universidad de león

Escuela de Ingenierías
Industrial, Informática y Aeroespacial

GRADO EN INGENIERÍA INFORMÁTICA

Sistemas de Información de Gestión y Business Intelligence

Memoria de la aplicación: YourBestcar

Sergio Rodríguez Valdeón

07 de noviembre de 2020

Índice

Descripción del problema 3
Herramientas 4
4. Aplicación 5
4.1 Base de datos ..... **¡Error! Marcador no definido.**
Algoritmo de recomendacion 9
Análisis de resultados 10
DAFO 11
Líneas de futuro 12
Lecciones aprendidas 13
Bibliografia 14

Descripción del problema

Para muchas familias, cuando llega la hora de cambiar de coche, encontrar uno que cubra todas sus necesidades se convierte en una odisea, ya que son bienes de gran valor y compras que se realizan cada muchos años hay tomar correctamente la decisión de cual elegir. Para ello normalmente todas las familias deciden invertir unos días en visitar todos los concesionarios de su ciudad, o incluso de alrededores, probando infinidad de coches. Una vez probados todos comparar los que más se adapten a tus necesidades y gustos, comparando: valor, motor, tipo de tracción, numero de puertas… A esto se le añade el preguntar a conocidos su opinión, o buscar en foros información acerca de los vehículos, si dan problemas, coste de los mantenimientos…

Por esto he decidido crear un sistema de recomendación de vehículos, que reduzca el tiempo de búsqueda y el primer filtrado, para facilitar la vida del usuario, y que pierda el menor tiempo posible, de esta manera el usuario solo tendrá que ir a los concesionarios donde de verdad están los coches que le interesan.

En el documento expuesto a continuación vamos a realizar una descripción completa sobre el desarrollo y funcionamiento del sistema de recomendación.

Herramientas

Lo primero que debemos comentar es que es un sistema de recomendación, como su nombre indica es una importante herramienta que establece un conjunto de recomendaciones de usuarios sobre unos datos para realizar predicción sobre recomendaciones a usuarios que le puedan ser de utilidad y así mejorar su experiencia de usuario. Estos sistemas suelen seguir una temática, en mi caso coches, pero tenemos ejemplo como música (Spotify), películas y series (Netflix) o video (Youtube).

Para este sistema vamos a utilizar un sistema de recomendación basado en una base de datos orientada a grafos. Para implementar esta base he decido usar NEO4J.

NEO4J es un servicio implementado en Java, el cual usa grafos para representar datos y relaciones entre ellos, se utilizan círculos para representar los nodos y aristas para representar las relaciones entre ellos, esto nos permite extraer valores con un gran rendimiento, de forma ágil, flexible y altamente escalable.

Para mí era una tecnología nueva, pero Neo4j ofrece gran cantidad de documentación, ya sea en forma de video o escrita. Además, realicé el curso inicial que te permite obtener un conocimiento completo del lenguaje y las funcionalidades para realizar operaciones de nivel medio.

Para crear esta base he decidido descargar un csv de Kaggle, esta página contiene gran cantidad de dataset de distintos tipos y con distintas características. Después de explorar un poco me quede con uno que, aunque tenía información que no necesitaba me pareció el más completo. En el apartado siguiente se explicará la normalización del dataset y la creación de la base.

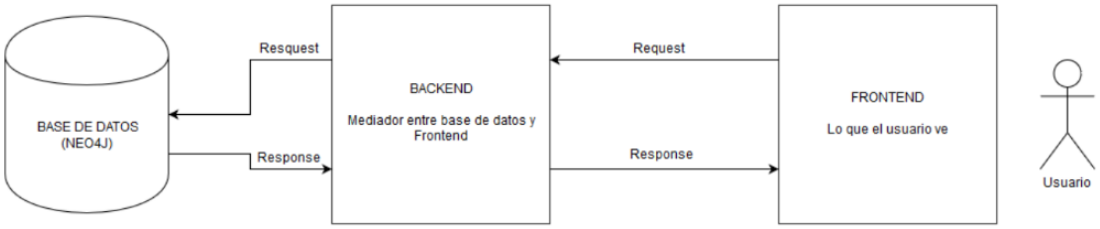
Decidí implementar el sistema en una aplicación web responsive, de esta manera permitir el acceso desde cualquier dispositivo. Estas aplicaciones constan de 2 partes, el Frontend, que sería la vista de la aplicación propiamente dicha, y el backend que es el que se comunica entre la base de datos y el Frontend.

Para crear el Frontend he decidido utilizar Vuetify, el cual es un framework que combina el popular framework para realizar SPA, VueJS con la estética de Material Desing. Esto me permite desarrollar una aplicación limpia y simple de manera sencilla.

