MÁSTER UNIVERSITARIO EN BIG DATA

TRABAJO FIN DE MÁSTER

DISEÑO E IMPLEMENTACIÓN DE UNA ARQUITECTURA DATA VAULT PARA ANALÍTICA DE DATOS EN UN ENTORNO E-COMMERCE UTILIZANDO SERVICIOS CLOUD

Autor: Miguel Ubierna Gutiérrez

Director: Federico Esteban Luna

Madrid

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

Diseño e implementación de una arquitectura Data Vault para analítica de datos en un entorno e-commerce utilizando servicios Cloud

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.: Miguel Ubierna Gutiérrez Fecha:22/06/2026

Autorizada la entrega del proyecto

EL DIRECTOR DEL PROYECTO

Fdo.: Federico Esteban Luna Fecha: 22/06/2026

Vº Bº del Coordinador de Proyectos

Fdo.: Carlos Morrás Ruiz-Falcó Fecha: 22/06/2026

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. **Miguel Ubierna Gutiérrez**

DECLARA ser el titular de los derechos de propiedad intelectual de la obra: **Diseño e implementación de una arquitectura Data Vault para analítica de datos en un entorno e-commerce utilizando servicios Cloud**, 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.

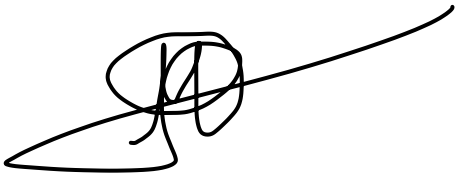
6º. Fines y funcionamiento del Repositorio Institucional.

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 22 de Junio de 2026

ACEPTA



Fdo. Miguel Ubierna Gutiérrez

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

MÁSTER UNIVERSITARIO EN BIG DATA

TRABAJO FIN DE MÁSTER

DISEÑO E IMPLEMENTACIÓN DE UNA ARQUITECTURA DATA VAULT PARA ANALÍTICA DE DATOS EN UN ENTORNO E-COMMERCE UTILIZANDO SERVICIOS CLOUD

Autor: Miguel Ubierna Gutiérrez

Director: Federico Esteban Luna

Madrid

DISEÑO E IMPLEMENTACIÓN DE UNA ARQUITECTURA DATA VAULT PARA ANALÍTICA DE DATOS EN UN ENTORNO E-COMMERCE UTILIZANDO SERVICIOS CLOUD

Autor: Ubierna Gutiérrez, Miguel.
Director: Esteban Luna, Federico.
Entidad Colaboradora: ICAI – Universidad Pontificia Comillas

RESUMEN DEL PROYECTO

Desarrollo de una arquitectura end-to-end de datos aplicada a un entorno de e-commerce. El proyecto cubre todo el ciclo de vida del dato desde la ingestión de datos en bruto de múltiples fuentes hasta la explotación analítica para la ayuda en la toma de decisiones del negocio. El núcleo técnico reside en el diseño de un DataWarehouse basado en una arquitectura DataVault para almacenar y gestionar el histórico. La arquitectura y los servicios serán desplegados utilizando herramientas cloud de Azure.

Palabras clave: Data Warehouse, Data Vault, Azure, Business Intelligence, ETL, Links, Hubs, Satélites, Star Schema, Data Lake, API, Historificación, Trazabilidad, Escalabilidad, Datos Sintéticos, Insights, Analítica, Raw Vault, Business Vault, Data Mart.

1. Introducción

En los entornos empresariales actuales, los datos suelen encontrarse repartidos entre distintos sistemas, aplicaciones y proveedores externos. Esto hace que, si no existe una arquitectura que unifique los datos, sea difícil tener una visión completa y fiable del negocio. Por este motivo, los Data Warehouse siguen siendo una pieza importante en los proyectos de datos, ya que permiten reunir información de distintas fuentes y prepararla para su análisis.

Este proyecto nace a partir de una situación parecida a la que puede darse en un trabajo real de consultoría tecnológica. Al no poder utilizar datos reales de empresa por motivos de confidencialidad, se decidió construir un caso propio con datos sintéticos. Para ello, se eligió un entorno e-commerce, ya que permite trabajar con varias entidades relacionadas entre sí, como clientes, pedidos, líneas de pedido, productos, categorías y envíos.

El objetivo del trabajo no ha sido únicamente almacenar datos, sino construir un flujo completo desde la generación e ingesta de la información hasta su visualización final. De esta forma, el proyecto cubre distintas fases habituales en una arquitectura analítica: preparación de fuentes, ingesta, almacenamiento inicial, transformación, modelado Data Vault, construcción de data marts y explotación mediante dashboards.

2. Definición del proyecto

El proyecto se ha planteado como el diseño e implementación de una arquitectura de datos capaz de simular un entorno empresarial real. Para ello, se han generado datos sintéticos mediante scripts en Python, buscando que las distintas entidades mantuviesen

coherencia entre sí y que el conjunto de datos pudiera utilizarse después en un flujo analítico completo.

En este escenario se han definido varias entidades propias de un entorno e-commerce, como clientes, pedidos, líneas de pedido, productos, categorías y envíos. La generación de datos se ha realizado teniendo en cuenta estas relaciones. Por ejemplo, los pedidos se asocian a clientes existentes, las líneas de pedido se relacionan con productos válidos y los envíos se vinculan con pedidos generados previamente. Esto permite que los datos resultantes puedan cargarse, transformarse y analizarse de forma parecida a como ocurriría en un entorno empresarial.

Además de generar los datos, se han simulado distintas fuentes de información. Los datos de clientes, pedidos y líneas de pedido se exponen mediante APIs, representando sistemas externos como un CRM o un ERP. Por otro lado, los productos y categorías se reciben en ficheros CSV, mientras que la información de envíos se trabaja en formato JSON, estos ficheros estaban almacenados en un Data Lake en la nube. Esta variedad de formatos permite representar un escenario más realista, donde no todos los sistemas entregan la información de la misma manera ni a través del mismo canal.

De esta forma, la definición del proyecto queda centrada en reproducir un caso empresarial manejable, pero suficientemente completo para trabajar con problemas habituales en proyectos de datos: variedad de fuentes, distintos formatos, relaciones entre entidades, cargas automatizadas y necesidad de preparar la información antes de poder analizarla.

3. Descripción del modelo/sistema/herramienta

La arquitectura desarrollada se organiza en varias capas, cada una con una función concreta dentro del flujo de datos. En primer lugar, los datos se cargan en la capa raw, donde se conservan en un estado cercano al origen. Esta capa permite mantener una primera copia de la información y facilita posibles reprocesos posteriores.

A partir de raw se construyen vistas stage, encargadas de preparar la información antes de cargarla en el modelo Data Vault. En esta fase se realizan tareas como tipado de campos, normalización de fechas, limpieza de formatos y generación de claves técnicas. La decisión de utilizar vistas permite evitar duplicar almacenamiento innecesario y mantener la lógica de preparación separada de los datos originales.

El núcleo del Data Warehouse se ha implementado mediante Data Vault. En la Raw Vault se modelan las entidades principales mediante hubs, las relaciones mediante links y los atributos descriptivos e históricos mediante satélites. Esta estructura permite mantener la trazabilidad de los datos, conservar su evolución en el tiempo y facilitar la incorporación de nuevas fuentes sin rediseñar por completo el modelo.

Sobre esta base se construye la Business Vault, donde se aplican reglas de negocio que no forman parte directamente del dato de origen, pero que ayudan a preparar la información para su análisis posterior. Finalmente, los data marts se organizan mediante un modelo dimensional en esquema estrella, facilitando el consumo de los datos desde Power BI.

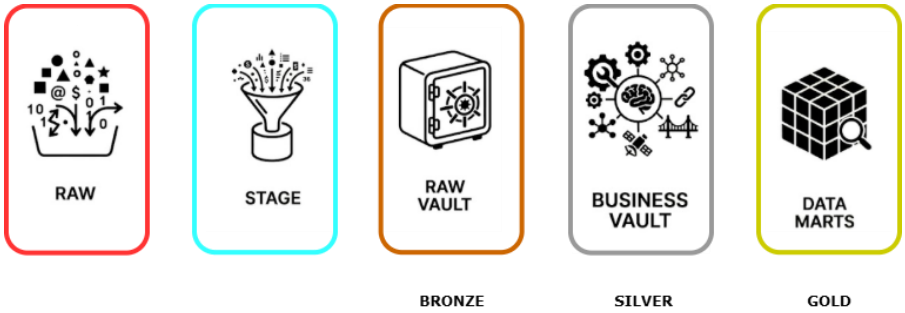


Ilustración 1. Capas del flujo de datos en Data Vault

Toda la arquitectura se ha desplegado sobre servicios cloud de Microsoft Azure, integrando distintos componentes para que trabajen de forma coordinada dentro de un mismo sistema. En este flujo, Azure Functions se utiliza para simular la exposición de datos mediante APIs, Azure Data Lake Storage Gen2 permite almacenar los ficheros de entrada y Azure SQL Database actúa como base principal del Data Warehouse.

El componente encargado de coordinar el movimiento de los datos es Azure Data Factory. A través de sus pipelines se orquestan las cargas desde las distintas fuentes hasta las capas internas del modelo. Estos procesos se han planteado como cargas batch, normalmente ejecutadas en ventanas de baja actividad, ya que el proyecto no requiere procesamiento en tiempo real.

De esta forma, los pipelines no solo permiten automatizar la entrada de datos, sino también controlar el orden de ejecución de las distintas fases del sistema. Esto resulta importante porque algunas capas dependen de que otras se hayan cargado previamente. La siguiente figura resume de forma general la arquitectura desarrollada y el papel de los distintos servicios dentro del flujo.

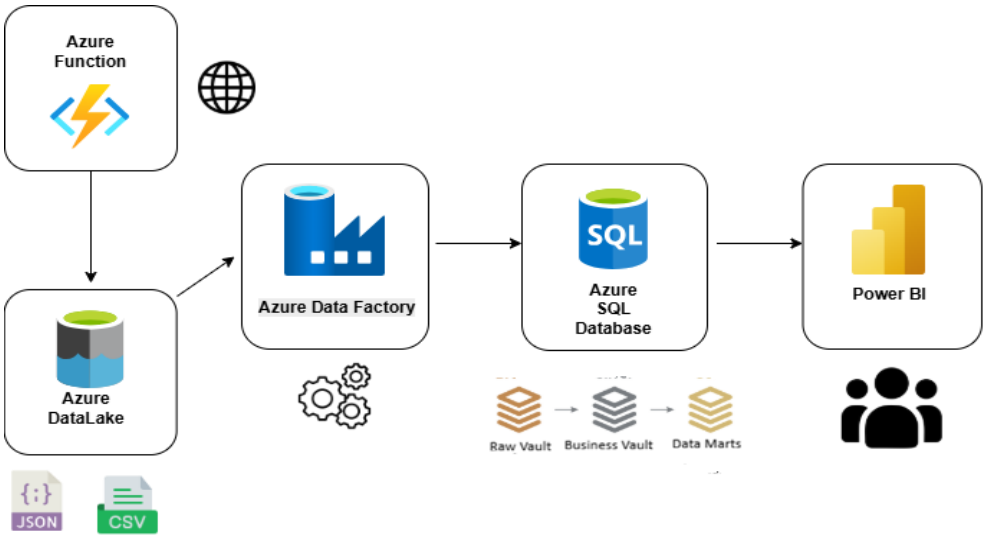


Ilustración 2 – Arquitectura en cloud del sistema desarrollado

4. Resultados

Una vez construidos los data marts, la información final se ha explotado mediante dashboards en Power BI. Estos cuadros de mando permiten analizar el negocio desde distintas perspectivas: clientes, ventas y productos. Aunque los datos utilizados son sintéticos, los dashboards representan el tipo de análisis que podría realizarse en un entorno empresarial real.

En el dashboard de clientes se observa la distribución entre clientes fidelizados y no fidelizados, así como la relación entre número de pedidos e importe gastado. Este análisis permite identificar perfiles de clientes con mayor valor, estudiar la antigüedad de la base de clientes y detectar posibles oportunidades de fidelización.

En el dashboard de ventas se analiza la facturación total, los métodos de pago utilizados, el estado de los envíos y la distribución geográfica de los pedidos. Los resultados permiten obtener una visión general del rendimiento comercial del e-commerce, así como identificar patrones relacionados con pagos, logística y concentración de pedidos por ciudad.

Por último, el dashboard de productos permite analizar marcas, productos más vendidos y valoraciones por categoría. Este tipo de información puede ayudar a tomar posibles decisiones relacionadas con stock, promociones o gestión del catálogo.

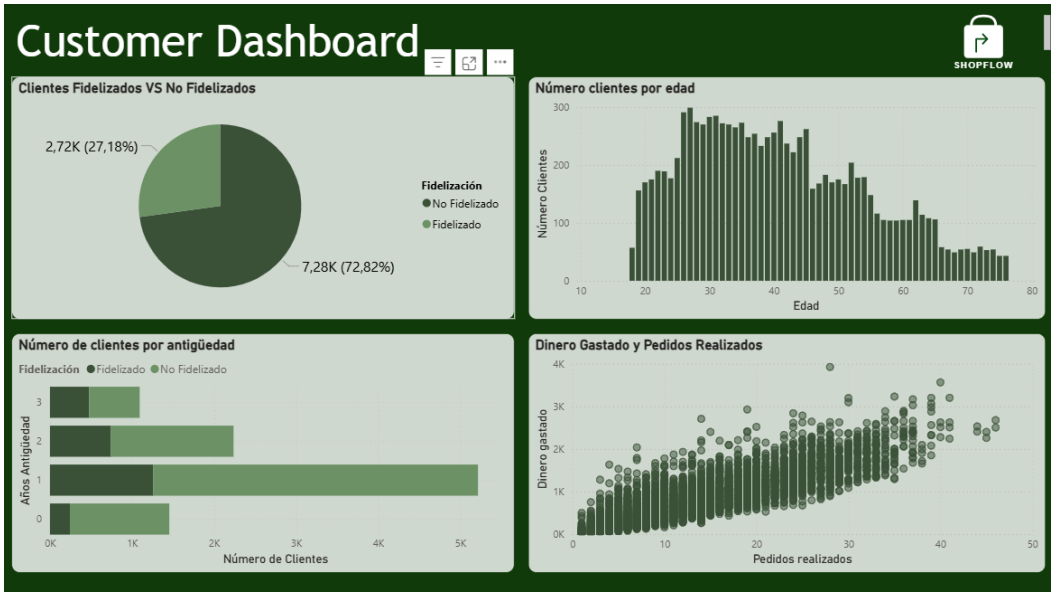


Ilustración 3 – Ejemplo de un dashboard desarrollado en Power BI

5. Conclusiones

El proyecto ha permitido construir una arquitectura de datos completa, desde la generación de datos sintéticos hasta la explotación final en Power BI. A lo largo del desarrollo se ha comprobado la importancia de separar correctamente las distintas fases del flujo de datos, evitando mezclar ingesta, transformación, historificación y análisis en una única estructura.

Una de las principales aportaciones del trabajo ha sido la implementación de un Data Warehouse basado en Data Vault. Este enfoque ha resultado adecuado para integrar

información de distintas fuentes, mantener trazabilidad, conservar histórico y organizar el modelo de forma escalable. El uso de claves hash, satélites historificados y campos técnicos de auditoría permite seguir el recorrido del dato desde su origen hasta las capas analíticas finales.

También se ha demostrado la importancia de construir data marts orientados al análisis. Aunque Data Vault aporta una base sólida para integración e histórico, no es el modelo más cómodo para la explotación directa desde herramientas de BI. Por ello, la creación de Star Schemas permite ofrecer a Power BI una estructura más simple, clara y preparada para construir indicadores y visualizaciones.

Como trabajo futuro, la arquitectura podría ampliarse incorporando nuevas fuentes de información, como campañas de marketing, comportamiento web, atención al cliente o datos de proveedores externos. Gracias al enfoque Data Vault, estas ampliaciones podrían realizarse de forma progresiva, añadiendo nuevas entidades, relaciones o satélites sin tener que rehacer toda la estructura existente.

Otra posible línea de evolución sería adaptar la solución a un entorno de mayor escala, utilizando servicios como Azure Synapse Analytics o Microsoft Fabric, así como incorporando controles de calidad más avanzados, monitorización automática y modelos analíticos predictivos.

6. Referencias

[1] Microsoft. “Implementing Data Vault 2.0 on Fabric Data Warehouse”. Microsoft Community. Disponible en: https://techcommunity.microsoft.com/blog/analyticsonazure/implementing-data-vault-2-0-on-fabric-data-warehouse/4227078

[2] Microsoft. “Data Vault 2.0 on Azure”. Microsoft Community. Disponible en: https://techcommunity.microsoft.com/blog/analyticsonazure/data-vault-2-0-on-azure/3860665

[3] Linstedt, D.; Olschimke, M. Building a Scalable Data Warehouse with Data Vault 2.0. Morgan Kaufmann, 2015.

[4] Linstedt, D. Super Charge Your Data Warehouse: Invaluable Data Modeling Rules to Implement Your Data Vault. CreateSpace Independent Publishing Platform, 2011.

DESIGN AND IMPLEMENTATION OF A DATA VAULT ARCHITECTURE FOR DATA ANALYTICS IN AN E-COMMERCE ENVIRONMENT USING CLOUD SERVICES

Author: Ubierna Gutiérrez, Miguel.
Supervisor: Esteban Luna, Federico.
Collaborating Entity: ICAI – Universidad Pontificia Comillas

ABSTRACT

Development of an end-to-end data architecture applied to an e-commerce environment. The project covers the full data lifecycle, from the ingestion of raw data from multiple sources to its analytical exploitation to support business decision-making. The technical core of the project is based on the design of a Data Warehouse using a Data Vault architecture, allowing historical data to be stored and managed. The architecture and services are deployed using Azure cloud tools.

Keywords: Data Warehouse, Data Vault, Azure, Business Intelligence, ETL, Links, Hubs, Satellites, Star Schema, Data Lake, API, Historification, Traceability, Scalability, Synthetic Data, Insights, Analytics, Raw Vault, Business Vault, Data Mart.

