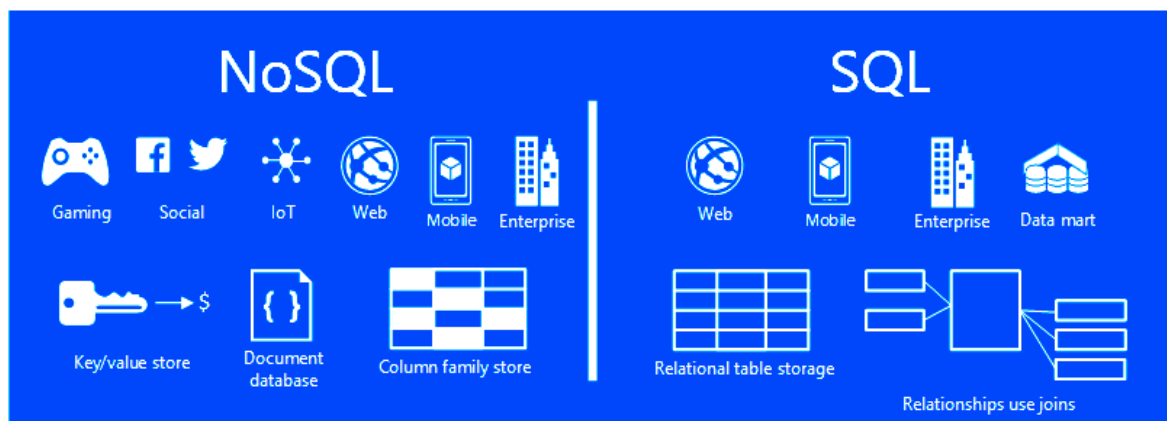


Trabajo BDs NoSQL



Realizado por:
Romero David

ÍNDICE

1. Introducción.
2. Bases de Datos NoSQL
 - 2.1. Definición.
 - 2.2. Características principales.
 - 2.3. Servicios destacables.
 - 2.4. Usos.
 - 2.5. Estructuras.
 - 2.6. Comparativas entre SQL y NoSQL.
 - 2.7. Ejemplos de BBDD NoSQL.
 - 2.8. Ejemplos de empresas u organizaciones que las usan.
 - 2.9. Dato interesante.
 - 2.10. Ejemplo de caso de uso.
 - 2.10.1. Imágenes.
 - 2.10.2. Tabla de comandos, principales, desde la consola MongoSH .
3. Conclusión final.

1. Introducción

Para poder comparar entre ambas bases de datos, primero veamos una breve descripción de cada una:

- Base de datos SQL (Structured Query Language) significa lenguaje de consulta estructurado, que es el lenguaje estándar utilizado para consultar y manipular datos en una base de datos relacional que organiza los datos en tablas con filas y columnas.
- Las NoSQL (Not Only SQL) son un tipo de base de datos que se aleja del modelo relacional tradicional y están diseñadas para manejar grandes volúmenes de datos con mayor flexibilidad y escalabilidad. A diferencia de las bases de datos SQL, que organizan la información en tablas con relaciones estrictas, las NoSQL permiten estructuras más dinámicas y adaptables.

Como vemos, la principal diferencia radica en la forma de estructurar los datos, siendo la primera más estricta en su estructura, con tablas, filas y columnas, frente a la otra que da más opciones a la hora de elegir la estructura de la misma, siendo más dinámica y adaptable según la necesidad. Por otra parte, la integridad y consistencia de los datos son mayor en las SQL con respecto a la NoSQL.

2. Bases de datos NoSQL

2.1 Definición:

Las bases de datos NoSQL (Not Only SQL) son un tipo de almacenamiento de datos diseñado para ser más flexible y escalable que las bases de datos relacionales tradicionales. Se utilizan comúnmente en aplicaciones que manejan grandes volúmenes de datos o requieren alta disponibilidad.

2.2 Características principales:

Escalabilidad horizontal: Se distribuyen fácilmente en múltiples servidores, permitiendo manejar grandes cantidades de datos sin depender de una única máquina potente.

Estructuras de datos flexibles: A diferencia de las bases de datos relacionales, NoSQL permite almacenar datos sin esquemas rígidos, lo que facilita cambios en la estructura sin afectar la funcionalidad.

Alto rendimiento: Están optimizadas para operaciones rápidas, lo que las hace ideales para aplicaciones en tiempo real.

Distribución y replicación: La mayoría de las bases de datos NoSQL soportan la replicación automática, lo que mejora la disponibilidad y resistencia ante fallos.

2.3 Servicios destacables:

Las bases de datos NoSQL sirven para almacenar y gestionar grandes volúmenes de información de manera eficiente, especialmente en aplicaciones modernas que requieren escalabilidad y flexibilidad.

2.4 Usos:

Las bases de datos NoSQL son clave en aplicaciones como redes sociales, análisis de big data, IoT, aplicaciones en tiempo real y más.

2.5 Estructuras:

Las bases de datos NoSQL se presentan en una variedad de tipos según su modelo de datos. Los tipos principales son:

- Almacenes de clave-valor: los datos se almacenan en un formato no estructurado con una clave única para recuperar valores. Los ejemplos son Redis y DynamoDB.
- Bases de datos de documentos: los datos se almacenan en formato de documento, como JSON. Los ejemplos son MongoDB y CouchDB.
- Bases de datos de gráficos: los datos se almacenan en nodos y bordes, optimizados para las relaciones de datos. Los ejemplos son Neo4j y JanusGraph.
- Bases de datos en columnas: los datos se almacenan en columnas en lugar de filas. Algunos ejemplos son Cassandra y HBase.

2.6 Comparativa entre SQL y NoSQL

	SQL	NoSQL
Modelo de datos	Relacional(Tablas)	No relacional (clave-valor, documento, familia de columnas, gráfico)
Esquema	Esquema fijo con estructura predefinida	Esquema dinámico, estructura flexible
Escalabilidad	Escalamiento vertical (agregar recursos a un solo servidor)	Escalamiento horizontal (distribución entre múltiples nodos)

Manejo de transacciones	Sigue las propiedades ACID* para mayor confiabilidad	Sigue el teorema CAP**, a menudo priorizando la disponibilidad y la tolerancia de partición.
Rendimiento	Optimizado para consultas y transacciones complejas	Optimizado para almacenamiento de datos a gran escala y análisis en tiempo real
Lenguaje de consulta	Utiliza SQL (lenguaje de consulta estructurado)	Varía según la base de datos (por ejemplo, consultas tipo JSON en MongoDB, CQL en Cassandra)
Flexibilidad	Estructura rígida con relaciones	Sin esquema o semiestructurado para mayor adaptabilidad
Casos de uso	Sistemas financieros, ERP, CRM	Aplicaciones de big data, IoT, análisis en tiempo real

*Atomicidad, Consistencia, Aislamiento y Durabilidad

**Consistencia, Disponibilidad y Tolerancia a Particiones.

2.7 Ejemplos de BBDD NoSQL:

- De clave-valor:

- Redis, abreviatura de «Remote Dictionary Server», es un sistema de gestión de bases de datos de código abierto, NoSQL de clave-valor.
- CouchDB, es un DBMS de clave-valor, de código abierto. Está optimizado para usarlo en dispositivos móviles gracias a sus capacidades de sincronización y replicación. Permite trabajar offline mientras la conexión de red no está disponible.
- BigTable, es un sistema de gestión de bases de datos completamente gestionado, de clave-valor y tabular. Está diseñado para grandes cargas de trabajo analíticas y operativas. Forma parte del portfolio de Google Cloud y, como tal, se utiliza en muchas de las aplicaciones de Google como Google Analytics, Google Maps o Gmail..

