ACTIVIDADES POR DESARROLLAR:

1. Introducción a las bases de datos no relacionales (NOSQL)
2. instalación de un sistema manejador de base de datos no relacional (MONGODB)
3. Comandos utilizados en MONGODB
4. Migrar base de datos relacionales a no relacional
5. Implementar FrontEnd para crear la conectividad hacia MONGODB

EVIDENCIA(S) A ENTREGAR:

1. Escribe un ensayo sobre que son las bases de datos NOSQL y aplicaciones existentes.
2. Elabora informe de la ejecución del plan de instalación de MONGODB.
3. Realiza los ejercicios propuestos en el presente taller.
4. Realiza la conversión de la base de datos SALUD hacia MONGODB y entrega el script de creación y el respectivo informe.
5. Desarrolla en Python un programa que se conecte a una base de datos MONGODB y a una colección mostrando por pantalla los datos guardados.

DURACIÓN: 24 HORAS 16 horas trabajo autónomo.

CONTROL DEL DOCUMENTO

	Nombre	Cargo	Dependencia	Fecha
Autor (es)	JOSE FERNANDO GALINDO SUAREZ	INSTRUCTOR	CGMLTI	14/09/2022

CONTROL DE CAMBIOS (diligenciar únicamente si realizan ajustes al taller)

	Nombre	Cargo	Dependencia	Fecha	Razón del Cambio
Autor (es)					

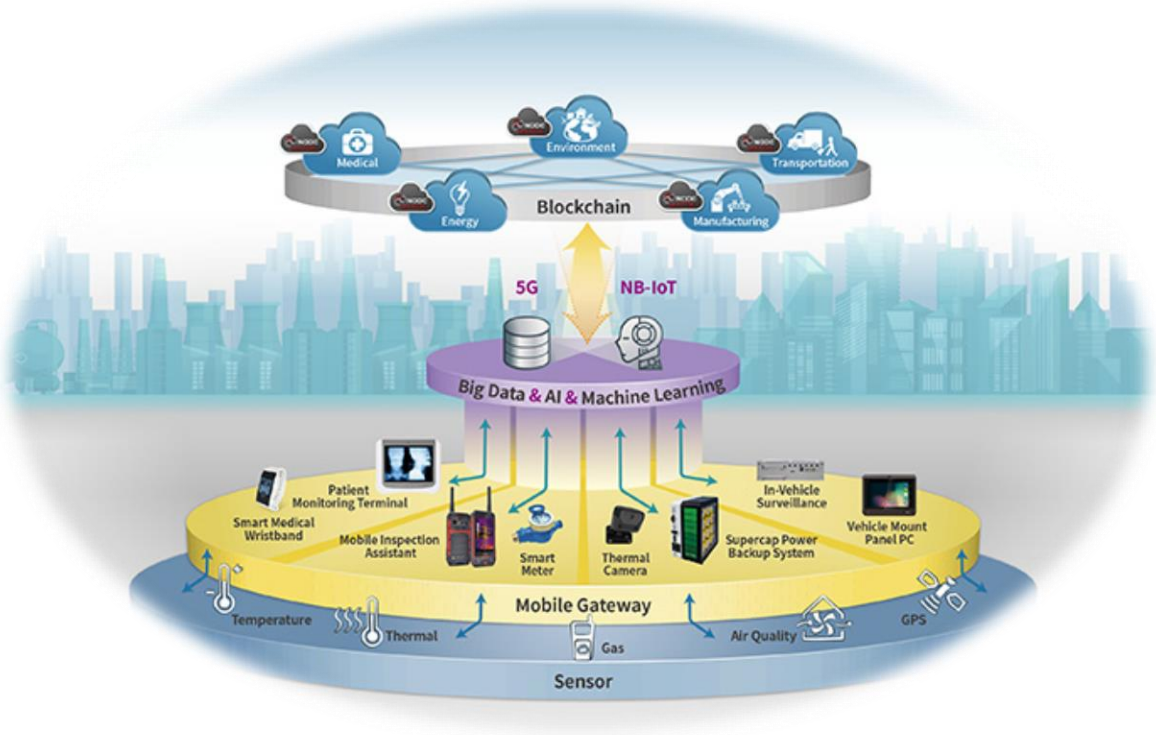
PRESENTACIÓN

El auge de internet ha propiciado la generación de grandes volúmenes de datos de fuentes de información como sistemas de información, correos electrónicos, redes sociales, sensores de internet de las cosas; en diferentes formatos o sin formato alguno, a esto lo llamamos información estructurada, semiestructura y no estructuradas.

