

Tema 4: MongoDB

1. Descarga e instalación

En este trabajo vamos a descargar, instalar y probar el SGBD NoSQL MongoDB en Ubuntu 20.04 LTS. Para ello, hemos accedido a la página

<https://docs.mongodb.com/manual/tutorial/install-mongodb-on-ubuntu/>

Y hemos seguido los pasos que se indican para la instalación, como podemos ver en las siguientes imágenes.

Primero, comprobamos que tenemos gnupg instalado y sus bibliotecas requeridas.

```
ubuntu@ubuntu-VirtualBox:~$ sudo apt-get install gnupg
Reading package lists... Done
Building dependency tree
Reading state information... Done
gnupg is already the newest version (2.2.19-3ubuntu2).
gnupg set to manually installed.
The following packages were automatically installed and are no longer required:
  libfprint-2-tod1 linux-headers-5.4.0-42 linux-headers-5.4.0-42-generic
  linux-image-5.4.0-42-generic linux-modules-5.4.0-42-generic
  linux-modules-extra-5.4.0-42-generic
Use 'sudo apt autoremove' to remove them.
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
```

Importamos la clave pública utilizada por el sistema de gestión de paquetes.

```
ubuntu@ubuntu-VirtualBox:~$ wget -qO - https://www.mongodb.org/static/pgp/server-4.4.asc | sudo apt-key add -
OK
```

Creamos un archivo necesario para MongoDB.

```
ubuntu@ubuntu-VirtualBox:~$ echo "deb [ arch=amd64,arm64 ] https://repo.mongodb.org/apt/ubuntu focal/mongodb-org/4.4 multiverse" | sudo tee /etc/apt/sources.list.d/mongodb-org-4.4.list
deb [ arch=amd64,arm64 ] https://repo.mongodb.org/apt/ubuntu focal/mongodb-org/4.4 multiverse
```

Instalamos los paquetes de MongoDB.

```
ubuntu@ubuntu-VirtualBox:~$ sudo apt-get install -y mongodb-org
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages were automatically installed and are no longer required:
  libfprint-2-tod1 linux-headers-5.4.0-42 linux-headers-5.4.0-42-generic
  linux-image-5.4.0-42-generic linux-modules-5.4.0-42-generic
  linux-modules-extra-5.4.0-42-generic
Use 'sudo apt autoremove' to remove them.
The following additional packages will be installed:
  libcurl4 mongodb-database-tools mongodb-org-database-tools-extra mongodb-org-mongos
  mongodb-org-server mongodb-org-shell mongodb-org-tools
The following NEW packages will be installed:
  libcurl4 mongodb-database-tools mongodb-org mongodb-org-database-tools-extra
  mongodb-org-mongos mongodb-org-server mongodb-org-shell mongodb-org-tools
0 upgraded, 8 newly installed, 0 to remove and 117 not upgraded.
Need to get 104 MB of archives.
After this operation, 201 MB of additional disk space will be used.
```

Fijamos la versión instalada de los paquetes.

```
ubuntu@ubuntu-VirtualBox:~$ echo "mongodb-org hold" | sudo dpkg --set-selections
ubuntu@ubuntu-VirtualBox:~$ echo "mongodb-org-server hold" | sudo dpkg --set-selections
ubuntu@ubuntu-VirtualBox:~$ echo "mongodb-org-shell hold" | sudo dpkg --set-selections
ubuntu@ubuntu-VirtualBox:~$ echo "mongodb-org-mongos hold" | sudo dpkg --set-selections
ubuntu@ubuntu-VirtualBox:~$ echo "mongodb-org-tools hold" | sudo dpkg --set-selections
```

Vemos el sistema que usa nuestra plataforma.

```
ubuntu@ubuntu-VirtualBox:~$ ps --no-headers -o comm 1
systemd
```

Iniciamos el proceso de MongoDB y verificamos que se ha activado correctamente.

