INSTITUTO TECNOLÓGICO DE ESTUDIOS SUPERIORES DE OCCIDENTE  
Departamento de Electrónica, Sistemas e Informática.

**Proyecto (Fase 2)**

|  |  |
| --- | --- |
| Materia: | Almacenes de Datos |
| Integrantes del equipo: | Alan Daniel Ríos López | 734869  Paulina Flores Sánchez | 745570 Sergio David Elizondo Silva | 745602 |
| Profesor: | Sergio Rubén Aceves Azuara |
| Fecha: | 05 de diciembre de 2025 |

1. **Introducción**

El presente proyecto tiene como objetivo la actualización del data Warehouse creado en la primera parte del proyecto, con el propósito de realizar recomendaciones para las entradas y salidas del almacén.

La primera parte del proyecto consiste en migrar la base de datos relacional (AutoPartes) a una base de datos de grafos en Neo4j. Para comenzar, una **base de datos de grafos** es una plataforma especializada en la creación y manipulación de datos. Los grafos contienen nodos y relaciones que representan y almacenan datos de una forma visual y compacta, lo cual no es posible con las bases de datos relacionales (Pykes, 2024).

Una de las principales diferencias entre las BDR y las BDG es la forma de consultar la información contenida en ellas. Las primeras utilizan SQL para sus operaciones, mientras que las segundas emplean algoritmos transversales, lo que las hace más rápidas incluso para grandes conjuntos de datos, además de más adecuadas para información con alta interconexión (Pykes, 2024).

El desarrollo de la base de datos de grafos será en **Neo4j**, la cual es una base de datos de NoSQL de código abierto, altamente escalable desarrollada con Java, la cual utiliza el lenguaje de consulta de Cypher. Entre las principales características de esta BDG se encuentran un modelado de datos intuitivo y flexible, una api nativa de alto rendimiento, un método de consulta de grafos potente y fácil de utilizar, y transacciones que cumplen con el formato ACID (Pykes, 2024).

Aunque la creación de la nueva base de datos sea dentro de la plataforma Neo4j, la migración de datos se hará dentro de la herramienta **KNIME**, la cual permite crear flujos de trabajo basados en el modelo ETL (Extraer, Transformar y Cargar información) con el fin de limpiar, preparar y combinar datos de diferentes fuentes para su análisis o almacenamiento (Davies, 2024). KNIME se usará para la migración de datos, así como para la creación de los archivos CSV y la integración de la nueva información al datawarehouse.

Por último, para representar los datos de una manera más visual e intuitiva para el usuario se expresará la información en un formato de tableros, especificamente usando **Tableau**. Tableau es un software de inteligencia empresarial que se usa para procesar grandes volúmenes de datos, transformándolos en visualizaciones para los usuarios del negocio. De esta forma, puede ayudar a los usuarios a tomar decisiones informadas sobre productos o predecir el rendimiento futuro basándose en el rendimiento actual (Coursera, 2025).

1. **Transformar el modelo relacional a uno basado en grafos**
2. **Modelar el grafo**

El primer paso hacia la migración de datos de una base de datos relacional a una de grafos es crear un borrador del modelo al que se quiere llegar. Como se necesitarán dos bases de datos de grafos para la correcta recomendación de artículos en cada una de ellas, el modelado también se dividirá en dos: entradas y salidas.

Diagrama

El contenido generado por IA puede ser incorrecto.

Diagrama

El contenido generado por IA puede ser incorrecto.

1. **Convertir las tablas a CSV**

El siguiente paso para la migración de la información es convertir las tablas a archivos CSV, ya que estos serán la fuente de donde los nodos sacarán la información para su poblamiento. Este proceso es realizado dentro de un flujo de trabajo de KNIME, cuyo resultado es el siguiente:

Gráfico, Diagrama

El contenido generado por IA puede ser incorrecto.

1. **Insertar la información de Entradas**

Una vez obtenidos los CSV necesarios para poblar la nueva base de datos de grafos, se crea otro flujo de trabajo dentro de KNIME que contendrá 2 diferentes tipos de nodos:

1. Neo4j Conection: se conecta a la instancia de Neo4j.
2. Neo4j Writer: introduce información en la base de datos.

Obteniendo como resultado un flujo de trabajo como el siguiente:

Diagrama

El contenido generado por IA puede ser incorrecto.

