Proyecto Final MongoDB

Almacenamiento y Gestión de la Información

Adrián Moreno Monterde
Pablo Cordón Hidalgo

Índice

1. Tecnologías utilizadas 2

2. Base de Datos 2

3. Código 4

4. Instrucciones para instalación y despliegue de la aplicación 7

5. Conclusiones personales 13

1. Tecnologías utilizadas

Java

Para la realización de este proyecto, se ha optado por usar el conocido lenguaje de programación *Java*, el cual nos aporta entre otras cosas, un *driver* con el que la conexión y consultas a MongoDB se realizan de manera muy sencilla y librerías de diseño de interfaz gráfico que nos ayudaran a la hora de representar los datos. En cuanto al desarrollo de la aplicación se ha usado el IDE *Netbeans*, concretamente la versión 12.4.

Github

Github[1] ha sido la plataforma utilizada en donde, de manera colaborativa, se ha ido desarrollando el proyecto de principio a fin. Al mismo tiempo se ha podido tener un control sobre las diferentes fases y versiones por las que ha ido pasando la aplicación.

Librería Java Swing

Una de las múltiples librerías que nos aporta *Java*, es la llamada *Swing*. Esta librería, está integrada con el IDE *Netbeans*, por lo que su utilización es bastante sencilla. A la hora utilizar esta librería solo hay que tener en cuenta que son contenedores (vistas) que almacenan componentes (etiquetas, campos de texto, etc.).

Patrón de diseño Modelo-Vista-Controlador

Se ha utilizado la propuesta de arquitectura de software Modelo-Vista-Controlador (MVC), la cual se basa en separar el código en función de las responsabilidades que tenga, por lo cual nuestro modelo tendrá las siguientes capas:

- **Modelo** → Contiene la funcionalidad necesaria para acceder a la información, en nuestro caso tweets, e incluso modificarla.
- **Vista** → Contiene las diferentes interfaces de usuario que han sido creadas con la librería Swing.
- **Controlador** → Es la capa que actúa de enlace entre las vistas y los modelos.

2. Base de Datos

La base de datos creada en atlas es un servidor compartido, siendo *Amazon Web Service* (AWS) como proveedor de nube y con ubicación en París, ya que es la mejor región para desplegar servicios desde el oeste de Europa.

El nivel de *clúster* seleccionado ha sido el *M0 Sandbox*, ya que es el único gratuito. Este nos aporta 512MB de almacenamiento y un conjunto de réplicas formatos por 3 servidores de soporte de datos.

[1] https://github.com/visumania/ProyectoFinalAGI

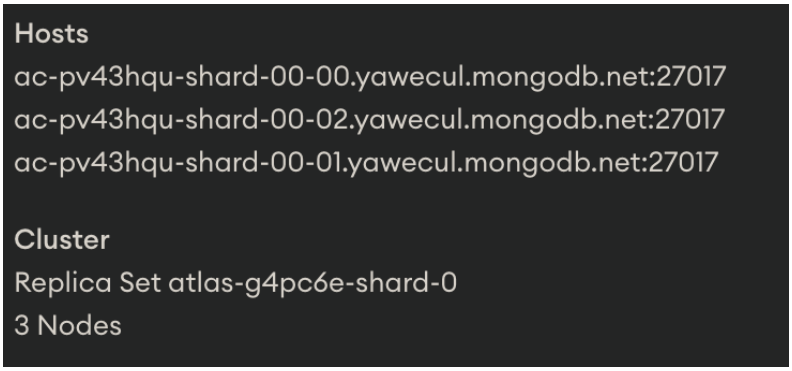

Ilustración 1.- Información del servidor

Mecanismos de seguridad

En cuanto a la seguridad, hemos implementado en la base de datos la autenticación y la autorización mediante el mecanismo SCRAM (*Challenge Response Authentication Mechanism*).