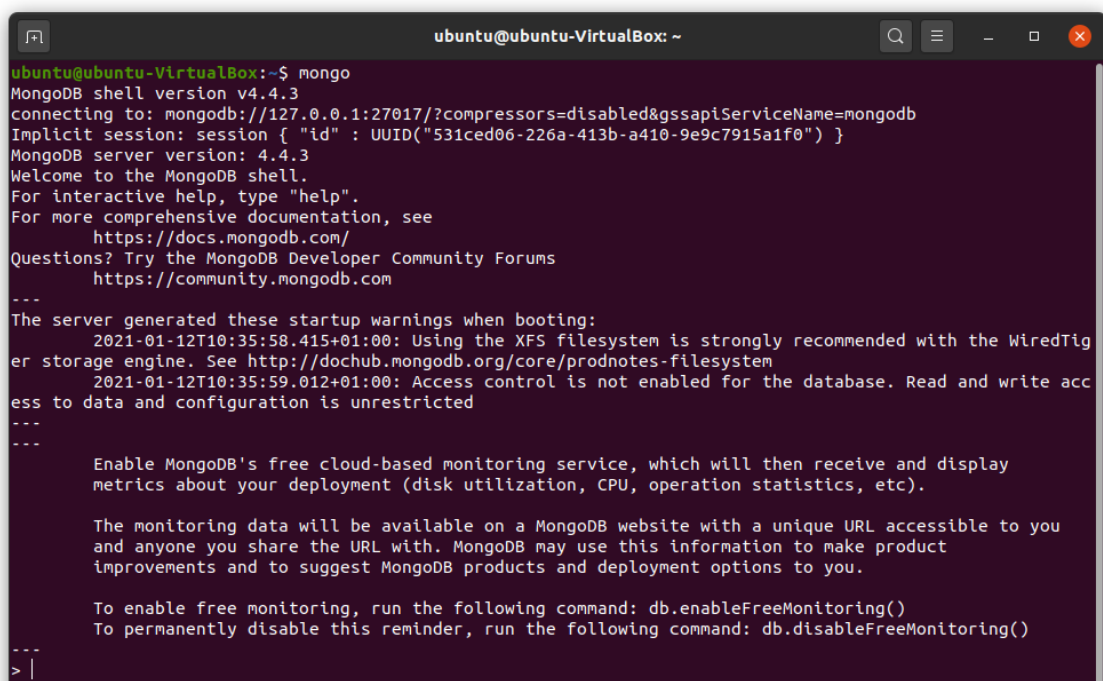
```
ubuntu@ubuntu-VirtualBox:~$ ps --no-headers -o comm 1
systemd
ubuntu@ubuntu-VirtualBox:~$ sudo systemctl start mongod
ubuntu@ubuntu-VirtualBox:~$ sudo systemctl status mongod
● mongod.service - MongoDB Database Server
   Loaded: loaded (/lib/systemd/system/mongod.service; disabled; vendor preset: enable)
   Active: active (running) since Tue 2021-01-12 10:35:58 CET; 40s ago
     Docs: https://docs.mongodb.org/manual
   Main PID: 5211 (mongod)
    Memory: 57.2M
    CGroup: /system.slice/mongod.service
            └─5211 /usr/bin/mongod --config /etc/mongod.conf

ene 12 10:35:58 ubuntu-VirtualBox systemd[1]: Started MongoDB Database Server.
lines 1-10/10 (END)
```

Con el siguiente comando, MongoDB se activará después de reiniciar el ordenador

```
ubuntu@ubuntu-VirtualBox:~$ sudo systemctl enable mongod
Created symlink /etc/systemd/system/multi-user.target.wants/mongod.service → /lib/systemd/system/mongod.service.
```

