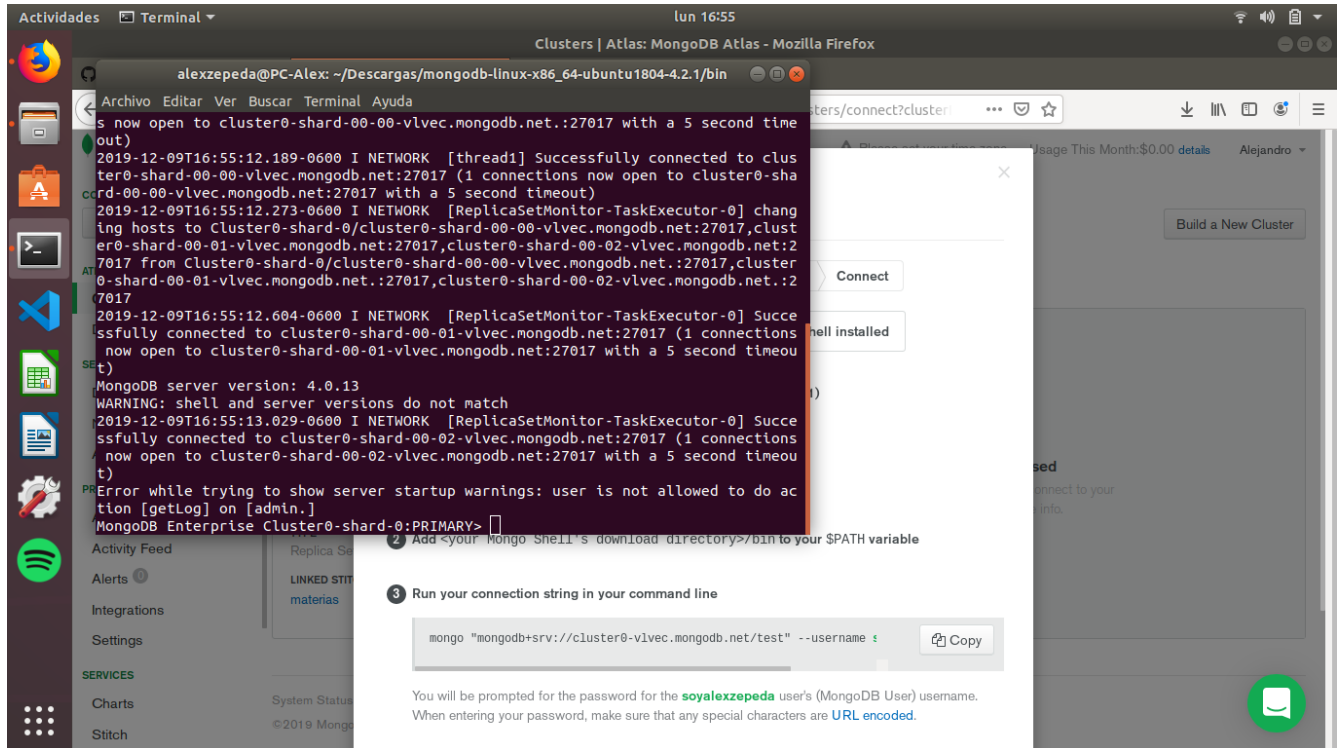
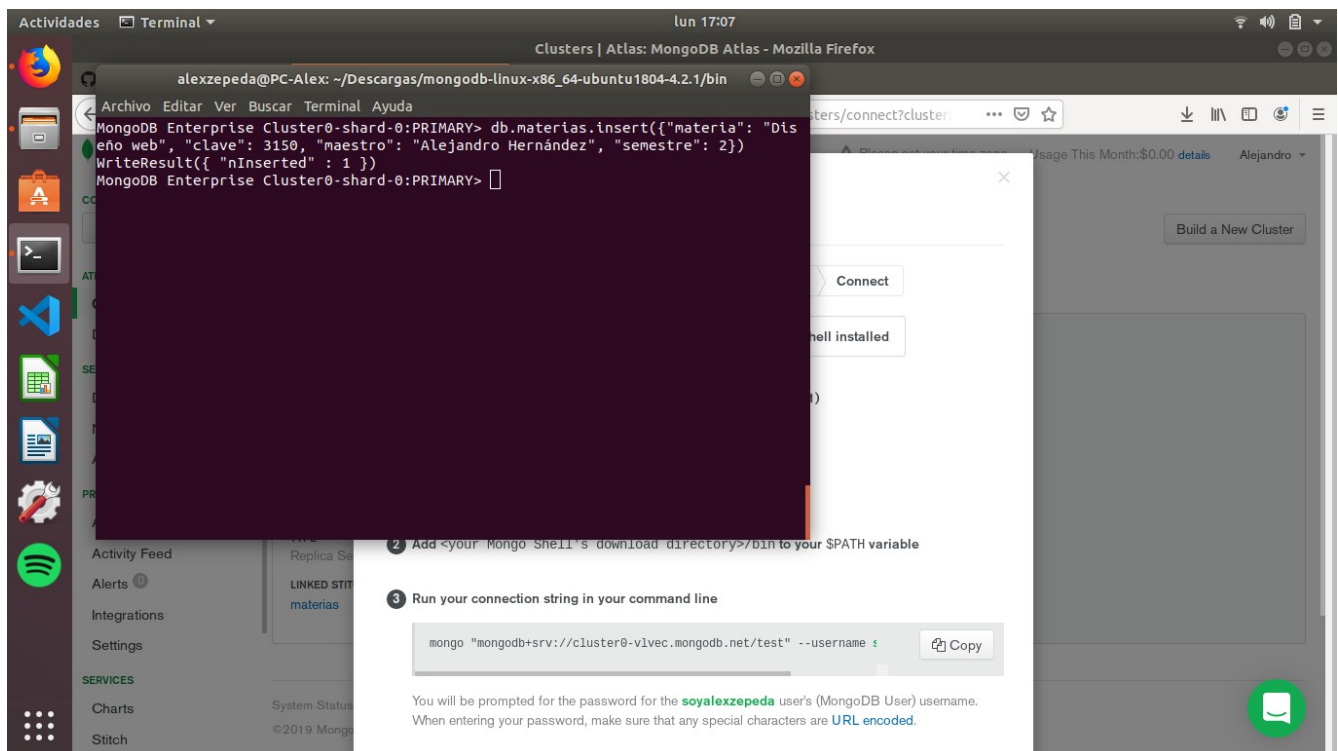