Se han creado 2 usuarios: un usuario administrador, el cual podrá administrar a otros usuarios y otro que solo tiene permisos de lectura y escritura sobre la base de datos Twitter.

```
db.createUser({
  user:"pablian",
  password:"pablian",
  roles:[
    {role:"root" db:"admin"},
    {role:"hostManager", db:"admin"}
  ]
})

db.createUser({
  user:"normal",
  password:"normal",
  roles:[
    {role:"readWrite", db:"Twitter"}
  ]
})
```

Ilustración 2.- Creación de usuarios de la base de datos

Estructura de los documentos

Al habernos descargado los tweets en formato simplificado, tenemos solamente 8 campos, los cuáles se pueden ver en la Ilustración 3. Como se podrá ver en los siguientes apartados, hemos desarrollados nuestras consultas en base a estos 8 campos que nos proporcionan los tweets en formato simplificado.

```
{
  "_id": {
    "$oid": "639ade58d716f019cfabef65"
  },
  "id": "1603309904638484482",
  "username": "_dgrzk",
  "followers": 88,
  "text": "RT @Sport360FootAnother vintage Messi moment ✨\n\nPure genius from the 🐐\n\n#GOAT #Qatar2022 https://t.co/HPGdQoN3ZU",
  "hashtags": [
    {
      "text": "GOAT",
      "indices": [
        74,
        79
      ]
    },
    {
      "text": "Qatar2022",
      "indices": [
        80,
        90
      ]
    }
  ],
  "language": "en",
  "created": {
    "$date": {
      "$numberLong": "1671093843000"
    }
  }
}
```


Ilustración 3.- Estructura de los documentos

3. Código

En esta sección se expondrán las diferentes consultas hechas en MongoDB de forma nativa para cumplir con las diferentes funcionalidades que tienen las pestañas de estadísticas y de consultas.

Funcionalidad de estadísticas

```
db.tweets.countDocument()
```

Ilustración 4.- Número de tweets que almacena la colección

```
db.tweets.aggregate([
  {$sort:{created:1}},
  {$project:{_id:0, created:1}},
  {$limit:1}
  ])

db.tweets.aggregate([
  {$sort:{created:-1}},
  {$project:{_id:0, created:1}},
  {$limit:1}
  ])
```

Ilustración 5.- Periodo de descargas. Fechas del primer y último tweet almacenado

```
db.tweets.aggregate([
  {$sort:{followers:-1}},
  {$project:{_id:0, username:1, followers:1}},
  {$limit:1}
  ])
```

Ilustración 6.- Usuario con más seguidores y número de seguidores

Como se puede observar en la Ilustración 7, esa consulta corresponde a la sección de estadísticas, donde se devuelven a los usuarios más mencionados. Exactamente esa consulta devuelve los textos de todos los tweets, pero después en *Java* se ha *tokenizado* cada tweet y se ha recorrido cada *token* para comprobar si era una mención para añadirlo a un *Map* compuesto por el nombre de usuario y el número de menciones que se iba contabilizando.

Del mismo modo, la consulta de la Ilustración 7 se ha usado para contabilizar el número de *retweets* que se tiene almacenado. Ya que después en *Java* se trata el resultado de la consultas de manera que si el campo texto empieza con "RT @" se contabiliza como *retweet*.

```
db.tweets.aggregate([
  {$project:{_id:0, text:1}}
  ])
```

Ilustración 7.- Usuarios más mencionados

```
db.tweets.aggregate([
  {$unwind:"$hashtags"},
  {$group:{_id:"$hashtags.text", frecuencia:{$sum:1}}},
  {$sort:{frecuencia:-1}},
  {$limit:5}
  ])
```

Ilustración 8.- Consulta que devuelve el top 5 Hashtags más frecuentes junto con su frecuencia

```
db.tweets.aggregate([
  {$group:{_id:"$language"}},
  {$count:"Numero de idiomas diferentes"}
  ])
```

Ilustración 9.- Consulta que devuelve el número de idiomas diferentes

```
db.tweets.aggregate([
  {$group:{_id:"$language", count:{$sum:1}}},
  {$sort:{count:-1}},
  {$limit:5}
  ])
```

Ilustración 10.- Consulta que devuelve top 5 idiomas más frecuentes

```
db.tweets.aggregate([
  {$group:{_id:"$username"}},
  {$count:"Número de usuarios diferentes: "}
  ])
```

Ilustración 11.- Consulta que devuelve el número de usuarios diferentes que almacena la colección

```
db.tweets.aggregate([
  {$group:{_id:"$username", count:{$sum:1}}},
  {$sort:{count:-1}},
  {$limit:5}
  ])
```

Ilustración 12.- Consulta que devuelve el top 5 de usuarios de los que se almacenan más tweets

Funcionalidad de consultas

Para esta sección, se ha creado un método en *Java* que recibe el nombre de *listaTweets*, el cual en función del valor de los parámetros con el que invoquemos al método, tendrá diferentes condiciones de búsqueda.