- Documentales:

- Elasticsearch, es un motor de analíticas y búsqueda RESTful, basado en la librería Lucene. Elasticsearch es el sucesor de un motor de búsqueda anterior llamado Compass, también diseñado por Shay Banon.
- MongoDB, es un DBMS de código abierto, NoSQL y orientado a documentos. MongoDB Inc. ofrece una suite integrada de servicios cloud de bases de datos, así como soporte comercial. Se suele usar para almacenar grandes volúmenes de datos..

- Orientadas a grafos:

- FlockDB.
- InfiniteGraph, es un DBMS orientado a grafos, multiplataforma, escalable y habilitado para la nube. Está pensado para soportar un rendimiento alto y optimizado para grandes conjuntos de datos, altamente conectados y complejos.
- OpenLink Virtuoso.

- Tabulares:

- HBase, es un sistema de gestión de bases de datos tabular, de código abierto. Al igual que Cassandra, está modelado a partir de BigTable. Está pensado para proporcionar una forma tolerante a fallos de almacenar grandes cantidades de datos dispersos. HBase forma parte del proyecto Hadoop de Apache.
- BigTable.
- Cassandra, es un DBMS gratuito, de código abierto y tabular —modelado a partir de BigTable—. Está diseñado para soportar grandes volúmenes de datos y garantizar alta disponibilidad sin puntos únicos de fallo.

- Orientadas a objetos:

- Object DB, es un DBMS orientado a objetos, multiplataforma. Requiere usar una de las APIS estándar de Java: JPA (Jakarta Persistence) o JDO (Java Data Objects). Está pensado para proporcionar un mejor rendimiento y aplicaciones más rápidas. Además, se presenta como la opción más productiva para desarrollar aplicaciones de BBDD Java usando la API Java Persistence.
- Zope Object Database (ZODB) es un sistema de gestión de BBDD orientado a objetos. Está diseñado para almacenar objetos Python de modo transparente y persistente. Aunque está incluido en el servidor de aplicación web de Zope, también se puede usar de forma independiente.
- Realm, es una base de datos orientada a objetos diseñada para aplicaciones móviles. Se considera una alternativa a SQLite, ya que no utiliza SQL y ha sido desarrollada desde cero.

2.8 Ejemplos de empresas u organizaciones que las usan:

PayPal: Usa bases de datos NoSQL para análisis en tiempo real y detección de fraudes.

Domino's Pizza: Aprovecha NoSQL para personalización de campañas de marketing.

Sky: Utiliza Couchbase para gestionar contenido multimedia y garantizar disponibilidad.

Viber: Ha conectado a más de mil millones de usuarios en todo el mundo a través de llamadas de audio y vídeo de alta calidad, mensajería y mucho más.

2.9 Dato interesante:

Los sistemas de bases de datos NoSQL crecieron con las principales redes sociales, como Google, Amazon, Twitter y Facebook. Estas tenían que enfrentarse a desafíos con el tratamiento de datos que los tradicionales sistemas de gestión de BBDD relacionales no solucionaban. Con el crecimiento de la web en tiempo real existía una necesidad de proporcionar información procesada a partir de grandes volúmenes de datos que tenían unas estructuras horizontales más o menos similares. Estas compañías se dieron cuenta de que el rendimiento y sus propiedades de tiempo real eran más importantes que la coherencia, en la que las bases de datos relacionales tradicionales dedicaban una gran cantidad de tiempo de proceso.

2.10 Ejemplo de caso de uso:

Para este ejemplo de caso de uso, usaré MongoDB y su aplicación MongoDB Compass junto con Docker.

2.10.1 Imágenes

A continuación muestro en imágenes los procesos realizados en este caso de uso:

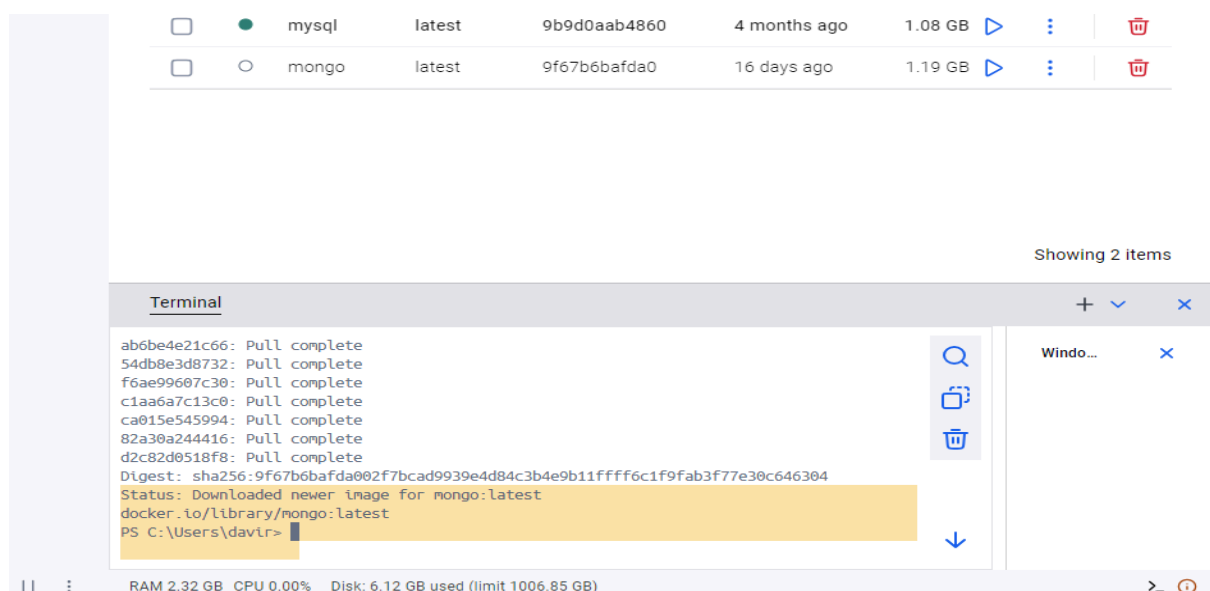


Imagen 1. Descarga Imagen Mongo en Docker

Uso del comando → `Docker pull mongo`. Este descarga la última imagen disponible desde el sitio oficial de MongoDB.

