UAB
Universitat Autònoma de Barcelona

Dipòsit digital de documents de la UAB

This is the **published version** of the bachelor thesis:

Pardo Barahona, Luke. *Migració i evolució de fluxos de dades RFID a arquitectures Cloud a Mango : Implementació d'un model Datalakehouse.* Treball de Final de Grau (Universitat Autònoma de Barcelona), 2026 (Grau en Enginyeria de Dades)

This version is available at https://ddd.uab.cat/record/337313

under the terms of the BY-NC-SA license.

Migració i evolució de fluxos de dades RFID a arquitectures Cloud a Mango: Implementació d'un model Datalakehouse

Luke Pardo Barahona

Resum—Aquest projecte se centra en la modernització tecnològica del flux de dades RFID a Mango, migrant d'una arquitectura Legacy cap a un model DataLakehouse basat en Databricks, DBT i AWS S3. L'objectiu principal és adequar el procés als estàndards actuals de la companyia per potenciar l'escalabilitat, el rendiment i la governança, sense alterar la lògica funcional existent. La solució implementa una Medallion Architecture (capes L0–L6) gestionada per Apache Airflow amb l'Arquitectura Launcher, Jenkins per a CI/CD i AWS S3 com a emmagatzematge Raw, garantint la continuïtat operativa i la retrocompatibilitat amb les eines de reporting actuals (Looker, SAP BO). A través del desplegament d'un entorn paral·lel de validació, s'ha aplicat una rigorosa estratègia de qualitat de dades per certificar una transició eficient assolint la paritat funcional total. Com a element innovador, s'ha avaluat la viabilitat de Copilot Agentic AI per automatitzar el modelatge dbt i s'ha validat la seva capacitat per generar codi alineat amb els estàndards corporatius.

Paraules clau—RFID, DataLakehouse, Databricks, dbt, Apache Airflow, Enginyeria de Dades, Intel·ligència Artificial, Snowflake, Data Quality, Arquitectura en Medalló, Launcher, macros, renaming, refactorització.

Abstract— This project focuses on the technological modernization of the RFID data pipeline at Mango, migrating from a legacy architecture to a DataLakehouse model based on Databricks, DBT, and AWS S3. The primary objective is to adapt the process to the company's current standards to enhance scalability, performance, and governance, without altering the existing functional logic. The solution implements a Medallion Architecture (layers L0–L6) managed by Apache Airflow with the Launcher Architecture, Jenkins for CI/CD, and AWS S3 for Raw storage, ensuring operational continuity and backward compatibility with current reporting tools (Looker, SAP BO). Through the deployment of a parallel validation environment, a rigorous data quality strategy has been applied to certify an efficient transition, achieving full functional parity. As an innovative element, the feasibility of Copilot Agentic AI for automating dbt modeling has been evaluated, and its ability to generate code aligned with corporate standards has been validated.

Index Terms— RFID, DataLakehouse, Databricks, dbt, Apache Airflow, Data Engineering, Artificial Intelligence, Snowflake, Data Quality, Medallion Architecture, Launcher, macros, renaming, refactoring.

1 Introducció

A la indústria del retail de moda global, la gestió de l'inventari i la traçabilitat del producte en temps real són factors crítics per a l'èxit operatiu i la presa de decisions estratègiques. Mango ha consolidat la tecnologia d'identificació per radiofreqüència (RFID) com un dels seus pilars operatius més importants, permetent un control precís de l'estoc al llarg de tota la seva cadena de subministrament. Aquesta tecnologia genera un volum massiu de dades d'alta freqüència, que un cop processades, tenen un alt valor analític per a la pressa de decisions, motiu pel qual, la transformació d'aquests esdeveniments en cru requereix una infraestructura tecnològica altament eficient, robusta i escalable.

Històricament, el flux de dades RFID de la companyia s'ha gestionat mitjançant una arquitectura tradicional de Data Warehousing considerada legacy. Tot i que aquest sistema basat en plataformes com Snowflake ha donat servei durant anys, l'avanç d'altres tecnologies, així com l'adopció d'un estàndard de treball diferent per part de la companyia ha posat de manifest que l'arquitectura actual impedeix una visibilitat àgil i precisa del cicle de vida de les dades i genera un deute tècnic considerable. L'existència de múltiples repositoris operant de manera aïllada, la rigidesa en l'orquestració de tasques i la dificultat per aplicar polítiques de governança globals, així com el desalineament amb l'estructura actual d'altres projectes de la companyia, ha generat un deute tècnic notable que penalitza tant el rendiment de la computació com els costos operatius.

Aquest projecte neix de la necessitat de superar les barreres d'aquesta arquitectura rígida i desactualitzada, proposant una convergència total cap al model DataLakehouse. La fragmentació actual —amb múltiples repositoris per a un mateix projecte i manca d'estandardització en el modelat i orquestració— dificulta el manteniment i la innovació, i és per aquest motiu que la migració es planteja no només com un repte tècnic, sinó com un pas necessari per democratitzar l'accés a les dades mitjançant una arquitectura

- *Luke Pardo Barahona. E-mail de contacte: luke.pardo@ autonoma.cat*
- *Treball tutoritzat per: Ramon Grau Sala (Departament d'Arquitectura de Computadors i Sistemes Operatius)*
- *Curs 2025-26*

moderna, eficient i robusta.

L'objectiu no és només resoldre el deute tècnic mitjançant una estructura de capes (Medallion Architecture) i una orquestració moderna amb Apache Airflow, sinó fer-ho garantint la continuïtat absoluta del negoci, assegurant la retrocompatibilitat de les eines de reporting actuals (Looker i SAP BO).

A més, en un context marcat per la recent explosió de la Intel·ligència Artificial, la companyia es planteja estratègicament com adoptar l'IA per optimitzar i agilitzar els seus processos transversals. En aquest sentit, s'ha considerat que aquest projecte de migració constitueix un entorn de proves idoni per testejar fins a quin punt un assistent com Github Copilot pot ajudar en el modelatge i l'automatització de dades, avaluant de forma pràctica el grau d'eficiència que pot aportar i quant de temps de desenvolupament pot arribar a estalviar a l'equip d'enginyeria.

L'objectiu principal d'aquest treball és la modernització integral i migració del flux de dades RFID de Mango cap a una arquitectura de dades de nova generació. Per assolir aquesta fita, es defineixen els següents objectius específics:

- Dissenyar i desplegar un model DataLakehouse, implementant l'arquitectura de capes a Databricks utilitzant dbt i millorant la governança i el rendiment del sistema.
- Substituir els sistemes de llançament legacy per fluxos de treball automatitzats i eficients mitjançant l'estructura Launcher Dag.
- Garantir la continuïtat de negoci assegurant zero interrupcions operatives i mantenint la retrocompatibilitat de les eines analítiques (Looker i SAP BO) sense alterar les dades.
- Avaluar l'ús d'IA en Data Engineering en la creació de codi i models de dades mitjançant l'assistent Github Copilot.

