Propuesta de categorización de productos de Voltaje S.R.L.

Alumno: Toledo, Alvaro Julian
Legajo: 52721
Cátedra: Práctica supervisada

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 “articulo”), 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 tendrá 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 “articulo” (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.

Diagrama de ontología

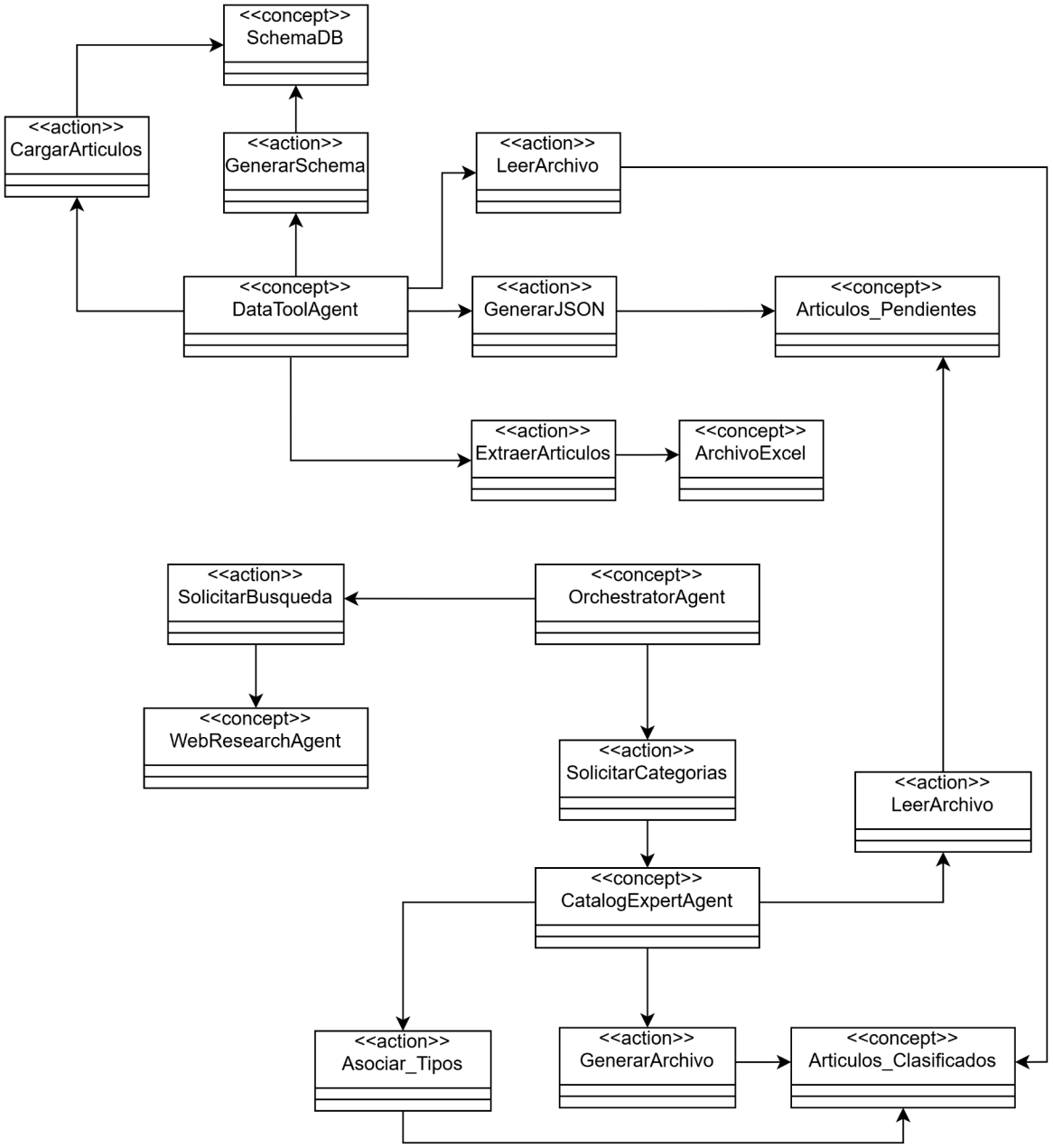
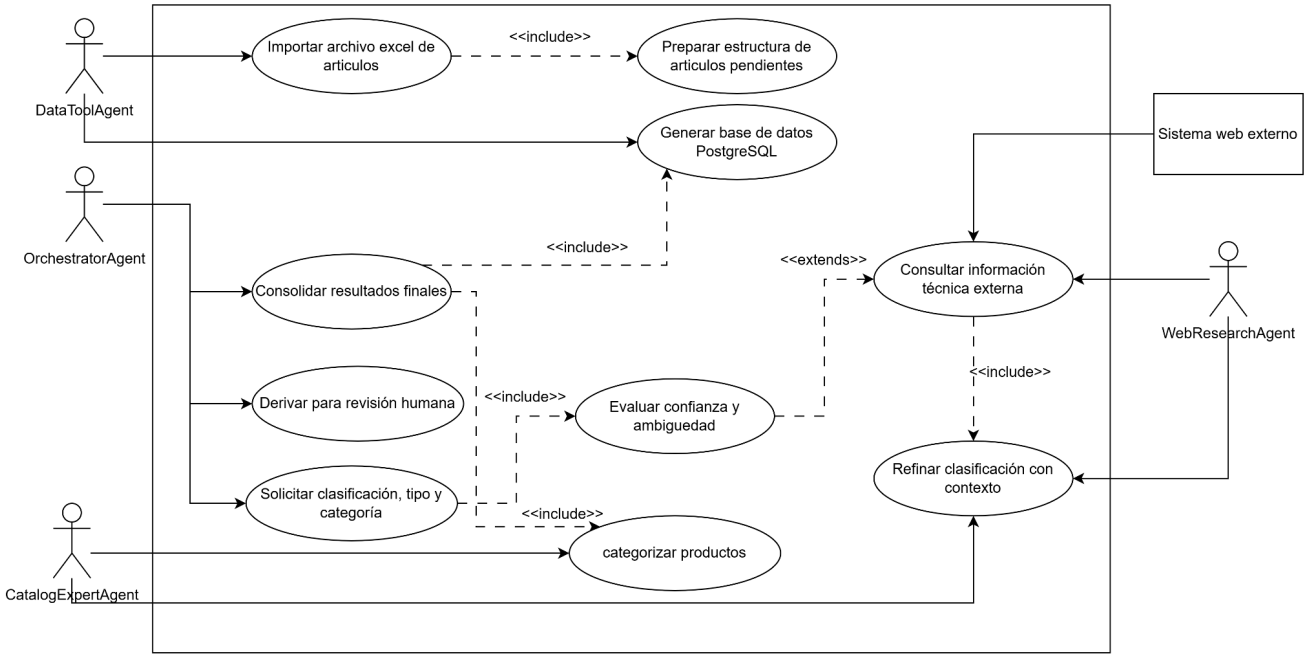


