

Ejercicio 7: Gestionando la seguridad en MongoDB

1. Cread un usuario "administrador" con login "super" y el password que queráis. Al menos debe tener los roles de "root" y "hostManager"

```
db.createUser({
  user: "super",
  pwd: "superpwd",
  roles: [
    {role: "root", db: "admin"},
    {role: "hostManager", db: "admin"}
  ]
})
```

2. Parad la instancia de MongoDB y arrancarla con el modo de seguridad activo.

Apagamos la instancia de MongoDB

```
db.adminCommand( { shutdown: 1 } )
```

Esta opción ha producido un "apagado inseguro", dejando bloqueado el arranque de la base de datos.

Para resolverlo, en Linux hemos optado por borrar el socket y reiniciar el servicio del sistema.

```
sudo rm /tmp/mongodb-27017.sock
sudo systemctl start mongod
```

Tras solventar el problema, activamos la autenticación

```
mongod --auth
```

3. Comprobad que lo habéis hecho bien. Para ello, intentad conectaros a la instancia sin autenticaros. En Robomongo no os debe permitir la conexión y, desde la consola, podréis entrar pero os debe dar error si intentáis ejecutar alguna sentencia (por ejemplo show collections)

- Inicio de la shell sin autenticación

Entramos en la shell de mongo sin autenticación

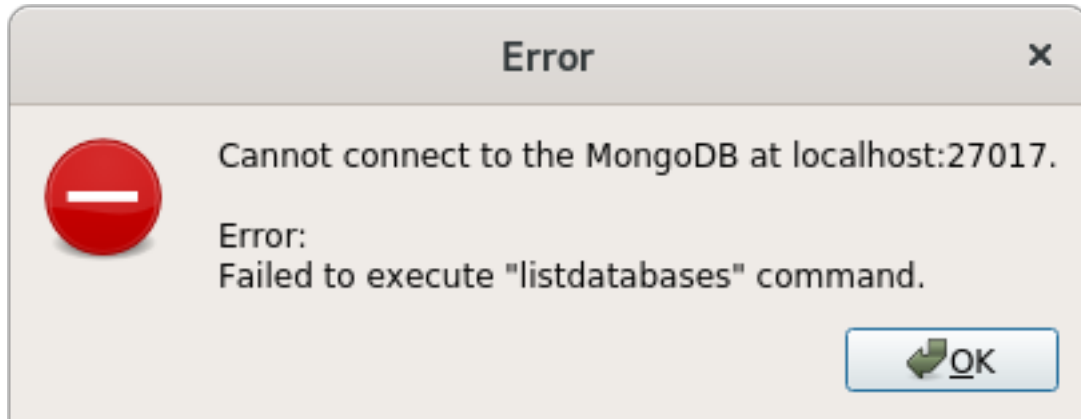
```
almu@debian:~$ mongo
MongoDB shell version v4.4.1
connecting to: mongodb://127.0.0.1:27017/?compressors=disabled&gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("22004e9f-f577-4572-b031-05f85cdab7fe") }
MongoDB server version: 4.4.1
> use elPais
switched to db elPais
> show collections
Warning: unable to run listCollections, attempting to approximate collection names by parsing
> db.noticias.find({})
uncaught exception: ReferenceError: noticias is not defined :
@(shell):1:4
> db.noticias
elPais.noticias
> db.getCollection('noticias').find({})
Error: error: {
  "ok" : 0,
  "errmsg" : "command find requires authentication",
  "code" : 13,
  "codeName" : "Unauthorized"
}
```

>

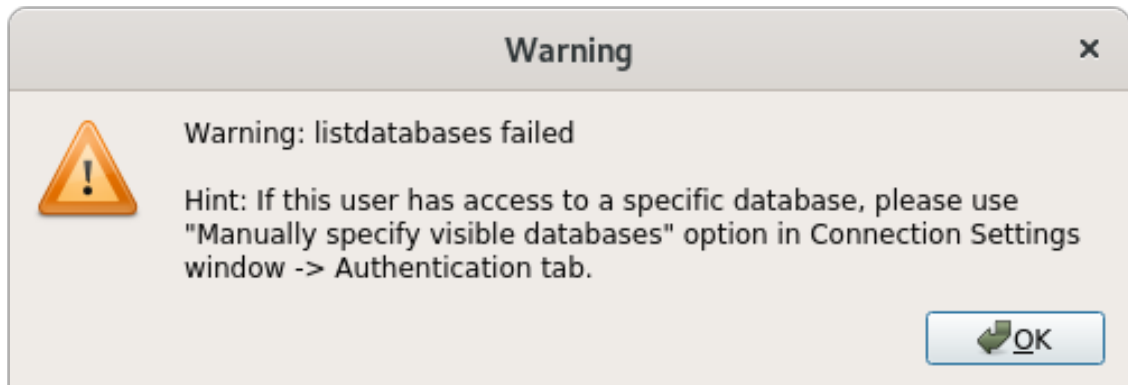
Vemos como algunos comandos, como `show collections`, fallan; y en algunos, como en la consulta sobre la colección `noticias`, se nos indica que se necesita autenticación.

- **Intento de inicio de sesión en Robo3T sin autenticación**

En Robo3T, al intentar iniciar sin autenticación, se nos muestra un error



También nos muestra un Warning, alertando de la posible causa del error



4. Desde la consola, conectarnos a la instancia con autenticación y probad que podéis ejecutarla sentencia del ejercicio anterior

Para iniciar sesión con autenticación, usamos el siguiente comando:

```
mongo --authenticationDatabase "admin" -u "super" -p
```

Introducimos nuestra clave y pulsamos enter para acceder

```
almu@debian:~$ mongo --authenticationDatabase "admin" -u "super" -p
MongoDB shell version v4.4.1
Enter password:
connecting to: mongodb://127.0.0.1:27017/?authSource=admin&compressors=disabled&gssapiServiceName=
Implicit session: session { "id" : UUID("c28e5b9b-a0f9-4447-a40b-9412c3c421e0") }
MongoDB server version: 4.4.1
---
The server generated these startup warnings when booting:
  2020-11-20T00:03:45.454+01:00: ***** SERVER RESTARTED *****
  2020-11-20T00:03:45.467+01:00: Using the XFS filesystem is strongly recommended with the V
  2020-11-20T00:03:46.305+01:00: /sys/kernel/mm/transparent_hugepage/enabled is 'always'. We
---
```

Enable MongoDB's free cloud-based monitoring service, which will then receive and display metrics about your deployment (disk utilization, CPU, operation statistics, etc).

The monitoring data will be available on a MongoDB website with a unique URL accessible to you and anyone you share the URL with. MongoDB may use this information to make product improvements and to suggest MongoDB products and deployment options to you.

To enable free monitoring, run the following command: `db.enableFreeMonitoring()`

To permanently disable this reminder, run the following command: `db.disableFreeMonitoring()`

```
> show collections
```

```
> db
```

```
test
```

```
> show dbs
```

```
Club          0.000GB
```

```
admin          0.000GB
```

```
config         0.000GB
```

```
elPais         0.000GB
```

```
equipos        0.000GB
```

```
estadosGrandes 0.000GB
```

```
local          0.000GB
```

```
zips           0.002GB
```

```
> use elPais
```

```
switched to db elPais
```

```
> db.noticias.find({"etiqueta_principal": "MARTE"})
```

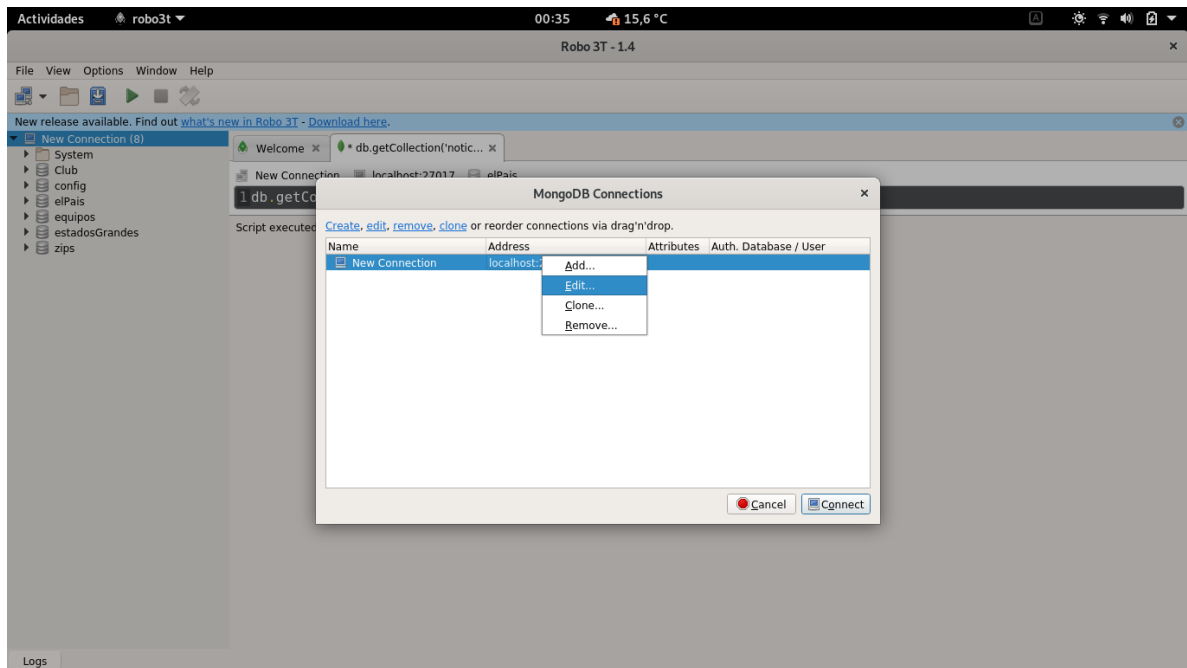
```
{ "_id" : ObjectId("5fa429d2d362ceb92e0f151a"), "etiqueta_principal" : "MARTE", "titular" : "Marte"
```

```
>
```