El document s'estructura de la manera següent: la secció 2 descriu el context, l'arquitectura legacy i l'estat de l'art sobre DataLakehouse i IA generativa aplicada a l'enginyeria de dades; la secció 3 detalla la metodologia, la planificació i l'arquitectura objectiu dissenyada; la secció 4 presenta els resultats obtinguts, el protocol de validació i el pla de productivització; i finalment la secció 5 exposa les conclusions del treball i les línies de continuació previstes.

2 Context i Estat actual

2.1 Context de Negoci

La tecnologia RFID permet a Mango disposar d'una traçabilitat granular de cada peça d'estoc al llarg de tota la cadena logística, des del centre de distribució fins a la botiga. Aquesta informació alimenta processos crítics com la reposició automàtica, l'inventari en temps real i l'anàlisi de la corba de talles. No obstant això, el procés de transformació digital de la companyia exigeix una infraestructura capaç de convertir aquest volum creixent de dades en informació de valor de manera escalable, cosa que l'arquitectura actual no pot garantir.

2.2 Estàndard DataLakehouse

Actualment, a Mango s'aplica un model estàndard de definició de models de dades que tots els projectes de la companyia haurien de seguir per garantir la governança i la consistència de la informació. En l'àmbit de l'enginyeria de dades, aquest estàndard es ramifica i es concreta mitjançant la diferenciació de dos repositoris de codi independents i especialitzats.

El primer d'ells és el repositori mng-dbt-xxx, destinat a la definició de les taules estructurades a dbt mitjançant el motor de plantilles Jinja, on tota la lògica de transformació s'executa íntegrament a Databricks a través de taules externes allotjades en S3. El flux dins d'aquest repositori s'inicia amb una capa Raw en cru, de la qual consumeix directament la capa Silver L0 sense patir cap mena de transformació, i a partir d'aquesta base es van construint progressivament les capes consecutives de transformació i enriquiment de negoci, anomenades Silver L1, Silver L2 i successives. Un cop completada l'última capa Silver a Databricks, es genera una taula LZS Iceberg a Snowflake, la qual opera únicament com un punter cap a la dada resident a Databricks sense duplicar-ne l'emmagatzematge físic. Finalment, es crea la capa DIM (Dimensió), que no afegeix cap lògica addicional, però serveix com a capa de consum neta a Snowflake per alimentar directament les eines de reporting corporatives com Looker o SAP BO.

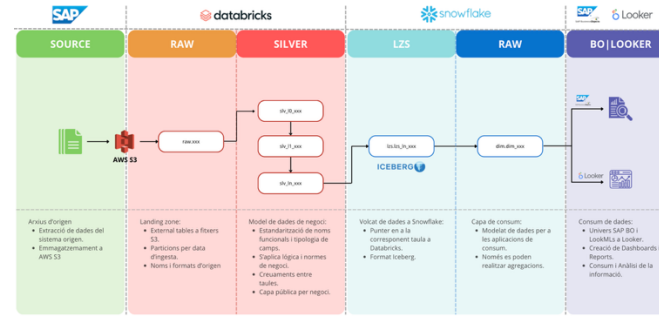

Fig. 1. Esquema de l'estàndard d'arquitectura establert a Mango, des del sistema d'origen fins a la capa de consum final.

D'altra banda, el segon repositori és *mng-airflow-xxx*, enfocat exclusivament a l'orquestració de tots els models de dades generats. Aquest repositori es basa en una abstracció pròpia de la companyia anomenada Launcher Dag, la qual centralitza de manera genèrica totes les definicions de l'orquestració dins d'Apache Airflow. A més de seqüenciar l'ordre d'execució de les tasques, aquest entorn s'encarrega de gestionar automàticament el desplegament de recursos a Amazon Web Services (AWS), controlant paràmetres crítics com la definició d'instàncies d'Amazon EC2, la configuració de les tasques i l'emmagatzematge de configuracions a AWS S3 així com les connexions amb Databricks i Snowflake. La principal virtut del Launcher Dag és que

redueix dràsticament la complexitat del graf de l'orquestrador, ja que genera únicament una sola tasca (task) per a la càrrega de dades a Databricks i una única tasca per a Snowflake, delegant per complet en dbt la responsabilitat de gestionar de manera interna totes les dependències i relacions entre taules. Gràcies a aquesta arquitectura, el procés d'expansió del projecte es simplifica al màxim: quan s'afegeix un nou model de dades al sistema, el desenvolupador només ha d'assignar-li l'etiqueta (tag) corresponent al DAG al qual pertany. D'aquesta manera, el model s'integra i s'executa automàticament dins del flux establert, facilitant enormement l'escalabilitat i el manteniment continu del projecte a llarg termini.

2.3 Arquitectura Legacy

Tradicionalment, el flux de dades RFID de Mango seguia un model pur de Data Warehousing on tota la lògica ETL (Extracció, Transformació i Càrrega) residia de manera íntegra dins de Snowflake, estructurada en quatre capes clàssiques: la capa LZS (Landing Zone Schema) per a la ingesta directa de les dades en cru; la ODS (Operational Data Store) per a la normalització tècnica de les dades; la EDW (Enterprise Data Warehouse) per a l'aplicació de la lògica de negoci complexa; i finalment, la capa DIMFAC (Dimensions i Fets) com a entorn final de consum per al reporting a través d'eines com Looker o SAP BO.

No obstant això, en un primer intent de modernització d'aquest flux, part de la lògica operativa del projecte es va traslladar a Databricks mitjançant una arquitectura en medalló (Raw-Silver-Gold) que no es corresponia amb l'estàndard corporatiu actual, generant així un nou repositori. Tanmateix, aquesta implementació no va substituir la lògica preexistent a Snowflake. Com que aquesta transició es va realitzar en una etapa anterior a la definició dels estàndards actuals de la companyia, el sistema no segueix l'estàndard actual de nomenclatures i utilitza macros marcades per a deprecació. A més no s'orquestra mitjançant el Launcher Dag, fet que obliga a generar una tasca d'Airflow independent per a cada model de dbt i a haver de definir de manera explícita i manual totes les dependències entre models, augmentant dràsticament la complexitat del manteniment i expansió del projecte.

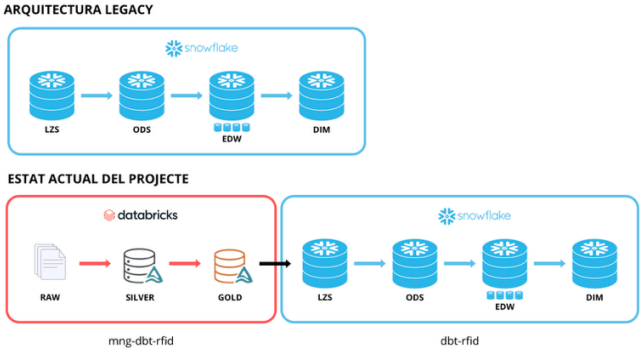

Fig. 2. Comparativa de l'arquitectura legacy original envers l'estat actual després del primer intent d'estandardització del projecte.