Al mismo tiempo, los nodos destinados para introducir información se dividen en 3 grandes secciones:

1. Creación de nodos: cada nodo dentro de la sección verde tiene el objetivo de crear un nodo diferente dentro de la base de datos de grafos, con base en el archivo csv que pertenece a cada uno de estos. Dentro de cada query, puede haber dos tipos diferentes de sintaxis para lograr el mismo objetivo, dependiendo de la cantidad de datos que se espera.
2. Más de 500,000 datos:

// Entrada

CALL apoc.periodic.iterate(

    'LOAD CSV WITH HEADERS FROM "file:///EntradaEncabezado.csv" AS row RETURN row',

    'CREATE (n: Entrada) SET n = row',

    {batchSize:10000, parallel:false}

)

1. Menos de 500,000 datos:

// Articulo

LOAD CSV WITH HEADERS FROM "file:///Articulo(ES).csv" AS row

CREATE (n: Articulo)

SET n = row;

1. Creación de índices: cada nodo dentro de la sección amarilla tiene el objetivo de asignar un índice a cada nodo creado anteriormente. El query para crear los índices es el mismo en todos los casos, un ejemplo es:

CREATE INDEX Index\_Entrada FOR (n: Entrada) ON (n.Folio)

1. Relaciones entre nodos: cada nodo dentro de la sección azul asigna las relaciones entre nodos. Al igual que con la creación de nodos, el query que se utilice depende de la cantidad de datos que contenga cada nodo.
2. Más de 500,000 datos:

// Entrada\_Detalle -> Entrada (DETALLA)

CALL apoc.periodic.iterate(

'MATCH (ed:Entrada\_Detalle)

MATCH (e:Entrada)

WHERE ed.Folio = e.Folio

RETURN ed, e',

'MERGE (ed)-[:DETALLA]->(e)',

{ batchSize:10000, parallel:false }

)

1. Menos de 500,000 datos:

// Entrada -> Cliente (SE\_VENDE\_A)

MATCH (e:Entrada)

MATCH (c:Cliente)

WHERE e.ClienteID = c.ClienteID

MERGE (e)-[:SE\_VENDE\_A]->(c)

Para comprobar la correcta creación de la nueva base de datos en Neo4j, se revisa la visualización dentro del mismo, obteniendo el siguiente modelado como resultado final:

Diagrama

El contenido generado por IA puede ser incorrecto.

1. **Insertar la información de Salidas**

Para la creación de la base de datos de grafos de salidas, se ejecuta el mismo procedimiento que en entradas, con la diferencia del nombre de los nodos, así como la fuente de la que obtienen la información estos.

Por lo que, el flujo de trabajo de KNIME para las salidas quedaría de la siguiente manera:



Obteniendo la siguiente visualización de la base de datos ya creada dentro de Neo4j:

Diagrama, Esquemático

El contenido generado por IA puede ser incorrecto.

1. **Realizar recomendaciones**
2. **Recomendaciones en Entradas**

Después de tener nuestra base de datos de grafos poblada y funcional, el siguiente paso es generar las recomendaciones entre el articulo 1 y el articulo 2 de las entradas al almacén.

Para comenzar a construir el query para la recomendación de artículos es realizar un caso de prueba dentro de Neo4j, en este caso, con el articulo 100-1931Y. Este primer caso de prueba es importante, ya que permite conocer si la estructura o la lógica del query es la correcta, sirviendo como base para hacer el query final.

// Ejemplo de ejecucion en Neo4j

MATCH

(a1:Articulo)<-[:COMPRA]-(ed1:Entrada\_Detalle)-[:DETALLA]->(e:Entrada)<-[:DETALLA]-(ed2:Entrada\_Detalle)-[:COMPRA]->(a2:Articulo)

WHERE a1 < a2 AND a1.ArticuloID = "100-1931Y"

RETURN

    a1.ArticuloID AS Articulo\_Original,

    a2.ArticuloID AS Recomendacion,

    count(e) AS Frecuencia

ORDER BY Frecuencia DESC

LIMIT 10

El resultado del query anterior se muestra en la siguiente tabla:

Interfaz de usuario gráfica, Texto, Aplicación

El contenido generado por IA puede ser incorrecto.

Con el correcto funcionamiento del query de prueba, el siguiente y último paso es crear el query para la recomendación de los artículos, en donde se usa la función apoc.export.csv.query, la cual guarda la información obtenida del query en un archivo csv.

