acceso al repositorio, se analizó en detalle la estructura del repositorio y los distintos componentes del *framework* interno de orquestación de procesos Spark ya existente en la entidad, sobre el cual se integraría la solución desarrollada en el presente trabajo. Dicha arquitectura incluye elementos como el Scheduler, los UEParallelJob, los Stage y el Controller, los cuales se explicarán más adelante en la memoria.

2.2.2 FASE 2: ANÁLISIS Y COMPRENSIÓN DE LAS FUENTES DE DATOS

Una vez comprendida la arquitectura, se procedió al análisis de las tablas de datos necesarias para el proceso de liquidación. Para ello se utilizó una herramienta interna de visualización desarrollada sobre Power BI, que permitió explorar la estructura de cada tabla, sus columnas y las relaciones entre ellas. Paralelamente, se gestionaron las solicitudes de acceso mediante reglas de firewall necesarias para poder operar con dichas fuentes.

2.2.3 FASE 3: IMPLEMENTACIÓN DE LA LÓGICA DE NEGOCIO

Con el conocimiento adquirido en las fases anteriores, se procedió a implementar la lógica de negocio del proceso de liquidación mediante Apache Spark utilizando Scala. Este desarrollo incluye el cálculo de retenciones para residentes y no residentes, así como el cálculo de impuestos.

Esta fase constituyó el núcleo del desarrollo técnico del *framework* y dio lugar al *DataFrame* consolidado utilizado posteriormente para construir las tablas finales de liquidación.

2.2.4 FASE 4: PRUEBAS CON DATOS REALES

Una vez implementada la lógica, se realizaron pruebas sobre datos reales para verificar el correcto funcionamiento del cálculo y detectar posibles errores o casos no contemplados en la lógica inicial.

2.2.5 FASE 5: GENERACIÓN DE DATOS SINTÉTICOS

Con el objetivo de validar y presentar los resultados reduciendo el riesgo de exposición de información sensible, se analizaron en Databricks las distribuciones y características de los campos relevantes de las fuentes reales.

Inicialmente, se evaluó el uso de una API interna de generación de datos sintéticos. Al no encontrarse disponible durante el periodo de desarrollo, se optó por implementar un generador propio en Python. Este generador reproduce la estructura de las nueve tablas, sus relaciones y las principales características funcionales necesarias para cubrir los escenarios de validación.

2.2.6 FASE 6: VALIDACIÓN DEL *FRAMEWORK*

Se validó la coherencia interna del conjunto de datos sintéticos y se ejecutó el pipeline sobre dichas fuentes. La validación de los datos incluyó la comprobación de las distribuciones definidas, la integridad referencial y la ausencia de registros huérfanos entre las tablas relacionadas.

Posteriormente, se verificó el comportamiento funcional del *framework* en escenarios representativos y casos límite relacionados con las retenciones de agentes residentes y no residentes y con el cálculo del impuesto territorial. También se comprobó la reproducibilidad de la fase de cálculo mediante ejecuciones consecutivas sobre las mismas entradas y parámetros.

2.2.7 FASE 7: CARGA DE TABLAS FINALES

La última fase comprendió la construcción de las tres tablas finales del proceso de liquidación y su materialización inicial en el esquema de trabajo de Databricks en formato Delta Lake. Esta capa intermedia permitió realizar comprobaciones sobre los resultados antes de su envío al sistema de destino.

Posteriormente, las tablas fueron cargadas en la base de datos Oracle corporativa, donde quedaron disponibles para su explotación por los sistemas consumidores. Esta integración final fue realizada mediante los mecanismos corporativos establecidos por los equipos responsables, por lo que su implementación técnica no formó parte del desarrollo realizado directamente en el presente trabajo.

2.3 PLANIFICACIÓN

El desarrollo del proyecto se llevó a cabo a lo largo de las semanas comprendidas entre marzo y mayo de 2026, estructurándose en las fases descritas en el apartado de metodología. La Figura 3 recoge la planificación temporal seguida, donde se refleja la distribución de las actividades realizadas.

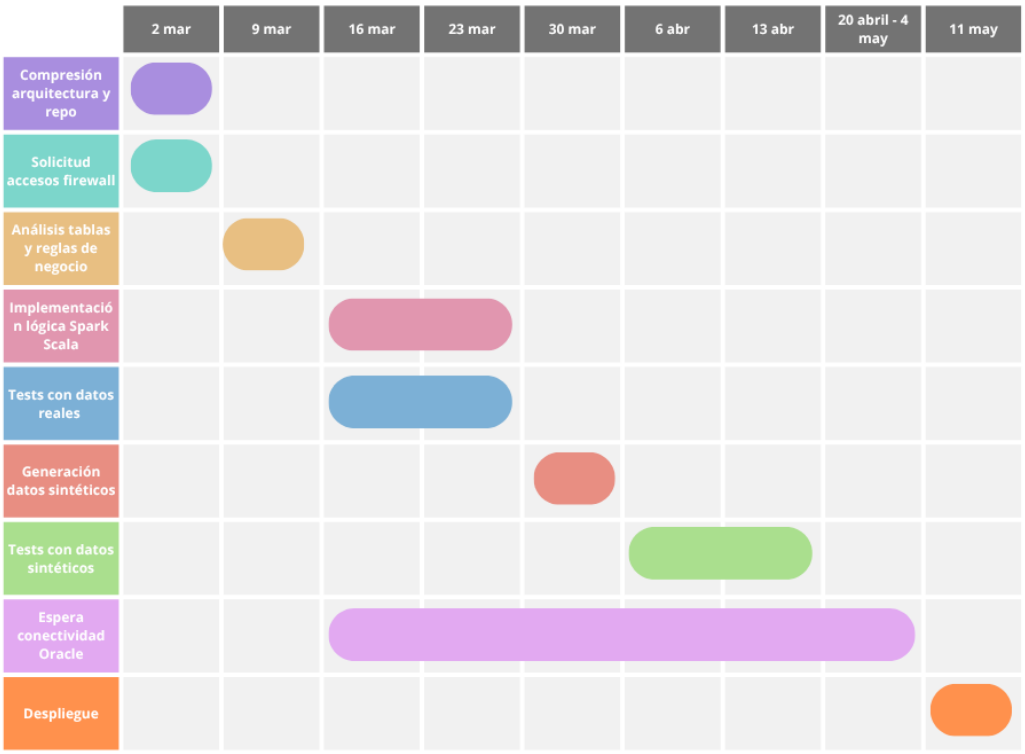


Figura 3. Desarrollo temporal del proyecto. Elaboración propia.

Capítulo 3. SISTEMA/MODELO DESARROLLADO

3.1 ANÁLISIS DEL SISTEMA

3.1.1 DESCRIPCIÓN DEL PROCESO DE LIQUIDACIÓN ECONÓMICA

El proceso de liquidación económica de terceros constituye uno de los procesos financieros más relevantes en el ámbito bancario, siendo su objetivo el cálculo de los importes a percibir por los agentes colaboradores externos en función de los contratos suscritos con la entidad. Estos agentes, denominados en adelante terceros, pueden ser tanto personas físicas como jurídicas, y su relación contractual con el banco determina las condiciones económicas aplicables en cada liquidación.

El proceso implica la integración de información procedente de múltiples fuentes: datos contractuales, información personal de los agentes, parámetros fiscales y datos territoriales. A partir de esta información, el sistema debe calcular dos magnitudes fundamentales para cada contrato: la retención aplicable al agente y el impuesto correspondiente según su situación fiscal y territorial.

El cálculo de la retención depende de la condición de residencia del agente. Para los agentes residentes en España, la retención se determina directamente a partir del código de retención asociado al contrato, consultando el porcentaje correspondiente en las tablas de parámetros fiscales. Para los agentes no residentes, el proceso es más complejo, ya que requiere verificar la existencia y vigencia de certificados fiscales y declaraciones juradas que acrediten su situación, determinando el porcentaje de retención aplicable en función de la documentación disponible.

El cálculo del impuesto, por su parte, depende de la provincia donde se ubica la oficina gestora del contrato y de si el agente está sujeto a dicho impuesto. A partir del código de

provincia de la oficina, se determina el tipo impositivo aplicable, y con este se consulta el porcentaje vigente en la fecha de liquidación.

Tradicionalmente, este proceso ha sido ejecutado de forma manual o mediante sistemas *legacy* con escasa capacidad de automatización, lo que introduce riesgos de error, dificulta la trazabilidad de los cálculos y limita la escalabilidad ante incrementos en el volumen de contratos. El *framework* desarrollado tiene como objetivo automatizar íntegramente este proceso, mejorando la reproducibilidad, trazabilidad y calidad de los resultados en un entorno bancario regulado.

3.1.2 FUENTES DE DATOS

El proceso de liquidación económica requiere la integración de información procedente de nueve tablas, que pueden clasificarse en tres categorías según su naturaleza: tablas transaccionales, tablas maestras y tablas de certificación de no residentes.

Todas las tablas se encuentran almacenadas en Databricks en formato Delta Lake e incluyen la columna técnica *data_date_part*, que representa la fecha de los datos procesados. Esta columna no forma parte de la lógica de negocio, sino que se utiliza para seleccionar la partición temporal correspondiente y reducir el volumen de información leído en cada ejecución. Su presencia es común a todas las tablas y debe tenerse en cuenta al acceder a ellas, por lo que el proceso aplica un filtro sobre esta columna para limitar la lectura a la fecha de datos correspondiente.

3.1.2.1 Tablas transaccionales

Las tablas transaccionales contienen la información operativa del proceso, es decir, los datos relativos a los contratos y a las personas que intervienen en ellos.