Para comenzar a usar MongoDB, ejecutamos el siguiente comando.



```
ubuntu@ubuntu-VirtualBox: ~
ubuntu@ubuntu-VirtualBox:~$ mongo
MongoDB shell version v4.4.3
connecting to: mongodb://127.0.0.1:27017/?compressors=disabled&gssapiServiceName=mongod
Implicit session: session { "id" : UUID("531ced06-226a-413b-a410-9e9c7915a1f0") }
MongoDB server version: 4.4.3
Welcome to the MongoDB shell.
For interactive help, type "help".
For more comprehensive documentation, see
  https://docs.mongodb.com/
Questions? Try the MongoDB Developer Community Forums
  https://community.mongodb.com
---
The server generated these startup warnings when booting:
  2021-01-12T10:35:58.415+01:00: Using the XFS filesystem is strongly recommended with the WiredTiger storage engine. See http://dochub.mongodb.org/core/prodnotes-filesystem
  2021-01-12T10:35:59.012+01:00: Access control is not enabled for the database. Read and write access to data and configuration is unrestricted
---
---
  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()
---
> |
```

De esta manera, nos aparece en pantalla la interfaz de texto de MongoDB en la cual podemos introducir sentencias del DDL y DML.

2. Descripción del DDL y DML

Detallamos a continuación los distintos métodos que MongoDB para administrar la base de datos. En el siguiente apartado veremos ejemplos de uso de dichos métodos.

- Operaciones de creación: se basan en añadir documentos a una colección. Si la colección no existe, las operaciones de inserción la crean automáticamente. MongoDB proporciona los siguientes métodos:
 - `db.collection.insertOne()`
 - `db.collection.insertMany()`

La estructura de la sentencias sería la siguiente:

```
db.collection_name.insertOne(  
  {  
    campo1: "valor1",  
    campo2: "valor2",  
    campo3: "valor3"  
  }  
)
```

Notemos que en el lenguaje de MongoDB no hay ninguna sentencia para indicar que se referencia una clave externa.

- Operaciones de lectura: permiten hacer consultas sobre los documentos de una colección. Proporciona el siguiente método:
 - `db.collection.find()`
- Operaciones de actualización: permiten modificar los documentos de una colección. MongoDB pone a disposición los siguientes métodos:
 - `db.collection.updateOne()`
 - `db.collection.updateMany()`
 - `db.collection.replaceOne()`
- Operaciones de borrado: elimina un documento de una colección. Encontramos los siguientes métodos:
 - `db.collection.deleteOne()`
 - `db.collection.deleteMany()`

3. Sentencias empleadas

A continuación, mostramos algunas sentencias empleadas en la Práctica 3 en SQL con su equivalente en MongoDB y el resultado que se obtiene al ejecutarla en el cliente de texto básico.

| | |
|-----------|--|
| SQL | <pre>INSERT INTO Empleado (DNI, Nombre, Apellidos, Telefono, Puesto, FechaNacimiento, NSeguridadSocial, Cuenta) VALUES ('33333333C', 'Manuel', 'Carrero', '633333333', 'Limpieza', TO_DATE('1965-01-10', 'YYYY-MM-DD'), '33333333', 'ES6621507418401233456365');</pre> |
| MongoDB | <pre>db.Empleado.insert({ DNI:"33333333C", Nombre:"Manuel", Apellidos:"Carrero", Telefono:"633333333", Puesto:"Limpieza", FechaNacimiento:new Date("1965, 1, 10"), NSeguridadSocial:"33333333", Cuenta:"ES6621507418401233456365" })</pre> |
| Resultado | <pre>WriteResult({ "nInserted" : 1 })</pre> |

| | |
|---------|---|
| SQL | <pre>INSERT INTO Empleado (DNI, Nombre, Apellidos, Telefono, Puesto, FechaNacimiento, NSeguridadSocial, Cuenta) VALUES ('22222222B', 'Carmen', 'San Diego', '622222222', 'Recepcionista', TO_DATE('1985-12-11', 'YYYY-MM-DD'), '22222222', 'ES6621000418401233456789');</pre> |
| MongoDB | <pre>db.Empleado.insert({ DNI:"22222222B", Nombre:"Carmen", Apellidos:"San Diego", Telefono:"622222222", Puesto:"Recepcionista", FechaNacimiento:new Date("1985, 12, 11"), NSeguridadSocial:"22222222", Cuenta:"ES6621000418401233456789" })</pre> |