// Query dentro de KNIME

CALL apoc.export.csv.query(

    'MATCH

    (a1:Articulo)<-[:COMPRA]-(ed1:Entrada\_Detalle)-[:DETALLA]->(e:Entrada)<-[:DETALLA]-(ed2:Entrada\_Detalle)-[:COMPRA]->(a2:Articulo)

    WHERE a1 < a2

    RETURN

        a1.ArticuloID AS Articulo\_Original,

        a2.ArticuloID AS Recomendacion,

        count(e) AS Frecuencia',

  "file:////recomendaciones\_entradas.csv",

    {

        batchSize: 10000,

        parallel: false

    }

)

YIELD file

RETURN file

1. **Recomendaciones en Salidas**

Para la creación de las recomendaciones de salidas, se sigue el mismo proceso que con las de entradas, en el cual primeramente se genera un caso de prueba, en este caso, el articulo 100-1931Y.

// Ejemplo de ejecucion en Neo4j

MATCH

(a1:Articulo)<-[:VENDE]-(sd1:Salida\_Detalle)-[:DETALLA]->(s:Salida)<-[:DETALLA]-(sd2:Salida\_Detalle)-[:VENDE]->(a2:Articulo)

WHERE a1 < a2 AND a1.ArticuloID = "100-1931Y"

RETURN

    a1.ArticuloID AS Articulo\_Original,

    a2.ArticuloID AS Recomendacion,

    count(s) AS Frecuencia

ORDER BY Frecuencia DESC

LIMIT 10

El resultado del query anterior se muestra en la siguiente tabla:

Interfaz de usuario gráfica, Texto

El contenido generado por IA puede ser incorrecto.

Seguidamente, se crea el query final para todas las recomendaciones posibles de los artículos de salida de almacén, utilizando la función apoc.export.csv.query, la cual guarda la información obtenida del query en un archivo csv.

// Query dentro de KNIME

CALL apoc.export.csv.query(

    'MATCH

    (a1:Articulo)<-[:VENDE]-(sd1:Salida\_Detalle)-[:DETALLA]->(s:Salida)<-[:DETALLA]-(sd2:Salida\_Detalle)-[:VENDE]->(a2:Articulo)

    WHERE a1 < a2

    RETURN

        a1.ArticuloID AS Articulo\_Original,

        a2.ArticuloID AS Recomendacion,

        count(s) AS Frecuencia',

  "file:////recomendaciones\_salidas.csv",

    {

        batchSize: 10000,

        parallel: false

    }

)

YIELD file

RETURN file

1. **Integrar la información del datawarehouse**
2. **Incluir las recomendaciones al datawarehouse como tablas de hechos**

**A diagram of a diagram

AI-generated content may be incorrect.**

*Salidas*

A diagram of a diagram

AI-generated content may be incorrect.

*Entradas*

1. **Aseguramiento de schema de Artículo**

**A screenshot of a computer

AI-generated content may be incorrect.**

1. **Agregar recomendaciones a data warehouse**

**A diagram of a diagram

AI-generated content may be incorrect.**

*Flujo knime general*

**Creación de tablas de recomendaciones**

Primero se crearon nodos SQL Executor para crear tablas de entradas y salidas con el mismo modelo de la dimensión artículo pero incluyendo los campos ArticuloRecomendadoId y Frecuencia. Una vez creadas las tablas se copió la información de dim\_artículo.

**A diagram of a computer

AI-generated content may be incorrect.**

**Llenado de recomendaciones**

Primero se leyeron los csv generados por un flujo previo de knime donde se obtienen ya las recomendaciones directamente de la base de datos de grafos. Ya que se leen, se ordenan por frecuencia para utilizar el producto con mayor frecuencia. Ya que se tienen los registros sorteados, se agrupan por producto para tener la recomendación más adecuada de cada producto, su producto con mayor frecuencia. Se cambian los nombres para poder hacer uso del nodo db update.

**A diagram of a groupy diagram

AI-generated content may be incorrect.**

**Limpieza de datos en tablas de recomendaciones**

Ya que se agregaron las recomendaciones, es posible que no todos los productos tengan una compra compuesta, es decir, que no podamos recomendar un producto en la compra de otro porque nunca se compraron al mismo tiempo desde un inicio. Esto puede hacer que existan registros con valores nulos, para eso usamos otro sql executor que limpia los valores nulos de las tablas de recomendaciones.

