

**Memoria Trabajo Práctico 2 (TP2)**

Bases de Datos Documentales: MongoDB

***Alumno****: Francisco Javier Piqueras Martínez*

***Asignatura****: Gestión y Almacenamiento de Información no Estructurada*

***Fecha******de******entrega****: 20 de abril de 2020*

Índice

[1. Descripción del documento 4](#_Toc38185065)

[2. Ejercicios 4](#_Toc38185066)

[2.1. Ejercicio 1: Caso práctico Blog de Noticias. 4](#_Toc38185067)

[2.1.1. Diseño de la base de datos documental. 5](#_Toc38185068)

[2.1.2. Definición de índices 6](#_Toc38185069)

[2.1.3. Inserciones 6](#_Toc38185070)

[2.1.4. Dificultades encontradas y opinión del ejercicio 6](#_Toc38185071)

[2.2. Ejercicio 2: Consultas sobre la colección de visitas a webs. 7](#_Toc38185072)

[2.2.1. Enunciado del ejercicio 7](#_Toc38185073)

[2.2.2. Realización del ejercicio 8](#_Toc38185074)

[2.2.3. Dificultades encontradas y opinión del ejercicio 8](#_Toc38185075)

[2.3. Ejercicio 3: *Clustering* con MongoDB 8](#_Toc38185076)

[2.3.1. Enunciado del ejercicio 8](#_Toc38185077)

[2.3.2. Realización del ejercicio 9](#_Toc38185078)

[2.3.3. Dificultades encontradas y opinión del ejercicio 25](#_Toc38185079)

Ilustraciones

[Ilustración 2‑1. Diagrama diseño Base de Datos. Blog de noticias. 5](#_Toc38185080)

[Ilustración 2‑2. Arquitectura de clustering 9](#_Toc38185081)

[Ilustración 2‑3. Contenedores de Docker iniciados 10](#_Toc38185082)

# Descripción del documento

Este documento consiste en la memoria del Trabajo Práctico 2, de la asignatura de “Gestión y Almacenamiento de Información no Estructurada” del “Máster en Ingeniería y Ciencia de Datos” de la UNED.

El trabajo consiste en la realización de tres ejercicios con el objetivo de familiarizarse con todos los aspectos de la utilización de bases de datos documentales, y en concreto de la tecnología MongoDB.

En el primer ejercicio, se explorará el diseño de una base de datos utilizando dicha tecnología. En el segundo ejercicio se proporcionarán los datos necesarios para poblar una base de datos MongoDB ya definida y se solicitará al estudiante la implementación de diversas consultas que permitan explorar las diferentes funcionalidades que proporciona la tecnología. Finalmente, en el tercer ejercicio se explorarán las características de MongoDB en lo relativo al clustering y la distribución de los datos, para lo cual el estudiante deberá́ seguir un tutorial propuesto para la creación de una estructura de clúster para la base de datos y la exploración de sus características.

# Ejercicios

## Ejercicio 1: Caso práctico Blog de Noticias.

En el primer ejercicio se propone un caso práctico concreto para el cual se solicita el diseño más adecuado para una base de datos documental:

* Se precisa diseñar un blog de noticias donde los usuarios registrados puedan publicar sus comentarios.
  + Cada autor tiene un nombre, un nombre de usuario, una cuenta de Twitter y una descripción. Además, de forma opcional, los usuarios pueden proporcionar como datos su dirección postal (pueden tener una como máximo, con calle, número, puerta, C.P y ciudad) o sus teléfonos de contacto (pueden tener varios).
  + Las noticias tienen un título, un cuerpo y una fecha de publicación. Son publicadas por un solo autor y pueden contener o no una lista de tags.
  + Las noticias reciben comentarios (de 0 a n), quedando registrado la persona que lo escribió, el comentario escrito y el momento en el que lo hizo.
* Las consultas más frecuentes en la base de datos son las siguientes:
  + Consultas por nombre de usuario.
  + Consultas por cuenta de Twitter.
  + Agregaciones por código postal (número de usuarios que tienen el mismo C.P.)
  + Consultas de noticias de un usuario, ordenadas por fecha (las "n" últimas noticias publicadas, de la más reciente a la más antigua).
  + Número de comentarios por noticia, por día o por usuario.

### Diseño de la base de datos documental.

Para explicar el diseño de la base de datos, se ha implementado el siguiente diagrama:

Captura de pantalla de un celular

Descripción generada automáticamente

Ilustración 2‑1. Diagrama diseño Base de Datos. Blog de noticias.

En el que se pueden observar las siguientes colecciones: Contact (que puede ser tanto Address como Phone), User, Comment y New.

* **User**: Cada usuario registrado en el blog. Ejemplo de la colección:

|  |
| --- |
| {  "name": "Javier Piqueras",  "username": "javipiquerasm",  "twitter\_username": "javipiquerasm",  "description": "Hi, my name is Javi, and I'm here to read all the interesting news related Covid-19 topic",  "contact": {  "phone\_numbers": ["655601565", "65894523", "914568759"]  }  } |

* **Contact**: Cada contacto asociado a un usuario (puede ser de tipo **Address** o **Phone**), dependiendo el tipo de contacto que el usuario haya preferido. Ejemplos:

|  |
| --- |
| {  "street": "C/ Donoso Cortés",  "number": "9",  "door": "B",  "postal\_code": "28015",  "city": "Madrid"  }  {  "phone\_numbers": ["655601565", "65894523", "914568759"]  } |

* **New**: Cada noticia publicada en el blog. Ejemplo:

|  |
| --- |
| {  "\_id": "46258",  "title": "Bajan el número de casos nuevos del coronavirus",  "body": "En las últimas 24 horas, el número de nuevos confirmados por coronavirus en España ha descendido un 4.4%",  "publication\_date": 2020-04-15T12:01:000,  "tags": ["virus", "coronavirus", "covid19"],  "author\_username": "marcorecuenco\_",  "comments": [  {  "content": "Buenísimo artículo, ¡muchas gracias pro compartirlo!",  "publication\_date": 2020-04-15T12:15:000,  "author\_username": "javipiqueras"  }  ]  } |

* **Comment**: Cada comentario publicado por un usuario registrado en una noticia. Ejemplo:

|  |
| --- |
| {  "content": "Buenísimo artículo, ¡muchas gracias pro compartirlo!",  "publication\_date": 2020-04-15T12:15:000,  "author\_username": "javipiqueras"  } |

Nótese que las relaciones son tanto de tipo “*One-to-one*”, “*One-to-many*” embebido, como “*One-to-many*” con referencias.

### Definición de índices

Para optimizar las operaciones que el usuario debe ser capaz de hacer, resultaría interesante añadir los siguientes índices:

* Sobre la colección Users, un índice compuesto sobre los siguientes campos:
  + username
  + twitter\_username
  + contact.postal\_code
* Sobre la colección News, un índice compuesto sobre los siguientes campos:
  + autor\_username
  + publication\_date

Esto optimizará la búsqueda del usuario por cualquiera de ambos campos.