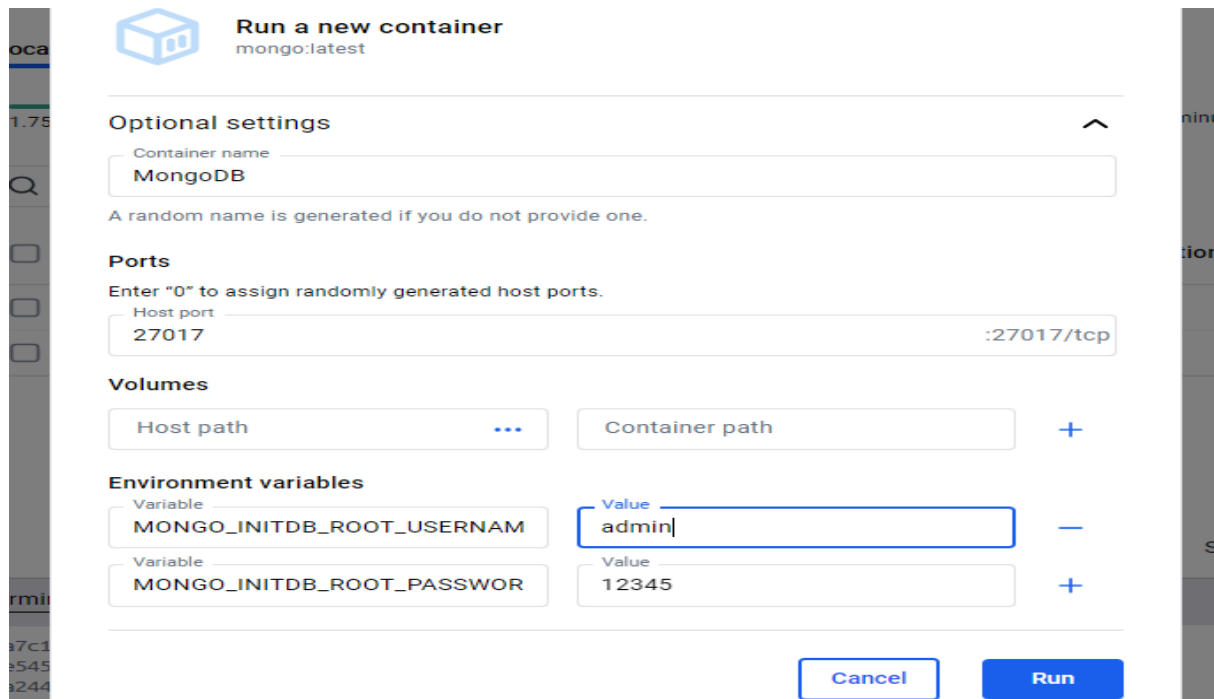
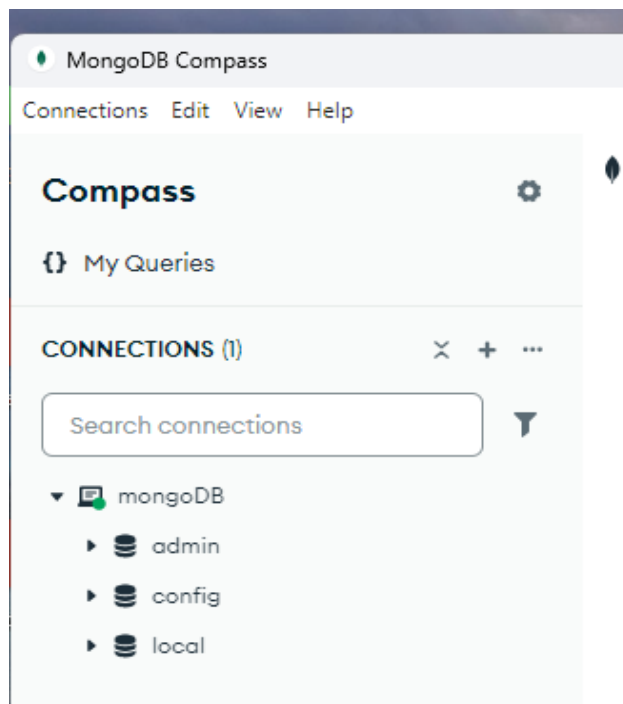


Imagen 2. Arranque del contenedor.

Desde Docker Desktop pulsamos el botón Run (Símbolo Play) y rellenamos los campos que consideremos oportunos (Imagen 2), tales como nombre del contenedor, nombre usuario y su contraseña. Y de obligada asignación, indicamos el puerto al que conectar. Usando el asignado por defecto 27017.



Pasos para conectar MongoDB Compass al contenedor:

1. Arrancar el contenedor en docker.
2. Abrir MongoDB Compass
3. Click derecho sobre la base de datos, y pulsamos conectar.

Imagen 3. Conexión en MongoDB Compass

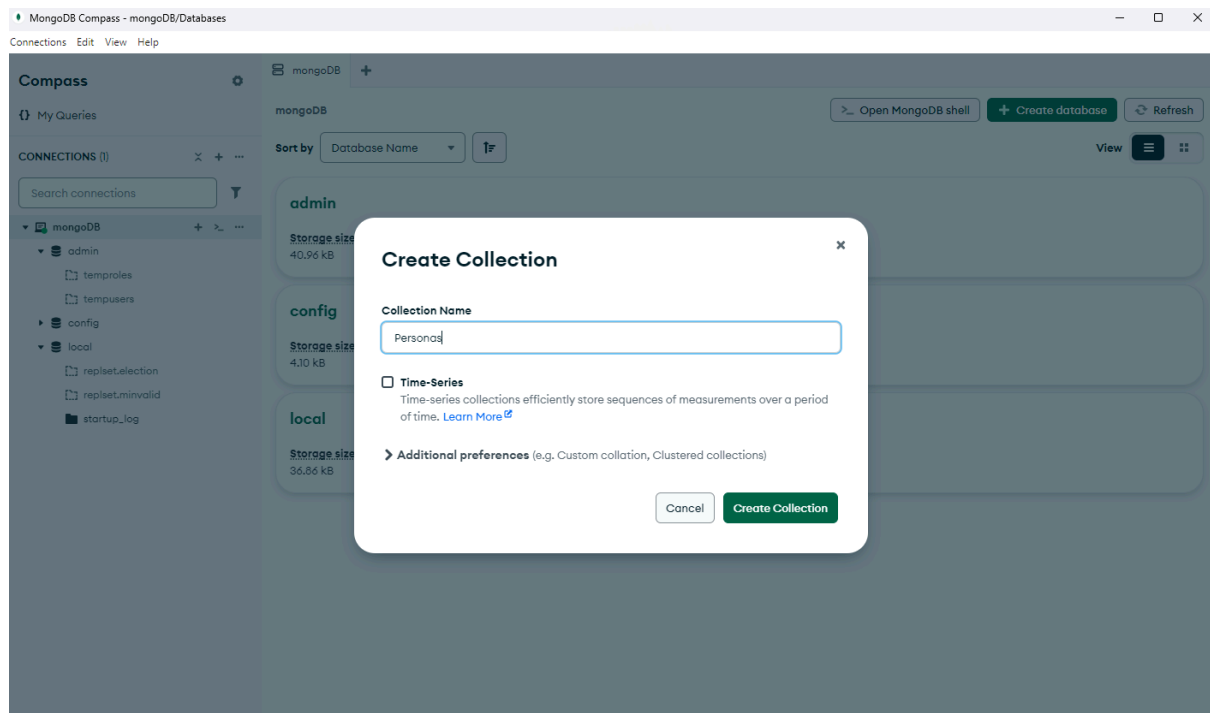


Imagen 4. Crear colección.