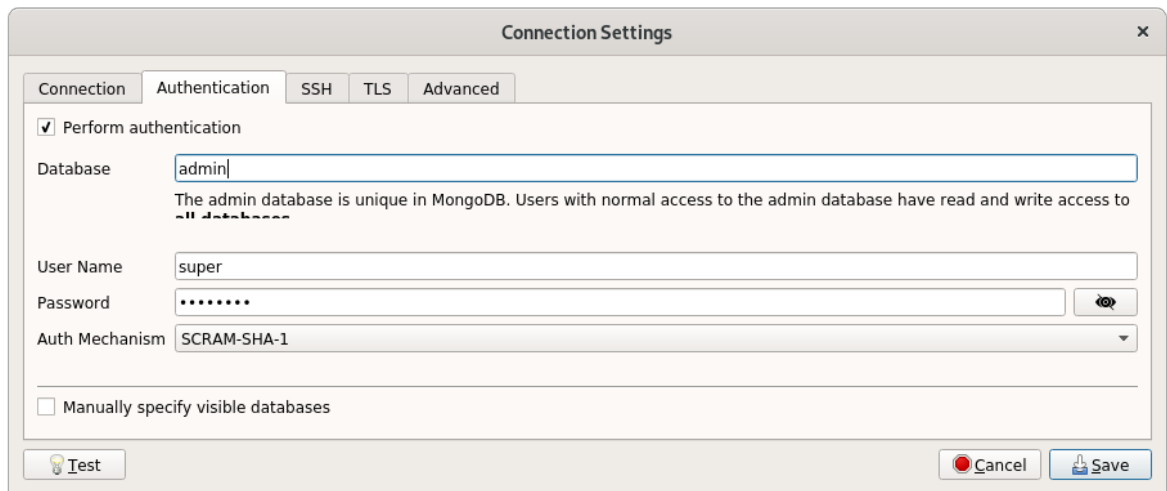
Comprobamos que, esta vez, los comandos funcionan correctamente.

5. Desde Robomongo, cread una nueva conexión con la autenticación del usuario creado en el ejercicio 1 para poder acceder a todas las bases de datos.

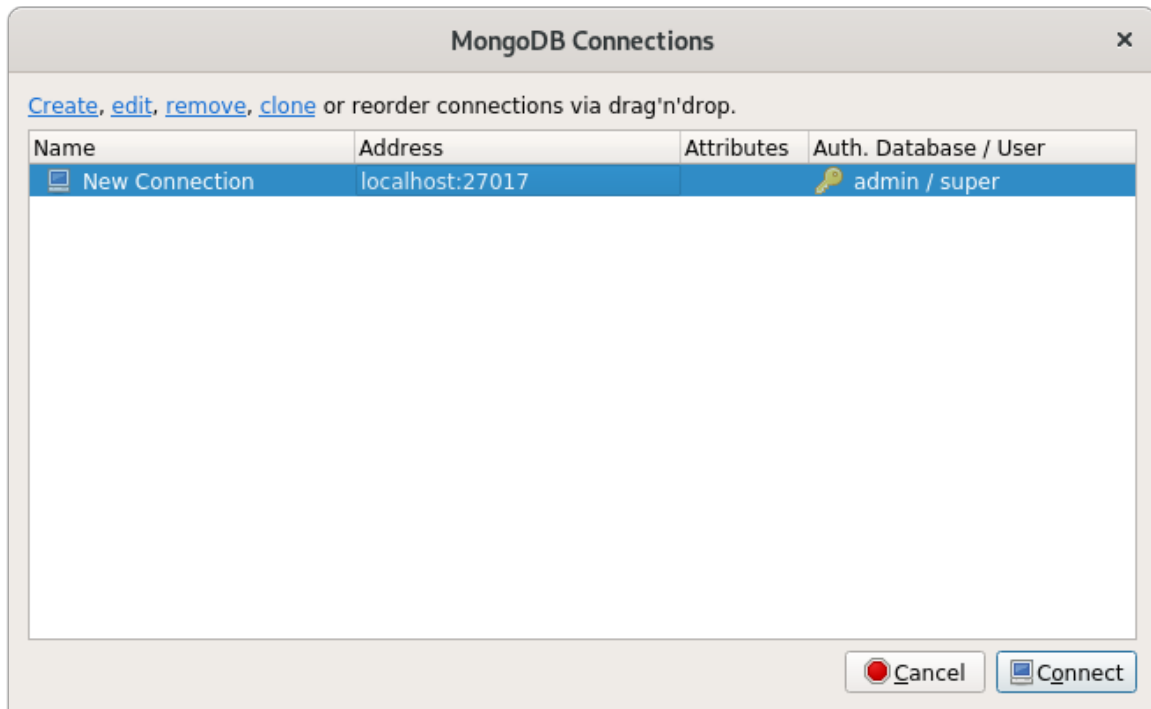
Editamos la conexión anterior, para añadir la autenticación



En el asistente, activamos la autenticación, pulsando en la pestaña "Authentication", y marcando la opción "Perform authentication".

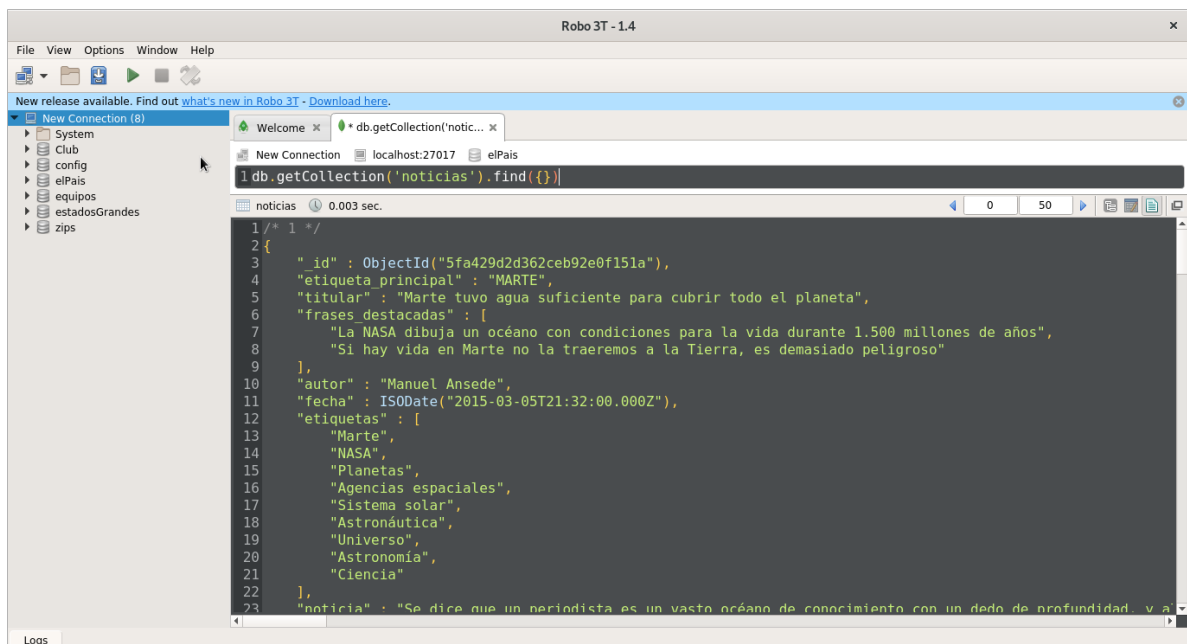


Introducimos nuestra base de datos, nombre de usuario y contraseña, y pulsamos en "Save" para guardar los cambios.



Vemos que los parámetros de la conexión se han modificado, añadiendo la autenticación (último campo a la derecha).

Pulsamos en "Connect" (o hacemos doble clic sobre la conexión) para iniciar sesión.



Vemos que esta vez hemos iniciado sesión correctamente, y las consultas funcionan sin problemas.

6. Cread un usuario (adminPais) que tenga privilegios de lectura y escritura sobre la base de datos "elpais" y otro usuario (usuarioPais) que sólo tenga permiso de lectura.

- Creando usuario "adminPais"

```
> db.createUser({
... user: "adminPais",
... pwd: "admpais",
... roles:[{role: "readWrite", db: "elPais"}]
... })
Successfully added user: {
  "user" : "adminPais",
  "roles" : [
    {
      "role" : "readWrite",
      "db" : "elPais"
    }
  ]
}
>
```

- **Creando usuario "userPais"**

```
> db.createUser({
... user: "userPais",
... pwd: "usrpais",
... roles: [{role: "read", db: "elPais"}]
... })
Successfully added user: {
  "user" : "userPais",
  "roles" : [
    {
      "role" : "read",
      "db" : "elPais"
    }
  ]
}
>
```

- **Iniciando sesión en la shell**

Para iniciar sesión, debemos indicar como base de datos "test".

