

Diseño y Optimización del Layout de Depósito mediante Algoritmos de Minería de Datos y Desarrollo de Agentes Inteligentes para la Categorización de Productos. En Voltaje S.R.L.

Fecha de presentación:

Nombre del Estudiante: Toledo, Alvaro Julian

Docente Supervisor: Dr. Ing. Araujo, Pedro Bernabé

Tutor Empresa: Biondi, Juan Carlos



Índice de contenidos

1. Introducción.....	3
1.1. Contexto.....	3
1.2. Alcance del informe.....	3
1.3. Objetivos del informe.....	4
2. Descripción del proyecto y objetivos.....	4
2.1. Descripción general.....	4
2.2. Objetivos.....	5
2.2.1. Objetivo General.....	5
2.2.2. Objetivos específicos.....	5
3. Planificación y cronograma.....	6
4. Desarrollo de la práctica supervisada.....	6
4.1. Relevamiento de la información.....	6
4.2. Problemas identificados.....	7
4.3. Análisis y estructuración de productos.....	8
4.4. Desarrollo de sistema multi-agente de categorización.....	9
4.4.1. Diagrama de ontología.....	10
4.4.2. Diagrama de casos de uso (DCU).....	11
4.4.3. Diagrama de secuencia.....	12
4.5. Desarrollo de sistema generador de ventas.....	17
4.5.1. Objetivo del proyecto.....	17
4.5.2. Arquitectura general del sistema.....	18
4.5.3. Funcionamiento.....	18
4.5.4. Resultado final.....	19
4.6. Desarrollo de pipeline ETL.....	20
4.6.1. Alcance del pipeline ETL.....	21
4.6.2. Objetivo general.....	21
4.6.3. Objetivos específicos.....	21
4.6.4. Arquitectura de la solución propuesta.....	22
4.6.5. Desarrollo del pipeline de datos.....	22
4.6.6. Capas del Data Warehouse.....	23
4.6.6.1. Capa Bronze.....	24
4.6.6.2. Capa Silver.....	24
4.6.6.3. Capa Datamining.....	24
4.6.6.4. Capa Gold.....	25
4.6.7. Resultados finales.....	25
4.7. Análisis de patrones de asociación.....	26
4.8. Diseño de Layout optimizado para el depósito.....	28
5. Conclusiones y Aprendizajes.....	31
6. Trabajos Futuros.....	31
7. Referencias bibliográficas.....	32

1. Introducción

1.1. Contexto

La práctica supervisada se desarrolló en el marco de un proyecto realizado para la sucursal Yerba Buena de la empresa Voltaje, dedicada a la comercialización de materiales eléctricos y luminarias. En la actualidad, el depósito de la sucursal presenta dificultades operativas relacionadas con la organización y localización de los productos, lo que impacta directamente en los tiempos de trabajo y en la eficiencia del personal.

Uno de los principales problemas identificados es la demora en el armado de pedidos, debido a que los estantes no cuentan con señalización que indique qué productos se encuentran almacenados en cada uno. Esta situación se ve agravada por la elevada rotación de personal en el área de depósito, lo que provoca que los nuevos empleados dependan constantemente del encargado del depósito para ubicar los productos. Esta dependencia genera interrupciones, pérdida de tiempo y una sobrecarga en las tareas del personal con mayor experiencia.

Asimismo, durante el proceso de abastecimiento del depósito también se producen demoras, ya que no existe una referencia clara sobre la ubicación asignada a cada producto. La falta de un layout definido y de criterios de categorización dificulta la reposición ordenada de los artículos, incrementando los tiempos operativos y la posibilidad de errores en la ubicación de los productos.

Frente a este contexto, el proyecto propone el diseño de un layout optimizado del depósito, acompañado por una correcta categorización y señalización de los productos. El objetivo es mejorar la organización del espacio, facilitar la identificación y localización de los artículos, reducir los tiempos de armado y reposición, y disminuir la dependencia del personal nuevo respecto del encargado del depósito, contribuyendo así a una operación más eficiente y ordenada.

1.2. Alcance del informe

Este informe documenta el proceso completo desarrollado durante la práctica supervisada para el análisis y diseño de una solución integral orientada a optimizar el layout del depósito de la sucursal Yerba Buena de la empresa Voltaje. El alcance incluye el relevamiento de la situación actual del depósito y la identificación de problemas operativos vinculados al armado de pedidos y al abastecimiento de productos, así como el diseño e implementación de un pipeline de datos que permite procesar información histórica de ventas de forma ordenada y reproducible. Sobre esta base, se desarrolló un sistema multi-agente encargado de analizar los datos mediante algoritmos de asociación como Apriori, FP-Growth y Eclat, con el objetivo de identificar patrones de co-ocurrencia entre productos. A partir de los resultados obtenidos, se elaboró una propuesta de layout optimizado del depósito, orientada a mejorar la organización de los artículos, reducir recorridos internos y disminuir los tiempos operativos. El informe presenta la arquitectura general de la solución, los modelos y diagramas utilizados, y la descripción del layout propuesto, con el objetivo de servir como base para una futura implementación en la sucursal.

1.3. Objetivos del informe

Los objetivos principales del presente informe son:

- Describir las actividades realizadas durante la práctica supervisada, detallando cada una de las etapas del proceso de análisis y diseño del layout optimizado del depósito, así como la implementación del pipeline de datos y del sistema multi-agente.
- Presentar evidencia del trabajo técnico realizado, incluyendo la arquitectura del pipeline, los diagramas del sistema multi-agente, los modelos de datos, los resultados obtenidos a partir de los algoritmos de asociación Apriori, FP-Growth y Eclat, y la propuesta final de layout.
- Reflejar el cumplimiento de los objetivos definidos en el plan de trabajo, mostrando cómo las soluciones desarrolladas contribuyen a mejorar la organización de los productos y la eficiencia operativa del depósito.
- Documentar de manera clara y ordenada todos los elementos necesarios para que la propuesta de layout y la solución analítica puedan ser comprendidas y utilizadas como base para una futura implementación o ampliación en la sucursal.

2. Descripción del proyecto y objetivos

2.1. Descripción general

El proyecto consiste en el análisis y diseño de un layout optimizado para el depósito de la sucursal Yerba Buena de la empresa Voltaje S.R.L. dedicada a la comercialización de materiales eléctricos y luminarias, con el objetivo de mejorar la organización de los productos y reducir los tiempos operativos asociados al armado de pedidos y al abastecimiento del depósito. La propuesta surge a partir de problemáticas detectadas en el funcionamiento actual, como la falta de categorización de productos, falta de señalización en estanterías, la alta rotación de personal y la dependencia constante del encargado del depósito para la localización de productos.

La solución desarrollada se apoya en un sistema multi-agente para la categorización automática de productos, un pipeline de datos ETL y técnicas de análisis de asociaciones aplicadas sobre información de ventas generadas sintéticamente. Este enfoque permite identificar patrones de compra y relaciones frecuentes entre productos, a partir de los cuales se propone un layout que agrupa estratégicamente los artículos más asociados y frecuentes, facilitando su acceso físico dentro del depósito. El análisis se basa en algoritmos de minería de datos como Apriori, FP-Growth y Eclat, aportando criterios objetivos y basados en datos para el diseño del layout, con el objetivo de mejorar la eficiencia operativa y reducir los tiempos de preparación de pedidos.

2.2. Objetivos

2.2.1. Objetivo General

Colaborar en el análisis y diseño de una propuesta de reorganización del depósito de la empresa Voltaje S.R.L., mediante el desarrollo de un agente inteligente de categorización de productos y la implementación de un pipeline de análisis de datos basado en técnicas de minería de asociaciones, con el fin de diseñar un layout físico optimizado que permita mejorar la organización del stock, reducir los tiempos de búsqueda y armado de pedidos, y aumentar la eficiencia operativa del proceso logístico.

2.2.2. Objetivos específicos

- Relevar y analizar el funcionamiento actual del depósito de la sucursal Yerba Buena, identificando las principales problemáticas asociadas a la falta de categorización de productos y a la distribución desorganizada del espacio.
- Diseñar e implementar un sistema multi-agente inteligente capaz de analizar las descripciones del inventario y asignar automáticamente categorías y tipos a los productos, utilizando criterios consistentes.
- Diseñar y ejecutar un proceso de generación de datos sintéticos de ventas, que permita contar con un conjunto de datos estructurado y controlado para el análisis, manteniendo coherencia entre clientes, ventas y detalles de venta.
- Diseñar e implementar un pipeline de datos que permita procesar registros históricos de ventas y preparar la información necesaria para su análisis.
- Aplicar algoritmos de minería de datos (Apriori, FP-Growth y Eclat) para identificar patrones de co-ocurrencia entre productos que son vendidos de manera conjunta.
- Analizar los resultados obtenidos a partir de los algoritmos de asociación para definir agrupamientos estratégicos de productos dentro del depósito.
- Elaborar una propuesta de layout físico optimizado, basada en los patrones de asociación y la categorización de productos, orientada a minimizar recorridos, facilitar el picking y mejorar el proceso de abastecimiento.
- Documentar de forma clara y estructurada todo el proceso técnico y metodológico realizado, dejando una base que permita una futura implementación o ampliación de la solución propuesta.

3. Planificación y cronograma

A continuación, se detalla el cronograma de actividades en un marco de 10 semanas:

Semana	Actividad	Duración (Horas)
1	Investigación y relevamiento	25

2	Análisis y estructuración de productos	20
3	Desarrollo de sistema multi-agente de categorización	50
4		
5	Construcción del pipeline de datos	50
6		
7	Análisis de patrones de asociación	50
8		
9	Diseño del layout físico optimizado	25
10	Documentación y presentación final	15
Total		235

4. Desarrollo de la práctica supervisada

4.1. Relevamiento de la información

La primera etapa de la práctica supervisada consistió en un relevamiento de información orientado a comprender el funcionamiento actual del depósito de la empresa Voltaje S.R.L., en particular el proceso de almacenamiento de productos, armado de pedidos y abastecimiento del stock. El objetivo de esta etapa fue identificar las principales dificultades operativas y los puntos críticos que afectan la eficiencia del trabajo diario dentro del depósito.

Para llevar a cabo el relevamiento, se realizaron observaciones directas del funcionamiento del depósito y conversaciones informales con el encargado del área y con personal operativo. Estas instancias permitieron conocer cómo se organizan actualmente los productos en los estantes, cómo se localizan los artículos al momento de preparar pedidos y qué problemas surgen ante la incorporación de nuevo personal. A partir de este análisis, se detectaron demoras en el armado de pedidos, dependencia del conocimiento individual de ciertos empleados y dificultades en el abastecimiento del depósito, lo que justificó la necesidad de una reorganización basada en criterios objetivos y apoyada en el análisis de datos.