A diagram of a software system

AI-generated content may be incorrect.

**Tablas en base de datos**

**A screenshot of a computer

AI-generated content may be incorrect.**

**1.-**

1. **Analizar la información**
2. Tablero Entradas



Búsqueda de 15 Artículos en Entradas

1. **CUARTO PUNTA TOYOTA PU 89-95 CROM TRAC NOR DER**



1. **CUARTO PUNTA TOYOTA PU 89-95 NEGRO DOBLE TRACC DER**
2. **CALAVERA TOYOTA PU 84-88 PINTADA (TODAS) DER**



1. **CALAVERA TOYOTA PU 89-95 (TODAS) IZQ**



1. **CUARTO PUNTA ISUZU PU / LUV 88-95 CROM AMBAR DER**



1. **CALAVERA ISUZU PU / LUV 83-87 CROMADA DER**



1. **CUARTO PUNTA MITS PU L200/RAM50 87-96 NEGRO BI DER**



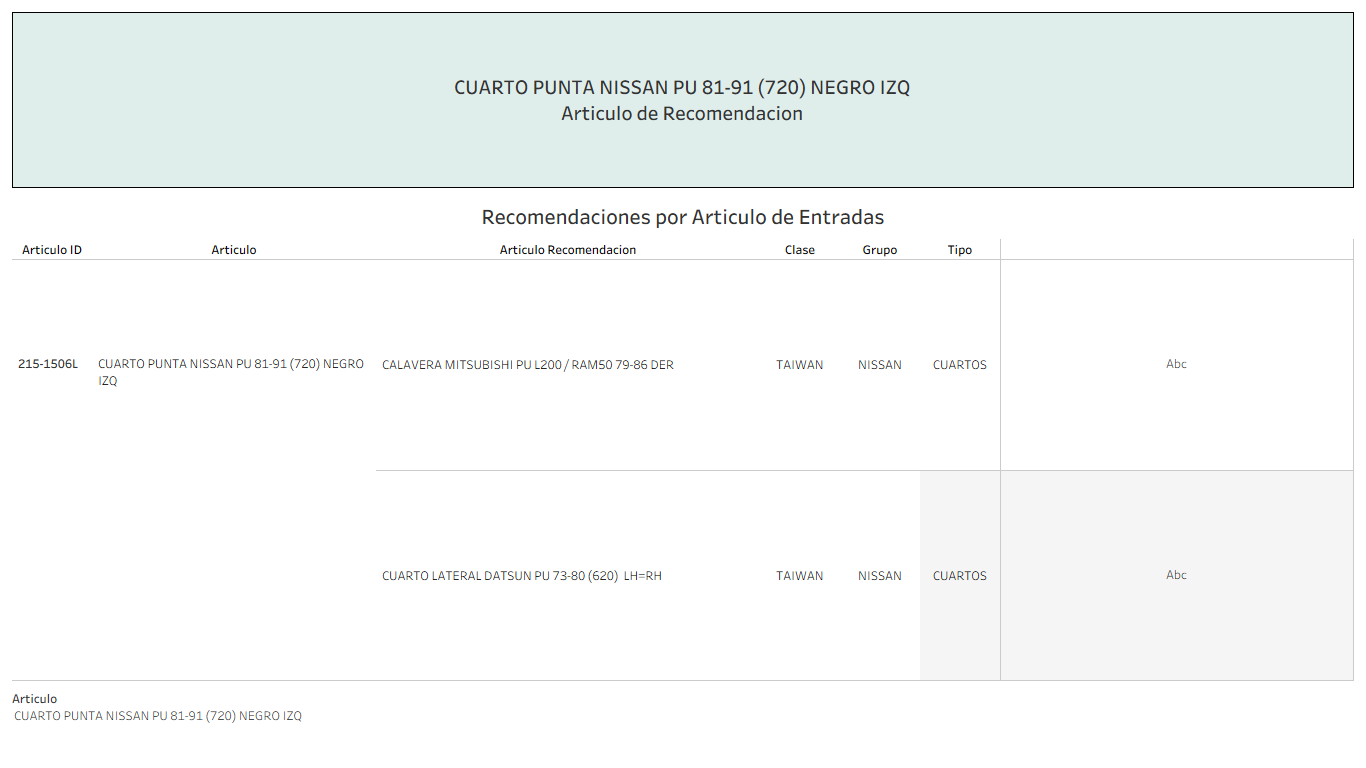
1. **FARO TSURU I SAMURAY 85-87 S/BASE S/AJUSTES IZQ**