1. Introduction

In today's business environments, data is often distributed across different systems, applications and external providers. This means that, without an architecture capable of unifying data, it becomes difficult to obtain a complete and reliable view of the business. For this reason, Data Warehouses continue to play an important role in data projects, as they make it possible to bring together information from different sources and prepare it for analysis.

This project arises from a situation similar to one that may occur in a real technological consulting project. Since real company data could not be used due to confidentiality reasons, a custom case study was created using synthetic data. An e-commerce environment was chosen because it allows several related entities to be represented, such as customers, orders, order lines, products, categories and shipments.

The objective of the project has not only been to store data, but to build a complete flow from the generation and ingestion of information to its final visualization. In this way, the project covers several common stages in an analytical architecture: source preparation, ingestion, initial storage, transformation, Data Vault modelling, data mart construction and dashboard-based exploitation.

2. Project Definition

The project has been designed as the development and implementation of a data architecture capable of simulating a real business environment. To achieve this, synthetic data has been generated using Python scripts, ensuring that the different entities remain coherent with each other and that the dataset can later be used in a complete analytical flow.

In this scenario, several entities typical of an e-commerce environment have been defined, such as customers, orders, order lines, products, categories and shipments. Data generation has been carried out taking these relationships into account. For example, orders are associated with existing customers, order lines are linked to valid products, and shipments are connected to previously generated orders. This allows the resulting data to be loaded, transformed and analysed in a similar way to how it would be done in a business environment.

In addition to generating the data, different sources of information have been simulated. Customer, order and order line data are exposed through APIs, representing external systems such as a CRM or an ERP. On the other hand, products and categories are received in CSV files, while shipment information is handled in JSON format. These files are stored in a cloud Data Lake. This variety of formats helps represent a more realistic scenario, where not all systems provide information in the same way or through the same channel.

In this way, the project definition focuses on reproducing a manageable business case, while still being complete enough to work with common challenges in data projects: source variety, different formats, relationships between entities, automated loads and the need to prepare information before it can be analysed.

3. Description of the model/system/tool

The developed architecture is organised into several layers, each with a specific role within the data flow. First, data is loaded into the raw layer, where it is kept in a state close to the original source. This layer makes it possible to maintain an initial copy of the information and facilitates possible future reprocessing.

From the raw layer, stage views are built to prepare the information before loading it into the Data Vault model. At this stage, tasks such as field typing, date normalization, format cleansing and technical key generation are carried out. The decision to use views avoids unnecessary storage duplication and keeps the preparation logic separate from the original data.

The core of the Data Warehouse has been implemented using Data Vault. In the Raw Vault, the main entities are modelled through hubs, relationships through links, and descriptive and historical attributes through satellites. This structure makes it possible to maintain data traceability, preserve its evolution over time and incorporate new sources without redesigning the whole model.

On top of this base, the Business Vault is built, where business rules that are not directly part of the source data are applied, helping to prepare the information for later analysis. Finally, the data marts are organised through a dimensional model based on a star schema, making data consumption from Power BI easier.

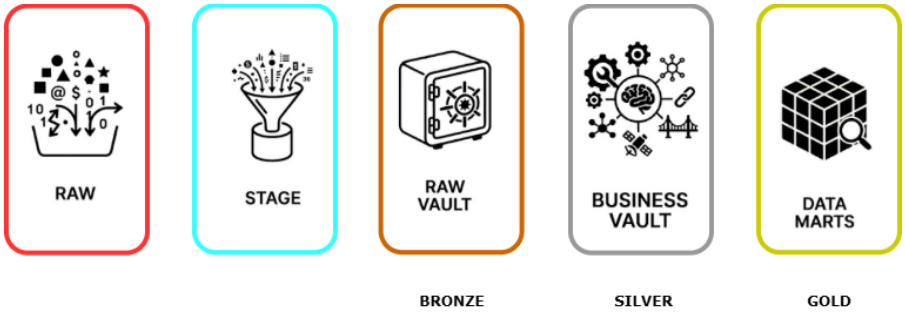


Figure 1. Layers of the Data Vault data flow

The entire architecture has been deployed using Microsoft Azure cloud services, integrating different components so that they work together within the same system. In this flow, Azure Functions is used to simulate data exposure through APIs, Azure Data Lake Storage Gen2 stores the input files, and Azure SQL Database acts as the main database for the Data Warehouse.

The component responsible for coordinating data movement is Azure Data Factory. Through its pipelines, loads are orchestrated from the different sources to the internal layers of the model. These processes have been designed as batch loads, usually executed during low-activity time windows, since the project does not require real-time processing.

In this way, pipelines not only automate data ingestion, but also control the execution order of the different stages of the system. This is important because some layers depend on others being loaded beforehand. The following figure provides a general overview of the developed architecture and the role of the different services within the flow.

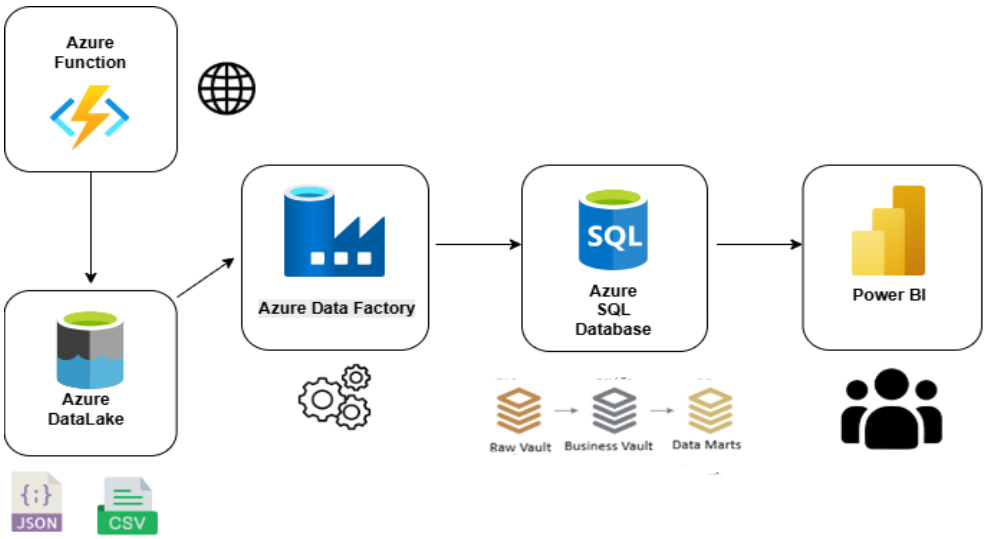


Figure 2. Cloud architecture of the developed system

4. Results

Once the data marts have been built, the final information has been exploited through Power BI dashboards. These dashboards make it possible to analyse the business from different perspectives: customers, sales and products. Although the data used is synthetic, the dashboards represent the type of analysis that could be carried out in a real business environment.

In the customer dashboard, the distribution between loyal and non-loyal customers can be observed, as well as the relationship between the number of orders and the amount spent. This analysis helps identify higher-value customer profiles, study the age of the customer base and detect possible loyalty opportunities.

In the sales dashboard, total revenue, payment methods, shipment status and the geographical distribution of orders are analysed. The results provide an overview of the commercial performance of the e-commerce business, while also identifying patterns related to payments, logistics and order concentration by city.

Finally, the product dashboard makes it possible to analyse brands, best-selling products and ratings by category. This type of information can help support possible decisions related to stock management, promotions or catalogue management.

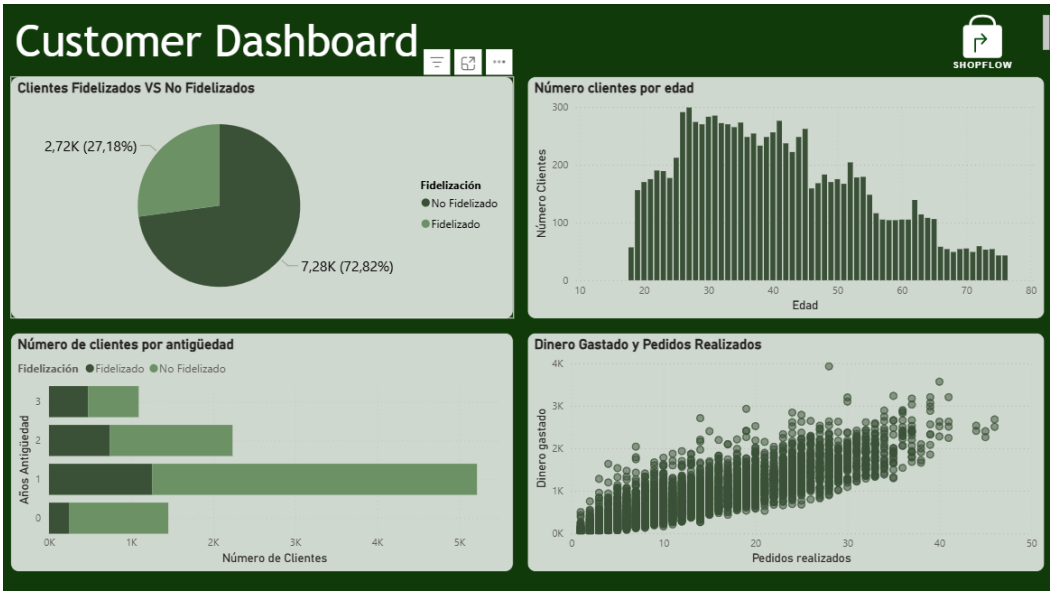


Figure 3. Example of a dashboard developed in Power BI

5. Conclusions

The project has made it possible to build a complete data architecture, from synthetic data generation to final exploitation in Power BI. Throughout the development, the importance of correctly separating the different stages of the data flow has been verified, avoiding the combination of ingestion, transformation, historification and analysis within a single structure.

One of the main contributions of the project has been the implementation of a Data Warehouse based on Data Vault. This approach has proven suitable for integrating information from different sources, maintaining traceability, preserving historical data

and organising the model in a scalable way. The use of hash keys, historified satellites and technical audit fields makes it possible to follow the path of the data from its origin to the final analytical layers.

The project has also shown the importance of building data marts oriented towards analysis. Although Data Vault provides a solid foundation for integration and historical tracking, it is not the most convenient model for direct exploitation from BI tools. For this reason, the creation of Star Schemas allows Power BI to work with a simpler, clearer structure prepared for building indicators and visualizations.

As future work, the architecture could be extended by incorporating new sources of information, such as marketing campaigns, web behaviour, customer service data or external provider data. Thanks to the Data Vault approach, these extensions could be carried out progressively by adding new entities, relationships or satellites without having to rebuild the existing structure.

Another possible line of evolution would be to adapt the solution to a larger-scale environment, using services such as Azure Synapse Analytics or Microsoft Fabric, as well as incorporating more advanced quality controls, automatic monitoring and predictive analytical models.

6. References

[1] Microsoft. “Implementing Data Vault 2.0 on Fabric Data Warehouse”. Microsoft Community. Available at: https://techcommunity.microsoft.com/blog/analyticsonazure/implementing-data-vault-2-0-on-fabric-data-warehouse/4227078

[2] Microsoft. “Data Vault 2.0 on Azure”. Microsoft Community. Available at: https://techcommunity.microsoft.com/blog/analyticsonazure/data-vault-2-0-on-azure/3860665

[3] Linstedt, D.; Olschimke, M. Building a Scalable Data Warehouse with Data Vault 2.0. Morgan Kaufmann, 2015.

[4] Linstedt, D. Super Charge Your Data Warehouse: Invaluable Data Modeling Rules to Implement Your Data Vault. CreateSpace Independent Publishing Platform, 2011.

Índice de la memoria

Índice de figuras

Capítulo 1. INTRODUCCIÓN

En este capítulo se presenta el contexto previo del proyecto y la motivación que ha llevado a su desarrollo. Por otro lado, se expone la problemática concreta que se busca resolver, la cual está centrada en la integración, transformación, historificación y explotación de datos comerciales. Finalmente, se delimita el alcance del trabajo, detallando tanto las principales fases abordadas como aquellos elementos que quedan fuera del proyecto.

1.1 CONTEXTUALIZACIÓN DEL PROYECTO

El presente Trabajo Fin de Máster se enmarca en el ámbito de la ingeniería de datos, la analítica empresarial y el diseño de arquitecturas orientadas a la integración y explotación de información. En el contexto del Máster, una parte significativa de los trabajos desarrollados por los alumnos se encuentra vinculada a proyectos reales realizados en empresas. Sin embargo, en determinados entornos profesionales, especialmente cuando se trabaja en consultoría, existen restricciones de confidencialidad que impiden exponer información, resultados o desarrollos asociados a clientes concretos.

A partir de esta situación, se decidió plantear un proyecto propio que permitiese aplicar metodologías y enfoques similares a los empleados en un entorno profesional, pero desarrollado íntegramente desde cero y sin utilizar datos reales de clientes. De este modo, el trabajo permite reproducir un escenario realista de ingeniería de datos, manteniendo al mismo tiempo la independencia del proyecto y evitando cualquier conflicto relacionado con la confidencialidad de la información.

Para ello, en las primeras fases del proyecto fue necesario generar un conjunto de datos sintéticos que representasen un entorno de negocio suficientemente completo como para justificar el diseño e implementación de una arquitectura de datos analítica. Se seleccionó el ámbito del comercio electrónico debido a la variedad de entidades que intervienen en este tipo de sistemas, tales como clientes, pedidos, líneas de pedido, productos, categorías y envíos. Además, en un entorno de ventas online los datos pueden proceder de distintas fuentes y presentarse en diferentes formatos, como ficheros CSV, estructuras JSON o servicios API, lo que permite simular una problemática habitual en proyectos reales de integración de datos.

Dentro de este contexto, el trabajo permite abordar diferentes roles habituales en proyectos de datos. Desde una perspectiva de arquitectura, se diseña una solución estructurada por capas y orientada a la escalabilidad, trazabilidad y mantenimiento. Desde el punto de vista

de la ingeniería de datos, se construyen procesos de carga y transformación que permiten integrar información procedente de distintas fuentes. Por último, desde una perspectiva analítica, se preparan modelos y cuadros de mando que facilitan el análisis de la información y la obtención de conclusiones de negocio.

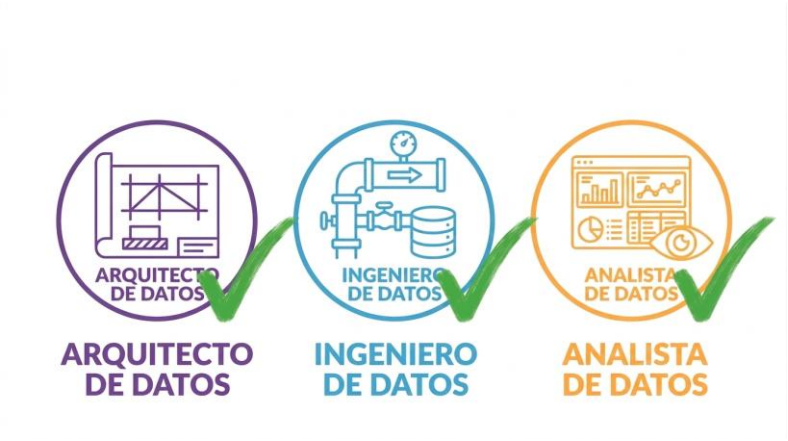


Figura 1. Roles principales cubiertos en el proyecto

De esta forma, el proyecto no se limita únicamente a la creación de visualizaciones, sino que cubre el flujo completo del dato, desde su generación y carga inicial hasta su transformación, modelado y posterior análisis. Esto resulta especialmente importante porque, antes de llegar a una herramienta de Business Intelligence, los datos deben pasar por distintas fases que garanticen su calidad, su coherencia y su trazabilidad. Por ello, el valor del proyecto no se encuentra únicamente en el dashboard final, sino en la arquitectura que permite que ese análisis sea fiable.

1.2 MOTIVACIÓN Y RELEVANCIA

La motivación de este proyecto parte de la importancia que han adquirido los datos en el contexto actual. Las empresas generan cada vez más información en sus procesos diarios, ya sea a través de ventas, pedidos, clientes, productos, operaciones logísticas o interacciones digitales. Esta información se ha convertido en un activo de gran valor, ya que permite entender mejor el funcionamiento del negocio, detectar patrones, medir resultados y apoyar la toma de decisiones.

Sin embargo, disponer de datos no implica necesariamente disponer de información útil. Para que los datos puedan aportar valor, es necesario aplicar un proceso previo de ingesta, tratamiento, transformación y modelado. Los datos en bruto, por sí solos, suelen estar dispersos, tener diferentes formatos y no estar preparados para responder directamente a preguntas de negocio. Por este motivo, uno de los principales intereses del proyecto es

cubrir el flujo completo del dato, desde su recopilación inicial hasta su posterior explotación mediante visualizaciones.

En este sentido, el trabajo no se centra únicamente en la creación de dashboards, sino en todo el proceso necesario para que esas visualizaciones sean fiables. Primero se parte de datos iniciales, que en su estado original no aportan un valor directo al negocio. Después, estos datos se cargan, se limpian, se transforman y se modelan para construir una base analítica más ordenada y coherente. Finalmente, la información resultante se representa mediante cuadros de mando que permiten consultar los datos de una forma más resumida, visual y comprensible.

Otra de las motivaciones del proyecto es la necesidad de centralizar información procedente de distintos sistemas. A medida que una empresa crece, es habitual que sus datos se encuentren repartidos entre diferentes aplicaciones, departamentos o fuentes de información. Esta descentralización puede dificultar el análisis global del negocio, ya que cada sistema puede tener su propia estructura, formato o lógica. Por ello, en este proyecto se plantea la construcción de un Data Warehouse que permita integrar la información en un entorno común y ofrecer una visión más unificada y preparada para el análisis.

En concreto, este Data Warehouse se construye siguiendo un enfoque Data Vault, ya que esta metodología encaja bien con escenarios en los que se integran datos de distintas fuentes y donde el modelo debe estar preparado para crecer con el tiempo. En un contexto empresarial, pueden incorporarse nuevas fuentes, aparecer nuevas entidades de negocio o cambiar las necesidades de análisis. Por este motivo, resulta importante contar con una arquitectura flexible, que permita ampliar el sistema de forma controlada sin tener que rediseñar por completo el modelo existente.