Com a resultat d'aquest procés incomplet, actualment existeix una duplicitat ineficient i una fragmentació en dues parelles de repositoris separades: d'una banda, mng-dbt-rfid i mng-airflow-rfid, que contenen la lògica allotjada a Databricks; i de l'altra, els repositoris legacy dbt-rfid i airflow-rfid, que continuen gestionant la part resident a Snowflake. Aquest deute tècnic i la manca d'unificació evidencien la necessitat crítica de dur a terme una reenginyeria tècnica integral per alinear el flux RFID amb els estàndards actuals de Mango.

2.4 IA Agèntica en DataOps

Donat que aquest és un projecte de migració molt extens i complex —que requereix la reestructuració i unificació de quatre repositoris fragmentats—, i en el context actual d'explosió de la Intel·ligència Artificial, la companyia es planteja com aprofitar la IA agèntica en l'àmbit de DataOps per accelerar i optimitzar el desenvolupament. A diferència dels assistents tradicionals d'autocompletat, l'arquitectura agèntica destaca per la seva capacitat de raonament per executar fluxos d'enginyeria d'extrem a extrem.

Per a la generació massiva de codi SQL, plantilles Jinja i fitxers de configuració YAML a dbt, s'ha utilitzat Copilot Agentic AI de Microsoft, que permet orquestrar models d'última generació com la família Claude Opus. Tot i que inicialment es va considerar l'ús de Claude Code per a aquesta Prova de Concepte (PoC), es va optar per la solució de Microsoft per motius estratègics de viabilitat econòmica i corporativa (en disposar ja de llicències actives a l'empresa) i per la versatilitat de la seva arquitectura per alternar entre models segons la dificultat de la tasca.

Finalment, per afrontar el repte de validar les dades durant la migració, eines natives com Databricks Genie Code i Snowflake Cortex Code aporten un valor crític. En accedir directament a les metadades i a l'historial de consultes, aquests agents automatitzen les tasques de Data Quality i el diagnòstic de discrepàncies, comparant de manera autònoma el sistema legacy amb el nou DataLakehouse per garantir-ne la paritat funcional total de forma ràpida.

3 Metodologia i Desenvolupament

La transformació del flux de dades RFID de Mango s'ha dut a terme mitjançant un conjunt d'actuacions tècniques seqüencials destinades a eliminar la fragmentació dels sistemes legacy i consolidar un model d'alta escalabilitat. El punt de partida indispensable per a aquest procés va ser la realització d'una anàlisi completa i profunda de l'estat inicial del projecte, avaluant minuciosament el deute tècnic acumulat, les limitacions dels repositoris dispersos i les dependències crítiques amb Snowflake. Aquesta auditoria exhaustiva va permetre redactar un pla de desenvolupament estratègic detallat, fixant els requeriments d'infraestructura, els criteris de qualitat i la ruta d'execució necessària per a la transició cap a la nova arquitectura.

2.4 Volumetria i Distribució del flux

Com a resultat directe d'aquesta auditoria i diagnòstic inicial, es va aconseguir quantificar amb precisió l'abast real del deute tècnic i el volum exacte d'estructures i fluxos operatius que componien l'ecosistema RFID de la companyia.

A Databricks capa Silver es van comptabilitzar un total de 45 models. D'aquests, 32 models llegeixen dades directament de la capa Raw (classificats com a capa L0), mentre que els 13 models restants consumeixen informació d'altres capes Silver (situant-se a la capa Lx). És important assenyalar que 4 d'aquests models Silver es trobaven actualment inactius en el sistema d'origen, ja que es va comprovar que no estaven inclosos en cap DAG de càrrega. En quant a la capa Gold, s'identifiquen 27 models actius que consumeixen dades directament de les estructures Silver, integrant-se plenament en la de classificació de la capa funcional Lx.

D'altre banda, a Snowflake (Capes LZS, ODS, EDW i DIM), es van trobar un total de 21 models. D'aquest grup, l'auditoria prèvia a la migració va permetre detectar que 4 models, tot i estar definits en l'arquitectura, no havien rebut cap mena de càrrega de dades en els darrers 6 mesos, evidenciant la necessitat d'una depuració.

Així mateix, l'anàlisi de l'orquestració va permetre mapejar la distribució i la freqüència de l'antic flux de treball dins d'Apache Airflow, organitzat en els següents cinc processos parametritzats:

- **Dag_dbt_rfid_daily**: Executat cada dia a la 01:00 UTC (03:00 hora local), s'encarrega de la càrrega de 16 models Silver i 9 models Gold, contenint dependències estructurals crítiques amb els models del flux horari.
- **Dag_dbt_rfid_daily_midday**: Executat cada dia a les 10:00 UTC (12:00 hora local), té la funció de carregar un únic model Silver operacional denominat rfid_shop_outbound_boxes.
- **Dag_dbt_rfid_hourly:** Executat 4 cops a dia a les 01:00 UTC, 08:00 UTC, 10:00 UTC i 14:00 UTC (03:00, 10:00, 12:00 i 15:00 hora local, respectivament), s'encarrega de la càrrega de 17 models Silver i 27 models Gold. Aquest procés conté models amb dependències directes amb el flux diari (daily).
- **Dag_dbt_rfid_weekly**: Executat un cop a la setmana, concretament els dilluns, s'encarrega de processar un únic model Gold analític de tendències denominat rfid_size_curve.
- **Dag_dbt_rfid_dwh**: Executat un cop al dia a les 03:45 UTC (05:45 hora local), s'encarrega de la càrrega de 17 models a Snowflake, iniciant el seu llançament de manera seqüencial just després de certificar que la càrrega diària (daily) a Databricks ha finalitzat correctament.

3.2 Arquitectura objectiu (TO-BE)

L'arquitectura DataLakehouse proposada centralitza la totalitat del codi en un repositori global harmonitzat i plenament integrat dins del domini de Stock. El nou flux de dades s'estructura a partir de cinc blocs funcionals i operatius clarament definits, treballant de forma connectada i seqüencial.

En primer lloc, pel que fa a la Ingesta (Raw), es manté de manera eficient l'ús d'Airbyte com a motor d'ingesta core per moure les dades de manera contínua des de les fonts d'origen cap a AWS S3, entorn d'emmagatzematge distribuït on es desen directament els fitxers estructurats en format Parquet.

Tot seguit, es passa al Processament a Databricks amb dbt (Silver L0–L6), on es migra de forma íntegra tota la lògica de transformació i càlcul que encara residia històricament a Snowflake cap a Databricks. En aquest punt s'implementa una divisió estricta en subcapes dins de l'entorn de dbt, on la capa L0 es destina en exclusiva a dades de control tècnic, mentre que les subcapes successives (Ln) assumeixen seqüencialment la lògica funcional i de negoci; un procés que s'aprofita posteriorment per reestructurar i actualitzar tota la nomenclatura d'objectes segons els estàndards genèrics globals de Mango.