Para crear una colección, sin que haya ningún parecido en su estructura, se podría decir que es el equivalente a una tabla en SQL.

Para llegar a la imagen 4, hay varios caminos, siendo el más directo el botón + en la base de datos local para crear la colección. Y ya después simplemente indicas el nombre, también hay opciones avanzadas, y se pulsa crear colección.

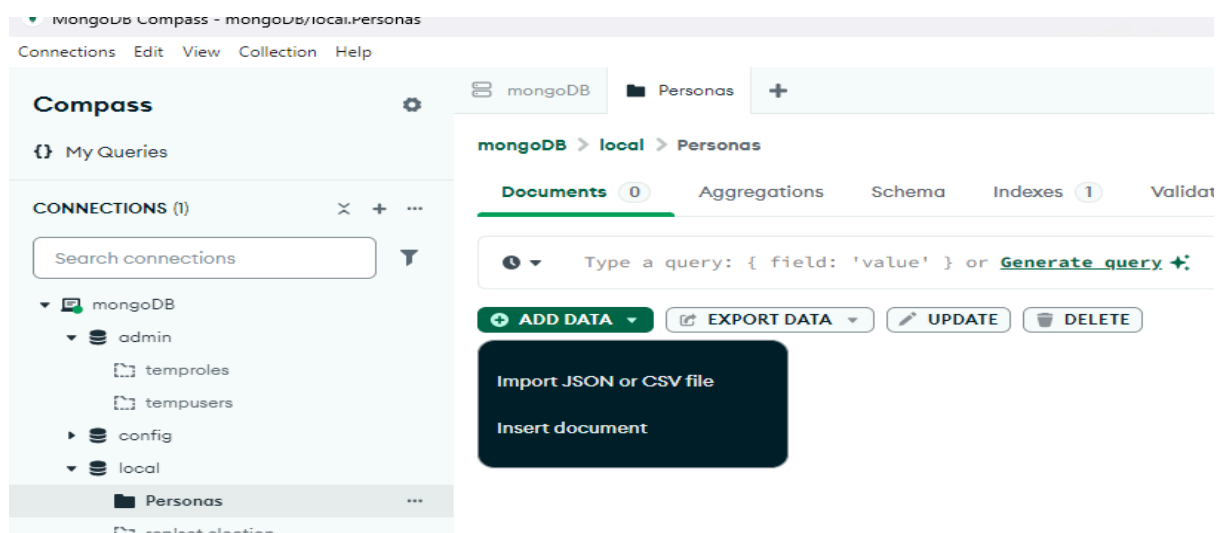
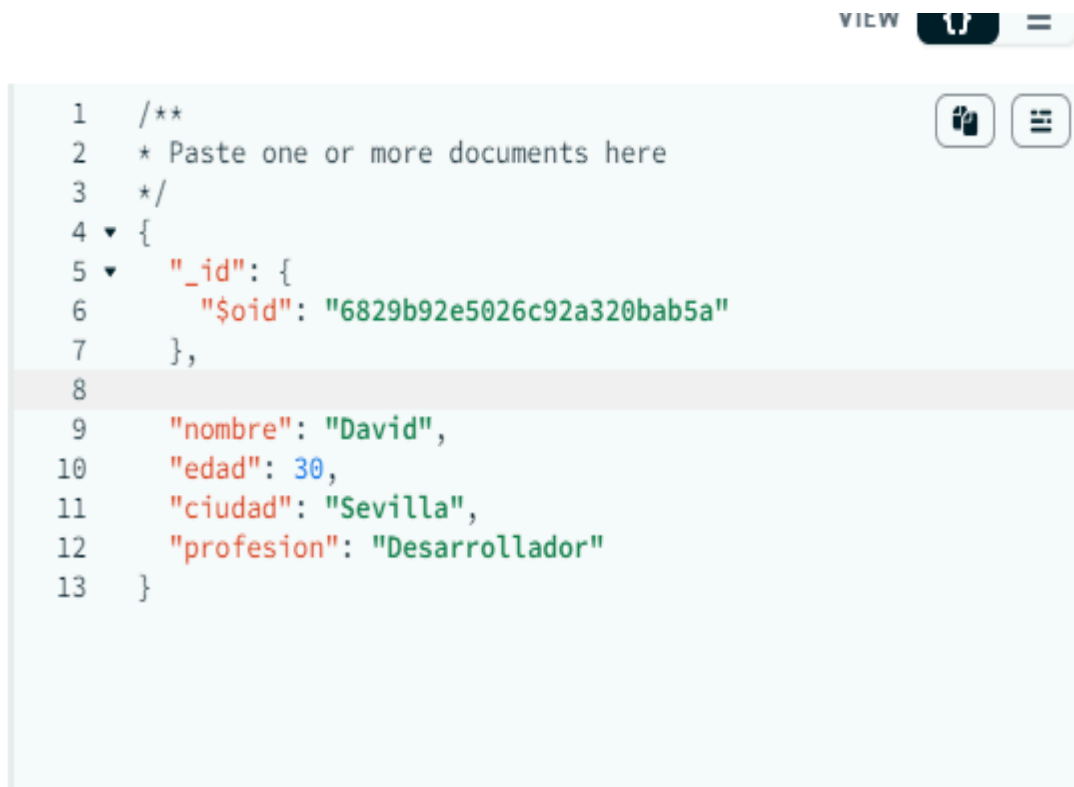


Imagen 5. Insertar documento

Una vez creada la colección Persona, pulsado ADD DATA, se elige entre Importar archivo o insertar documento (sintaxis JSON).



```
1  /**
2  * Paste one or more documents here
3  */
4  {
5    "_id": {
6      "$oid": "6829b92e5026c92a320bab5a"
7    },
8
9    "nombre": "David",
10   "edad": 30,
11   "ciudad": "Sevilla",
12   "profesion": "Desarrollador"
13 }
```

Imagen 6. Introducir datos en el documento.

Una vez elegido insertar documento, se escribe el archivo en formato JSON. Para este ejercicio inserte 4. Aquí, igual que pasa con las colecciones, se podría decir que su equivalente en SQL sería cada fila con todas sus columnas.

