Base de datos MongoDB para datos eléctricos

Alejandro Sanabria Cabeza
Alumno de Ingeniería del Software
Universidad de Sevilla
Sevilla, España
alesancab@alum.us.es

Fernando José González García
Alumno de Ingeniería del Software
Universidad de Sevilla
Sevilla, España
fergongar1@alum.us.es

ÍNDICE GENERAL

***Resumen*—**
El aumento de la demanda energética, el alto precio de la electricidad en España, el escaso número de centrales de producción energética no renovables que existe en España, la cifra de producción energética de las centrales renovables y no renovables. Estos hechos hacen más que necesario un estudio con un gran volumen de datos para analizar el motivo de los precios estratosféricos que pagamos por consumir energía en nuestro país.

En este proyecto nos centraremos en implementar una base de datos no relacional mediante el sistema de bases de datos "MongoDB" que incluya datos de la producción energética de los diferentes tipos de centrales en España, tanto renovables como no renovables. Además, realizaremos múltiples consultas sobre agregados sobre las colecciones creadas y representaremos gráficamente los resultados para extraer conclusiones sobre el problema planteado anteriormente.

***Palabras clave*—Base de datos no relacionales, NoSQL, MongoDB, pymongo, consumo eléctrico.**

I. INTRODUCCIÓN

En la actualidad, los precios que pagamos los españoles por consumir energía son bastante elevados. Día tras día podemos ver noticias como "*Las exportaciones a Francia encarecen la luz en España... y seguirán aumentando tras el tope*", "*Se dispara 500% el precio de la luz en España en tres años*", "*El Gobierno avanza en su plan de cierre de las nucleares pese al precio de la energía y el criterio de Bruselas*", etc. Para frenar esta crisis energética, urge realizar una investigación con un gran volumen de datos para sacar conclusiones de este gran problema que afecta a todos los españoles [1] [2].

Las bases de datos no relacionales se caracterizan por el buen rendimiento que presentan al albergar un gran volumen de datos. Como en este proyecto se utilizarán una gran cantidad de datos para presentar una tesis contrastable, qué solución más óptima que utilizar una base de datos no relacional puede existir.

A. Descripción del problema

En este proyecto se abarcan diversos problemas:

- **Despliegue e implementación de una base de datos no relacional MongoDB en cloud**: en este punto nos centraremos en la creación de una base de datos MongoDB y desplegarla en la nube mediante la herramienta MongoDB Atlas. Además, poblaremos la base de datos con dos datasets elegidos previamente.
- **Consultas mediantes agregados**: las operaciones basadas en agregaciones que nos proporciona MongoDB nos permite realizar operaciones sobre los datos almacenados para obtener información útil que usaremos para la demostración del estudio. La dificultad de este problema consistirá en lograr extraer datos que contrasten la teoría principal por la que se ha abordado este proyecto.
- **Representación y análisis de los resultados**: una vez realizados los agregados, procesaremos los resultados para representarlos gráficamente mediante la librería de Python "Matplotlib". Esto nos ayudará a realizar conclusiones mucho más efectivas ya que observaremos los resultados de una forma mucho más visual.

B. Objetivos

En este trabajo diferenciamos dos tipos objetivos. El primero de ellos es como trabajo de investigación, donde estudiamos distintas tecnologías y aplicamos su uso para sacar conclusiones sobre la electricidad en España:

- Investigación sobre los precios de la luz en España.
- Investigación sobre los diferentes tipos de centrales energéticas en España.
- Estudio sobre los beneficios que aportan las bases de datos no relacionales.
- Estudio sobre el sistema de gestión de bases de datos MongoDB.
- Estudio sobre el servicio de bases de datos en la nube MongoDB Atlas.
- Estudio sobre la herramienta de desarrollo en línea Google Colab.

Por otro lado, como trabajo de implementación, desarrollaremos el código necesario para usar las herramientas y sacar los datos para los objetivos previamente mencionados:

- Desplegar y poblar una base de datos con un dataset elegido previamente.
- Aplicar consultas mediante agregados a la base de datos creada para extraer información útil.
- Representación gráfica de los datos.

II. Descripción de la tecnología

En este apartado realizaremos una descripción de las tecnología con las que trabajamos para la implementación del proyecto. Comentaremos los beneficios de usar estas tecnologías, para qué sirven, funcionalidades que poseen, y demás.

A. No SQL

No SQL es un tipo de base de datos no relacional que almacena y accede a datos mediante pares clave-valor. En lugar de almacenar datos en filas y columnas como una base de datos tradicional, un SGBD NoSQL almacena cada elemento individualmente con una clave única. Además, una base de datos NoSQL no requiere un esquema estructurado que defina cada tabla y las columnas relacionadas.

Si bien las bases de datos relacionales (como MySQL) son ideales para almacenar datos estructurados, su estructura rígida dificulta agregar nuevos campos y escalar rápidamente la base de datos. NoSQL aporta un enfoque no estructurado o "semiestructurado" que es perfecto para capturar y almacenar contenido generado por el usuario. Esto puede incluir texto, imágenes, archivos de audio, videos, secuencias de clics, tweets u otros datos. Mientras que las bases de datos relacionales a menudo se vuelven más lentas e ineficientes a medida que crecen, las bases de datos NoSQL son altamente escalables. De hecho, puede agregar miles o cientos de miles de nuevos registros a una base de datos NoSQL con una disminución mínima del rendimiento.

Debido a la flexibilidad y escalabilidad de NoSQL, muchas grandes empresas y organizaciones han comenzado a utilizar bases de datos

NoSQL para almacenar datos de usuarios. Son especialmente comunes en las aplicaciones de computación en la nube y se han convertido en la solución de almacenamiento más popular para big data.