Immediatament després de la capa Silver se situava la capa Gold. En el nou model, aquesta estructura deixa de ser necessària en residir tota la lògica a la capa Silver. No obstant això, i de manera temporal per no afectar els usuaris que actualment consumeixin directament d'aquesta capa, es creen vistes amb el mateix nom antic *gold.rfid_xxx*, però que apunten de forma transparent nou model equivalent a Silver.

A continuació, la Integració amb Snowflake (LZS/DIM) garanteix la persistència híbrida, ja que la darrera capa de Databricks es connecta amb Snowflake utilitzant el format de taula obert Apache Iceberg, la qual cosa li permet actuar virtualment com un punter natiu cap a la dada resident. Gràcies a aquesta interconnexió, a Snowflake només es mantenen les capes LZS —mitjançant l'ús de taules externes— i la capa DIM com a interfície de consum final, la qual s'encarrega d'incorporar directament regles i processos homogenis de deduplicació d'esdeveniments.

Finalment, la capa d'Orquestració i CI/CD tanca el cercle operatiu amb l'adopció formal de l'Arquitectura Launcher a Apache Airflow. Aquesta implementació elimina per complet la definició manual de codi a l'orquestrador per a les dependències entre taules, delegant aquesta tasca de gestió interna en el motor de dbt. Com a tancament del flux de treball, el desplegament continu de tota la infraestructura i el codi s'automatitza de forma integral mitjançant pipelines de Jenkins, assegurant un cicle de vida de programari robust, net i sense friccions.

3.3 Estratègia de migració segura – Duplicitat controlada

Per tal de garantir la continuïtat operativa i minimitzar el risc d'impacte en el negoci, s'ha implementat una estratègia de duplicitat controlada a nivell d'infraestructura global. Sota aquest enfocament, s'ha realitzar una copia de tots els models i taules afegint sufix _dlkh, per evitar la col·lisió amb els recursos existents i assegurar l'aïllament del nou desenvolupament. Aquestes noves estructures realitzen la lectura de dades directament des de les mateixes taules Raw originals, de manera que s'evita la complexitat afegida i els costos de duplicar els sistemes d'ingesta ja gestionats de forma nativa per Airbyte i Apache Kafka.

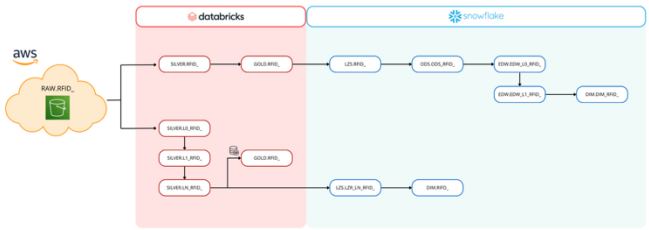

Fig. 3. Esquema del desenvolupament en paral·lel. A la part superior, s'observa el flux actual mentre que a la part inferior, es detalla el nou flux RFID (temporalment amb el sufix _dlkh).

Tanmateix, tot i compartir l'origen en cru, el nou sistema materialitza els seus resultats en rutes d'AWS S3, esquemes de Databricks i bases de dades de Snowflake completament independents i segregades de l'entorn antic. Aquesta decisió de disseny evita haver de desmantellar posteriorment una infraestructura d'ingesta duplicada i, al mateix temps, permet a l'equip d'enginyeria desenvolupar, iterar i testejar sobre els orígens de dades reals de Mango sense cap risc d'interferir o corrompre els fluxos que negoci consumeix actualment.

Gràcies a aquesta abstracció arquitectònica, es garanteix que cap eina de reporting ni procés extern d'ETL perdi l'accés a la informació o pateixi errors de connexió durant tot el període de convivència i permet validar la integritat de les dades generades pel nou flux, assegurant que no existeixen discrepàncies que alterin el resultat de les materialitzacions.

3.4 Separació en capes

El procés de desenvolupament va començar amb la reestructuració integral de la capa de dades, prioritzant la migració de la capa Gold cap a una nova estructura Silver modular L0 a L6. Tot i que inicialment es va plantejar com una etapa posterior a l'orquestració, es va decidir avançar-la, ja que l'arquitectura Launcher no suportava els models Gold, els quals han quedat obsolets, i la seva coexistència provocava comportaments inesperats en l'orquestració. Es va descartar realitzar aquesta migració de manera seqüencial per evitar la ineficiència d'un doble esforç d'enginyeria i el risc d'inconsistències, optant per una refactorització completa en un sol cicle que garantís una arquitectura robusta i escalable.

Aquesta reestructuració es va basar en un modelat progressiu de la complexitat. La L0 realitza una ingesta directa de les taules Raw sense transformacions. Les capes L1 a L6 funcionen com a capes intermèdies per a transformacions, càlculs de negoci i JOINs atòmics que permeten la reutilització de models, fins a la capa de consolidació final i consum.

Finalment, per garantir que aquesta evolució tècnica fos invisible per al negoci, es va implementar una capa de compatibilitat mitjançant vistes que repliquen els noms dels camps originals de la capa Gold, tot apuntant a la nova taula Silver Lx corresponent. Aquesta estratègia va blindar els serveis de reporting i les aplicacions externes, permetent que continuïn consumint les dades sense necessitat de reconfiguracions davant el canvi estructural a les capes inferiors.

3.5 Modelatge amb Intel·ligència Artificial Agèntica

Per fer front a la magnitud d'aquesta reestructuració i optimitzar els temps de desenvolupament, es va integrar un agent d'intel·ligència artificial (Copilot Agentic AI) sota estrictes protocols de governança de dades.

A diferència dels assistents de programació tradicionals, que actuen de manera reactiva com a simples generadors de línies de codi o consultes SQL aïllades, un agent d'IA es defineix com un sistema capaç d'entendre un objectiu complex, planificar de manera autònoma la seqüència de passos necessària, interactuar amb el seu entorn i prendre decisions basades en el context i les restriccions del projecte. En lloc d'esperar una instrucció concreta per a cada acció, l'agent raona sobre l'estat actual del codi, identifica la transformació necessària i l'executa iterativament fins a satisfer els criteris de qualitat establerts.