El código que inserta los índices se encuentra en el notebook correspondiente al ejercicio\_1 (ejercicio\_1.ipynb), en el apartado “Creación de Índices”.

### Inserciones

El código necesario para las inserciones se encuentra en el notebook correspondiente al ejercicio 1 (ejercicio\_1.ipynb), en el apartado “Inserciones”.

### Dificultades encontradas y opinión del ejercicio

Personalmente, este ejercicio me ha gustado para ponernos en la piel de quien debe diseñar una base de datos documental, que es el punto de partida. Sin embargo, he echado un poco en falta algún ejemplo sobre como poder “diagramar” una base de datos documental.

## Ejercicio 2: Consultas sobre la colección de visitas a webs.

### Enunciado del ejercicio

Para la realización del ejercicio 2 se proporciona el fichero “webs.json”, una colección que contiene información sobre las paginas web que han visitado distintos usuarios, y sobre la que vamos a realizar consultas y agregaciones. Cada documento se puede entender como una “sesión” realizada por el usuario y contiene el nombre de usuario, la fecha de inicio de la sesión, el navegador utilizado y un array que contiene las páginas visitadas. Para cada página visitada se almacena la URL y el número de segundos en los que la página ha estado activa.

El primer paso para la realización de este ejercicio será importar la colección almacenada en “webs.json” a nuestra base de datos MongoDB.

Una vez que se ha importado la colección, nos conectaremos a la base de datos tal y como se ha indicado en el apartado 2 a través del notebook Jupyter y pymongo.

Se pide ejecutar las siguientes consultas y agregaciones:

* **Consulta 1**: Número de sesiones (documentos) en los que se ha visitado la web “https://www.mikokoloko.com”. Nota: si una sesión incluye dos visitas solo se contará una vez.
* **Consulta 2**: Mostrar los distintos navegadores que se han utilizado en toda la colección.
* **Consulta 3**: Mostrar únicamente el campo “navegador” de las sesiones realizadas por el usuario “Bertoldo” entre las fechas “20/02/2016” y “25/02/2016” (a las 00:00:00.000 horas en ambos casos).
* **Agregación 1**: Obtener el número de sesiones por cada navegador, ordenadas en orden ascendente. La salida mostrará, para cada navegador, un “\_id” que contiene el nombre del navegador, y un campo "total" con el número de sesiones de dicho navegador.
* **Agregación 2**: Obtener la persona con mayor número de sesiones totales en la colección. La salida mostrará para cada persona (sólo 1) un “\_id” conteniendo el nombre de la persona y un campo “total” con el número de sesiones totales.
* **Agregación 3**: Para cada persona indicar el numero de URLs visitadas durante más de 100 segundos (se cuentan las repeticiones). La salida mostrará para cada persona un “\_id” conteniendo el nombre de la persona y un campo “num” con el número de URLs. Pista: utilizar la opción $unwind de aggregate.
* **Agregación 4**: Nombre de la persona que visita más páginas de media por sesión. La salida mostrará un “\_id” conteniendo el nombre de la persona y un campo “media” con el valor de la media de páginas visitadas. Pistas: calcular el número de URLs para cada sesión y luego calcular la media de ese dato para cada persona / No hace falta utilizar $unwind.
* **Agregación 5**: Calcular el total de sesiones con el navegador “Explorer” en 2016. La salida mostrará el “\_id” conteniendo el año (2016) y un campo “total” con el número total de sesiones. Pista: para obtener el año de una fecha se puede utilizar {$year:”$fecha”}.
* **Agregación 6**: Para cada persona calcular el tiempo acumulado de todas las URLs visitadas, en minutos. La salida mostrará un campo “nombre” con el nombre de la persona, y un campo “tiempomin” con el tiempo acumulado. La salida estará́ ordenada alfabéticamente por los nombres de los usuarios. Pista: utilizar la opción $unwind de aggregate.
* **Agregación 7**: Obtener el nombre de las personas que han visitado alguna página que comience por “https”. La salida mostrará para cada persona únicamente el campo “nombre” que contendrá su nombre. Pista: utilizar el operador de proyección $substr.

### Realización del ejercicio

El código necesario para las consultas y agregaciones se encuentra en el notebook correspondiente al ejercicio 2 (ejercicio\_2.ipynb).

### Dificultades encontradas y opinión del ejercicio

Para la realización de este ejercicio, no se han encontrado dificultades. Además, como opinión, no daría tantas pistas en las últimas agregaciones. Con ellas se encontraba la solución directamente.

## Ejercicio 3: *Clustering* con MongoDB

### Enunciado del ejercicio

El último ejercicio de la práctica pretende que el estudiante se familiarice con las posibilidades que ofrece MongoDB para realizar clustering para la distribución de datos.

El primer paso consistirá en clonar el repositorio de github indicado en el siguiente enlace: <https://github.com/minhhungit/mongodb-cluster-docker-compose.git>.

Dentro de esta carpeta podremos encontrar todos los archivos necesarios para seguir el tutorial que se indica en el apartado “Setup” de la página de github indicada anteriormente.

En el primer paso (“Step 1”) se construyen todos los contenedores mediante “docker-compose”, para a continuación (“Step 2”) inicializar cada replica set asignado a cada uno de los shards previamente creados. En el tercer paso (“Step 3”) se crea el router que nos permitirá conectarnos a la base de datos. Finalmente, en el cuarto paso (“Step 4”) se habilita el sharding y se crea una colección que contiene una shard key. Los últimos pasos que se indican en el tutorial pertenecen a operaciones de verificación y comprobación de la base de datos.

Se pide:

* Completar todos los pasos para levantar la base de datos y realizar las comprobaciones para verificar que se han creado correctamente tres shards, cada uno con un replica set de un nodo primario y dos secundarios. Documentar todo el proceso seguido con los comandos utilizados, las salidas obtenidas y las observaciones que el estudiante considere oportunas.

### Realización del ejercicio

Los componentes que vamos a construir son los siguientes:

* Config Server (3-member replica set): configsvr01, configsvr02, configsvr03
* 3 Shards (each a 3 member PSS replica set):
  + shard01-a, shard01-b, shard01-c
  + shard02-a, shard02-b, shard02-c
  + shard03-a, shard03-b, shard03-c
* 2 Routers (mongos): router01, router02

Imagen que contiene texto

Descripción generada automáticamente

Ilustración 2‑2. Arquitectura de clustering

Para inicializar todos los *containers*, se ejecuta el siguiente comando:

$ docker-compose up -d

Captura de pantalla de un celular con letras

Descripción generada automáticamente

Ilustración 2‑3. Contenedores de Docker iniciados

Seguidamente, se van a inicializar todos los conjuntos replicados y routers con las siguientes instrucciones ejecutadas en la línea de comandos:

$ docker-compose exec configsvr01 sh -c "mongo < /scripts/init-configserver.js"

|  |
| --- |
| (/Users/javierpiquerasmartinez/UNED/MASTER-INGENIERIA-CIENCIA-DATOS/GAE/environment) MacBook-Pro-de-Javier:mongodb-cluster-docker-compose javierpiquerasmartinez$ docker-compose exec configsvr01 sh -c "mongo < /scripts/init-configserver.js"  MongoDB shell version v4.0.17  connecting to: mongodb://127.0.0.1:27017/?gssapiServiceName=mongodb  Implicit session: session { "id" : UUID("a4c1ced9-c5bb-47ea-bbb5-436213222cdc") }  MongoDB server version: 4.0.17  {  "ok" : 1,  "operationTime" : Timestamp(1587230885, 1),  "$gleStats" : {  "lastOpTime" : Timestamp(1587230885, 1),  "electionId" : ObjectId("000000000000000000000000")  },  "lastCommittedOpTime" : Timestamp(0, 0),  "$clusterTime" : {  "clusterTime" : Timestamp(1587230885, 1),  "signature" : {  "hash" : BinData(0,"AAAAAAAAAAAAAAAAAAAAAAAAAAA="),  "keyId" : NumberLong(0)  }  }  }  bye |

El servidor ya está levantado y corriendo. A continuación, se inicializan cada uno de los shards:

$ docker-compose exec shard01-a sh -c "mongo < /scripts/init-shard01.js"

|  |
| --- |
| (/Users/javierpiquerasmartinez/UNED/MASTER-INGENIERIA-CIENCIA-DATOS/GAE/environment) MacBook-Pro-de-Javier:mongodb-cluster-docker-compose javierpiquerasmartinez$ docker-compose exec shard01-a sh -c "mongo < /scripts/init-shard01.js"  MongoDB shell version v4.0.17  connecting to: mongodb://127.0.0.1:27017/?gssapiServiceName=mongodb  Implicit session: session { "id" : UUID("223c61e9-4c4e-4e47-9fd5-2bdc496cb2b8") }  MongoDB server version: 4.0.17  {  "ok" : 1,  "operationTime" : Timestamp(1587230953, 1),  "$clusterTime" : {  "clusterTime" : Timestamp(1587230953, 1),  "signature" : {  "hash" : BinData(0,"AAAAAAAAAAAAAAAAAAAAAAAAAAA="),  "keyId" : NumberLong(0)  }  }  }  bye |

$ docker-compose exec shard02-a sh -c "mongo < /scripts/init-shard02.js"

|  |
| --- |
| (/Users/javierpiquerasmartinez/UNED/MASTER-INGENIERIA-CIENCIA-DATOS/GAE/environment) MacBook-Pro-de-Javier:mongodb-cluster-docker-compose javierpiquerasmartinez$ docker-compose exec shard02-a sh -c "mongo < /scripts/init-shard02.js"  MongoDB shell version v4.0.17  connecting to: mongodb://127.0.0.1:27017/?gssapiServiceName=mongodb  Implicit session: session { "id" : UUID("ebbcb1bf-0c93-49f5-926e-34c999fc5296") }  MongoDB server version: 4.0.17  {  "ok" : 1,  "operationTime" : Timestamp(1587230971, 1),  "$clusterTime" : {  "clusterTime" : Timestamp(1587230971, 1),  "signature" : {  "hash" : BinData(0,"AAAAAAAAAAAAAAAAAAAAAAAAAAA="),  "keyId" : NumberLong(0)  }  }  }  bye |

$ docker-compose exec shard03-a sh -c "mongo < /scripts/init-shard03.js"

|  |
| --- |
| (/Users/javierpiquerasmartinez/UNED/MASTER-INGENIERIA-CIENCIA-DATOS/GAE/environment) MacBook-Pro-de-Javier:mongodb-cluster-docker-compose javierpiquerasmartinez$ docker-compose exec shard03-a sh -c "mongo < /scripts/init-shard03.js"  MongoDB shell version v4.0.17  connecting to: mongodb://127.0.0.1:27017/?gssapiServiceName=mongodb  Implicit session: session { "id" : UUID("88469ddf-e27d-473f-835f-610c3f2f542f") }  MongoDB server version: 4.0.17  {  "ok" : 1,  "operationTime" : Timestamp(1587231002, 1),  "$clusterTime" : {  "clusterTime" : Timestamp(1587231002, 1),  "signature" : {  "hash" : BinData(0,"AAAAAAAAAAAAAAAAAAAAAAAAAAA="),  "keyId" : NumberLong(0)  }  }  }  bye |

Y, finalmente, el router:

$ docker-compose exec router01 sh -c "mongo < /scripts/init-router.js"

|  |
| --- |
| (/Users/javierpiquerasmartinez/UNED/MASTER-INGENIERIA-CIENCIA-DATOS/GAE/environment) MacBook-Pro-de-Javier:mongodb-cluster-docker-compose javierpiquerasmartinez$ docker-compose exec router01 sh -c "mongo < /scripts/init-router.js"  MongoDB shell version v4.0.17  connecting to: mongodb://127.0.0.1:27017/?gssapiServiceName=mongodb  Implicit session: session { "id" : UUID("adae3b8d-56e1-4708-890f-0b5d639539fb") }  MongoDB server version: 4.0.17  {  "shardAdded" : "rs-shard-01",  "ok" : 1,  "operationTime" : Timestamp(1587231079, 4),  "$clusterTime" : {  "clusterTime" : Timestamp(1587231079, 4),  "signature" : {  "hash" : BinData(0,"AAAAAAAAAAAAAAAAAAAAAAAAAAA="),  "keyId" : NumberLong(0)  }  }  }  {  "shardAdded" : "rs-shard-01",  "ok" : 1,  "operationTime" : Timestamp(1587231079, 4),  "$clusterTime" : {  "clusterTime" : Timestamp(1587231079, 4),  "signature" : {  "hash" : BinData(0,"AAAAAAAAAAAAAAAAAAAAAAAAAAA="),  "keyId" : NumberLong(0)  }  }  }  {  "shardAdded" : "rs-shard-01",  "ok" : 1,  "operationTime" : Timestamp(1587231079, 4),  "$clusterTime" : {  "clusterTime" : Timestamp(1587231079, 4),  "signature" : {  "hash" : BinData(0,"AAAAAAAAAAAAAAAAAAAAAAAAAAA="),  "keyId" : NumberLong(0)  }  }  }  {  "shardAdded" : "rs-shard-02",  "ok" : 1,  "operationTime" : Timestamp(1587231079, 10),  "$clusterTime" : {  "clusterTime" : Timestamp(1587231079, 10),  "signature" : {  "hash" : BinData(0,"AAAAAAAAAAAAAAAAAAAAAAAAAAA="),  "keyId" : NumberLong(0)  }  }  }  {  "shardAdded" : "rs-shard-02",  "ok" : 1,  "operationTime" : Timestamp(1587231079, 10),  "$clusterTime" : {  "clusterTime" : Timestamp(1587231079, 10),  "signature" : {  "hash" : BinData(0,"AAAAAAAAAAAAAAAAAAAAAAAAAAA="),  "keyId" : NumberLong(0)  }  }  }  {  "shardAdded" : "rs-shard-02",  "ok" : 1,  "operationTime" : Timestamp(1587231079, 10),  "$clusterTime" : {  "clusterTime" : Timestamp(1587231079, 10),  "signature" : {  "hash" : BinData(0,"AAAAAAAAAAAAAAAAAAAAAAAAAAA="),  "keyId" : NumberLong(0)  }  }  }  {  "shardAdded" : "rs-shard-03",  "ok" : 1,  "operationTime" : Timestamp(1587231079, 16),  "$clusterTime" : {  "clusterTime" : Timestamp(1587231079, 16),  "signature" : {  "hash" : BinData(0,"AAAAAAAAAAAAAAAAAAAAAAAAAAA="),  "keyId" : NumberLong(0)  }  }  }  {  "shardAdded" : "rs-shard-03",  "ok" : 1,  "operationTime" : Timestamp(1587231079, 16),  "$clusterTime" : {  "clusterTime" : Timestamp(1587231079, 16),  "signature" : {  "hash" : BinData(0,"AAAAAAAAAAAAAAAAAAAAAAAAAAA="),  "keyId" : NumberLong(0)  }  }  }  {  "shardAdded" : "rs-shard-03",  "ok" : 1,  "operationTime" : Timestamp(1587231079, 16),  "$clusterTime" : {  "clusterTime" : Timestamp(1587231079, 16),  "signature" : {  "hash" : BinData(0,"AAAAAAAAAAAAAAAAAAAAAAAAAAA="),  "keyId" : NumberLong(0)  }  }  }  bye |