La tabla *CONTRATO_AGENTE* almacena la información de los contratos suscritos entre la entidad y los agentes colaboradores. Sus campos relevantes para el proceso son el identificador del contrato (compuesto por empresa, centro, producto y número de contrato), el código de retención aplicable al agente (*retenalf*) y el indicador de sujeción al impuesto

(*fisij*). Constituye la tabla central del proceso, a partir de la cual se articulan el resto de las consultas y cálculos.

La tabla *CONTRATO_PERSONA* recoge la relación entre las personas y los contratos en los que intervienen, aportando el tipo y código de persona necesarios para identificar al agente y acceder a su información personal. Sus claves son el identificador del contrato y el identificador de la persona, junto con el tipo de intervención y el orden de la misma.

La tabla *PERSONA_FISICA* contiene los datos básicos de las personas físicas, siendo el campo de país de residencia (*paisresi*) el de mayor relevancia para el proceso, ya que determina si el agente debe ser tratado como residente o no residente a efectos fiscales.

3.1.2.2 Tablas maestras y paramétricas

Las tablas maestras proporcionan información de referencia utilizada en los cálculos fiscales y territoriales.

La tabla *VALOR_IMPUESTO* es la tabla paramétrica central del proceso fiscal. Contiene los porcentajes de retención e impuesto aplicables en función del tipo de impuesto (*tipimpos*) y del identificador del impuesto (*idimpto*), con control de vigencia temporal mediante fechas de inicio y fin de vigencia (*fecinvig, fecfivig*). Es consultada en tres momentos distintos del proceso: para la retención de residentes, para la retención de no residentes y para el cálculo del impuesto.

La tabla *CENTRO_OFICINA* contiene la información de las oficinas gestoras de los contratos, aportando el código de provincia (*codprov*) que permite determinar el ámbito territorial del impuesto aplicable.

La tabla *PROVINCIA* relaciona cada código de provincia con su tipo impositivo (*tipimpos*), siendo el nexo entre la información territorial del contrato y los parámetros fiscales de la tabla *VALOR_IMPUESTO*.

3.1.2.3 Tablas de certificación de no residentes

Estas tablas contienen la documentación fiscal asociada a los agentes no residentes y son consultadas secuencialmente para determinar el porcentaje de retención aplicable.

La tabla *CERTIFICADO_FISCAL* registra la existencia de un certificado fiscal activo para una persona no residente, identificada por tipo y código de persona. Su presencia en esta tabla es el primer criterio de verificación en el flujo de cálculo para no residentes.

La tabla *CERTIFICADO_FISCAL_NR* almacena los certificados fiscales de no residentes con sus fechas de inicio y fin de vigencia, permitiendo verificar si el agente dispone de un certificado vigente en la fecha de liquidación.

La tabla *CERTIFICADO_JURADO_NR* recoge las declaraciones juradas de no residentes, con una estructura análoga a la anterior. Su consulta se realiza de forma complementaria a la de *CERTIFICADO_FISCAL_NR*, siendo necesaria la presencia del agente en ambas tablas para aplicar el tipo de retención reducido.

La siguiente tabla resume las características principales de cada fuente de datos:

Tabla	*Tipo*	*Campos clave*	*Propósito en el proceso*
CONTRATO_AGENTE	Transaccional	idempr, idcent, idprod, idcontr	Datos del contrato y parámetros fiscales del agente
CONTRATO_PERSONA	Transaccional	tipo_pers, cod_pers, idcontr	Relación persona-contrato

PERSONA_FISICA	Transaccional	tipo_pers, cod_pers	País de residencia del agente
VALOR_IMPUESTO	Maestra	tipimpos, idimpto, fecvig	Porcentajes fiscales vigentes
CENTRO_OFICINA	Maestra	idempr, idcent	Provincia de la oficina gestora
PROVINCIA	Maestra	codprov	Tipo de impositivo por territorio
CERTIFICADO_FISCAL	Certificación NR	tipers, codpers	Existencia de certificado fiscal NR
CERTIFICADO_FISCAL_NR	Certificación NR	tipopers, codpers, fecini	Vigencia del certificado fiscal NR
CERTIFICADO_JURADO_NR	Certificación NR	tipopers, codpers, fecini	Vigencia de la declaración jurada NR

Tabla 2. Características principales de cada tabla del modelo de datos.

3.1.3 LÓGICA DE NEGOCIO

El núcleo del proceso de liquidación económica reside en la aplicación de un conjunto de reglas de negocio que determinan, para cada contrato, los porcentajes de retención e impuesto correspondientes. Estas reglas se organizan en tres bloques principales: el cálculo de la retención para agentes residentes, el cálculo de la retención para agentes no residentes y la determinación del impuesto aplicable.

Cada bloque utiliza información contractual, fiscal y territorial procedente de distintas fuentes corporativas. Las reglas se aplican teniendo en cuenta la situación del agente y la vigencia temporal de los datos en la fecha de liquidación.

La Figura 4 muestra de forma simplificada el flujo general del proceso.

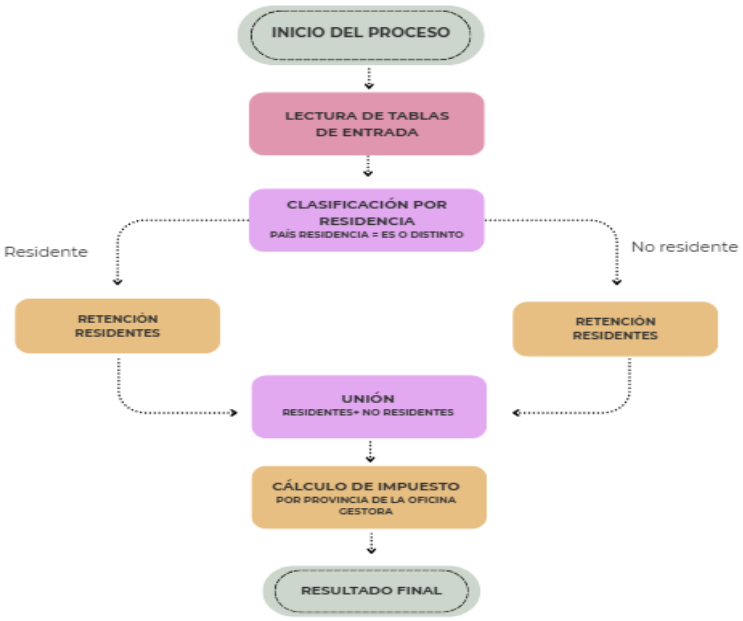


Figura 4. Flujo general del proceso de liquidación económica. Elaboración propia.

3.1.3.1 Cálculo de la retención para residentes

La identificación de los agentes residentes se realiza a partir de la información personal asociada al contrato y, en particular, de su país de residencia fiscal.

Para este colectivo, el cálculo comienza con la consulta de la información fiscal registrada en el contrato. En primer lugar, el *framework* determina si existe una retención aplicable. Cuando el contrato no se encuentra sujeto a retención o no dispone de un identificador fiscal válido, el porcentaje se establece en cero.

Cuando existe una retención aplicable, el valor registrado en el contrato se utiliza para localizar la parametrización correspondiente en la tabla *VALOR_IMPUESTO*. La consulta considera la categoría fiscal asociada y selecciona únicamente los registros cuya vigencia comprende la fecha de liquidación.

Este diseño evita incorporar los porcentajes directamente en el código y permite que los valores aplicables se obtengan de forma dinámica desde las fuentes paramétricas de la entidad. De este modo, una actualización de los porcentajes o de su periodo de vigencia no requiere modificar la estructura principal del *pipeline*.

3.1.3.2 Cálculo de la retención para no residentes

Los agentes no residentes se identifican mediante la información de residencia fiscal disponible en la tabla *PERSONA_FISICA*. En este caso, el cálculo incorpora una serie de validaciones adicionales, ya que el tratamiento fiscal depende tanto de la existencia como de la vigencia de la documentación asociada al agente.

En primer lugar, el *framework* comprueba la disponibilidad de información fiscal en las fuentes de certificación. A continuación, consulta las tablas documentales de no residentes y selecciona, para cada agente, los registros temporalmente relevantes. Cuando existen varios registros, se prioriza el más reciente y se verifica que continúe vigente en la fecha de liquidación.

El resultado conjunto de estas comprobaciones permite clasificar al agente según su situación documental. Cuando la documentación requerida se encuentra disponible, completa y vigente, se selecciona el tratamiento fiscal correspondiente. En caso de ausencia, caducidad o falta de alguno de los elementos necesarios, se aplica el tratamiento alternativo establecido por la entidad.

Una vez determinada la categoría fiscal aplicable, el porcentaje de retención se recupera de la tabla *VALOR_IMPUESTO*, filtrando nuevamente los registros según su vigencia temporal. De este modo, la lógica combina validaciones documentales, selección de registros por fecha y consulta de parámetros fiscales externos.

La Figura 5 representa de forma simplificada el flujo de decisión aplicado a los agentes no residentes, omitiendo los códigos y condiciones específicas de la parametrización interna.