Además, Data Vault encaja especialmente bien con la necesidad de mantener trazabilidad e histórico de la información. Su estructura permite conocer de dónde procede cada dato, cómo se ha integrado dentro del modelo y cómo ha ido evolucionando. Esto resulta especialmente útil en entidades que cambian con el tiempo, como los envíos, donde no solo interesa conocer el estado actual, sino también conservar los estados anteriores. Por ello, el proyecto no se limita a guardar una foto fija de la información, sino que busca construir una base de datos analítica capaz de recoger la evolución del negocio.

1.3 PLANTEAMIENTO DEL PROBLEMA

1.3.1 INTEGRACIÓN DE DATOS HETEROGÉNEOS PROCEDENTES DE MÚLTIPLES FUENTES

Uno de los principales problemas que aborda el proyecto es la integración de datos procedentes de distintas fuentes y con formatos diferentes. En un entorno empresarial, la

información necesaria para analizar el negocio no suele encontrarse concentrada en un único sistema, sino distribuida entre varias aplicaciones internas y proveedores externos.

En el caso planteado, los datos de clientes se obtienen mediante llamadas a la API del CRM, mientras que la información relativa a pedidos y líneas de pedido se ingesta a través de la API del ERP. De esta forma, parte de la información de negocio se obtiene directamente desde sistemas corporativos mediante integraciones basadas en servicios API.

Sin embargo, no todos los proveedores externos disponen de APIs que permitan una integración directa. En estos casos, la información se deposita en un servidor SFTP en forma de ficheros, que posteriormente son transferidos al entorno cloud para su almacenamiento y procesamiento dentro del lakehouse. Esta situación introduce una mayor variedad en los mecanismos de ingesta y obliga a tratar datos con estructuras y formatos distintos.

Dentro de estos ficheros, la información de productos y categorías se recibe en formato CSV, mientras que los datos correspondientes a los envíos son proporcionados por el proveedor logístico en formato JSON. Además, cada fuente puede utilizar sus propias claves internas, nomenclaturas, tipos de datos y formatos de campos, como ocurre habitualmente con las fechas o los identificadores. Por este motivo, antes de cargar la información en el Data Warehouse, es necesario transformar, normalizar y unificar los datos para que puedan integrarse de forma coherente en un modelo común.

1.3.2 MODELADO RELACIONAL Y PREPARACIÓN ANALÍTICA DE LOS DATOS

Una vez recopilada la información procedente de las distintas fuentes, el siguiente problema consiste en organizarla de forma adecuada para que pueda ser explotada analíticamente. Los datos en bruto no son suficientes para construir informes fiables, ya que antes deben estructurarse, relacionarse y adaptarse a un modelo que permita representar correctamente las entidades de negocio y las relaciones que existen entre ellas.

En este tipo de proyectos, el Data Warehouse se trabaja normalmente sobre una base relacional porque la información de negocio está formada por entidades conectadas entre sí. En el caso de un entorno de e-commerce, un cliente puede realizar varios pedidos, cada pedido puede estar formado por distintas líneas, cada línea hace referencia a un producto, los productos pertenecen a categorías y los pedidos pueden tener envíos asociados. Por tanto, es necesario modelar estas relaciones para que los datos puedan consultarse de forma coherente y para evitar análisis aislados o inconsistentes.

En este proyecto, el modelado del Data Warehouse se realiza siguiendo el enfoque Data Vault. Aunque en ocasiones se habla de arquitectura Data Vault, en este trabajo se entiende principalmente como una metodología de modelado orientada a la integración, trazabilidad e historificación de los datos. Esta forma de modelar mantiene una estructura relacional, pero organiza la información de una manera particular mediante hubs, links y satélites. Los

hubs representan las entidades principales del negocio, los links permiten reflejar las relaciones entre esas entidades y los satélites almacenan los atributos descriptivos y su evolución en el tiempo.

Además del modelo Data Vault, también resulta necesario preparar una capa orientada al consumo analítico. Para ello, se diseñan data marts que simplifican la consulta de la información y la adaptan a las necesidades de análisis del proyecto. En esta capa final, el modelo se aproxima a un esquema en estrella, formado por tablas de hechos y dimensiones. Esta estructura permite que la información pueda ser utilizada posteriormente en Power BI, donde se desarrollan métricas, filtros y visualizaciones relacionadas con ventas, pedidos, clientes, productos y envíos.

1.3.3 HISTORIFICACIÓN DE CAMBIOS Y TRAZABILIDAD DEL DATO

Otro de los problemas relevantes en este tipo de proyectos es la necesidad de conservar la evolución de los datos a lo largo del tiempo. En muchos sistemas operacionales, cuando un registro cambia, se actualiza su valor anterior y únicamente queda almacenado el estado más reciente. Esto puede ser suficiente para ciertos procesos del día a día, pero supone una limitación importante desde el punto de vista analítico, ya que se pierde la posibilidad de conocer cómo ha evolucionado la información.

Este problema se puede observar claramente en entidades como los envíos. Un envío no tiene siempre el mismo estado, sino que puede pasar por distintas fases, como creado, enviado, en reparto o entregado. Si el sistema solo almacenase el último estado, no sería posible analizar la evolución del envío, conocer cuándo se produjo cada cambio o detectar posibles incidencias en el proceso logístico. Por este motivo, el proyecto necesita un modelo que no solo guarde la situación actual de los datos, sino también su histórico.

Para resolver esta necesidad, Data Vault trabaja con una filosofía de historificación cercana a las Slowly Changing Dimensions de tipo 2. Este tipo de gestión del cambio consiste en no sobrescribir directamente la información anterior, sino en conservar una versión histórica del registro y crear una nueva versión cuando se detecta una modificación. En el caso de Data Vault, esta historificación se materializa principalmente en los satélites, donde se almacenan los atributos descriptivos de las entidades y sus cambios a lo largo del tiempo.

De esta forma, el modelo se basa en una estrategia principalmente insert-only: los registros no se eliminan ni se actualizan de forma destructiva, sino que se añaden nuevas versiones cuando cambia la información. Para identificar qué versión está vigente y durante qué periodo fue válida cada una, se utilizan campos como la fecha de inicio de validez, la fecha de fin de validez y un indicador de última versión. Este planteamiento permite reconstruir el estado de la información en un momento concreto del tiempo y analizar la evolución de los datos de forma más completa.

Además de la historificación, otro aspecto importante es la trazabilidad. Aunque dentro del Data Warehouse se generen nuevos identificadores propios para organizar y relacionar la información, en el proyecto se conservan también los identificadores procedentes de los sistemas de origen. Mantener estos IDs originales permite relacionar los datos analíticos finales con la información inicial de la que proceden. De esta manera, si aparece una inconsistencia en un data mart o en una visualización, es posible seguir el rastro del dato hasta su origen y comprobar de qué sistema, fichero o entidad procede.

1.4 ALCANCE DEL TRABAJO

Para delimitar el trabajo realizado, se enumeran a continuación las principales fases incluidas dentro del proyecto:

- Generación y preparación de datos sintéticos de un entorno e-commerce mediante scripts de Python, utilizando diferentes librerías estadísticas y de generación de datos.
- Ingesta de los datos desde diferentes fuentes y servicios, y almacenamiento inicial de la información en una capa raw, conservando los datos en un estado cercano al original y sin aplicar transformaciones previas relevantes.
- Transformación y normalización de los datos mediante tareas de limpieza, tipado de campos y homogeneización de formatos, con el objetivo de preparar la información para su posterior carga en el Data Warehouse.
- Modelado del Data Warehouse mediante Data Vault. Se diseña un modelo basado en hubs, links y satélites, siguiendo un enfoque diferente al de los modelos dimensionales tradicionales y orientado a la integración, trazabilidad e historificación de los datos.
- Implementación de procesos de carga en el Data Warehouse, automatización de los procesos y separación de la información en distintas capas, como raw, business y data marts. Este planteamiento guarda cierta relación conceptual con la arquitectura medallion, donde los datos evolucionan desde una capa inicial en bruto hasta capas más refinadas y preparadas para el consumo analítico.
- Construcción de un modelo en estrella a partir de los data marts, relacionando tablas de hechos y dimensiones para crear una capa final orientada al análisis.
- Desarrollo de dashboards en Power BI, creando visualizaciones y mostrando KPIs que permiten analizar la información y apoyar la toma de decisiones de negocio basada en datos.

También resulta importante definir los aspectos que quedan fuera del alcance del proyecto. En primer lugar, no se trabaja con datos reales de empresa debido a motivos de confidencialidad, sino con datos sintéticos generados específicamente para reproducir un escenario de negocio representativo. Del mismo modo, no se implementan sistemas reales de terceros como un CRM, un ERP o una plataforma logística productiva, sino que se simula la información que podría proceder de este tipo de sistemas.

Asimismo, el proyecto no consiste en el desarrollo de un sistema transaccional. Un sistema transaccional es aquel orientado a registrar y gestionar operaciones del día a día, como la creación de pedidos, la actualización de clientes, la gestión de pagos o el seguimiento operativo de envíos. En este caso, el objetivo no es construir una aplicación que gestione esas operaciones en tiempo real, sino una arquitectura analítica que permita integrar, modelar e interpretar los datos generados por ese tipo de procesos.

Por último, tampoco se incluyen técnicas avanzadas de Machine Learning o Data Science para realizar predicciones a partir de los datos. Aunque los datos modelados podrían servir como base para futuros modelos predictivos, el trabajo se centra en los roles de arquitectura, ingeniería y análisis de datos. Por tanto, el objetivo principal es construir una solución end-to-end que permita pasar desde datos en bruto hasta información estructurada y visualizaciones analíticas, sin entrar en el desarrollo de modelos estadísticos o predictivos avanzados.

Capítulo 2. DESCRIPCIÓN DE LAS TECNOLOGÍAS

En este capítulo se describen las principales tecnologías utilizadas para la realización del proyecto. El apartado se centra especialmente en los servicios cloud empleados para la ingesta, almacenamiento, transformación y explotación de los datos, aunque también se incluyen otras herramientas necesarias para el desarrollo del trabajo, como Python, SQL y Power BI.

El objetivo de este capítulo no es explicar todavía el diseño completo de la solución, sino presentar las tecnologías que sirven de base para su implementación. Por este motivo, se describen de forma general las herramientas utilizadas y el papel que desempeñan dentro del proyecto, dejando el detalle de la arquitectura desarrollada para capítulos posteriores.

Cabe señalar que en este capítulo no se profundiza en el modelado Data Vault. Aunque Data Vault constituye una parte central del proyecto, su explicación detallada se aborda en el Capítulo 5. , dedicado al sistema y modelo desarrollado. De esta forma, se evita repetir información y se reserva la descripción de hubs, links, satélites, cargas e historificación para el apartado en el que se presenta la solución implementada.

2.1 LIBRERÍAS PARA LA GENERACIÓN DE DATOS SINTÉTICOS

Para la generación y preparación de los datos sintéticos utilizados en el proyecto se ha empleado Python, debido a su flexibilidad y a la gran cantidad de librerías disponibles para el tratamiento, generación y exportación de datos. Tal y como se ha comentado antes, en este proyecto no se ha podido trabajar con datos reales de la empresa por motivos de confidencialidad, por lo que se tuvo que construir un conjunto de datos ficticio, pero coherente con un entorno e-commerce.

La librería principal utilizada ha sido pandas, empleada para trabajar con datos en formato tabular. Esta librería permite leer, transformar, combinar y exportar conjuntos de datos de forma sencilla mediante estructuras llamadas DataFrames. En el proyecto se ha utilizado para leer y generar ficheros CSV, modificar columnas, eliminar campos no necesarios, combinar tablas mediante claves de negocio y realizar validaciones sobre los datos generados.

Para aportar más realismo a los datos generados, también se ha utilizado la librería numpy. En lugar de asignar todos los valores de forma completamente uniforme, esta librería permite trabajar con probabilidades y distribuciones, haciendo que los datos se parezcan más a los que podrían aparecer en un caso real. Por ejemplo, no todos los métodos de pago tienen la misma frecuencia, no todos los clientes realizan el mismo número de pedidos y no todas las categorías tienen el mismo peso dentro del catálogo. En el proyecto, numpy se ha utilizado para reflejar este tipo de comportamientos en la generación de clientes, pedidos, productos y líneas de pedido.

Junto con numpy, se ha utilizado también la librería random, propia de Python, para introducir variabilidad en decisiones más sencillas del proceso de generación. Esta librería se ha empleado, por ejemplo, para seleccionar ciudades, marcas, nombres de productos, fechas dentro de un rango o estados concretos. Además, en los scripts se fija una semilla de generación, lo que permite obtener los mismos resultados si el proceso se ejecuta de nuevo. Esto resulta útil para poder repetir pruebas y mantener una base de datos sintética estable durante el desarrollo del proyecto.

Otra librería importante dentro del proyecto ha sido Faker. Esta herramienta se utiliza para generar datos ficticios con una apariencia similar a la de datos reales, lo que resulta especialmente útil cuando no se pueden utilizar datos de clientes o empresas por motivos de confidencialidad. En este proyecto se ha empleado principalmente para crear información asociada a clientes y envíos, como nombres, apellidos, direcciones y ciudades.

```
cliente_id,nombre,apellidos,email,ciudad,pais,codigo_postal,fecha_registro,fidelizado,fecha_nacimiento,fecha_actualizacion
CUST_00001,Feliciana,Cantón Amaya,feliciana.canton@gmail.com,Barcelona,España,8859,2025-07-14,True,1956-12-04,2026-04-05
CUST_00002,Manuela,Llopis Hierro,manuela.llopis@outlook.com,Bilbao,España,48704,2025-04-06,True,2005-03-11,2026-04-05
CUST_00003,Jacinto,Chaparro Calatayud,jacinto.chaparro@email.com,Pamplona,España,31127,2024-02-17,False,1972-12-24,2026-04-05
CUST_00004,Inmaculada,Bonet Roca,inmaculada.bonet@hotmail.com,Sevilla,España,41703,2025-06-11,False,1997-03-26,2026-04-05
CUST_00005,Amílcar,Andrés Bueno,amilcar.andres@hotmail.com,Zaragoza,España,50320,2025-05-08,False,1996-02-06,2026-04-05
CUST_00006,Isaías,Guerrero Vazquez,isaias.guerrero@email.com,Málaga,España,29370,2025-12-21,True,1990-08-08,2026-04-05
CUST_00007,Eligia,Amor Flor,eligia.amor@gmail.com,Barcelona,España,8400,2026-01-11,True,1981-09-29,2026-04-05
CUST_00008,Lilia,Rivas Giménez,lilia.rivas@gmail.com,Madrid,España,28333,2025-05-01,False,2001-10-31,2026-04-05
```

Figura 2. Muestra de datos de clientes generados con la librería Faker

Finalmente, también se han utilizado algunos módulos auxiliares de Python para tareas más concretas de tratamiento y formateo de datos, como datetime, json o math. datetime ha servido para trabajar con fechas y calcular intervalos temporales, json para generar y manipular ficheros en formato JSON, y math para realizar algunos cálculos puntuales. No son librerías tan centrales como pandas, numpy, random o Faker, pero han sido útiles para completar ciertas partes de los scripts y preparar correctamente los ficheros de datos.

2.2 PLATAFORMA CLOUD Y SERVICIOS UTILIZADOS

Para el despliegue de la parte cloud del proyecto se ha utilizado Microsoft Azure. La elección de esta plataforma se debe principalmente a la disponibilidad de una suscripción de estudiante, que proporciona un crédito inicial de 100 euros, y a la familiaridad con el ecosistema de Microsoft. Además, Azure encaja bien con algunas de las herramientas utilizadas en el proyecto, como Azure SQL Database, Azure Data Factory y Power BI.

El uso de esta suscripción ha permitido desplegar los servicios necesarios sin incurrir en un coste económico elevado. Algunos recursos de Azure ofrecen niveles gratuitos o consumos reducidos dentro de ciertos límites, por lo que se ha podido desarrollar la solución ajustando la infraestructura al alcance académico del trabajo.

Para organizar los servicios utilizados, se ha creado un grupo de recursos en Azure. Dentro de este grupo se agrupan los componentes principales del proyecto, entre los que se encuentran la cuenta de almacenamiento para los ficheros de entrada, Azure Data Factory para la ingesta y orquestación de procesos, Azure SQL Database para almacenar las capas del Data Warehouse y Azure Functions para la parte relacionada con la simulación de servicios API.

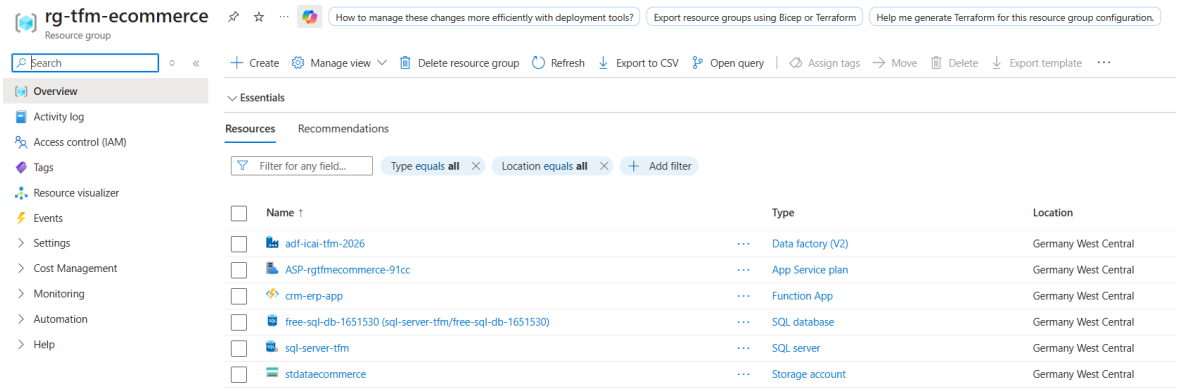


Figura 3. Grupo de Recursos y servicios utilizados

En los siguientes subapartados se explica el papel que tiene cada uno de estos servicios dentro del proyecto. Además, se incluye una breve aclaración sobre posibles alternativas en entornos profesionales, ya que en proyectos reales podrían emplearse otros servicios más avanzados o integrados, como Azure Synapse Analytics o Microsoft Fabric.