Nuestra consulta parametrizada solamente tiene un campo obligatorio, que es el del rango de seguidores, el cuál por defecto siempre será un campo que en la aplicación esté relleno con valores. El resto de los campos que tienen influencia en la condición de búsqueda son:

- Nombre de usuario (*username*)
- Hashtag (*hashtags.text*)
- Palabra específica del tweet (*text*)
- Idioma/s (*language*)

En la Ilustración 13 se muestra se muestra cómo se realizaría una consulta de manera nativa sobre MongoDB.

```
db.tweets.aggregate([
  {
    $match:{
      $and:[
        {followers:{$gte:1, $lte:1500000}},
        {username:"visumania"},
        {"hashtags.text":"UHU"},
        {$text:{$search:"Tharsis"}},
        {language:{$in:["en","es"]}}
        ]}
  }
])
```

Ilustración 13.- Consulta parametrizada completa de manera nativa

No obstante, también se han implementado en *Java* otras consultas cuya función es la de rellenar distintos objetos pertenecientes a la librería *Java Swing*. Por ejemplo, para rellenar el *JList* en función de la colección con la que estemos trabajando se ha implementado el método que se puede observar en la Ilustración 14.

```
db.tweets.aggregate([
  {$group:{_id:"$language"}},
  {$sort:{_id:1}}
  ])
```

Ilustración 14.- Obtener el listado de idiomas

También se han implementado dos métodos para rellenar el contenido de los *ComboBox* con el nombre de las colecciones (Ilustración 15) y el contenido del *JSpinner* del mínimo de la búsqueda por rango de seguidores (Ilustración 16).

```
db.getCollectionNames()
```

Ilustración 15.- Obtener nombre de todas las colecciones

```
db.tweets.aggregate([
  {$sort:{followers:1}},
  {$limit:1}
  ])
```

Ilustración 16.- Obtener el usuario con menos seguidores

Con respecto a los índices creados, para realizar la búsqueda por el contenido que tiene un tweet, se ha creado un índice de texto en inglés, ya que el idioma predominante. No se ha podido crear un índice de texto general porque nos hemos encontrado con idiomas que no son compatibles con MongoDB como el tailandés.

```
db.tweets.createIndex(
  { "text": "text" },
  { default_language: "english", language_override: "english" }
)
```

Ilustración 17.- Índice de texto creado

También se han creado un índice ascendente y otro descendente sobre el campo de *username*, De manera que el tiempo ejecución de las búsquedas que incluyen ese campo disminuye.

```
db.tweets.createIndex({username:1},{name:"usernameUp"})
db.tweets.createIndex({username:-1},{name:"usernameDown"})
```

Ilustración 18.- Creación de índices sobre el campo username

Las Ilustraciones 19 y 20 nos muestran las diferencias de tiempo de ejecución de cuando se usa y no se usa uno de los índices creados para el campo *username*.