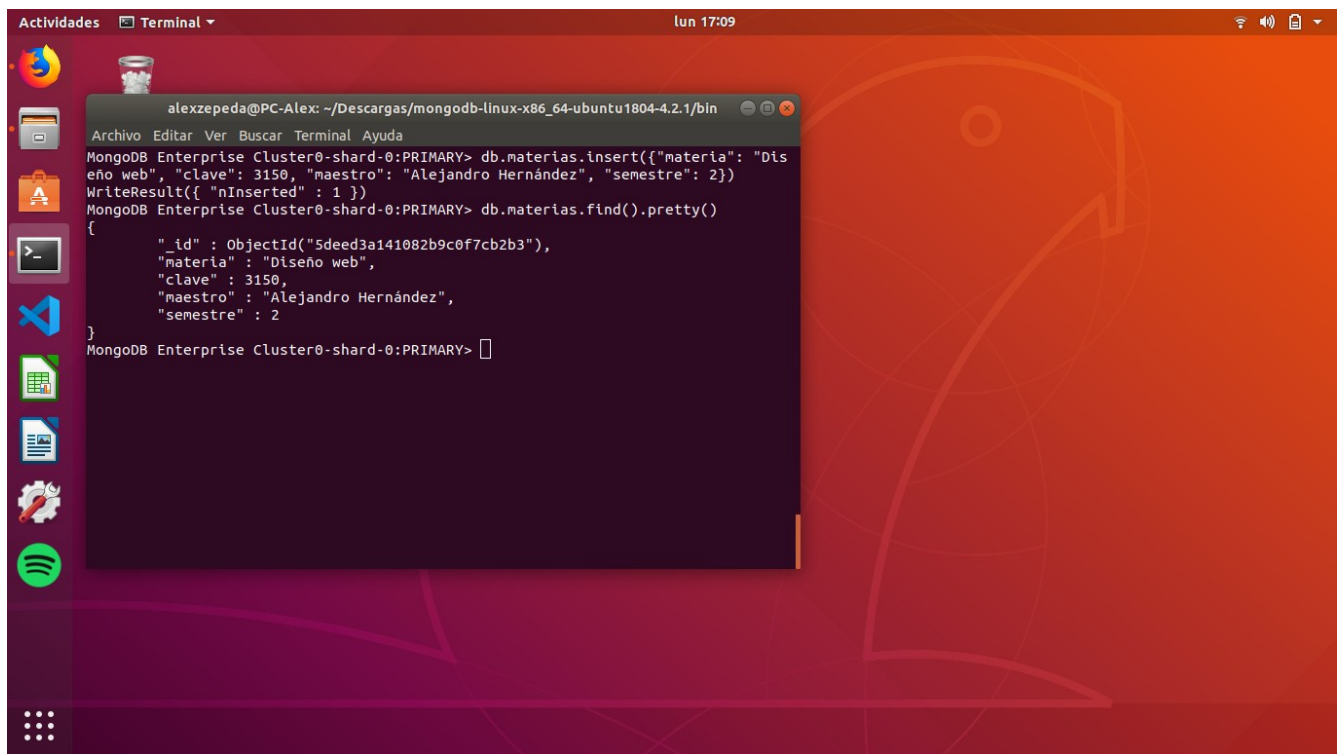
Para la creación de mi aplicativo use el sistema operativo Ubuntu, distribución de Linux.