| | |
|-----------|----------------------------------|
| Resultado | WriteResult({ "nInserted" : 1 }) |
|-----------|----------------------------------|

| | |
|-----------|--|
| SQL | INSERT INTO Cliente (DNI, CorreoElectronico) VALUES ('12345678S', 'pepitojd97@go.ugr.es'); |
| MongoDB | db.Cliente.insert({ DNI:"12345678S", Nombre:"pepitojd97@go.ugr.es" }) |
| Resultado | WriteResult({ "nInserted" : 1 }) |

| | |
|-----------|--|
| SQL | INSERT INTO Reserva (Identificador, DNI, TipoHab, FechaEntrada, FechaSalida) VALUES ('R00000001', '12345678S', 'I', TO_DATE('2021-12-11', 'YYYY-MM-DD'), TO_DATE('2021-12-14', 'YYYY-MM-DD')); |
| MongoDB | db.Reserva.insert({ Identificador:"12345678S", DNI:"12345678S", Tipo:"I", FechaEntrada:new Date("2021, 12, 11"), FechaSalida:new Date("2021, 12, 14") }) |
| Resultado | WriteResult({ "nInserted" : 1 }) |

| | |
|-----------|---|
| SQL | SELECT * FROM Empleado; |
| MongoDB | db.Empleado.find() |
| Resultado | { "_id" : ObjectId("5fffd8352d03ec6897539442a"), "DNI" : "33333333C", "Nombre" : "Manuel", "Apellidos" : "Carrero", "Telefono" : "633333333", "Puesto" : "Limpieza", "FechaNacimiento" : ISODate("1965-01-09T23:00:00Z"), "NSeguridadSocial" : "33333333", "Cuenta" : "ES6621507418401233456365" } { "_id" : ObjectId("5fffd835ed03ec6897539442b"), "DNI" : "22222222B", "Nombre" : "Carmen", "Apellidos" : "San |

| | |
|--|---|
| | Diego", "Telefono" : "62222222", "Puesto" : "Recepcionista", "FechaNacimiento" : ISODate("1985-12-10T23:00:00Z"), "NSeguridadSocial" : "22222222", "Cuenta" : "ES6621000418401233456789" } |
|--|---|

| | |
|-----------|---|
| SQL | SELECT * FROM Empleado WHERE DNI='22222222B'; |
| MongoDB | db.Empleado.find({DNI:"22222222B"}) |
| Resultado | { "_id" : ObjectId("5fffd835ed03ec6897539442b"), "DNI" : "22222222B", "Nombre" : "Carmen", "Apellidos" : "San Diego", "Telefono" : "62222222", "Puesto" : "Recepcionista", "FechaNacimiento" : ISODate("1985-12-10T23:00:00Z"), "NSeguridadSocial" : "22222222", "Cuenta" : "ES6621000418401233456789" } |

| | |
|-----------|---|
| SQL | UPDATE Cliente SET CorreoElectronico='pepito2@correo2.com' WHERE DNI='12345678S'; |
| MongoDB | db.Cliente.update({DNI:"12345678S"}, {CorreoElectronico:"pepito2@correo2.com"}) |
| Resultado | WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 }) |

| | |
|-----------|---|
| SQL | DELETE FROM Empleado WHERE DNI='33333333C'; |
| MongoDB | db.Empleado.deleteOne({DNI:"33333333C"}) |
| Resultado | { "acknowledged" : true, "deletedCount" : 1 } |

4. Conexión desde una aplicación

Vamos a realizar la conexión entre MongoDB y una aplicación en Python.