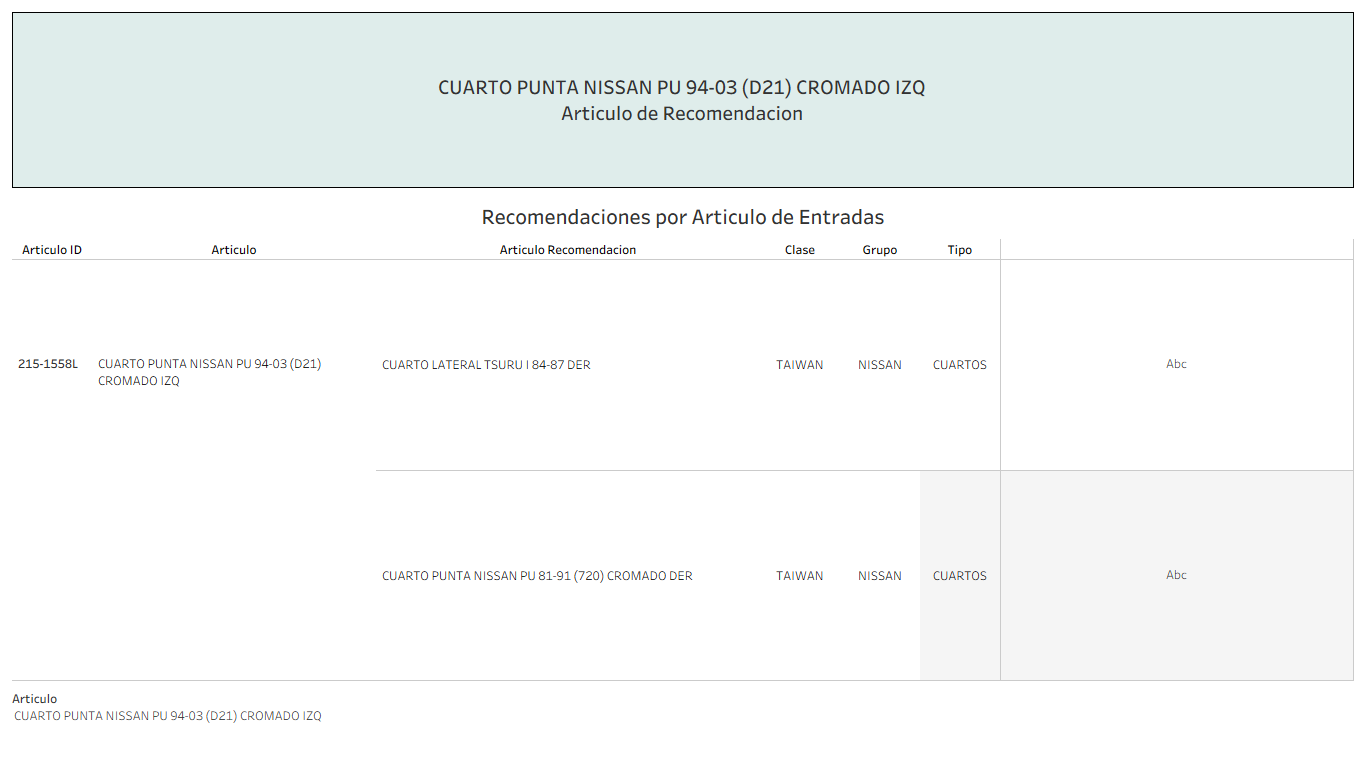
1. **CUARTO PUNTA NISSAN PU 92-93 (720) FILO CROMAD DER**



