**Tema 4: MongoDB**

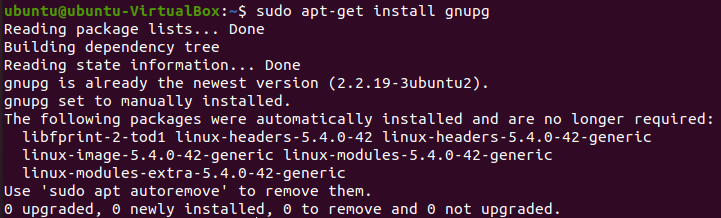
# **1. Descarga e instalación**

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.

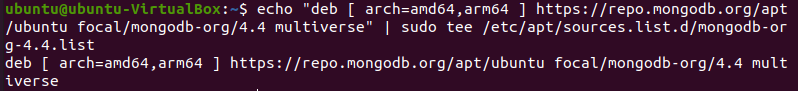
Primero, comprobamos que tenemos gnupg instalado y sus bibliotecas requeridas.



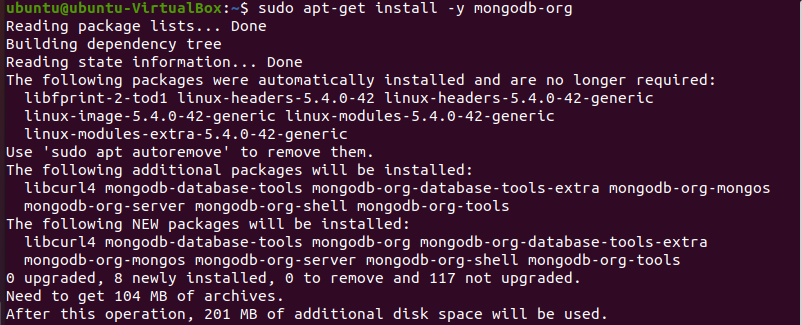
Importamos la clave pública utilizada por el sistema de gestión de paquetes.



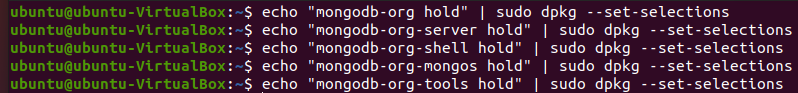
Creamos un archivo necesario para MongoDB.



Instalamos los paquetes de MongoDB.



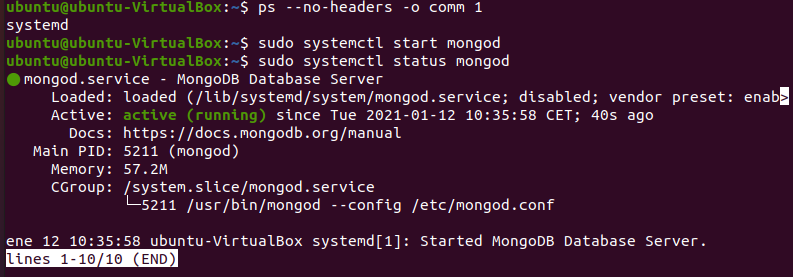
Fijamos la versión instalada de los paquetes.



Vemos el sistema que usa nuestra plataforma.