Ejemplos de preguntas realizadas:

Recepción de materiales

- ¿Cómo reciben los productos de los proveedores?
- ¿Se verifica el estado y cantidad del material al ingresar?
- ¿Existe un registro de entrada (manual o digital)?

Almacenamiento

- ¿El depósito está organizado por categorías (ej. luminarias, cables, disyuntores) o por ubicación libre?
- ¿Cómo saben dónde está guardado cada producto? ¿Lo llevan en la memoria, en un plano, en un sistema o en Excel?
- ¿Los productos de alta rotación tienen una ubicación diferenciada?
- ¿Existen estanterías, códigos, señalización o numeración de estantes y pasillos?
- ¿Tienen problemas frecuentes de stock mal ubicado o productos extraviados?
- ¿Quién prepara el pedido? ¿Una sola persona o varios?
- ¿En qué orden buscan los productos dentro del depósito?

4.2. Problemas identificados

Durante el relevamiento se identificaron diversas dificultades en el funcionamiento actual del depósito de la empresa Voltaje S.R.L., relacionadas principalmente con la organización física de los productos y la gestión operativa diaria. Si bien el depósito permite almacenar y abastecer mercadería a las sucursales, la ausencia de criterios definidos para la ubicación de los productos genera ineficiencias que impactan directamente en los tiempos de trabajo y en la productividad del personal.

Las principales problemáticas detectadas fueron las siguientes:

- Falta de organización por categorías: los productos no se encuentran organizados según familias, usos o características comunes, sino que se ubican en los estantes en función del espacio disponible en el momento del ingreso.
- Demoras en el armado de pedidos: al no existir información clara sobre la ubicación de cada producto, los operarios deben recorrer el depósito para localizar los artículos, lo que incrementa los tiempos de preparación de pedidos.
- Dependencia del conocimiento individual: los empleados nuevos o con menor experiencia dependen del encargado del depósito o de operarios con mayor antigüedad para encontrar los productos, generando interrupciones constantes y pérdida de tiempo.
- Alta rotación de personal: la incorporación frecuente de nuevo personal dificulta la transmisión informal del conocimiento sobre la ubicación de los productos y acentúa los problemas de localización.
- Dificultades en el abastecimiento del depósito: al no contar con un criterio fijo de ubicación, durante el reabastecimiento los productos se colocan donde hay espacio disponible, lo que refuerza el desorden y la falta de consistencia en la distribución.
- Ausencia de criterios basados en datos: la distribución actual del depósito no se apoya en información histórica de ventas ni en el análisis de qué productos suelen venderse en conjunto, desaprovechando una fuente clave para mejorar la organización.
- Impacto en la eficiencia operativa: todas estas situaciones derivan en mayores tiempos de búsqueda, mayor esfuerzo físico para los operarios y una menor eficiencia general en el proceso logístico.

En resumen, aunque el depósito cumple su función básica de almacenamiento, la falta de una organización sistematizada y basada en datos genera demoras y dependencia del

conocimiento individual. Estas limitaciones motivaron el desarrollo de una propuesta de reorganización del layout apoyada en la categorización de productos, la generación y análisis de datos de ventas y el uso de técnicas de minería de datos para mejorar la eficiencia operativa.

4.3. Análisis y estructuración de productos

Durante el relevamiento inicial se identificó que el depósito de la empresa Voltaje S.R.L. no cuenta con una organización basada en categorías de productos, lo que genera desorden y dificulta tanto el almacenamiento como la localización de los artículos. Los productos se ubican en los estantes según el espacio disponible y no siguiendo una lógica de agrupación por tipo, uso o familia, lo cual impacta directamente en la eficiencia operativa del depósito.

Sin embargo, al analizar el catálogo disponible en la página web oficial de Voltaje, se observó que los productos comercializados sí se encuentran organizados en categorías bien definidas. El catálogo digital está estructurado en dos grandes grupos principales: Materiales eléctricos e Iluminación, y dentro de cada uno de ellos existen categorías específicas que permiten una clasificación clara de los artículos. Esta diferencia entre la organización del catálogo digital y la del depósito físico evidencia una falta de coherencia entre ambos entornos.

Dado que el catálogo digital ya cuenta con una estructura de categorización establecida, se consideró conveniente respetar y reutilizar dichas categorías como base para la reorganización del depósito. De esta manera, se busca mantener coherencia entre la información que se presenta al cliente y la organización interna de los productos, facilitando tanto la gestión del stock como la identificación de los artículos por parte del personal del depósito.

Adicionalmente, se detectó la existencia de productos que no se encuentran publicados en la página web y que tampoco poseen una categoría o tipo asignado dentro del catálogo. Esta información fue obtenida a partir de un archivo Excel provisto por el jefe de depósito, el cual contiene el inventario completo de productos almacenados. A partir de este archivo, se identificaron artículos recientemente incorporados al stock que aún no han sido clasificados, lo que refuerza la necesidad de definir nuevas categorías que permitan integrarlos de manera ordenada y consistente tanto al catálogo como al depósito físico.

4.4. Desarrollo de sistema multi-agente de categorización

La propuesta consiste en diseñar un sistema multi-agente que, a partir de un archivo Excel con los productos de Voltaje S.R.L. (columna “artículo”), sea capaz de asignar un tipo y una categoría a cada producto de forma autónoma, sin partir de un listado de palabras clave previo.

Cada agente tiene un rol específico, coordinado mediante un framework llamado AutoGen, una herramienta que permite orquestar varios modelos de lenguaje y módulos de código para que interactúen entre sí, se repartan tareas y llamen herramientas externas (como lectura de archivos o acceso web) de manera controlada y reproducible.

El sistema trabaja con dos tipos posibles de producto:

- Iluminación
- Materiales eléctricos

Cada producto pertenece obligatoriamente a uno de estos tipos y, dentro de ese tipo, se le asigna una categoría específica que representa el grupo funcional al que pertenece.

Descripción de los agentes

- DataToolAgent

DataToolAgent se encarga de la extracción y estructuración inicial de los datos, así como de la generación del script SQL final. Lee el archivo Excel de origen, toma exclusivamente el contenido de la columna artículo y transforma cada registro en un objeto JSON con la siguiente estructura:

```
{
  "id_articulo_origen": 123,          // identificador original en el Excel (fila, código, etc.)
  "articulo": "TOMA CORRIENTE DOBLE 10A EMBUTIR BLANCA",
  "tipo": null,                      // Iluminación | Materiales eléctricos
  "categoria": null,                 // se completa luego (ej. "Llaves de luz y fichas")
  "estado_clasificacion": "pendiente", // valores posibles: pendiente | clasificado |
  revision_manual
  "fuente_web": null,                // opcional: breve referencia o resumen de
  WebResearchAgent
  "confianza": null                  // opcional: valor cualitativo o numérico de confianza
}
```

A continuación, reúne todos los objetos en un único archivo JSON, por ejemplo data/articulos_pendientes.json, que contiene la lista completa de artículos a clasificar.

Una vez finalizado el proceso de clasificación, genera el script SQL de creación de la base de datos PostgreSQL, incluyendo las tablas articulo, tipo y categoria con sus campos y relaciones.

- OrchestratorAgent

Agente encargado de coordinar la secuencia de pasos sobre cada producto. Recibe el archivo JSON preparado por el DataToolAgent, solicita una propuesta de tipo y categoría al CatalogExpertAgent, evalúa si se requiere información adicional y, en caso necesario, activa al WebResearchAgent. Gestiona las iteraciones entre CatalogExpertAgent y WebResearchAgent y decide cuándo la clasificación es suficientemente confiable o cuándo el registro debe marcarse para revisión humana.

- CatalogExpertAgent

Agente especializado en interpretar las descripciones de los productos y proponer un tipo y una categoría. A partir del texto de la columna "artículo" (y, cuando corresponda, de la información adicional recibida desde WebResearchAgent), identifica patrones lingüísticos, similitudes entre productos y relaciones implícitas, generando etiquetas de tipo y categoría junto con un nivel de confianza. Puede revisar y ajustar sus propuestas cuando el OrchestratorAgent le reenvía un producto con contexto ampliado.

- WebResearchAgent

Agente dedicado a la obtención de información externa. Cuando el OrchestratorAgent lo requiere, realiza búsquedas en la web utilizando el nombre del artículo y posibles datos complementarios (marca, modelo, etc.), y sintetiza los resultados en un breve resumen técnico. Este resumen describe qué es el producto y para qué se utiliza, y se devuelve al OrchestratorAgent para que sea incorporado por el CatalogExpertAgent en una nueva propuesta de clasificación.

4.4.1. Diagrama de ontología

El siguiente diagrama pretende representar, a nivel conceptual, la ontología propuesta del proyecto: cuáles son las entidades principales (datos, archivos y resultados), qué operaciones se consideran relevantes dentro del dominio y cómo se relacionan entre sí mediante dependencias y flujos, dejando explícita la separación entre conceptos y acciones y la trazabilidad desde las entradas hasta los productos de información finales.