2.2.1 AZURE FUNCTIONS PARA LA EXPOSICIÓN DE APIS

Azure Functions es un servicio serverless de Microsoft Azure que permite ejecutar pequeñas piezas de código en la nube sin necesidad de administrar directamente servidores. Este tipo

de servicio es muy útil cuando se necesitan procesos ligeros, independientes y activados bajo demanda, por ejemplo, mediante una petición HTTP.

En este proyecto, Azure Functions se ha utilizado para crear una serie de endpoints HTTP que simulan la exposición de datos procedentes de sistemas externos. En concreto, se ha creado un endpoint para clientes, que representa la información que podría obtenerse desde un CRM, y otros dos endpoints para pedidos y líneas de pedido, simulando información habitual de un ERP. Con esto se consigue que una parte de los datos del proyecto no llegue directamente en forma de fichero, sino mediante peticiones HTTP a servicios desplegados en la nube.

En conclusión, este servicio se ha utilizado como mecanismo para representar un escenario lo mas parecido a un sistema empresarial real en el que distintos servicios y componentes de sistemas informacionales se comunican via APIs.

Los endpoints expuestos son los siguientes:

Endpoint	Descripción
/api/clientes	Devuelve la información de clientes simulando datos procedentes de un CRM. (Full load)
/api/pedidos	Devuelve la información de pedidos simulando datos procedentes de un ERP. (Full load)
/api/lineas-pedido	Devuelve la información de las líneas de pedido asociadas a los pedidos del ERP. (Full load)
/api/clientes?fecha_actualizacion=2026-04-01	Devuelve clientes filtrados por fecha de actualización.
/api/pedidos?desde=2026-04-01&hasta=2026-04-07	Devuelve pedidos filtrados por un rango de fechas.
/api/líneas-pedido?desde=2026-04-01&hasta=2026-04-07	Devuelve líneas de pedido filtradas por un rango de fechas.

Figura 4. Listado de endpoints definidos como fuente de datos

2.2.2 AZURE DATA LAKE STORAGE GEN2

La cuenta de almacenamiento de Azure se ha utilizado como zona de almacenamiento inicial para los ficheros del proyecto. En este caso, se ha configurado para actuar como un Data Lake, habilitando la opción de espacio de nombres jerárquico. Esta característica

permite organizar los datos mediante una estructura de carpetas y subcarpetas, de forma similar a un sistema de ficheros tradicional, lo que facilita la separación de los datos por fuente o entidad.

Dentro de la cuenta de almacenamiento se han creado dos contenedores para separar los ficheros según su origen y tipo de información. Un contenedor es como una agrupación lógica dentro de la cuenta de almacenamiento, similar a una carpeta principal. Esta separación permite organizar mejor los datos y facilita su posterior consumo desde los pipelines de ingesta.

El primer contenedor, denominado product-categories-data, se utiliza para almacenar los ficheros CSV correspondientes a productos y categorías. Estos ficheros contienen información con estructura tabular, por lo que el formato CSV resulta adecuado para su almacenamiento e ingesta. El segundo contenedor, llamado shipping-data, recibe los ficheros JSON con la información de envíos proporcionada por el proveedor logístico. En este caso, el formato JSON permite representar datos semiestructurados asociados a la evolución y características de los envíos.

2.2.3 AZURE SQL DATABASE Y SQL SERVER COMO DATA WAREHOUSE

Azure SQL Database se ha utilizado como servicio de base de datos relacional en la nube para implementar el Data Warehouse del proyecto. Para poder crear esta base de datos en Azure, primero es necesario disponer de un SQL Server. Este servidor es un recurso que puede agrupar o gestionar una o varias bases de datos. Es muy importante la correcta creación previa de este SQL Server ya que en él se definen aspectos como el nombre del servidor, la región, la autenticación, las reglas de firewall y la configuración de acceso.

En este proyecto se ha utilizado la modalidad gratuita disponible para Azure SQL Database, que permite trabajar dentro de ciertos límites sin incurrir en costes. Esta opción ofrece hasta 32 GB de almacenamiento de datos, 32 GB de almacenamiento para copias de seguridad y 100.000 segundos vCore de cómputo serverless al mes. Esto son límites suficientemente grandes para poder trabajar con los volúmenes de datos correspondientes a este TFM.

En cuanto a la base de datos Azure SQL Database, aquí es donde se ha implantado el modelo Data Vault del proyecto, almacenando en primer lugar los datos en crudo y vistas de staging y posteriormente los datos de las capas raw, business y data marts.

Además, en esta misma base de datos se definen los procedimientos almacenados encargados de ejecutar parte de la lógica de transformación y carga. Estos procedimientos permiten automatizar tareas como la carga de datos desde las vistas stage hacia las estructuras Data Vault, la inserción de nuevos registros, la gestión de cambios en los satélites y la preparación de las tablas finales de los data marts. Posteriormente, estos procedimientos son invocados desde los pipelines de Azure Data Factory, lo que permite coordinar las cargas de forma ordenada y reproducible.

La integración entre Azure SQL Database y Azure Data Factory resulta especialmente útil en este proyecto, ya que existe compatibilidad directa entre ambos servicios. Desde Data Factory es posible conectarse a la base de datos, cargar información en tablas, ejecutar consultas y lanzar procedimientos almacenados como parte de un pipeline. De esta forma, la base de datos no solo actúa como repositorio de información, sino también como el lugar donde se concentra una parte importante de la lógica de transformación y modelado del Data Warehouse.

2.2.4 AZURE DATA FACTORY PARA LA EJECUCIÓN DE PIPELINES

Azure Data Factory se ha utilizado como servicio principal de orquestación e integración de datos dentro del proyecto. Su función es coordinar la ejecución de los distintos procesos que forman parte del flujo de datos, desde la ingesta inicial hasta la carga de las capas posteriores del Data Warehouse. A través de este servicio se definen pipelines que permiten automatizar tareas llamadas "Actividades" dentro del servicio y establecer un orden de ejecución entre ellas.

En el proyecto, Data Factory se encarga de conectar con las distintas fuentes de información utilizadas. Por un lado, permite consumir los endpoints creados con Azure Functions, desde los que se obtienen los datos de clientes, pedidos y líneas de pedido. Por otro lado, también permite acceder a los ficheros almacenados en la cuenta de almacenamiento, donde se encuentran los CSV de productos y categorías y los JSON de envíos. De esta forma, Data Factory actúa como punto central de integración entre las fuentes de origen y el Data Warehouse.

Además de la ingesta, Azure Data Factory permite automatizar la ejecución de los procedimientos almacenados responsables de las cargas del Data Warehouse. En el proyecto, estos procedimientos se utilizan para cargar las estructuras Data Vault, gestionar la historificación en los satélites y preparar los data marts finales. Gracias a esta integración, las distintas fases del proceso pueden ejecutarse de forma controlada, sin necesidad de lanzar manualmente cada carga desde la base de datos.

Otra ventaja de Data Factory es que permite representar visualmente el flujo de trabajo mediante pipelines. Esto facilita entender qué procesos se ejecutan, en qué orden y con qué dependencias. En una arquitectura por capas, esta capacidad resulta especialmente útil, ya que la carga de una capa depende de que la anterior se haya completado correctamente. Por ejemplo, antes de ejecutar los procedimientos de Data Vault, es necesario que los datos estén disponibles en raw y preparados mediante las vistas stage.

Otra utilidad fundamental de Azure Data Factory es la posibilidad de monitorizar las ejecuciones de los pipelines. Desde el propio servicio se puede comprobar si una carga se ha ejecutado correctamente, cuánto tiempo ha tardado y en qué punto se encuentra cada actividad. En caso de error, la herramienta permite identificar en qué paso ha fallado el proceso, revisar el mensaje asociado, corregir el problema y volver a lanzar la ejecución.

Esto facilita mucho el control de las cargas y evita tener que revisar manualmente cada una de las fases del flujo de datos.

A continuación, se muestra una imagen de un ejemplo de uno de los pipelines desarrollados, estos pipelines serán explicados más en detalle en el Capítulo 5.

Figura 5. Ejemplo pipeline de carga de datos en Data Vault

2.3 POWER BI PARA EXPLOTACIÓN ANALÍTICA

Power BI se ha utilizado como herramienta de Business Intelligence para cerrar el flujo end-to-end del proyecto. Una vez que los datos han sido ingeridos, transformados, historificados y preparados en los data marts, es necesario disponer de una capa final que permita analizar la información de forma visual y comprensible. En esta fase, los datos ya no se encuentran en bruto, sino que han pasado por las distintas capas de la arquitectura, se han aplicado las reglas de negocio necesarias y se han organizado en un modelo orientado al análisis.

La elección de Power BI se debe principalmente a su facilidad de integración con el ecosistema Microsoft. Al trabajar con servicios como Azure SQL Database y Azure Data Factory, Power BI permite conectarse de forma sencilla a las tablas finales del modelo y construir informes sobre la información preparada en los data marts. Además, su uso permite crear medidas, filtros, gráficos e indicadores que facilitan la interpretación de los datos por parte del usuario final.

Con esta herramienta se han desarrollado distintos cuadros de mando orientados al análisis del negocio. En concreto, se han creado dashboards centrados en clientes, ventas, productos y categorías de producto. Estos informes permiten visualizar indicadores relevantes, analizar tendencias, comparar comportamientos y detectar patrones que podrían ayudar en la toma de decisiones.

Capítulo 3. ESTADO DE LA CUESTIÓN

En este capítulo se revisan distintos enfoques y soluciones existentes para la construcción de arquitecturas analíticas de datos. El objetivo es situar el proyecto dentro del contexto actual y analizar cómo se suelen abordar problemas como el almacenamiento de información, el modelado para análisis, la integración de fuentes heterogéneas y la explotación visual de los datos. También se tienen en cuenta aspectos como el rendimiento de las consultas y la eficiencia en el acceso a la información, ya que no todos los modelos responden igual ante grandes volúmenes de datos o ante necesidades de análisis frecuentes.

Esta revisión permite comparar alternativas como los Data Warehouse tradicionales, los modelos dimensionales, los Data Lake, las arquitecturas Lakehouse, la integración con sistemas terceros y las herramientas de Business Intelligence, sirviendo como base para justificar posteriormente las decisiones adoptadas en el proyecto.

3.1 ENFOQUES TRADICIONALES DE DATA WAREHOUSE

Los Data Warehouse tradicionales surgen con el objetivo de centralizar información procedente de distintos sistemas y prepararla para su análisis. En muchas organizaciones, los datos operacionales se encuentran repartidos en aplicaciones diferentes, como sistemas de ventas, clientes, logística o facturación. El Data Warehouse actúa como una base de datos central donde esa información se integra, se limpia y se organiza para facilitar consultas, informes y toma de decisiones.

De esta manera, el Data Warehouse permite centralizar la información de la empresa en un único repositorio robusto, reduciendo la aparición de silos de información entre departamentos o sistemas. Al consolidar los datos en una misma plataforma, se facilita que las áreas de negocio trabajen con una versión común de la información, más controlada, fiable y almacenada bajo criterios de seguridad y gobierno del dato.

Normalmente, este tipo de almacenes se alimentan mediante procesos de carga planificados, ejecutados de forma periódica en modo batch. Estas cargas suelen programarse en horarios nocturnos o en momentos de baja actividad, con el objetivo de no afectar al rendimiento de los sistemas operacionales de origen. De esta forma, se evita sobrecargar aplicaciones

críticas del negocio y se aprovechan mejor los recursos disponibles para mover, transformar y consolidar la información dentro del Data Warehouse.

Aunque en algunos escenarios actuales también se trabaja con cargas más frecuentes o cercanas al tiempo real, el enfoque batch sigue siendo muy habitual en arquitecturas de Data Warehouse tradicionales. Esto se debe a que muchos informes analíticos no requieren disponer del dato al segundo, sino contar con información consistente, validada y preparada para su análisis en determinados momentos del día.

3.1.1 ENFOQUE TOP-DOWN O MODELO COMPARATIVO DE INMON

El modelo propuesto por Inmon fue uno de los primeros enfoques en defender la necesidad de disponer de una base de datos analítica corporativa, separada de los sistemas operacionales e integrada de forma centralizada. Bajo este planteamiento, el Data Warehouse reúne información procedente de distintas fuentes para ofrecer una visión común y fiable de la organización.

Una característica habitual de este enfoque es el uso de modelos normalizados, cercanos a la tercera forma normal. Esto permite reducir duplicidades y representar cada entidad de negocio de forma ordenada, relacionando conceptos como clientes, pedidos o productos mediante claves. A partir de este repositorio central pueden construirse data marts orientados a áreas concretas del negocio, concepto que también se utiliza en arquitecturas basadas en Data Vault para facilitar la explotación analítica final.

Sin embargo, este modelo puede resultar algo rígido en entornos tecnológicos actuales, donde las fuentes de datos y las necesidades de negocio cambian con frecuencia. Algunos cambios pueden obligar a rediseñar parte de la estructura analítica y modificar procesos de carga complejos. Además, el uso habitual de claves secuenciales puede generar cuellos de botella en cargas de gran volumen o procesos paralelos, mientras que enfoques como Data Vault suelen apoyarse en claves de negocio y claves hash para facilitar la integración y la carga paralela.

3.1.2 ENFOQUE BOTTOM-UP O MODELO DIMENSIONAL DE KIMBALL

El enfoque propuesto por Kimball plantea que el Data Warehouse debe construirse pensando en procesos de negocio concretos, como ventas, clientes, productos o pedidos. En lugar de diseñar primero una estructura global para toda la empresa, este modelo se centra en crear data marts que sean fáciles de consultar por los usuarios de negocio. Por este motivo, se apoya en el modelado dimensional, formado por tablas de hechos y dimensiones.

En este proyecto también se utilizan hechos y dimensiones, pero en la capa final de data marts, no como núcleo principal del Data Warehouse.

Una limitación de este enfoque es que, para cargar las tablas de hechos y dimensiones, los datos deben limpiarse y transformarse previamente, aplicando ya determinadas reglas de negocio. Si estas reglas cambian con el tiempo, puede ser necesario recalcular parte del modelo analítico.

En este tipo de modelos, cuando cambia un atributo de una dimensión, como la ciudad de residencia de un cliente o la categoría de un producto, en muchas implementaciones se actualiza directamente el registro existente para mantener únicamente la versión más reciente del dato. Esto simplifica el modelo y facilita la consulta, pero también limitar el análisis histórico, ya que no siempre permite reconstruir cómo ha evolucionado una entidad a lo largo del tiempo. Además, al sobrescribirse los valores anteriores, también se reduce la trazabilidad del dato, porque resulta más difícil saber cuándo se produjo un cambio, desde qué fuente llegó o qué valor tenía antes de la actualización.

3.2 STAR SCHEMAS Y SNOWFLAKE SCHEMAS

El modelo dimensional es uno de los enfoques más utilizados para preparar información orientada al análisis de negocio. Se basa en organizar los datos mediante tablas de hechos y dimensiones. Las tablas de hechos contienen los eventos o métricas principales, como ventas, pedidos, importes o cantidades, mientras que las dimensiones aportan el contexto de análisis, como cliente, producto, categoría, fecha o ubicación.

Dentro de este tipo de modelado existen dos estructuras habituales: el star schema y el snowflake schema. En el star schema, la tabla de hechos se sitúa en el centro del modelo y se relaciona directamente con sus dimensiones. Esto reduce el número de relaciones necesarias para consultar la información.

En el snowflake schema, las dimensiones se normalizan y se dividen en varias tablas relacionadas entre sí. Por ejemplo, una dimensión de producto podría separarse en producto, categoría y subcategoría. Este enfoque reduce cierta redundancia y permite representar jerarquías de forma más detallada, pero también aumenta la complejidad del modelo y puede hacer que las consultas sean menos directas.

En este proyecto se ha utilizado un enfoque basado en star schema en los data marts finales porque permite ofrecer a Power BI un modelo más simple, directo y preparado para la consulta. Al tener las dimensiones conectadas directamente con las tablas de hechos, se reducen las relaciones intermedias y se facilita la creación de medidas, filtros e indicadores.

3.3 *DATA LAKE Y ARQUITECTURAS LAKEHOUSE*

Los Data Lake aparecen como una forma de almacenar datos de manera más flexible, especialmente cuando la información llega desde muchas fuentes y no siempre con la misma estructura. A diferencia de un Data Warehouse tradicional, donde normalmente los datos se transforman y se adaptan a un modelo antes de almacenarse, en un Data Lake es posible guardar la información en un estado más parecido al original. Esto resulta útil cuando se trabaja con ficheros CSV, JSON, logs u otros datos semiestructurados, ya que permite conservarlos primero y decidir después cómo tratarlos según las necesidades del análisis.

Una de las principales ventajas de este enfoque es su flexibilidad. Los datos pueden almacenarse primero y transformarse posteriormente según las necesidades del análisis, lo que se conoce habitualmente como un enfoque schema-on-read. Sin embargo, esta flexibilidad también puede convertirse en una limitación si no existe una buena organización, control de calidad y gobierno del dato. Un Data Lake sin estructura clara puede acabar convirtiéndose en un repositorio difícil de mantener, donde no siempre es sencillo saber qué datos son válidos, actualizados o fiables.

Como evolución de este concepto aparecen las arquitecturas Lakehouse, que intentan combinar la flexibilidad de un Data Lake con algunas capacidades propias de un Data Warehouse. Este enfoque busca permitir el almacenamiento de datos en formatos abiertos y escalables, pero incorporando mecanismos que faciliten el análisis, la gestión de esquemas, la calidad del dato y el acceso mediante consultas analíticas. De esta forma, el Lakehouse se plantea como una alternativa intermedia para entornos donde se necesita trabajar tanto con datos estructurados como semiestructurados.

En este proyecto se ha utilizado Azure Data Lake Storage Gen2 (DataLake de Microsfot Azure) como capa de landing para los ficheros recibidos. Los datos de productos, categorías y envíos se almacenan inicialmente en esta capa antes de ser cargados y transformados en la base de datos analítica. De esta forma, se conserva una copia inicial de los ficheros en un estado cercano al original, lo que permite separar la recepción de los datos de su posterior procesamiento. No obstante, la explotación final no se realiza directamente sobre el Data Lake, sino que los datos se integran después en el Data Warehouse mediante Data Vault y se preparan en data marts para su análisis en Power BI.