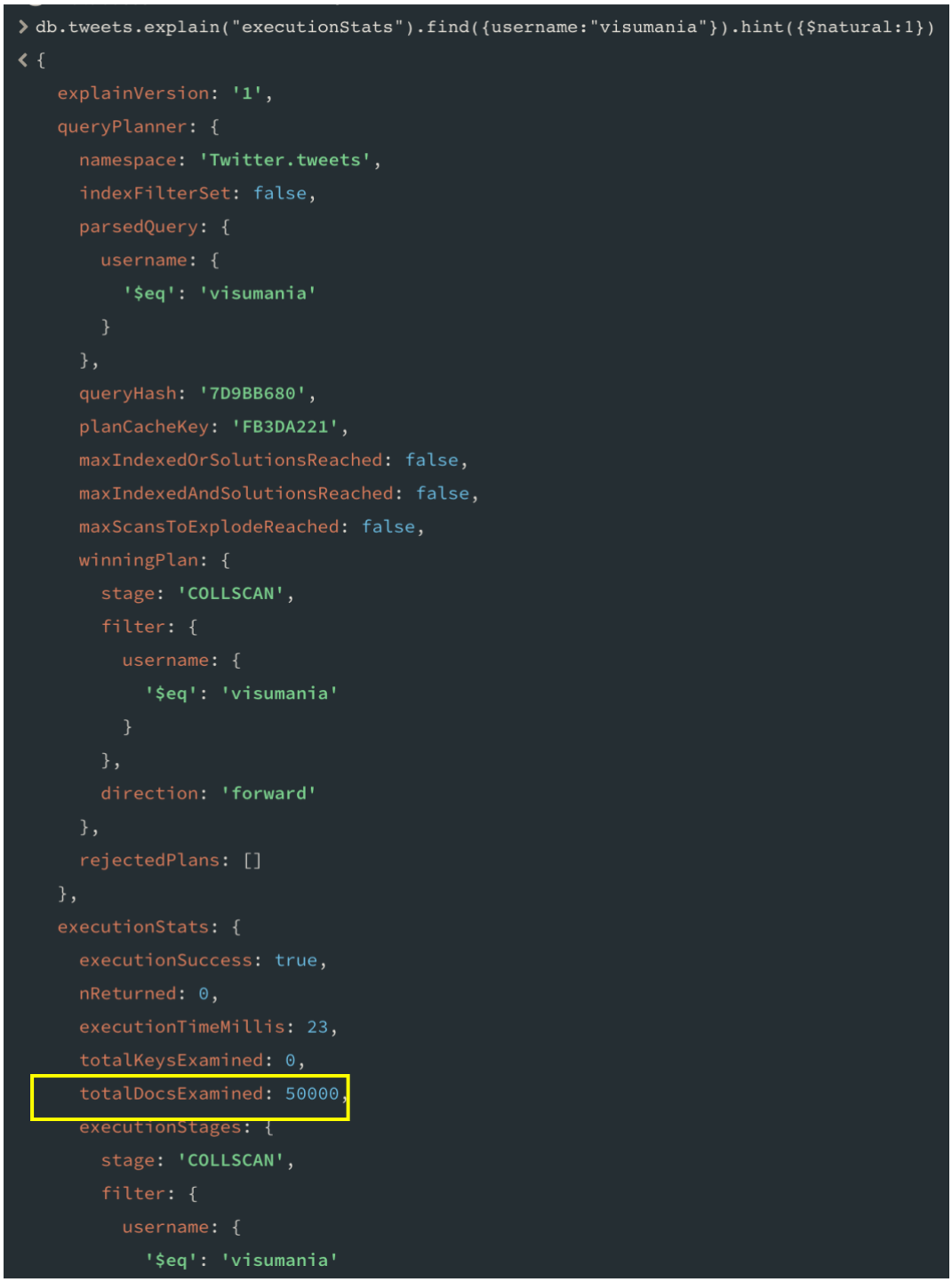

Ilustración 19.- Búsqueda por nombre de usuario sin utilizar índices

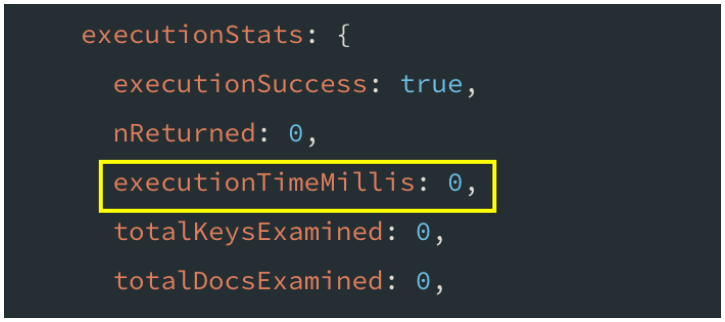

Ilustración 20.- Búsqueda de nombre de usuario dejando que mongo elija que índice usar

Del mismo modo se han creado dos índices (uno ascendente y otro descendente) sobre el campo *hashtags.text*, como se puede observar en la Ilustración 21 y uno descendente sobre el campo *followers*, el cual se puede observar en la Ilustración 22.

```
db.tweets.createIndex({"hashtags.text":1},{name:"hashtags.textUp"})
db.tweets.createIndex({"hashtags.text":-1},{name:"hashtags.textDown"})
```

Ilustración 21.- Creación de índices sobre el campo hashtags.text

```
db.tweets.createIndex({followers:-1},{name:"followersDown"})
```