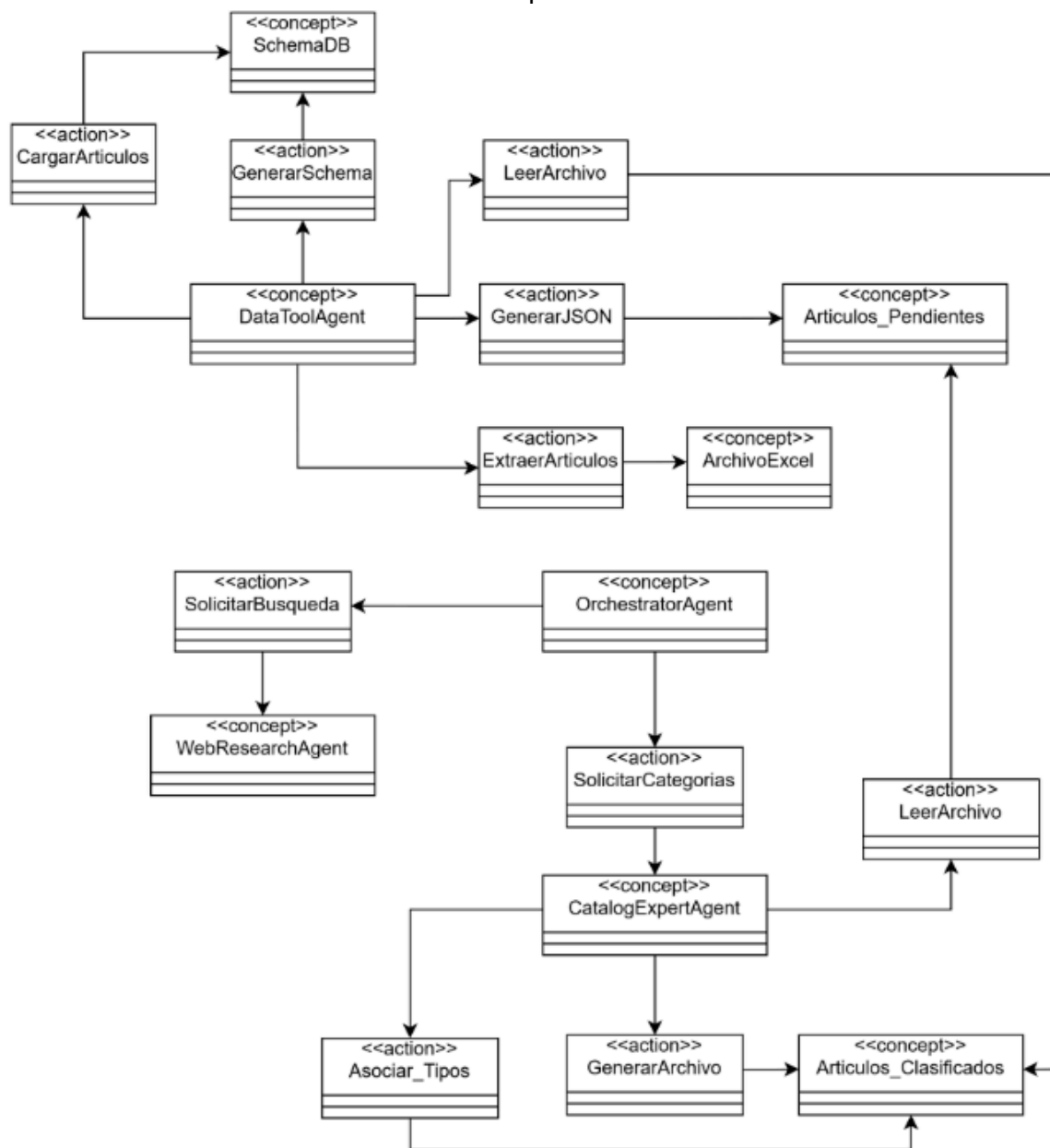


Imagen 1: Diagrama de ontología de la solución

4.4.2. Diagrama de casos de uso (DCU)

El siguiente diagrama de casos de uso pretende visualizar el alcance funcional de la solución y las interacciones entre los distintos actores (internos y externos) y el sistema, mostrando qué objetivos se cubren (importación de datos, preparación de pendientes,

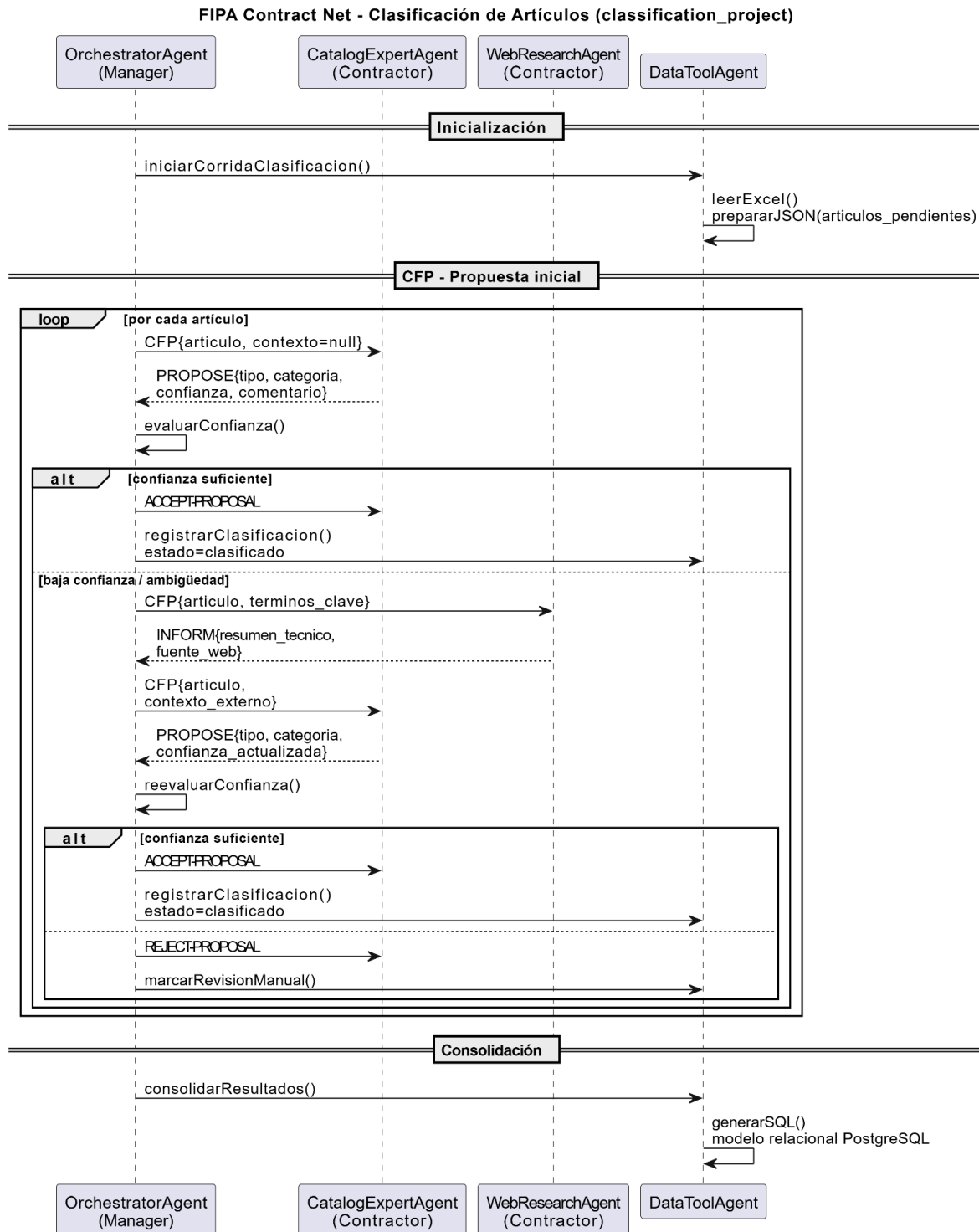


Imagen 3: Diagrama de secuencia de la solución

Flujo de funcionamiento

1. Extracción y generación de JSON

- DataToolAgent lee el archivo Excel de origen.
- Para cada fila, extrae el valor de la columna articulo (y, opcionalmente, un identificador interno).

- Construye un objeto JSON con la estructura definida.
- Agrupa estos objetos en uno o varios archivos (por ejemplo, data/articulos_pendientes.json) dentro de la carpeta data.
- Todos los artículos quedan en estado inicial “pendiente de clasificación”.

2. Primera clasificación

- OrchestratorAgent recorre los objetos JSON almacenados en data.
- Para cada artículo, envía al CatalogExpertAgent el contenido del campo articulo y solicita una primera propuesta de tipo y categoría, junto con un indicador de confianza o estado.

3. Uso de información externa cuando es necesario

- Si la respuesta del CatalogExpertAgent presenta baja confianza o indica ambigüedad, OrchestratorAgent invoca a WebResearchAgent.
- WebResearchAgent realiza consultas en la web sobre el nombre del artículo y devuelve un resumen técnico.
- OrchestratorAgent reenvía al CatalogExpertAgent el artículo junto con este contexto adicional para obtener una nueva propuesta de tipo y categoría.

4. Determinación del estado final del artículo

En función de la propuesta revisada, OrchestratorAgent decide si:

- La clasificación es aceptable y se marca como definitiva, o
- El artículo permanece con un estado de “requiere revisión humana”.
- El resultado final (tipo, categoría, indicador de confianza y estado) se asocia al objeto JSON correspondiente.

5. Consolidación de resultados y generación de base de datos

- Una vez procesados todos los artículos, OrchestratorAgent entrega a DataToolAgent el conjunto de artículos clasificados (incluyendo tipos, categorías y estados).

DataToolAgent utiliza esta información para:

- Derivar el conjunto de tipos y categorías únicas.
- Generar el código SQL de creación de la base de datos en PostgreSQL, que incluirá las tablas tipo, categoria y articulo, con sus campos y relaciones.
- El código SQL se guarda, por ejemplo, en un archivo schema_voltaje.sql, listo para ser ejecutado en el motor PostgreSQL.

6. Resultados

Este proyecto implementa un flujo simple y reproducible para la clasificación de artículos y la persistencia de los resultados en una base de datos PostgreSQL. A lo largo de todo el proceso se utilizan archivos JSON como artefactos intermedios, lo que permite inspeccionar y auditar el estado de los datos en cada etapa sin depender directamente de la base de datos.

El flujo completo consta de tres pasos principales: extracción, clasificación y carga a base de datos.

El paso 1 (extracción) se realiza mediante el módulo dataTool.py, utilizando la función extract_articles. En esta etapa se toma el archivo Excel exportado desde el ERP (por ejemplo, input/productos.xlsx) y se genera el archivo data/articulos_pendientes.json. Cada

registro del JSON contiene el identificador de origen (`id_articulo_origen`) y el campo `articulo`, que almacena el nombre o descripción textual del producto tal como proviene del sistema de origen. Los campos `tipo` y `categoria` se inicializan en `null` y el registro se marca con `estado_clasificacion = "pendiente"`. En esta fase no se realiza ninguna clasificación; únicamente se construye el dataset de entrada en el formato acordado.

Este paso se ejecuta con el siguiente comando:

```
uv run dataTool.py extract --excel input/productos.xlsx --output data/articulos_pendientes.json
```

El paso 2 (clasificación) es orquestado por el archivo `main.py`. Este módulo lee el archivo `data/articulos_pendientes.json`, recorre los artículos uno por uno y llama al agente `CatalogExpertAgent`, definido en `agents.py`, pasando como entrada el identificador del artículo y el contenido del campo `articulo`. A partir de la respuesta del agente, `main.py` construye el JSON de salida respetando el formato establecido, conservando el campo `articulo` sin modificaciones e incorporando los campos `tipo`, `categoria`, `estado_clasificacion`, `confianza` y `comentario`. El resultado del proceso se persiste en el archivo `data/articulos_clasificados.json`. En este flujo, el estado `pendiente` sólo existe en el JSON de entrada, mientras que los registros de salida quedan en estado `clasificado` o `pendiente_revision`.

Este paso se ejecuta mediante el comando:

```
uv run main.py
```

La lógica de clasificación se encuentra encapsulada en `agents.py`. El agente utilizado es `CatalogExpertAgent`, el cual envuelve internamente un agente LLM de `AutoGen` denominado `catalog_expert`. Este agente recibe exclusivamente el texto del campo `articulo` y devuelve una propuesta de clasificación siguiendo reglas de negocio predefinidas, como la restricción a dos valores posibles de `tipo` y la validación de categorías permitidas. Antes de devolver el resultado, `CatalogExpertAgent` valida y normaliza la respuesta del modelo, y en los casos en que no se obtiene una clasificación válida, marca el registro como `pendiente_revision` con un comentario explicativo.