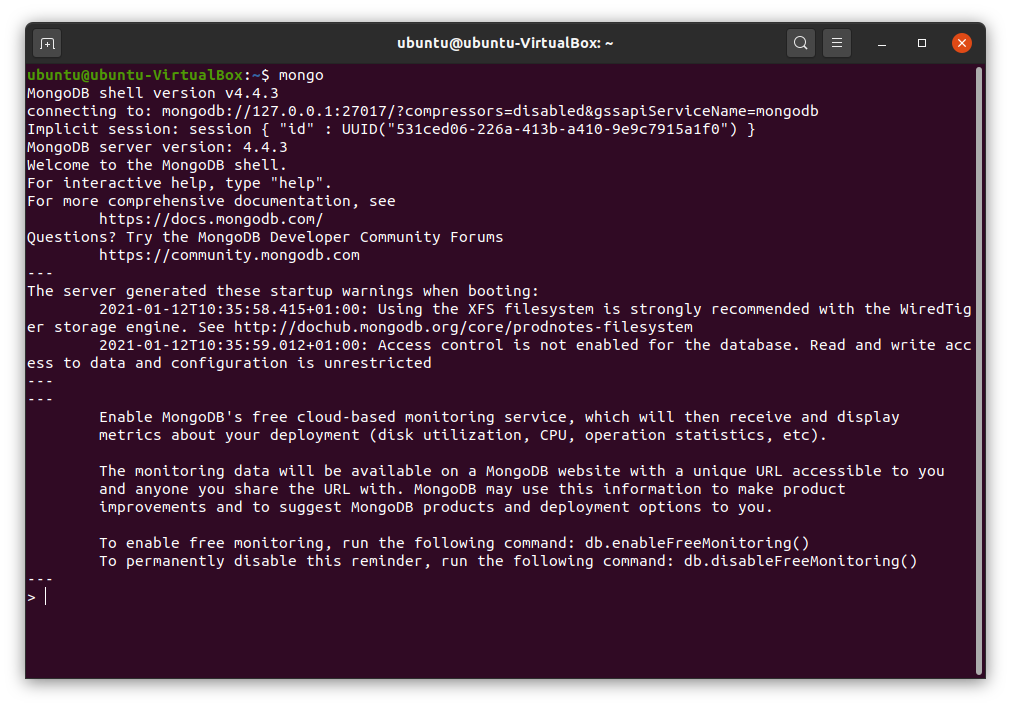
Iniciamos el proceso de MongoDB y verificamos que se ha activado correctamente.



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



Para comenzar a usar MongoDB, ejecutamos el siguiente comando.



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 | 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'); |
| MongoDB | db.Empleado.insert(  {  DNI:"33333333C",  Nombre:"Manuel",  Apellidos:"Carrero",  Telefono:"633333333",  Puesto:"Limpieza",  FechaNacimiento:new Date("1965, 1, 10"),  NSeguridadSocial:"33333333",  Cuenta:"ES6621507418401233456365"  }  ) |
| Resultado | WriteResult({ "nInserted" : 1 }) |

|  |  |
| --- | --- |
| SQL | 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'); |
| MongoDB | db.Empleado.insert(  {  DNI:"22222222B",  Nombre:"Carmen",  Apellidos:"San Diego",  Telefono:"622222222",  Puesto:"Recepcionista",  FechaNacimiento:new Date("1985, 12, 11"),  NSeguridadSocial:"22222222",  Cuenta:"ES6621000418401233456789"  }  ) |
| Resultado | WriteResult({ "nInserted" : 1 }) |

|  |  |
| --- | --- |
| SQL | INSERT INTO Cliente (DNI, CorreoElectronico) VALUES ('12345678S', '[pepitojd97@go.ugr.es](mailto:pepitojd97@go.ugr.es)'); |
| MongoDB | db.Cliente.insert(  {  DNI:"12345678S",  Nombre:"[pepitojd97@go.ugr.es](mailto: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("5ffd8352d03ec6897539442a"), "DNI" : "33333333C", "Nombre" : "Manuel", "Apellidos" : "Carrero", "Telefono" : "633333333", "Puesto" : "Limpieza", "FechaNacimiento" : ISODate("1965-01-09T23:00:00Z"), "NSeguridadSocial" : "33333333", "Cuenta" : "ES6621507418401233456365" }  { "\_id" : ObjectId("5ffd835ed03ec6897539442b"), "DNI" : "22222222B", "Nombre" : "Carmen", "Apellidos" : "San Diego", "Telefono" : "622222222", "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("5ffd835ed03ec6897539442b"), "DNI" : "22222222B", "Nombre" : "Carmen", "Apellidos" : "San Diego", "Telefono" : "622222222", "Puesto" : "Recepcionista", "FechaNacimiento" : ISODate("1985-12-10T23:00:00Z"), "NSeguridadSocial" : "22222222", "Cuenta" : "ES6621000418401233456789" } |

|  |  |
| --- | --- |
| SQL | UPDATE Cliente SET CorreoElectronico=’[pepito2@correo2.com](mailto:pepito2@correo2.com)’ WHERE DNI=’12345678S’; |
| MongoDB | db.Cliente.update({DNI:"12345678S"}, {CorreoElectronico:"[pepito2@correo2.com](mailto: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:



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.