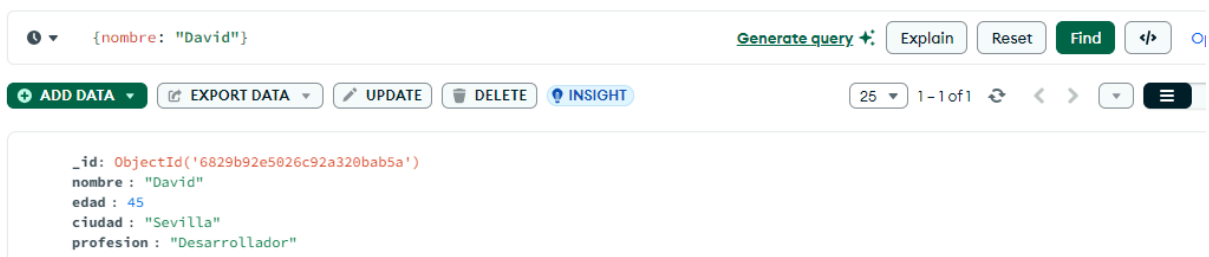


Imagen 7. Búsqueda por nombre.

La interfaz de usuario ofrece un campo específico para realizar consultas en la parte superior, cuando estás en la ventana de la colección. En este caso, la búsqueda se realiza, por como están guardados los datos por clave-valor.

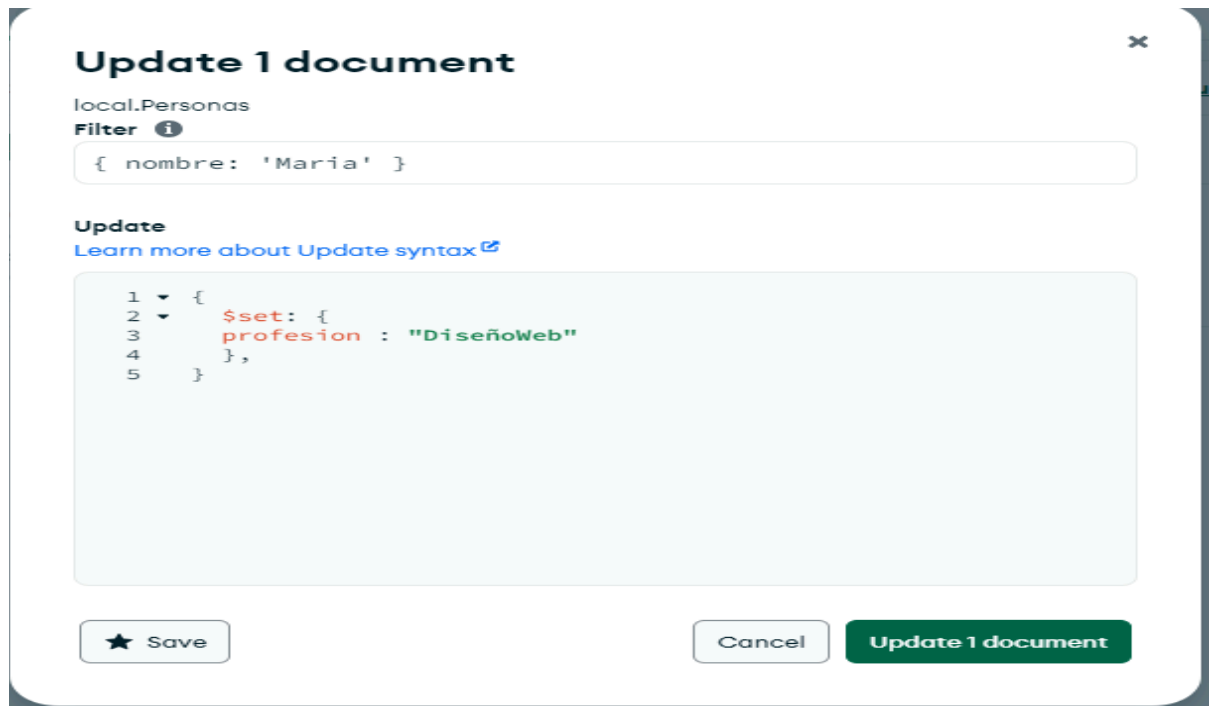


Imagen 8. Modificar datos mediante update.

Otra de las opciones de la interfaz, cuando estás en una colección, es el botón update que permite realizar modificaciones en los datos del documento. Se puede realizar sobre un documento o varios a la vez. Si quieres hacer algún tipo de filtrado, se realiza primero según lo visto en el apartado anterior (después aparece reflejado como se puede ver en la imagen 8). Al fin y al cabo, este botón de update no hace otra cosa que preparar un comando \$set.

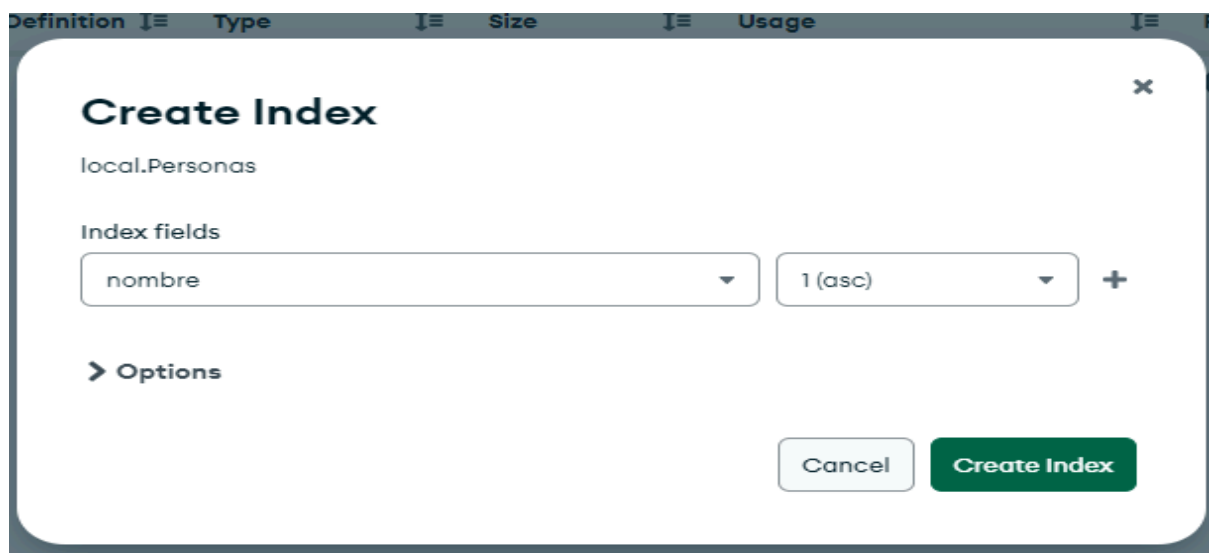


Imagen 9. Crear índice.