El paso 3 (carga a base de datos) se realiza nuevamente en `dataTool.py`, mediante la función `generate_sql_from_json`. En esta etapa se lee el archivo `data/articulos_clasificados.json` y se genera un script SQL que contiene tanto la definición del esquema como los datos a insertar.

El modelo relacional final queda definido como `tipo (1) → categoria (N) → articulo (N)`: la tabla `categoria` almacena la clave foránea `id_tipo`, mientras que la tabla `articulo` referencia únicamente a `categoria` mediante `id_categoria` y almacena el nombre o descripción del producto en el campo `articulo`.

De este modo, el tipo de un artículo se obtiene exclusivamente mediante operaciones `JOIN` (`articulo → categoria → tipo`). Para asegurar la consistencia de la carga, los artículos que tengan `categoria = null` en el JSON no se insertan en la tabla `articulo`.

La generación del SQL se realiza con el siguiente comando:

```
uv run dataTool.py sql --json data/articulos_clasificados.json --output sql/voltaje_schema_and_data.sql
```

Una vez generado el script SQL, se levanta la infraestructura de base de datos definida en

docker-compose.yaml. Esto permite ejecutar el script generado y consultar los resultados desde PostgreSQL.

Para ello, a continuación del último comando se ejecuta:

```
docker compose up -d
```

En conjunto, los resultados demuestran que el sistema permite clasificar artículos a partir de descripciones textuales, mantener un pipeline trazable mediante JSON intermedios y persistir los datos en un modelo relacional normalizado, listo para su posterior consulta y explotación.

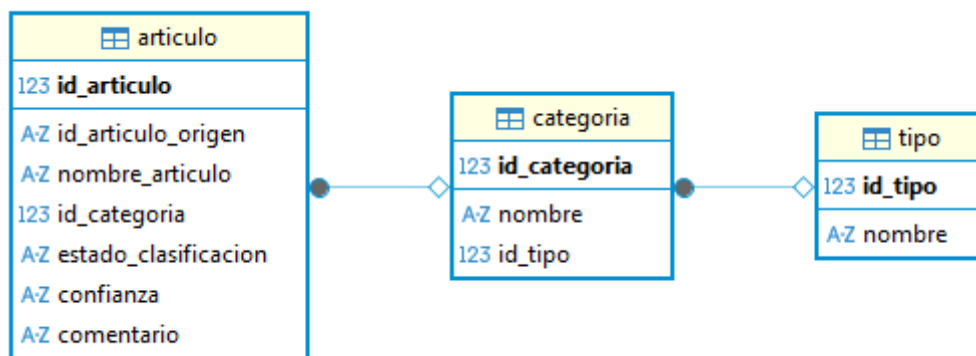


Imagen 4: Modelo relacional desarrollado

Las categorías obtenidas como resultado del proceso de clasificación corresponden al catálogo de productos analizado y se agrupan bajo dos tipos principales: Iluminación y Materiales eléctricos. A continuación se presenta el listado de categorías identificadas, las cuales representan familias funcionales claras y reutilizables dentro del dominio del negocio:

1. Térmicas y disyuntores
2. Comando y señalización
3. Llaves de luz y fichas
4. Herramientas
5. Apliques techo
6. Línea hierro
7. Apliques pared
8. LED
9. Spots embutidos
10. Cables
11. Gabinetes eléctricos
12. Colgantes
13. Línea aluminio
14. Línea PVC
15. Iluminación exterior
16. Puesta a tierra
17. Iluminación interior
18. Plafones
19. Emergencia y seguridad

- 20. Ventilación
- 21. Lámparas de pie y veladores
- 22. Apliques exterior
- 23. Accesorios de iluminación
- 24. Cajas
- 25. Timbres y porteros
- 26. Conectores
- 27. Pilas y baterías
- 28. Adhesivos

A continuación se presenta la estructura del proyecto, donde se muestran los principales archivos y directorios que lo componen:

```
classification_project/  
├── agents.py  
├── data/  
│   ├── articulos_clasificados.json  
│   └── articulos_pendientes.json  
├── dataTool.py  
├── docker-compose.yaml  
├── input/  
│   └── productos.xlsx  
├── main.py  
├── pyproject.toml  
├── README.md  
├── sql/  
│   └── voltaje_schema_and_data.sql  
└── uv.lock
```

Imagen 5: Estructura del sistema multi-agente para la categorización de productos

Para que el agente pudiera asociar correctamente los productos y crear las categorías, se tomó como punto de partida el conjunto de categorías ya definidas en el catálogo digital existente de la empresa, garantizando coherencia entre la clasificación automática y la estructura previamente utilizada. Cuando un producto no encajaba de manera adecuada en ninguna de las categorías existentes, el sistema permitía la creación de una nueva categoría, propuesta de forma autónoma por el CatalogExpertAgent a partir del análisis semántico de la descripción del artículo. De este modo, el proceso combinó reutilización del conocimiento previo con la capacidad de ampliación dinámica del catálogo.

4.5. Desarrollo de sistema generador de ventas

4.5.1. Objetivo del proyecto

El proyecto `synthetic_sales_generator` tiene como objetivo generar un conjunto completo de datos sintéticos de ventas y cargarlos en una base de datos MySQL para su posterior

consulta y análisis. El sistema simula el funcionamiento de un escenario real de ventas, incluyendo clientes, ventas y detalle de ventas, manteniendo coherencia entre los datos generados.

El propósito principal del proyecto es disponer de un dataset de ventas realista sin utilizar datos reales, que pueda usarse para prácticas académicas, análisis de datos y pruebas de consultas, sin depender de información sensible ni de sistemas productivos.

4.5.2. Arquitectura general del sistema

Por un lado, PostgreSQL se usa únicamente como fuente del catálogo de productos. En esta base existen las tablas articulo, categoria y tipo, que provienen del proyecto previo `classification_project`, donde los productos ya fueron clasificados y organizados.

Por otro lado, MySQL es la base de datos destino donde se almacenan los datos sintéticos generados. Allí se guardan los clientes, las ventas y el detalle de cada venta. Esta separación permite mantener el catálogo de productos independiente de los datos de ventas, imitando una arquitectura común en sistemas reales.

La infraestructura de MySQL se levanta mediante Docker Compose, lo que permite crear, reiniciar o eliminar la base de datos de forma simple y controlada.

4.5.3. Funcionamiento

Generación de clientes

La generación de clientes se realiza como una etapa independiente del proceso. Para ello, el proyecto cuenta con un agente que utiliza la API de OpenAI para generar clientes con datos completos y coherentes, como nombre, apellido, DNI, teléfono, correo electrónico y domicilio.

Se indica explícitamente que los nombres y apellidos deben ser típicos de Argentina. El agente trabaja por lotes y valida que cada cliente tenga la estructura correcta. El resultado de esta etapa es un archivo JSON (`data/clientes.json`), que funciona como entrada para la carga posterior en la base MySQL.

Extracción del catálogo de productos

Antes de generar las ventas, el sistema se conecta a la base PostgreSQL para leer el catálogo de productos. En esta etapa se obtiene, para cada artículo, su identificador, nombre, categoría y tipo. Además, el sistema intenta detectar si el artículo tiene algún precio definido en la base; si no lo tiene, ese valor se generará luego dentro de un rango configurado.

Esta etapa define el conjunto de productos que pueden aparecer en las ventas y garantiza que todos los artículos utilizados existan realmente en el catálogo clasificado.

Generación de ventas y detalles

La generación de ventas se realiza simulando un año completo de actividad comercial. Para cada día se crea una cantidad variable de ventas, y cada venta se asocia a un cliente y contiene varias líneas de detalle. En esta etapa se generan las fechas de venta, los clientes involucrados, las cantidades de artículos, los precios unitarios y los totales, manteniendo coherencia entre todos estos valores.

Un aspecto clave de esta etapa es la forma en que se seleccionan los artículos dentro de

cada venta. En lugar de elegirlos de manera uniforme, el sistema aplica un sesgo de popularidad, implementado en el módulo `generate_sales.py`. Al inicio del proceso se asigna a cada artículo un peso de popularidad, de modo que algunos productos tengan muchas más probabilidades de aparecer que otros. Este enfoque refleja un comportamiento común en escenarios reales, donde unos pocos artículos concentran gran parte de las ventas y la mayoría se vende con menor frecuencia. Estos pesos se mantienen constantes durante todo el año, lo que hace que los mismos productos tiendan a dominar las ventas a lo largo del tiempo.

Para evitar que la popularidad dependa del orden original de los artículos, los rangos de popularidad se asignan de forma aleatoria. Luego, en cada venta, los artículos se seleccionan mediante un muestreo ponderado sin reemplazo. Esto significa que dentro de una misma venta no se repite el mismo artículo, pero que los artículos más populares tienen mucha más probabilidad de ser elegidos. Como en cada venta se seleccionan varios artículos usando esta misma lógica, los productos más vendidos tienden a aparecer juntos de forma natural, sin necesidad de reglas explícitas. Como resultado, el dataset final presenta pocos artículos con muchas apariciones, muchos artículos con pocas ventas y combinaciones de productos que se repiten de manera consistente durante el año, lo que lo hace más adecuado para análisis posteriores.

Carga de datos en MySQL

Una vez generados los datos, el sistema crea las tablas destino en MySQL: `cliente`, `venta` y `detalle_venta`. Primero se insertan los clientes, luego las ventas y finalmente los detalles de cada venta.

La tabla `detalle_venta` no solo almacena el identificador del artículo, sino también su nombre, categoría y tipo. Esta decisión permite realizar consultas y análisis directamente sobre MySQL sin necesidad de conectarse nuevamente a PostgreSQL, facilitando el trabajo posterior con los datos.

Todo el proceso de creación de tablas, carga de clientes, generación de ventas e inserción de datos se ejecuta desde un único script principal, lo que simplifica la ejecución completa del proyecto.