El desplegament d'aquest agent es va estructurar mitjançant dos mecanismes clau: els fitxers d'instruccions contextuals i les habilitats (skills) personalitzades.
Pel que fa als fitxers d'instruccions contextuals, van actuar com el marc de governança global de l'agent. Per construir-los, es va crear una carpeta compartida amb la totalitat dels projectes DataLakehouse existents a Mango, i es va encarregar a un agent d'alta capacitat la lectura i l'anàlisi d'aquests repositoris per extreure de manera autònoma els patrons arquitectònics, les convencions de nomenclatura i les regles de disseny comunes. El procés va ser iteratiu: l'agent generava una proposta d'instruccions, s'avaluava la precisió i el nivell d'abstracció, es corregien les desviacions i es tornava a executar fins a obtenir un conjunt d'instruccions precís i exhaustiu. El resultat va ser un fitxer de coneixement general sobre l'estructura de fitxers, la nomenclatura de Mango i les regles de disseny de la nova arquitectura, per a la part de dbt, i un fitxer amb el coneixement del Launcher i les seves convencions per a la part d'Airflow, que va permetre que l'agent passés de generar codi genèric a comprendre l'objectiu del projecte.

D'altra banda, les skills personalitzades van funcionar com a filtres i capacitats tècniques especialitzades per a tasques concretes que requerien una profunditat de context addicional o bé que presentaven un patró repetitiu i ben definit. Es van crear cinc skills diferenciades: una per a la separació dels models en capes Silver L0–L6, una per al trasllat de la lògica de transformació de Snowflake a Databricks, una per a la llibreria específica mng-airflow-utils, que conté totes les funcions per construir DAGs amb l'estructura Launcher Dag, una per a la substitució de macros obsoletes per les noves macros estàndard de la companyia —a excepció de la macro mng_incremental_filters—, i finalment, una per al renaming de columnes (normalització de tots els camps al nou estàndard de prefixos corporatius).

Un aspecte clau de la implementació va ser la selecció dinàmica del model d'IA en funció de la complexitat de cada tasca, amb l'objectiu d'optimitzar la relació entre qualitat i cost computacional. Per a tasques estructurades i repetitives com la substitució de macros, es van emprar models més lleugers i econòmics com Claude Sonnet 4.5, suficients per replicar patrons ben definits amb alta fiabilitat. En canvi, per a tasques d'alta complexitat contextual —com el renaming de columnes, on l'agent havia de mantenir la coherència de les referències entre desenes de models sense poder accedir a les taules materialitzades més enllà de la seva definició en dbt, o la separació en capes Silver, que requeria raonament profund sobre el lineage de les dades— va ser necessari recórrer a models de major capacitat com Claude Opus 4.7, capaços de gestionar finestres de context àmplies i mantenir la coherència semàntica al llarg de refactoritzacions extenses.

Aquesta integració va ser fonamental per automatitzar de manera sistemàtica tant la separació en subcapes de transformació com la creació de la capa final de vistes de compatibilitat. Al disposar d'un agent conscient de l'arquitectura de destí, es va reduir dràsticament el temps de refactorització manual dels antics models Gold, minimitzant els errors de sintaxi o de concordança en els canvis de nom de les variables i garantint que tot el codi resultant nasqués adaptat de manera nativa als estàndards de qualitat de la companyia.

3.6 Migració a Launcher Dag

Tot seguit, es va dur a terme la migració de l'orquestració a l'arquitectura Launcher, permetent que el motor de dbt gestioni les dependències de forma dinàmica i unificada, superant així la complexitat del model legacy.

Inicialment, per simplificar l'estructura, es va intentar agrupar l'execució del DAG daily i part de l' hourly en una única unitat (dag_dbt_daily), deixant un segon DAG (dag_dbt_frequent) per a les execucions horàries restants. No obstant això, aquesta reestructuració va generar problemes crítics: les finestres temporals d'execució van variar, provocant reprocessaments no desitjats i alterant dades que, per definició de negoci, haurien de ser immutables.

Per resoldre-ho, es va descartar l'enfocament d'agrupació total, però va sorgir el repte de gestionar les dependències entre DAGs. En el model original, cada tasca independent incloïa sensors que bloquejaven l'execució fins a finalitzar les dependències externes; en consolidar els models en una única tasca Launcher, aquesta granularitat es va perdre. Donat que el negoci exigia les dades disponibles a les 06:00 AM, es va optar per dividir l'execució de Databricks en dues sub-tasques:

- Tasca 1 (Models Independents): Agrupa els models que no tenen cap dependència amb processos externs, permetent una execució immediata i optimitzada.
- Tasca 2 (Models Dependents): Inclou tots els models i les seves cadenes (upstreams/downstreams) que tenen relacions amb altres DAGs. En el DAG daily, aquesta tasca integra un sensor que bloqueja el seu inici fins que la tasca equivalent del flux freqüent hagi finalitzat correctament. Addicionalment, es manté un sensor que controla la finalització d'un dag extern al projecte (mng-dbt-mongodb-full-stock).

Finalment, pel que fa a l'orquestració de Snowflake, es va decidir integrar el seu flux dins del DAG daily, executant-se un cop finalitzades totes les tasques de Databricks. Tot i que alguns models de Snowflake depenen d'informació generada en el DAG freqüent, es va decidir mantenir l'ordre d'execució original —posterior al daily— per evitar alteracions en les finestres temporals i assegurar la coherència de les dades. A la figura 5 es pot observar l'estructura final de DAGs del projecte.

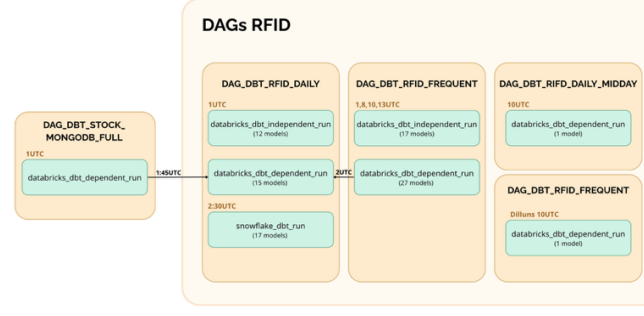

Fig. 4. Esquema de l'estructura de tots els DAGs del projecte. Per cada dag, es pot observar l'hora programada a la que s'inicia la càrrega, les tasques a realitzar amb el nombre de models a executar i les dependències (si les hi ha) entre DAGs.

3.7 Data quality

Una vegada el flux va estar completament actualitzat i abans del pas a producció, es va dur a terme un procés de validació exhaustiu per garantir la integritat de les dades processades pel nou flux i assegurar la seva total concordança amb el sistema legacy. Aquesta fase, però, va revelar inicialment tres conflictes crítics que impedien la paritat absoluta: l'existència de registres en capes d'origen absents a la destinació, registres orfes en la capa final sense

equivalent a l'origen, i la presència de valors NULL en camps definits com a claus primàries (PK).

Davant d'aquest escenari, es va prioritzar l'estabilitat del model Datalakehouse mitjançant dues decisions estratègiques. Primer, es va consensuar assumir la pèrdua d'aquells registres històrics sense consum actiu, evitant un sobrecost de reprocessament que no aportava valor analític. Segon, es va establir una directriu de tolerància zero amb els valors nuls en les claus primàries. Per resoldre-ho, es va dur a terme una revisió exhaustiva dels models a Databricks, implementant l'estratègia de transformació mitjançant la funció NVL. Aquesta solució permet castejar els valors nuls a valors per defecte estandarditzats (com 1900-01-01 per a timestamps, 999 per a codis, o 9999999999999 per a codis EAN), assegurant així la integritat del model sense descartar moviments d'estoc reals.