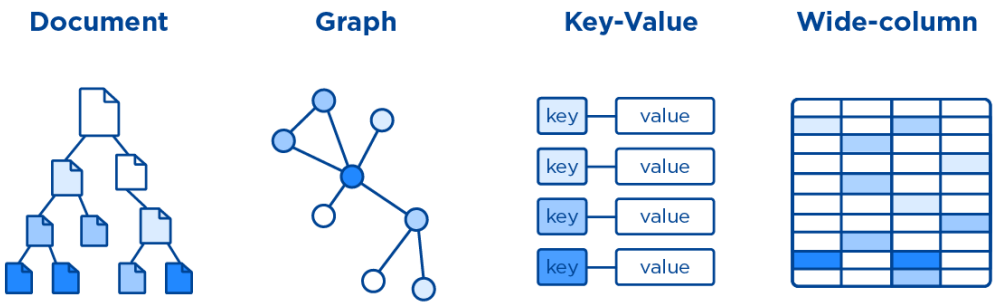
Las bases de datos no relacionales llamadas NOSQL de la sigla "NOT ONLY SQL", es un sistema manejador de base de datos para trabajos de BIGDATA orientada a colecciones y documentos, utilizando para los documentos el formato JSON (Java Script Object Notation).

BSON es un formato de intercambio de datos usado principalmente para su almacenamiento y transferencia en la base de datos MongoDB. Es una representación binaria de estructuras de datos y mapas. El nombre BSON está basado en el término JSON y significa Binary JSON (JSON Binario).[i]

Fuentes generadoras de información



TIPO DE BASE DE DATOS NOSQL.



ORIENTADAS A COLUMNAS

- Se almacena en columna como llave y valor
- Se encuentran en la misma partición
- Se incrementa la velocidad

ORIENTADAS A DOCUMENTOS

- Almacena la información en forma de documentos
- Utiliza formatos semi estructurados como JSON
- Naturaleza de tipo jerárquica (documentos dentro de documentos)
- Una colección es un conjunto de documentos

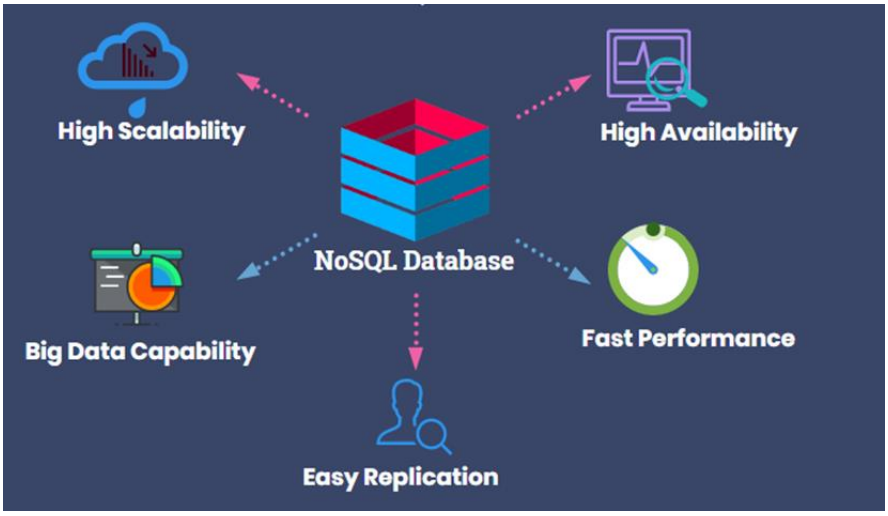
ORIENTADOS POR CLAVE-VALOR

- Se almacena la información como clave-valor
- La clave es el identificador único
- Incrementa la velocidad de búsqueda
- Algunas se trabajan directamente en la memoria

ORIENTADAS A GRAFOS

- Representan la información como nodos
- Representan las relaciones como aristas
- Estas almacenan información como nodos y aristas

CARACTERISTICAS DE LAS BASES DE DATOS NOSQL



- Normalmente son distribuidas
- Escalables
- Flexibles
- No soportan subconsulta y uniones de tablas (JOIN)
- No soportan los métodos ACID (Atomicy, Consistency, Isolation, Durability)
- Garantiza el paradigman BASE (Base Ability, Soft State, Eventual Consistency)
- Se almacena la información de forma flexible, donde un documento puede variar con respecto a las propiedades de otro documento de la colección
- Se utilizan para el manejo de grandes volúmenes de datos (BIGDATA)

ESTRUCTURAS UTILIZADAS CON JSON

COLUMNAS

- Identificador HASH dado por el SMBD-NOSQL (_id)
- Propiedades Clave-Valor

```
{
  "idemp": 1,
```

```
  "nom": "Pedro Palitos",
  "genero": "M",
  "edad": 23,
  "dni": 19407970
}
```