Principalmente se usa por si interesa crear algún índice (como referencia extra) además del Id por defecto.

```
>_MONGOSH
profesion: 'Desarrollador'
}
{
  _id: ObjectId('6829ba3f5026c92a320bab5b'),
  nombre: 'María',
  edad: 30,
  ciudad: 'Sevilla',
  profesion: 'DiseñoWeb'
}
{
  _id: ObjectId('6829ba7d5026c92a320bab5c'),
  nombre: 'Antonio',
  edad: 25,
  ciudad: 'Huelva',
  profesion: 'Editor Gráfico'
}
{
  _id: ObjectId('6829bb115026c92a320bab5d'),
  nombre: 'Lidia',
  edad: 48,
  ciudad: 'Sevilla',
  profesion: 'Tester'
}
> db.Personas.getIndexes()
< [
  { v: 2, key: { _id: 1 }, name: '_id_' },
  { v: 2, key: { nombre: 1 }, name: 'nombre_1' }
]
local>
```

Imagen 10. Uso del terminal MongoSh. Consultar índices.

Si bien, parece que la interfaz de usuario de MongoDB, es bastante completa e intuitiva, existe una consola de comandos propia, llamada MongoSh. Y por ahí se pueden realizar todos los pasos descritos en este ejercicio y más.

```
>_MONGOSH
> db.Personas.aggregate([
  { "$group": { "_id": "$ciudad", "total": { "$sum": 1 } } }
])
< {
  _id: 'Huelva',
  total: 1
}
{
  _id: 'Sevilla',
  total: 3
}
local>
```

Imagen 11. Uso del terminal MongoSh. Agrupar por ciudad.

Aquí uso el comando `getIndexes` para mostrar cuántos índices tiene la colección para la imagen 10 y el comando `aggregate` (el equivalente a `group by`) para agrupar por ciudad y saber cuantas hay de cada grupo en la imagen 11.

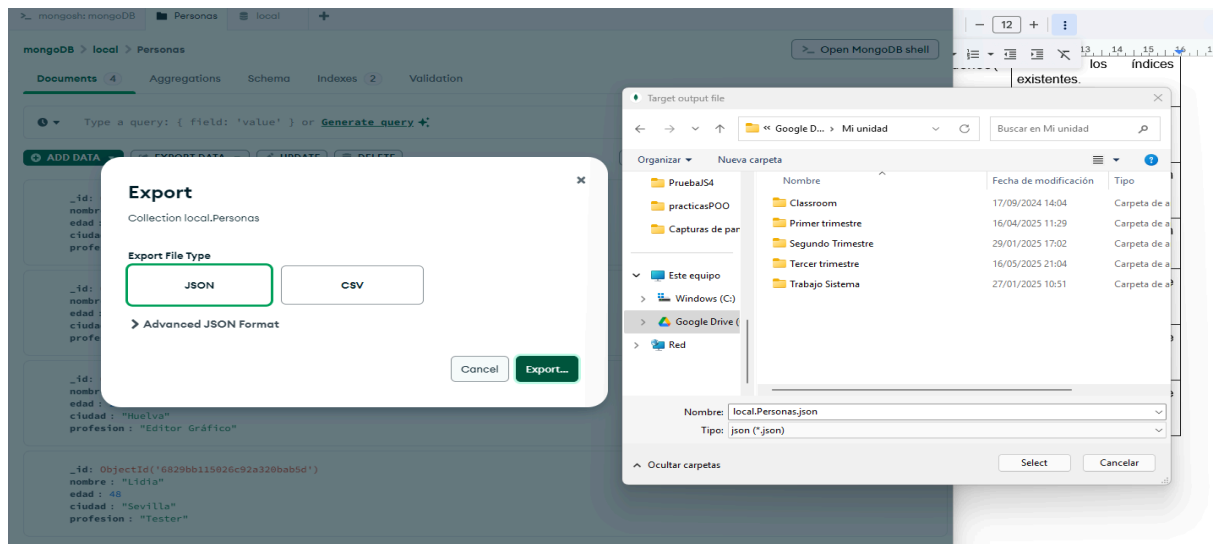


Imagen 12. Exportar un archivo.

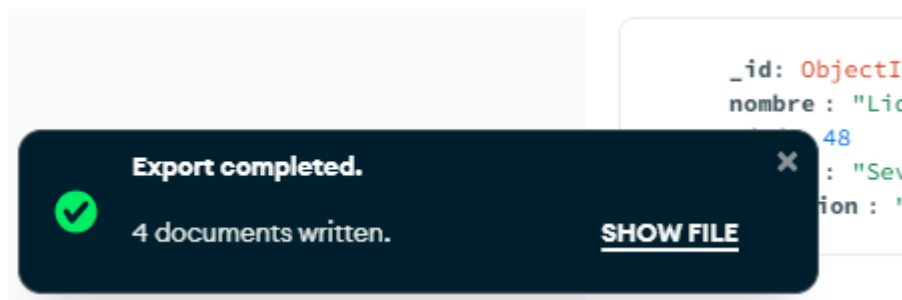


Imagen 13. Confirmación exportación exitosa.

Aquí guardo (exporto) el archivo en mi almacenamiento interno. Para ello se debe pulsar el botón exportar, e indicar el tipo de archivo (JSON o CSV) y la ruta donde quieres que se guarde (Imagen 12)..

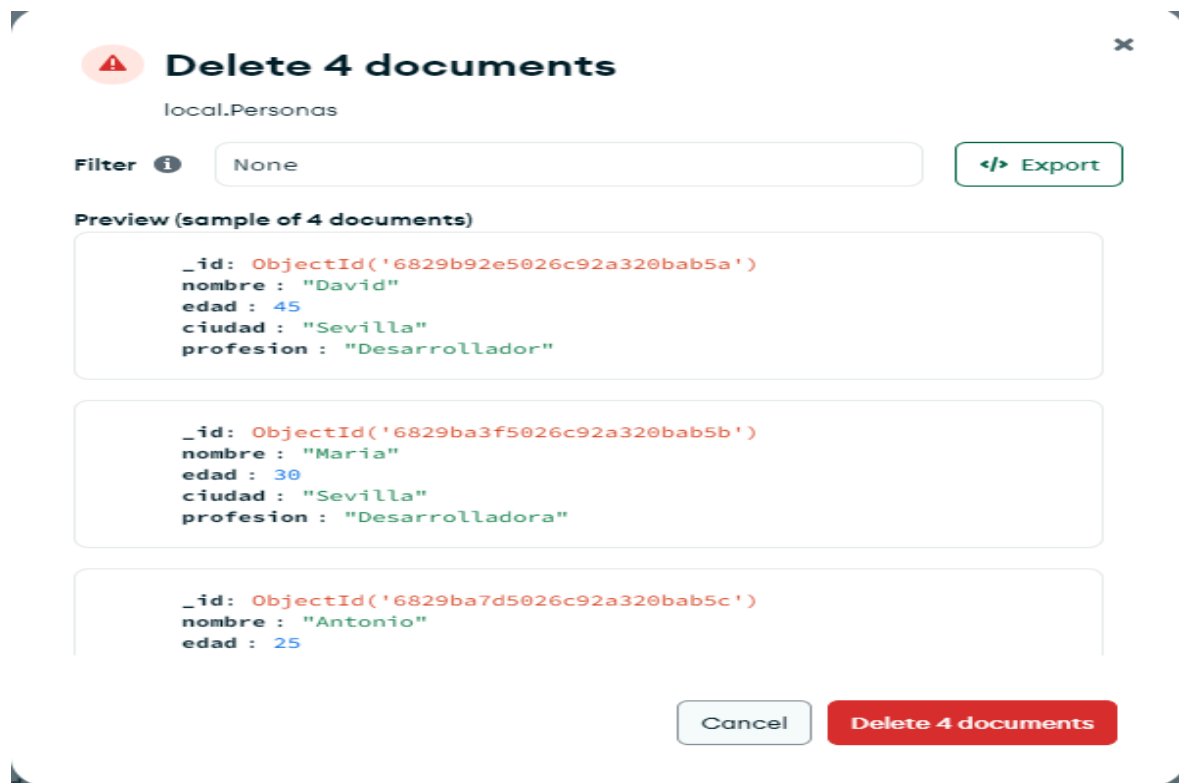


Imagen 14. Borrar documentos.