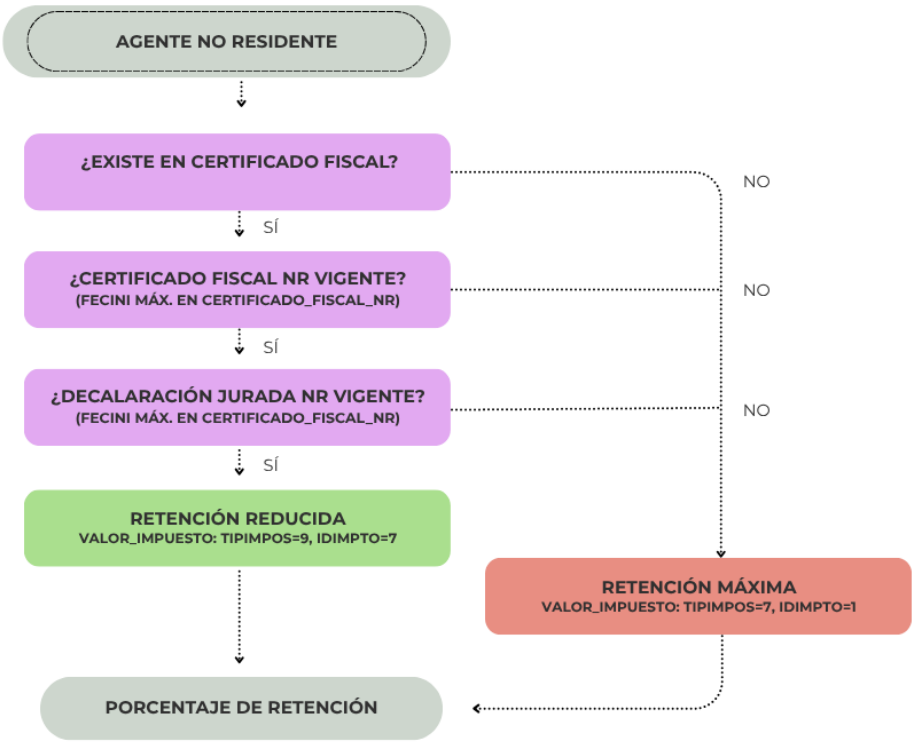


Figura 5. Diagrama de flujo del cálculo de retención para no residentes. Elaboración propia.

3.1.3.3 Cálculo de impuesto

El cálculo del impuesto se realiza después de consolidar los resultados correspondientes a agentes residentes y no residentes.

Primero el sistema consulta la información fiscal del contrato para determinar si este se encuentra sujeto al impuesto. Cuando el contrato no está sujeto, el porcentaje se establece en cero y no es necesario continuar con las consultas territoriales.

Cuando el impuesto resulta aplicable, el sistema utiliza la oficina gestora asociada al contrato para determinar su ámbito territorial. Para ello, relaciona la información de *CENTRO_OFICINA* con la tabla *PROVINCIA* y obtiene la categoría impositiva correspondiente al territorio identificado.

A partir de dicha categoría, se consulta la tabla *VALOR_IMPUESTO* para recuperar el porcentaje vigente en la fecha de liquidación. La selección se realiza mediante las fechas de inicio y fin de vigencia, evitando utilizar registros que no sean temporalmente válidos para el periodo procesado.

Al igual que en los cálculos de retención, los porcentajes concretos no se encuentran codificados directamente en la lógica de transformación, sino que se obtienen de las fuentes paramétricas corporativas. Esta separación facilita el mantenimiento del proceso y reduce el impacto técnico de posibles actualizaciones en los valores fiscales.

3.1.4 REQUISITOS FUNCIONALES Y NO FUNCIONALES

El desarrollo del *framework* de automatización de liquidaciones económicas viene determinado por un conjunto de requisitos que definen tanto el comportamiento esperado del sistema como las propiedades de calidad que debe satisfacer.

3.1.4.1 Requisitos funcionales

Los requisitos funcionales describen las capacidades que el sistema debe proporcionar para dar respuesta a las necesidades del proceso de liquidación.

El sistema debe ser capaz de leer e integrar la información procedente de las nueve tablas que intervienen en el proceso, realizando los cruces necesarios entre ellas para construir el conjunto de datos sobre el que se aplicará la lógica de negocio. Esta integración debe contemplar el filtrado por vigencia temporal en aquellas tablas que lo requieran, como es el caso de la tabla *VALOR_IMPUESTO*.

El sistema debe implementar la lógica de cálculo definida para el proceso, incluyendo el tratamiento diferenciado para agentes residentes y no residentes, la validación de la información fiscal y documental necesaria y la determinación del impuesto aplicable a partir de la información contractual y territorial. Los códigos concretos de parametrización y las condiciones internas utilizadas para seleccionar cada tratamiento fiscal no se incluyen por motivos de confidencialidad.

El *framework* debe construir las tablas finales de liquidación y persistirlas inicialmente en el esquema de trabajo de Databricks en formato Delta Lake, permitiendo comprobar los resultados antes de su integración con el sistema de destino. Posteriormente, estas tablas deben ser cargadas en la base de datos Oracle corporativa mediante los mecanismos establecidos por los equipos responsables de la entidad.

El sistema debe permitir su ejecución en los diferentes entornos habilitados para el proceso, adaptando las configuraciones de acceso a las fuentes, las rutas de almacenamiento y los parámetros de ejecución al entorno activo.

El sistema debe admitir la ejecución en modo de prueba sobre datos sintéticos, posibilitando la validación del *framework* sin necesidad de acceder a datos reales de producción.

3.1.4.2 Requisitos no funcionales

Los requisitos no funcionales definen las propiedades de calidad que el sistema debe satisfacer con independencia de la funcionalidad específica que implementa.

La escalabilidad es uno de los requisitos centrales dado el volumen de contratos que caracteriza al proceso en producción. El sistema debe aprovechar las capacidades de procesamiento distribuido de Apache Spark para ejecutar los cruces y transformaciones de forma eficiente y permitir su adaptación ante posibles incrementos en el volumen de información, sin que sea necesario rediseñar por completo la solución.

Estrechamente ligada a la escalabilidad, la mantenibilidad exige que el sistema esté estructurado de forma modular, separando claramente la capa de infraestructura de la lógica de negocio. Esta separación permite modificar o extender las reglas de cálculo sin alterar los componentes de orquestación y ejecución, garantizando la evolución sostenible del sistema.

La trazabilidad adquiere especial relevancia en un entorno bancario regulado. Cada ejecución debe registrar información suficiente para identificar la fecha de proceso, el entorno utilizado, las fases ejecutadas y las posibles incidencias detectadas. Además, la

persistencia intermedia de los resultados permite realizar comprobaciones antes de la carga en el sistema de destino y facilita la revisión posterior del proceso.

En cuanto a la fiabilidad, el sistema debe generar resultados consistentes y reproducibles ante las mismas entradas, eliminando la variabilidad que introducía la intervención manual en el proceso anterior.

Finalmente, el cumplimiento de los mecanismos corporativos de seguridad requiere que el acceso a los datos y la ejecución del procesamiento se realicen dentro de los entornos autorizados por la entidad. Asimismo, la validación académica debe realizarse mediante datos sintéticos, evitando la extracción o exposición de información real de carácter sensible.

3.2 DISEÑO DEL SISTEMA

3.2.1 ARQUITECTURA GENERAL DE LA SOLUCIÓN

La solución desarrollada se integra sobre la infraestructura tecnológica existente en la entidad, adoptando un enfoque basado en el paradigma ELT descrito en el capítulo 1. La arquitectura general de la solución se articula en tres capas diferenciadas: la capa de fuentes de datos, la capa de procesamiento y la capa de persistencia.

La Figura 6 muestra la arquitectura general de la solución desarrollada y resume el flujo completo del proceso, desde la lectura de las fuentes de datos hasta la generación y persistencia de las tablas finales. En ella se distinguen las tres capas principales de la solución: fuentes, procesamiento y persistencia; así como los componentes transversales relacionados con la gestión de entornos, la trazabilidad y la orquestación del pipeline.

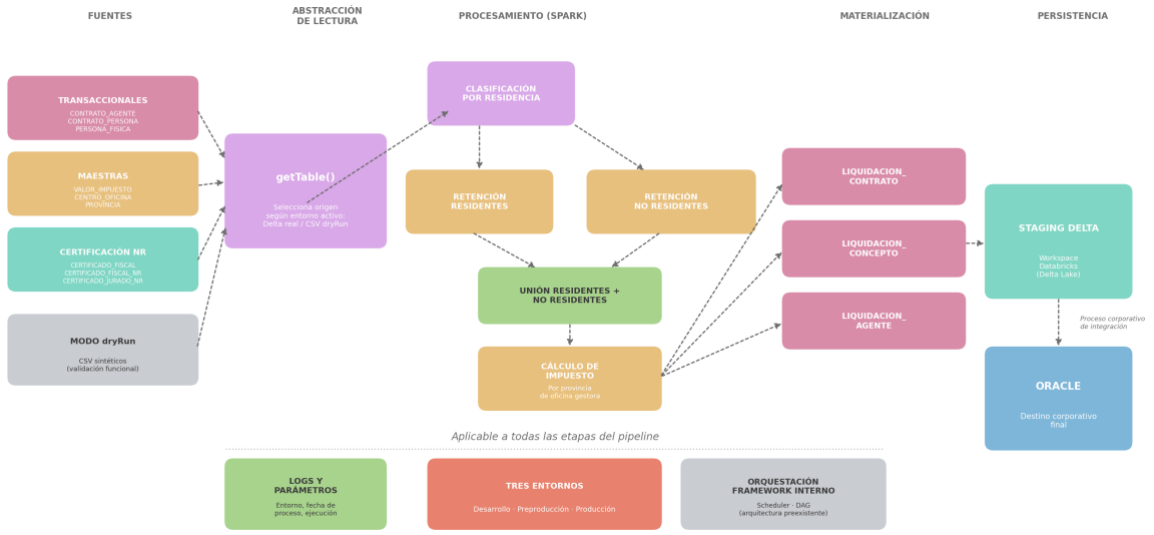


Figura 6. Arquitectura general del framework de automatización de liquidaciones económicas. Elaboración propia.