En las trazas, se pueden visualizar que el router ha añadido a su configuración inicial los nueve shards (tres por cada server).

Seguidamente, se habilita el “sharding” y se configura la estrategia de sharding que en este caso va a ser mediante “Hash based sharding” cuya key es el campo “supplierId”. Esto va a repartir los datos de forma aleatoria entre todos los shards.

|  |
| --- |
| (/Users/javierpiquerasmartinez/UNED/MASTER-INGENIERIA-CIENCIA-DATOS/GAE/environment) MacBook-Pro-de-Javier:mongodb-cluster-docker-compose javierpiquerasmartinez$ **docker-compose exec router01 mongo --port 27017**  MongoDB shell version v4.0.17  connecting to: mongodb://127.0.0.1:27017/?gssapiServiceName=mongodb  Implicit session: session { "id" : UUID("724c2b6a-5536-4aa7-b3d1-8763940e9b28") }  MongoDB server version: 4.0.17  Welcome to the MongoDB shell.  For interactive help, type "help".  For more comprehensive documentation, see  http://docs.mongodb.org/  Questions? Try the support group  http://groups.google.com/group/mongodb-user  Server has startup warnings:  2020-04-18T17:22:53.910+0000 I CONTROL [main]  2020-04-18T17:22:53.910+0000 I CONTROL [main] \*\* WARNING: Access control is not enabled for the database.  2020-04-18T17:22:53.910+0000 I CONTROL [main] \*\* Read and write access to data and configuration is unrestricted.  2020-04-18T17:22:53.910+0000 I CONTROL [main]  mongos> **sh.enableSharding("MyDatabase")**  {  "ok" : 1,  "operationTime" : Timestamp(1587231444, 5),  "$clusterTime" : {  "clusterTime" : Timestamp(1587231444, 5),  "signature" : {  "hash" : BinData(0,"AAAAAAAAAAAAAAAAAAAAAAAAAAA="),  "keyId" : NumberLong(0)  }  }  }  mongos> **db.adminCommand( { shardCollection: "MyDatabase.MyCollection", key: { supplierId: "hashed" } } )**  {  "collectionsharded" : "MyDatabase.MyCollection",  "collectionUUID" : UUID("cbe19c6a-c2f0-4d95-8d08-44cc65daf8e3"),  "ok" : 1,  "operationTime" : Timestamp(1587231456, 44),  "$clusterTime" : {  "clusterTime" : Timestamp(1587231456, 44),  "signature" : {  "hash" : BinData(0,"AAAAAAAAAAAAAAAAAAAAAAAAAAA="),  "keyId" : NumberLong(0)  }  }  }  mongos> **sh.status()**  --- Sharding Status ---  sharding version: {  "\_id" : 1,  "minCompatibleVersion" : 5,  "currentVersion" : 6,  "clusterId" : ObjectId("5e9b38b0856eda63e5559506")  }  shards:  { "\_id" : "rs-shard-01", "host" : "rs-shard-01/shard01-a:27017,shard01-b:27017,shard01-c:27017", "state" : 1 }  { "\_id" : "rs-shard-02", "host" : "rs-shard-02/shard02-a:27017,shard02-b:27017,shard02-c:27017", "state" : 1 }  { "\_id" : "rs-shard-03", "host" : "rs-shard-03/shard03-a:27017,shard03-b:27017,shard03-c:27017", "state" : 1 }  active mongoses:  "4.0.17" : 2  autosplit:  Currently enabled: yes  balancer:  Currently enabled: yes  Currently running: no  Failed balancer rounds in last 5 attempts: 0  Migration Results for the last 24 hours:  No recent migrations  databases:  { "\_id" : "MyDatabase", "primary" : "rs-shard-02", "partitioned" : true, "version" : { "uuid" : UUID("31e8d049-beec-4e94-b486-196f53b5809e"), "lastMod" : 1 } }  MyDatabase.MyCollection  shard key: { "supplierId" : "hashed" }  unique: false  balancing: true  chunks:  rs-shard-01 2  rs-shard-02 2  rs-shard-03 2  { "supplierId" : { "$minKey" : 1 } } -->> { "supplierId" : NumberLong("-6148914691236517204") } on : rs-shard-01 Timestamp(1, 0)  { "supplierId" : NumberLong("-6148914691236517204") } -->> { "supplierId" : NumberLong("-3074457345618258602") } on : rs-shard-01 Timestamp(1, 1)  { "supplierId" : NumberLong("-3074457345618258602") } -->> { "supplierId" : NumberLong(0) } on : rs-shard-02 Timestamp(1, 2)  { "supplierId" : NumberLong(0) } -->> { "supplierId" : NumberLong("3074457345618258602") } on : rs-shard-02 Timestamp(1, 3)  { "supplierId" : NumberLong("3074457345618258602") } -->> { "supplierId" : NumberLong("6148914691236517204") } on : rs-shard-03 Timestamp(1, 4)  { "supplierId" : NumberLong("6148914691236517204") } -->> { "supplierId" : { "$maxKey" : 1 } } on : rs-shard-03 Timestamp(1, 5)  { "\_id" : "config", "primary" : "config", "partitioned" : true }  config.system.sessions  shard key: { "\_id" : 1 }  unique: false  balancing: true  chunks:  rs-shard-01 1  { "\_id" : { "$minKey" : 1 } } -->> { "\_id" : { "$maxKey" : 1 } } on : rs-shard-01 Timestamp(1, 0) |

Tras la ejecución de sh.status() se puede observar toda la información que se ha comentado anteriormente relativa a la arquitectura de clustering.