SUPER COLUMNA

Documento compuesto por varias columnas.

```
{
  "idemp":12,
  "nom":"Carlos Villagran",
  "edad":62,
  "dni":{
    "tipo":"CC",
    "numero":19407906,
    "fecha":"14/07/1979"
  }
}
```

FAMILIA DE COLUMNAS

Un documento puede guardar otro documento.

```
{
  "idemp":12,
  "nom":"Carlos Villagran",
  "edad":62,
  "dni":{
    "tipo":"CC",
    "numero":19407906,
    "fecha":"14/07/1979"
  },
  "cursos":["python","java","php","mongodb"],
  "mensajes":[
    {
      "subject":"Reunión de planeación",
      "body":"agenda de desarrollo web",
      "from":"sotico@gmail.com",
      "fecha":"2022-02-27"
    }, {
      "body":"Entrega de stickholder del proyecto",
```

```
      "from":"cpinilla@gmail.com",
      "fecha":"2022-03-12"
    }
  ]
}
```

Antes de insertarlo se debe verificar que el JSON fue escrito de forma correcta, para esto se debe verificar con la pagina [Validador y formateador JSON Online - JSON Lint](#).

PRACTICA MONGODB

Para comenzar la práctica, se debe descargar los archivos con GIT (ver "VIDEOS/Instalación/ clonarrepositoriogit.cmd" o sin GIT (ver "VIDEOS/Instalación/ descargar archivos mongodb.cmd"). Los archivos se encuentran en: http://github.com/fegasu/EDTMONGODB o descargar Aquí

Descargar los siguientes archivos de instalación:

- Servidor MongoDB (SW/mongoserver.cmd)
- Consola MongoDB (SW/mongosh.cmd)
- Mongo Compas (SW/mongocompass.cmd)

INSTALACION SERVIDOR MONGODB

Para la instalación del servidor MONGODB se recomienda ver el video: "VIDEOS/Instalación/mongoinstalacion.cmd"

INSTALACION CONSOLA MONGODB

Para la instalación de la consola de MONGODB se recomienda ver el video: "VIDEOS/Instalación/instalarmongosh.cmd"

MOSTRAR LAS BASE DE DATOS CREADAS

Ingresar a MONGOSH y digitar el comando:

```
show databases;
```

CREAR UNA BASE DATOS

```
use prueba;
```

Tener en cuenta cuando se escriba db se refiere a la base de datos que se está usando. El comando use se debe utilizar para seleccionar una base de datos.

CREANDO UNA COLECCIÓN

Una colección guarda un documento o muchos documentos, utilice el siguiente comando:

```
db.createCollection("emp")
```

INSERTAR UN DOCUMENTO DENTRO DE UNA COLECCIÓN

Se utiliza la clausula "insertOne" de la siguiente manera para insertarlo dentro de la colección "emp", para esto se debe escribir el documento en el formato JSON así:

```
emp1={
  "idemp": 1,
  "nom": "Pedro Palitos",
  "genero": "M",
  "edad": 23,
  "dni": 19407970
}
```

Para insertarlo dentro de la colección "emp" de la base de datos prueba

```
db.emp.insertOne(emp1);
```

Inserte los siguientes datos:

Idemp	Nom	Genero	Edad	dni
1	Pedro Palitos	M	23	19456970
2	Maria Casquito	F	32	
3	Celia Gacha	F	43	

INSERTANDO MAS DE UN DOCUMENTO A LA VEZ

Construimos los documentos emp4 y emp5 así:

```
emp4={
    "idemp":4,
    "nom":"Maria la bandida",
    "genero":"F",
    "edad":32
}
emp5={
    "idemp": 5,
    "nom": "ROSA MELO",
    "genero": "F",
    "edad":25
}
```

Utilizamos la cláusula "insertMany" así:

```
db.emp.insertMany([emp4,emp5])
```

También podemos hacerlo así:

```
db.emp.insertMany([{
    "idemp":6,
    "nom":"Juan Tolomeo",
    "genero":"M",
    "edad":37
},
{
    "idemp": 7,
    "nom": "Carlos Pichamata",
    "genero": "M",
    "edad":52
}])
```

Ingresar los siguientes empleados:

Idemp	Nom	Genero	Edad	dni
8	Manuela Beltran	F	15	123456
9	Monica Galindo	F	23	6789123
10	Elver Galindo	M	21	
11	Simón Tolomeo	M	51	1357901