Se puede usar la barra de búsquedas para filtrar antes de borrar uno o varios documentos. Una vez filtrados, de ser requerido, pulsamos el botón Delete.

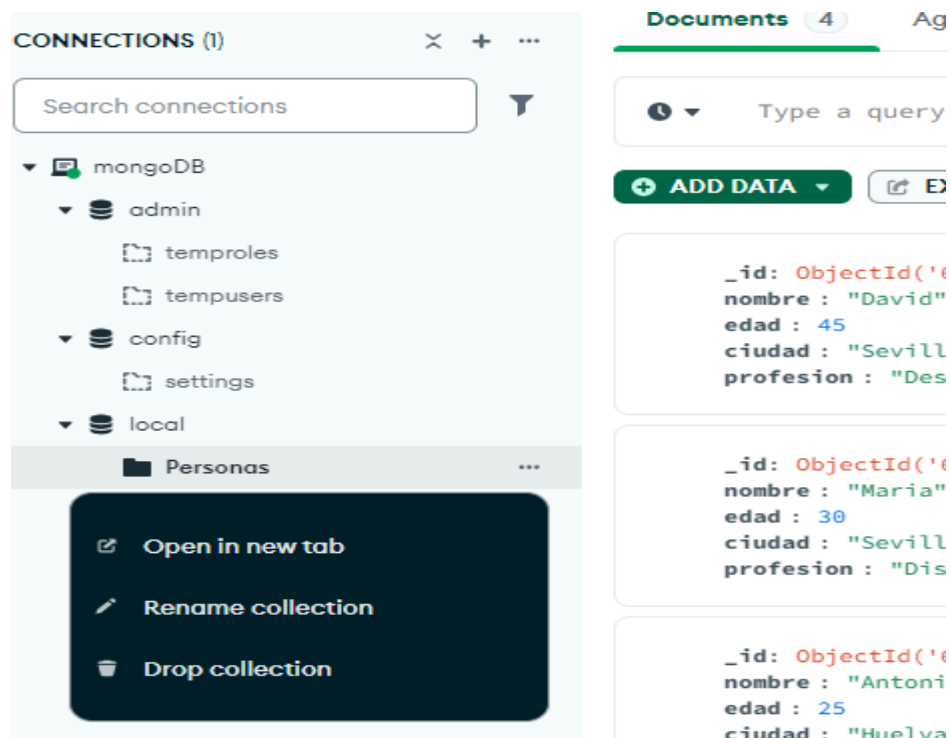


Imagen 15. Borrar Colección

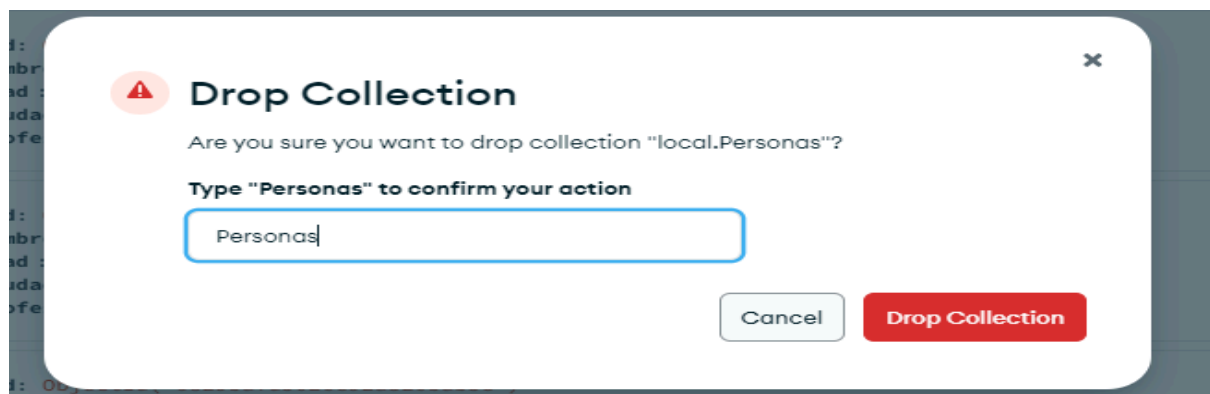


Imagen 16. Confirmar borrar colección.

Si quieres eliminar una colección entera, pulsamos con doble click sobre la colección deseada y pulsamos Drop Collections (Imagen 15) y tras verificar que se desea realizar el borrado (Imagen 16) estaría todo listo y la colección eliminada.