La capa de fuentes de datos comprende las tablas almacenadas en Databricks, descritas en detalle en el apartado 3.1.2, que contienen la información contractual, personal, territorial y fiscal necesaria para el cálculo de las liquidaciones. El acceso a estas fuentes se realiza mediante la abstracción proporcionada por el propio *framework*, que gestiona la lectura de las tablas en función del entorno de ejecución activo.

La capa de procesamiento está implementada sobre Apache Spark, ejecutándose en la plataforma Databricks. En esta capa se aplica la totalidad de la lógica de negocio del proceso de liquidación, incluyendo la integración de las distintas fuentes de datos, el cálculo de retenciones e impuestos y la construcción del resultado final. El procesamiento distribuido de Spark permite ejecutar los cruces y transformaciones sobre el volumen de información asociado al proceso, aprovechando los recursos del clúster disponible.

La capa de persistencia se estructura en dos niveles. En primer lugar, las tablas finales se materializan en el esquema de trabajo de Databricks en formato Delta Lake. Esta capa intermedia permite conservar los resultados del procesamiento y realizar comprobaciones antes de su envío al sistema de destino.

Posteriormente, las tablas son cargadas en la base de datos Oracle corporativa, donde quedan disponibles para su explotación por parte de los sistemas consumidores. Esta carga se realiza mediante los mecanismos de integración establecidos por la entidad y su implementación técnica fue gestionada por los equipos responsables, por lo que no forma parte de la aportación directa desarrollada en el presente trabajo.

La solución contempla tres entornos de ejecución diferenciados: desarrollo, preproducción y producción. Cada uno dispone de su propia configuración de acceso a las fuentes de datos, rutas, parámetros de ejecución y credenciales, gestionada mediante ficheros de configuración específicos para cada entorno. Esta separación contribuye a aislar las actividades de desarrollo y validación respecto a los procesos y datos del entorno productivo.

3.2.2 ESTRUCTURA DEL *FRAMEWORK*

El pipeline de liquidación se desarrolla sobre un *framework* interno de la entidad diseñado para la construcción y orquestación de procesos Spark. Este establece un modelo de ejecución basado en etapas independientes entre las que se detectan automáticamente las dependencias, construyendo un grafo acíclico dirigido (DAG) que permite ejecutar en paralelo aquellos procesos que no tienen dependencias entre sí.[15]

Este enfoque facilita la organización del pipeline y el aprovechamiento de los recursos computacionales disponibles, evitando que el desarrollador tenga que gestionar manualmente toda la secuencia de ejecución.

La figura 7 ilustra de forma conceptual este mecanismo de ejecución, mostrando cómo el sistema organiza los procesos en niveles en función de sus dependencias.

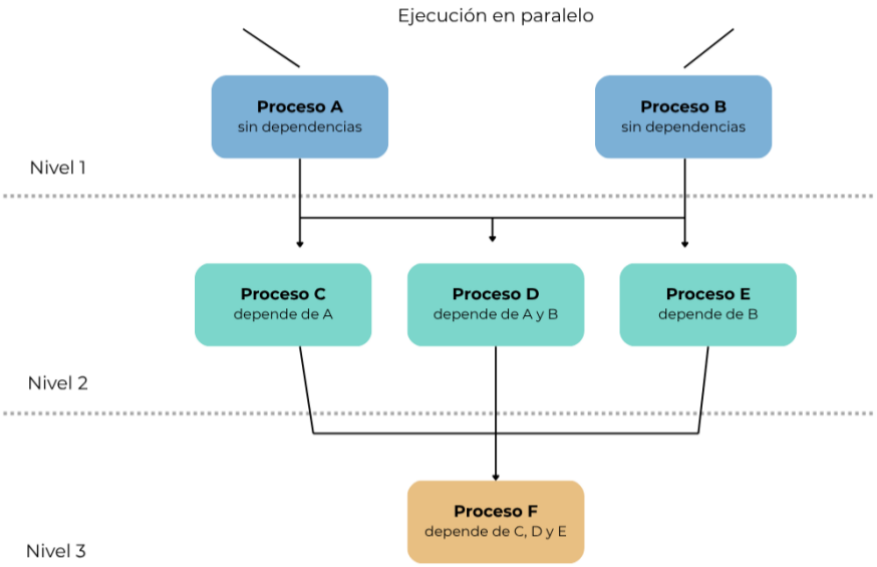


Figura 7. Ejemplo conceptual de un DAG. Elaboración propia.

Desde el punto de vista del desarrollo, el *framework* proporciona dos plantillas sobre las que implementar los procesos: una plantilla ligera para procesos simples y una plantilla estructurada, denominada *FlujoProceso*, orientada a *pipelines* Spark completos. Esta última divide el proceso en dos fases diferenciadas: una fase de preparación en la que se aplican todas las transformaciones y se construye el *DataFrame* resultado, y una fase de materialización en la que dicho resultado se persiste en el sistema de destino.

El proceso de liquidación económica se implementa sobre la plantilla *FlujoProceso*, aprovechando esta separación para mantener la lógica de negocio desacoplada de la escritura de resultados. La fase de preparación concentra la integración de las fuentes y la aplicación de las reglas de cálculo, mientras que la fase de materialización construye y persiste inicialmente las tablas finales en el esquema de Databricks en formato Delta Lake. Posteriormente, estas tablas son cargadas en la base de datos Oracle como hemos indicado previamente.

3.2.3 DISEÑO DEL PIPELINE DE LIQUIDACIÓN

La fase de preparación comienza con la lectura de las nueve tablas de entrada descritas en el apartado 3.1.2, cargándolas como *DataFrames* de Spark. La tabla *VALOR_IMPUESTO*

recibe un tratamiento especial en este momento, aplicando un filtro de vigencia temporal que selecciona únicamente los registros activos en la fecha de liquidación.

Una vez disponibles los datos, el proceso se bifurca según la condición de residencia de cada agente. Para los residentes, se cruzan las tablas *CONTRATO_AGENTE*, *CONTRATO_PERSONA* y *PERSONA_FISICA* filtrando por país de residencia igual a España, aplicando a continuación la lógica de retención descrita en el apartado 3.1.3. Para los no residentes, el cálculo es más complejo: se ejecuta el flujo de verificación secuencial de certificados fiscales, haciendo uso de funciones de ventana para recuperar los registros más recientes de cada persona en las tablas *CERTIFICADO_FISCAL*, *CERTIFICADO_FISCAL_NR* y *CERTIFICADO_JURADO_NR*.

Los resultados de ambas ramas se unen en un único *DataFram*e que consolida todos los contratos con su porcentaje de retención calculado. Sobre este *DataFrame* unificado se calcula el impuesto, cruzándolo con las tablas *CENTRO_OFICINA*, *PROVINCIA* y *VALOR_IMPUESTO* para determinar el tipo impositivo territorial y recuperar el porcentaje vigente en la fecha de liquidación.

La fase de materialización recibe el *DataFrame* resultado y lo transforma en las estructuras requeridas por las distintas tablas de salida, manteniendo desacoplada la lógica de negocio del mecanismo de persistencia.

3.3 IMPLEMENTACIÓN

3.3.1 IMPLEMENTACIÓN DEL PIPELINE DE LIQUIDACIÓN

El pipeline de liquidación se implementa en Spark Scala sobre la plantilla *FlujoProceso* descrita en el apartado 3.2.2. A diferencia del apartado anterior, que describía el diseño lógico del proceso, este apartado se centra en las decisiones técnicas de implementación adoptadas para dar respuesta a los requisitos funcionales y no funcionales definidos en el apartado 3.1.4.

El acceso a las tablas de entrada se gestiona mediante una función auxiliar de abstracción que adapta la lectura al entorno activo. Cuando el proceso se ejecuta en modo *dryRun*, esta función redirige la lectura hacia los ficheros CSV sintéticos almacenados localmente, configurando Spark en modo local con particionado mínimo para reducir el consumo de recursos. En los entornos de integración y producción, accede directamente a las tablas *Delta* del catálogo de Databricks. Este mecanismo permite validar el comportamiento del pipeline sin acceder a datos reales, facilitando el desarrollo iterativo y las pruebas unitarias.

La lógica de cálculo se organiza en tres funciones privadas. La primera calcula la retención para agentes residentes mediante un *join* con la tabla *VALOR_IMPUESTO*, aplicando una expresión condicional *when/otherwise* que asigna un porcentaje de retención explícito a cada contrato: cero si el campo de retención está vacío o es igual a cero, y el porcentaje recuperado de la tabla paramétrica en caso contrario. Esta expresión garantiza que todos los contratos reciben un valor definido independientemente de su casuística, evitando nulos en el resultado.

La segunda función gestiona el cálculo para agentes no residentes y es la operación más costosa, como reflejan los tiempos observados en la Spark UI durante las pruebas en el entorno de integración. La complejidad computacional responde al uso de *joins* de tipo *left* encadenados con las tres tablas de certificación y al empleo de funciones de ventana para recuperar el registro más reciente de cada persona en cada tabla. Como se ha explicado en capítulos anteriores, la decisión sobre el porcentaje aplicable se resuelve a partir del resultado conjunto de las verificaciones documentales. Una vez determinada la situación del agente, se asigna el tratamiento fiscal correspondiente utilizando los valores recuperados previamente de la tabla paramétrica, evitando realizar *joins* adicionales en esta fase.