3.4 *INTEGRACIÓN CON SISTEMAS TERCEROS*

En las arquitecturas de datos empresariales es habitual que la información proceda de sistemas externos o aplicaciones especializadas. Por ejemplo, los datos de clientes pueden encontrarse en un CRM, los pedidos en un ERP, los envíos en una plataforma logística y los productos en sistemas propios o de proveedores. Por este motivo, la integración con sistemas

de terceros es una parte fundamental de cualquier solución analítica, ya que permite reunir información dispersa y convertirla en una visión común para el análisis.

Una de las formas más habituales de integración es el consumo de APIs. En este caso, un sistema expone determinados endpoints para que otras aplicaciones puedan consultar sus datos de forma controlada. Este enfoque es muy utilizado cuando se necesita acceder a información de aplicaciones externas sin tener acceso directo a sus bases de datos. Las APIs permiten definir qué datos se comparten, aplicar filtros, controlar la seguridad y automatizar la extracción desde procesos de ingesta.

Otra forma común de integración es el intercambio de ficheros. En muchos entornos, especialmente cuando intervienen proveedores externos, los datos se entregan en formatos como CSV, JSON, XML o Excel, normalmente a través de un SFTP, una cuenta de almacenamiento cloud o un área compartida. Aunque puede parecer un enfoque más simple que una API, sigue siendo muy utilizado porque resulta fácil de implementar y permite desacoplar al proveedor del sistema que consume la información.

También existen integraciones directas con bases de datos, en las que los procesos de datos se conectan a tablas o vistas de un sistema origen para extraer la información necesaria. Este enfoque puede ser eficiente, pero debe utilizarse con cuidado, ya que una mala planificación puede afectar al rendimiento de los sistemas operacionales. Por este motivo, las cargas suelen ejecutarse en ventanas horarias concretas o sobre réplicas preparadas para consulta.

En arquitecturas más avanzadas también pueden utilizarse colas de mensajes o integración basada en eventos. En este caso, los sistemas publican eventos cuando ocurre algo relevante, como la creación de un pedido, la actualización de un cliente o el cambio de estado de un envío. Esta forma de integración permite trabajar con datos de manera más cercana al tiempo real, aunque suele requerir una arquitectura más compleja y un mayor control sobre la gestión de eventos, errores y reintentos.

Capítulo 4. DEFINICIÓN DEL TRABAJO

4.1 JUSTIFICACIÓN

Una vez revisados los principales enfoques existentes para la construcción de arquitecturas analíticas, resulta necesario justificar las decisiones tomadas en este proyecto. El objetivo de este apartado es explicar por qué se ha optado por una solución basada en Data Vault como núcleo del Data Warehouse, por qué se ha utilizado un modelo dimensional en la capa final de análisis y por qué la integración con sistemas externos se ha planteado mediante APIs y ficheros.

Estas decisiones responden a las necesidades concretas del proyecto: integrar información procedente de distintas fuentes, conservar el histórico de los datos, mantener trazabilidad desde el origen hasta las capas analíticas y preparar un modelo final sencillo para su explotación en Power BI. De esta forma, la arquitectura propuesta no solo busca almacenar datos, sino organizar todo el flujo de información desde la ingesta inicial hasta la visualización final.

4.1.1 JUSTIFICACIÓN DEL USO DE DATA VAULT COMO NÚCLEO DEL DATA WAREHOUSE

Una de las principales razones para utilizar Data Vault en este proyecto es la separación clara que propone entre entidades de negocio, relaciones y atributos descriptivos. Para ello, el modelo se organiza en tres tipos principales de estructuras: hubs, links y satélites. Esta división permite construir un Data Warehouse más flexible, trazable y preparado para incorporar cambios o nuevas fuentes de información.

Los hubs representan las entidades principales del negocio y almacenan sus claves de negocio. En el contexto del proyecto, ejemplos de hubs serían cliente, pedido, producto, categoría o envío. Estas estructuras no contienen todos los atributos descriptivos de la entidad, sino principalmente la clave que permite identificarla de forma única dentro del Data Warehouse, junto con campos técnicos de carga y trazabilidad.

hk_cliente	cliente_id	dv_load_date	dv_source
00047438AC5682E310CD8201F91F5DC6C149...	CUST_04063	2026-05-18T17:35:03.3102246	api_clientes

Figura 6. Ejemplo de un registro de un Hub de cliente en DataVault

Los links se utilizan para representar las relaciones entre hubs. Por ejemplo, un pedido está asociado a un cliente, una línea de pedido relaciona un pedido con un producto, o un envío está vinculado a un pedido. De esta forma, las relaciones entre entidades se modelan de manera independiente, evitando mezclar en una misma tabla la identificación de las entidades, sus atributos y sus relaciones.

hk_cliente_pedido	hk_cliente	hk_pedido	dv_load_date	dv_source
355E3866C0E3C47AB9F56342F80...	00047438AC5682E310CD8201F91...	25D3D63A15192EF80DFED6FCDC...	2026-05-18T17:35:08.4754398	api_pedidos

Figura 7. Ejemplo de un registro de un Link entre cliente y pedido en DataVault

Por último, los satélites almacenan los atributos descriptivos y su evolución en el tiempo. Por ejemplo, un satélite de cliente puede guardar información como nombre, ciudad o correo electrónico, mientras que un satélite de envío puede almacenar el estado del envío y sus cambios a lo largo del tiempo. Esta separación permite conservar distintas versiones de la información sin sobrescribir directamente los valores anteriores.

hk_cliente	nombre	apellidos	email	dv_hash_value	dv_valid_from	dv_valid_to	dv_flag_last_ve...
00047438AC5682E...	Amaro	Luque Sebastián	amaro.luque@hot...	F5F26E5FB6630FAC...	2026-04-11T00:00:...	*NULL*	True

Figura 8. Ejemplo de un registro de un Satélite de cliente en DataVault

Frente a modelos más tradicionales, donde suele existir una tabla principal por entidad con sus claves, relaciones y atributos en una misma estructura, Data Vault separa cada concepto según su función. Esto facilita la incorporación de nuevas fuentes o nuevos atributos sin tener que rediseñar completamente las tablas existentes. Por ejemplo, si en el futuro se añadiese un nuevo sistema que aportase información adicional de clientes, podría incorporarse mediante un nuevo satélite asociado al hub de cliente, manteniendo separada la información de esa fuente y reduciendo el impacto sobre el modelo ya construido.

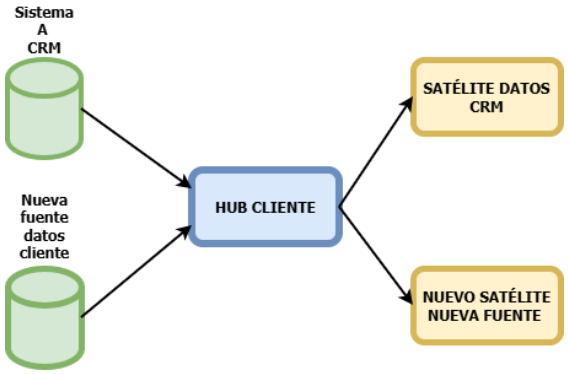


Figura 9. Escalabilidad nuevas fuentes de datos en Data Vault

4.1.2 USO DE CLAVES HASH FRENTE A CLAVES SECUENCIALES

Otra decisión importante dentro del diseño del Data Warehouse ha sido utilizar claves hash en lugar de depender de claves secuenciales generadas por la base de datos. En muchos modelos relacionales es habitual usar identificadores incrementales, es decir, valores numéricos que se van asignando en orden según se insertan los registros. Este enfoque funciona correctamente en muchos casos, pero puede crear dependencias entre cargas, ya que algunas tablas necesitan conocer primero el identificador generado en otras para poder relacionarse con ellas.

En Data Vault, las claves se calculan a partir de las claves de negocio que llegan desde los sistemas origen. Por ejemplo, el identificador de un cliente, de un pedido o de un producto se transforma mediante una función hash y se utiliza como clave técnica dentro del Data Warehouse. Esto hace que la clave no dependa del orden de inserción ni de un contador interno de la base de datos: si vuelve a llegar el mismo cliente o pedido, se volverá a generar la misma clave hash.

Esto facilita mucho la ejecución de las cargas. Al poder calcular las claves antes de insertar los datos, distintos procesos pueden ejecutarse de forma más independiente. Por ejemplo, se puede generar la clave de cliente en el hub de clientes y utilizar esa misma clave en un link relacionado con pedidos sin tener que esperar a que la base de datos devuelva un identificador secuencial. De esta forma, se reducen dependencias entre procesos y se facilita que las cargas puedan lanzarse de manera paralela o en bloques separados.

4.1.3 HISTORIFICACIÓN MEDIANTE INSERCIONES Y CONTROL DE VIGENCIA

Otro motivo importante para utilizar Data Vault en el proyecto es la posibilidad de conservar la evolución de los datos a lo largo del tiempo. En algunos modelos más tradicionales, cuando cambia la información de un registro, se actualiza directamente el valor existente. Esto permite mantener la última versión del dato, pero provoca que los valores anteriores se pierdan si no se ha definido una lógica específica de histórico.

En este proyecto, cuando se detecta un cambio en los atributos de un registro, no se sobrescribe la información anterior, sino que se inserta un nuevo registro en el satélite correspondiente. De esta forma, cada versión queda almacenada y es posible consultar cómo ha ido cambiando la información con el paso del tiempo.

Para controlar esta evolución se utilizan campos técnicos como dv_valid_from, dv_valid_to y dv_flag_last_version. El campo dv_valid_from indica desde qué momento es válida una versión del dato, mientras que dv_valid_to indica hasta cuándo estuvo vigente. Por su parte,

dv_flag_last_version permite identificar de forma sencilla cuál es la versión actual del registro.

Un ejemplo claro de esta necesidad aparece en la entidad de envíos. Un envío no tiene un único estado durante todo su ciclo de vida, sino que puede pasar por estados como creado, en preparación, enviado, en reparto o entregado. Si cada cambio se gestionase mediante actualizaciones directas, solo se conservaría el último estado del envío. En cambio, mediante la historificación en satélites, se puede mantener el recorrido completo del envío y analizar cuánto tiempo ha permanecido en cada estado o cuándo se produjo cada cambio.

hk_envio	envio_id	transportista	direccion_envio	estado	dv_valid_from	dv_valid_to	dv_flag_last_ve...
52C434BDB67007F...	ENV_0000008	MRW	C. Leopoldo Santan...	Preparación	2026-04-04T00:00:...	2026-04-05T00:00:...	False
52C434BDB67007F...	ENV_0000008	MRW	C. Leopoldo Santan...	Enviado	2026-04-05T00:00:...	2026-04-06T00:00:...	False
52C434BDB67007F...	ENV_0000008	MRW	C. Leopoldo Santan...	En reparto	2026-04-06T00:00:...	2026-04-07T00:00:...	False
52C434BDB67007F...	ENV_0000008	MRW	C. Leopoldo Santan...	En tránsito	2026-04-07T00:00:...	2026-04-08T00:00:...	False
52C434BDB67007F...	ENV_0000008	MRW	C. Leopoldo Santan...	Creado	2026-04-03T00:00:...	2026-04-04T00:00:...	False
52C434BDB67007F...	ENV_0000008	MRW	C. Leopoldo Santan...	Entregado	2026-04-08T00:00:...	*NULL*	True

Figura 10. Ejemplo histórico entidad envíos

Este enfoque permite evitar la pérdida de información histórica y mejora la capacidad de análisis temporal. Además, resulta especialmente útil en una arquitectura analítica, ya que no solo interesa conocer el valor actual de un dato, sino también poder reconstruir su evolución y entender qué información estaba vigente en un momento determinado.

4.1.4 TRAZABILIDAD Y AUDITORÍA DEL DATO

Otro aspecto importante en la solución desarrollada es la trazabilidad del dato. Al integrar información procedente de distintas fuentes, no basta con almacenar el dato final ya transformado, sino que también es necesario poder conocer su origen y seguir el recorrido que ha tenido dentro de la arquitectura. Esto resulta especialmente útil cuando aparece una incoherencia en un dashboard de BI o cuando se necesita comprobar de dónde procede un determinado valor.

En este proyecto, la trazabilidad se refuerza manteniendo identificadores procedentes de los sistemas origen y campos técnicos asociados a la carga. Por ejemplo, en las estructuras del Data Warehouse se conservan claves de negocio como el identificador del cliente, del pedido, del producto o del envío. Además, se incorporan campos como la fuente del dato, la fecha de carga y los campos de control propios de los satélites, que permiten saber cuándo se incorporó la información y cuál es su estado dentro del modelo.

El uso de claves hash también contribuye a esta trazabilidad, ya que permite generar identificadores técnicos de forma consistente a partir de las claves de negocio. Si una misma entidad vuelve a aparecer en una carga posterior, se puede volver a calcular la misma clave hash y relacionarla con la información ya existente. Esto facilita mantener una conexión clara entre el dato recibido desde el origen y las estructuras internas del Data Warehouse.

Esta capacidad de auditoría permite investigar errores o diferencias entre fuentes de una forma más controlada. Por ejemplo, si en un dashboard aparece un importe incorrecto o un estado de envío inesperado, se puede retroceder desde el data mart hasta las capas anteriores para comprobar qué registro lo generó, de qué fuente procedía y en qué momento fue cargado. De esta forma, la arquitectura no solo permite explotar datos, sino también revisar y justificar la información que llega hasta los informes finales.

4.1.5 SEPARACIÓN POR CAPAS: RAW, BUSINESS Y DATA MARTS

Otra decisión importante del proyecto ha sido separar la arquitectura en distintas capas: raw, business y data marts. Esta separación permite que cada parte del flujo tenga una función concreta. La capa raw conserva los datos en un estado cercano al original, la capa business integra e historifica la información mediante Data Vault, y los data marts preparan los datos para su análisis final en Power BI.

En enfoques más tradicionales o simples de Data Warehouse, es habitual cargar los datos, aplicar transformaciones y dejarlos directamente preparados para reporting en una misma capa o en un número reducido de tablas analíticas. Esto puede funcionar en escenarios pequeños, pero también hace que el modelo sea más difícil de mantener cuando aparecen errores, cambios en las reglas de negocio o nuevas fuentes de información. Si no se conserva una capa inicial con el dato original, resulta más complicado revisar de dónde venía un valor o reprocesar la información sin volver a consultar el sistema origen.

La separación por capas reduce este problema. Si una regla de negocio cambia, no es necesario rehacer todo el flujo desde cero ni depender únicamente del dato ya transformado. Se puede volver a partir de la capa raw, aplicar de nuevo las transformaciones necesarias en la capa business y regenerar los data marts finales. Además, esta organización evita mezclar en una misma estructura objetivos distintos: almacenar el dato recibido, integrarlo e historificarlo, y prepararlo para consulta.

En comparación con un modelo dimensional construido directamente desde las fuentes, esta arquitectura ofrece más control y trazabilidad. El modelo dimensional es muy útil para el análisis final, pero si se carga directamente desde los sistemas origen, parte de la lógica de integración e histórico queda demasiado ligada a las tablas de hechos y dimensiones. En este proyecto, en cambio, el modelo dimensional se reserva para los data marts, mientras que la integración y la historificación se gestionan previamente en la capa business mediante Data Vault.

4.1.6 USO DE APIS FRENTE A CONEXIONES DIRECTAS A BASES DE DATOS DE TERCEROS

Otra decisión importante del proyecto ha sido plantear la integración con determinados sistemas externos mediante APIs, en lugar de conectarse directamente a sus bases de datos. En un entorno empresarial real, sistemas como un CRM, un ERP o una plataforma logística suelen ser aplicaciones críticas para el negocio. Por ello, acceder directamente a sus tablas internas puede generar problemas de seguridad, rendimiento y mantenimiento.

El uso de APIs permite que cada sistema exponga únicamente la información necesaria y de una forma más controlada. En lugar de consultar directamente la base de datos interna de un tercero, se consumen endpoints definidos previamente, donde se puede limitar qué datos se devuelven, aplicar filtros y establecer mecanismos de autenticación. Esto reduce el acoplamiento entre sistemas, ya que el consumidor de los datos no necesita conocer la estructura interna de la base de datos origen.

Además, conectarse directamente a una base de datos de terceros puede hacer que la arquitectura sea más frágil. Si el sistema origen cambia el nombre de una tabla, modifica una columna o reorganiza su modelo interno, los procesos de ingesta podrían dejar de funcionar. En cambio, una API actúa como una capa intermedia más estable: mientras el contrato de la API se mantenga, los cambios internos del sistema no tienen por qué afectar directamente a los procesos de carga.

4.2 OBJETIVOS

El objetivo principal del proyecto consiste en el diseño e implementación de una arquitectura de datos end-to-end en un entorno e-commerce. A partir de este objetivo más general, se han definido los siguientes objetivos específicos.

- Generar un conjunto de datos sintético y coherente, que permita representar un escenario de e-commerce con distintas entidades de negocio. Las entidades creadas fueron: cliente, pedido, línea de pedido, producto, categoría y envío.
- Simular varias fuentes de información, combinando datos expuestos mediante APIs, ficheros CSV y ficheros JSON almacenados en un Data Lake, con el fin de reflejar una arquitectura en un entorno real.
- Diseñar una arquitectura cloud que permita almacenar, ingerir y procesar los datos utilizando diferentes servicios de Microsoft Azure como Azure Functions, Azure SQL Database, Azure Data Lake Storage Gen2.
- Implementar una capa de almacenamiento inicial donde se conserven los datos recibidos en un estado cercano al original, facilitando la trazabilidad y posibles reprocesos.