Por último, distinguimos 4 tipos de base de datos NoSQL:

Almacenamiento de datos clave-valor

Consiste en filas y columnas como una base de datos relacional, pero tiene sólo dos columnas: clave y valor. Cada clave es única y se utiliza para recuperar los valores asociados con él. Los almacenamientos son leves esquemas que no necesitan un modelo de datos fijo. Estos sistemas de almacenamiento de datos están diseñados para manejar datos masivos y la velocidad de consulta es más rápida que las bases de datos relacionales. Los SGBD Nosql de tipo clave-valor son Raik, Dynamo, Voldemort, Redis, Tokyo Cabinet.

Almacenamiento de datos orientado a columnas

Son similares a los SGBD de clave-valor, excepto que los datos se almacenan en columnas. La columna de datos relacionados se almacena en el mismo archivo que se llama familia de columnas. La familia de columnas contiene la clave de fila que consta de super columnas. Una súper columna es una columna que contiene otras columnas. Y una columna es un par de nombre y valor. Cassandra, Google's Big Table, Hyper Table y HBase son famosos SGBD orientados a columnas.

Almacenamiento de datos de documentos

En las bases de datos de documentos, los datos se almacenan en forma de documento con el formato de JSON, BSON o XML. El documento se almacena en las colecciones. Es un esquema liviano por lo que cualquier número de campos se puede agregar a la base de datos sin mantener espacios vacíos para otro documento en la colección. No hay necesidad de JOIN como en SGBDR debido a documentos incrustados y arreglos. Las bases de datos de documentos más populares son MongoDB, CouchDB, Terrastore, ThruDb.

Almacenamiento de datos gráficos

Las bases de datos de gráficas usan estructuras de gráficos como nodos, bordes para almacenar y representar los datos. Es un SGBD adyacente sin índice. Cada nodo está conectado al nodo adyacente. Son más adecuados para aplicaciones con muchas interconexiones como las redes sociales. OrientDB, HyperGraphDB, Neo4j, AllegroGraph son bases de datos populares de esta categoría.

B. MongoDB

MongoDB es una base de datos NOSQL de código abierto que cae bajo la clasificación de base de datos de documentos. Fue iniciado por la empresa 10gen. Fue escrito en C++. En MongoDB los documentos se almacenan en formato binario de JSON llamado formato BSON. BSON admite cadenas, enteros, fechas, booleanos, flotantes y tipos binarios. MongoDB es esquema liviano por lo que da la libertad al usuario para insertar nuevos campos al documento o actualizar la estructura anterior del documento. No usa JOINS como las bases de datos relacionales ya que tiene documentos incrustados a los que se puede acceder rápidamente. MongoDB también es compatible con Maestro-Esclavo donde los nodos esclavos contienen las réplicas de nodos maestros y se utilizan para copias de seguridad y lecturas. El clúster de MongoDB se compone de tres componentes:

- Nodo de fragmento: contiene uno o más fragmentos. Cada fragmento maestro tiene uno o más fragmentos esclavos. Los fragmentos esclavos contienen copias de los datos reales que se pueden utilizar en el momento de las fallas.
- La operación de lectura/escritura se realiza utilizando los datos presentes en el nodo maestro.
- Servidor de configuración: un grupo de servidores en MongoDB son llamados servidores de configuración. El papel de la configuración servidor es almacenar la información de enrutamiento y enviar la

solicitud al fragmento correcto.

Algunas características de MongoDB son las siguientes:

- Flexibilidad: MongoDB almacena datos en formato de documento utilizando JSON. Es un documento sin esquema y se asigna a tipos de lenguajes de programación nativos.
- Lenguaje de consulta enriquecido: Da la característica de SGBDR lo que estamos acostumbrados con características adicionales propias. Consultas dinámicas, clasificación, índices secundarios, ricas actualizaciones, fácil agregación, upsert (actualizar si el documento existe e insertar si no lo hace) son algunas de las características de SGBDR y flexibilidad y escalabilidad son los adicionales en MongoDB.
- Fragmentación: La auto fragmentación nos permite escalar nuestro clúster linealmente agregando más máquinas. Es posible aumentar la eficiencia que es muy importante en la web cuando la carga puede aumentar repentinamente y derribar el sitio web.
- Facilidad de uso: Es muy fácil de instalar, usar, mantener y configurar.
- Alto rendimiento: Proporciona datos de alto rendimiento en términos de persistencia. Reduce la actividad de E/S en el sistema de base de datos al soporte de documentos incrustados. El uso de la indexación admite consultas más rápidas.
- Alta disponibilidad: instalación de replicación de soporte de MongoDB llamado "ReplicaSet". ReplicaSet es un grupo de servidores que mantiene el mismo conjunto de datos. Proporciona conmutación por error automática, redundancia y mayor disponibilidad de datos.
- Soporte para múltiples motores de almacenamiento: Soporta múltiples motores de almacenamiento como el motor de almacenamiento WiredTiger, motor de almacenamiento MMAPv1. También es compatible con la API de motor de almacenamiento conectable que permite a terceros.

La tabla a continuación compara términos y los conceptos de una base de datos relacional con MongoDB.

Relational Database	**MongoDB database**
Fixed Schema	Schema less
Table	Collection
Rows, columns	Documents
Joins	Embedded documents
Vertical scaling	Vertical/Horizontal scaling

TABLA I: Diferencias MongoDB con DB relacional

Nuestro proyecto se basa en gran parte en las operaciones basadas en agregaciones que nos proporciona MongoDB. Estas operaciones, nos permiten procesar los datos que tenemos almacenados para obtener resultados que pueden ser de bastante utilidad.

Existen varias alternativas para poder utilizar estas operaciones, en concreto tres: tubería de agregación, operaciones de agregación con finalidad simple y la función MapReduce. En nuestro caso, usamos las tuberías de agregación, también conocidas como aggregation pipeline.

Para utilizar las tuberías de agregación utilizaremos el método aggregate sobre la colección con la que deseamos operar. El nombre de tubería se debe a que los datos pasan por etapas en el orden que se indique dentro del método agreggate, es decir, los datos resultantes de la primera etapa son la entrada de la siguiente etapa y así hasta la última etapa que retornará el resultado final.

```
db.collection.aggregate([{<etapa1>},
                {<etapa2>}, ... ]);
```

Existen multitud de operaciones dentro del tipo de tuberías de agregación. A continuación, describiremos las operaciones de las cuales hacemos uso en el proyecto [6].