Ilustración 22.- Creación de índice descendente sobre el campo followers

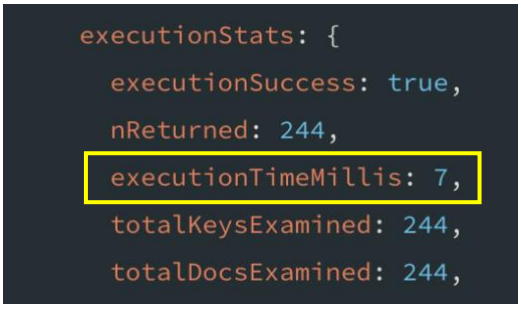

Ilustración 23.- Búsqueda sobre el campo hashtags.text usando el índice que decida MongoDB

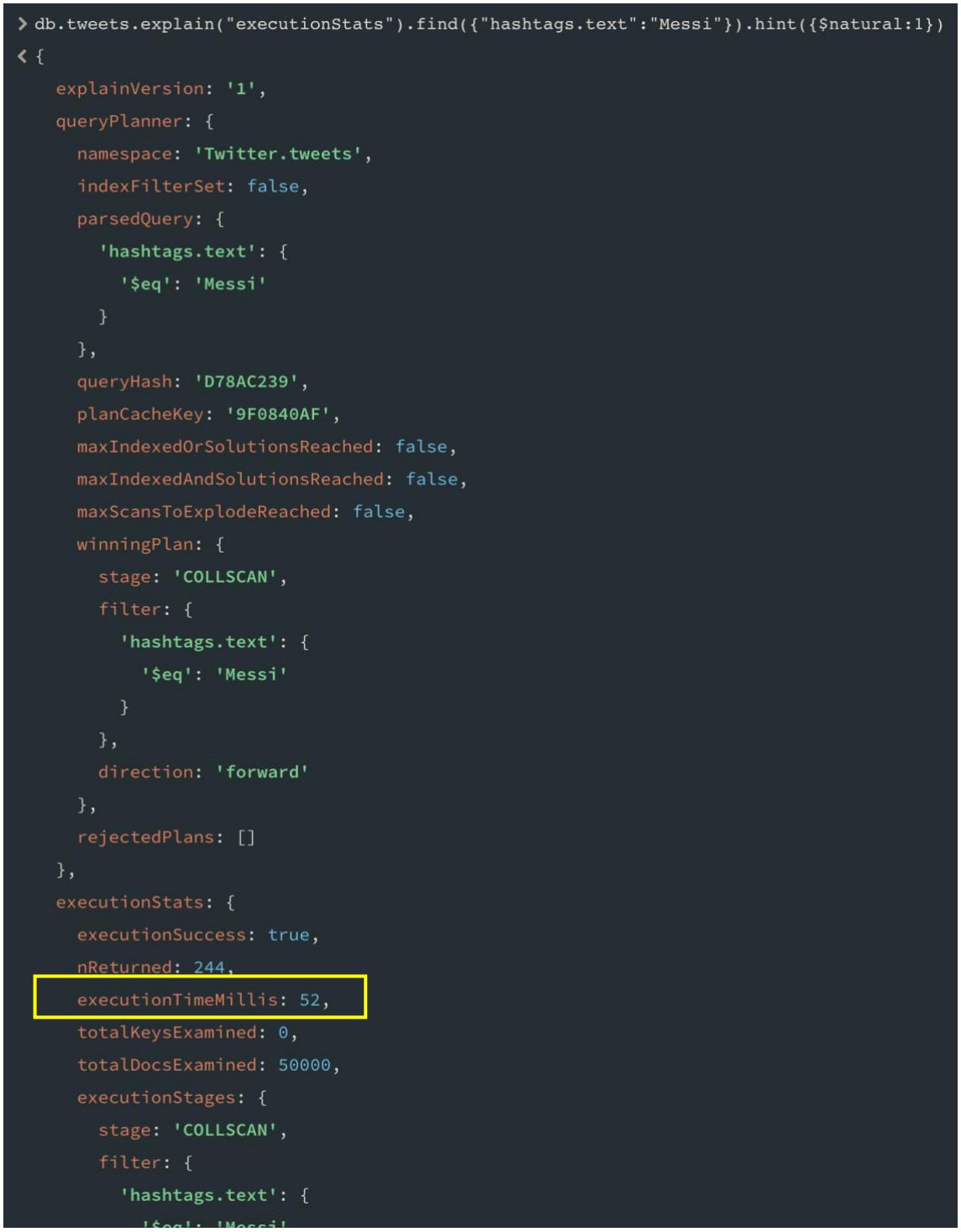

Ilustración 24.- Búsqueda sobre el campo hashtags.text sin usar ningún índice

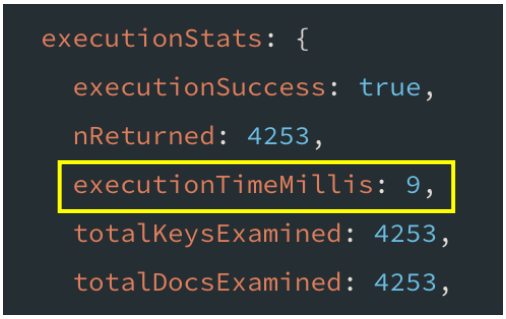

Ilustración 25.- Consulta sobre el campo followers usando el índice creado