- Construir un Data Warehouse basado en Data Vault, separando entidades de negocio, relaciones y atributos mediante hubs, links y satélites. Utilización de Data Vault como herramienta de modelado que nos permite una correcta historificación y auditoría que permite que se conserve la evolución de los datos y tener trazabilidad de estos.
- Automatizar los procesos de carga y transformación mediante pipelines de Azure Data Factory y procedimientos almacenados en la base de datos.
- Crear data marts orientados al análisis, utilizando esquemas en estrella que simplifiquen el consumo de la información desde herramientas de BI.
- Desarrollar dashboards en Power BI que permitan analizar la información final desde distintas perspectivas y tomar decisiones de negocio en base a la información recopilada y transformada.
- Validar que el flujo completo de datos funciona correctamente, desde la fuente original hasta la visualización final, demostrando la viabilidad de la arquitectura planteada.

4.3 METODOLOGÍA

La metodología de este proyecto se ha organizado en fases, ya que la arquitectura implementada permite seguir de forma natural el recorrido del dato desde su origen hasta su explotación final. Este planteamiento ha permitido trabajar de una manera más estructurada e iterativa, algo habitual en proyectos de datos, Big Data y desarrollo de software, donde cada bloque depende en parte del resultado de las fases anteriores.

Además, durante el desarrollo del proyecto algunas fases posteriores han requerido pequeños ajustes sobre tareas ya realizadas previamente, como la generación de datos, las cargas o el modelado. Este tipo de revisiones no ha supuesto una desviación relevante respecto a la planificación inicial, ya que las estimaciones se han planteado con cierto margen para poder absorber correcciones y cambios menores durante el desarrollo.

A continuación, se describen las principales tareas realizadas, organizadas por semanas.

4.3.1 FASE 1: PLANTEAMIENTO INICIAL Y DEFINICIÓN DEL CASO DE USO

Durante esta primera fase se definió la idea inicial del proyecto. Debido a que el proyecto desarrollado durante las prácticas en empresa no podía presentarse como Trabajo Fin de Máster por motivos de confidencialidad, se decidió adaptar el enfoque trabajado a un entorno propio, utilizando una arquitectura basada en Data Vault, pero aplicada sobre datos sintéticos.

A partir de esta decisión, se seleccionó un escenario de e-commerce. En primer lugar, se identificaron las entidades principales que formarían parte del proyecto, así como los campos necesarios para cada una de estas entidades, teniendo en cuenta las necesidades analíticas que se querían cubrir en fases posteriores.

4.3.2 FASE 2: GENERACIÓN Y PREPARACIÓN DE DATOS SINTÉTICOS

En esta fase se desarrollaron los scripts de Python para la generación de datos sintéticos para las diferentes entidades definidas en la fase anterior. Estos scripts fueron desarrollados en local, a diferencia de fases posteriores que se desarrollaron en la nube. Se utilizaron diferencias librerías estadísticas y de generación de datos sintéticos para la generación de datos lo mas reales posibles además de implementación de lógica para que los datos entre entidades fuesen coherentes.

Finalmente, los datos generados se almacenaron en ficheros en diferentes formatos intentando simular un entorno Big Data lo más real posible.

4.3.3 FASE 3: PREPARACIÓN DE MECANISMOS DE INGESTA

En esta fase se prepararon las distintas formas de entrada de datos que se utilizarían posteriormente en la arquitectura. Por un lado, se crearon los endpoints necesarios para simular la exposición de información mediante APIs, representando sistemas externos como un CRM o un ERP. Para la exposición de estos endpoints se utilizaron las Azure Functions, servicio de Microsoft Azure que nos permite el desarrollo de APIs serverless. Por otro lado, se organizaron los ficheros CSV y JSON generados en la fase anterior, que se almacenarían en el Azure Data Lake, almacenamiento compatible con los pipelines de ingesta de Azure Data Factory.

4.3.4 FASE 4: DISEÑO DE LA ARQUITECTURA CLOUD EN AZURE

En esta fase se completó el diseño de la arquitectura cloud del proyecto en Azure. Algunos servicios ya se habían preparado en fases anteriores, como Azure Functions para la simulación de APIs y Azure Data Lake Storage Gen2 para el almacenamiento inicial de ficheros. A partir de ahí, se levantaron los servicios restantes necesarios para construir el flujo completo de datos, principalmente el servidor lógico de SQL, Azure SQL Database y Azure Data Factory.

El trabajo de esta fase no consistió únicamente en crear los recursos, sino en conectarlos entre sí para que funcionasen como una arquitectura de datos integrada. Se configuró la comunicación entre Data Factory, el Data Lake, las APIs expuestas mediante Azure Functions y la base de datos SQL, asegurando que los datos pudieran moverse correctamente desde las fuentes hasta las capas analíticas. De esta forma, los distintos servicios dejaron de

funcionar como componentes aislados y pasaron a formar parte de un sistema cloud coherente para la ingesta, almacenamiento, transformación y carga de datos.

4.3.5 FASE 5: MODELADO, TRANSFORMACIÓN Y CARGA EN DATA VAULT

En esta fase se desarrolló la parte principal del Data Warehouse dentro de Azure SQL Database. A partir de los datos procedentes de las fuentes, se crearon las tablas raw para almacenar la información inicial y las vistas stage para preparar los datos antes de cargarlos en el modelo. Estas vistas permitieron aplicar las primeras transformaciones, como normalización de formatos, tipado de campos y preparación de claves técnicas.

Posteriormente, se construyeron las capas RaVa y BuVa siguiendo el enfoque Data Vault. En la RaVa se implementaron las estructuras principales del modelo, separando entidades, relaciones y atributos mediante hubs, links y satélites. Sobre esta base, la BuVa permitió aplicar reglas de negocio y dejar la información más preparada para su uso analítico posterior.

Además del modelado, en esta fase se desarrollaron los procedimientos almacenados encargados de realizar las cargas entre capas. Estos procedimientos permiten automatizar la inserción de datos en las distintas estructuras, controlar cambios en los satélites y mantener la información historificada. También se integraron estas cargas en pipelines de Azure Data Factory, de forma que los procesos pudieran ejecutarse de manera ordenada desde la ingesta inicial hasta las capas finales.

4.3.6 FASE 6: CONSTRUCCIÓN DATA MARTS Y VISUALIZACIÓN EN POWER BI

En esta última fase se construyeron los data marts finales a partir de la información ya integrada en el Data Warehouse. El objetivo fue transformar el modelo Data Vault, más orientado a integración e historificación, en estructuras más sencillas y preparadas para el análisis. Para ello, se diseñó un modelo en estrella, separando tablas de hechos y dimensiones, de forma que la información pudiera ser consumida de manera más directa desde Power BI.

Una vez creados los data marts, se conectó Power BI a la base de datos para desarrollar los dashboards del proyecto. De esta forma, los datos ya tratados y organizados se transformaron en información visual útil para extraer conclusiones de negocio.

4.3.7 DIAGRAMA DE GANTT POR SEMANAS

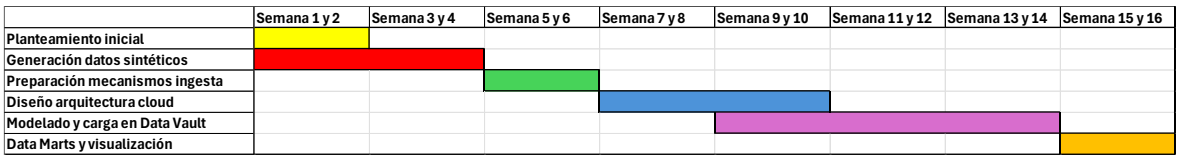


Figura 11. Diagrama de Gantt

4.4 ESTIMACIÓN ECONÓMICA

Aunque el proyecto desarrollado no corresponde a un encargo real de una empresa, sí puede entenderse como una simulación de un proyecto de consultoría tecnológica aplicado al ámbito del Big Data. La solución se ha construido tomando como referencia el tipo de proyectos realizados en un entorno profesional, pero utilizando datos sintéticos y un caso de uso propio. Por este motivo, no existe un presupuesto cerrado asociado a un cliente, ni una facturación real por entregables.

En un proyecto real de consultoría, la estimación económica se definiría normalmente durante las fases iniciales, junto con el alcance, los objetivos, los plazos y los recursos necesarios. Antes de comenzar el desarrollo, se analizarían las necesidades del cliente, las fuentes de datos disponibles, los sistemas con los que habría que integrarse, el volumen de información, la frecuencia de carga, los requisitos de seguridad y el tipo de informes o cuadros de mando esperados. A partir de este análisis se establecería una planificación de trabajo y una estimación de horas por perfil.

Este tipo de estimación suele depender principalmente de la dedicación del equipo. En un proyecto como el planteado podrían intervenir perfiles como arquitecto de datos, ingeniero de datos, ingeniero cloud, analista BI o Project Manager. Cada perfil tendría un coste e impacto distinto y una dedicación determinada en función de las tareas asignadas.

Además del coste asociado al equipo de trabajo, también habría que considerar el coste de la infraestructura cloud y de las herramientas utilizadas. En este TFM se ha trabajado con una suscripción académica a Microsoft Azure y con servicios ajustados al alcance del proyecto, por lo que el coste real ha sido reducido. Sin embargo, en un entorno empresarial habría que estimar el consumo de servicios como almacenamiento, bases de datos, ejecución de pipelines, funciones serverless y licencias de Power BI. Estos costes dependerían del volumen de datos, la frecuencia de ejecución de los procesos, el número de usuarios y los requisitos de disponibilidad y rendimiento.

Por tanto, la estimación económica de este trabajo debe entenderse como una aproximación teórica. No se trata de valorar un proyecto real facturado a un cliente, sino de identificar qué elementos tendrían impacto económico si la solución se implantase en una empresa.

Capítulo 5. SISTEMA/MODELO DESARROLLADO

En este capítulo se describe el sistema desarrollado desde un punto de vista estructural y funcional. El objetivo no es volver a explicar las tecnologías utilizadas, ya tratadas en apartados anteriores, sino mostrar cómo se ha organizado la solución y cómo encajan entre sí las distintas partes que forman el proyecto.

Para ello, se presentan los principales modelos de datos utilizados, incluida la adaptación de los datos al modelo Data Vault. También se muestra la arquitectura cloud construida en Azure, el flujo seguido por los datos desde su llegada en bruto hasta su preparación en data marts, y la forma en la que se orquestan los procesos de carga mediante pipelines.

De esta manera, el capítulo permite entender el recorrido completo de la información dentro del sistema. Desde las fuentes iniciales hasta los modelos finales de análisis, cada capa cumple una función concreta dentro de la arquitectura: recibir los datos, prepararlos, integrarlos, historificarlos y dejarlos listos para su explotación analítica.

5.1 ARQUITECTURA CLOUD DESARROLLADA EN AZURE

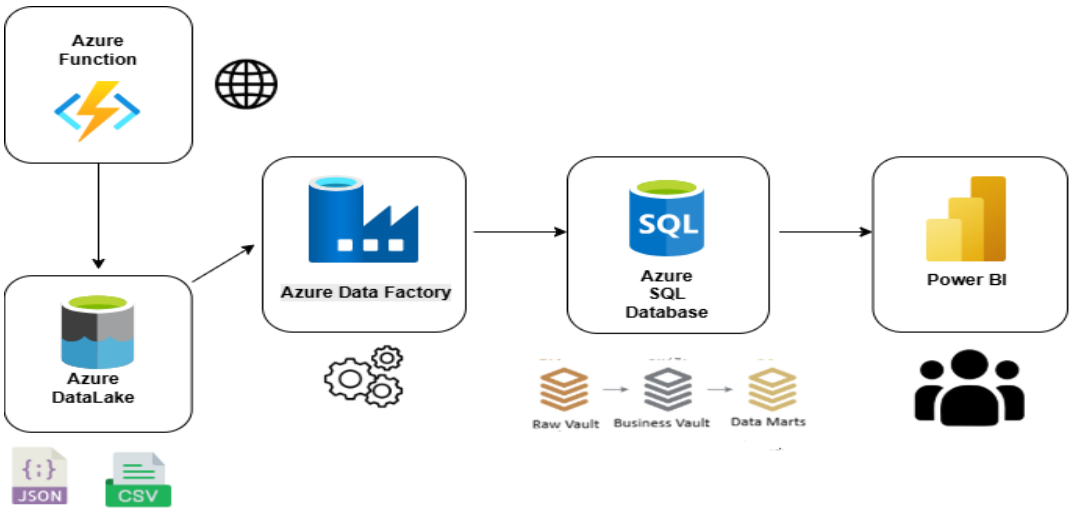


Figura 12. Arquitectura cloud en Azure

La arquitectura cloud desarrollada representa la unión de los distintos servicios desplegados en Azure para formar un sistema analítico completo. En este apartado no se pretende volver a describir en detalle cada herramienta, ya que estas fueron explicadas en el Capítulo 2. , sino mostrar cómo se conectan entre sí dentro de la solución implementada y qué función cumple cada componente dentro del flujo general de datos.

El punto de partida de la arquitectura son las fuentes de información. En el proyecto se simulan sistemas externos que proporcionan datos de clientes, pedidos y líneas de pedido mediante APIs expuestas con Azure Functions. Además, se utilizan ficheros CSV y JSON almacenados en Azure Data Lake Storage Gen2 para representar otras fuentes de datos, como productos, categorías y envíos.

La ingesta y orquestación de estos datos se realiza mediante Azure Data Factory. Este servicio actúa como elemento central del flujo, conectándose tanto a las APIs como al Data Lake y a la base de datos SQL. A través de pipelines se automatizan las cargas, se controla el orden de ejecución y se lanzan los procesos necesarios para mover la información desde las fuentes hasta el Data Warehouse. Además de la ingesta inicial, Data Factory permite ejecutar procedimientos almacenados en la base de datos, lo que facilita la carga de las distintas capas del modelo.

Una vez que los datos llegan a Azure SQL Database, comienza el flujo interno del Data Warehouse. En primer lugar, la información se almacena en la capa raw y posteriormente se prepara mediante vistas stage. A partir de ahí se cargan las estructuras del modelo Data Vault, incluyendo hubs, links y satélites. Sobre esta base se aplica posteriormente la lógica de negocio necesaria y se generan las capas finales orientadas al análisis. El detalle de este flujo interno dentro de SQL se desarrolla en el apartado 5.3.

Finalmente, los datos ya tratados y preparados en los data marts se consumen desde Power BI. Esta última capa permite convertir la información almacenada en el Data Warehouse en dashboards y reportes analíticos.

5.2 MODELO DATA VAULT IMPLEMENTADO

En este apartado se presenta el modelo Data Vault implementado a través del diagrama entidad-relación del esquema Raw Vault. Se ha decidido utilizar este esquema como referencia principal porque representa la base estructural del Data Warehouse y muestra cómo se han organizado las entidades y relaciones procedentes de las distintas fuentes antes de aplicar reglas de negocio más específicas.

El Raw Vault resulta especialmente útil para explicar el diseño del modelo, ya que contiene la información integrada de forma trazable e historificada, pero todavía mantiene una relación cercana con los datos de origen. En cambio, el Business Vault se apoya sobre esta base para aplicar lógica de negocio, cálculos o transformaciones derivadas, por lo que depende más de las necesidades analíticas concretas del proyecto. Por este motivo, el diagrama del Raw Vault permite mostrar de forma más clara la estructura principal y más estable del modelo.

Una diferencia visible respecto a un diagrama entidad-relación tradicional es que el modelo Data Vault contiene un mayor número de tablas. Esto se debe a que Data Vault separa de forma explícita las entidades, las relaciones y los atributos, en lugar de agrupar toda la información en una única tabla por entidad. Por ejemplo, la información de un cliente no se almacena únicamente en una tabla de clientes, sino que puede dividirse entre un hub de cliente y uno o varios satélites asociados.

Esta mayor separación tiene ventajas importantes. Permite incorporar nuevas fuentes de datos con menor impacto sobre el modelo existente, facilita conservar el histórico de los cambios y mejora la trazabilidad de la información. Además, al separar los atributos en satélites, es posible controlar mejor qué información cambia, cuándo cambia y desde qué fuente procede.

Como contrapartida, el modelo puede resultar más complejo de leer que un modelo relacional tradicional, ya que aumenta el número de tablas y relaciones. También puede requerir más trabajo en las consultas si se accede directamente al Raw Vault, debido a la necesidad de unir hubs, links y satélites. Sin embargo, esta complejidad se compensa con la construcción posterior de capas más orientadas al consumo, como el Business Vault y los data marts, que simplifican el acceso a la información para el análisis final.

La Figura 13 muestra una visión general del modelo Raw Vault implementado en el proyecto. Se trata de un diagrama simplificado, en el que se representan únicamente las estructuras principales y sus relaciones, sin incluir todavía el detalle de los atributos de cada tabla. Esta decisión se ha tomado para facilitar la lectura del modelo dentro de la memoria, ya que incluir todos los campos técnicos y descriptivos en un único diagrama dificultaría su visualización.

Posteriormente, se incluirán diagramas más detallados con los atributos de cada entidad. En estos, se explicarán algunos campos importantes para comprender el funcionamiento del modelo, como las claves hash, los identificadores procedentes de los sistemas origen, los campos de auditoría y los campos utilizados para controlar la vigencia e historificación de los datos.