En cambio, para el backend tuve dudas entre realizarlo en Python o JavaScript, ya que Python te ofrece mayor potencia de cálculo, pero no tengo conocimiento para utilizarlo en un sistema de tipo, por lo que tuve que optar por JavaScript ya que aprender Python me llevaría demasiado tiempo. Para ello utilice NodeJs que es un entorno de tiempo de ejecución de Javascript., y te permite ejecutar los js con un comando. Además para realizar la comunicación entre el backend y la base se utilizó el neo4j-driver, un módulo que ya viene implementado en JavaScript y habilita la comunicación. Y se utilizó Expess que permite la comunicación entre Backend y Frontend.

4. Aplicación

Como hemos explicado en el apartado anterior la estructura básica de la misma sería:



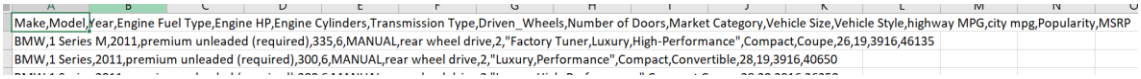
Su funcionamiento básico seria, por ejemplo, el usuario quiere obtener un listado de todos los vehículos, se envía esta petición desde el Frontend al backend, y este a su vez realiza la petición a la base, en este caso lenguaje Cypher, la respuesta realiza el camino inverso y el Frontend se encarga en mostrarlos, en forma de lista, tabla, solo el nombre…

Este tipo de peticiones se realizan en un breve espacio de tiempo, menos de 1 segundo, lo que más puede retrasar la petición es una consulta a la base de datos que se demore más de lo debido.

Una vez sabido el funcionamiento, vamos a dividirlo en 3 grandes bloques: Base de datos, Backend y Frontend.

4.1 Base de datos

Para crear la base, partimos del csv, contaba con 11914 registros y las siguientes columnas:


Make,Model,Year,Engine Fuel Type,Engine HP,Engine Cylinders,Transmission Type,Driven_Wheels,Number of Doors,Market Category,Vehicle Size,Vehicle Style,highway MPG,city mpg,Popularity,MSRP
BMW,1 Series M,2011,premium unleaded (required),335,6,MANUAL,rear wheel drive,2,"Factory Tuner,Luxury,High-Performance",Compact,Coupe,26,19,3916,46135
BMW,1 Series,2011,premium unleaded (required),300,6,MANUAL,rear wheel drive,2,"Luxury,Performance",Compact,Convertible,28,19,3916,40650

Lo primero que vi es que había valores N/A y 0, por lo que procedí a borrar las líneas con valores defectuosos, esto lo hice realizando una conversión a .xlxs y usando la funcionen de búsqueda. Después de una valoración decidí eliminar las columnas: Engine Fuel Type,Engine CylindersMarket Category,Vehicle Style y Popularity, también se formateo el nombre de las columnas para eliminar espacios y que no hubiese problemas a la hora de leer datos.

Comencé creando un nodo centra llamado Cars, continué creando un nodo Brand con el nombre de cada marca y relacionándolos entre sí con la relación MARCA.

```
LOAD CSV WITH HEADERS FROM "file:///data.csv" AS line
Merge (:Brand { name: line.Make})
Create(:Cars)
MATCH (a:Cars),(b:Brand)
CREATE (a)-[r:MARCA]->(b)
```

A continuación, se importaron todos los vehículos con las propiedades necesarias:

```
LOAD CSV WITH HEADERS FROM "file:///data1.csv" AS line

Create (:Car { model: line.Model,marca:line.Make,transmision: line.Tra
nsmission, traccion:line.Driven_Wheels, puertas:line.Number_Doors, tam
año:line.Vehicle_Size, year: line.Year, cv: line.Engine_HP,consumo_urb
ano: line.city_mpg, consumo_extraurbano: line.highway_MPG, precio:line
.MSRP})
```

Tuve que normalizar varios datos ya que el dataset no estaba correcto del todo, se muestran a continuación:

- Propiedad “tracción”: existía el valor “four Wheel drive” y “all wheel drive”, por lo que se modificó y se sustituyeron el primer valor por el segundo en todos los nodos.

```
MATCH (n:Car {traccion:"four wheel drive"}) set n.traccion="all wheel
drive"
```

- Propiedad Transmision: se cambiaron los valores de “UNKNOWN” por “MANUAL”; “DIRECT_DRIVE” Y “AUTOMATED_MANUAL” por “AUTOMATIC”

```
MATCH (n:Car {transmision:"UNKNOWN"}) set n.transmision="MANUAL"
MATCH (n:Car {transmision:"DIRECT_DRIVE"}) set n.transmision="AUTOMATI
C"
MATCH (n:Car {transmision:"AUTOMATED_MANUAL"}) set n.transmision="AUTO
MATIC"
```