- **$project**: esta función nos permite modificar los campos de la colección mostrando únicamente los datos que necesitemos.
- **$group**: gracias al comando group, podemos agrupar datos según una expresión dada.
- **$sort**: nos permite ordenar los datos obtenidos en la tubería de agregación. Este método recibe como parámetro el objeto con los campos sobre

los que se desee ordenar el resultado final. Tenemos la opción de ordenar de manera tanto ascendente como descendente.
- **$addFields**: sirve para añadir campos a una colección. Como parámetros recibe el nombre de los nuevos campos que se desean añadir y una expresión con la que se generará el nuevo campo.
- **$limit**: con este método podemos limitar los resultados de la agregación al número que especifiquemos como parámetro.
- **$match**: gracias a match podemos filtrar entre los datos de la colección según la consulta que introducimos con parámetro de entrada.
- **$out**: nos permite agregar los datos obtenidos mediante las etapas anteriores de una tubería a otra colección específica.
- **$lookup**: este operador puede ser el más complejo y curioso de todos los existentes. Es análogo a la función left join en SQL. Mediante un campo de la colección local, y otro de la colección que se quiere juntar, la instrucción compara los valores donde estos dos campos tienen el mismo valor, y los junta mediante un nuevo campo en la colección local, donde se insertan como array todas los documentos coincidentes de la colección ajena.
- **$replaceRoot**: Reemplaza todos los campos existentes en el documento de entrada, incluido el campo _id, con el documento especificado. Puede usarse para mover un documento incrustado existente (mediante la operación anterior, *$lookup*) a la raíz del documento, usandose conjuntamente con otros operadores como *$mergeObjects*.

C. PyMongo

PyMongo es una librería de Python que nos permite trabajar desde este lenguaje con la base de datos MongoDB, mediante operaciones CRUD o consultas [7]. En nuestro caso usaremos PyMongo para desarrollar en un entorno de Python, para poder procesar los datasets desde este, realizar la población a la base de datos, y obtener el resultado de las consultas, lo que nos permitirá representarla de manera gráfica mediante el uso de otras librerías que veremos a continuación.

D. Matplotlib

Matplotlib es una de las bibliotecas más exitosas y comúnmente utilizadas en Python, que proporciona varias herramientas para la visualización de datos en Python [10].

Se trata de una biblioteca multiplataforma de visualización de datos y trazado gráfico para Python y su extensión numérica NumPy. Como tal, ofrece una alternativa viable de código abierto a MATLAB. Los desarrolladores también pueden usar las API de matplotlib para incrustar gráficos en aplicaciones GUI. Proporciona al usuario la visualización de datos utilizando una variedad de diferentes tipos de gráficos para que los datos sean comprensibles. Puede usar estos diferentes tipos de gráficos (diagramas de dispersión, histogramas, gráficos de barras, gráficos de error, diagramas de caja, etc.) escribiendo unas pocas líneas de código en Python.

Utilizaremos esta librería para representar gráficamente los datos obtenidos mediante las tuberías de agregación.

III. Herramientas de desarrollo

A continuación describiremos las herramientas de desarrollo de las que hemos hecho uso para la implementación del proyecto. Las herramientas que hemos elegido para el desarrollo han sido MongoDB Atlas para el despliegue de la base de datos en la nube y Google Colab como herramienta de ejecución del código desarrollado.

A. MongoDB Atlas

MongoDB Atlas es un sistema de administración de bases de datos en la nube completamente personalizable que maneja toda la complejidad de implementar, administrar y reparar sus implementaciones en el proveedor de servicios en la nube de su elección (AWS, Azure y GCP) [5]. MongoDB Atlas es la mejor manera de implementar, ejecutar y escalar MongoDB en la nube. Con Atlas, tendrá una base de datos MongoDB ejecutándose con solo unos pocos clics

y en solo unos minutos.

Posee posibilidades de migración de bases de datos alojadas en otras plataformas muy efectivas y seguras. Además, proporciona extensas capas de seguridad. Tanto por el proveedor que suministra los servidores (AWS, Google Cloud o Azure) como por la misma plataforma. Ofrece todo tipo de soluciones en cuanto a recursos contratados dependiendo de la escala de la base de datos y el volumen de acceso a ella y también ofrece la posibilidad de adquirir un plan gratuito bastante completo.

El motivo por el cuál hacemos uso de este servicio es por su comodidad y eficiencia. Gracias a esta herramienta hemos sido capaces de lograr crear una base de datos MongoDB común sobre la que trabajar desde cualquier sitio. Además, consumiendo los recursos de la plataforma logramos ahorrar recursos para conseguir una mayor eficiencia del entorno de ejecución utilizado.

B. Google Colab

Google Colab también conocido como Google Colaboratory, es un servicio de Google que nos permite escribir y ejecutar código del lenguaje Python en el navegador que prefiramos [8]. Está orientado para el desarrollo de Inteligencia Artificial (como aprendizaje automático), análisis de datos y educación. Hablando desde un punto de vista más técnico, Google Colab es al fin y al cabo un sistema que permite alojar cuadernos de Jupyter que no requiere instalación previa y es totalmente gratuito.

Gracias a la suite de google, todos los cuadernos se almacenan en la aplicación de almacenamiento en la nube Google Drive. Además, también se puede sincronizar directamente desde GitHub.

IV. Implementación

A continuación se procede a dar una explicación de la implementación realizada. El código se puede observar y ejecutar de manera online en Google Colab.

A. Datasets

En este trabajo, se utilizan dos datasets para poblar las colecciones y realizar consultas.