LISTAR TODOS LOS DOCUMENTOS DE UNA COLECCIÓN

Se utiliza la cláusula "find()" así:

```
db.emp.find();
```

Para buscar con una condición simple, por ejemplo, buscar el empleado 6:

```
db.emp.find({
    idemp:"6"
})
```

Para encontrar los empleados mayores a 30

```
db.emp.find(
    {edad:{$gt:30}}
)
```

Para encontrar los empleados mayores o iguales a 30

```
db.emp.find(
    {edad:{$gte:30}})
```

Para encontrar los empleados menore a 30

```
db.emp.find(
    {edad:{$lt:30}}
)
```

Para encontrar los empleados menores o iguales a 30

```
db.emp.find(
    {edad:{$lte:30}}
)
```

Para encontrar los empleados de 23, 42 y 21

```
db.emp.find(
              {
                  edad:{$in:[23,42,21]}
              }
            )
```

UTILIZANDO BUSQUEDAS CON OPERADORES LOGICOS

Utilizando el operador lógico AND

```
db.emp.find({
        $and:[
        {edad:{$lte:35}},
        {genero:"F"}]
        })
```

Utilizando el operador lógico OR

```
db.emp.find({
        $or:[
        {edad:{$lte:35}},
        {genero:"F"}]
        })
```

Encuentre los empleados que tienen “dni”:

```
db.emp.find(
              {
                 dni:{$exists:true}
               }
             )
```

ORDENANDO LA SALIDA DE LOS DOCUMENTOS

De forma descendente:

```
              db.emp.find().sort(
                {
                  edad:-1
                 })
```

Encontrar el empleado que tiene la máxima edad

```
db.emp.find().sort(
  {
    edad:-1
  }
).limit(1)
```

UTILIZAR EXPRESIONES REGULARES

Encontrar empleados que comience con el patrón palabra "Simón"

```
db.emp.find(
  {
    nom:/^ Simón /
  })
```

Encontrar empleados que contenga la palabra "ver"

```
db.emp.find(
  {
    nom:/ver/
  }
)
```

Cuantos documentos hay en la colección "emp"

```
db.emp.find().count()
7
```

Cuantos empleados hay con una edad menor o igual a 25

```
db.emp.find({edad:{$lte:25}}).count()
4
```

Mostrar dos documentos a partir del tercer documento

```
db.emp.find().skip(2).limit(2)
```

MOSTRAR EL NOMBRE DE LOS DOCUMENTOS DE LA COLECCION

```
db.emp.find().forEach(df =>print(df.nom))
```

ASIGNAR A UNA VARIABLE UN DOCUMENTO

```
var user=db.emp.findOne()
user.nom='PEDRO PALOS GORDOS'
```

ACTUALIZAR UN DOCUMENTO

```
db.emp.updateOne(
  {idemp: 1} ,
  {$set:{"nom":" Juan Charrasqueado"}}
)
```

ADICIONAR UN NUEVO ATRIBUTO A UN DOCUMENTO

```
db.emp.updateOne(
  {idemp: 1} ,
  {$set:{"Correo":"fegasu@misena.edu.co"}}
)
```

INCREMENTAR UN ATRIBUTO EN TODOS LOS DOCUMENTOS

```
db.emp.updateMany(
  {} ,
  {$inc:{edad:1}}
)
```

DECREMENTAR UN ATRIBUTO EN TODOS LOS DOCUMENTOS

```
db.emp.updateMany(
  {} ,
  {$inc:{edad:-1}}
)
```

BORRAR UN DOCUMENTO

```
db.emp.deleteOne(
  {idemp:7}
)
```

ADICIONAR ATRIBUTOS A UN DOCUMENTOS

```
Emp12={
  "idemp":12,
  "nom":"Carlos Villagran",
  "edad":62,
  "dni":{
    "tipo":"CC",
    "numero":19407906,
    "fecha":"14/07/1979"
  },
  "cursos":["python","java","php","mongodb"],
  "mensajes":[
    {
      "subject":"Reunión de planeación",
```

```
      "body":"agenda de desarrollo web",
      "from":"sotico@gmail.com",
      "fecha":"2022-02-27"
    },
    {
      "body":"Entrega de stickholder del proyecto",
      "from":"cpinilla@gmail.com",
      "fecha":"2022-03-12"
    }
  ]
}
db.emp.insertOne(emp12)
```

AGREGA LA PROPIEDAD A LOS DOCUMENTOS QUE NO LO TENGA.