Para conectarme al cluster de mongodb, fue necesario descargar un shell o instalar mongodb donde tenia que entrar a la ruta para ejecutar mi cadena de conexión.

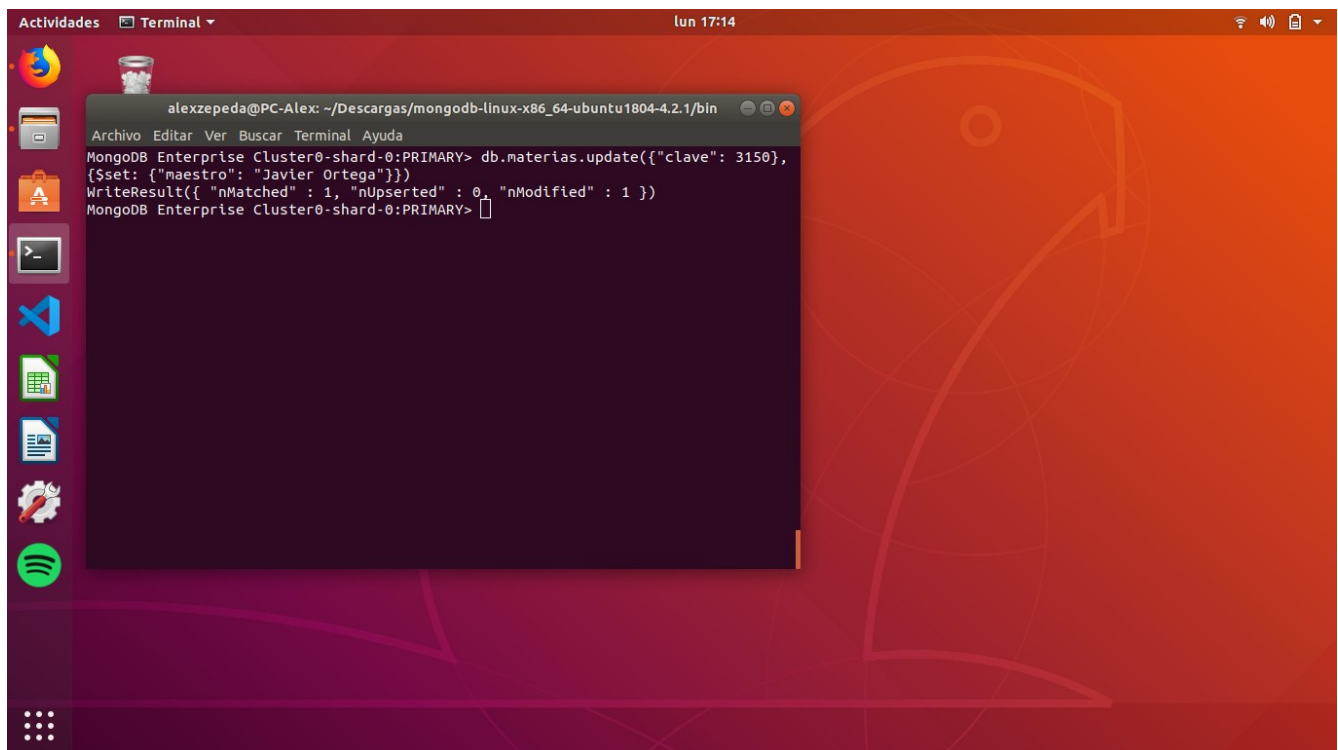


Para ingresar un documento a mi colección fué necesario utilizar la instrucción db.materias.insert.

A terminal window on a Linux desktop with an orange background. The terminal shows the execution of MongoDB commands. First, an insert command is run, and then a find command is used to retrieve the inserted document. The output of the find command is displayed in a pretty-printed JSON format.