La tercera función calcula el impuesto sobre el *DataFrame* unificado resultante de la unión de ambas ramas. Encadena tres *joins* sucesivos con *CENTRO_OFICINA, PROVINCIA* y *VALOR_IMPUESTO*, utilizando el campo de sujeción al impuesto como condición previa.

3.3.2 IMPLEMENTACIÓN DE LA CARGA DE TABLAS DE RESULTADOS

La fase de materialización recibe el *DataFrame* resultado de la fase de preparación, que en este punto contiene para cada contrato el porcentaje de retención y el porcentaje de impuesto calculados, y procede a construir y persistir las tres tablas de resultados del proceso.

Las tres tablas se crean sobre el esquema del *workspace* de Databricks, que actúa como repositorio intermedio de *staging*, en formato Delta Lake, aprovechando las capacidades transaccionales y de auditoría que esta tecnología proporciona, descritas en el apartado 1.2.2. Todas ellas están particionadas por fecha de proceso, lo que permite sobreescrituras parciales eficientes y facilita la consulta de versiones anteriores dentro del periodo de conservación disponible mediante la funcionalidad de *time travel* de Delta Lake.

La primera tabla presenta los resultados con el mayor nivel de detalle del proceso. Su construcción agrupa la información según la clave contractual definida y acumula los importes económicos correspondientes, incorporando también campos técnicos y de auditoría.

La segunda tabla organiza los resultados según los conceptos de liquidación. Esta estructura proporciona un nivel de agregación diferente y añade la información necesaria para relacionar cada liquidación con el agente y la entidad responsable.

La tercera tabla recoge el resultado económico consolidado por agente. Para construirla, se realiza una agregación intermedia sobre la que se aplican los porcentajes fiscales obtenidos durante la fase de preparación. A partir de estas operaciones se calculan los importes asociados al impuesto, a la retención y al resultado líquido final, de acuerdo con las reglas internas del proceso.

Con el fin de preservar la confidencialidad de la lógica de negocio, no se detallan las fórmulas exactas utilizadas para combinar estos importes. Desde el punto de vista técnico, los cálculos se implementan mediante expresiones sobre columnas de Spark y utilizan la

función *coalesce* para tratar como cero los posibles porcentajes nulos, evitando que estos se propaguen al resultado final.

Esta tabla incorpora también las fechas que delimitan el periodo de liquidación. La fecha de finalización se obtiene mediante funciones temporales de Spark a partir de la fecha inicial del ciclo, permitiendo representar de forma consistente el periodo procesado.

El mecanismo de materialización varía según el modo de ejecución. Durante las pruebas, los *DataFrames* se registran como vistas temporales globales, lo que permite inspeccionar los resultados y ejecutar comprobaciones sin persistirlos en el catálogo. En el modo ordinario, las tablas se escriben en formato Delta mediante sobreescritura selectiva de la partición correspondiente a la fecha de proceso, preservando el resto del histórico.

3.4 GENERACIÓN DE DATOS SINTÉTICOS

3.4.1 MOTIVACIÓN Y ENFOQUE

El desarrollo y validación del *framework* en un entorno bancario regulado plantea una restricción fundamental: los datos reales sobre los que opera el proceso contienen información sensible de agentes y contratos que no puede utilizarse fuera del entorno productivo de la entidad. Esta limitación, enmarcada en las restricciones del GDPR descritas en el apartado 1.2.6, impide emplear datos reales tanto en las fases de prueba durante el desarrollo como en la presentación y validación académica del trabajo.

Para abordar esta restricción se optó por la generación de datos sintéticos que reprodujeran la estructura y las propiedades estadísticas de los datos reales sin contener información sensible o identificable. En una primera instancia se intentó utilizar una API de generación de datos sintéticos disponible internamente en el banco, pero ante su falta de operatividad se desarrolló un generador propio en Python. Este generador cubre las nueve tablas que intervienen en el proceso de liquidación y constituye una pieza fundamental del trabajo, ya

que permite ejecutar y validar en condiciones representativas reduciendo el riesgo de comprometer la confidencialidad de la información.

3.4.2 ANÁLISIS EXPLORATORIO DE LOS DATOS REALES

El punto de partida para el diseño del generador fue el análisis de las distribuciones de los datos reales mediante consultas ejecutadas directamente en Databricks, filtrando por *data_date_part* = '2026-04-30' para trabajar con la partición más reciente disponible y evitar los problemas de rendimiento que surgían al consultar el volumen completo de algunas tablas.

El análisis se centró en los campos que determinan qué rama del cálculo se ejecuta para cada contrato. En la tabla *CONTRATO_AGENTE*, que cuenta con 103.936 registros en la partición analizada, se estudiaron los campos *fisij* y *retenalf*, observándose en ambos casos una clara concentración en los valores más frecuentes con una presencia residual de los casos menos habituales. En *CONTRATO_PERSONA*, que con más de 673 millones de registros es la tabla de mayor volumen del proceso, se analizó la proporción entre personas físicas y jurídicas, relevante porque solo las primeras atraviesan la lógica de cálculo de retención.

En cuanto a la dimensión geográfica, la tabla *CENTRO_OFICINA* muestra una fuerte concentración de los 16.732 centros en Madrid y Barcelona, con una presencia muy reducida de centros en el extranjero, concretamente en Reino Unido y Alemania, que se incorporaron al generador para cubrir casos límite. La tabla *PROVINCIA* contiene exactamente 52 registros, con predominio del tipo impositivo estándar y valores especiales únicamente en los territorios con régimen fiscal diferenciado.

En *PERSONA_FISICA* se analizó la distribución del país de residencia sobre más de 20 millones de registros, identificando que la gran mayoría corresponde a residentes en España y que el colectivo de no residentes, con Reino Unido y Francia como países más frecuentes, representa una minoría del total.

Si bien el análisis de distribuciones sirvió como referencia para dotar a los datos sintéticos de un grado razonable de realismo, el criterio prioritario en el diseño del generador no fue la fidelidad estadística sino la cobertura funcional: garantizar que el *dataset* sintético contiene suficientes casos de cada escenario relevante para la lógica de negocio (residentes con y sin retención, no residentes con documentación completa, parcial y ausente, contratos sujetos y no sujetos a impuesto) de modo que el *framework* pueda ser validado de forma exhaustiva sobre todos los escenarios funcionales relevantes definidos dentro del alcance del proyecto.

3.4.3 DISEÑO E IMPLEMENTACIÓN DEL GENERADOR

El generador se implementa en Python como un *script* modular, con una función dedicada a la generación de cada tabla. Para garantizar la reproducibilidad de los datos generados, la semilla aleatoria se fija al valor 42 al inicio de la ejecución, de modo que sucesivas ejecuciones del generador producen siempre el mismo conjunto de datos. El *dataset* resultante comprende 300 centros y 1.200 contratos, con una fecha de partición fijada en 2026-04-01 y una fecha de liquidación de 2026-02-03, alineada con la configurada en el pipeline.

La generación sigue un orden que respeta las dependencias entre tablas, comenzando por las maestras y paramétricas y avanzando progresivamente hacia las transaccionales: *PROVINCIA, CENTRO_OFICINA, VALOR_IMPUESTO, CONTRATO_AGENTE, CONTRATO_PERSONA, PERSONA_FISICA* y finalmente las tres tablas de certificación de no residentes. Este orden garantiza que todas las claves foráneas puedan resolverse correctamente en el momento de la generación.

La tabla *PROVINCIA* incluye las unidades territoriales necesarias para representar el ámbito nacional, junto con algunos casos adicionales destinados a cubrir situaciones menos frecuentes. A partir de ella, la tabla *CENTRO_OFICINA* se construye mediante una distribución geográfica ponderada, incrementando la presencia de las zonas con mayor actividad e incorporando también centros asociados a territorios minoritarios o extranjeros para ampliar la cobertura de las pruebas.

La tabla *VALOR_IMPUESTO* se genera incluyendo las distintas categorías fiscales requeridas por el proceso y registros correspondientes a diferentes periodos de vigencia. Para cada categoría se crean tanto valores activos como caducados, de modo que el pipeline deba seleccionar correctamente aquellos que resultan válidos en la fecha de liquidación. Los códigos de parametrización, porcentajes y correspondencias concretas utilizados en el entorno real no se reproducen en la memoria por motivos de confidencialidad.

Los campos más relevantes para la lógica de negocio se generan con probabilidades que combinan las distribuciones observadas en los datos reales con el criterio de cobertura funcional descrito anteriormente. Las tablas siguientes recogen los valores aplicados:

VALOR	*PROBABILIDAD*
Sujeto a impuesto	94%
No sujeto a impuesto	5.5%
Valor no informado	0.5%

Tabla 3. Campo fisij (sujeción al impuesto) - Tabla CONTRATO_AGENTE

VALOR	*PROBABILIDAD*
Sin retención aplicable	65%
Tratamiento ordinario	34%
Tratamiento especial	0.94%
Valor no informado	0.06%

Tabla 4. Campo retenalf (código de retención) - Tabla CONTRATO_AGENTE

VALOR	*PROBABILIDAD*
Agente residente	90%
Agente no residente	10%

Tabla 5. Campo paisresi (país de residencia) - Tabla PERSONA_FISICA

TABLA	*PROB. DE APARICIÓN*
Información fiscal general	80%
Certificación fiscal de no residente	65%
Declaración complementaria de no residente	65%

Tabla 6. Probabilidad de aparición en tablas de certificación

La tabla *CONTRATO_PERSONA* mantiene la relación entre los contratos y sus intervinientes. El generador permite que determinados contratos estén asociados a más de una persona, reproduciendo de este modo relaciones de cardinalidad más complejas. A partir de estas relaciones se construye *PERSONA_FISICA*, donde se asigna la información de residencia necesaria para diferenciar entre agentes residentes y no residentes.