Ahora, se comprueba el estado de las réplicas para cada “shard”:

|  |
| --- |
| (/Users/javierpiquerasmartinez/UNED/MASTER-INGENIERIA-CIENCIA-DATOS/GAE/environment) MacBook-Pro-de-Javier:mongodb-cluster-docker-compose javierpiquerasmartinez$ **docker exec -it rydell-shard-01-node-a bash -c "echo 'rs.status()' | mongo --port 27017"**  MongoDB shell version v4.0.17  connecting to: mongodb://127.0.0.1:27017/?gssapiServiceName=mongodb  Implicit session: session { "id" : UUID("9e0831ac-f75c-4399-bf74-7f512df540bb") }  MongoDB server version: 4.0.17  {  "set" : "rs-shard-01",  "date" : ISODate("2020-04-18T17:43:22.317Z"),  "myState" : 1,  "term" : NumberLong(1),  "syncingTo" : "",  "syncSourceHost" : "",  "syncSourceId" : -1,  "heartbeatIntervalMillis" : NumberLong(2000),  "optimes" : {  "lastCommittedOpTime" : {  "ts" : Timestamp(1587231798, 1),  "t" : NumberLong(1)  },  "readConcernMajorityOpTime" : {  "ts" : Timestamp(1587231798, 1),  "t" : NumberLong(1)  },  "appliedOpTime" : {  "ts" : Timestamp(1587231798, 1),  "t" : NumberLong(1)  },  "durableOpTime" : {  "ts" : Timestamp(1587231798, 1),  "t" : NumberLong(1)  }  },  "lastStableCheckpointTimestamp" : Timestamp(1587231745, 1),  "electionCandidateMetrics" : {  "lastElectionReason" : "electionTimeout",  "lastElectionDate" : ISODate("2020-04-18T17:29:24.933Z"),  "electionTerm" : NumberLong(1),  "lastCommittedOpTimeAtElection" : {  "ts" : Timestamp(0, 0),  "t" : NumberLong(-1)  },  "lastSeenOpTimeAtElection" : {  "ts" : Timestamp(1587230953, 1),  "t" : NumberLong(-1)  },  "numVotesNeeded" : 2,  "priorityAtElection" : 1,  "electionTimeoutMillis" : NumberLong(10000),  "numCatchUpOps" : NumberLong(0),  "newTermStartDate" : ISODate("2020-04-18T17:29:24.938Z"),  "wMajorityWriteAvailabilityDate" : ISODate("2020-04-18T17:29:25.710Z")  },  "members" : [  {  "\_id" : 0,  "name" : "shard01-a:27017",  "health" : 1,  "state" : 1,  "stateStr" : "PRIMARY",  "uptime" : 1227,  "optime" : {  "ts" : Timestamp(1587231798, 1),  "t" : NumberLong(1)  },  "optimeDate" : ISODate("2020-04-18T17:43:18Z"),  "syncingTo" : "",  "syncSourceHost" : "",  "syncSourceId" : -1,  "infoMessage" : "",  "electionTime" : Timestamp(1587230964, 1),  "electionDate" : ISODate("2020-04-18T17:29:24Z"),  "configVersion" : 1,  "self" : true,  "lastHeartbeatMessage" : ""  },  {  "\_id" : 1,  "name" : "shard01-b:27017",  "health" : 1,  "state" : 2,  "stateStr" : "SECONDARY",  "uptime" : 848,  "optime" : {  "ts" : Timestamp(1587231798, 1),  "t" : NumberLong(1)  },  "optimeDurable" : {  "ts" : Timestamp(1587231798, 1),  "t" : NumberLong(1)  },  "optimeDate" : ISODate("2020-04-18T17:43:18Z"),  "optimeDurableDate" : ISODate("2020-04-18T17:43:18Z"),  "lastHeartbeat" : ISODate("2020-04-18T17:43:22.013Z"),  "lastHeartbeatRecv" : ISODate("2020-04-18T17:43:20.773Z"),  "pingMs" : NumberLong(1),  "lastHeartbeatMessage" : "",  "syncingTo" : "shard01-a:27017",  "syncSourceHost" : "shard01-a:27017",  "syncSourceId" : 0,  "infoMessage" : "",  "configVersion" : 1  },  {  "\_id" : 2,  "name" : "shard01-c:27017",  "health" : 1,  "state" : 2,  "stateStr" : "SECONDARY",  "uptime" : 848,  "optime" : {  "ts" : Timestamp(1587231798, 1),  "t" : NumberLong(1)  },  "optimeDurable" : {  "ts" : Timestamp(1587231798, 1),  "t" : NumberLong(1)  },  "optimeDate" : ISODate("2020-04-18T17:43:18Z"),  "optimeDurableDate" : ISODate("2020-04-18T17:43:18Z"),  "lastHeartbeat" : ISODate("2020-04-18T17:43:22.011Z"),  "lastHeartbeatRecv" : ISODate("2020-04-18T17:43:20.773Z"),  "pingMs" : NumberLong(0),  "lastHeartbeatMessage" : "",  "syncingTo" : "shard01-a:27017",  "syncSourceHost" : "shard01-a:27017",  "syncSourceId" : 0,  "infoMessage" : "",  "configVersion" : 1  }  ],  "ok" : 1,  "operationTime" : Timestamp(1587231798, 1),  "$gleStats" : {  "lastOpTime" : Timestamp(0, 0),  "electionId" : ObjectId("7fffffff0000000000000001")  },  "lastCommittedOpTime" : Timestamp(1587231798, 1),  "$configServerState" : {  "opTime" : {  "ts" : Timestamp(1587231799, 3),  "t" : NumberLong(1)  }  },  "$clusterTime" : {  "clusterTime" : Timestamp(1587231799, 3),  "signature" : {  "hash" : BinData(0,"AAAAAAAAAAAAAAAAAAAAAAAAAAA="),  "keyId" : NumberLong(0)  }  }  }  Bye  (/Users/javierpiquerasmartinez/UNED/MASTER-INGENIERIA-CIENCIA-DATOS/GAE/environment) MacBook-Pro-de-Javier:mongodb-cluster-docker-compose javierpiquerasmartinez$ **docker exec -it rydell-shard-02-node-a bash -c "echo 'rs.status()' | mongo --port 27017"**  MongoDB shell version v4.0.17  connecting to: mongodb://127.0.0.1:27017/?gssapiServiceName=mongodb  Implicit session: session { "id" : UUID("5a91adad-d0ff-4af4-a548-f8aeb90617fd") }  MongoDB server version: 4.0.17  {  "set" : "rs-shard-02",  "date" : ISODate("2020-04-18T17:45:30.769Z"),  "myState" : 1,  "term" : NumberLong(1),  "syncingTo" : "",  "syncSourceHost" : "",  "syncSourceId" : -1,  "heartbeatIntervalMillis" : NumberLong(2000),  "optimes" : {  "lastCommittedOpTime" : {  "ts" : Timestamp(1587231921, 1),  "t" : NumberLong(1)  },  "readConcernMajorityOpTime" : {  "ts" : Timestamp(1587231921, 1),  "t" : NumberLong(1)  },  "appliedOpTime" : {  "ts" : Timestamp(1587231921, 1),  "t" : NumberLong(1)  },  "durableOpTime" : {  "ts" : Timestamp(1587231921, 1),  "t" : NumberLong(1)  }  },  "lastStableCheckpointTimestamp" : Timestamp(1587231881, 1),  "electionCandidateMetrics" : {  "lastElectionReason" : "electionTimeout",  "lastElectionDate" : ISODate("2020-04-18T17:29:41.352Z"),  "electionTerm" : NumberLong(1),  "lastCommittedOpTimeAtElection" : {  "ts" : Timestamp(0, 0),  "t" : NumberLong(-1)  },  "lastSeenOpTimeAtElection" : {  "ts" : Timestamp(1587230971, 1),  "t" : NumberLong(-1)  },  "numVotesNeeded" : 2,  "priorityAtElection" : 1,  "electionTimeoutMillis" : NumberLong(10000),  "numCatchUpOps" : NumberLong(0),  "newTermStartDate" : ISODate("2020-04-18T17:29:41.362Z"),  "wMajorityWriteAvailabilityDate" : ISODate("2020-04-18T17:29:42.320Z")  },  "members" : [  {  "\_id" : 0,  "name" : "shard02-a:27017",  "health" : 1,  "state" : 1,  "stateStr" : "PRIMARY",  "uptime" : 1355,  "optime" : {  "ts" : Timestamp(1587231921, 1),  "t" : NumberLong(1)  },  "optimeDate" : ISODate("2020-04-18T17:45:21Z"),  "syncingTo" : "",  "syncSourceHost" : "",  "syncSourceId" : -1,  "infoMessage" : "",  "electionTime" : Timestamp(1587230981, 1),  "electionDate" : ISODate("2020-04-18T17:29:41Z"),  "configVersion" : 1,  "self" : true,  "lastHeartbeatMessage" : ""  },  {  "\_id" : 1,  "name" : "shard02-b:27017",  "health" : 1,  "state" : 2,  "stateStr" : "SECONDARY",  "uptime" : 959,  "optime" : {  "ts" : Timestamp(1587231921, 1),  "t" : NumberLong(1)  },  "optimeDurable" : {  "ts" : Timestamp(1587231921, 1),  "t" : NumberLong(1)  },  "optimeDate" : ISODate("2020-04-18T17:45:21Z"),  "optimeDurableDate" : ISODate("2020-04-18T17:45:21Z"),  "lastHeartbeat" : ISODate("2020-04-18T17:45:30.595Z"),  "lastHeartbeatRecv" : ISODate("2020-04-18T17:45:29.396Z"),  "pingMs" : NumberLong(0),  "lastHeartbeatMessage" : "",  "syncingTo" : "shard02-a:27017",  "syncSourceHost" : "shard02-a:27017",  "syncSourceId" : 0,  "infoMessage" : "",  "configVersion" : 1  },  {  "\_id" : 2,  "name" : "shard02-c:27017",  "health" : 1,  "state" : 2,  "stateStr" : "SECONDARY",  "uptime" : 959,  "optime" : {  "ts" : Timestamp(1587231921, 1),  "t" : NumberLong(1)  },  "optimeDurable" : {  "ts" : Timestamp(1587231921, 1),  "t" : NumberLong(1)  },  "optimeDate" : ISODate("2020-04-18T17:45:21Z"),  "optimeDurableDate" : ISODate("2020-04-18T17:45:21Z"),  "lastHeartbeat" : ISODate("2020-04-18T17:45:30.594Z"),  "lastHeartbeatRecv" : ISODate("2020-04-18T17:45:29.396Z"),  "pingMs" : NumberLong(0),  "lastHeartbeatMessage" : "",  "syncingTo" : "shard02-a:27017",  "syncSourceHost" : "shard02-a:27017",  "syncSourceId" : 0,  "infoMessage" : "",  "configVersion" : 1  }  ],  "ok" : 1,  "operationTime" : Timestamp(1587231921, 1),  "$gleStats" : {  "lastOpTime" : Timestamp(0, 0),  "electionId" : ObjectId("7fffffff0000000000000001")  },  "lastCommittedOpTime" : Timestamp(1587231921, 1),  "$configServerState" : {  "opTime" : {  "ts" : Timestamp(1587231928, 2),  "t" : NumberLong(1)  }  },  "$clusterTime" : {  "clusterTime" : Timestamp(1587231928, 2),  "signature" : {  "hash" : BinData(0,"AAAAAAAAAAAAAAAAAAAAAAAAAAA="),  "keyId" : NumberLong(0)  }  }  }  bye  (/Users/javierpiquerasmartinez/UNED/MASTER-INGENIERIA-CIENCIA-DATOS/GAE/environment) MacBook-Pro-de-Javier:mongodb-cluster-docker-compose javierpiquerasmartinez$ **docker exec -it rydell-shard-03-node-a bash -c "echo 'rs.status()' | mongo --port 27017"**  MongoDB shell version v4.0.17  connecting to: mongodb://127.0.0.1:27017/?gssapiServiceName=mongodb  Implicit session: session { "id" : UUID("a947944f-3057-474d-84d9-4560599564a1") }  MongoDB server version: 4.0.17  {  "set" : "rs-shard-03",  "date" : ISODate("2020-04-18T17:45:51.895Z"),  "myState" : 1,  "term" : NumberLong(1),  "syncingTo" : "",  "syncSourceHost" : "",  "syncSourceId" : -1,  "heartbeatIntervalMillis" : NumberLong(2000),  "optimes" : {  "lastCommittedOpTime" : {  "ts" : Timestamp(1587231943, 1),  "t" : NumberLong(1)  },  "readConcernMajorityOpTime" : {  "ts" : Timestamp(1587231943, 1),  "t" : NumberLong(1)  },  "appliedOpTime" : {  "ts" : Timestamp(1587231943, 1),  "t" : NumberLong(1)  },  "durableOpTime" : {  "ts" : Timestamp(1587231943, 1),  "t" : NumberLong(1)  }  },  "lastStableCheckpointTimestamp" : Timestamp(1587231913, 1),  "electionCandidateMetrics" : {  "lastElectionReason" : "electionTimeout",  "lastElectionDate" : ISODate("2020-04-18T17:30:13.268Z"),  "electionTerm" : NumberLong(1),  "lastCommittedOpTimeAtElection" : {  "ts" : Timestamp(0, 0),  "t" : NumberLong(-1)  },  "lastSeenOpTimeAtElection" : {  "ts" : Timestamp(1587231002, 1),  "t" : NumberLong(-1)  },  "numVotesNeeded" : 2,  "priorityAtElection" : 1,  "electionTimeoutMillis" : NumberLong(10000),  "numCatchUpOps" : NumberLong(0),  "newTermStartDate" : ISODate("2020-04-18T17:30:13.274Z"),  "wMajorityWriteAvailabilityDate" : ISODate("2020-04-18T17:30:14.190Z")  },  "members" : [  {  "\_id" : 0,  "name" : "shard03-a:27017",  "health" : 1,  "state" : 1,  "stateStr" : "PRIMARY",  "uptime" : 1376,  "optime" : {  "ts" : Timestamp(1587231943, 1),  "t" : NumberLong(1)  },  "optimeDate" : ISODate("2020-04-18T17:45:43Z"),  "syncingTo" : "",  "syncSourceHost" : "",  "syncSourceId" : -1,  "infoMessage" : "",  "electionTime" : Timestamp(1587231013, 1),  "electionDate" : ISODate("2020-04-18T17:30:13Z"),  "configVersion" : 1,  "self" : true,  "lastHeartbeatMessage" : ""  },  {  "\_id" : 1,  "name" : "shard03-b:27017",  "health" : 1,  "state" : 2,  "stateStr" : "SECONDARY",  "uptime" : 949,  "optime" : {  "ts" : Timestamp(1587231943, 1),  "t" : NumberLong(1)  },  "optimeDurable" : {  "ts" : Timestamp(1587231943, 1),  "t" : NumberLong(1)  },  "optimeDate" : ISODate("2020-04-18T17:45:43Z"),  "optimeDurableDate" : ISODate("2020-04-18T17:45:43Z"),  "lastHeartbeat" : ISODate("2020-04-18T17:45:50.475Z"),  "lastHeartbeatRecv" : ISODate("2020-04-18T17:45:51.365Z"),  "pingMs" : NumberLong(0),  "lastHeartbeatMessage" : "",  "syncingTo" : "shard03-a:27017",  "syncSourceHost" : "shard03-a:27017",  "syncSourceId" : 0,  "infoMessage" : "",  "configVersion" : 1  },  {  "\_id" : 2,  "name" : "shard03-c:27017",  "health" : 1,  "state" : 2,  "stateStr" : "SECONDARY",  "uptime" : 949,  "optime" : {  "ts" : Timestamp(1587231943, 1),  "t" : NumberLong(1)  },  "optimeDurable" : {  "ts" : Timestamp(1587231943, 1),  "t" : NumberLong(1)  },  "optimeDate" : ISODate("2020-04-18T17:45:43Z"),  "optimeDurableDate" : ISODate("2020-04-18T17:45:43Z"),  "lastHeartbeat" : ISODate("2020-04-18T17:45:50.475Z"),  "lastHeartbeatRecv" : ISODate("2020-04-18T17:45:51.365Z"),  "pingMs" : NumberLong(0),  "lastHeartbeatMessage" : "",  "syncingTo" : "shard03-a:27017",  "syncSourceHost" : "shard03-a:27017",  "syncSourceId" : 0,  "infoMessage" : "",  "configVersion" : 1  }  ],  "ok" : 1,  "operationTime" : Timestamp(1587231943, 1),  "$gleStats" : {  "lastOpTime" : Timestamp(0, 0),  "electionId" : ObjectId("7fffffff0000000000000001")  },  "lastCommittedOpTime" : Timestamp(1587231943, 1),  "$configServerState" : {  "opTime" : {  "ts" : Timestamp(1587231949, 9),  "t" : NumberLong(1)  }  },  "$clusterTime" : {  "clusterTime" : Timestamp(1587231949, 9),  "signature" : {  "hash" : BinData(0,"AAAAAAAAAAAAAAAAAAAAAAAAAAA="),  "keyId" : NumberLong(0)  }  }  }  bye |