4.5.4. Resultado final

Como resultado del proyecto, se obtiene una base de datos MySQL completamente cargada con clientes, ventas y detalles de ventas, basada en un catálogo real previamente clasificado. El dataset generado es consistente, fácil de regenerar y adecuado para análisis de ventas, consultas SQL y prácticas académicas.

```

synthetic_sales_generator/
├── __pycache__/
├── data/
├── mysql-data/
├── venv/
├── .gitignore
├── agente_clientes.py
├── config.py
├── db.py
├── docker-compose.yaml
├── env.example.txt
├── generate_clients.py
├── generate_sales.py
├── insert_sales.py
├── load_articles.py
├── load_clients.py
├── main.py
├── mysql_schema.py
└── requirements.txt

```

Imagen 6: Estructura del proyecto generador de ventas

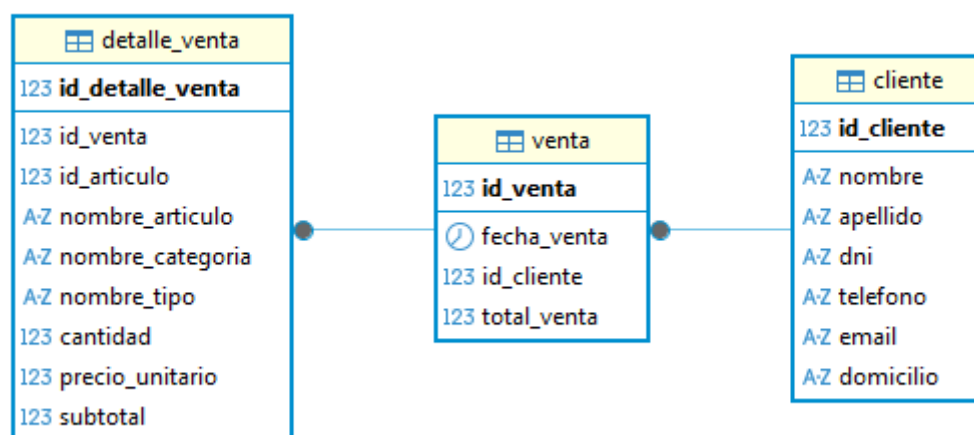


Imagen 4: Modelo de datos generado

4.6. Desarrollo de pipeline ETL

La presente propuesta describe el diseño de un pipeline ETL (Extracción, Transformación y Carga) para preparar los datos de ventas de la empresa para su análisis. Como insumo se utilizarán las ventas correspondientes al período enero–diciembre de 2025, provistas por la empresa como fuente de datos.

Los artículos incluidos en las ventas contarán previamente con una clasificación por tipo y categoría obtenida mediante agentes desarrollados en Python. A partir de ese resultado, este trabajo se enfocará únicamente en el procesamiento de los datos dentro del pipeline ETL.

4.6.1. Alcance del pipeline ETL

El pipeline ETL tendrá como objetivo organizar y preparar los datos de ventas para su uso analítico. El flujo completo será orquestado mediante Apache Airflow, permitiendo definir, ejecutar y monitorear cada etapa del proceso de forma controlada. El pipeline procesa la totalidad de las ventas generadas sintéticamente del año 2025.

4.6.2. Objetivo general

Diseñar un pipeline ETL que permita ingerir, transformar y almacenar los datos de ventas de la empresa correspondientes al año 2025, dejándolos preparados para su posterior análisis.

4.6.3. Objetivos específicos

- Ingerir los datos de ventas provistos por la empresa mediante Airbyte y almacenarlos sin modificaciones en una base de datos PostgreSQL como capa Bronze.
- Aplicar transformaciones sobre los datos ingeridos mediante dbt y lógica desarrollada en Python, conformando la capa Silver.
- Cargar los datos transformados en un data warehouse PostgreSQL que represente la capa Gold.
- Orquestrar el flujo completo del pipeline utilizando Apache Airflow.
- Ejecutar la infraestructura del pipeline mediante Docker, asegurando portabilidad y reproducibilidad.
- Dejar los datos finales preparados para su uso en algoritmos de minería de datos y análisis posteriores.

Ingesta de datos (Bronze)

La etapa de ingesta consistirá en cargar los datos de ventas en una base de datos PostgreSQL utilizada como punto de entrada del pipeline. La ingesta se realizará mediante Airbyte, ejecutado dentro de un entorno Docker de manera local, permitiendo extraer los datos desde la fuente provista por la empresa y cargarlos sin aplicar transformaciones.

Esta etapa corresponderá a la capa Bronze de la arquitectura Medallion, ya que su función es conservar los datos en su estado original como referencia del origen.

Transformación de datos (Silver)

En la capa Silver se trabajará sobre los datos provenientes de la capa Bronze. Las transformaciones se realizarán utilizando dbt, complementado con lógica en Python cuando sea necesario, para limpiar, validar y generar nuevos datos a partir de los existentes.

Esta capa corresponderá a Silver porque los datos dejan de ser crudos y pasan a estar procesados y enriquecidos, aunque todavía no sean el resultado final de análisis.

Carga en el data warehouse (Gold)

Los datos transformados se cargarán en una segunda base de datos PostgreSQL que funcionará como data warehouse. Esta base representará la capa Gold de la arquitectura Medallion y será alimentada a partir de los modelos finales definidos en dbt.

Esta etapa será Gold porque contendrá los datos finales, consolidados y listos para ser utilizados directamente en análisis y en la aplicación de algoritmos de minería de datos como Apriori, FP-Growth o Eclat. El desarrollo de estos análisis se abordará en una etapa posterior del proyecto.

4.6.4. Arquitectura de la solución propuesta

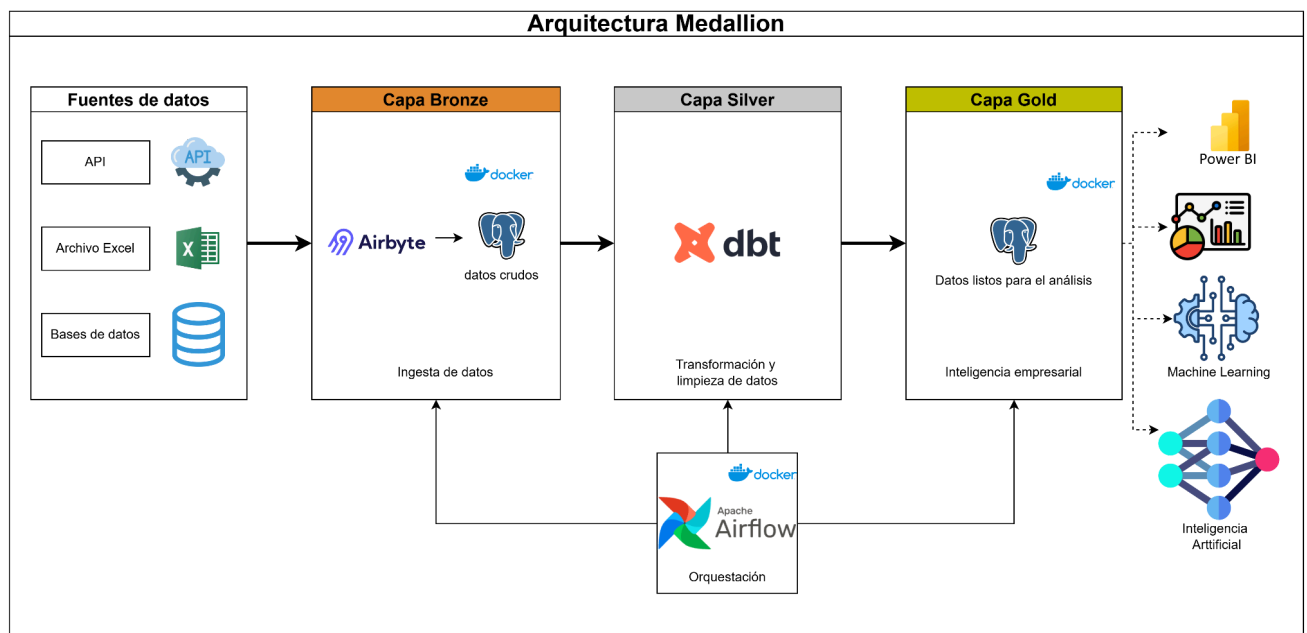


Imagen 7: Arquitectura del pipeline propuesto

4.6.5. Desarrollo del pipeline de datos

El proyecto `voltaje_etl_pipeline` implementa un pipeline de datos completo utilizando Docker Compose, Apache Airflow y dbt, siguiendo la arquitectura Medallion. El objetivo principal es construir, de forma reproducible y automatizada, distintas capas de datos dentro de un Data Warehouse en PostgreSQL, sin necesidad de ejecutar comandos manuales dentro de los contenedores. El pipeline está pensado para trabajar sobre datos de ventas y clientes obtenidos del dataset generado anteriormente en el proyecto `Synthetic_Sales_Generator`, permitiendo distintos usos según el nivel de transformación de la información.

La infraestructura se levanta mediante un único archivo `docker-compose.yaml`, que define todos los servicios necesarios. Se utilizan dos instancias de PostgreSQL: una para la metadata de Airflow y otra para el Data Warehouse. Esta última expone el puerto hacia el host y se inicializa de forma automática con los schemas necesarios: `bronze`, `silver`, `gold`, `datamining` y `silver_datamining`. Además, se incluye Redis como sistema de colas para el ejecutor Celery de Airflow. Todos los servicios de Airflow utilizan una misma imagen custom, lo que simplifica el mantenimiento y evita inconsistencias entre componentes.

La imagen de Airflow se construye a partir de una imagen oficial y agrega dbt dentro de un entorno virtual aislado. Esto permite que Airflow y dbt convivan sin conflictos de dependencias. El diseño evita modificar el entrypoint original de Airflow, lo que asegura una inicialización correcta del sistema y previene errores comunes relacionados con la carga de módulos. Durante el arranque, un servicio de inicialización ejecuta automáticamente las migraciones de Airflow y crea el usuario administrador, dejando el entorno listo para su uso.

El Data Warehouse se alimenta siguiendo el enfoque Medallion. La capa Bronze contiene los datos crudos cargados por Airbyte desde sistemas externos y no es responsabilidad de dbt. A partir de esta capa, dbt construye las capas Silver, Gold y Datamining. En Silver se realizan tareas de limpieza básica, tipado de columnas y normalización mínima, sin agregar métricas. Gold contiene tablas agregadas orientadas a reporting y dashboards, mientras que Datamining genera una tabla plana pensada específicamente para análisis de canastas

de compra.

La orquestación del pipeline se realiza mediante DAGs de Airflow. El DAG principal llamado `voltage_medallion_etl.py` verifica primero que las tablas Bronze existan en el Data Warehouse antes de ejecutar cualquier transformación. Una vez confirmada la disponibilidad de los datos, se ejecutan de forma ordenada los modelos de dbt para Silver, Datamining y Gold, junto con tests de calidad en la capa final. De esta manera, cada ejecución garantiza que las capas superiores se construyan siempre a partir de datos consistentes y actualizados.