Con el usuario "userPais" iniciamos así:

```
almu@debian:~$ mongo --authenticationDatabase "test" -u "userPais" -p
MongoDB shell version v4.4.1
Enter password:
connecting to: mongod://127.0.0.1:27017/?authSource=test&compressors=disabled&gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("a4b8aa41-977d-4a3c-aae8-307f40eadaef") }
MongoDB server version: 4.4.1
> exit
bye
```

Y repetimos con "adminPais"

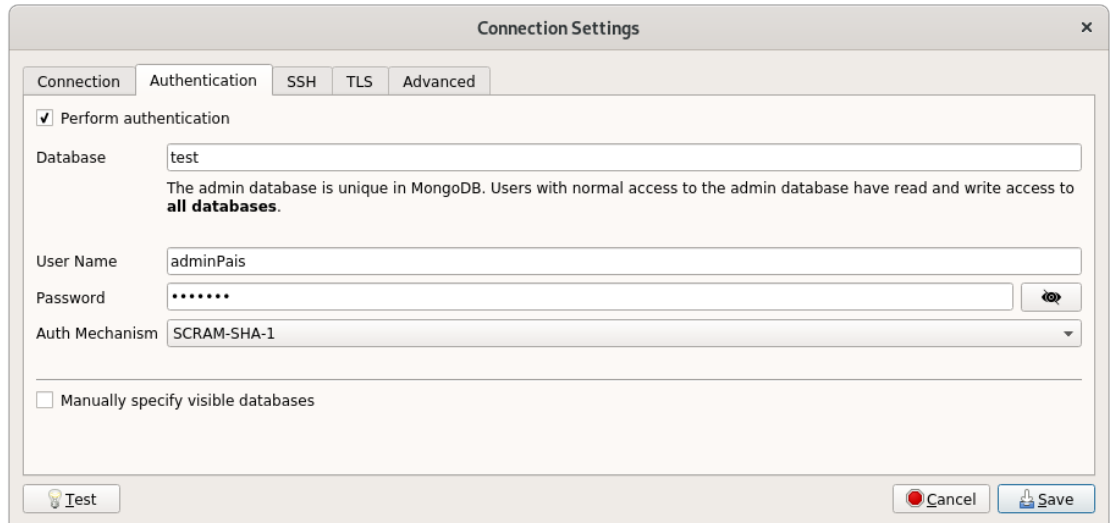
```
almu@debian:~$ mongo --authenticationDatabase "test" -u "adminPais" -p
MongoDB shell version v4.4.1
Enter password:
connecting to: mongod://127.0.0.1:27017/?authSource=test&compressors=disabled&gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("1b0523b6-f9c2-48ec-80da-98aacb27a67d") }
MongoDB server version: 4.4.1
> exit
bye
```

Vemos que el inicio de sesión de ambos usuarios es correcto.

7. Desde Robomongo, cread dos nuevas conexiones con la autenticación de los usuarios creados en el ejercicio anterior y probad que, con la conexión del usuario "adminPais", se puede actualizar un documento pero que, con la de "usuarioPais", no se permite.

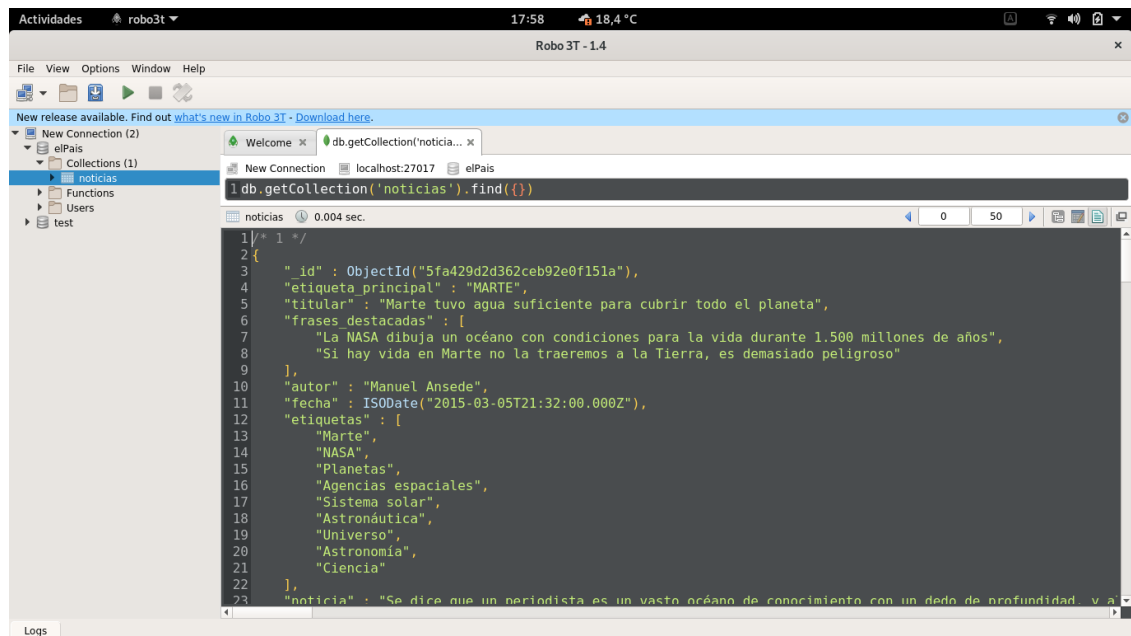
En Robo3T, creamos dos nuevas conexiones, activando la autenticación con los parámetros de nuestros usuarios.

- Iniciando sesión con adminPais



The screenshot shows the 'Connection Settings' dialog box with the 'Authentication' tab selected. The 'Perform authentication' checkbox is checked. The 'Database' field is set to 'test'. Below it, a note states: 'The admin database is unique in MongoDB. Users with normal access to the admin database have read and write access to all databases.' The 'User Name' field is 'adminPais', the 'Password' field is masked with dots, and the 'Auth Mechanism' is 'SCRAM-SHA-1'. There is an unchecked checkbox for 'Manually specify visible databases'. At the bottom, there are 'Test', 'Cancel', and 'Save' buttons.

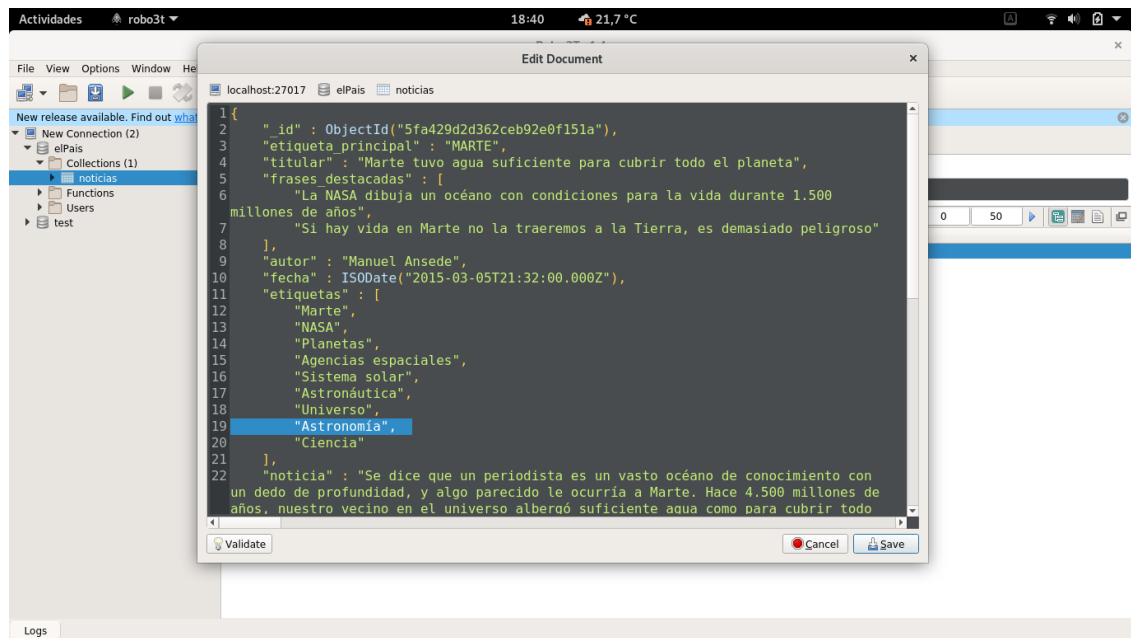
Configuración del usuario adminPais



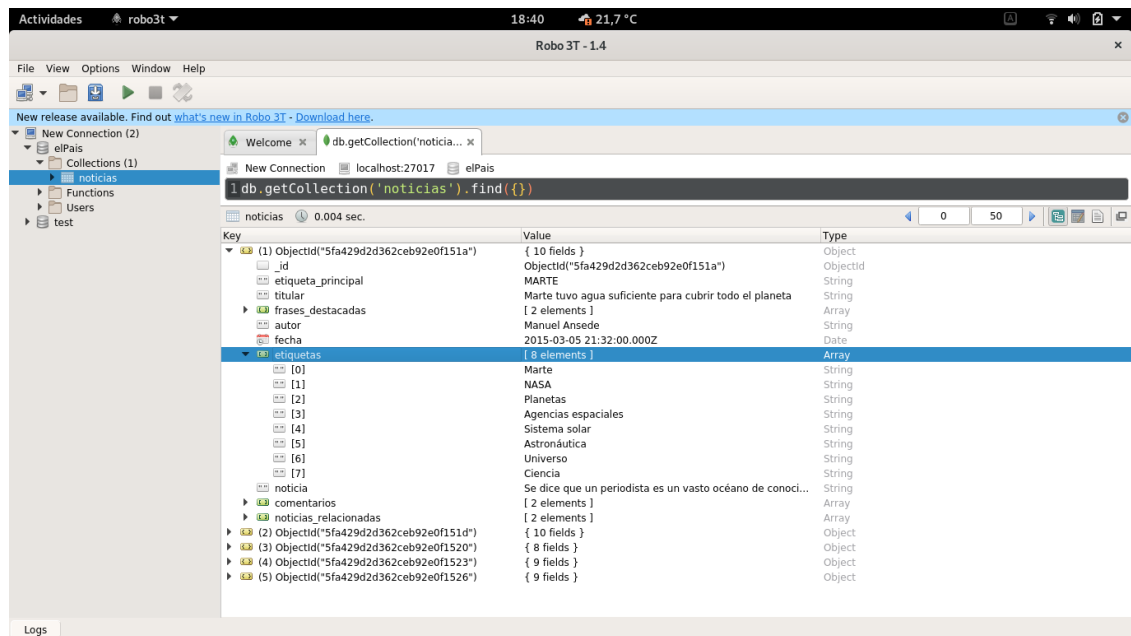
Observamos como, en este usuario, solo se muestran las BBDD "test" y "elPais"