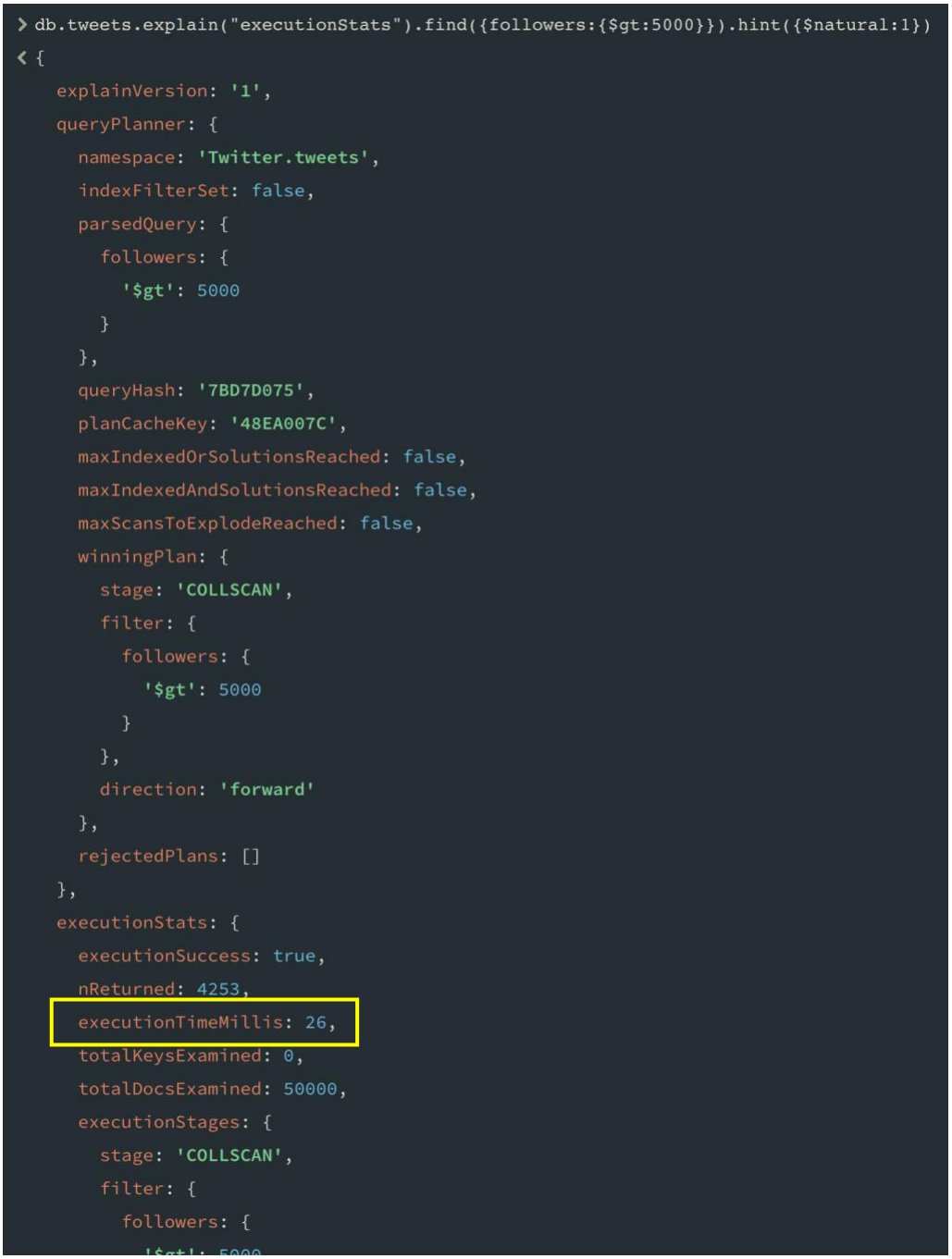

Ilustración 26.- Consulta sobre el campo followers sin usar el índice creado

La creación del índice descendente sobre el campo *followers*, además de aportarnos una mejora en los tiempos de búsqueda, también nos aporta el detalle de que cuando mostramos el listado de los tweets cuando desplegamos la pestaña de consultas, estos aparecen ordenados en función al campo *followers* de manera descendente.