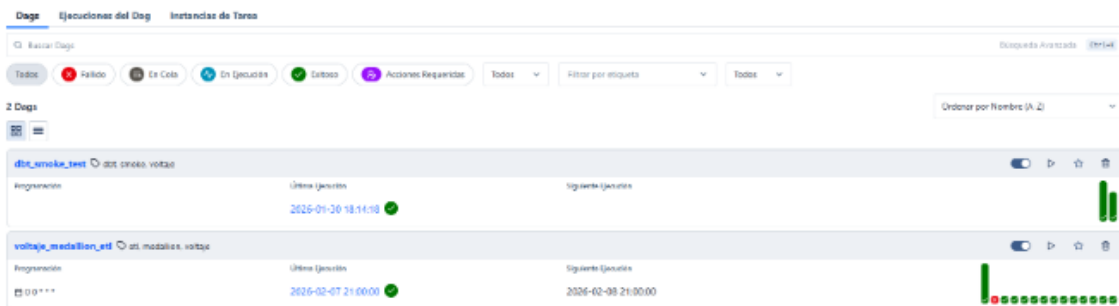


Imagen 6: DAGs generados

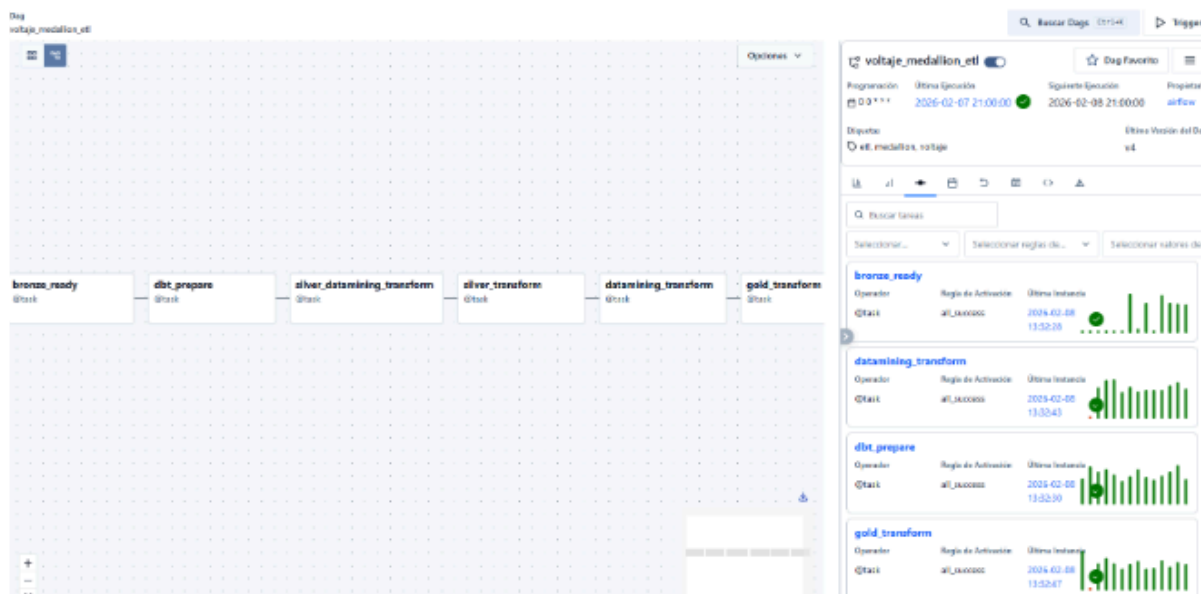


Imagen 4: Vista de DAG ejecutado desde Airflow UI

4.6.6. Capas del Data Warehouse

El Data Warehouse del proyecto fue diseñado siguiendo una arquitectura por capas que permite organizar el flujo de datos de manera progresiva, garantizando trazabilidad, calidad, reutilización analítica y separación clara de responsabilidades. Para ello, se adoptó el enfoque Medallion, compuesto por las capas Bronze, Silver y Gold, y se extendió con una capa adicional de Datamining, orientada específicamente a la aplicación de modelos de minería de datos.

Cada capa cumple un rol bien definido dentro del pipeline, tanto desde el punto de vista técnico como analítico, y está pensada para distintos tipos de consumidores de información.

4.6.6.1. Capa Bronze

La capa Bronze representa el punto de entrada de los datos al Data Warehouse. En esta capa se almacenan los datos crudos tal como son obtenidos desde los sistemas fuente, sin aplicar transformaciones ni validaciones. Su principal objetivo es conservar la información original para fines de trazabilidad, auditoría y depuración de errores a lo largo del pipeline.

Elementos de la capa Bronze:

- Datos sin procesar (raw data).
- Estructura idéntica a la fuente de origen.
- Ausencia de reglas de negocio.
- Tablas físicas.
- Carga completa de datos (truncate & insert).

4.6.6.2. Capa Silver

La capa Silver actúa como una capa intermedia cuyo propósito es limpiar, estandarizar y normalizar los datos provenientes de Bronze. En esta etapa se aplican transformaciones técnicas que permiten corregir inconsistencias, homogeneizar formatos y preparar la información para su posterior consumo analítico, sin introducir lógica de negocio.

Esta capa constituye la base confiable sobre la cual se construyen tanto la capa Gold como la capa Datamining.

Elementos de la capa Silver:

- Datos limpios y estandarizados.
- Eliminación de valores nulos inválidos.
- Normalización de formatos y tipos de datos.
- Enriquecimiento básico de datos.
- Tablas físicas.
- Carga completa de datos.

4.6.6.3. Capa Datamining

La capa Datamining se incorpora como una extensión del enfoque Medallion con el objetivo de soportar análisis avanzados y la aplicación de modelos de minería de datos. Esta capa consume exclusivamente datos ya depurados desde Silver y los reorganiza en una estructura adecuada para técnicas de análisis de asociaciones, como Apriori, FP-Growth y Eclat.

El diseño de esta capa se basa en una única tabla transaccional, donde cada fila representa un producto dentro de una transacción, facilitando el análisis de patrones de compra conjunta sin afectar las capas destinadas al reporting.

Elementos de la capa Datamining:

- Datos limpios provenientes de Silver.
- Estructura transaccional, donde cada fila contiene el detalle de la venta de un producto.

- Preparada para Market Basket Analysis.
- Tablas físicas optimizadas para notebooks.
- Sin lógica de negocio.
- Separada del entorno de reporting.

4.6.6.4. Capa Gold

La capa Gold contiene los datos finales listos para ser consumidos por herramientas de análisis y visualización. En esta etapa se integran los datos, se aplican reglas de negocio y se generan métricas y agregaciones que responden a necesidades específicas del negocio. Esta capa está diseñada para análisis descriptivo y reporting, y no para la aplicación de modelos de minería de datos, ya que prioriza la claridad y estabilidad de los indicadores.

Elementos de la capa Gold:

- Datos consolidados y listos para negocio.
- Aplicación de reglas de negocio.
- Integración de múltiples entidades.
- Agregaciones y métricas.
- Vistas o tablas analíticas.
- Modelado orientado a consumo (esquema estrella o tablas planas).

4.6.7. Resultados finales

Como resultado del proyecto, se obtiene un Data Warehouse en PostgreSQL completamente estructurado, con múltiples capas de datos claramente separadas según su nivel de procesamiento. La capa Bronze contiene los datos tal como fueron cargados por Airbyte, incluyendo metadatos técnicos de la ingesta. La capa Silver ofrece tablas limpias y tipadas, listas para ser reutilizadas sin problemas de calidad básica.

La capa Gold produce tablas agregadas que permiten analizar ventas diarias, mensuales, comportamiento de clientes, artículos más vendidos y facturación, entre otros indicadores. Estas tablas están pensadas para ser consumidas directamente por herramientas de visualización o reporting, sin necesidad de transformaciones adicionales. Cada ejecución del pipeline reconstruye estas tablas para reflejar siempre el estado más reciente de los datos.

Por su parte, la capa Datamining genera una tabla plana con una fila por transacción y producto, sin precios ni datos de clientes. Este diseño permite que la tabla sea utilizada directamente desde notebooks para aplicar algoritmos de Market Basket Analysis, como Apriori o FP-Growth. En conjunto, el pipeline produce un entorno de datos listo tanto para análisis descriptivo como para análisis más avanzados.

transaction_items_datamining	
A-Z	transaction_item_id
123	transaction_id
123	product_id
A-Z	product_name
A-Z	product_category
A-Z	product_type
123	quantity
🕒	transaction_date

Imagen 8: Dataset generado para análisis en Schema Datamining

4.7. Análisis de patrones de asociación

En la presente sección del informe se documenta el desarrollo de un análisis de Market Basket Analysis aplicado a las ventas sintéticas generadas y previamente normalizadas, con el objetivo de obtener información útil para el diseño del layout del depósito. El enfoque busca dejar de lado decisiones basadas sólo en la experiencia o la intuición, y apoyarse en datos concretos que permitan identificar qué productos y categorías suelen comprarse juntos. A partir de estos patrones, se propone una organización física que reduzca recorridos innecesarios y mejore los tiempos de picking y reposición.

Para construir el dataset se extrajo información desde PostgreSQL, utilizando la base de datos llamada dw y la tabla datamining.transaction_items_datamining. Se recuperaron los campos necesarios para el análisis, como transacción, producto, nombre, categoría, tipo y cantidad, considerando únicamente registros con valores de cantidad mayores a cero. Luego se realizó una limpieza básica para asegurar consistencia, normalizando los nombres de productos y descartando registros incompletos. Como validación inicial, se analizó el volumen de datos y la estructura de las compras, observando la cantidad de transacciones, la variedad de productos y el tamaño promedio de las cestas, verificando que existiera suficiente repetición para encontrar patrones relevantes.

Una vez validado el dataset, se construyeron las cestas de compra. Cada identificador de transacción se consideró una compra individual y se transformó en un conjunto de productos presentes en esa venta. El análisis se basó en un esquema de presencia o ausencia, ya que el objetivo es detectar relaciones del tipo cuando aparece un producto, suele aparecer otro. Las transacciones con un solo ítem fueron descartadas, dado que no aportan combinaciones útiles. Como apoyo al análisis, se generaron estadísticas descriptivas que permitieron observar las combinaciones más frecuentes y tener una primera visión del comportamiento de compra.

A continuación se definieron los criterios necesarios para aplicar los algoritmos de asociaciones. Se estableció un umbral mínimo de soporte expresado como cantidad de transacciones, el cual luego se transformó en una proporción para facilitar la comparación entre resultados. También se filtraron productos con muy baja frecuencia para mejorar la eficiencia del procesamiento sin afectar la calidad del análisis.

Métricas utilizadas para el análisis:

support = cantidad de transacciones donde aparece el conjunto / total de transacciones

$\text{confidence} = \text{frecuencia de aparición conjunta de A y B} / \text{frecuencia de aparición de A}$

$\text{lift} = \text{confidence} / \text{frecuencia esperada de B}$

Las métricas obtenidas permiten interpretar la relevancia de las asociaciones detectadas entre productos. El support indica qué tan frecuente es un conjunto de productos dentro del total de transacciones, por lo que refleja su peso real en el volumen de ventas. La confidence mide la probabilidad de que un producto B aparezca en una transacción dado que ya está presente el producto A, lo que ayuda a entender la fuerza direccional de la relación entre ambos. Por su parte, el lift compara esa relación con lo que ocurriría por azar, indicando si la asociación entre A y B es más fuerte, igual o más débil que la esperada si fueran independientes, siendo valores mayores a uno una señal de afinidad real entre los productos.