Para el subconjunto de agentes no residentes se generan registros en *CERTIFICADO_FISCAL*, *CERTIFICADO_FISCAL_NR* y *CERTIFICADO_JURADO_NR*. La presencia de cada persona en estas tablas se determina de forma probabilística y se combinan registros vigentes, caducados y ausentes. Esta variación permite comprobar el

comportamiento del pipeline ante distintos grados de disponibilidad y validez de la información documental.

Las probabilidades utilizadas en el generador tienen una finalidad exclusivamente experimental. Aunque se apoyan en las características generales identificadas durante el análisis, no reproducen de forma literal las distribuciones existentes en los sistemas reales. Algunos casos minoritarios se encuentran sobrerrepresentados para asegurar que las distintas ramas funcionales del proceso puedan validarse con un número suficiente de observaciones.

En conjunto, el *dataset* sintético reproduce la estructura de las nueve fuentes, sus relaciones y las principales casuísticas necesarias para ejecutar el pipeline. De este modo, es posible comprobar su comportamiento ante información completa, incompleta, ausente o fuera de vigencia, sin exponer los parámetros concretos ni las condiciones exactas que determinan el tratamiento fiscal aplicado por la entidad.

3.4.4 ENFOQUE DE VALIDACIÓN DEL FRAMEWORK

La validación del *framework* se articula en dos niveles complementarios que se desarrollan de forma secuencial. En primer lugar, se comprueba la coherencia del conjunto de datos sintéticos generado y, posteriormente, se valida el comportamiento funcional del pipeline al ejecutarse sobre dichos datos. Estas comprobaciones se realizan tanto en modo local como en el entorno de integración o preproducción de la entidad. Los resultados concretos de las distintas validaciones se presentan en el Capítulo 4.

El primer nivel tiene como objetivo verificar que el conjunto de datos sintéticos es estructural y funcionalmente coherente antes de utilizarlo como entrada del pipeline. Para ello, se comprueba que las distribuciones de los campos relevantes se aproximan a las proporciones definidas en el generador, que no existen registros huérfanos entre las tablas relacionadas y que se mantiene la integridad referencial en toda la cadena de dependencias.

Entre las comprobaciones realizadas se incluye la correspondencia entre los códigos territoriales asignados a los centros y los registros existentes en la tabla *PROVINCIA,* la

presencia previa en *PERSONA_FISICA* de las personas incluidas en las tablas de certificación y la adecuación de la distribución de residentes y no residentes a la configuración establecida para el conjunto sintético. Estas verificaciones permiten confirmar que los datos cumplen las condiciones necesarias para ejecutar el pipeline de forma consistente.

El segundo nivel se centra en comprobar que el pipeline aplica correctamente las reglas funcionales definidas. Para ello, el *framework* se ejecuta sobre los datos sintéticos y los resultados obtenidos se analizan mediante consultas de validación.

Los escenarios de prueba cubren los principales bloques funcionales descritos en el apartado 3.1.3. En el cálculo de retenciones se comprueba que el sistema distingue correctamente los casos sujetos y no sujetos, selecciona los parámetros vigentes y aplica el tratamiento correspondiente en función de la información fiscal y documental disponible. Para los agentes no residentes se incluyen situaciones con documentación completa, incompleta, ausente y caducada, permitiendo verificar el comportamiento de las diferentes ramas del proceso sin reproducir las condiciones fiscales concretas asociadas a cada una.

En el cálculo del impuesto se comprueba que el sistema diferencia los contratos sujetos y no sujetos y que, cuando resulta aplicable, obtiene el valor correspondiente a partir de la información territorial y de los parámetros vigentes en la fecha de liquidación. También se verifica la coherencia aritmética de los importes generados en la tabla *LIQUIDACION_AGENTE*. Para ello, se recalculan de forma independiente los componentes económicos a partir de los datos de entrada y se comparan con los valores producidos por el framework, sin reproducir en la memoria las fórmulas internas exactas del proceso.

La validación local se realiza con la configuración dryRun activada y utilizando los ficheros CSV sintéticos como fuentes de entrada. En este modo, las tablas se registran como vistas temporales en memoria y se evita la escritura en sistemas externos. Esta modalidad permite comprobar de forma aislada las transformaciones y ejecutar pruebas controladas con un consumo limitado de recursos.

Posteriormente, el *framework* se ejecuta en el entorno de integración de la entidad, utilizando la configuración y los recursos de una infraestructura próxima al entorno productivo. Esta validación permite comprobar el comportamiento del pipeline fuera del entorno local, verificar la lectura de las fuentes, la ejecución distribuida de las transformaciones y la materialización de los resultados en Databricks. Además, permite analizar las distintas etapas de ejecución mediante las herramientas de monitorización disponibles en la plataforma.

Como comprobación adicional, se desarrolló una prueba de reproducibilidad de la fase de preparación del *pipeline*. Su objetivo es verificar que la lógica de cálculo produce el mismo resultado cuando se ejecuta repetidamente sobre las mismas fuentes y con idénticos parámetros.

Capítulo 4. ANÁLISIS DE RESULTADOS

En este capítulo se recogen los resultados obtenidos tras la implementación y ejecución del *framework* de automatización de liquidaciones económicas descrito en el capítulo anterior. Dado que el valor fundamental del proyecto reside en la automatización del proceso en sí, el análisis de resultados se estructura en torno a tres ejes complementarios que permiten evaluar el sistema desde distintas perspectivas.

Se presenta la validación funcional de la solución desarrollada mediante su ejecución sobre el conjunto de datos sintéticos generado específicamente para este propósito, comprobando que la lógica de negocio produce resultados correctos en todos los escenarios relevantes definidos en el apartado 3.1.3. Además, se analizan las métricas de rendimiento obtenidas a partir de la ejecución del *pipeline* sobre datos reales en el entorno de integración, con el objetivo de caracterizar el comportamiento del sistema a escala productiva. Finalmente, se presenta una comparativa estructurada entre el proceso manual que existía anteriormente y el *framework* desarrollado, poniendo de manifiesto las mejoras introducidas en términos de trazabilidad, mantenibilidad y reducción del riesgo operacional.

Con carácter introductorio, la Figura 8 recoge de forma sintética los principales indicadores cuantitativos obtenidos durante la validación y ejecución, que se desarrollan en detalle en los apartados siguientes.

Figura 8. Panel de indicadores clave de los resultados obtenidos. Elaboración propia

4.1 VALIDACIÓN FUNCIONAL DEL FRAMEWORK

La validación funcional tiene como objetivo verificar que el *framework* produce resultados correctos para todos los escenarios relevantes de la lógica de negocio, tal y como fueron definidos en el apartado 3.1.3. Para ello se ejecutó el pipeline sobre el conjunto de datos sintéticos descrito en el apartado 3.4, compuesto por 1.200 contratos únicos, que generan 1.222 relaciones contrato-agente al permitir el generador más de un interviniente por contrato. Estas 1.222 relaciones constituyen la unidad de cálculo utilizada en la validación, de las cuales 1.070 corresponden a agentes residentes en España y 152 a agentes no residentes.

La validación se estructura en tres bloques que reproducen los tres módulos de cálculo del pipeline: retención de residentes, retención de no residentes e impuesto territorial. Para cada escenario se verifica que el resultado obtenido coincide con el esperado según las reglas de negocio, comprobando que el número de registros conformes es igual al total de registros del escenario en todos los casos.

4.1.1 BLOQUE 1: RETENCIÓN DE AGENTES RESIDENTES

El primer bloque comprende los escenarios asociados al cálculo de la retención de agentes residentes. Las pruebas contemplan contratos sin retención aplicable, registros en los que la información fiscal no se encuentra informada y contratos asociados a diferentes categorías de retención.

Los casos C1a y C1b verifican que el *framework* identifica correctamente las situaciones en las que no procede aplicar retención, independientemente de la forma en que dicha situación se represente en los datos de entrada. Los casos C2 y C3 comprueban que, cuando existe una categoría de retención informada, el pipeline recupera de la tabla VALOR_IMPUESTO el parámetro correspondiente y vigente en la fecha de liquidación.

Los resultados obtenidos confirman que todos los registros analizados fueron clasificados correctamente y recibieron el tratamiento esperado según la configuración sintética utilizada.

4.1.2 BLOQUE 2: RETENCIÓN DE AGENTES NO RESIDENTES

El segundo bloque valida el tratamiento de los agentes no residentes, cuya ejecución incorpora comprobaciones adicionales sobre la disponibilidad y vigencia de la documentación fiscal.

El caso C4 representa agentes cuya información documental cumple las condiciones previstas para uno de los tratamientos fiscales configurados. Por su parte, el caso C5 agrupa situaciones en las que la documentación se encuentra incompleta, ausente o fuera de vigencia y, por tanto, debe aplicarse el tratamiento alternativo definido para estos escenarios.

En ambos casos se comprobó que el pipeline seleccionó correctamente la categoría fiscal correspondiente y recuperó de la tabla paramétrica el valor vigente. La totalidad de los registros incluidos en cada escenario presentó el resultado esperado.

4.1.3 BLOQUE 3: IMPUESTO TERRITORIAL

El tercer bloque valida el cálculo del impuesto sobre el conjunto de contratos procesados, con independencia de la condición de residencia del agente.