1. **CUARTO PUNTA NISSAN PU 81-91 (720) NEGRO IZQ**



1. **CUARTO PUNTA NISSAN PU 94-03 (D21) CROMADO IZQ**



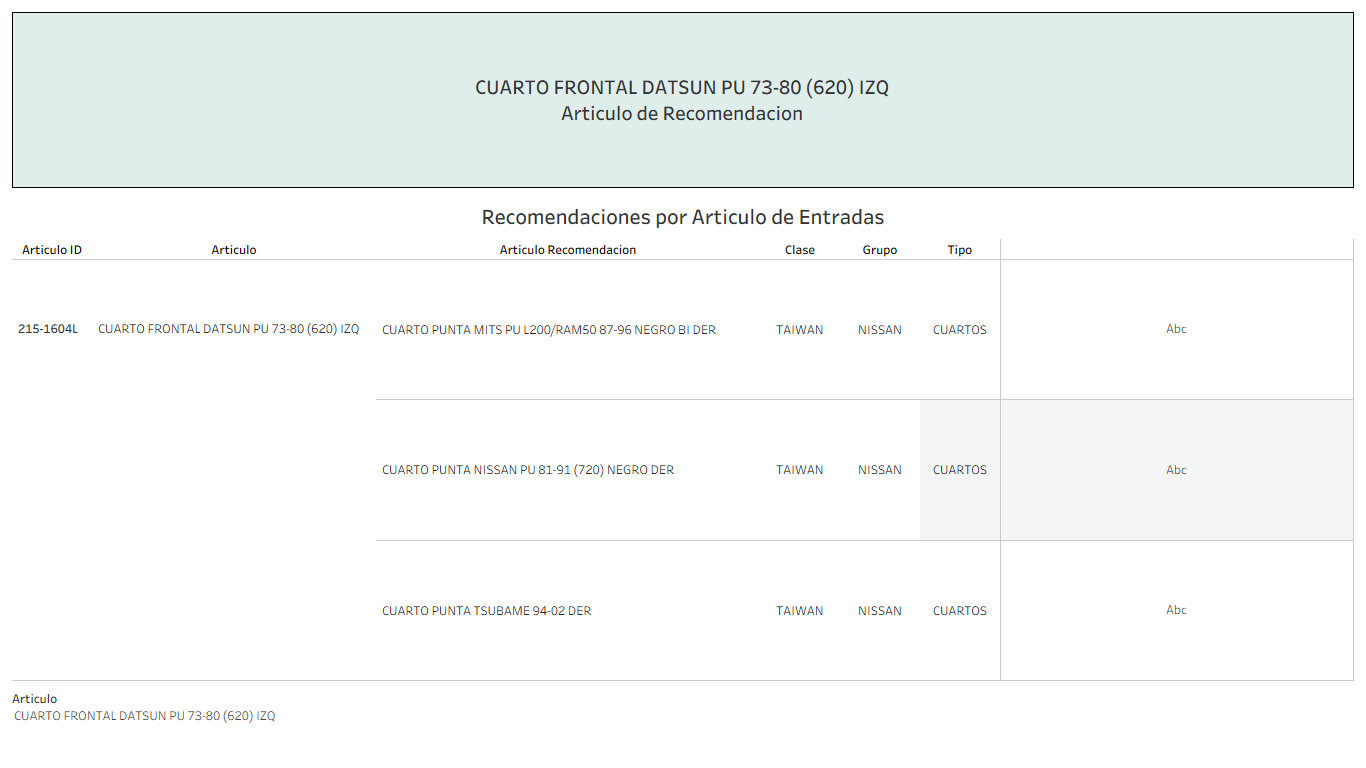
1. **CUARTO PUNTA TSURU III 01-03 IZQ**



1. **CUARTO PUNTA TSURU III 01-03 DER**



1. **CUARTO FRONTAL DATSUN PU 73-80 (620) IZQ**



1. **FARO FORD PU 92-96 C/AJUSTES IZQ**



1. Tablero Salidas



Búsqueda de 15 Artículos en Salidas

1. FARO TSURU III 01-03 C/AJUSTES IZQ



1. CUARTO PUNTA TSURU II 90-91 AMBAR DER



1. CUARTO PUNTA PATHFINDER 97-98 CROMADO DER



1. CALAVERA NISSAN PU 87-97 D-21 AMER (LARGA) IZQ



1. CALAVERA NISSAN PU 87-97 D-21 AMER (LARGA) DER



1. CUARTO PUNTA MAZDA PU 86-93 BICOLOR DER



1. FARO FORD PU 92-96 C/AJUSTES DER