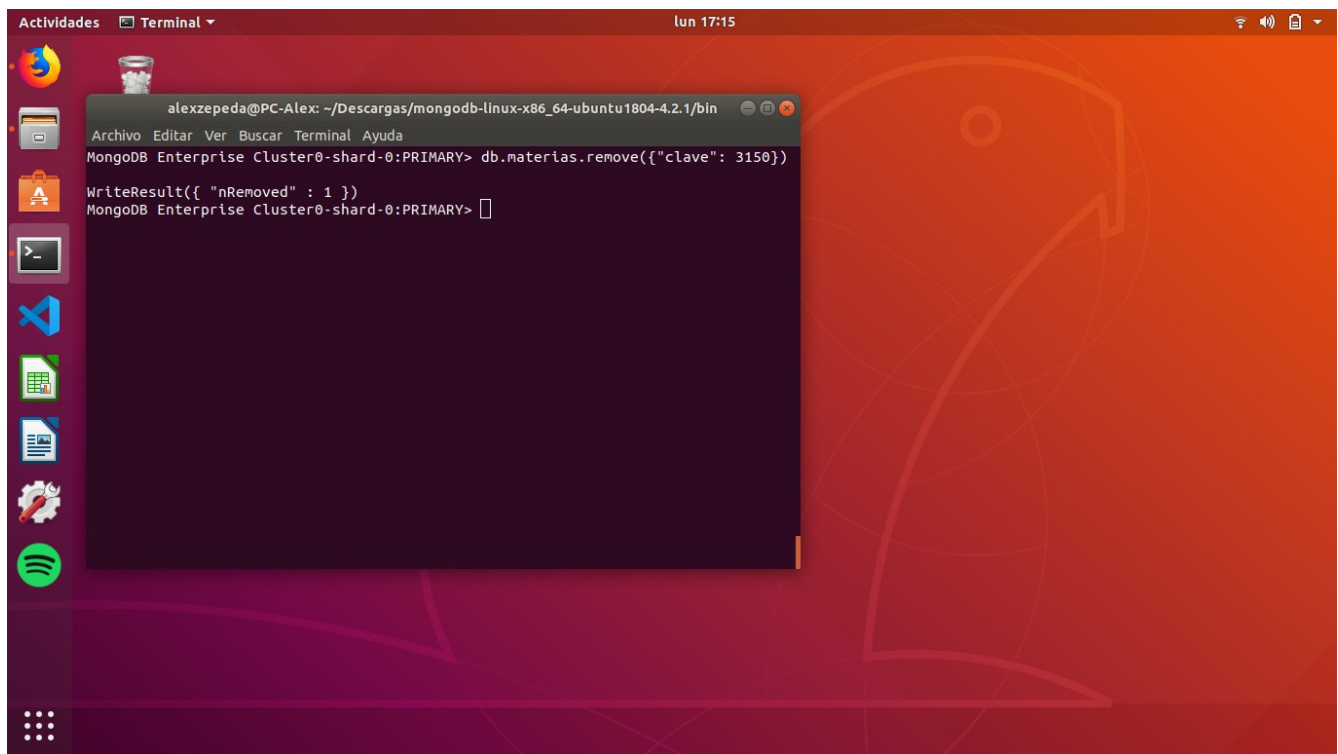
```
alexzepeda@PC-Alex: ~/Descargas/mongodb-linux-x86_64-ubuntu1804-4.2.1/bin
MongoDB Enterprise Cluster0-shard-0:PRIMARY> db.materias.insert({"materia": "Diseño web", "clave": 3150, "maestro": "Alejandro Hernández", "semestre": 2})
WriteResult({"nInserted" : 1 })
MongoDB Enterprise Cluster0-shard-0:PRIMARY> db.materias.find().pretty()
{
  "_id" : ObjectId("5deed3a141082b9c0f7cb2b3"),
  "materia" : "Diseño web",
  "clave" : 3150,
  "maestro" : "Alejandro Hernández",
  "semestre" : 2
}
MongoDB Enterprise Cluster0-shard-0:PRIMARY>
```

Aquí se muestra que mi documento se ingresó correctamente a la colección.

A terminal window on a Linux desktop with an orange background. The terminal shows the execution of a MongoDB update command. The command updates the 'maestro' field of a document where the 'clave' is 3150. The output shows that one document was matched and updated.

```
alexzepeda@PC-Alex: ~/Descargas/mongodb-linux-x86_64-ubuntu1804-4.2.1/bin
MongoDB Enterprise Cluster0-shard-0:PRIMARY> db.materias.update({"clave": 3150},
{$set: {"maestro": "Javier Ortega"}})
WriteResult({"nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
MongoDB Enterprise Cluster0-shard-0:PRIMARY>
```