- Editando documento con adminPais

Editamos el documento borrando una de sus etiquetas

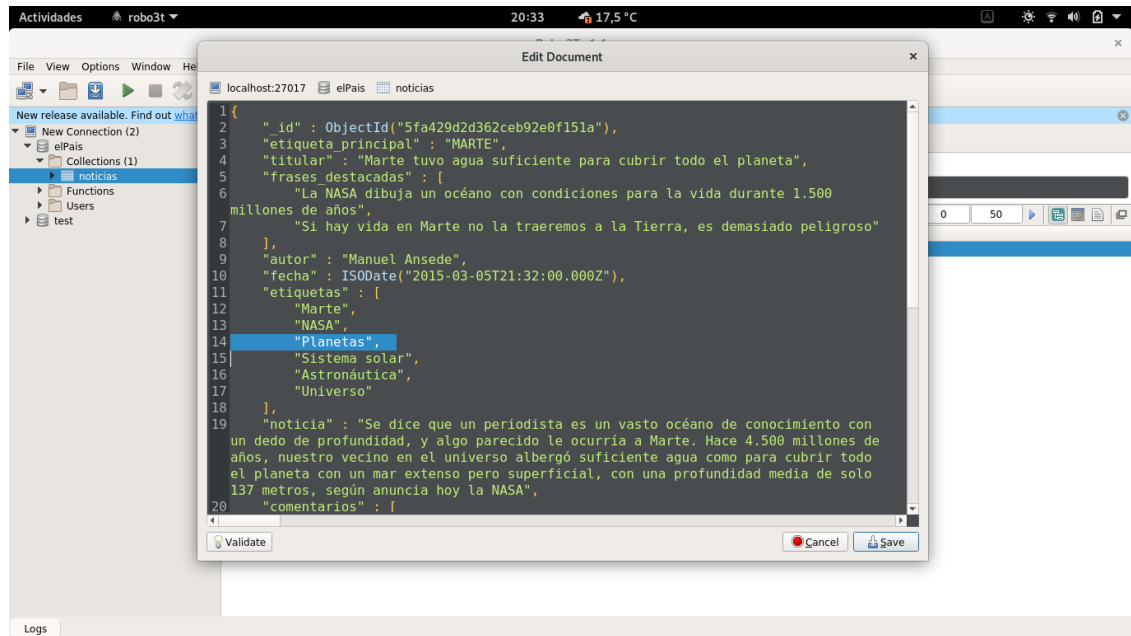


Vemos que la edición se ha realizado correctamente

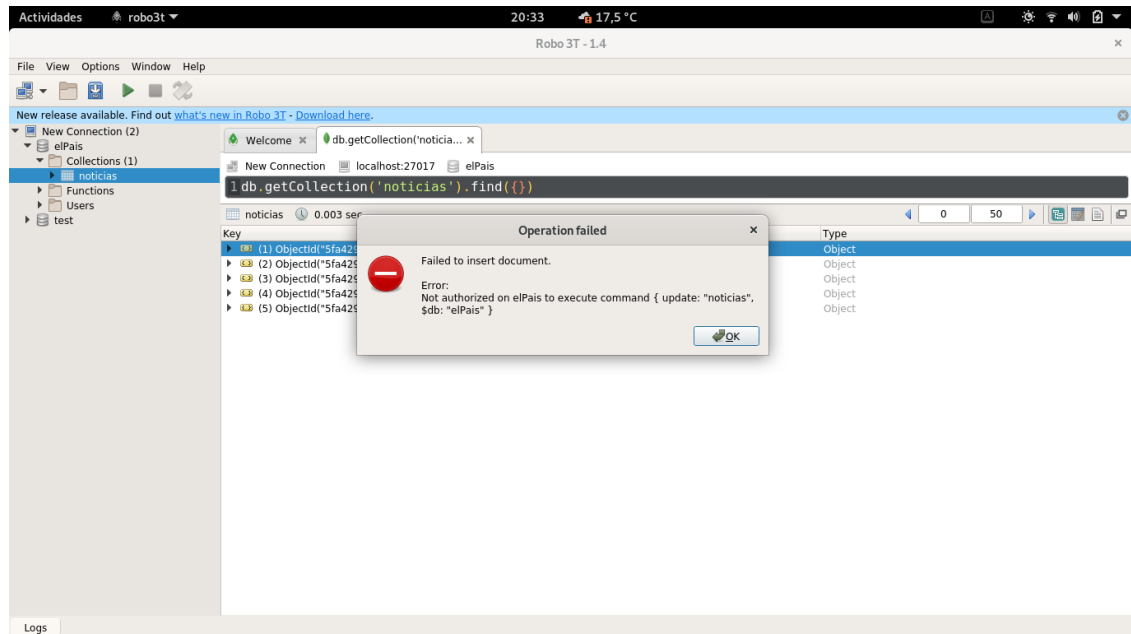


- **Editando documento con userPais**

Repetimos el proceso anterior con el usuario "userPais".



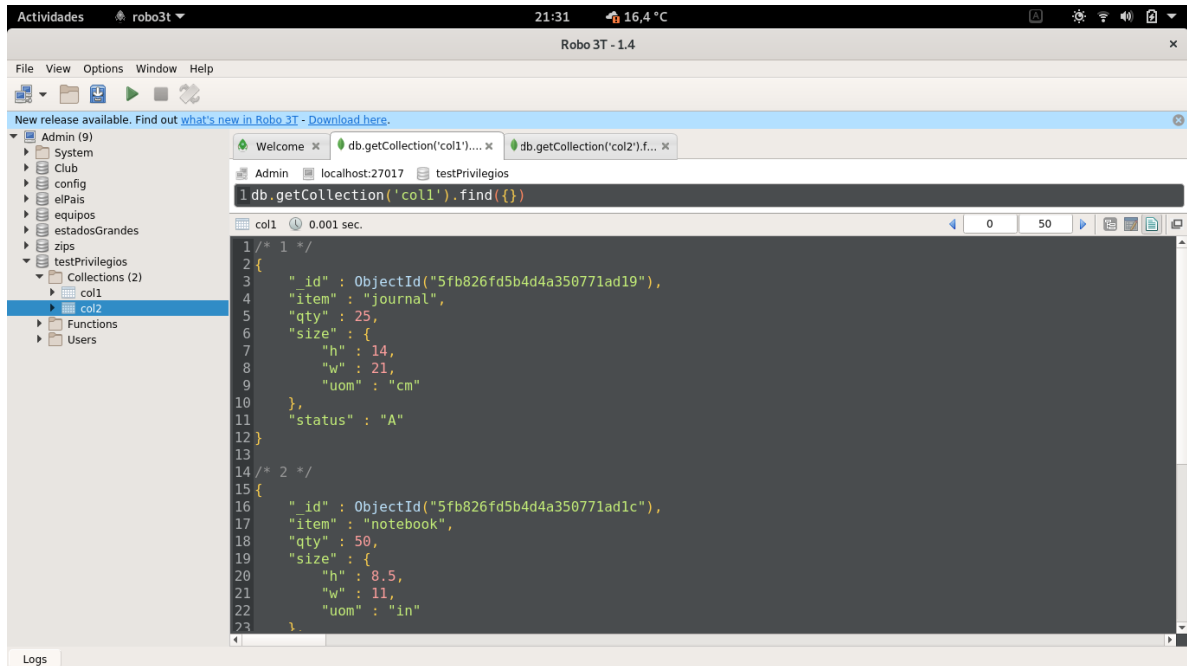
Editamos un documento, borrando una de sus etiquetas, e intentamos guardar los cambios.



Robo3T nos manda un error, indicando que no tenemos autorización para dicha acción.