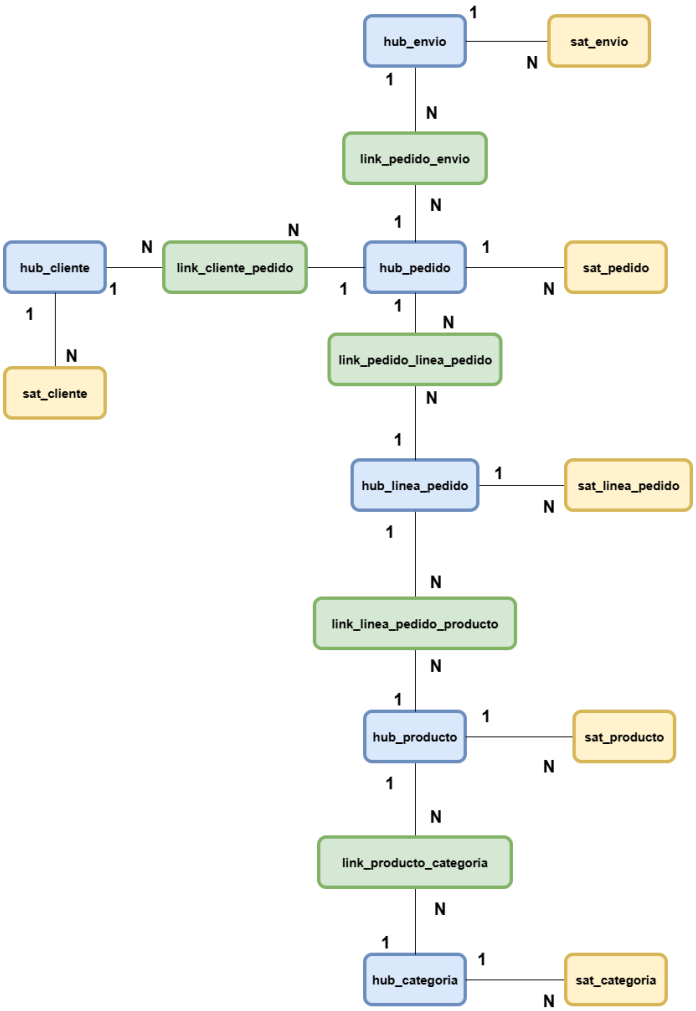


Figura 13. Diagrama entidad-relación modelo Data Vault

5.2.1 ENTIDADES HUBS

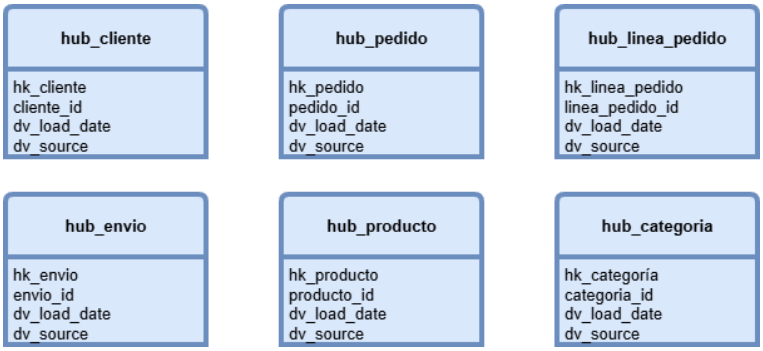


Figura 14. Entidades Hubs

En la Figura 14 se muestran los hubs definidos en el modelo Raw Vault. El campo principal de cada hub es la clave hash, representada con el prefijo hk_. Por ejemplo, hk_cliente, hk_pedido o hk_producto. Esta clave se genera a partir del identificador de negocio de la entidad y se utiliza como clave técnica dentro del modelo Data Vault. Su uso permite relacionar los hubs con links y satélites de forma consistente, sin depender de claves secuenciales generadas por la base de datos.

Junto a la clave hash se conserva también el identificador original procedente del sistema fuente, como cliente_id, pedido_id, producto_id o envio_id. Estos campos son importantes porque mantienen la trazabilidad con el dato de origen. De esta forma, aunque el modelo utilice claves técnicas internas, siempre es posible saber qué identificador tenía el registro en el sistema inicial.

Por último, los hubs incluyen campos técnicos como dv_load_date y dv_source. El campo dv_load_date indica el momento en el que el registro fue cargado en el Data Warehouse, mientras que dv_source permite identificar la fuente de la que procede la información. Estos campos son fundamentales para auditoría y trazabilidad, ya que permiten conocer cuándo se incorporó cada registro y desde qué origen llegó.

5.2.2 ENTIDADES LINKS

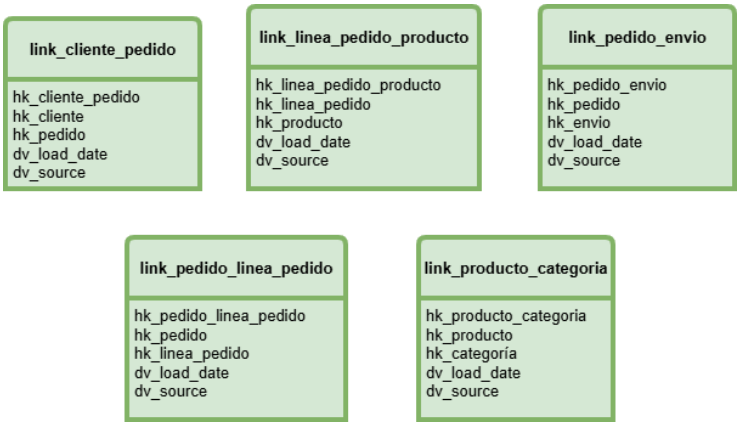


Figura 15. Entidades Links

Cada link contiene una clave hash propia, identificada con el prefijo hk_, como hk_cliente_pedido, hk_pedido_envio o hk_producto_categoria. Esta clave identifica de forma única la relación dentro del Data Warehouse. Además, el link almacena las claves hash de los hubs que participan en dicha relación. Por ejemplo, link_cliente_pedido contiene hk_cliente y hk_pedido, mientras que link_linea_pedido_producto contiene hk_linea_pedido y hk_producto.

Esta forma de modelar permite separar las relaciones de las entidades y de sus atributos descriptivos. En lugar de guardar toda la información en una misma tabla, Data Vault mantiene por separado qué entidad existe, con qué otras entidades se relaciona y qué atributos tiene. Esto facilita que el modelo pueda crecer o adaptarse si aparecen nuevas relaciones entre entidades.

Al igual que los hubs, los links incluyen campos técnicos como dv_load_date y dv_source. Estos campos permiten conocer cuándo se cargó la relación en el Data Warehouse y desde qué fuente procede.

5.2.3 ENTIDADES SATÉLITES

Figura 16. Entidades Satélites

En la Figura 16 se muestran los satélites definidos en el modelo Raw Vault. Estas tablas almacenan los atributos descriptivos de cada entidad de negocio y permiten conservar su evolución en el tiempo.

Cada satélite se relaciona con su hub correspondiente mediante la clave hash de la entidad. Por ejemplo, `sat_cliente` utiliza `hk_cliente`, `sat_pedido` utiliza `hk_pedido` y `sat_producto` utiliza `hk_producto`. De esta forma, los atributos quedan separados de la identificación principal de la entidad, permitiendo que el modelo sea más flexible y que los cambios en los datos descriptivos no afecten directamente a la estructura de los hubs.

El campo más importante de los satélites es dv_hash_value. Este campo se genera a partir de los atributos descriptivos de cada registro y permite detectar si la información ha cambiado respecto a una carga anterior. Si el hash value es diferente, significa que alguno de los atributos ha variado y, por tanto, debe registrarse una nueva versión del dato.

Para controlar esta historificación se utilizan los campos dv_valid_from, dv_valid_to y dv_flag_last_version. El campo dv_valid_from indica desde qué momento es válida una versión del registro, mientras que dv_valid_to indica hasta cuándo estuvo vigente. Cuando dv_valid_to tiene valor NULL, significa que ese registro sigue siendo la versión vigente. En el momento en que llega un nuevo registro con la misma clave hash, pero con un dv_hash_value diferente, se entiende que alguno de sus atributos ha cambiado. En ese caso, el registro anterior deja de estar vigente, se actualiza su dv_valid_to con la fecha correspondiente y se inserta una nueva versión con dv_valid_to a NULL.

Por su parte, dv_flag_last_version permite identificar de forma directa cuál es la última versión del registro. Este campo facilita las consultas cuando solo se quiere recuperar la información actual, sin tener que analizar todo el histórico almacenado en el satélite.

Al igual que en hubs y links, los satélites incorporan también campos técnicos como dv_load_date y dv_source, que permiten saber cuándo se cargó la información y de qué fuente procede. Esto aporta trazabilidad al modelo y facilita revisar el origen de los datos en caso de errores, cambios inesperados o análisis históricos.

5.3 FLUJO DE DATOS POR CAPAS

El flujo de datos del proyecto se ha organizado en diferentes capas, cada una con una función concreta dentro de la arquitectura. Esta separación permite que el dato avance de forma ordenada desde su carga inicial hasta su preparación final para el análisis, evitando mezclar en una misma estructura los datos en bruto, las transformaciones, la lógica de negocio y los modelos de consumo.

Dentro del Data Warehouse implementado en SQL Server, cada capa se ha definido como un esquema independiente. De esta forma, las estructuras quedan mejor organizadas y resulta más sencillo diferenciar qué función cumple cada objeto dentro de la base de datos. Por ejemplo, las tablas de datos en bruto se agrupan en el esquema raw, las vistas de preparación en stage, las estructuras principales de Data Vault en RaVa (Raw Vault), la lógica de negocio en BuVa (Business Vault) y los modelos finales de análisis en los data marts.

Además, no todas las capas se han implementado de la misma forma. Algunas se construyen mediante tablas materializadas, cuando es necesario almacenar físicamente la información y conservar los resultados de cada carga, como ocurre en raw, RaVa . En cambio, otras capas se han implementado mediante vistas, como stage, BuVa y los data marts.

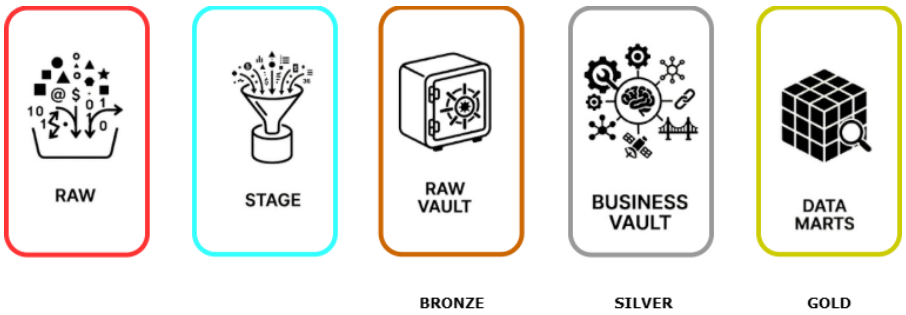


Figura 17. Capas del flujo de datos

5.3.1 CAPA RAW

La capa raw es la primera capa del Data Warehouse y actúa como punto inicial de almacenamiento dentro de SQL Server. En esta capa se cargan los datos procedentes de las distintas fuentes del proyecto, tanto los obtenidos mediante APIs como los recibidos a través de ficheros CSV y JSON.

En la capa raw los datos se almacenan con una tipología muy cercana al origen. La mayoría de los campos se han definido como NVARCHAR, con el objetivo de evitar errores de carga provocados por conversiones de tipos demasiado tempranas. De esta forma, aunque un dato llegue con un formato inesperado, puede almacenarse igualmente en la primera capa del Data Warehouse y revisarse posteriormente en las fases de preparación.

En el caso de las fechas, se ha utilizado el tipo DATETIME2, ya que estos campos son necesarios para controlar momentos de carga, fechas de actualización y vigencia de determinados registros. Esta decisión permite trabajar con fechas de forma más precisa sin perder la capacidad de análisis temporal en capas posteriores.

Esta forma de diseñar la capa raw refuerza la trazabilidad del dato. Si se aplicasen conversiones estrictas desde el primer momento, un error de formato podría impedir que el registro se cargase y, por tanto, se perdería incluso su entrada inicial en el sistema. Al almacenar primero la información de forma flexible, se asegura que el dato queda registrado en raw y que cualquier problema de tipado o calidad puede tratarse después en la capa stage, sin perder la referencia original.

5.3.2 CAPA STAGE

La capa stage se ha implementado mediante vistas sobre las tablas raw. Su objetivo es preparar los datos antes de cargarlos en el modelo Data Vault, aplicando las transformaciones necesarias sin modificar directamente la información almacenada en la capa inicial. De esta forma, raw se mantiene como una copia cercana al origen, mientras que stage actúa como una capa de preparación y normalización.

En estas vistas se realizan tareas como el tipado de campos, la limpieza de formatos y la normalización de fechas. También se generan campos técnicos necesarios para el modelo Data Vault, como claves hash, valores hash de atributos y campos de auditoría. Esto permite que la información llegue a hubs, links y satélites con una estructura más controlada.

La decisión de implementar stage como vistas permite evitar duplicar almacenamiento de forma innecesaria. Al no materializar esta capa en tablas físicas, los datos preparados se calculan a partir de raw en el momento en que son consultados o utilizados por los procedimientos de carga. Esto hace que la capa sea más flexible, ya que cualquier ajuste en la lógica de preparación puede realizarse modificando la vista, sin necesidad de recargar una tabla intermedia completa.

5.3.3 CAPA RAW VAULT

La capa Raw Vault es la primera capa donde los datos pasan a estar modelados siguiendo Data Vault. A partir de la información preparada en stage, se cargan las tablas principales del modelo, separando las entidades de negocio, sus relaciones y los atributos que pueden cambiar con el tiempo.

En este proyecto, la Raw Vault se ha creado mediante tablas físicas, ya que es necesario conservar los datos cargados y mantener su histórico. En esta capa se almacenan los hubs, links y satélites, junto con campos técnicos como la fecha de carga, la fuente del dato, las claves hash y los campos de vigencia. Esto permite que la información quede integrada y pueda seguirse su evolución sin perder la relación con el origen.

La idea de esta capa no es aplicar todavía reglas de negocio complejas, sino dejar los datos bien organizados dentro del modelo Data Vault. De esta forma, la Raw Vault actúa como una base estable sobre la que después se puede construir la Business Vault y, finalmente, los data marts orientados al análisis.

5.3.4 CAPA BUSINESS VAULT

Esta capa se construye a partir de la información ya integrada en la Raw Vault. En esta capa se empiezan a aplicar reglas de negocio sin modificar la información original almacenada en las capas anteriores. En esta capa se trabajan reglas que no vienen directamente en el dato de origen, pero que son necesarias para analizarlo mejor después.

En este proyecto, la Business Vault se ha implementado principalmente mediante vistas. Esto permite aplicar reglas de negocio de forma flexible, sin duplicar almacenamiento ni alterar los datos historificados de la Raw Vault. Por ejemplo, una vista de Business Vault podría identificar clientes fidelizados con mucha antigüedad y clasificarlos como clientes premium. Esta categoría no existe necesariamente en el dato original, sino que se obtiene aplicando una regla definida sobre la información cargada en las tablas RaVa.

5.3.5 CAPA DATA MART

La capa de data marts es la última capa del flujo dentro del Data Warehouse. A diferencia de la Raw Vault y la Business Vault, los data marts se construyen pensando directamente en el análisis y en el consumo desde Power BI.

En este proyecto, los data marts se han implementado mediante vistas, organizando la información en un modelo más sencillo y cercano al esquema en estrella. Esto permite trabajar con tablas de hechos y dimensiones. El objetivo de esta capa es ocultar la complejidad de las capas anteriores y presentar los datos en una estructura más fácil de utilizar.

5.4 MODELO ANALÍTICO STAR SCHEMA

Tal y como se ha comentado en el apartado 5.3.5 los data marts se han diseñado como la capa final de consumo analítico. En este apartado se muestra el modelo concreto construido para el análisis de ventas, organizado mediante un esquema en estrella.

La tabla central del modelo es fact_ventas, que recoge la información principal de las operaciones de venta y actúa como punto de conexión con el resto de las dimensiones. A partir de esta tabla se relacionan las dimensiones dim_cliente, dim_producto, dim_fecha y dim_logistica, que permiten analizar la información desde distintas perspectivas.

Este diseño permite estudiar las ventas según el cliente que realiza la compra, el producto vendido, el momento temporal de la operación y la información logística asociada al pedido o envío. De esta forma, el modelo queda preparado para construir medidas, filtros y visualizaciones en Power BI de manera precisa y amigable.

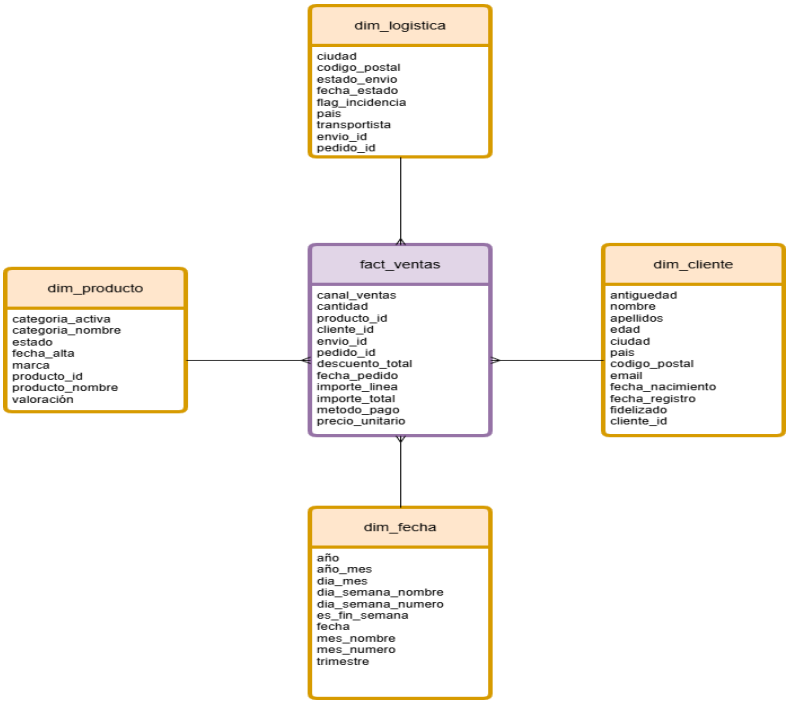


Figura 18. Star Schema para consumo analítico

5.5 ORQUESTACIÓN DE PROCESOS MEDIANTE PIPELINES

Los pipelines desarrollados se han planteado para ejecutarse en procesos batch, normalmente en ventanas nocturnas o momentos de baja actividad. En este proyecto no era necesario trabajar en tiempo real, ya que el objetivo principal era alimentar una arquitectura analítica y no un sistema operacional que requiriese respuesta inmediata.

En la Figura 19 se muestran los pipelines creados en Azure Data Factory para organizar la ingesta y carga de datos. Cada uno de ellos cumple una función concreta dentro del flujo, desde la obtención de datos desde las fuentes hasta la carga completa del modelo Data Vault.