Para ello, primero habría que instalar el driver necesario, en la documentación oficial recomiendan PyMongo, por lo que usaremos ese. Para instalarlo ejecutamos:

```
python -m pip install pymongo
```

Ahora, en la aplicación debemos conectarnos a la base de datos mediante este drive. Un código de conexión sería como se indica:

```
from pymongo import MongoClient
# conectamos a MongoDB cambiando <<MongoDB URL>> a nuestra URL
client = MongoClient(<<MongoDB URL>>)
db = client.admin
# Comprobamos el estado del servidor y lo mostramos por pantalla
serverStatusResult = db.command("serverStatus")
print(serverStatusResult)
```

Al ejecutar un programa en python con dichas sentencias se establece una conexión a la base de datos de MongoDB y se muestra por pantalla el estado actual de la base de datos. Generalmente, tras abrir la conexión, para crear un objeto de la base de datos referenciando a una nueva base de datos llamada "business":

```
db = client.business
```

Una vez creado, podemos realizar algunas sentencias de creación, lectura, modificación y borrado. Aquí un código de ejemplo:

```
from pymongo import MongoClient
from random import randint
#Step 1: Connect to MongoDB - Note: Change connection string as needed
client = MongoClient(port=27017)
db=client.business
#Step 2: Create sample data
names = ['Kitchen','Animal','State', 'Tastey', 'Big','City','Fish', 'Pizza','Goat', 'Salty','Sandwich','Lazy', 'Fun']
company_type = ['LLC','Inc','Company','Corporation']
company_cuisine = ['Pizza', 'Bar Food', 'Fast Food', 'Italian', 'Mexican', 'American', 'Sushi Bar', 'Vegetarian']
for x in range(1, 501):
    business = {
        'name':names[randint(0,(len(names)-1))]+ ' '+names[randint(0,(len(names)-1))] + ' '+company_type[randint(0,(len(company_type)-1))],
        'rating':randint(1, 5),
        'cuisine':company_cuisine[randint(0, (len(company_cuisine)-1))]
    }
    #Step 3: Insert business object directly into MongoDB via insert_one
    result=db.reviews.insert_one(business)
    #Step 4: Print to the console the ObjectID of the new document
    print('Created {0} of 500 as {1}'.format(x,result.inserted_id))
#Step 5: Tell us that you are done
print('finished creating 500 business reviews')
```

Para realizar consultas, podemos por ejemplo obtener todos los restaurantes con 5 estrellas:

```
fivestar = db.reviews.find_one({'rating': 5})
print(fivestar)
```

Para actualizar un valor existente, se usaría

```
result = db.reviews.update_one({'_id' : ASingleReview.get('_id') },
{'$inc': {'likes': 1}})
print('Number of documents modified: '+str(result.modified_count))
```

Para borrar documentos se usa la siguiente sentencia:

```
result = db.restaurants.delete_many({"category": "Bar Food"})
```

Puede consultarse más información al respecto en:

<https://www.mongodb.com/blog/post/getting-started-with-python-and-mongodb>

5. ¿Sería adecuado el SGBD MongoDB para implementar el SI de la práctica?

No sería la mejor opción para el sistema de información desarrollado en la práctica.

Los SGBD NoSQL como MongoDB están enfocados en sistemas sin estructuras fijas y que son cambiantes, pues proporcionan una alta flexibilidad y tienen un buen escalamiento horizontal. En este sentido, están adaptados para trabajar con datos semiestructurados y no estructurados. Además, el mantenimiento de la consistencia está separado de los datos, de modo que debe mantenerse desde las aplicaciones. En particular, MongoDB guarda la estructura de los datos en documentos BSON con un esquema dinámico, lo que implica que no existe un esquema predefinido.

En contraposición, el sistema de información para la gestión de un hotel que estamos implementando en la práctica tiene una estructura fija y no compleja y además el uso de un SGBD relacional resulta más conveniente en tanto que permite integrar las restricciones de los datos junto a los mismos, liberando a las aplicaciones de controlar su mantenimiento.