Agrega la propiedad dni a los documentos que no lo tienen

```
db.emp.updateMany(
  {
    'dni':{$exists:false}
  },
  {
    $set:{"dni":{
      'tipo':'CC',
      'numero':0,
      'fecha':''
    }
  }
  }
)
```

ADICIONA DOCUMENTOS AL FINAL.

```
db.emp.updateOne(
  {
    idemp:9
  },
  {$push:{
    mensajes:{
      subject:"Pruebas TYT",
      body:"Pruebas TyT Tecnologos Sena",
```

```
      from:"secretaria@sena.edu.co",
      fecha:"2022-08-27"
    }
  }
}
)
```

AGREGACIONES[ii]

Las operaciones basadas en agregaciones permiten procesar la data registrada para obtener resultados que aportan información útil, permitiendo darle más valor a la data al transformarla en información.

- Operadores de etapa
  - $match, $project, $sort, $limit, $skip, $count, $group, $sample, $out, $addFields, y $sortByCount
- Operadores de expresiones
  - Operadores comparativos
    - $eq, $ne, $gt, $gte, $lt, $lte y $cmp
  - Operadores booleanos
    - $and, $or y $not
  - Operadores aritméticos
    - $abs, $add, $multiply, $substract, $divide, $mod, $pow, $sqrt, $exp, $ln, $log, $ceil, $floor y $trunc
  - Operadores sobre strings
    - $concat, $split, $indexOfBytes, $strcasecmp, $toLower, $toUpper y $strLenCP
  - Operadores de grupos (set):
    - $setEquals, $setIntersection, $setUnion, $setDifference, $setInSubset, $anyElementTrue y $allElementsTrue
  - Operadores sobre arregos (arrays)
    - $arrayElementAt, $concatArrays, $filter, $in, $indexOfArray, $isArray,
    - $map, $range, $reduce, $reverseArray, $size, $slice y $zip
  - Operador de variable
    - $let
  - Operador literal
    - $literal
  - Operador de tipo de dato
    - $type
  - Operadores condicionales
    - $cond, $ifNull y $switch
  - Operadores sobre fechas
    - $dayOfYear, $dayOfMonth, $dayOfWeek, $year, $month, $week, $hour, $minute, $second, $millisecond y $dateToString
  - Acumuladores

- $sum, $avg, $max, $min, $first y $last

¿QUÉ ES EL SHARDING?

El *sharding* es una forma de segmentar los datos de una base de datos de forma horizontal, es decir, dividirla en varias bases de datos más pequeñas y repartiendo la información, para conseguir una escalabilidad mucho más rápida.

Cuando no es posible almacenar la totalidad de datos en un único servidor, MongoDB permite una operación de escalado horizontal, repartiéndose en varios servidores, asegura la tolerancia a fallos de cada shard por separado, independientemente a cuál de ellos almacene un dato concreto. De esto se encarga el proceso MongoS.

¿CUÁNDO UTILIZAR SHARDING?

Antes de utilizar el método de fragmentado (sharding) se debería comprobar la viabilidad de un escalado vertical (es decir, ampliar las capacidades de RAM, CPU, red o disco en el servidor), teniendo en cuenta ciertos aspectos:

- *Coste económico: si el incremento de rendimiento no es proporcional al incremento de coste dentro de unos márgenes, entonces se debe descartar. Imaginemos que por necesidades se debe ampliar la memoria RAM del servidor, pero la ampliación nos supone un desembolso económico de 10 veces el coste inicial para una mejora en rendimiento del doble de rendimiento inicial. El escalado vertical no será la mejor opción.*
- *Coste operacional: se debe tener en cuenta, por ejemplo, que una ampliación de la capacidad de disco también supone una ampliación de los tiempos necesarios para backup y restore, así como cierto impacto sobre las conexiones de red. En este caso, distribuir los datos en varios fragmentos permitirá una copia más rápida por unidad, al permitir paralelización y un uso menor de ancho de banda. También se debe tener en cuenta que unos*

ficheros de datos de 15Tb implicará unos ficheros de índice de tamaño proporcional. Unos índices grandes requerirán mucho acceso a disco y el uso de una mayor cantidad de RAM por unidad de proceso

Cuando la información se encuentra distribuida en shards (fragmentos), la comunicación del cliente no se realiza directamente con los conjuntos replicados que almacenan la información, sino que se realiza con el proceso MongoS.

Cuando se ejecuta una consulta sobre un dato del que no se disponen metadatos, MongoS lanza la consulta a todos los shards del conjunto, y posteriormente los junta. A este proceso se le llama SHARD_MERGE. Una vez se hayan juntado todos los datos, se devuelve toda la información al cliente.

¿QUÉ ES EL BALANCEO?

El balanceo es la operación mediante la cual MongoDB distribuye de la manera más uniforme posible los datos entre los distintos shards.

La información dentro de los shards se distribuye en chunks, que son agrupaciones lógicas de documentos que pertenecen a un rango concreto en función de su clave.