8. Desde el usuario administrador, cread la base de datos "testPrivilegios" con las colecciones "col1" y "col2".

Creamos las bases de datos con los documentos



Comprobamos que los documentos se han creado correctamente

9. Cread un rol que tenga permiso de lectura, inserción, actualización y borrado sobre la colección "col1" de la base de datos "testPrivilegios", y sólo permiso de lectura sobre la colección "col2" de la misma base de datos.

```
db.createRole({
  role: "col_role",
  privileges:[
    {resource: {db: "testPrivilegios", collection: "col1"},
      actions: ["insert","update", "remove"]},
    {resource: {db: "testPrivilegios", collection: "col2"},
      actions: ["find"]}
  ],
  roles:[
    {role: "read", db: "testPrivilegios"}
  ]
})
```

10. Desde el usuario "usuarioPais",acceded a la colección "col1" de la base de datos "testPrivilegios". Si todo lo anterior está bien, debe producirse un error al consultar la información. En ese caso, añadid al usuario "usuarioPais" el rol del ejercicio anterior y comprobad que puede hacer las operaciones correspondientes, es decir, consultar, actualizar, insertar y borrar sobre la colección "col1" y sólo consultar sobre la colección "col2"

- Accediendo a "col1" de "testPrivilegios" desde "userPais"

Iniciamos sesión con "userPais", en la base de datos "test". Cambiamos a la BBDD "testPrivilegios", e intentamos ejecutar un find() sobre "col1".

```
almu@debian:~$ mongo --authenticationDatabase "test" -u "userPais" -p
MongoDB shell version v4.4.2
Enter password:
connecting to: mongod://127.0.0.1:27017/?authSource=test&compressors=disabled&gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("6c7b9b44-d473-4d0e-967c-e8cb63bb4385") }
```

```

MongoDB server version: 4.4.1
> use testPrivilegios
switched to db testPrivilegios
> db.col1.find({})
Error: error: {
  "ok" : 0,
  "errmsg" : "not authorized on testPrivilegios to execute command { find: \"col1\", filter: {},
  "code" : 13,
  "codeName" : "Unauthorized"
}
>

```

Vemos como la shell de mongo nos devuelve un error, indicando que no estamos autorizados para ejecutar la consulta con `find()`

- **Asignando rol a UserPais**

Asignamos el rol creado en el paso anterior, a "userPais". Para ello, iniciamos sesión en la shell de mongo con el usuario "admin", y cambiamos a la base de datos "test" para acceder al usuario "UserPais"

Para ello, usamos el comando

```
db.grantRolesToUser("userPais", [{role: "col_role", db: "testPrivilegios"}])
```

Nos autenticamos como administrador en la shell de mongo, y lo ejecutamos.

```

almu@debian:~$ mongo --authenticationDatabase "admin" -u "super" -p
MongoDB shell version v4.4.2
Enter password:
connecting to: mongodb://127.0.0.1:27017/?authSource=admin&compressors=disabled&gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("f01ef93a-6ee9-4936-a6c2-859f04834a9a") }
MongoDB server version: 4.4.1
---
> use test
switched to db test
> db.grantRolesToUser("userPais", [{role: "col_role", db: "testPrivilegios"}])
>

```

- **Probando el acceso desde userPais**

Iniciamos sesión con userPais, e intentamos acceder a la colección "col1" de testPrivilegios

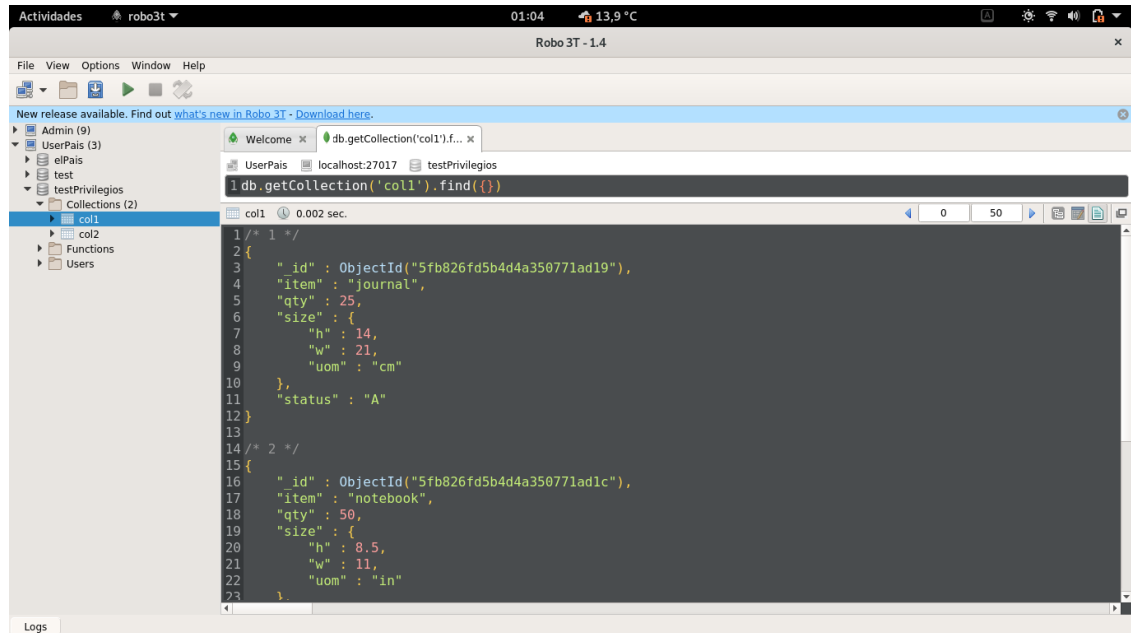
```

almu@debian:~$ mongo --authenticationDatabase "test" -u "userPais" -p
MongoDB shell version v4.4.2
Enter password:
connecting to: mongodb://127.0.0.1:27017/?authSource=test&compressors=disabled&gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("36e79a84-dfe5-4712-94bf-1fc5cf56b904") }
MongoDB server version: 4.4.1
> use testPrivilegios
switched to db testPrivilegios
> db.col1.find({})
{ "_id" : ObjectId("5fb826fd5b4d4a350771ad19"), "item" : "journal", "qty" : 25, "size" : { "h" : 10, "l" : 10, "w" : 10 }, "type" : "journal" }
{ "_id" : ObjectId("5fb826fd5b4d4a350771ad1c"), "item" : "notebook", "qty" : 50, "size" : { "h" : 10, "l" : 10, "w" : 10 }, "type" : "notebook" }
{ "_id" : ObjectId("5fb826fd5b4d4a350771ad1f"), "item" : "paper", "qty" : 100, "size" : { "h" : 10, "l" : 10, "w" : 10 }, "type" : "paper" }
>

```

Vemos que, esta vez, podemos acceder sin problemas.

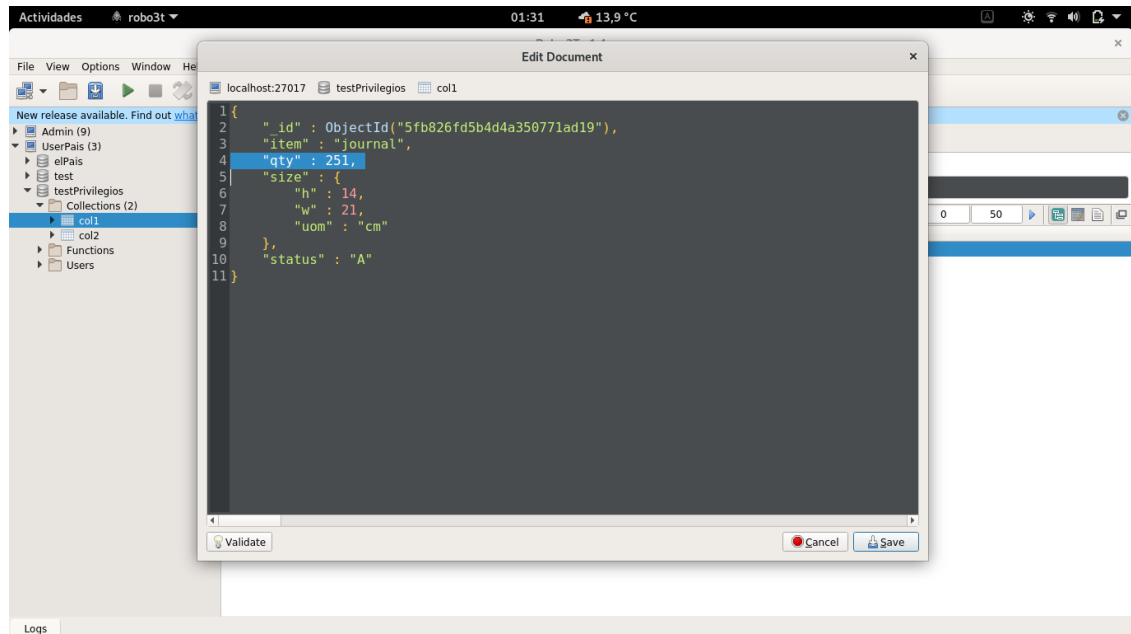
Entramos en Robo3T, y vemos que la lista de bases de datos "userPais" ya incluye a "testPrivilegios" con "col1" y "col2"



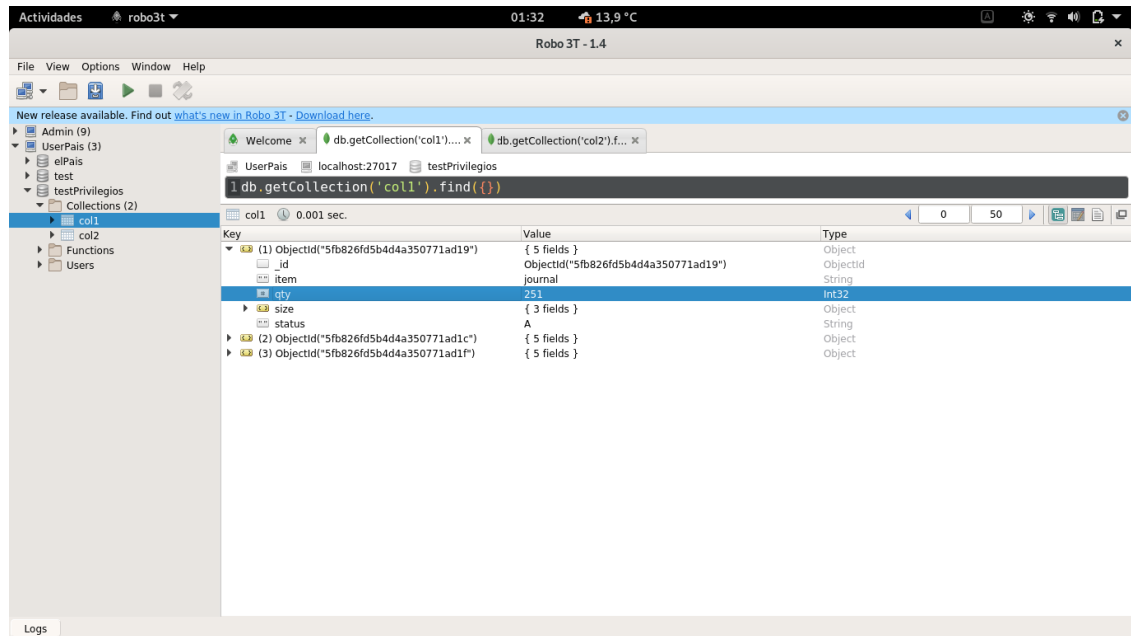
También vemos que el usuario tiene acceso a las colecciones, pudiendo hacer consultas sobre ellos.