Con las cestas ya definidas, se prepararon los datos para la aplicación de los algoritmos Apriori, FP-Growth y Eclat. Para los dos primeros se utilizó una matriz de transacciones por producto en formato binario, mientras que para Eclat se empleó un enfoque vertical basado en conteos de co-ocurrencia. Se ejecutaron los tres algoritmos utilizando los mismos parámetros, con el objetivo de comparar resultados de manera consistente.

En la etapa de modelado se obtuvieron conjuntos frecuentes y reglas de asociación, junto con las métricas de soporte, confianza y lift. También se midió el tiempo de ejecución de cada algoritmo. Dado que el objetivo del análisis es apoyar el diseño del layout, se priorizaron reglas simples entre pares de productos, por ser las más fáciles de interpretar y aplicar en un contexto operativo. Si bien los tres algoritmos produjeron el mismo conjunto de reglas bajo los umbrales definidos, se observaron diferencias importantes en el tiempo de ejecución, por lo que se seleccionó automáticamente el algoritmo Eclat para continuar con el análisis, debido a su mejor desempeño.

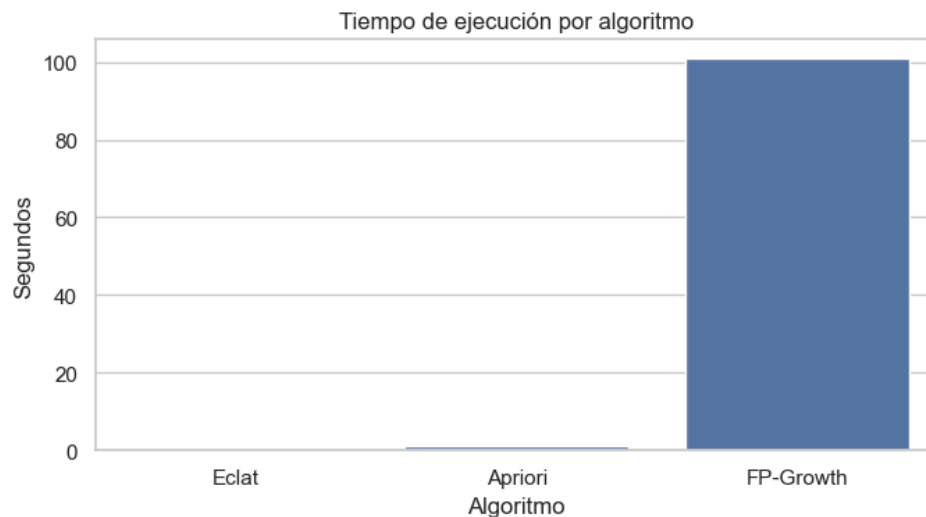


Imagen 9: Comparación de performance

A partir del algoritmo seleccionado se generó una tabla final con las relaciones entre productos, que incluye para cada par sus métricas y un nivel de proximidad recomendado clasificado como alto, medio o bajo. Esta clasificación se definió mediante reglas sencillas basadas en los valores de lift y confianza, con el objetivo de traducir resultados estadísticos en recomendaciones claras para el layout del depósito. Como complemento, se generaron visualizaciones que permitieron evaluar la distribución general de las métricas y facilitar la interpretación de los resultados.

Las métricas permitieron interpretar el valor real de las asociaciones detectadas y cómo usarlas en el resultado final. El support indicó que las combinaciones de productos aparecen en una proporción baja del total de transacciones, lo que muestra que no son patrones masivos pero sí existentes. La confidence reflejó que, cuando un producto A está presente en una venta, el producto B aparece con una frecuencia relativamente alta, lo que señala una relación consistente y útil para pensar ubicaciones cercanas. Por su parte, el lift cercano a 1 mostró que muchas asociaciones no aumentan de forma significativa la probabilidad de compra conjunta más allá de lo esperado, por lo que no todas justifican cambios fuertes en el layout.

A partir de esta interpretación, los resultados se aplicaron de manera equilibrada al diseño del layout, priorizando las asociaciones con mayor confidence y mejor lift, y combinándolas con información de rotación de productos. Esto permitió evitar reorganizaciones basadas solo en co-ocurrencias débiles y enfocar los cambios en aquellos casos donde la cercanía física podía generar un impacto operativo real.

Además del análisis de co-ocurrencia, se incorporó la rotación de productos como una dimensión adicional. Para cada producto se calculó su rotación tanto por cantidad de transacciones como por unidades vendidas, y se clasificó en niveles alto, medio o bajo utilizando percentiles. Esta información se combinó con las reglas de asociación para construir un puntaje que prioriza aquellas relaciones que no solo se compran juntas, sino que además tienen un alto movimiento operativo.

Finalmente, el trabajo se consolidó en tablas persistentes dentro de PostgreSQL para facilitar su uso posterior. Se generó una tabla de recomendaciones de layout por producto, que incluye rotación, nivel de co-ocurrencia, prioridad de ubicación y una sugerencia general de posición dentro del depósito. También se aplicó el mismo enfoque a nivel de categorías, generando tablas que resumen afinidad entre categorías y prioridades de ubicación. Estas salidas permiten abordar el rediseño del depósito tanto a nivel de productos individuales como por zonas, facilitando una implementación gradual y su validación con el personal del depósito y las restricciones físicas existentes.

	product_category	tipo	frecuencia_transacciones	unidades_totales_vendidas	nivel_rotacion	nivel_coocurrencia	prioridad_layout	ubicacion_sugerida	categorias_cercanas_sugeridas
4	LED	Iluminación	27518	259111	Alta	Alta	Muy Alta	Próxima a la entrada (máxima accesibilidad)	Puesta a tierra, Apliques pared, Llaves de luz y fichas
0	Apliques pared	Iluminación	49028	451186	Alta	Media	Alta	Cerca de la entrada	Llaves de luz y fichas, Comando y señalización, Cables
1	Llaves de luz y fichas	Materiales eléctricos	37749	365310	Alta	Media	Alta	Cerca de la entrada	Apliques pared, Comando y señalización, Cables
2	Comando y señalización	Materiales eléctricos	31678	313727	Alta	Media	Alta	Cerca de la entrada	Apliques pared, Llaves de luz y fichas, Cables
3	Cables	Materiales eléctricos	31114	305132	Alta	Media	Alta	Cerca de la entrada	Apliques pared, Llaves de luz y fichas, Comando y señalización
5	Herramientas	Materiales eléctricos	27226	239868	Alta	Media	Alta	Cerca de la entrada	Apliques pared, Llaves de luz y fichas, Comando y señalización
6	Gabinets eléctricos	Materiales eléctricos	18961	158280	Media	Alta	Alta	Cerca de la entrada	Puesta a tierra, Apliques pared, Llaves de luz y fichas
7	Térmicas y disyuntores	Materiales eléctricos	15417	132925	Media	Alta	Alta	Cerca de la entrada	Pilas y baterías, Apliques pared, Llaves de luz y fichas
9	Línea PVC	Materiales eléctricos	9928	81741	Media	Alta	Alta	Cerca de la entrada	Pilas y baterías, Apliques pared, Llaves de luz y fichas

Imagen 10: Tabla generada y persistida en base de datos

4.8. Diseño de Layout optimizado para el depósito

El diseño del layout se implementa como una visualización interactiva del depósito, donde cada estantería se representa como un rectángulo dibujado sobre un plano de referencia. El objetivo es transformar un plan de organización (prioridades y cercanías recomendadas entre categorías) en una representación física clara, que permita validar decisiones y comunicar la propuesta al área operativa. La interacción se centra en navegar el plano

(zoom y desplazamiento) y en consultar información puntual de cada estante al pasar el cursor.

La decisión técnica principal es que el layout se construye con coordenadas definidas manualmente, en lugar de intentar detectar estantes automáticamente a partir de la imagen.

Cada estante se describe mediante un rectángulo con coordenadas normalizadas (proporciones entre 0 y 1 respecto del ancho y alto del plano), lo que permite mantener las proporciones aunque cambie el tamaño de la imagen y ajustar el dibujo con precisión cuando sea necesario. La imagen del plano se utiliza para respetar escala y proporciones, pero la geometría de estanterías proviene de esa definición manual.

Además del dibujo de estantes, se incorporan zonas guía dentro del plano (por ejemplo, áreas funcionales como entrada, salida, recepción u escaleras), que no se comportan como estantes sino como referencias visuales. Estas zonas ayudan a interpretar el espacio y sirven como puntos de referencia para ordenar y clasificar estanterías en función de su ubicación dentro del depósito.

Para asignar categorías a los estantes, el sistema consume un plan de categorías que contiene, para cada categoría, su prioridad, su ubicación sugerida (por ejemplo, más accesible o más alejada) y, cuando existe, recomendaciones de cercanía con otras categorías. Ese plan proviene de una fuente persistente (base de datos) o de una copia exportada a un formato portable, de forma que la visualización pueda ejecutarse aun sin conexión a la fuente original. Si no se dispone del plan, el sistema mantiene un comportamiento tolerante: deja estantes sin asignación para no interrumpir la generación del plano.

La numeración de estantes se define de forma consistente y operativa. En lugar de numerar en el orden en que fueron dibujados, se ordenan por cercanía a un punto de referencia, en este caso desde la entrada en la planta baja y desde las escaleras en el primer piso, y se asigna una numeración que refleja el recorrido esperado: primero los estantes más próximos, luego los intermedios y finalmente los más alejados. Esta misma lógica permite que la numeración sea “global” entre sectores o niveles: la secuencia continúa sin reiniciarse, facilitando el relevamiento y la comunicación en campo.

Con esa numeración ya establecida, la asignación de categorías se realiza con una estrategia simple y defendible: se ordenan las categorías según el plan (prioridad y ubicación sugerida) y se les distribuye una cantidad de estantes casi uniforme, formando bloques contiguos de números para cada categoría. El orden intenta, cuando es posible, mantener juntas categorías que el plan recomienda ubicar cercanas, para que queden contiguas en la asignación y sea más natural su implementación por zonas.

El resultado final es un plano interactivo donde los estantes se muestran con un código visual de proximidad, mediante colores que distinguen estantes cercanos, intermedios y lejanos respecto al punto de referencia, donde se utiliza el verde para los que están próximos a la entrada, azul para los que están a medio camino y naranja para los que están alejados, y con etiquetas/indicadores que permiten identificar cada estante.

Al pasar el cursor se presenta un resumen con el número de estante y la categoría asignada, lo que habilita revisar rápidamente si el diseño respeta la intención del plan (categorías prioritarias más accesibles, categorías secundarias más alejadas y agrupamientos coherentes). Adicionalmente, se genera una tabla estructurada con la asignación final (estante → categoría → proximidad) para poder reutilizar el resultado en otros sistemas o reportes sin depender de la visualización.