4. Instrucciones para instalación y despliegue de la aplicación

Para el despliegue de la instalación simplemente se requiere que tenga inicializado el fichero de configuración de MongoDB como se puede apreciar en la Ilustración 27. Seguidamente al ejecutar la aplicación el usuario podrá elegir si conectarse a una instancia local, indicando el puerto o a un *clúster*, insertando el URI.

Seguidamente, es necesario que se tenga creada una base de datos llamada Twitter, con una colección llamada tweets, la cual será el punto de partida para desplegar la sección de estadísticas y de crear nuevas colecciones a partir de las consultas parametrizadas. Esta contendrá los 50000 tweets que fueron descargados directamente desde el *script* de *Python*.

```
mongod --config /usr/local/etc/mongod.conf
```

Ilustración 27.- Inicialización del fichero de configuración de MongoDB

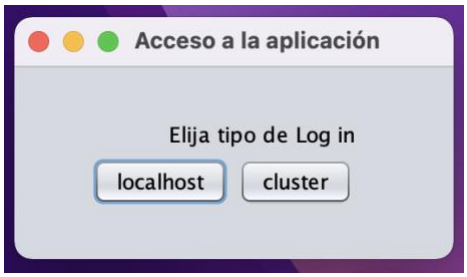

Ilustración 28.- Elección del tipo de conexión

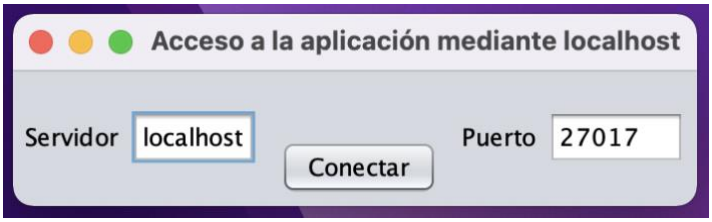

Ilustración 29.- Conexión de manera local

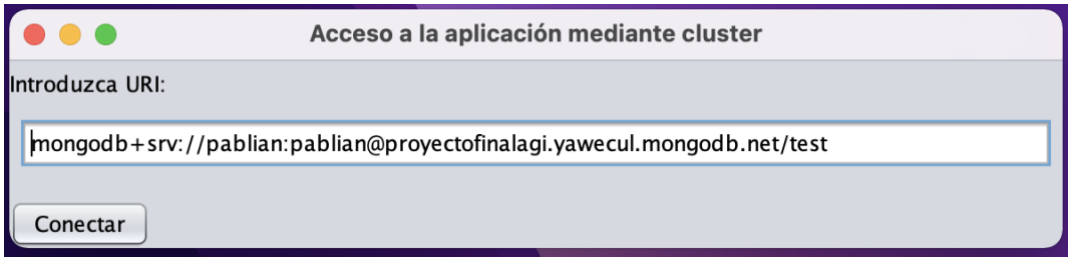

Ilustración 30.- Conexión a un clúster mediante la inserción de un URI

5. Conclusiones personales

Como conclusión nos gustaría remarcar la importancia que tiene hoy en día la gestión de los datos, tanto como el almacenamiento (usando una base de datos MongoDB) como su representación (haciendo uso de la librería *Java Swing*).

Al trabajar MongoDB con documentos en vez de tablas, nos aporta esa flexibilidad a la hora de modelar los datos que no nos aportan las bases de datos relacionales. Además, como se ha podido ver en los anteriores apartados, la creación de los índices optimiza de manera notable las consultas, por lo que, aunque se trabaje con grandes volúmenes de datos, se puede seguir manteniendo un tiempo eficiente en las diferentes operaciones que se realicen sobre la base de datos.

Como trabajo futuro, proponemos la gestión de la información de todos los metadatos de un tweet (al completo), lo cual nos podría permitir la posibilidad de generar muchas estadísticas adicionales a las implementadas y realizar consultas en función a otros campos diferentes a las realizadas en este proyecto.