- **Intentando editar documentos en "col1"**

Desde Robo3T, abrimos uno de los documentos de "col1" y editamos una de sus líneas



Guardamos los cambios, y comprobamos que se han aplicado correctamente

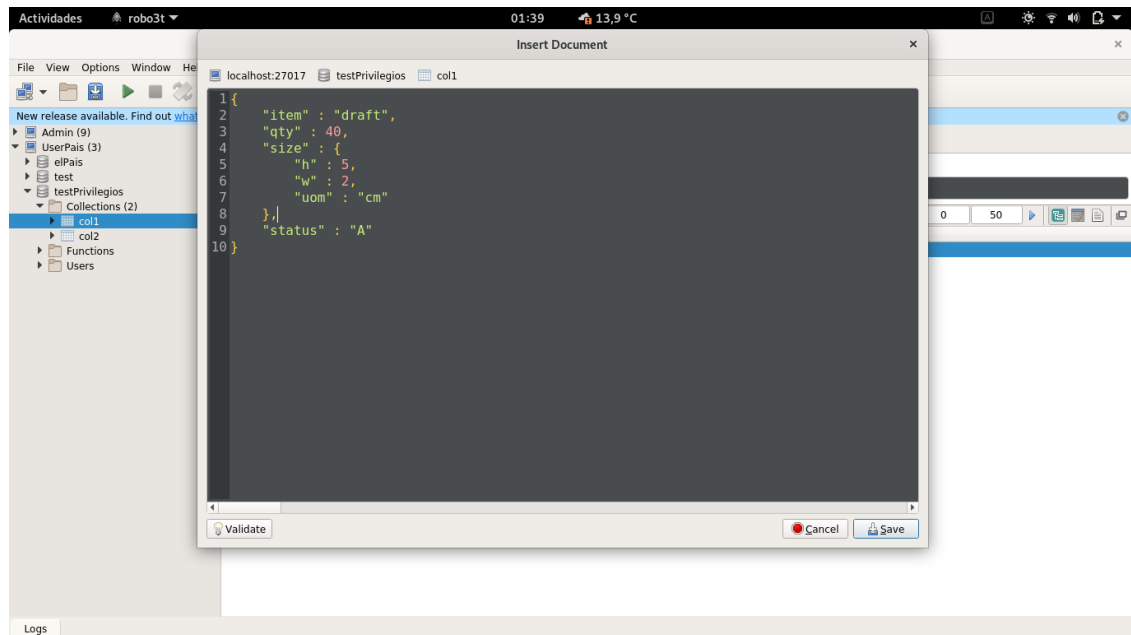


Robo3T no devuelve ningún error, y vemos que el valor se ha modificado correctamente

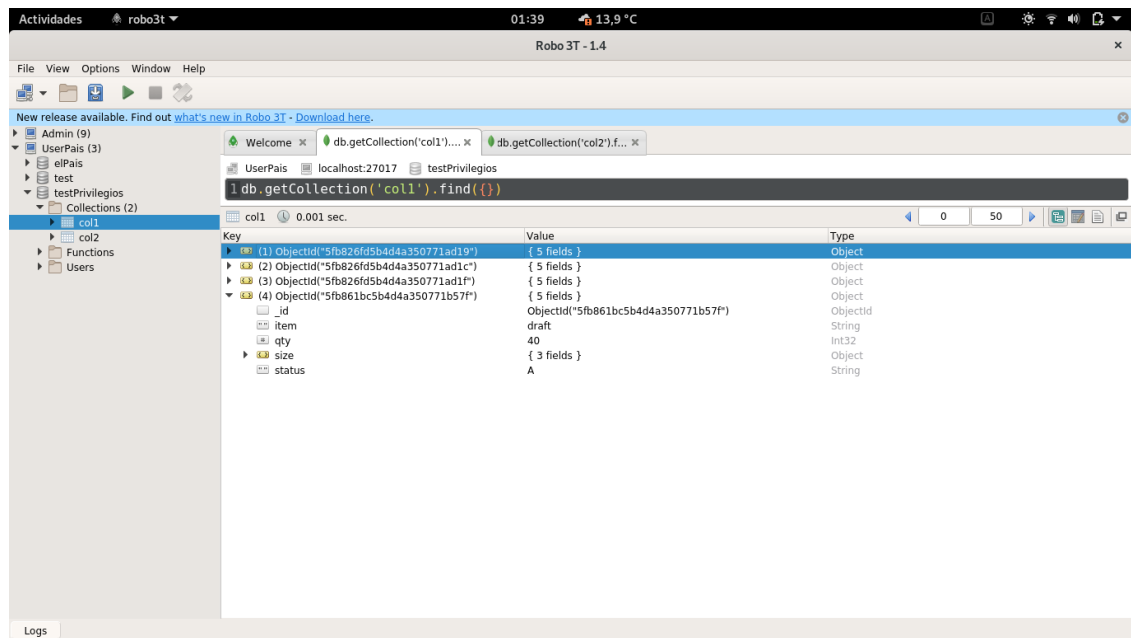
- **Intentando insertar documentos en "col1"**

Hacemos otra prueba, intentando añadir un nuevo documento a la colección

Creamos el documento, y guardamos los cambios

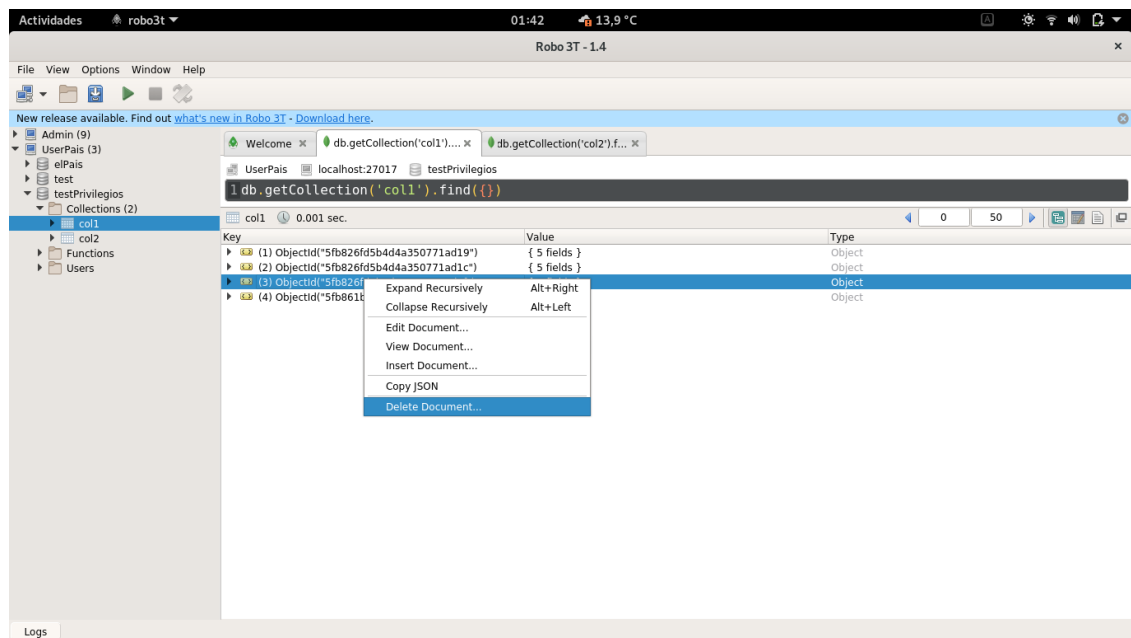


Comprobamos que el documento se ha añadido correctamente a la colección, y Robo3T no muestra ningún error.