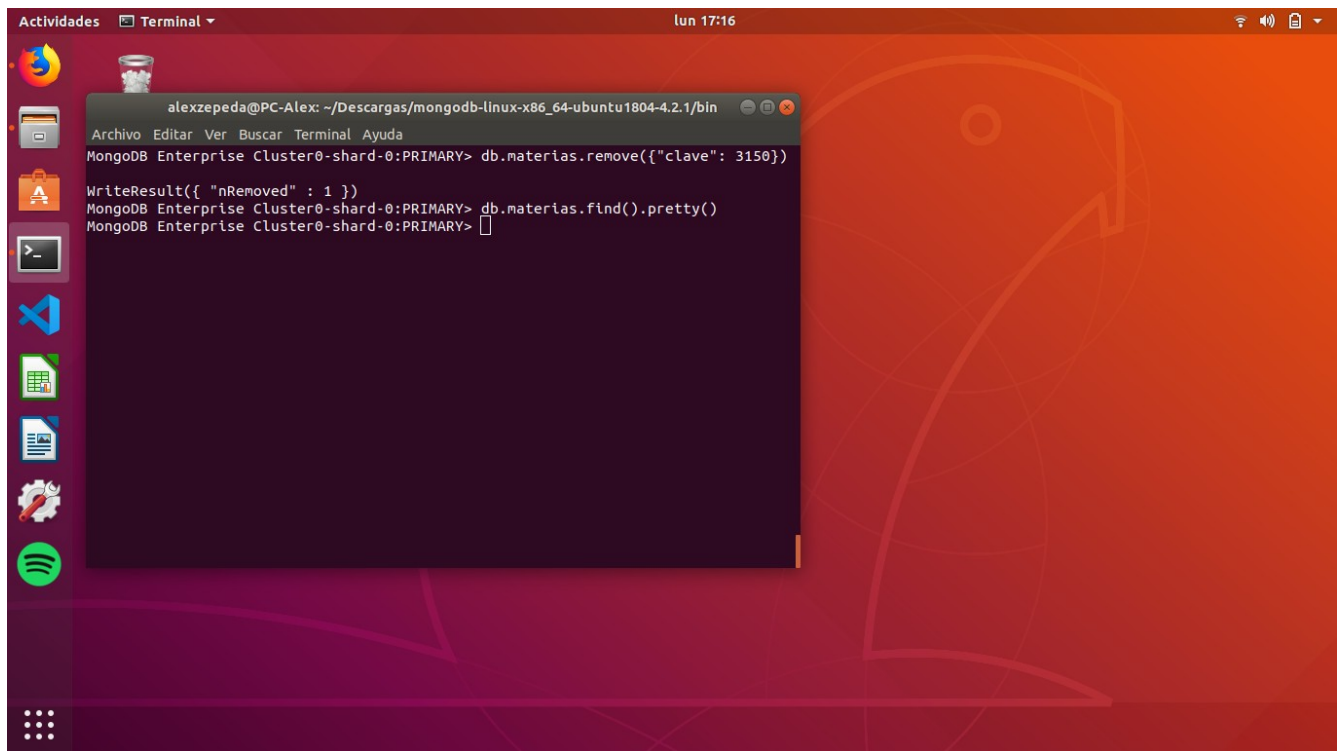
Para actualizar un dato de nuestro documento es necesario conocer un parametro de nuestro objeto para actualizarlo.

A screenshot of a Linux terminal window titled 'Terminal' with the date 'lun 17:15'. The terminal shows a MongoDB shell session. The user has entered the command 'db.materias.remove({"clave": 3150})'. The output is 'WriteResult({"nRemoved" : 1 })'. The terminal window is titled 'alexzepeda@PC-Alex: ~/Descargas/mongodb-linux-x86\_64-ubuntu1804-4.2.1/bin'.

```
alexzepeda@PC-Alex: ~/Descargas/mongodb-linux-x86_64-ubuntu1804-4.2.1/bin
MongoDB Enterprise Cluster0-shard-0:PRIMARY> db.materias.remove({"clave": 3150})

WriteResult({"nRemoved" : 1 })
MongoDB Enterprise Cluster0-shard-0:PRIMARY> 
```

Para eliminar de la misma manera, se utiliza un parametro requerido, en este caso utilice clave como parametro del objeto.

A screenshot of a Linux terminal window titled 'Terminal' with the date 'lun 17:16'. The terminal shows a MongoDB shell session. The user has entered the command 'db.materias.remove({"clave": 3150})'. The output is 'WriteResult({"nRemoved" : 1 })'. The user has then entered the command 'db.materias.find().pretty()'. The terminal window is titled 'alexzepeda@PC-Alex: ~/Descargas/mongodb-linux-x86\_64-ubuntu1804-4.2.1/bin'.