Un cop aplicades aquestes correccions, el protocol de Data Quality es va articular en tres nivells de verificació: una comparativa volumètrica per auditar el recompte de files, una validació d'esquemes per certificar la identitat de les metadades, i una auditoria de contingut mitjançant l'operador EXCEPT. L'agilitat d'aquest procés va ser possible gràcies a la integració de tecnologies d'IA avançades com Databricks Genie Code i Snowflake Cortex Code, els. Aquestes eines van permetre automatitzar la generació de consultes de validació i diagnosticar de forma ràpida l'origen de les discrepàncies, estalviant hores de revisió manual.

Finalment, els resultats van confirmar que les darreres divergències detectades eren exclusivament fruit de dos factors tècnics controlats. D'una banda, la nova política de claus primàries va habilitar una deduplicació nativa, detectant duplicats que el sistema antic ignorava, fet que va optimitzar la qualitat de la dada, reduint el nombre de registres duplicats. D'altra banda, l'asincronia provocada pel desfase d'execució necessari per evitar conflictes de concurrència, va comportar que el nou sistema capturés fitxers més actuals, generant discrepàncies temporals que desapareixeran un cop desactivat el flux legacy. Aquesta anàlisi, que certifica la paritat funcional de la lògica de transformació, ha estat documentada exhaustivament a Confluence, garantint la transparència total del sistema i la seva aptitud per a la producció.

3.8 Posada en producció

Un cop validada la paritat de dades i garantida la robustesa de l'entorn dlkh, es va procedir a la fase de productivització. Per evitar els riscos d'un desplegament massiu al final del projecte, es va definir una estratègia de transició controlada que va permetre substituir el sistema legacy garantint en tot moment la continuïtat operativa del negoci.

Per assegurar la integritat del traspàs, tenint en compte que la gran volumetria de les taules i la naturalesa sensible de les dades impedia realitzar càrregues completes (full-refresh) —ja que el reprocessament de l'històric podria alterar la coherència de les dades acumulades—, es va optar per una estratègia de migració controlada. En aquest sentit, es van crear les noves taules buides mitjançant la modificació temporal dels models a dbt, forçant que les càrregues tipus full-refresh generessin estructures buides mitjançant la condició 1=0. Un cop definides aquestes estructures, es van executar operacions d'inserció (INSERT INTO) per migrar la totalitat de les dades històriques als nous models, garantint així la persistència i precisió de la informació. Un cop validada la sincronització, es va procedir a la neteja de l'orquestració, desactivant els DAGs legacy a Airflow i eliminant les taules originals, no sense abans haver generat els backups corresponents per garantir la seguretat de la informació, en cas de futura fallada.

Aquesta posada en producció es va dur a terme en una finestra horària estratègica, iniciant a les 6:30 del matí finalitzant abans de la primera càrrega de les 8:00. Aquesta programació va permetre monitorar el comportament del sistema durant la resta de la jornada, garantint així una transició segura i l'estabilitat del nou entorn sota condicions reals de càrrega.

3.9 Macros i Nomenclatura

Finalment, el darrer pas de la migració va consistir en la substitució de les macros obsoletes per les noves macros estàndard de la companyia, així com en un procés exhaustiu de renaming de columnes per garantir la concordança i coherència tant dins com fora del projecte.

Pel que fa a l'actualització de macros, aquest procés es va dividir estratègicament en dues etapes per minimitzar riscos. En la primera etapa, el reemplaçament de les macros estàndard es va realitzar de manera automatitzada mitjançant l'assistència de IA. Per a fer-ho, es va crear una skill específica per a l'agent, alimentada amb la documentació interna i les directrius de governança de la companyia, la qual instruïa a l'IA sobre el procediment exacte de substitució.

No obstant això, la segona etapa es va executar de manera estrictament manual per abordar la macro mng_incremental_filters, la qual controla la gestió de la finestra de dades en les càrregues incrementals. Aquesta decisió es va prendre perquè es tracta de l'element amb major sensibilitat tècnica, ja que pot alterar directament tant les dades resultants com el comportament del model. A diferència de les macros generals, l'anàlisi de mng_incremental_filters requeria una inspecció casuística en cada model, atès que no tots els processos gestionen la càrrega incremental de la mateixa manera. Aquest enfocament mixt, combinant l'agilitat de l'IA per a tasques estandarditzades amb el rigor de l'anàlisi manual per a lògiques complexes, va garantir que la migració mantingués la integritat i la coherència funcional de tota la cadena de transformació.

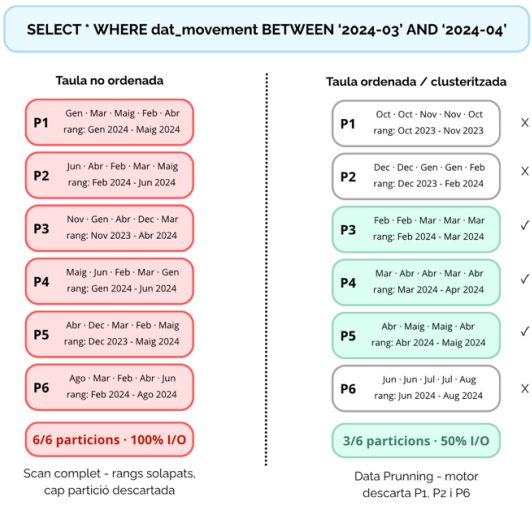

Fig. 5. Comparativa de la lectura de microparticions realitzada per el motor de Snowflake per una taula no ordenada envers una taula ordenada i/o clusteritzada.

Finalment, es va dur a terme una estandardització total de la nomenclatura de les columnes per garantir una coherència terminològica absoluta tant dins com fora del projecte. Aquest estàndard exigeix que tots els camps adoptin prefixos específics en anglès: ts_, dat_ i hou_ per a timestamps, dates i hores, val_ per a quantitats, des_ per a descriptors i is_ per a valors booleans, assegurant així que una mateixa entitat mantingui la mateixa denominació en tots els models.

Per executar aquest canvi amb eficiència, es va adoptar un enfocament híbrid: inicialment, es van realitzar manualment un parell de models com a exemple de referència, els quals van servir per entrenar i parametritzar una IA que s'encarregaria de replicar el procés en la resta del projecte. Tot i el gran volum de feina automatitzada per l'agent, va ser imprescindible realitzar una revisió manual i minuciosa, model a model, sobre els canvis registrats al worktree per verificar la precisió de les modificacions, resultant en moltes correccions humanes de la feina feta per l'agent.