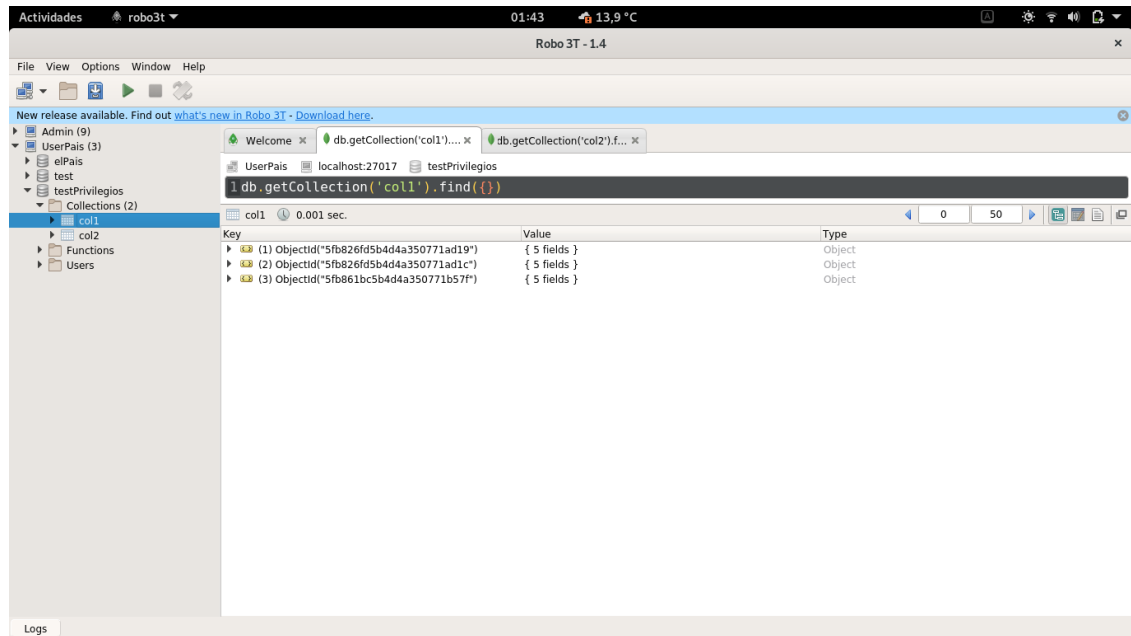


- **Borrando un documento en "col1"**

Desde Robo3T, intentamos eliminar uno de los documentos de la colección "col1"

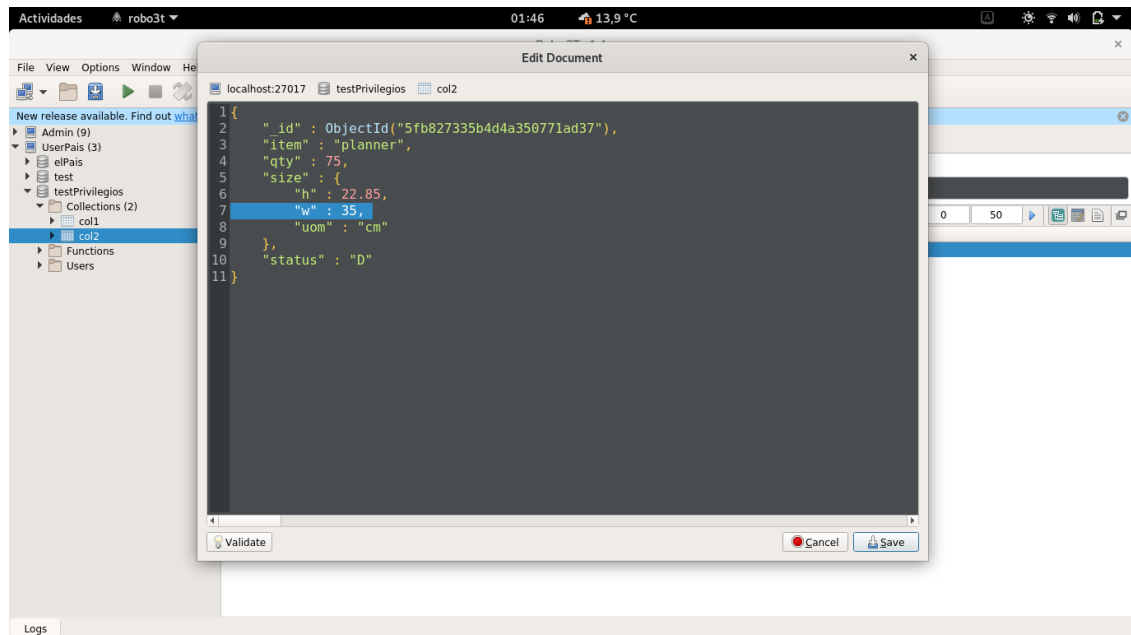


Comprobamos que el documento se ha borrado correctamente

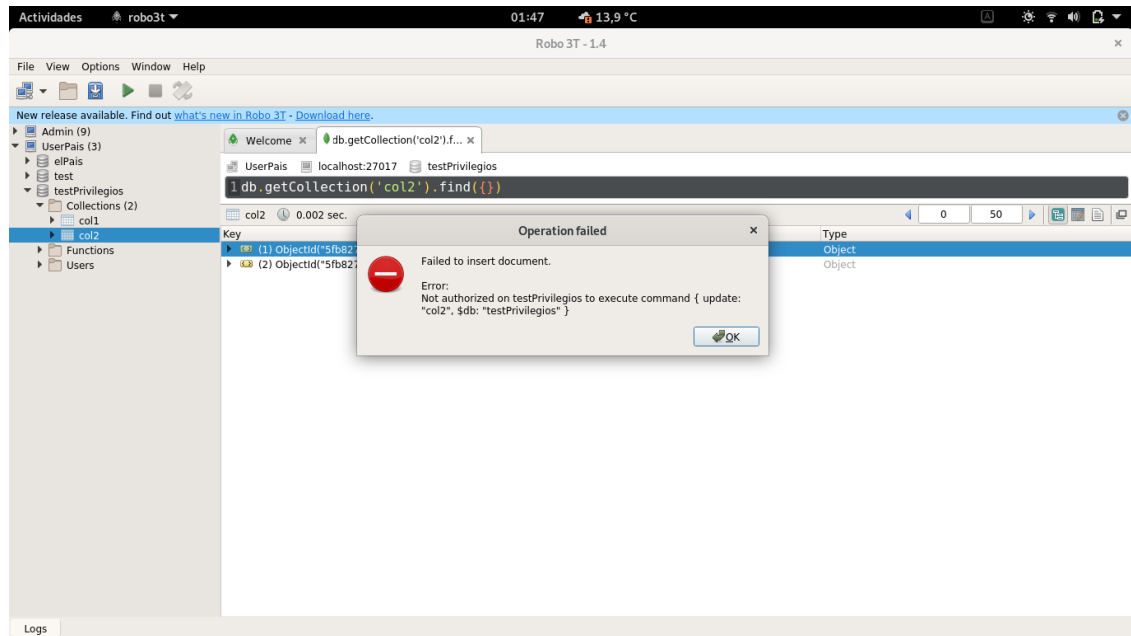


- **Intentando editar un documento en "col2"**

Comprobados los permisos sobre "col1", repetimos la prueba con "col2", intentando editar un documento.



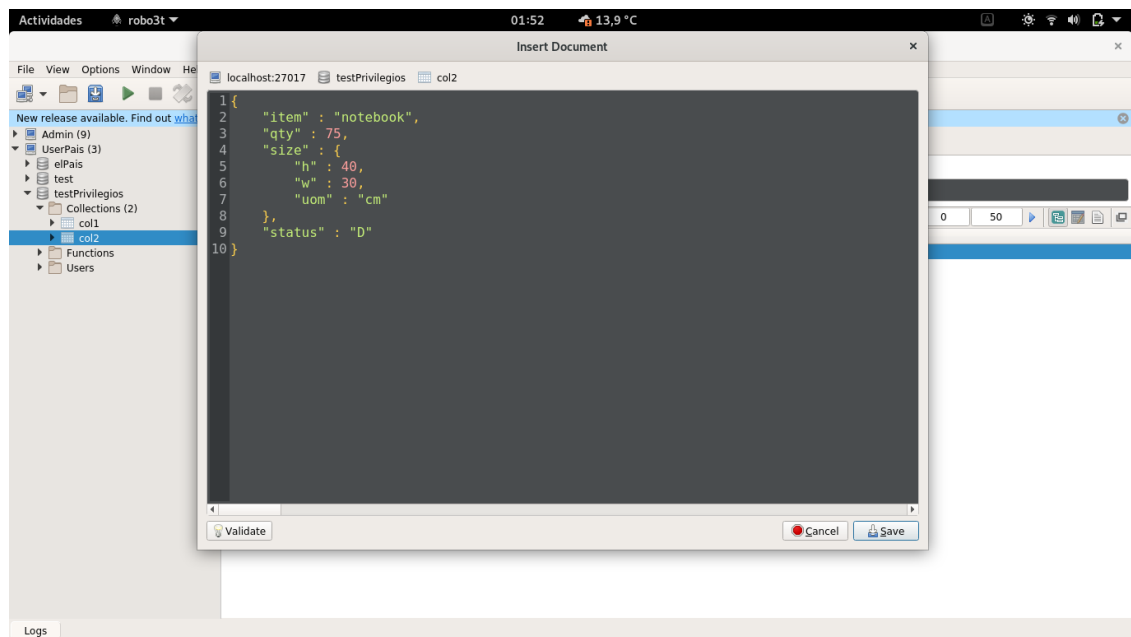
Modificamos el documento e intentamos guardar los cambios.