El caso C6 comprueba que el *framework* identifica correctamente los contratos no sujetos al impuesto. Los casos C7 y C8 verifican que, cuando el impuesto resulta aplicable, el pipeline obtiene la categoría territorial correspondiente a partir de la información de la oficina gestora y recupera el parámetro fiscal vigente asociado a dicha categoría.

El conjunto sintético incluye tanto situaciones correspondientes al régimen territorial general como casos vinculados a territorios con un tratamiento diferenciado, lo que permite validar las distintas ramas del cálculo. En todos los escenarios, el resultado generado por el sistema coincidió con el valor esperado definido para los datos de prueba.

4.1.4 TABLA RESUMEN DE CASOS DE VALIDACIÓN

La tabla siguiente recoge de forma consolidada los ocho escenarios validados, el número de registros involucrados en cada caso y el resultado de la verificación:

Caso	***Descripción***	***Resultado esperado***	***Número de registros***	***Conformidad***
C1a	Contratos sin retención aplicable	*Asignación de retención nula*	643	100%
C1b	Información de retención no informada	*Asignación de retención nula*	37	100%
C2	Categoría ordinaria de residente	*Recuperación del porcentaje vigente asociado a la categoría ordinaria*	381	100%

C3	Categoría especial de residente	*Recuperación del porcentaje vigente asociado a la categoría especial*	9	100%
C4	No residente CON certificados válidos	*Aplicación del tratamiento fiscal previsto para documentación válida*	102	100%
C5	No residente SIN certificados completos	*Aplicación del tratamiento fiscal alternativo*	50	100%
C6	No sujeto a impuesto	*Asignación de impuesto nulo*	70	100%
C7	Sujeto a impuesto (régimen general)	*Recuperación del porcentaje vigente del régimen general*	1092	100%
C8	Sujeto a impuesto (régimen especial)	*Recuperación del porcentaje vigente del régimen territorial correspondiente*	40	100%

Tabla 7. Resumen de casos de validación funcional sobre el dataset sintético

Nota: Por motivos de confidencialidad, los resultados esperados se expresan mediante categorías funcionales y no incluyen códigos internos, porcentajes concretos ni condiciones específicas de parametrización. No obstante, durante la validación se compararon los valores obtenidos con los valores exactos definidos en el conjunto de datos sintéticos.

Los 1.200 contratos procesados superan la validación en su totalidad, con un resultado conforme en los ocho escenarios definidos. Este resultado acredita que el *framework* implementa correctamente la lógica de negocio del proceso de liquidación económica en todos los casos funcionales definidos, incluyendo los casos límite asociados a campos vacíos, documentación fiscal caducada y regímenes fiscales diferenciados.

4.1.5 VALIDACIÓN DE LA REPRODUCIBILIDAD DE LA FASE DE PREPARACIÓN

Con el objetivo de comprobar que el proceso genera resultados reproducibles ante posibles reejecuciones, se ejecutó dos veces consecutivas la fase de preparación del pipeline utilizando las mismas tablas sintéticas de entrada y los mismos parámetros de proceso.

Antes de la comparación, los dos *DataFrames* resultantes se ordenaron por la totalidad de sus columnas para evitar diferencias asociadas al orden no determinista de los registros en Apache Spark. Posteriormente, se verificó la igualdad del esquema, la cardinalidad y el contenido completo de ambas salidas.

La prueba finalizó correctamente, sin detectarse diferencias entre las dos ejecuciones. Ambas generaron las mismas columnas, el mismo número de registros y los mismos valores fila a fila. Este resultado confirma que la lógica de cálculo implementada es determinista y reproducible ante entradas idénticas, por lo que una reejecución de la fase de preparación no altera el resultado obtenido.

La comprobación se realizó en modo local y con *dryRun* activado, por lo que su alcance se limita a la reproducibilidad de las transformaciones ejecutadas durante la fase de preparación y no incluye el comportamiento del mecanismo de persistencia sobre las tablas Delta u Oracle.

4.2 RENDIMIENTO EN EL ENTORNO DE INTEGRACIÓN/ PREPRODUCCIÓN

Con el objetivo de caracterizar el comportamiento del *framework* sobre un volumen de datos representativo del entorno productivo, se ejecutó el *pipeline* sobre datos reales en el entorno de integración de la entidad. La ejecución procesó los 103.936 contratos correspondientes a la partición de datos con *data_date_part* = 2026-04-30, completándose en un tiempo total de aproximadamente 32 minutos sobre un clúster Databricks destinado a la ejecución de procesos Scala, con paralelismo configurado a 50 particiones.

La Spark UI registró 487 *jobs* y 487 *stages* completados durante la ejecución, lo que permitió analizar el comportamiento de las distintas etapas del pipeline y localizar las operaciones con mayor coste computacional.

La etapa de mayor duración correspondió al procesamiento de los agentes no residentes, con un tiempo aproximado de dos minutos y 6.932 tareas encargadas de procesar 5,8 GB de datos de entrada. Este comportamiento es coherente con la mayor complejidad técnica de esta rama, que requiere varias operaciones de unión y el uso de funciones de ventana para seleccionar los registros temporalmente relevantes.

La segunda etapa más significativa tuvo una duración aproximada de 1,3 minutos y procesó 11,6 GB de información durante la materialización de los resultados en las tablas Delta utilizadas como capa intermedia.

El tiempo total de 32 minutos debe interpretarse dentro de las condiciones específicas del entorno de integración, cuyo dimensionamiento y carga de trabajo pueden diferir de los del entorno productivo. En consecuencia, este resultado permite caracterizar la ejecución observada, pero no extrapolar directamente el tiempo que se obtendría en producción, ya que este dependerá de factores como los recursos asignados al clúster, la concurrencia, el estado de la caché, la distribución de los datos y las optimizaciones habilitadas.

4.3 COMPARATIVA ENTRE EL PROCESO MANUAL Y EL FRAMEWORK AUTOMATIZADO

Antes de la implementación del *framework*, el proceso de liquidación económica se ejecutaba de forma manual mediante consultas directas sobre la base de datos Oracle de la entidad. Este enfoque, si bien funcional, presentaba limitaciones estructurales significativas en términos de trazabilidad, mantenibilidad y riesgo operacional. La siguiente tabla recoge una comparativa estructurada entre ambos enfoques en las dimensiones más relevantes:

Dimensión	*Proceso manual*	*Framework automatizado*
Modo de ejecución	Ejecución manual de consultas	Automático, *pipeline* Spark orquestado
Tiempo de cálculo	Variable y condicionado por la intervención manual	Aproximadamente 32 minutos (entorno integración)
Trazabilidad	Dependiente de los registros conservados durante cada ejecución	Registro de parámetros, fecha de proceso, entorno y etapas ejecutadas
Modificación reglas de negocio	Revisión y modificación manual de las consultas afectadas	Lógica centralizada y parámetros fiscales mantenidos en fuentes de referencia
Capacidad de procesamiento	Condicionada por la ejecución secuencial de consultas y la intervención del usuario	Procesamiento distribuido sobre Apache Spark
Separación de entornos	Sin separación formal	Configuraciones diferenciadas para desarrollo, integración/preproducción y producción
Validación de resultados	Manual, sin proceso sistemático	Automatizada mediante *tests* sobre los datos
Riesgo de error	Mayor dependencia de la intervención manual	Reducción de tareas manuales y comportamiento reproducible

Auditoría	Reconstrucción condicionada por las evidencias conservadas	Apoyada por logs, particionado temporal y versionado de las tablas Delta

Tabla 8. Comparativa entre el proceso manual previo y el framework automatizado

Una de las principales mejoras aportadas por el *framework* se encuentra en la mantenibilidad. En el procedimiento anterior, una modificación de las reglas fiscales podía exigir la identificación y actualización de distintas consultas, aumentando el riesgo de introducir inconsistencias. En la solución desarrollada, la lógica se encuentra centralizada dentro del pipeline y los valores fiscales variables se recuperan de las fuentes paramétricas correspondientes. De este modo, las actualizaciones de los porcentajes y de sus periodos de vigencia pueden realizarse en las tablas de referencia sin modificar las transformaciones principales, siempre que no cambie la estructura de la regla de negocio.

La trazabilidad constituye otra mejora relevante en un entorno bancario regulado. El *framework* registra información asociada a cada ejecución, como la fecha de proceso, el entorno activo, los parámetros utilizados y el estado de las distintas etapas. Además, la persistencia de los resultados en tablas particionadas por fecha y las capacidades de versionado de Delta Lake facilitan la revisión de versiones anteriores dentro del periodo de conservación disponible. Estos mecanismos mejoran la capacidad de reconstruir y analizar una ejecución respecto al procedimiento manual.

La separación de entornos también contribuye a reducir los riesgos operativos. Las configuraciones independientes de desarrollo, integración o preproducción y producción permiten adaptar las fuentes, rutas, parámetros y credenciales al contexto de ejecución. Esta organización reduce la posibilidad de que las pruebas afecten accidentalmente a datos o procesos productivos y facilita una evolución controlada de la solución.

Por último, la automatización permite ejecutar de manera reproducible una secuencia de transformaciones previamente definida. Aunque esto no elimina completamente la

posibilidad de errores, reduce la variabilidad asociada a la ejecución manual y facilita la incorporación de controles sistemáticos antes de que los resultados sean utilizados por los sistemas consumidores.

Capítulo 5. CONCLUSIONES Y TRABAJOS FUTUROS

5.1 CONCLUSIONES

El presente Trabajo de Fin de Máster ha tenido como objetivo diseñar, desarrollar y validar un *framework* para automatizar el proceso de liquidación económica de terceros en un entorno bancario regulado. La solución sustituye el procedimiento anterior, basado en la ejecución manual de consultas, por un pipeline estructurado sobre Apache Spark y Databricks. A partir de los resultados obtenidos, puede concluirse que los objetivos planteados al inicio del proyecto se han alcanzado satisfactoriamente.