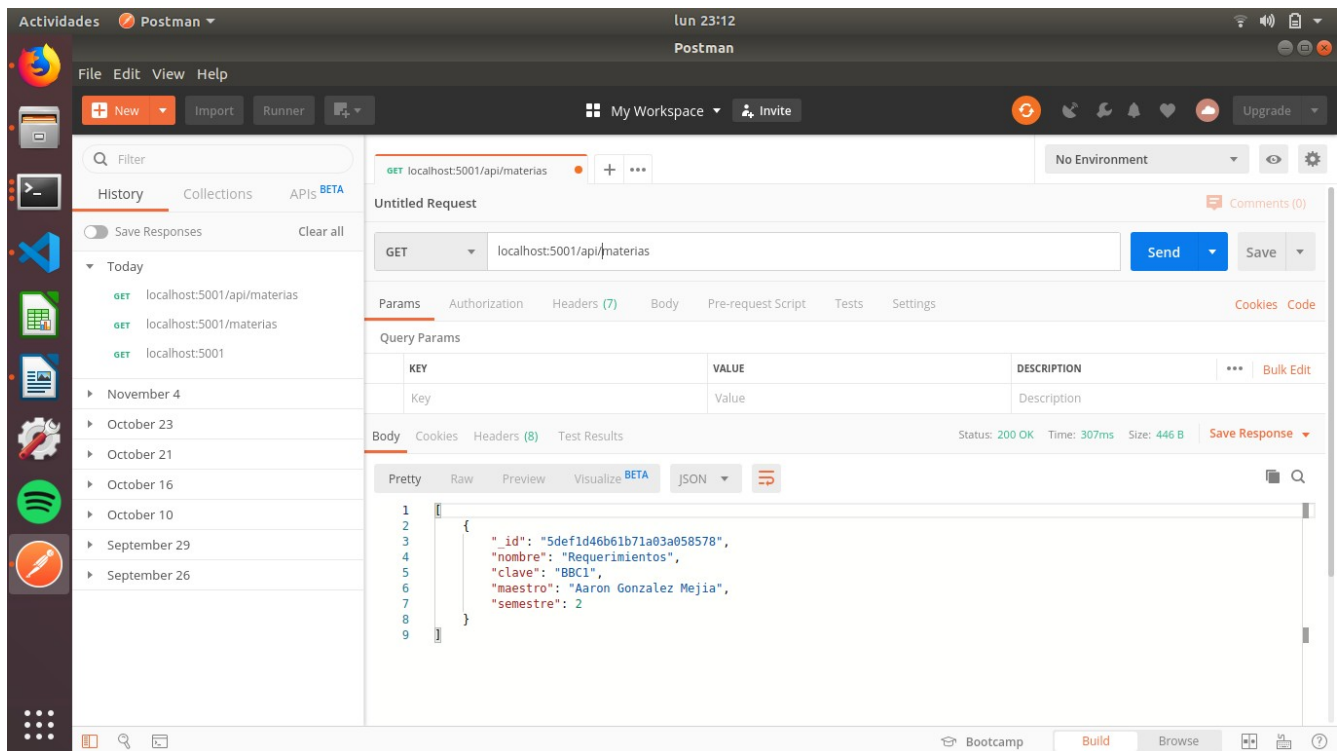
```
alexzepeda@PC-Alex: ~/Descargas/mongodb-linux-x86_64-ubuntu1804-4.2.1/bin
MongoDB Enterprise Cluster0-shard-0:PRIMARY> db.materias.remove({"clave": 3150})

WriteResult({"nRemoved" : 1 })
MongoDB Enterprise Cluster0-shard-0:PRIMARY> db.materias.find().pretty()
MongoDB Enterprise Cluster0-shard-0:PRIMARY> 
```

Uso la instrucción `db.materias.find().pretty()` para consultar de manera ordenada mis documentos, en este caso no hay ninguno porque se uso la instrucción delete.