El primero de ellos es un dataset de dominio público que nos aportará información sobre la producción energética segmentada por distintos tipos de centrales, así como la demanda y precio de la energía [3]. Estos datos fueron tomados en España, desde el año 2015 hasta el 2018. Los campos que usaremos en este trabajo son los siguientes:

- '*Time*': Indica la marca de tiempo (Fecha y hora) en el que se hace el registro. Se hace un registro cada hora.
- '*Price actual*': El precio de la electricidad, medido en €/MWh.
- '*Total load actual*': La demanda energética, en MW.
- Producción por tipo de central: Abarca 21 campos, cada uno correspondiente a la energía producida por un tipo concreto de central a esa hora, medida en MW. En estas columnas hay muchos valores 0 o nulos, correspondientes a horas en las que esas centrales no produjeron energía.

El segundo corresponde a los datos del consumo eléctrico de una vivienda. Estos datos se pueden descargar desde la página de la compañía eléctrica a la que se esté contratada. En concreto se usan tres datos:

- Fecha: El día en el que se hace el registro.
- Hora: Tramo horario en el que se hace el registro (tramos de una hora).
- Consumo: La energía consumida en Wh.

B. Creación DB en MongoDB Atlas

Para crear la base de datos utilizamos el servicio en la nube *MongoDB Atlas*, que nos ofrecerá un despliegue de una base de datos MongoDB que podremos usar de manera conjunta mediante Google Collab. Para poder utilizarla, nos registramos en la página de MongoDb [5] y, tras crear un nuevo proyecto, creamos un clúster mediante la opción gratuita, eligiendo región y proveedor.

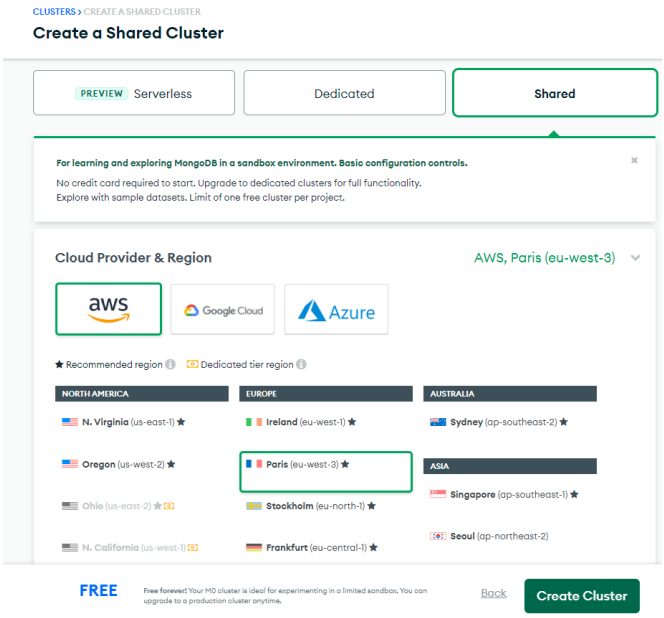

Fig 1. Creación de un clúster en MongoDB Atlas.

Una vez creada, para permitir el acceso a nuestra DB desde cualquier dirección, desde el panel de control vamos al menú “Network Access”, pulsamos la opción “Add IP Address” y añadimos la dirección IP 0.0.0.0/0. A continuación, necesitamos crear un usuario para proporcionar las credenciales, para lo que vamos al menú “Database Access” y pulsamos la opción “Add New Database User”. Nos dan a elegir entre varios metodos de autenticación, seleccionamos el metodo mediante contraseña y creamos a un nuevo usuario. Por defecto tiene todos los permisos de lectura y escritura.

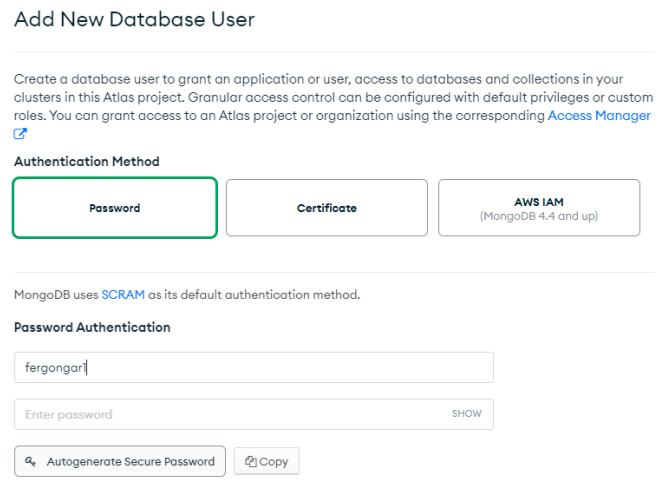

Fig 2. Creación de un usuario en MongoDB Atlas.

A continuación, desde el menú principal seleccionamos la opción “Connect” del clúster recién creado, y desde la ventana que se nos abre seleccionamos “Connect your application”. Tras ello nos pedirá el lenguaje y versión que utilizamos en nuestro proyecto, y nos dará un string que podemos usar para la conexión.

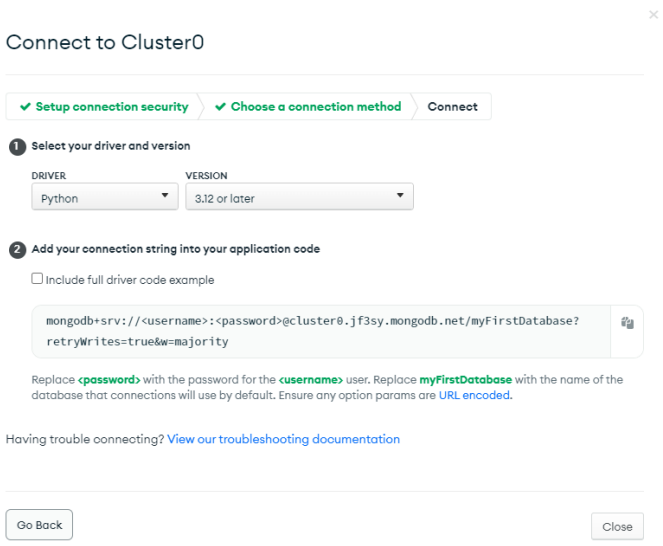

Fig 3. Conexión mediante MongoDB Atlas.