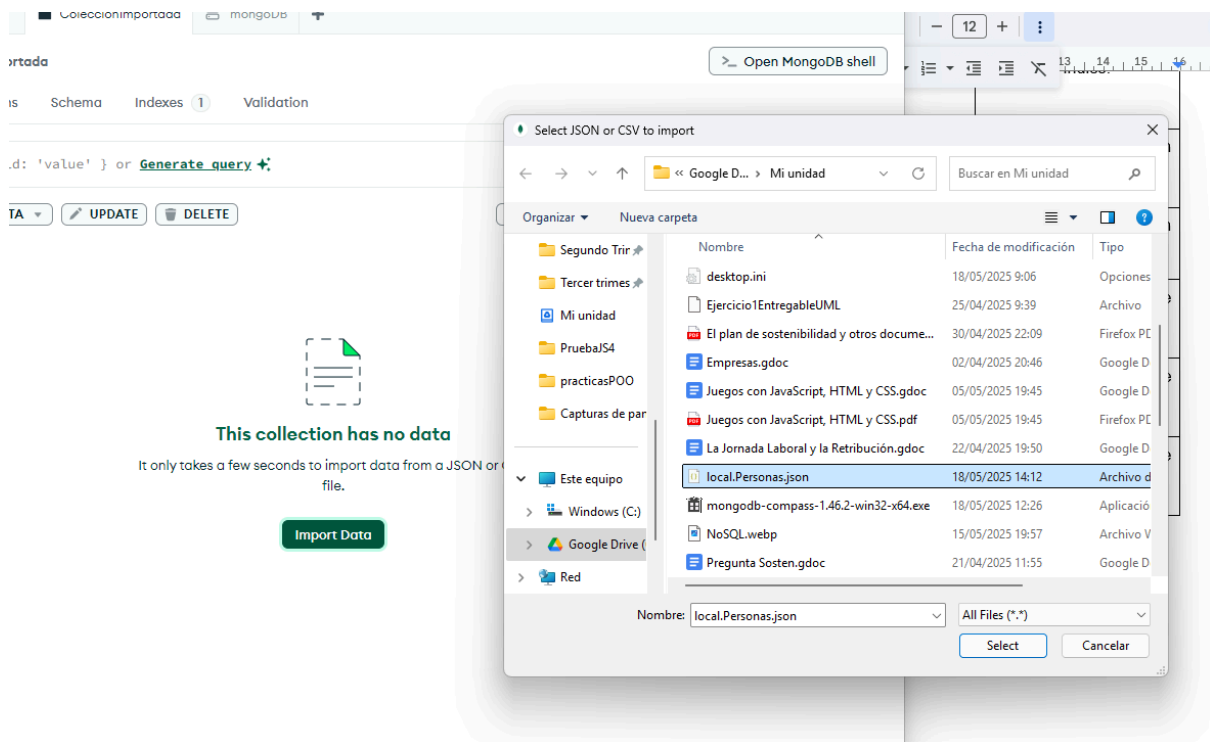


Imagen 17. Importar archivo.

Nos situamos en la colección (vacía) donde queremos importar el archivo JSON o CSV, y pulsamos Import Data y se nos abre una ventana para poder elegir el archivo deseado y su ruta.

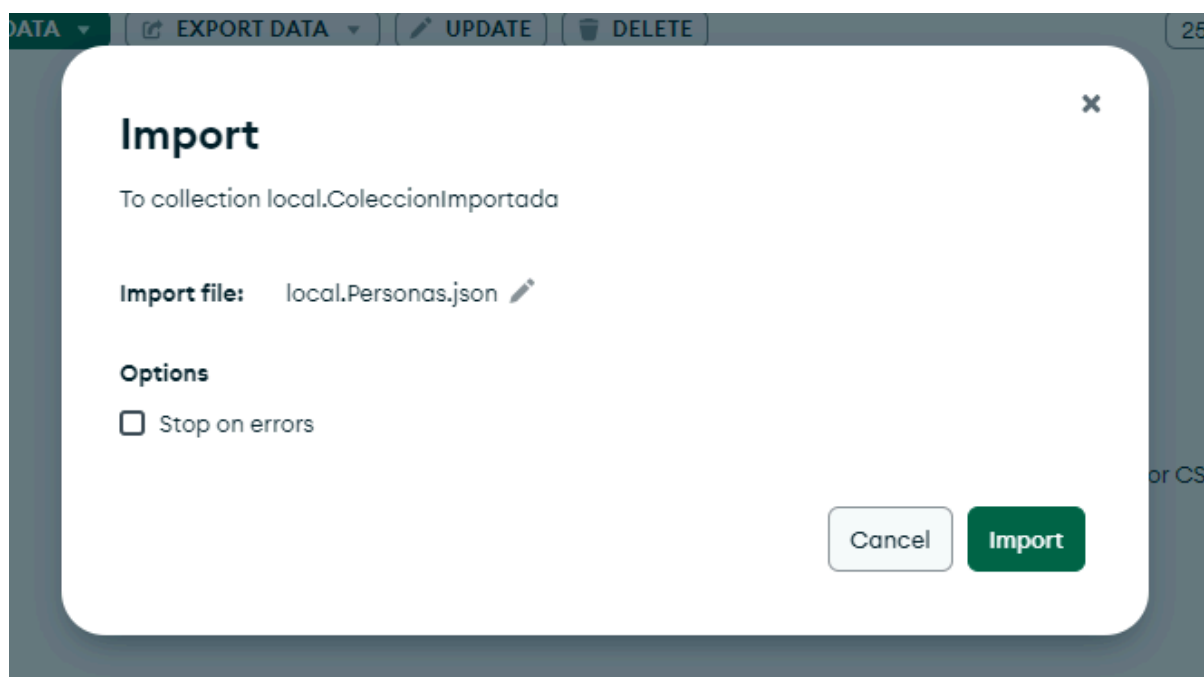


Imagen 18. Confirmar importación.

Una vez confirmado los pasos, archivo a importar correcto y colección de destino correcta, simplemente pulsamos Import y ya tendríamos nuestro archivo cargado.

2.10.2 Tabla de comandos, principales, desde la consola MongoSH

Categoría	Comando	Descripción
Conexión	mongosh	Inicia la consola de MongoDB.
	use miBaseDatos	Cambia a una base de datos específica.
Ver bases y colecciones	show dbs	Lista todas las bases de datos disponibles.
	show collections	Lista las colecciones dentro de la base activa.
Operaciones CRUD	db.miColeccion.insertOne({...})	Inserta un documento en una colección.

	<code>db.miColeccion.insertMany([...])</code>	Inserta varios documentos a la vez.
	<code>db.miColeccion.find({})</code>	Recupera todos los documentos.
	<code>db.miColeccion.find({campo: valor})</code>	Busca documentos con un filtro específico.
	<code>db.miColeccion.updateOne({...})</code>	Modifica un solo documento.
	<code>db.miColeccion.updateMany({...})</code>	Modifica múltiples documentos.
	<code>db.miColeccion.replaceOne({...})</code>	Reemplaza un documento completamente.
	<code>db.miColeccion.deleteOne({...})</code>	Elimina un solo documento.
	<code>db.miColeccion.deleteMany({...})</code>	Elimina varios documentos según el filtro.
Índices	<code>db.miColeccion.createIndex({campo: 1})</code>	Crea un índice para optimizar búsquedas.
	<code>db.miColeccion.getIndexes()</code>	Muestra los índices existentes.
	<code>db.miColeccion.dropIndex("nombre_1")</code>	Elimina un índice.
Agregaciones	<code>db.miColeccion.aggregate([...])</code>	Ejecuta una agregación avanzada.

Usuarios y permisos	<code>db.createUser({...})</code>	Crea un usuario con permisos específicos.
	<code>db.runCommand({ connectionStatus: 1 })</code>	Muestra el estado de conexión del usuario.
Exportar e importar	<code>mongodump --db miBaseDatos --out /backup</code>	Exporta una base de datos.
	<code>mongorestore --db miBaseDatos /backup</code>	Restaura una base de datos desde un backup.
	<code>mongoexport --db miBaseDatos --collection Personas --out personas.json --jsonArray</code>	Exporta un documento específico.
	<code>mongoimport --db miBaseDatos --collection Personas --type=csv --headerline --file personas.csv</code>	Exporta un documento específico.

3. Conclusión Final

Las bases de datos NoSQL se usan, y crecieron, para manejar gran cantidad de datos, por lo que la escalabilidad y agilidad de la misma es importante. Frente a la integridad y marcada estructura de las relacionales. No hay un lenguaje estandarizado, como en las relacionales, incluso algunos tienen sintaxis propia o se basan en JSON, BSON. Y además permite, o se divide, en múltiples formas de estructuras de datos.