1. FARO MUSTANG 94-98 C/AJUSTES IZQ



1. CUARTO PUNTA RANGER 93-97 DER



1. CUARTO PUNTA GRAND MARQUIS 98-02 BICOLOR DER



1. CALAVERA RANGER 93-97 IZQ



1. CALAVERA RANGER 93-97 DER



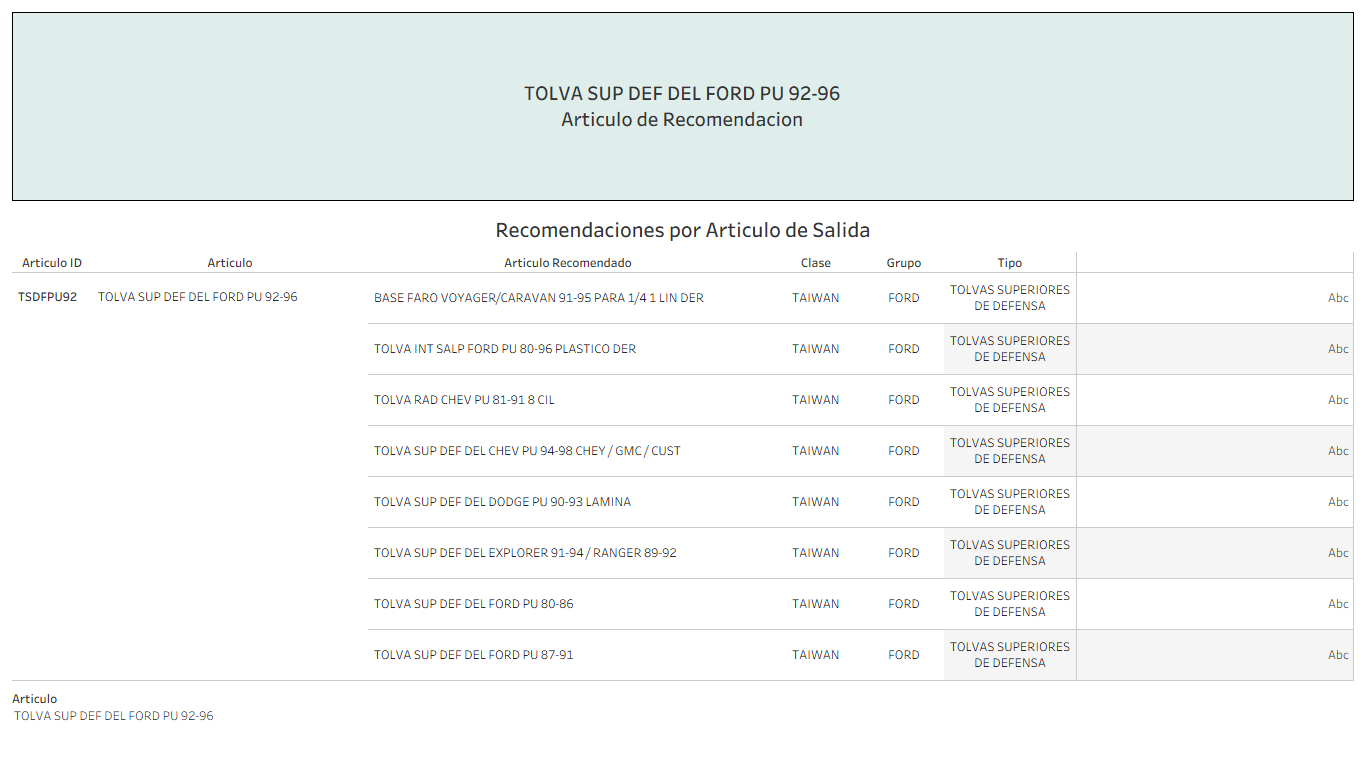
1. CALAVERA RANGER 83-92 IZQ



1. FARO CHEV PU CHEYENNE / GMC 92-98 C/BASE IZQ



1. TOLVA SUP DEF DEL FORD PU 92-96



1. **Casos de prueba**
2. Entradas
3. Salidas
4. Referencias

* Pykes, K. (2024, 27 de abril*). ¿Qué es una base de datos de grafos? Guía para principiantes.* Datacamp. <https://www.datacamp.com/es/blog/what-is-a-graph-database>
* Coursera. (2025, 13 de octubre). *¿Qué es Tableau? Características, casos de uso y más*. Coursera. <https://www.coursera.org/articles/tableau>