Cuando MongoDB detecta que hay un exceso de chunks en un shard, mueve los sobrantes a otro shard asegurando así una distribución regular de los mismos.

El balanceador se ejecuta en el miembro primario del conjunto replicado de configuración (Config Server Replica Set).

Este proceso de balanceo comprueba la distribución de los chunks entre los distintos shards, y analiza los umbrales de migración.

Cuando detecta una descompensación, empiezan las rondas de balanceo.

Este proceso se ejecuta en paralelo, pero un shard no puede participar en más de una migración de forma simultánea.

Para saber cuántos chunks se pueden migrar de forma simultánea, dividiremos por 2 el número de shards y lo redondearemos a la baja.

Por ejemplo, para un escenario de 3 shards, sólo se podrá mover de forma simultánea 1 chunk.

Las rondas de balanceo se repiten tantas veces como sea necesario para obtener una distribución correcta de los chunks.

El balanceador tiene la posibilidad también de dividir los chunks en varias partes en caso de que detecte esta necesidad, o como parte de la definición de rangos de chunks al realizar zone sharding.

Cualquier operación de balanceo impacta en el rendimiento.

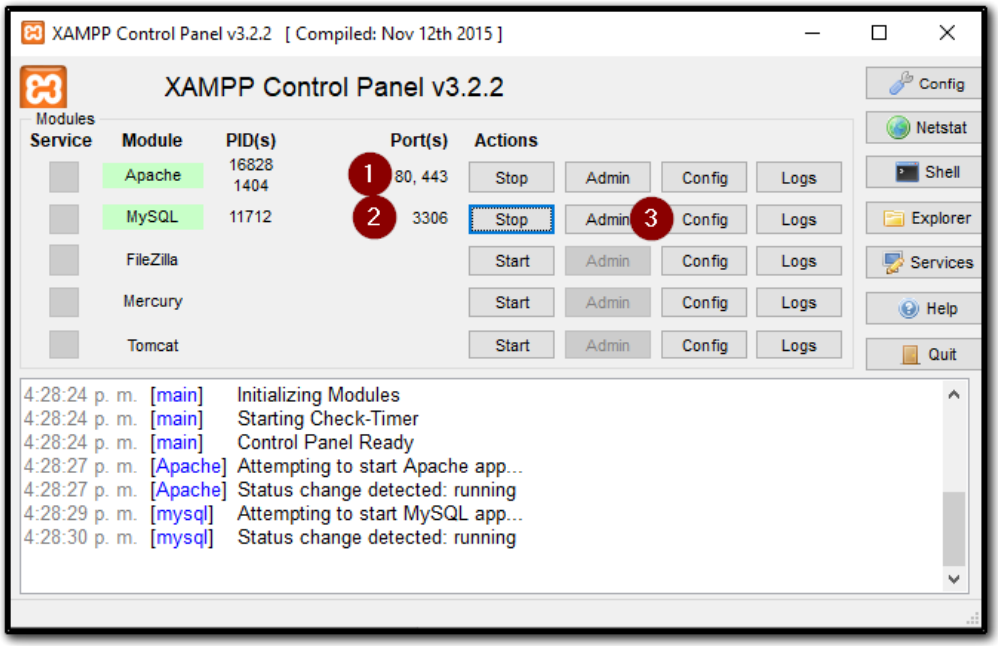
MongoDB dispone de 3 métodos para el control del comportamiento del balanceador:

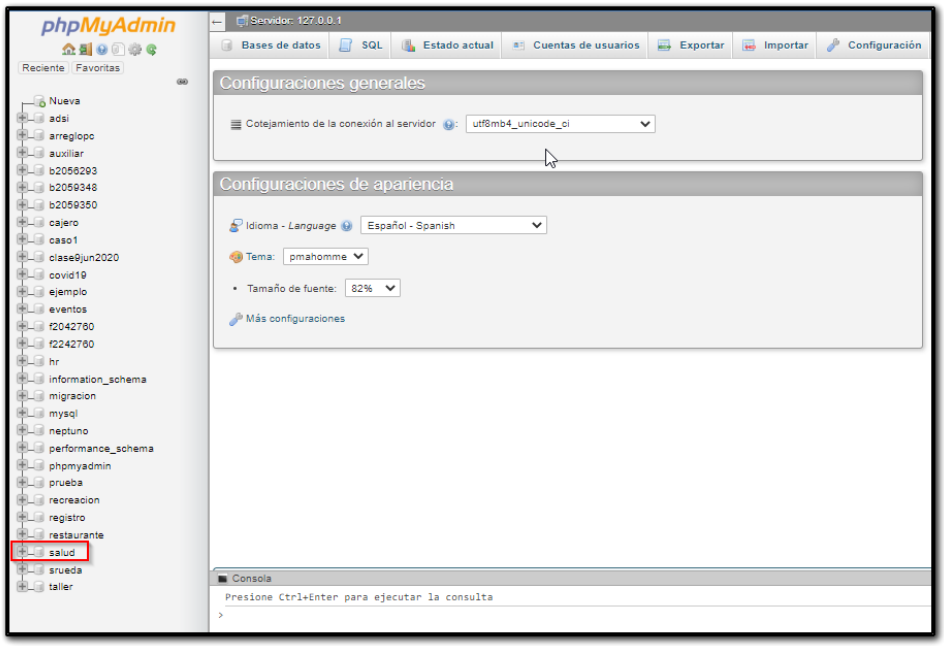
- *sh.startBalancer(timeout, intervalo) inicia el proceso de balanceo. Permite establecer un tiempo de espera (timeout) para que se inicie el proceso. El intervalo especifica el tiempo que el cliente deberá esperar antes de comprobar nuevamente el estado del balanceador.*
- *sh.stopBalancer(timeout, intervalo) detiene el proceso de balanceo. En caso de detenerse en mitad de una ronda de balanceo, éste se detiene cuando finaliza dicha ronda. Permite establecer un tiempo de espera (timeout) para que se detenga el proceso. El intervalo especifica el tiempo que el cliente deberá esperar antes de comprobar nuevamente el estado del balanceador.*

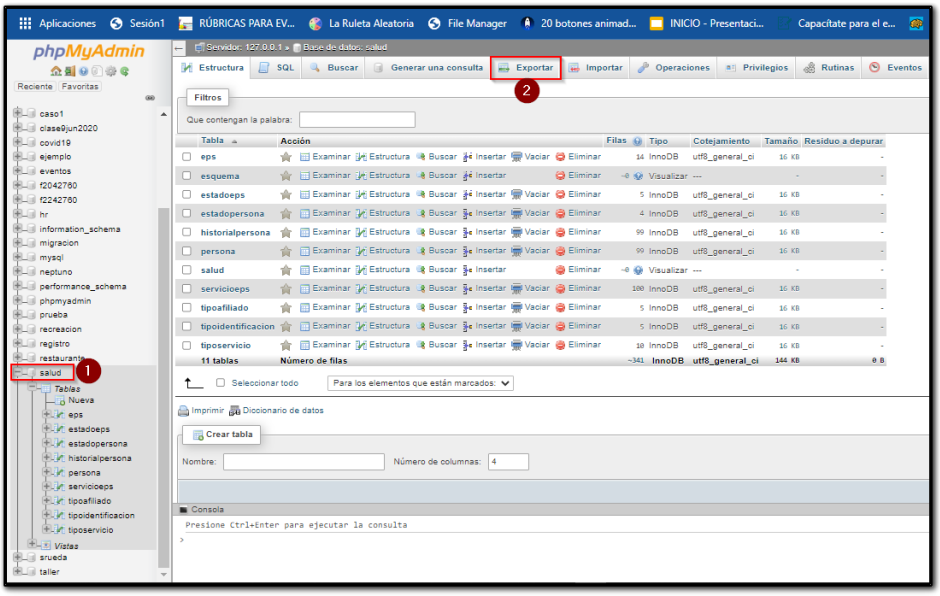
- *sh.setBalancerState(boolean) activa o desactiva el balanceador*