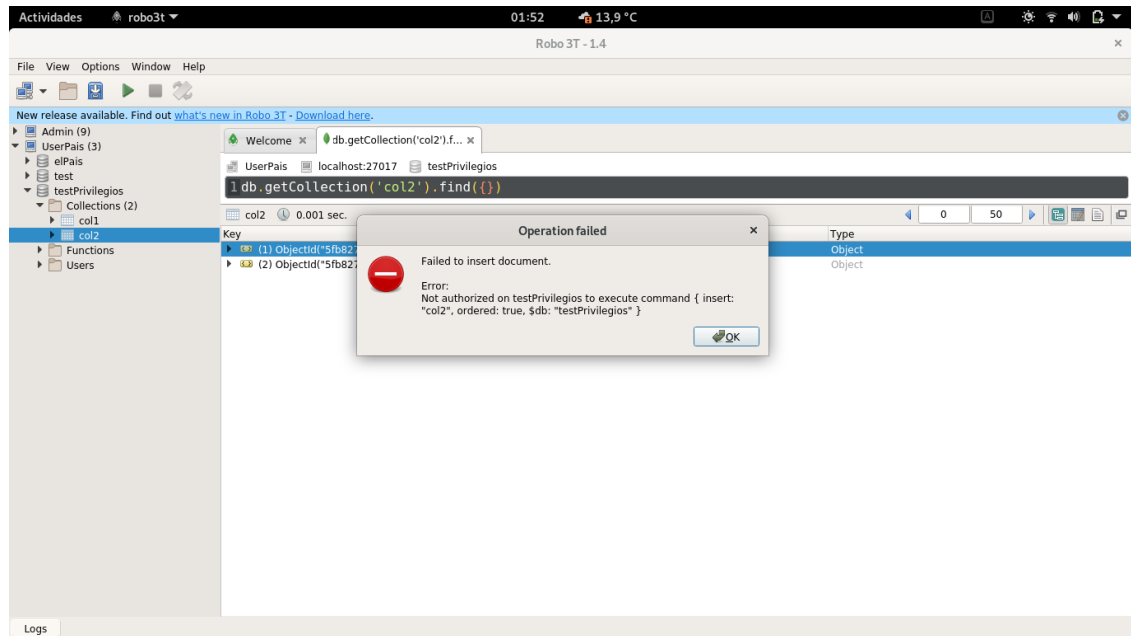
Vemos como Robo3T nos muestra un error, indicando que no tenemos permisos para realizar dicha operación.

- **Intentando añadir un documento en "col2"**

Ahora intentamos crear un nuevo documento y añadirlo a "col2"



Rellenamos el documento e intentamos guardar los cambios.

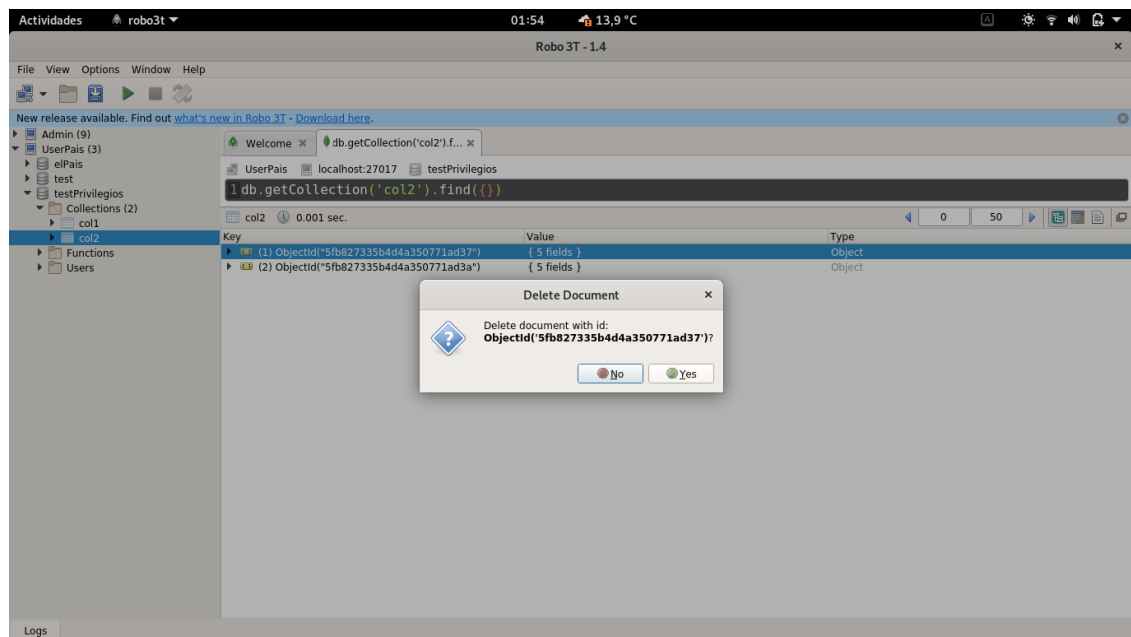


Robo3T nos muestra otro error, indicando que no tenemos permisos para la operación.

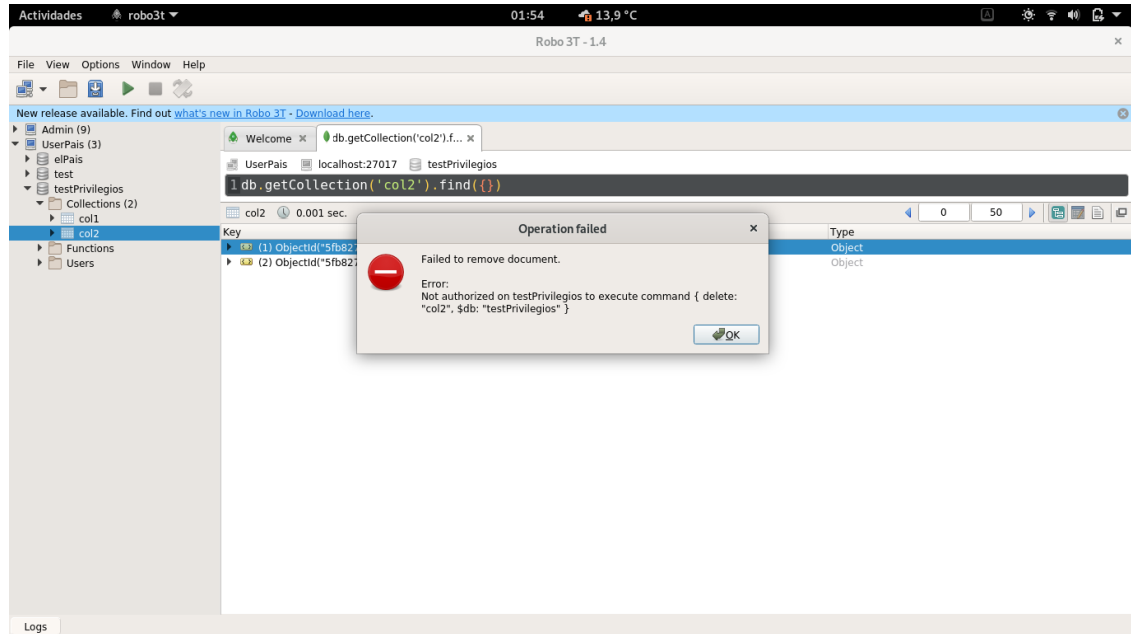
- **Intentando borrar un documento en "col2"**

Para finalizar, comprobamos si podemos borrar un documento en "col2".

Desde Robo3T, intentamos borrar un documento



Seleccionamos un documento para eliminarlo, y confirmamos.



Vemos como Robo3T nos muestra otro nuevo error, indicando que no tenemos permisos para dicha operación.

11. Con el usuario "super", realizad una copia de seguridad de la base de datos "testPrivilegios"

Dado que el acceso a la base de datos requiere privilegios, añadimos la opción `--authenticationDatabase` para realizar la autenticación en la base de datos "admin".

El comando a ejecutar es:

```
mongodump -u "super" -p "superpwd" \
  --authenticationDatabase admin \
  -d testPrivilegios -o dump_testprivilegios
```

Comprobamos que se ha ejecutado correctamente

```
almu@debian:~$ mongodump -u "super" -p "superpwd" --authenticationDatabase admin -d testPrivilegios
2020-11-21T15:39:50.672+0100   writing testPrivilegios.col2 to dump_testprivilegios/testPrivilegios.col2
2020-11-21T15:39:50.674+0100   writing testPrivilegios.col1 to dump_testprivilegios/testPrivilegios.col1
2020-11-21T15:39:50.674+0100   done dumping testPrivilegios.col2 (2 documents)
2020-11-21T15:39:50.675+0100   done dumping testPrivilegios.col1 (3 documents)
```

12. Con el mismo usuario, restaurad sólo la colección "col2" en una nueva colección llamada "col999"

Añadimos la opción `--drop` para borrar el contenido anterior. El comando a ejecutar es:

```
mongorestore --drop --nsInclude "testPrivilegios.col2" \
  dump_testprivilegios -u "super" -p "superpwd" \
  --authenticationDatabase admin
```

Comprobamos que se ha ejecutado correctamente, recuperando 2 documentos de "col2".

```
almu@debian:~$ mongorestore --drop --nsInclude "testPrivilegios.col2" dump_testprivilegios -u "super"
2020-11-21T15:49:12.351+0100   preparing collections to restore from
2020-11-21T15:49:12.357+0100   reading metadata for testPrivilegios.col2 from dump_testprivilegios
2020-11-21T15:49:12.368+0100   restoring testPrivilegios.col2 from dump_testprivilegios/testPrivilegios.col2
2020-11-21T15:49:12.379+0100   no indexes to restore
```

```
2020-11-21T15:49:12.379+0100 finished restoring testPrivilegios.col2 (2 documents, 0 failures)
2020-11-21T15:49:12.379+0100 2 document(s) restored successfully. 0 document(s) failed to restore
```