* Conectarse a la base de datos creada utilizando el siguiente comando:

$ docker-compose exec router01 mongo --port 27017

|  |
| --- |
| (/Users/javierpiquerasmartinez/UNED/MASTER-INGENIERIA-CIENCIA-DATOS/GAE/environment) MacBook-Pro-de-Javier:mongodb-cluster-docker-compose javierpiquerasmartinez$ **docker-compose exec router01 mongo --port 27017**  MongoDB shell version v4.0.17  connecting to: mongodb://127.0.0.1:27017/?gssapiServiceName=mongodb  Implicit session: session { "id" : UUID("980f3c44-36d5-4a12-8273-de7c21b3f209") }  MongoDB server version: 4.0.17  Server has startup warnings:  2020-04-18T17:22:53.910+0000 I CONTROL [main]  2020-04-18T17:22:53.910+0000 I CONTROL [main] \*\* WARNING: Access control is not enabled for the database.  2020-04-18T17:22:53.910+0000 I CONTROL [main] \*\* Read and write access to data and configuration is unrestricted.  2020-04-18T17:22:53.910+0000 I CONTROL [main]  mongos> |

Consola de mongo de nuestra base de datos.

* Acceder a la base de datos creada en el “Step 4” y comprobar que existe la colección “MyCollection”.

Para acceder a dicha base de datos, se hará uso del comando “ $ use MyDatabase”:

|  |
| --- |
| **mongos> use MyDatabase**  switched to db MyDatabase |

Para comprobar que existe la colección:

|  |
| --- |
| **mongos> db.runCommand( { listCollections: 1 } );**  {  "cursor" : {  "id" : NumberLong(0),  "ns" : "MyDatabase.$cmd.listCollections",  "firstBatch" : [  **{**  **"name" : "MyCollection",**  **"type" : "collection",**  **"options" : {**    **},**  **"info" : {**  **"readOnly" : false,**  **"uuid" : UUID("cbe19c6a-c2f0-4d95-8d08-44cc65daf8e3")**  **},**  **"idIndex" : {**  **"v" : 2,**  **"key" : {**  **"\_id" : 1**  **},**  **"name" : "\_id\_",**  **"ns" : "MyDatabase.MyCollection"**  **}**  **}**  ]  },  "ok" : 1,  "operationTime" : Timestamp(1587232441, 1),  "$clusterTime" : {  "clusterTime" : Timestamp(1587232441, 1),  "signature" : {  "hash" : BinData(0,"AAAAAAAAAAAAAAAAAAAAAAAAAAA="),  "keyId" : NumberLong(0)  }  }  } |

Y se verifica que efectivamente existe.

* Poblar la colección MyCollection utilizando el siguiente código:

> for (var i=0;i<2000;i++){db.MyCollection.insert({‘supplierId’:i})}

|  |
| --- |
| **mongos> for (var i=0;i<2000;i++){db.MyCollection.insert({'supplierId':i})}**  WriteResult({ "nInserted" : 1 }) |

* ¿Qué hemos hecho en la operación anterior? ¿Por qué hemos utilizado el campo “supplierId” en concreto?

En la operación anterior se han realizado dos mil inserciones dándole al campo supplierId un valor correlativo del 0 al 1999.

Se ha utilizado ese campo porque en la definición de la colección “MyCollection” se ha definido dicho campo como clave hash para repartir de forma aleatoria entre los shards las diferentes inserciones y equilibrar así la carga.