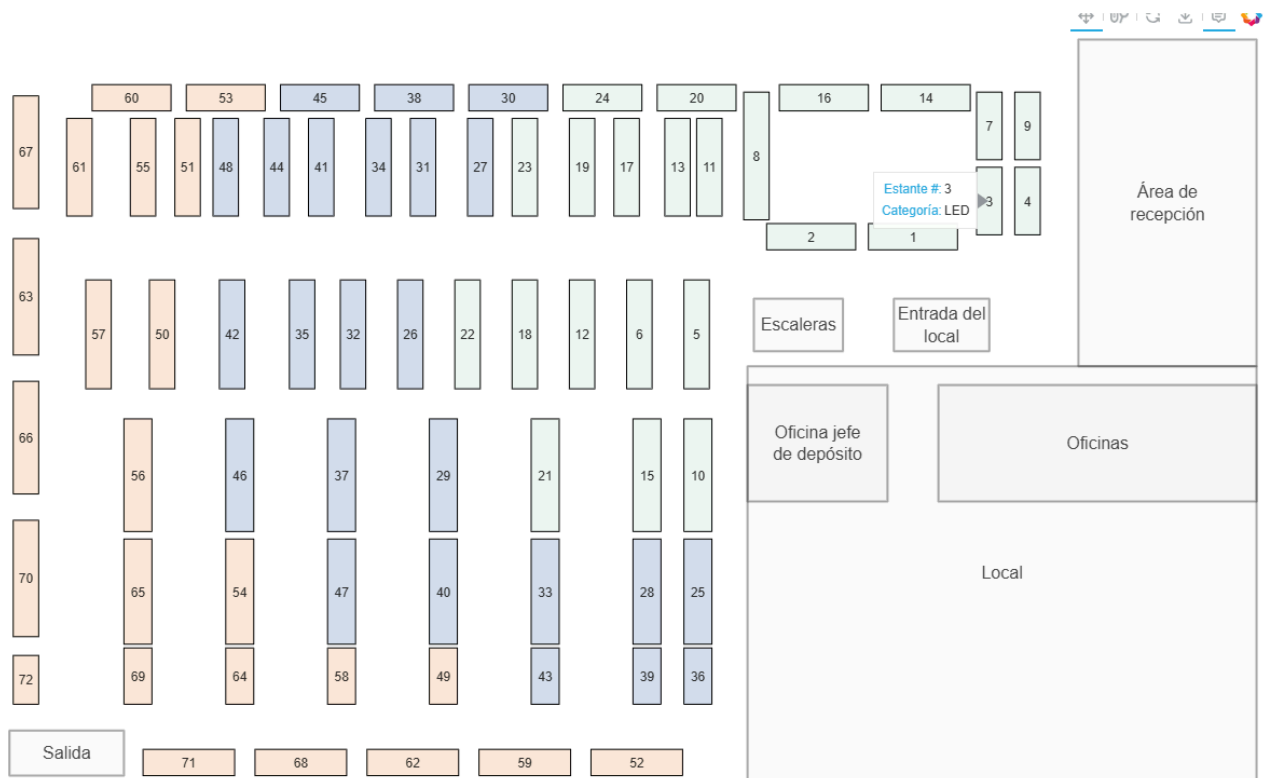


Imagen 11: Layout de planta baja optimizado

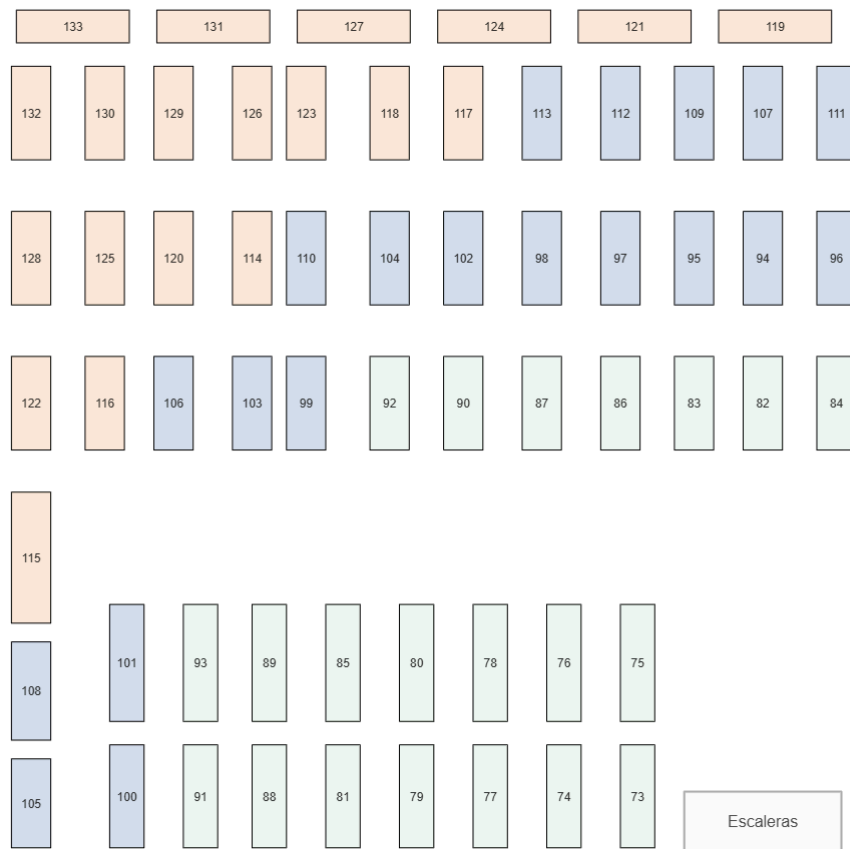


Imagen 12: Layout de primer piso optimizado

5. Conclusiones y Aprendizajes

La realización de esta práctica supervisada me permitió aplicar de manera integrada los conocimientos adquiridos a lo largo de la carrera de Ingeniería en Sistemas de Información, enfrentándome a una problemática real vinculada a la organización, clasificación y explotación de datos dentro de un contexto comercial concreto. A lo largo del proyecto pude analizar el dominio del negocio, estructurar información inicialmente desordenada y diseñar soluciones orientadas a mejorar procesos operativos como la organización del depósito y el armado de pedidos.

En particular, el desarrollo del sistema multi-agente de categorización y del pipeline de datos me permitió profundizar en conceptos de ingeniería de datos, automatización y uso de modelos de lenguaje para la clasificación de información textual. La experiencia de diseñar un flujo completo, desde la extracción de datos desde archivos Excel, pasando por su transformación y clasificación, hasta la carga en bases de datos relacionales y la generación de datos sintéticos de ventas, reforzó la importancia de la trazabilidad, la coherencia de los datos y el diseño de arquitecturas reproducibles.

Más allá de los aspectos técnicos, el proyecto me ayudó a comprender la relevancia de alinear las soluciones tecnológicas con estructuras ya existentes del negocio, como el catálogo de productos digital, y de pensar los sistemas con una visión integral y escalable. Considero que esta práctica fue una experiencia formativa muy valiosa, ya que me permitió ganar confianza en mis capacidades, aplicar conocimientos en un contexto real y sentar una base sólida para continuar desarrollándome profesionalmente en el área de ingeniería de datos y sistemas inteligentes.

6. Trabajos Futuros

Como trabajos futuros, se propone avanzar hacia una implementación del layout optimizado utilizando datos reales provenientes del sistema ERP con el que cuenta Voltaje S.R.L., integrando dichos datos mediante un proceso ETL que permita extraer, transformar y cargar de forma periódica información de ventas, stock y movimientos de productos en un datawarehouse analítico. Esto permitiría trabajar con datos actualizados y mantener el análisis alineado con la operación real del negocio. A su vez, el modelado del layout podría enriquecerse incorporando criterios físicos del depósito, como el peso y volumen de los productos, definiendo en qué niveles de los estantes deberían ubicarse (por ejemplo, productos más pesados en filas inferiores), con el objetivo de mejorar la ergonomía, la seguridad y la eficiencia del proceso de armado de pedidos.

Además, se podría incorporar un sistema de señalización física para los pasillos y estanterías, indicando claramente qué tipos y categorías de productos se encuentran en cada sector.

En paralelo, el sistema multi-agente de categorización podría evolucionar hacia una solución integrada al flujo del ERP, de modo que, ante la incorporación de un nuevo producto al catálogo, este sea automáticamente clasificado, categorizado y persistido en la

base de datos. Finalmente, el layout optimizado podría modelarse en un entorno tridimensional, permitiendo visualizar el depósito en 3D, simular recorridos y evaluar distintos escenarios de distribución antes de su implementación física, contribuyendo a una planificación más precisa y a la mejora continua de la operación logística.

7. Referencias bibliográficas

- [1] Han, J., Kamber, M., & Pei, J. (2012). *Data Mining: Concepts and Techniques* (3rd ed.). Morgan Kaufmann.
- [2] Tan, P.-N., Steinbach, M., & Kumar, V. (2018). *Introduction to Data Mining* (2nd ed.). Pearson.
- [3] Agrawal, R., & Srikant, R. (1994). Fast algorithms for mining association rules in large databases. In *Proceedings of the 20th VLDB Conference*.
- [4] Russell, S., & Norvig, P. (2020). *Artificial Intelligence: A Modern Approach* (4th ed.). Pearson.
- [5] Kimball, R., & Ross, M. (2013). *The Data Warehouse Toolkit: The Definitive Guide to Dimensional Modeling* (3rd ed.). Wiley.
- [6] Tompkins, J. A., White, J. A., Bozer, Y. A., & Tanchoco, J. M. A. (2010). *Facilities Planning* (4th ed.). Wiley.
- [7] Frazelle, E. H. (2002). *World-Class Warehousing and Material Handling*. McGraw-Hill.
- [8] Reis, J., & Housley, M. (2022). *Data Engineering with Python: Work with Massive Datasets to Design Data Models and Automate Data Pipelines Using Python* (1st ed.). O'Reilly Media.
- [9] Kretz, A. (2021). *The Data Engineering Cookbook*. Self-published.
- [10] Nikles, M. (2023). *Automating Workflows with n8n*. Self-published.
- [11] Apache Airflow. (2024). Apache Airflow – Plataforma de orquestación de flujos de trabajo – Documentación oficial. Disponible en <https://airflow.apache.org/>.
- [12] Microsoft. (2023). Arquitectura Medallion en el lago de datos. *Microsoft Learn*. Disponible en <https://learn.microsoft.com/es-es/fabric/real-time-intelligence/architecture-medallion>
- [13] Mecalux. (s. f.). ¿Cómo lograr un depósito eficiente? *Blog Mecalux Argentina*. Disponible en <https://www.mecalux.com.ar/blog/deposito-eficiente>
- [14] Deposeguro. (2023). Tipos de sistemas de almacenaje: ¿cuál elegir para tu depósito? *Blog Deposeguro*. Disponible en <https://deposeguro.com/blog/tipos-de-almacenaje/>
- [15] AutoGen Team. (2024). *AutoGen Documentation*. Disponible en <https://microsoft.github.io/autogen/>