La implementació d'aquests canvis de nomenclatura va requerir estratègies diferents segons la plataforma per garantir la integritat i disponibilitat dels serveis. A Databricks, el procés de renaming va obligar a recrear les taules i executar novament operacions d' INSERT INTO. Aquesta mesura va ser necessària perquè no es permet realitzar un canvi de nom de columnes sense ascendir la versió de les taules, un fet que hauria pogut comprometre l'accessibilitat des de certs notebooks existents. En canvi, a Snowflake, el sistema va permetre aplicar els canvis directament mitjançant sentències ALTER TABLE, evitant així la necessitat de recrear les estructures.

Aprofitant la intervenció a Snowflake durant la fase de renaming, es va utilitzar Snowflake Cortex Code per realitzar una anàlisi exhaustiva del rendiment de les taules i identificar colls d'ampolla en la persistència de les dades. Com a resultat, es van optimitzar diversos models mitjançant processos d'ordenació i clusterització estratègica, aplicant cada tècnica de manera selectiva en funció del comportament de càrrega de cada taula.

Per entendre l'impacte d'aquestes tècniques, cal comprendre primer com Snowflake emmagatzema físicament les dades. Quan una taula rep dades de manera incremental al llarg del temps, Snowflake les organitza internament en blocs immutables denominats micro-particions: unitats d'emmagatzematge d'entre 50 i 500 MB de dades sense comprimir, cadascuna amb metadades que registren els valors mínims i màxims de cada columna dins del bloc. Quan l'ordre d'inserció no segueix cap criteri lògic de negoci —per exemple, quan registres de dates molt dispars s'acumulen en una mateixa micro-partició—, els rangs de valors de les metadades s'amplien i se solapen entre particions, obligant el motor d'execució a llegir un nombre molt superior de blocs per respondre a qualsevol consulta filtrada per data o identificador.

L'ordenació (mitjançant una clàusula ORDER BY a la pròpia ETL) consisteix a reorganitzar les files dins de cada càrrega incremental abans d'inserir-les, de manera que les noves micro-particions generades en aquella execució continguin rangs de valors estrets i ben delimitats. És important matisar, però, que l'ordenació només actua sobre els registres de la càrrega actual: no altera l'ordre físic de les microparticions ja existents a la taula. Per a les taules on cada execució insereix o actualitza únicament dades del dia en curs —és a dir, on la finestra temporal de la càrrega és sempre recent i no solapada amb càrregues anteriors—, l'ordenació a la ETL és suficient per garantir el data pruning, ja que cada execució genera naturalment micro-particions amb rangs no solapats respecte a les anteriors.

La clusterització, en canvi, és el mecanisme que Snowflake ofereix per reorganitzar de manera automàtica i contínua les micro-particions ja existents a la taula, independentment de quan van ser inserides. Aquesta tècnica es va aplicar selectivament només a les taules on la càrrega incremental insereix habitualment registres amb dates del passat —per exemple, correccions retroactives de moviments d'estoc o relectures de dades tardanes—, ja que en aquests casos un ORDER BY a la ETL no és suficient: els nous registres passats es barregen físicament amb micro-particions antigues que ja existien, ampliant els seus rangs i anul·lant el data pruning aconseguit prèviament. La clusterització resol aquest problema reorganitzant periòdicament tota la taula, però implica un cost de computació continu que no és justificable per a taules que mai insereixen dades retroactives.

El benefici directe d'ambdues tècniques és el Data Pruning: quan el motor d'execució rep una consulta amb un filtre concret, pot comparar el filtre amb les metadades de cada micro-partició i descartar automàticament tots els blocs el rang dels quals no intersecciona amb el valor buscat, sense necessitat de llegir-los. Amb taules desordenades, el solapament de rangs obliga a llegir la majoria de particions;

amb taules ordenades o clusteritzades, el motor pot eliminar fins al 70-90% dels blocs de lectura abans d'executar la consulta, reduint dràsticament l'entrada/sortida (I/O) i l'ús de CPU.

El balanç de rendiment obtingut és altament favorable. Tot i que l'ordenació i la clusterització van comportar un increment marginal del 7% en els temps de càrrega de les ETL —a causa del cost computacional associat a la reestructuració física de les dades en cada ingesta incremental—, l'impacte en l'experiència de l'usuari final ha estat molt significatiu: s'ha assolit una reducció mitjana del 30% en el temps de resposta de les consultes realitzades des de plataformes de reporting com Looker o SAP BO. Aquestes xifres confirmen que el desplaçament d'una petita fracció de temps des de la càrrega cap a l'explotació de la dada ha estat una estratègia encertada, garantint un sistema molt més àgil i escalable per als usuaris de negoci.

4 Resultats

Els resultats assolits després de la reenginyeria han superat les expectatives inicials, no només aconseguint la paritat funcional i l'estandardització, sinó també una optimització rellevant en l'eficiència operativa del sistema.

Pel que fa al rendiment, la migració ha permès una reducció notable dels temps d'execució: els fluxos freqüent han passat d'uns 45 minuts a intervals d'entre 20 i 30 minuts, mentre que el procés daily s'ha escurçat de 2 hores a 1 hora i 30 minuts. Una part fonamental d'aquesta millora s'atribueix al canvi d'infraestructura de warehouse, passant d'un model classic a un serverless, que redueix els temps d'inici del warehouse sobre el que s'executa el codi de 10min a 1, per un cost per minut mes elevat. No obstant, s'ha mesurat que la reducció en el temps d'execució, compensa l'increment en el preu per minut.

Més enllà de les mètriques, el valor d'aquest projecte resideix en la seva nova cohesió operativa. En alinear el codi amb els estàndards corporatius, s'ha eliminat l'ambigüitat tècnica que presentava el model legacy, convertint l'entorn en un espai molt més intuïtiu i còmode per als desenvolupadors. L'adopció d'una nomenclatura estandarditzada i l'ús de macros robustes faciliten que qualsevol membre de l'equip pugui entendre i mantenir els models sense friccions, assegurant que el projecte sigui escalable i fàcil d'evolucionar. En definitiva, s'ha passat d'una arquitectura complexa a un ecosistema net i totalment integrable, consolidant el Datalakehouse com un actiu sòlid per a l'empresa.

5 Conclusions

L'estat actual de la reenginyeria del sistema de dades RFID permet extreure un conjunt de conclusions sòlides sobre la viabilitat de la nova arquitectura i el valor aportat a l'organització. A continuació, es detallen les conclusions articulades en diferents dimensions: tècnica, operativa, estratègica i d'aprenentatge personal.