Por último, desde el menú principal se puede usar la opción “Browse collections” del clúster, que nos otorgará una interfaz gráfica con la que operar nuestras bases de datos. Desde ahí, creamos nuestra DB, a la que llamamos *ElectricDB*.

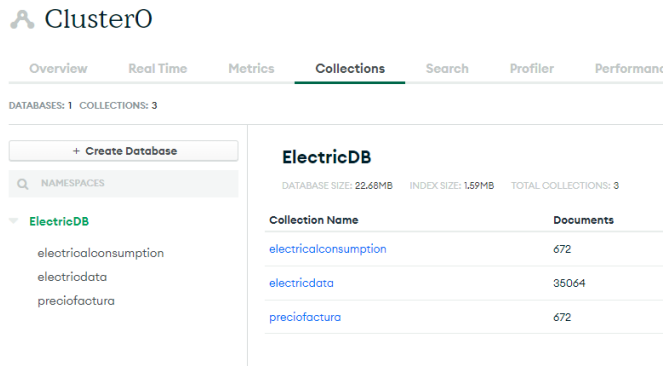

Fig 4. Interfaz gráfica de MongoDB Atlas.

C. Conexión a la DB con pymongo y MongoDB Atlas

En primer lugar es necesario la instalación de la librería pymongo, que se puede realizar con el siguiente comando:

```
python -m pip install pymongo[srv]
```

Una vez creada y configurada la base de datos en MongoDB Atlas, obtenemos un string de conexión como se explicó en el apartado anterior. Este string posee el siguiente formato:

```
"mongodb+srv://<usuario>:<password>
@<nombre-cluster>.mongodb.net/
<nombre-bd>?retryWrites=true&w
=majority"
```

Con este string podemos realizar la conexión al clúster de MongoDB Atlas, y consecuentemente, a la base de datos de la siguiente forma:

```
import pymongo
client = pymongo.MongoClient(
    <key-mongodb-atlas>)
db = client.ElectricDB
```

A continuación se leen los ficheros csv para volcar los datos en la DB. Antes de realizarlo, se le hace un preprocesado a los datos. Las marcas de tiempo se formatean como tipo datetime para poder hacer operaciones con éstas en las consultas. Además, como se ha mencionado anteriormente, algunos de los campos en el fichero de producción energética tenían valores de 0 o nulos. Esto sería un problema en un modelo SQL tradicional, ya que todas las entradas de una tabla deben de tener las mismas columnas. Sin embargo, se puede aprovechar NoSQL para borrar este campo de las entradas en las que este valor no sea significativo, reduciendo así la cantidad de datos a almacenar.

Los datos se guardan como una lista con un diccionario por cada entrada, y se guardan en la colección de la siguiente manera:

```
consumo = db.electricalconsumption
consumo.insert_many(ec_data)
```

D. Consultas de agregados

A continuación se procede a explicar las diferentes consultas realizadas. Para ello, se expondrá el motivo por el que se considera interesante la consulta, se explicará la tubería de agregación definida para ello, y se expondrán gráficamente los resultados, que se comentarán brevemente.

Evolución del precio y la demanda energética durante un día.

El precio de la luz varía a lo largo del día, según varios factores, principalmente la demanda energética. Es conocido que en las horas de madrugada se consume menos electricidad, y que las horas de mayor demanda son las horas centrales del día, correspondientes al horario laboral. Se puede realizar una consulta para ver los datos del precio y la demanda, y comprobar cómo estas situaciones se dan en la práctica.

La tubería de agregación definida para obtener la evolución del precio durante un día contiene las siguientes etapas:

- Con '*$match*' filtramos los documentos para obtener únicamente los del día indicado, seleccionando un rango de fechas con los operadores '*$gte*' y '*$lt*'.
- Usando '*$project*' nos quedamos con los dos únicos campos de interés para la consulta, 'price actual' y 'time'.
- Finalmente, con '*$sort*' ordenamos de menor a mayor las fechas, de manera que los datos queden de 0:00 a 23:00.

La figura a continuación muestra los resultados seleccionando el día 7 de enero de 2017, correspondiente a un día laboral:

Fig 5. Precio de la luz a lo largo del día.

Se puede ver claramente como el precio se mantiene bajo por la noche y sube en torno a las 9:00, coincidiendo con la entrada al trabajo. Después de una bajada el precio vuelve a subir en torno a las 19:00, hora en la que empieza a oscurecer y se enciende la iluminación y los diferentes usos domésticos como calefacción, televisión, cocina...

Para obtener la demanda, hacemos una consulta similar pero seleccionando el campo de 'total load actual' mediante '*$project*'. En la siguiente figura vemos los resultados obtenidos para el mismo día, 7 de enero de 2017:

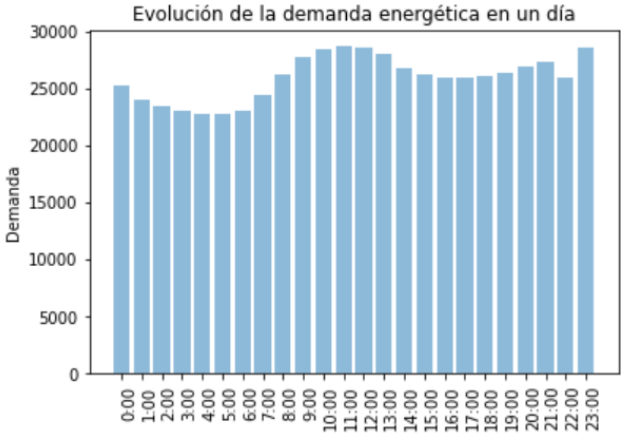

Fig 6. Demanda eléctrica a lo largo del día.

Donde se puede ver que aunque la relación no es exacta, la curva posee una forma muy similar.

Evolución del precio de la luz en un año.