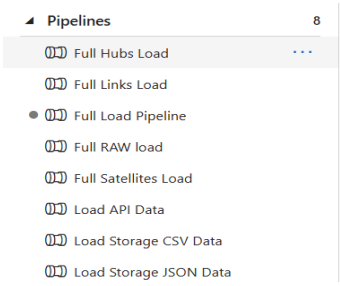


Figura 19. Pipelines Azure Data Factory

- **Load API Data:** Pipeline que se encarga de consumir los datos expuestos mediante APIs. En el proyecto se utiliza para obtener la información de clientes, pedidos y líneas de pedido desde los endpoints creados con Azure Functions.

- **Load Storage CSV Data:** Este pipeline carga los ficheros CSV almacenados en la cuenta de almacenamiento. Se utiliza principalmente para incorporar los datos de productos y categorías al flujo de datos.

- **Load JSON Data:** Este pipeline se encarga de cargar los ficheros JSON almacenados en el Data Lake. En este caso, se utiliza para incorporar la información de envíos procedente del proveedor logístico simulado.

- **Full RAW Load:** Pipeline que agrupa las cargas iniciales hacia la capa raw. Su función es dejar los datos disponibles dentro de SQL Server en un estado cercano al origen, antes de aplicar las transformaciones y cargas posteriores.

- **Full Hubs Load:** Este pipeline ejecuta los procesos de carga de los hubs del modelo Data Vault. A través de él se incorporan las entidades principales del negocio, como clientes, pedidos, productos, categorías, líneas de pedido y envíos.

- **Full Links Load:** Este pipeline lanza las cargas de los links, encargados de representar las relaciones entre las distintas entidades del modelo. Por ejemplo, la relación entre cliente y pedido, pedido y envío, o producto y categoría.

- **Full Satellites Load**: Este pipeline ejecuta las cargas de los satélites, donde se almacenan los atributos descriptivos y se controla la evolución histórica de la información. En esta parte se gestionan los cambios de atributos y la vigencia de los registros.

- **Full Load Pipeline:** Este pipeline actúa como proceso principal de carga completa (como si fuese un orquestador). Su función es coordinar la ejecución del resto de pipelines en el orden correcto, asegurando que primero se carguen los datos en raw y después se alimenten las estructuras del modelo Data Vault.

Capítulo 6. ANÁLISIS DE RESULTADOS

En este capítulo se analizarán los resultados obtenidos a partir de la solución desarrollada. Como bien se ha comentado en apartados anteriores, una vez se ha construido el flujo completo de datos, los datos finales se explotarán mediante Power BI. En este apartado no se van a entrar en aspectos técnicos como en los anteriores, sino que se van a tratar las conclusiones que pueden extraerse a partir de los datos ya limpios, tratados y preparados para el análisis.

A continuación, se incluyen distintas capturas de los dashboards desarrollados y se comentan los principales insights que se pueden obtener de ellos. Esta parte resulta especialmente importante en proyectos de datos, ya que el objetivo no es únicamente almacenar información o construir una arquitectura técnica, sino convertir los datos en conocimiento útil para el negocio. Sin una fase de análisis e interpretación, los datos quedarían como información en bruto o tratada, pero sin aportar valor real a la toma de decisiones.

Es importante destacar que este análisis realizado debe entenderse dentro del contexto del proyecto, ya que los datos utilizados son sintéticos y no proceden de una empresa real. Aun así, los dashboards permiten representar el tipo de análisis que podría realizarse en un entorno empresarial, identificando tendencias y patrones de negocio.

6.1 INSIGHTS CUADRO DE MANDO CLIENTES

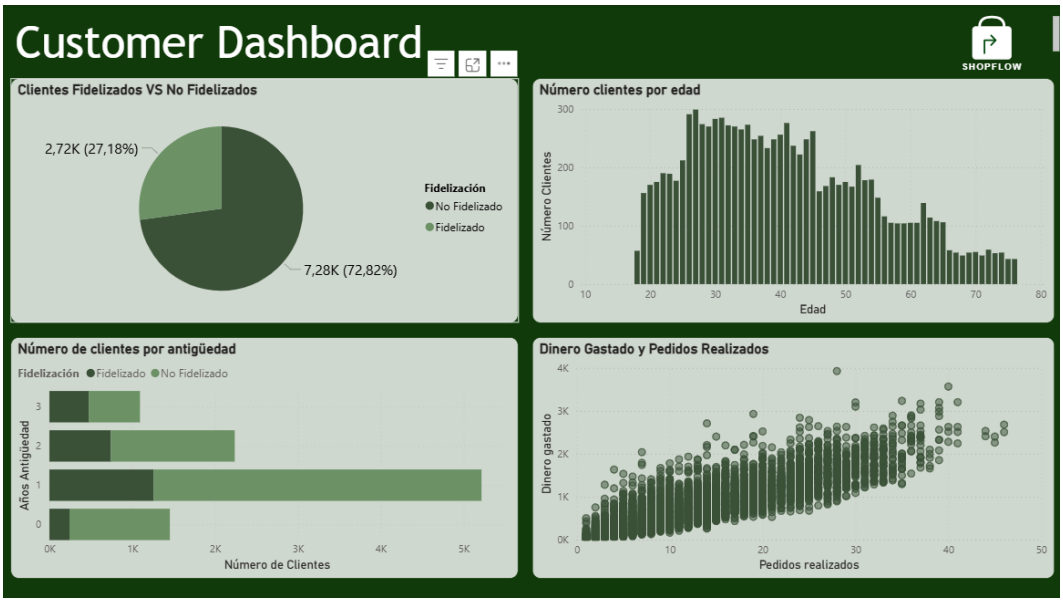


Figura 20. Cuadro de mando Clientes

- Se observa que la mayor parte de los clientes corresponde a usuarios no fidelizados. De esta manera se puede identificar que existen oportunidades de mejora en estrategias de retención de clientes y fidelización de clientes.
- No se consigue apreciar una relación clara entre la fidelización de los clientes y sus años de antigüedad. Se observa que la mayoría de los clientes son recientes (entre 0 y 1 años)
- El pico de clientes está entre los 25 y 40 años. Se recomienda el lanzamiento de campañas de marketing, descuentos y ofertas a perfiles en esos rangos de edades.
- El dinero gastado por los clientes es directamente proporcional a los pedidos realizados. Se pueden identificar clientes con una actividad no muy alta y gastos bastante altos, estos pueden ser clientes relevantes a la hora de realizar las campañas comerciales.

6.2 *INSIGHTS CUADRO DE MANDO PEDIDOS*

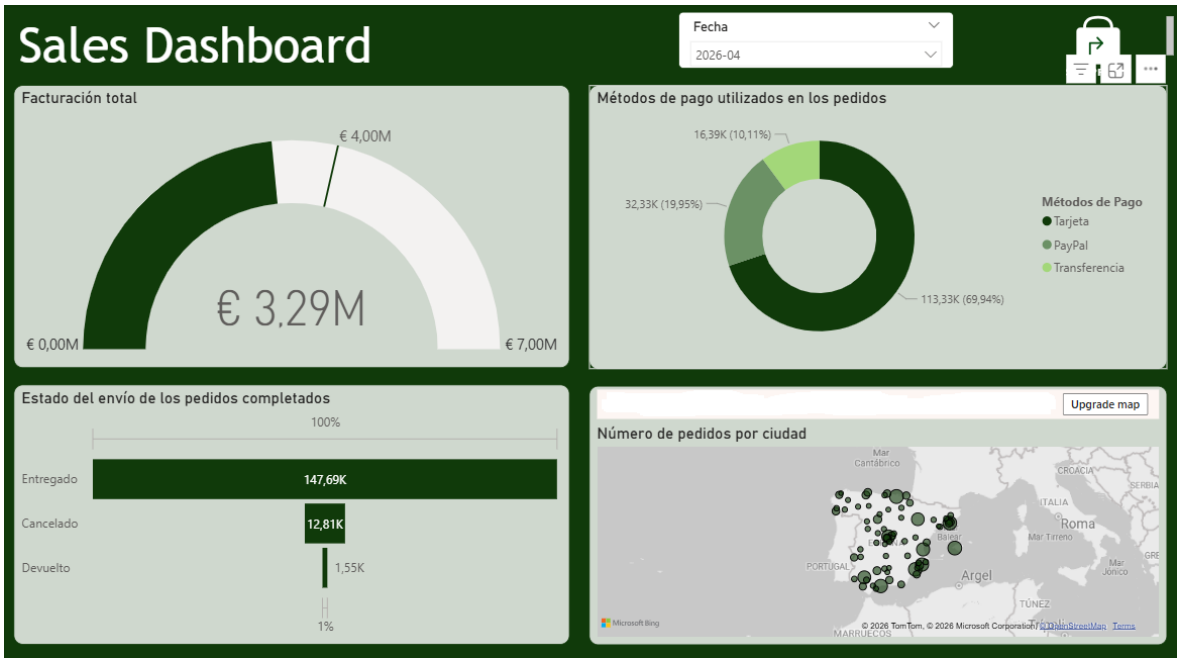


Figura 21. Cuadro de mando Ventas

- En el periodo temporal analizado, aunque la facturación fue buena, sigue estando por debajo de su objetivo. No se esta alcanzando el número de ventas deseado y este factor afecta directamente en la facturación.
- El pago con tarjeta es el método de pago dominante en el total de las ventas en ese periodo temporal.
- La mayoría de los pedidos han sido entregados satisfactoriamente, esto es un buen indicador de que el servicio de logística esta trabajando correctamente. Sin embargo,

se puede apreciar que hay un alto volumen de pedidos cancelados, sería conveniente analizar el porqué de esta situación.

- Se observa centralización de pedidos en distintas zonas de España, especialmente en grandes núcleos urbanos. Este tipo de visualización puede ser útil para detectar ciudades con mayor demanda, planificar estrategias comerciales por zona o analizar posibles necesidades logísticas en determinadas áreas.

6.3 *INSIGHTS CUADRO DE MANDO PRODUCTOS*

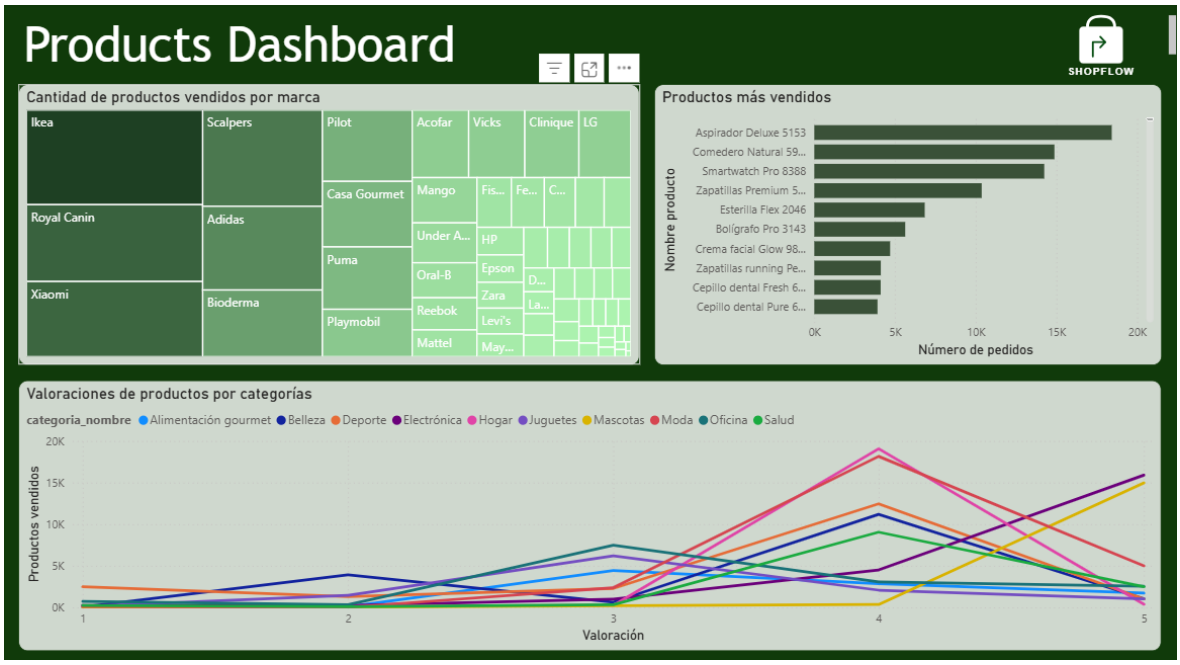


Figura 22. Cuadro de mando Productos

- Productos de marcas como Ikea, Royal Canin, Xiaomi, Scalpers o Adidas son top ventas. Estos fabricantes tienen un mayor peso en el catálogo.
- Se observa en el ranking de productos vendidos aquellos productos que acumulan un número de pedidos mayor que el resto. Esto permite detectar los bestsellers y es útil para el lanzamiento de campañas, promociones o stock disponible.
- Los productos de electrónica o belleza están muy bien valorados entre los clientes mientras que los de oficina o juguetes se puede apreciar que obtienen valoraciones más bajas. Esto nos ayuda a detectar aquellas categorías de productos más estratégica y tomar decisiones en base a estos datos.

Capítulo 7. CONCLUSIONES Y TRABAJOS FUTUROS

7.1.1 CONCLUSIONES Y TRABAJOS FUTUROS

El desarrollo del proyecto ha permitido comprobar cómo una arquitectura de datos puede organizarse de forma progresiva, desde la llegada inicial de la información hasta su explotación final en cuadros de mando. A lo largo del trabajo se ha construido un flujo completo en el que cada capa cumple una función concreta, evitando que la ingesta, la transformación, la historificación y el análisis queden mezclados en una misma estructura.

Una de las ideas más importantes que se extrae del proyecto es que el valor del dato no aparece únicamente por almacenarlo. Para que la información sea realmente útil, es necesario integrarla, limpiarla y prepararla para que pueda ser entendida por perfiles de negocio. En este sentido, el modelo desarrollado permite pasar de datos dispersos y con distintos formatos a una estructura ordenada, capaz de alimentar informes.

El uso de Data Vault ha resultado especialmente adecuado para este tipo de escenario. La separación entre entidades, relaciones y atributos permite que el modelo sea más claro desde el punto de vista técnico y, al mismo tiempo, más preparado para evolucionar. Además, la historificación de los satélites permite conservar cambios que en otros enfoques podrían perderse, algo importante cuando se quiere analizar no solo el estado actual de la información, sino también su evolución.

También se ha comprobado la importancia de mantener trazabilidad durante todo el flujo. Conservar identificadores de origen, fechas de carga, fuentes y campos técnicos permite seguir el recorrido del dato y entender cómo ha llegado hasta las capas finales. Esto resulta clave en cualquier arquitectura analítica, ya que un dashboard pierde parte de su valor si no es posible justificar de dónde procede la información que muestra.

Otro aspecto relevante ha sido la separación entre el modelo interno del Data Warehouse y el modelo final de análisis. Data Vault aporta una estructura sólida para integrar e historificar, pero no es la forma más cómoda para que un usuario final construya informes. Por ello, la creación de data marts en esquema estrella ha permitido simplificar el consumo de los datos y ofrecer a Power BI una capa más directa para crear indicadores, filtros y visualizaciones.

Como línea de trabajo futuro, la arquitectura podría ampliarse incorporando nuevas fuentes de información. Por ejemplo, en un escenario empresarial más amplio podrían añadirse datos procedentes de marketing, campañas publicitarias o atención al cliente. Una de las ventajas del enfoque seguido es que estas nuevas fuentes podrían incorporarse de forma progresiva, añadiendo nuevas entidades al modelo sin tener que rehacer toda la arquitectura existente.

También sería interesante evolucionar la solución hacia un entorno con mayor volumen de datos y mayores exigencias de rendimiento. En ese caso, podrían valorarse plataformas más avanzadas como Azure Synapse Analytics o Microsoft Fabric, así como técnicas de particionado, optimización de consultas, control de calidad automático y monitorización más completa de los procesos de carga.

Por último, la capa analítica podría crecer con nuevos dashboards e indicadores. A partir de la base construida se podrían desarrollar análisis más específicos sobre fidelización, rentabilidad, comportamiento de compra, eficiencia logística o segmentación de clientes. Incluso podría plantearse una evolución hacia modelos predictivos, siempre partiendo de la base ya realizada.

Capítulo 8. BIBLIOGRAFÍA

[5] Microsoft. "Data Vault 2.0 on Azure". Microsoft Community. Disponible en: https://techcommunity.microsoft.com/blog/analyticsonazure/data-vault-2-0-on-azure/3860665

[6] Microsoft. "Implementing Data Vault 2.0 on Fabric Data Warehouse". Microsoft Community. Disponible en: https://techcommunity.microsoft.com/blog/analyticsonazure/implementing-data-vault-2-0-on-fabric-data-warehouse/4227078

[7] Microsoft. "Azure Data Lake Storage hierarchical namespace". Microsoft Learn. Disponible en: https://learn.microsoft.com/en-us/azure/storage/blobs/data-lake-storage-namespace

[8] Microsoft. "Azure Data Factory documentation". Microsoft Learn. Disponible en: https://learn.microsoft.com/en-us/azure/data-factory/

[9] Microsoft. "Power BI documentation". Microsoft Learn. Disponible en: https://learn.microsoft.com/en-us/power-bi/

[10] Microsoft. "SQL Server technical documentation". Microsoft Learn. Disponible en: https://learn.microsoft.com/en-us/sql/sql-server/

[11] FastAPI. "FastAPI framework documentation". FastAPI. Disponible en: https://fastapi.tiangolo.com/

[12] Kimball, R. "Slowly Changing Dimensions". Kimball Group, 2008. Disponible en: https://www.kimballgroup.com/2008/08/slowly-changing-dimensions/

[13] Microsoft. "Understand star schema and the importance for Power BI". Microsoft Learn. Disponible en: https://learn.microsoft.com/en-us/power-bi/guidance/star-schema

[14] Linstedt, D.; Olschimke, M. Building a Scalable Data Warehouse with Data Vault 2.0. Morgan Kaufmann, 2015.

[15] Linstedt, D. Super Charge Your Data Warehouse: Invaluable Data Modeling Rules to Implement Your Data Vault. CreateSpace Independent Publishing Platform, 2011.

[16] Kimball, R.; Ross, M. The Data Warehouse Toolkit: The Definitive Guide to Dimensional Modeling. 3rd ed. Wiley, 2013.