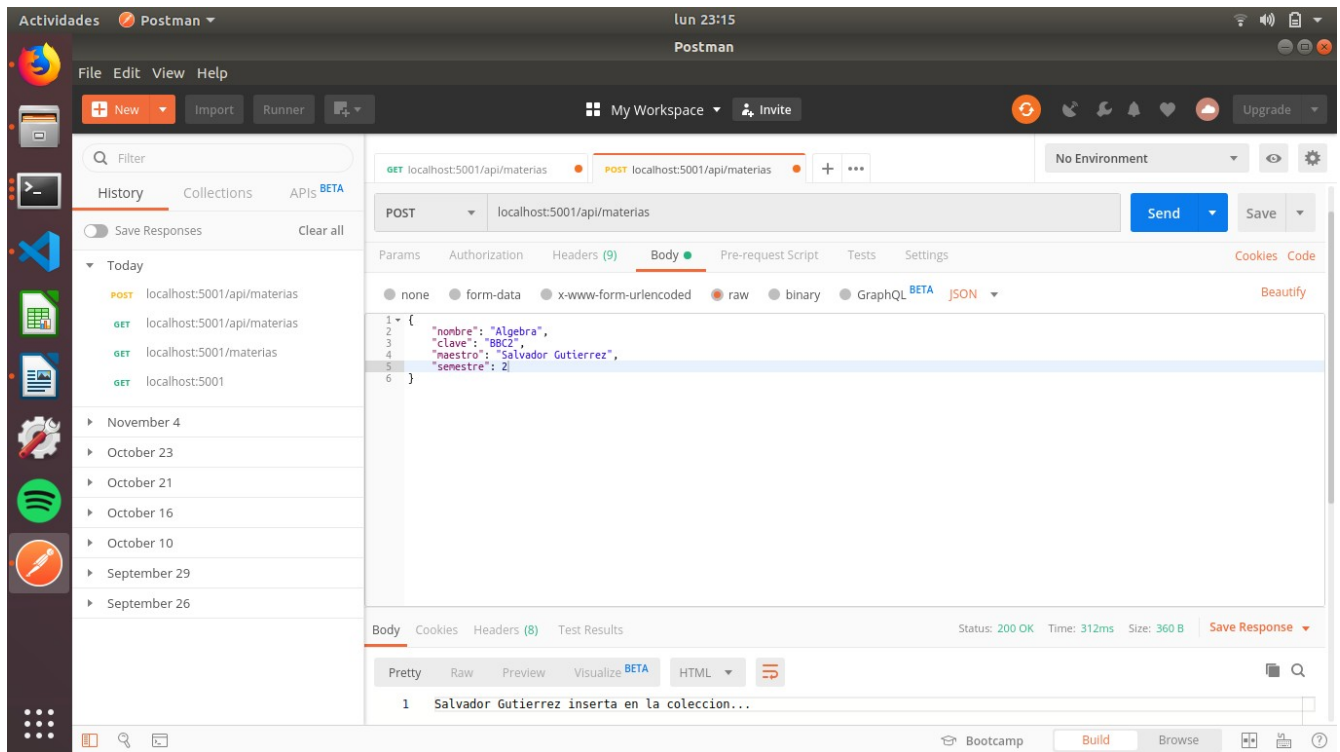
```
aplicacion > JS app.js > ...
1 //-----
2 //1.-Requerir librerias y drivers
3 const express = require('express');
4 const bodyParser = require('body-parser');
5 const mongoose = require('mongoose');
6 const MongoDBUrl = require('./keys');
7 const MateriasController = require(
8 //-----
9 //2.-Configurar web server y parseo
10 const app = express();
11 const port = 5001;
12 //instruccion para heroku, la asigna
13 //var port =process.env.PORT || 5001;
14 app.use(bodyParser.json());
15 app.use(function(req, res, next) {
16   res.header("Access-Control-Allow-Origin: *");
17   res.header("Access-Control-Allow-Headers: *");
18   next();
19 });
20
21 //-----
22 //3.- Definir paths disponibles
23 app.get('/', (req, res) => { res.send('Bienvenidos'); });
24
25 app.get('/api/materias/', MateriasController.get);
26 app.post('/api/materias/', MateriasController.create);
27
28
29
30 //-----
31 //4.- Encender webserver y dbserver
32 app.listen(port, ()=>{
  console.log(`Server Inicializado en el puerto: 5001`);
  console.log(`Server mongodb Conectado...`);
});
```

Para inicializar mi servidor fue necesario crear mi cadena de conexión a partir de localhost ubicado en el puerto 5001.