El precio de la luz varía no solo a lo largo de las horas del día, si no que también al pasar de los días, por lo que es interesante ver como este precio fluctúa a través de un año. Para ello definimos la siguiente tubería:

- Filtramos con '*$match*' los documentos del periodo indicado, en este caso el año 2017.
- Añadimos un campo con el mes del documento mediante la instrucción '*$addFields*'.
- Agrupamos estos datos según el campo recién creado, con el uso de '*$group*'. Calculamos el precio medio de la electricidad en ese mes mediante el operador '*$avg*'.
- Por último usamos '*$sort*' para ordenar los documentos de enero a diciembre.

Los resultados obtenidos son los siguientes:

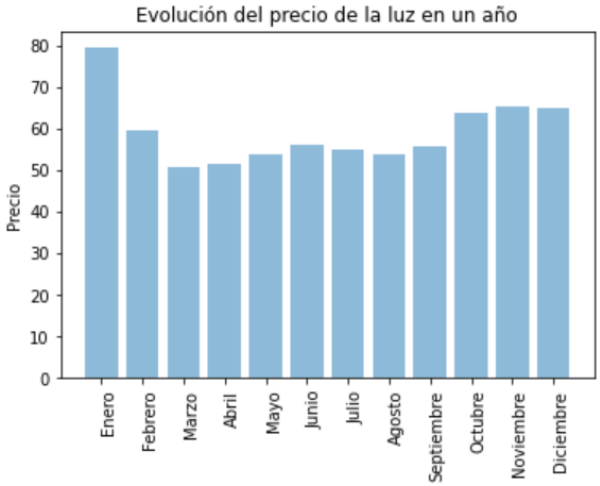

Fig 7. Demanda eléctrica a lo largo del día.

Donde se puede ver que para ese año el mes más caro fue enero.

Producción de distintas fábricas a lo largo de un día.

Otro factor importante en el precio de la luz es la producción generada por los distintos tipos de centrales. No todas las centrales producen con el mismo ritmo: las centrales eólicas dependen del viento que se produzca ese día, las solares solo producirán en las horas diurnas, las centrales térmicas producen a demanda, y las centrales

nucleares producen siempre una cantidad estable.

Para fijar el precio de la luz, se hace una subasta horaria, en la que distintas centrales ofertan su energía a un determinado precio. Centrales como la solar o la eólica, que no tienen un coste de materia prima, ofrecen sus energías a precios más bajos, mientras que las centrales térmicas fijarán un precio mayor para que el gasto de combustible les salga rentable. Las ofertas se ordenan de mayor a menor precio, y , mediante una estimación de la demanda, quedan fuera del reparto aquellas con mayor precio que no fuesen necesarias para cubrirla, mientras que el precio del resto se fija a la de la mayor oferta de las restantes.

Por todo esto, es muy interesante poder obtener los datos asociados a la producción de los distintos tipos de centrales. Para ello, se realiza se define la siguiente tubería de agregación:

- Mediante '*$match*' filtramos los documentos para obtener únicamente los del día indicado.
- Con '*$project*' filtramos todos los campos que no sean de interes, manteniendo solo la fecha y la producción de las distintas centrales.
- Finalmente, con '*$sort*' volvemos a ordenar las fechas de mayor a menor.

Los resultados obtenidos se recogen en la siguiente figura. Junto a la gráfica se aporta una leyenda que indica en que color se representa cada tipo de central.

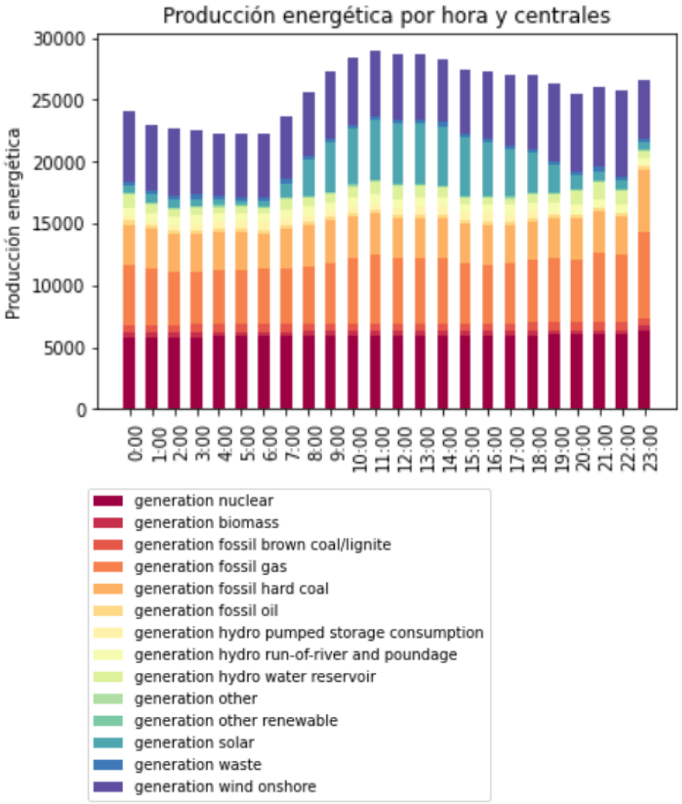

Fig 8. Producción eléctrica a lo largo del día.

Podemos ver como la producción total se ajusta a la demanda previamente vista. Se puede observar como la central nuclear, en rojo, produce una cantidad estable de energía, mientras que centrales como la eólica y la solar, en morado y azul respectivamente, varían considerablemente a lo largo del día.

Día de mayor producción.

No todos los días se consume y produce la misma electricidad, este valor puede variar por factores como las horas de luz, que provoquen más consumo en iluminación, la temperatura, que provoque más gastos en uso de aire acondicionado... Por ello puede ser interesante como referencia obtener un máximo de energía producida.