5.1 Assoliment de fites i paritat funcional

La conclusió principal és la confirmació que el model Datalakehouse dissenyat és funcionalment equivalent al sistema legacy, però amb una eficiència estructural molt superior. Mitjançant les proves de paritat executades tant a Databricks com a Snowflake, s'ha demostrat que la lògica de negoci s'ha preservat íntegrament. Les discrepàncies detectades no es van deure a errors de càlcul, sinó a la correcció de deute tècnic acumulat (com la gestió de valors nuls) i a la millora de la precisió en el tractament de les càrregues incrementals. Aquest fet valida que una migració d'aquesta complexitat es pot realitzar sense comprometre la integritat de la dada històrica, assegurant que els departaments de negoci puguin seguir confiant en la informació per a la presa de decisions.

5.2 Robustesa de la nova orquestració

L'arquitectura Launcher basada en Airflow ha demostrat ser la solució òptima per a la gestió de dependències complexes. La resolució del problema amb els sensors de temps —mitjançant la implementació de filtres per schedule— posa de manifest que la flexibilitat del disseny permet corregir anomalies en temps real sense comprometre la lògica global. S'ha passat d'un sistema rígid a un entorn resilient on els sensors són capaços d'identificar l'execució exacta i continuar el flux de manera autònoma, garantint que la necessitat de disposar de les dades a les 05:00 UTC es compleixi de manera constant.

5.3 L'impacte de la IA en l'enginyeria de dades

L'impacte de la Intel·ligència Artificial en aquest projecte ha representat un canvi de paradigma fonamental, consolidant-se com un element disruptiu que ha redefinit les capacitats de l'equip d'enginyeria de dades. La integració d'eines com Copilot Agentic AI, Snowflake Cortex Code i Databricks Genie ha actuat com un autèntic multiplicador de la productivitat. Aquesta simbiosi tecnològica ha permès reduir dràsticament el temps invertit en tasques mecàniques i repetitives, com ara la generació de fitxers de configuració YAML o la redacció de consultes de diagnòstic.

Un exemple clar de l'eficiència guanyada és la planificació temporal: mentre que l'estimació inicial del projecte, prèvia a la implementació d'aquestes eines, calculava una durada d'uns 5 mesos, la col·laboració amb la IA ha permès completar la reenginyeria en aproximadament 3,5 mesos. A més, la IA ha facilitat l'auditoria de milions de registres per identificar patrons de discrepància complexos que haurien requerit setmanes de diagnòstic manual. Malgrat aquesta velocitat, l'experiència confirma que la supervisió humana continua sent imprescindible per avaluar l'impacte funcional real de les dades, ja que la IA realitza una replicació tècnica que requereix, ineludiblement, el criteri i el context de negoci de l'enginyer.

5.4 Optimització de recursos i eficiència de costos

La reenginyeria ha comportat una optimització profunda dels recursos computacionals. L'eliminació de models deprecats i la supressió de capes intermèdies legacy ha reduït de manera dràstica la computació tant a Databricks com a Snowflake. Aquesta neteja, combinada amb la gestió estratègica del pas de warehouses classic a serverless, ha permès fer més amb menys, simplificant els fluxos i reduint els temps totals de càrrega. El projecte ha permès adquirir coneixements crítics en FinOps, aprenent a ajustar la potència de càlcul segons les necessitats reals i garantint un model de dades financerament sostenible.

5.5 Reflexió final sobre el rol del Data Engineer

A nivell professional, el projecte permet concloure que la figura del Data Engineer ha evolucionat d'un simple gestor de pipelines a un arquitecte de solucions transversals. La capacitat de coordinar-se amb els equips de Data Platform i de negoci, dissenyar plans de transició detallats i col·laborar amb agents d'IA, ha estat fonamental. En resum, el sistema és ara més robust, ràpid i segur, preparat per afrontar els reptes futurs del volum de dades RFID a Mango gràcies a un pla de transició detallat, una documentació exhaustiva a Confluence i una execució orquestrada que ha garantit l'èxit de la transformació.

AGRAÏMENTS

Vull expressar el meu sincer agraïment a l'equip de Platform Data Engineering de Mango per la seva orientació en l'estandardització dels processos i el suport en la migració cap a l'arquitectura DataLakehouse. Sense la seva visió estratègica i el suport en la resolució de dubtes i problemes, aquest projecte no hauria assolit el seu nivell d'eficiència actual.

De la mateixa manera, vull agrair a l'equip de Data Engineering - Stock la seva confiança i col·laboració constant. El seu suport en la validació funcional dels models i la seva dedicació per garantir que la transició fos transparent per al negoci han estat fonamentals per mantenir la continuïtat operativa durant aquests mesos.

Finalment, vull agrair a tots els companys que han compartit el seu coneixement i m'han donat suport tècnic durant el desenvolupament d'aquesta reenginyeria. La col·laboració entre equips ha estat el pilar central que ha permès assolir aquests resultats amb èxit.

BIBLIOGRAFIA

[1] Mango (2024). Arquitectura – Puntos básicos de un desarrollo. Departament Data & AI, Equip Data Engineering Cloud. Document intern de caràcter confidencial.

[2] Mango (2022). dbtLauncher – Introducción al Launcher. Departament Data & AI, Equip Data Engineering Cloud. Document intern de caràcter confidencial.

[3] Databricks (s.f.). Data warehousing concepts. Disponible a: https://docs.databricks.com/aws/en/sql/get-started/data-warehousing-concepts

[4] dbt Labs (s.f.). Incremental models. Disponible a: https://docs.getdbt.com/docs/build/incremental-models

[5] Microsoft (s.f.). Custom instructions for GitHub Copilot in VS Code. Disponible a: https://code.visualstudio.com/docs/copilot/customization/custom-instructions

[6] Abhishek, T. (2025). Best practices for dbt and Snowflake integration: A hands-on guide for 2025. Medium. Disponible a: https://medium.com/tech-with-abhishek/best-practices-for-dbt-and-snowflake-integration

[7] Apache Software Foundation (s.f.). Apache Airflow Documentation. Disponible a: https://airflow.apache.org/docs/

[8] Snowflake Inc. (s.f.). Cortex AI: Code generation & analysis. Disponible a: https://docs.snowflake.com/en/user-guide/cortex-code/cortex-code

[9] Databricks (2024). Introducing Genie: The AI-powered data assistant. Disponible a: https://www.databricks.com/es/blog/introducing-genie-code

[10] GitHub (s.f.). Adding a GPG key to your GitHub account. Disponible a: https://docs.github.com/en/authentication/managing-commit-signature-verification/adding-a-gpg-key-to-your-github-account

[11] Mango (2025). Dbt Macros. Departament Data & AI, Equip Data Engineering. Document intern de caràcter confidencial.

[12] Mango (2025). Dbt Development – Filtros incrementales. Departament Data & AI, Equip Data Engineering. Document intern de caràcter confidencial.

[13] Abhishek, T. (2025). Best practices for dbt materializations in Snowflake with pros & cons. Medium. Disponible a: https://medium.com/tech-with-abhishek/best-practices-for-dbt-materializations-in-snowflake

[14] Databricks (s.f.). Data engineering: Tables and views. Disponible a: https://docs.databricks.com/aws/en/data-engineering/tables-views