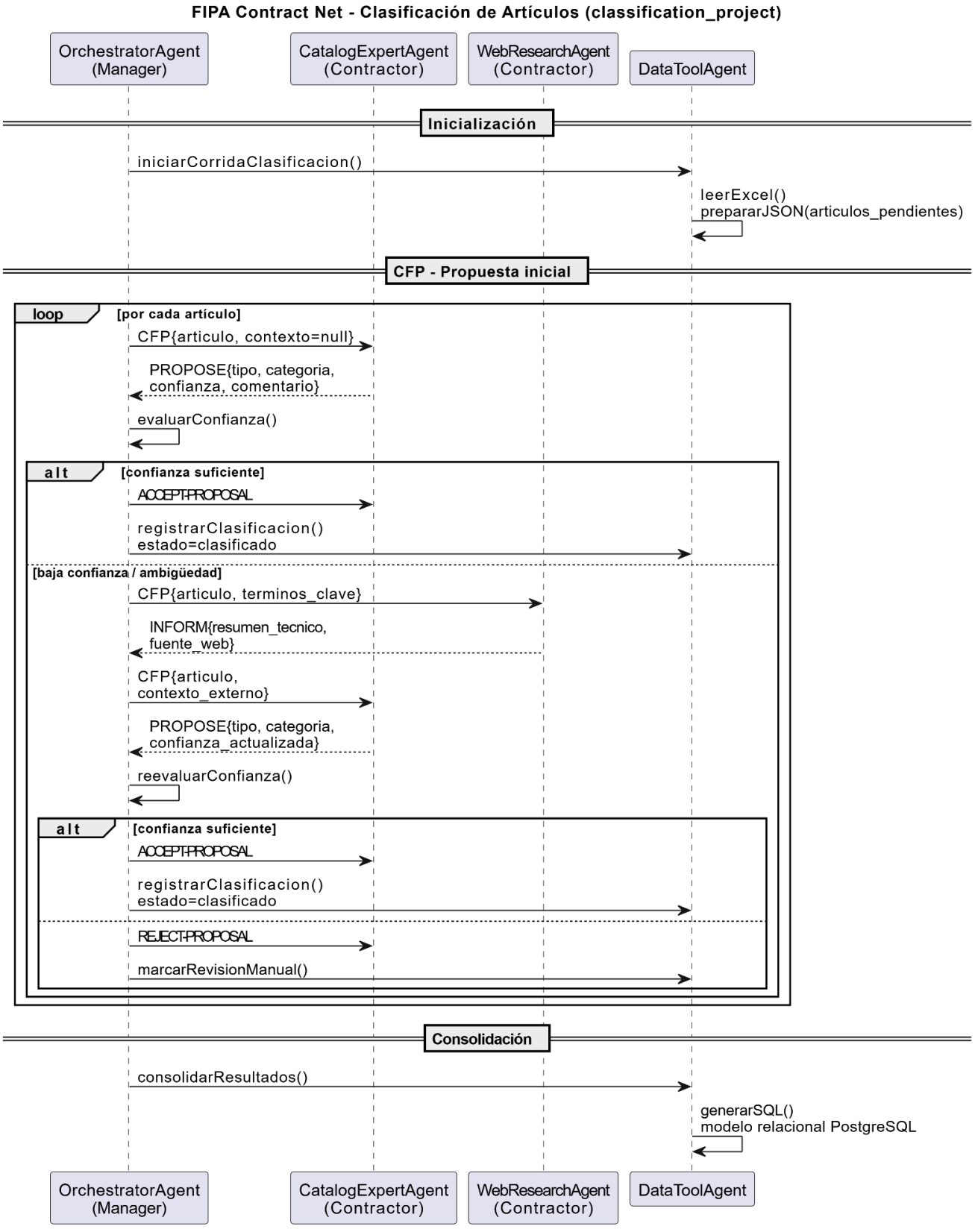
Diagrama de casos de uso (DCU)