Para esto se realiza se define la siguiente tubería:

- Utilizando '*$addFields*' separamos la fecha de las horas, obteniendo año día y mes.
- Mediante '*$group*' agrupamos por año, mes y día para juntar los 24 documentos asociados

a cada día. Sumamos la energía producida en todo ese día para cada central mediante el operador '*$sum*'.
- Volvemos a utilizar '*$addFields*' para añadir un campo que contenga el total de la energía producida englobando todas las centrales.
- Usamos '*$sort*' para ordenar de mayor a menor según el valor recién obtenido para el total.
- Por último, mediante '*$limit*' limitamos los resultados a uno para obtener solo el día de mayor producción.

Haciendo esta consulta se obtiene que este día fue el 17 de enero de 2017, en el que se produjeron 909035 MW en total. Esto coincide con un día de invierno, por lo que se podría deber a las menores horas de luz y al uso de la calefacción.

Producción media de distintas fabricas en un periodo de tiempo.

Como se ha mencionado anteriormente, la producción energética, a excepción de la estable central nuclear, varía mucho siguiendo factores ambientales, en las centrales de energías renovables, o de la demanda, en las centrales fósiles. Por esto puede ser interesante ver la producción no de un solo día específico, si no la media de un periodo.

Para ello se realiza la siguiente consulta:

- Con el uso de '*$match*' seleccionamos únicamente los documentos del periodo deseado.
- Mediante '*$addFields*' se añade un campo de hora del día a la que se toma el registro.
- Utilizando '*$group*' agrupamos los documentos según el campo de hora previamente añadido. Además añadimos la media de la producción de cada central durante esa hora.
- Por último usamos '*$sort*' para ordenar de mayor a menor según la id (Que con el '*$group*' previo se había establecido como la hora), de manera que los datos queden de 0:00 a 23:00.

Usando como periodo el mes de enero de 2017, se obtienen los siguientes resultados:

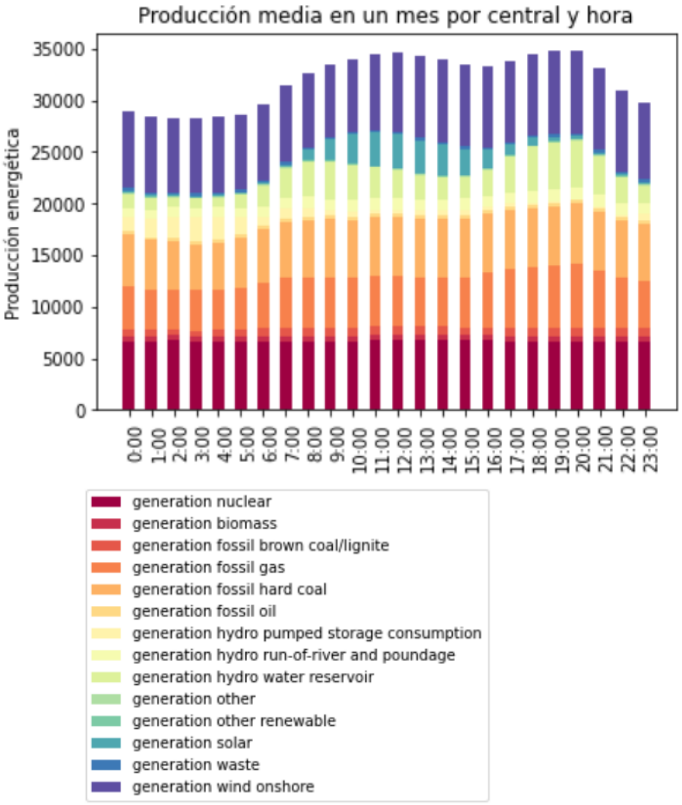

Fig 9. Producción energética media.

Comparando a la figura anterior, donde se medía la producción de un único día en específico, se puede ver que de media ese mes se produjo mucho menos energía solar, y considerablemente más hidráulica y fósiles.

Energía renovable producida.

Las energías renovables son una cuestión que ha tomado mucha relevancia en los últimos años, en búsqueda de una producción sostenible y limpia. Podemos ver como actúa España al respecto viendo el porcentaje de su energía total producida correspondiente a energías renovables durante los años 2015 hasta el 2018.

Para ello, se realiza se define la siguiente tubería de agregación:

- Se emplea '*$group*', con '*_id*' None, para agrupar todos los documentos de la colección, y sumar la energía total producida por cada tipo de central.

- Seguidamente, se utiliza '*$project*' para obtener como resultado la suma de todas las energías de centrales renovables, y la suma de las no renovables.

Los resultados obtenidos son los siguientes:

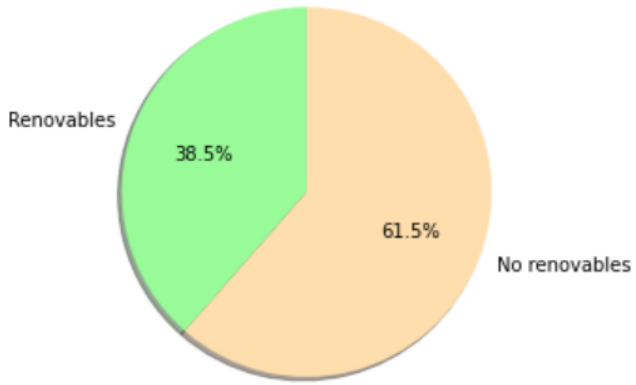

Fig 10. Producción de energías renovables.

Se puede observar que en España un 38.5% de la energía producida fue renovable durante los años 2015 hasta el 2018. Teniendo en cuenta que, para 2017, un 30% de la energía producida en Europa era renovable, esto nos coloca por encima de la media [4].

Factura eléctrica.