* Realizar una consulta “find()” genérica sobre la colección y utilizar el método que se ha estudiado en la parte teórica para la monitorización de consultas, especificando la opción "executionStats".

|  |
| --- |
| **mongos> db.MyCollection.find().explain('executionStats')**  {  "queryPlanner" : {  "mongosPlannerVersion" : 1,  "winningPlan" : {  "stage" : "SHARD\_MERGE",  "shards" : [  {  "shardName" : "rs-shard-03",  "connectionString" : "rs-shard-03/shard03-a:27017,shard03-b:27017,shard03-c:27017",  "serverInfo" : {  "host" : "89ef7dca3dbe",  "port" : 27017,  "version" : "4.0.17",  "gitVersion" : "0bc918c73390f6e4d6349660e4cd233f5900b69a"  },  "plannerVersion" : 1,  "namespace" : "MyDatabase.MyCollection",  "indexFilterSet" : false,  "parsedQuery" : {    },  "winningPlan" : {  "stage" : "SHARDING\_FILTER",  "inputStage" : {  "stage" : "COLLSCAN",  "direction" : "forward"  }  },  "rejectedPlans" : [ ]  },  {  "shardName" : "rs-shard-02",  "connectionString" : "rs-shard-02/shard02-a:27017,shard02-b:27017,shard02-c:27017",  "serverInfo" : {  "host" : "25cc34b76d1b",  "port" : 27017,  "version" : "4.0.17",  "gitVersion" : "0bc918c73390f6e4d6349660e4cd233f5900b69a"  },  "plannerVersion" : 1,  "namespace" : "MyDatabase.MyCollection",  "indexFilterSet" : false,  "parsedQuery" : {    },  "winningPlan" : {  "stage" : "SHARDING\_FILTER",  "inputStage" : {  "stage" : "COLLSCAN",  "direction" : "forward"  }  },  "rejectedPlans" : [ ]  },  {  "shardName" : "rs-shard-01",  "connectionString" : "rs-shard-01/shard01-a:27017,shard01-b:27017,shard01-c:27017",  "serverInfo" : {  "host" : "160b5995b01c",  "port" : 27017,  "version" : "4.0.17",  "gitVersion" : "0bc918c73390f6e4d6349660e4cd233f5900b69a"  },  "plannerVersion" : 1,  "namespace" : "MyDatabase.MyCollection",  "indexFilterSet" : false,  "parsedQuery" : {    },  "winningPlan" : {  "stage" : "SHARDING\_FILTER",  "inputStage" : {  "stage" : "COLLSCAN",  "direction" : "forward"  }  },  "rejectedPlans" : [ ]  }  ]  }  },  "executionStats" : {  "nReturned" : 2000,  "executionTimeMillis" : 10,  "totalKeysExamined" : 0,  "totalDocsExamined" : 2000,  "executionStages" : {  "stage" : "SHARD\_MERGE",  "nReturned" : 2000,  "executionTimeMillis" : 10,  "totalKeysExamined" : 0,  "totalDocsExamined" : 2000,  "totalChildMillis" : NumberLong(6),  "shards" : [  {  "shardName" : "rs-shard-03",  "executionSuccess" : true,  "executionStages" : {  "stage" : "SHARDING\_FILTER",  "nReturned" : 682,  "executionTimeMillisEstimate" : 0,  "works" : 684,  "advanced" : 682,  "needTime" : 1,  "needYield" : 0,  "saveState" : 5,  "restoreState" : 5,  "isEOF" : 1,  "invalidates" : 0,  "chunkSkips" : 0,  "inputStage" : {  "stage" : "COLLSCAN",  "nReturned" : 682,  "executionTimeMillisEstimate" : 0,  "works" : 684,  "advanced" : 682,  "needTime" : 1,  "needYield" : 0,  "saveState" : 5,  "restoreState" : 5,  "isEOF" : 1,  "invalidates" : 0,  "direction" : "forward",  "docsExamined" : 682  }  }  },  {  "shardName" : "rs-shard-02",  "executionSuccess" : true,  "executionStages" : {  "stage" : "SHARDING\_FILTER",  "nReturned" : 689,  "executionTimeMillisEstimate" : 0,  "works" : 691,  "advanced" : 689,  "needTime" : 1,  "needYield" : 0,  "saveState" : 5,  "restoreState" : 5,  "isEOF" : 1,  "invalidates" : 0,  "chunkSkips" : 0,  "inputStage" : {  "stage" : "COLLSCAN",  "nReturned" : 689,  "executionTimeMillisEstimate" : 0,  "works" : 691,  "advanced" : 689,  "needTime" : 1,  "needYield" : 0,  "saveState" : 5,  "restoreState" : 5,  "isEOF" : 1,  "invalidates" : 0,  "direction" : "forward",  "docsExamined" : 689  }  }  },  {  "shardName" : "rs-shard-01",  "executionSuccess" : true,  "executionStages" : {  "stage" : "SHARDING\_FILTER",  "nReturned" : 629,  "executionTimeMillisEstimate" : 0,  "works" : 631,  "advanced" : 629,  "needTime" : 1,  "needYield" : 0,  "saveState" : 4,  "restoreState" : 4,  "isEOF" : 1,  "invalidates" : 0,  "chunkSkips" : 0,  "inputStage" : {  "stage" : "COLLSCAN",  "nReturned" : 629,  "executionTimeMillisEstimate" : 0,  "works" : 631,  "advanced" : 629,  "needTime" : 1,  "needYield" : 0,  "saveState" : 4,  "restoreState" : 4,  "isEOF" : 1,  "invalidates" : 0,  "direction" : "forward",  "docsExamined" : 629  }  }  }  ]  }  },  "serverInfo" : {  "host" : "433f1ff10afd",  "port" : 27017,  "version" : "4.0.17",  "gitVersion" : "0bc918c73390f6e4d6349660e4cd233f5900b69a"  },  "ok" : 1,  "operationTime" : Timestamp(1587282930, 1),  "$clusterTime" : {  "clusterTime" : Timestamp(1587282935, 2),  "signature" : {  "hash" : BinData(0,"AAAAAAAAAAAAAAAAAAAAAAAAAAA="),  "keyId" : NumberLong(0)  }  }  } |

* ¿Qué resultados se obtienen? ¿Cuál es la explicación en lo relativo al sharding?

La consulta devuelve todos los registros desordenados ya que ésta ha sido repartida entre tres shards distintos según el campo supplierId. Cada shard ha tratado más o menos unas 630 consultas según vemos en las trazas que se pueden visualizar gracias al valor “executionStats” del parámetro “verbosity” del método explain().

### Dificultades encontradas y opinión del ejercicio

Este ejercicio es el que más dificultad ha tenido para mí. Y no en cuanto a dificultad técnica (puesto que había un tutorial que nos decía exactamente lo que hacer) sino en cuanto a la interpretación de los logs.

La parte positiva es que he podido ver practicando con mis propias manos la arquitectura de clustering en mongodb, pero he echado en falta alguna explicación previa en cuanto a la interpretación de los logs, por sacarle alguna pega al ejercicio.