El diagrama de casos de uso representa las funcionalidades principales del proyecto classification_project y las interacciones entre los actores y el sistema de clasificación automática de artículos. En él se modelan tanto los agentes internos del sistema (OrchestratorAgent, CatalogExpertAgent, WebResearchAgent y DataToolAgent) como el sistema web externo.

El diagrama permite identificar las responsabilidades de cada agente, destacando el rol del OrchestratorAgent como coordinador del proceso, la especialización del CatalogExpertAgent en la asignación de tipo y categoría, y el apoyo del WebResearchAgent en situaciones de baja confianza o ambigüedad. Asimismo, se explicita el flujo general desde la importación del archivo Excel, la clasificación automática y la evaluación de confianza, hasta la consolidación de los resultados y la generación del modelo de datos final.

Diagrama de secuencia



El diagrama de secuencia describe el comportamiento dinámico del sistema durante una ejecución típica de clasificación de artículos, siguiendo el patrón de comunicación FIPA Contract Net propio de sistemas multi-agente. En este diagrama se observa cómo el OrchestratorAgent actúa como manager, iniciando la interacción y solicitando propuestas de clasificación a los agentes especializados.

A lo largo de la secuencia se modelan los distintos escenarios posibles: aceptación directa de una clasificación cuando la confianza es suficiente, refinamiento mediante consultas a fuentes externas cuando existe ambigüedad, y derivación a revisión humana cuando no se alcanza un nivel adecuado de certeza. Finalmente, el diagrama muestra la etapa de consolidación, donde los resultados se persisten y se genera la estructura de datos en PostgreSQL, reflejando el cierre completo del proceso automatizado.

Flujo de funcionamiento

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”.

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.

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**.

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.

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.

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.

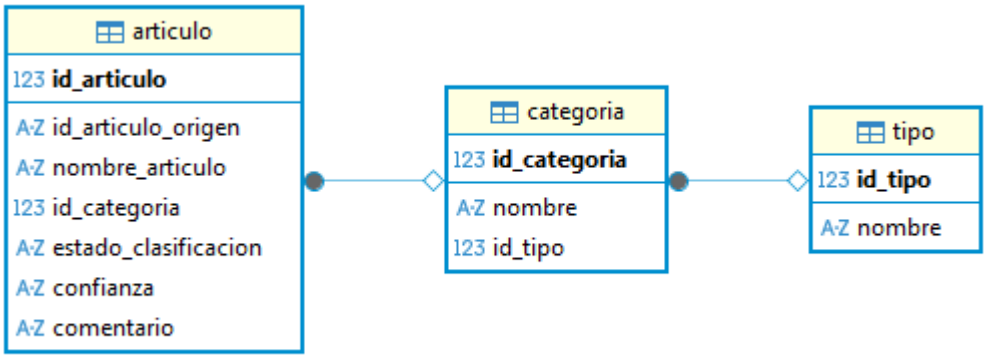


Imagen 1: Modelo relacional desarrollado

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:

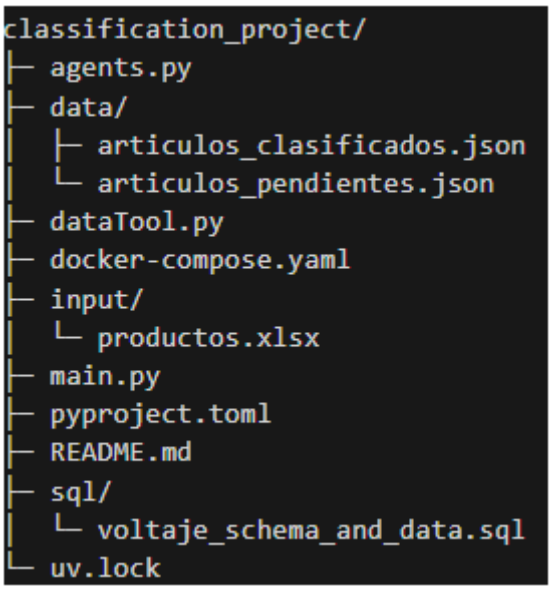


Imagen 2: Estructura del proyecto

Conclusión

El trabajo desarrollado permitió implementar un pipeline completo y reproducible para la clasificación de artículos a partir de descripciones textuales, integrando extracción de datos, clasificación asistida por modelos de lenguaje y persistencia en una base de datos relacional normalizada. El uso de archivos JSON como artefactos intermedios aseguró trazabilidad y control del proceso, mientras que el modelo relacional adoptado garantizó consistencia y ausencia de redundancias. En conjunto, la solución demuestra la viabilidad de aplicar técnicas modernas de automatización y clasificación en un contexto real, sentando una base sólida para futuras extensiones y mejoras del sistema.