Para esta consulta usamos un segundo conjunto de datos, correspondientes al consumo de un hogar desde el 21 de febrero hasta el 20 de mayo. Teniendo los datos tanto del consumo como el precio de la energía por hora, podemos calcular la factura correspondiente. La tubería definida para ello es la siguiente:

- Con la instrucción '*$lookup*' añadimos a los documentos correspondientes a la colección del consumo un nuevo campo, que contendrá el documento asociado al precio de la luz en esa misma hora.
- Mediante '*$replaceRoot*' se llevan a la raíz todos los campos del documento recién añadido.
- Se usa '*$project*' para quedarse unicamente con los datos del consumo y la hora. Además se añade un nuevo campo '*coste*', en el que se calcula el coste de esa hora multiplicando el consumo por el precio, mediante el operador '*$multiply*'. Cabe destacar que estan en unidades distintas, Wh y MWh respectivamente, por lo que hace falta multiplicar el resultado por 0,000001 para convertir unidades. El resultado se aproxima a las ocho unidades decimales mediante el operador '*$round*'.
- Por último, el resultado se guarda en una nueva colección, *preciofactura*, mediante la instrucción '*$out*'.

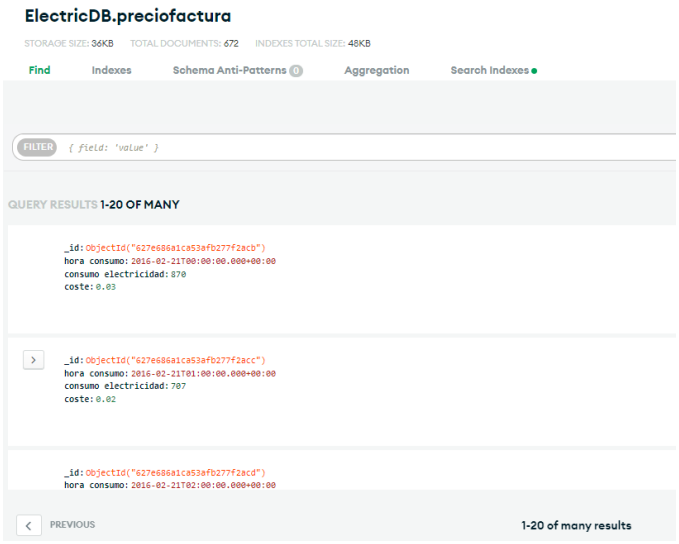

Fig 10. Colección creada desde el agregado vista desde la interfaz gráfica.

Desde esta colección podemos calcular el precio total con otra sencilla consulta:

- Se emplea '*$group*', con '*_id*' None, para agrupar todos los documentos de la colección, y sumar los costes de cada hora.
- Seguidamente, se utiliza '*$project*' para obtener un único campo, el precio total de la factura, usando el operador '*$round*' para aproximar el resultado a las dos unidades decimales.

Haciendo esta consulta sobre los datos introducidos, se obtiene un precio total de 19,25€. Aunque este precio no incluye impuestos, comparándola con una factura actual sin duda se hace notar la subida en el precio de la electricidad en los últimos años.

V. Conclusiones

Para dar cierre a este proyecto de investigación y auto formación en tecnologías con las que no estábamos familiarizados haremos dos conclusiones. Por una parte conclusiones sobre la tecnología usada a lo largo del desarrollo del proyecto y por otra conclusiones en cuanto a los resultados obtenidos.

A. Conclusiones sobre la tecnología

El crecimiento de la tecnología en el mundo actual ha dado lugar a la necesidad de bases de datos NOSQL que puedan almacenar datos estructurados, no estructurados y semiestructurados. Los sistemas de gestión de bases de datos no relacionales son fáciles de usar y los datos dinámicos pueden ser fácilmente almacenados en ellos. Los documentos son independientes por lo que mejora el rendimiento y disminuye los efectos secundarios de la concurrencia. En conclusión, el nacimiento de estos tipos de bases de datos son necesarios para el continuo avance de la tecnología actual. Vivimos en un mundo de cantidades de datos inimaginables donde sin este tipo de tecnología estaríamos perdidos.

B. Conclusiones sobre los resultados obtenidos

Gracias a las gráficas obtenidas mediante las consultas a estos datos, se puede conseguir un conocimiento más profundo de la demanda energética y como se establecen los precios por hora.

Por otro lado, vemos que aunque España es uno de los países con más energía verde de Europa, una importante parte de su producción son energías fósiles. Estas energías, aunque pueden producirse a demanda y no dependen de factores meteorológicos y ambientales como las energías renovables, tienen la gran desventaja de requerir un uso de combustible, con el coste extra que esto supone, y creando el problema adicional de la falta de suministro, problema muy grave actualmente debido a los conflictos políticos. Todo esto sin mencionar la contaminación que producen. Además, considerando que una alta parte de la energía consumida en España es importada, y gran parte de esa energía es también fósil, el problema se acentúa.

Los altos precios que se pagan por el consumo de electricidad en España comparado con la mayoría de países ha pasado a ser unos de los mayores problemas a los que se enfrenta nuestro país. Gracias a nuestro estudio, podemos ver ahora a alguno de los culpables. La apuesta por más energías verdes es sin duda una gran alternativa a largo plazo, ya no solo de cara a los precios, si no también a la salud del planeta. Por otro lado, también hemos visto que estas centrales tienen una producción muy irregular, por lo que la energía nuclear se presenta como una buena opción a corto plazo, con una producción estable de energía más barata y mucho menos contaminante que las fósiles.

REFERENCIAS

[1] https://www.larazon.es/economia/20220412/wsxritqdfbeaxpvy2q4n43b4qm.html
[2] https://www.jornada.com.mx/notas/2021/10/08/economia/se-dispara-500-el-precio-de-la-luz-en-espana-en-tres-anos/
[3] https://www.kaggle.com/datasets/nicholasjhana/energy-consumption-generation-prices-and-weather
[4] https://gruposinelec.com/las-renovables-generan-el-30-de-la-electricidad-de-europa/
[5] https://www.mongodb.com/es/atlas/database
[6] https://www.mongodb.com/docs/manual/reference/operator/aggregation/
[7] https://pymongo.readthedocs.io/en/stable/
[8] https://research.google.com/colaboratory/intl/es/faq.html
[9] https://aws.amazon.com/es/nosql/
[10] https://matplotlib.org/