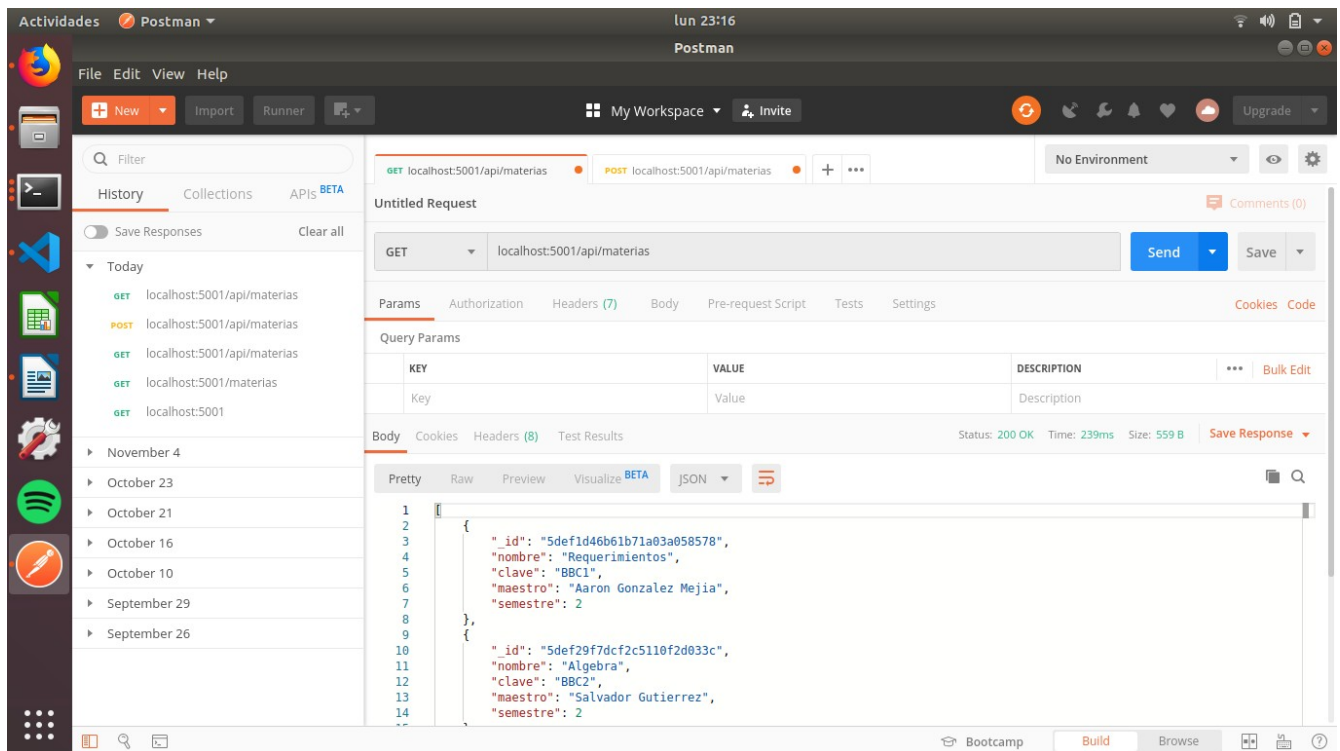


Hacemos en postman un GET para consultar nuestros documentos. Comprobando que nuestro servidor esta activo.





Hacemos un POST, ingresando un nuevo documento a nuestra base de datos, podemos ver que nuestro objeto maestro fue ingresado correctamente.



Nuevamente realizamos un GET y comprobamos que nuestro objeto ya esta ingresado en nuestra base de datos.

1. Códigos de error REST

200 Código de éxito  
300 Código de redirecciones  
400 Código de error por parte del cliente  
500 Código de error por parte del servidor

2. Métodos o verbos REST

GET: Consulta  
POST: Insertar  
PUT: Actualizar  
DELETE: Eliminar

3. Tipos de parámetros y como viajan

Path Params: Parametros Requeridos que estan en {} y viajan por al URI  
Query Params: Parametros opcionales despues del ? Y viajan por la URI  
Body Params: Parametros que viajan por el body

4. Partes de una solicitud REST

HEADER, URL única, params

5. Partes de una respuesta REST

HEADER, body, status

6. Métodos que siempre llevan parámetros requeridos

PUT y DELETE

7. Crear cluster cloud mongodb

Accedemos a mongodb y nos registramos en la pagina oficial, iniciamos sesión, configuramos el cluster con nuestra ip

8. Conectarse por shell a BD cloud

Descargamos el shell, nos ubicamos en la carpeta de mongo y copiamos la cadena de conexión que nos da mongodb

9. Cargar BD muestra / Importar BD

Mongodb nos da la opción de importar nuestras bases de datos a partir de un documento en json, configurando nuestro usuario, contraseña, nombre del documento y extensión

10. Crear consultas simples

```
db.materias.find() //Filtrar consultas
db.materias.sort() //Ordenar consultas
db.materias.insert() //Ingresar documentos
db.materias.update() //Actualizar documentos
db.materias.remove() //Eliminar documentos
db.materias.count() //Contar documentos existentes
db.materias.drop() //Eliminar una colección
```

11. Insertar y borrar documentos en mongodb

```
db.materias.insert() //Ingresar documentos
db.materias.remove() //Eliminar documentos
```
12. Convertir consultas en servicios expuestos
13. Instalar CLI de docker
14. listar contenedores activos e inactivos

```
docker ps -a
```
15. Listar imágenes

```
docker images
```
16. Descargar imágenes

```
docker pull "name image" (sin comillas)
```
17. Correr contenedor basado en una imagen

```
docker-compose up
```
18. Acceder al shell de un servidor (contenedor)
19. Encender, detener y borrar contenedor

```
docker stop "contenedor" (sin comillas)
docker start "contenedor" (sin comillas)
docker rm "contenedor" (sin comillas)
```
20. Borrar contenedor

```
docker rm "contenedor" (sin comillas)
```
21. Configurar puerto de un contenedor
22. Monitorear eventos de un contenedor

```
docker logs "contenedor" (sin comillas)
```
23. Construir imagen de un contenedor y publicarla

```
docker build -t api .
docker-compose up
```
24. Crear yaml de armado y despliegue de contenedor
25. Orquestar contenedores
26. Desplegar aplicación en un contenedor