Cree los nodos para las propiedades, esto ayudaría en un futuro a mejorar la eficacia de las recomendaciones. Estos fueron:

3 de tipo Traccion para los 3 valores posibles:

```
create(:Traccion {name:"all wheel drive"})
create(:Traccion {name:"front wheel drive"})
create(:Traccion {name:"rear wheel drive"})
```

2 de tipo Puerta para cada valor (2,4):

```
create(:Puerta {numero:4})
create(:Puerta {numero:2})
```

2 de tipo Transmision para cada valor(AUTOMATIC, MANUAL):

```
CREATE (:Transmision {name:"AUTOMATIC"})
CREATE (:Transmision {name:"MANUAL"})
```

3 de tipo Tamaño para cada valor (Compact,Midsize,Large):

```
create(:Tamaño {name:"Compact"})
create(:Tamaño {name:"Midsize"})
create(:Tamaño {name:"Large"})
```

Posteriormente, relacione cada vehículo con el nodo el cual poseía su propiedad de la siguiente manera:

```
  MATCH (b:Brand),(c:Car)
WHERE c.brand = b.name
CREATE (b)-[r:FABRICA]->(c)

MATCH (p:Puerta),(c:Car)
WHERE toInteger(c.puertas) = p.numero
CREATE (c)-[r:TIENE_PUERTAS]->(p)

MATCH (t:Traccion),(c:Car)
WHERE c.traccion= t.name
CREATE (c)-[r:TRACCION_EN]->(t)

MATCH (t:Transmision),(c:Car)
WHERE c.transmision= t.name
CREATE (c)-[r:TRANSMISION_IS]->(t)

MATCH (t:tamaño),(c:Car)
WHERE c.tamaño= t.name
CREATE (c)-[r:TAMAÑO_IS]->(t)
```

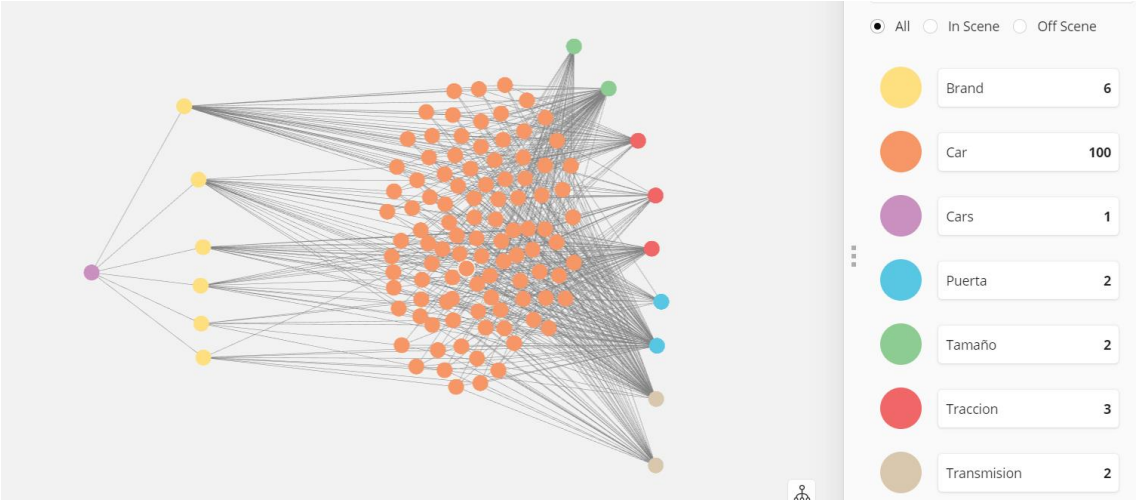
Y una vez relacionados, elimine las propiedades para que no hubiese información repetida y fallos de inconsistencia:

```
MATCH (c:Car )
remove c.marca,c.puertas,c.traccion,c.transmision,c.`tamaño`
```

Algunos datos números se había guardado como texto por lo que realice la conversión:

```
MATCH (c:Car) where c.cv = toString(c.cv)  set c.cv = toInteger(c.cv)
return c
MATCH (c:Car) where c.year = toString(c.year)  set c.year = toInteger(
c.year) return c
MATCH (c:Car) where c.consumo_urbano = toString(c.consumo_urbano)  set
 c.consumo_urbano = toInteger(c.consumo_urbano) return c
```

Finalmente, el resultado de la base fue 11.871 nodos y 59.108 relaciones, a falta de añadir los nodos de cada usuario y la relacione entre los vehículos que le gusten. Esta es una pequeña muestra ya que no se puede representar entera:



Algoritmo de recomendacion

Análisis de resultados

DAFO

Líneas de futuro

Lecciones aprendidas

Bibliografia