En primer lugar, se ha implementado la lógica de cálculo del proceso mediante Apache Spark utilizando Scala, integrando las nueve fuentes de datos necesarias y estructurando el procesamiento en los principales bloques funcionales identificados: tratamiento de agentes residentes, tratamiento de agentes no residentes y cálculo del impuesto territorial. Por motivos de confidencialidad, la memoria presenta estas reglas con un nivel de abstracción que permite comprender el funcionamiento del sistema sin exponer los códigos, porcentajes ni condiciones internas de parametrización utilizados por la entidad.

La validación funcional confirmó el comportamiento esperado del sistema en los ocho escenarios definidos para las pruebas. Para ello, se utilizó un conjunto compuesto por 1.200 contratos sintéticos, que generaron 1.222 relaciones entre contratos y agentes. El *dataset* fue diseñado para representar tanto los casos habituales como determinadas situaciones minoritarias, incluyendo información ausente, documentación fuera de vigencia y diferentes configuraciones fiscales. En todos los escenarios analizados, los registros procesados obtuvieron el resultado esperado conforme a la configuración de prueba.

Se verificó también la reproducibilidad de la fase de preparación del pipeline. Dos ejecuciones realizadas sobre las mismas fuentes sintéticas y con idénticos parámetros generaron *DataFrames* equivalentes en esquema, cardinalidad y contenido. Esta

comprobación permite concluir que las transformaciones implementadas presentan un comportamiento determinista ante entradas equivalentes, aunque su alcance no incluye la idempotencia del mecanismo posterior de persistencia.

En segundo lugar, la ejecución del pipeline sobre datos reales en el entorno de integración o preproducción permitió evaluar su comportamiento sobre un volumen representativo del proceso. En esta prueba se procesaron 103.936 contratos en aproximadamente 32 minutos. El resultado demuestra que la solución es capaz de ejecutar las transformaciones definidas sobre el volumen analizado utilizando el procesamiento distribuido de Spark. No obstante, este tiempo corresponde a una configuración concreta del entorno de integración y no permite extrapolar directamente el rendimiento que se obtendría en producción ni afirmar una escalabilidad lineal ante volúmenes superiores.

También se completó el ciclo de generación y persistencia de las tres tablas finales de liquidación. Los resultados fueron materializados inicialmente en Databricks en formato Delta Lake, lo que permitió realizar comprobaciones intermedias antes de su incorporación a la base de datos Oracle corporativa. La implementación técnica de esta última integración fue realizada mediante los mecanismos establecidos por los equipos responsables de la entidad y no formó parte de la aportación directa desarrollada en este trabajo.

Además de automatizar el cálculo, la solución aporta una mayor estructuración del proceso. La separación entre lectura de fuentes, aplicación de transformaciones, construcción de resultados y materialización facilita el mantenimiento del código y reduce la dependencia de consultas ejecutadas manualmente. La configuración diferenciada por entornos, el registro de las ejecuciones y la utilización de datos sintéticos para las pruebas contribuyen igualmente a mejorar la trazabilidad, reproducibilidad y seguridad del desarrollo.

La aportación del proyecto debe situarse dentro de la evolución tecnológica prevista para los procesos de liquidación de la entidad. El *framework* desarrollado constituye una solución de carácter táctico, orientada a resolver una necesidad operativa concreta en un plazo acotado y aprovechando la arquitectura corporativa de orquestación de procesos Spark ya disponible.

Esta naturaleza táctica no supone una carencia del sistema, sino una decisión coherente con la necesidad de automatizar el procedimiento existente sin esperar al desarrollo de una plataforma estratégica de mayor alcance.

A este respecto, el proyecto proporciona una base técnica y metodológica que puede servir de referencia para una futura solución estratégica. Su estructura modular, la utilización de fuentes paramétricas para los valores fiscales y la separación entre la lógica de transformación y los componentes de infraestructura facilitan su evolución e integración en una plataforma corporativa más amplia. No obstante, dicha evolución requeriría revisar la arquitectura, ampliar las pruebas y adaptar las reglas y componentes a los requisitos específicos de los nuevos procesos incorporados.

5.2 TRABAJOS FUTUROS

Como continuación natural de este TFM se proponen las siguientes líneas de trabajo futuro:

- Validación continuada en producción. Una vez completado el primer ciclo de carga en Oracle, resultaría conveniente realizar un seguimiento de varias liquidaciones mensuales consecutivas en el entorno productivo. Los resultados obtenidos podrían contrastarse con las comprobaciones realizadas por el equipo de negocio, con el fin de evaluar la estabilidad del *framework* y detectar posibles casuísticas no contempladas durante la fase inicial de validación.
- Extensión funcional del sistema. La arquitectura modular desarrollada permitiría adaptar y reutilizar los componentes del *framework* en otros procesos de liquidación o cálculo fiscal de la entidad que presenten una estructura similar. Esta extensión favorecería el aprovechamiento del desarrollo realizado y reduciría el esfuerzo necesario para automatizar nuevos procesos relacionados.
- Evolución hacia la solución estratégica. Tal como se ha planteado en las conclusiones, una línea de trabajo futuro de mayor alcance consistiría en integrar la presente solución dentro de una plataforma corporativa más amplia, que unifique los distintos procesos relacionados con la gestión económica de terceros bajo una misma

arquitectura, reduciendo así la fragmentación actual entre sistemas y facilitando su mantenimiento a largo plazo.

- Incorporación de monitorización y alertas automáticas. Con el objetivo de reforzar la fiabilidad del proceso en producción, se propone incorporar mecanismos de monitorización que permitan detectar anomalías en los datos de entrada, incidencias durante la ejecución o desviaciones inesperadas en los resultados calculados. Estas alertas facilitarían la identificación temprana de errores antes de que la información fuera utilizada por los sistemas consumidores.

Capítulo 6. BIBLIOGRAFÍA

[1] W. L. Chang and N. Grady, NIST Big Data Interoperability Framework: Volume 1, Definitions, NIST Special Publication 1500-1r2. Gaithersburg, MD, USA: National Institute of Standards and Technology, 2019, doi: 10.6028/NIST.SP.1500-1r2.

[2] J. Dean and S. Ghemawat, "MapReduce: Simplified Data Processing on Large Clusters," in Proceedings of the 6th Symposium on Operating Systems Design and Implementation (OSDI '04), San Francisco, CA, USA, 2004, pp. 137–150.

[3] M. Golfarelli, S. Rizzi, and J. Vrdoljak, "Data Warehouse Design from XML Sources," in Proceedings of the 4th ACM International Workshop on Data Warehousing and OLAP, Atlanta, GA, USA, 2001, pp. 40–47.

[4] M. Derakhshannia, C. Gervet, H. Hajj-Hassan, A. Laurent, and A. Martin, "Data Lake Governance: Towards a Systemic and Natural Ecosystem Analogy," Future Internet, vol. 12, no. 8, art. no. 126, 2020, doi: 10.3390/fi12080126.

[5] M. Armbrust et al., "Lakehouse: A New Generation of Open Platforms that Unify Data Warehousing and Advanced Analytics," in Proceedings of the 11th Biennial Conference on Innovative Data Systems Research (CIDR), 2021.

[6] R. Kimball and J. Caserta, The Data Warehouse ETL Toolkit: Practical Techniques for Extracting, Cleaning, Conforming, and Delivering Data. Indianapolis, IN, USA: Wiley, 2004.

[7] M. Zaharia et al., "Resilient Distributed Datasets: A Fault-Tolerant Abstraction for In-Memory Cluster Computing," in Proceedings of the 9th USENIX Symposium on Networked Systems Design and Implementation (NSDI '12), San Jose, CA, USA, 2012, pp. 15–28.

[8] Amazon Web Services. Diferencia entre ETL y ELT. AWS. Disponible en: https://aws.amazon.com/es/compare/the-difference-between-etl-and-elt/

[9] W. M. P. van der Aalst, M. Bichler, and A. Heinzl, "Robotic Process Automation," Business & Information Systems Engineering, vol. 60, no. 4, pp. 269–272, 2018, doi: 10.1007/s12599-018-0542-4.

[10] B. von Halle, Business Rules Applied: Building Better Systems Using the Business Rules Approach. New York, NY, USA: Wiley, 2001.

[11] IBM. What is data quality? IBM Think. Available from: https://www.ibm.com/think/topics/data-quality

[12] Parlamento Europeo y Consejo de la Unión Europea. Reglamento (UE) 2016/679 del Parlamento Europeo y del Consejo, de 27 de abril de 2016, relativo a la protección de las personas físicas en lo que respecta al tratamiento de datos personales y a la libre circulación de estos datos (Reglamento General de Protección de Datos). Diario Oficial de la Unión Europea. 2016; L119:1–88. Disponible en: https://www.boe.es/doue/2016/119/L00001-00088.pdf

[13] Bolba C. Data warehouse vs data lake vs data lakehouse: Know the differences. Medium. Published 2023. Available from: https://cassio-bolba.medium.com/data-warehouse-vs-data-lake-vs-data-lakehouse-know-the-differences-51bb3f82e137

[14] Microsoft. Introducción a Azure Databricks. Microsoft Learn. Disponible en: https://learn.microsoft.com/es-es/azure/databricks/introduction/

[15] Databricks. What is a DAG (Directed Acyclic Graph)? Databricks Blog. Available from: https://www.databricks.com/es/blog/what-is-dag