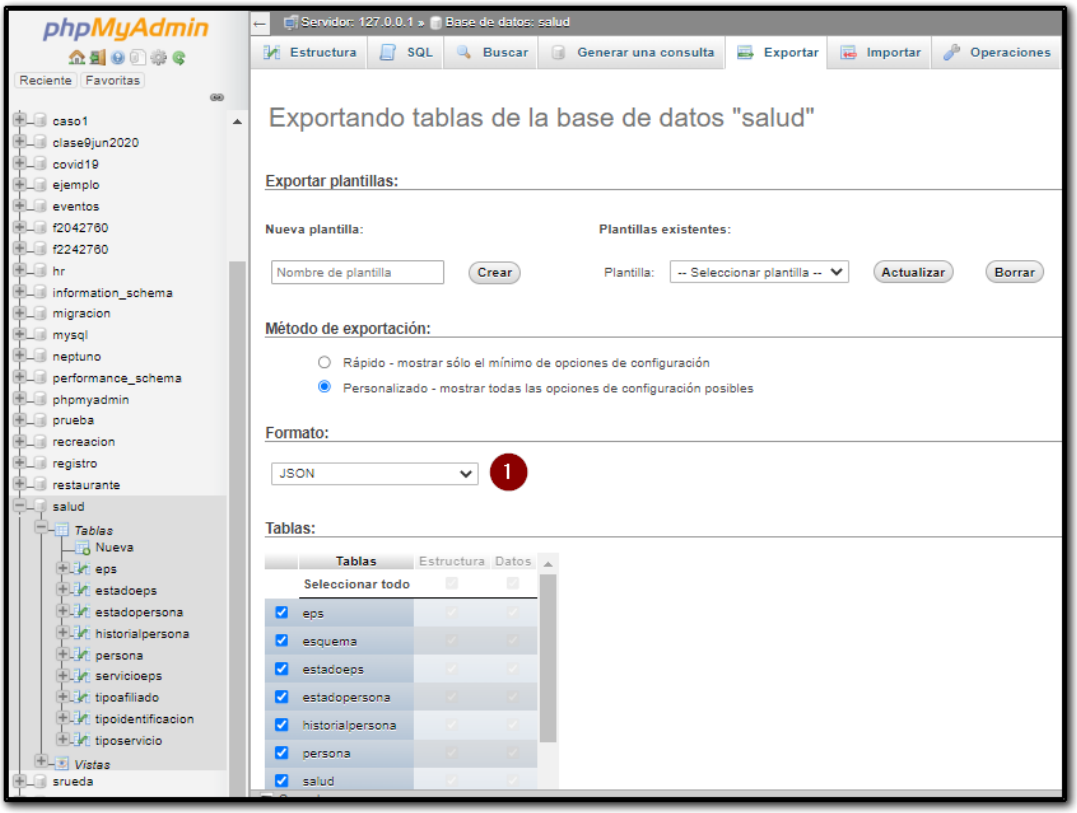
Mediante el fichero de configuración, se puede establecer una ventana de ejecución para el balanceador.

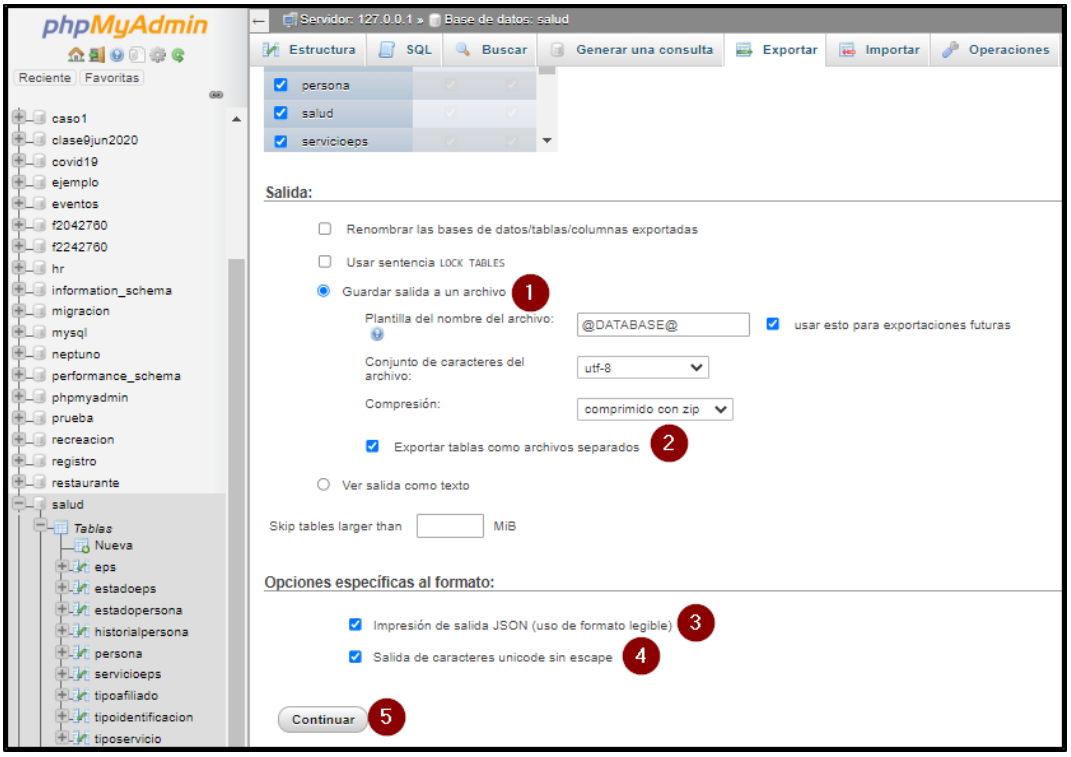
MIGRAR BASE DE DATOS MYSQL A MONGODB











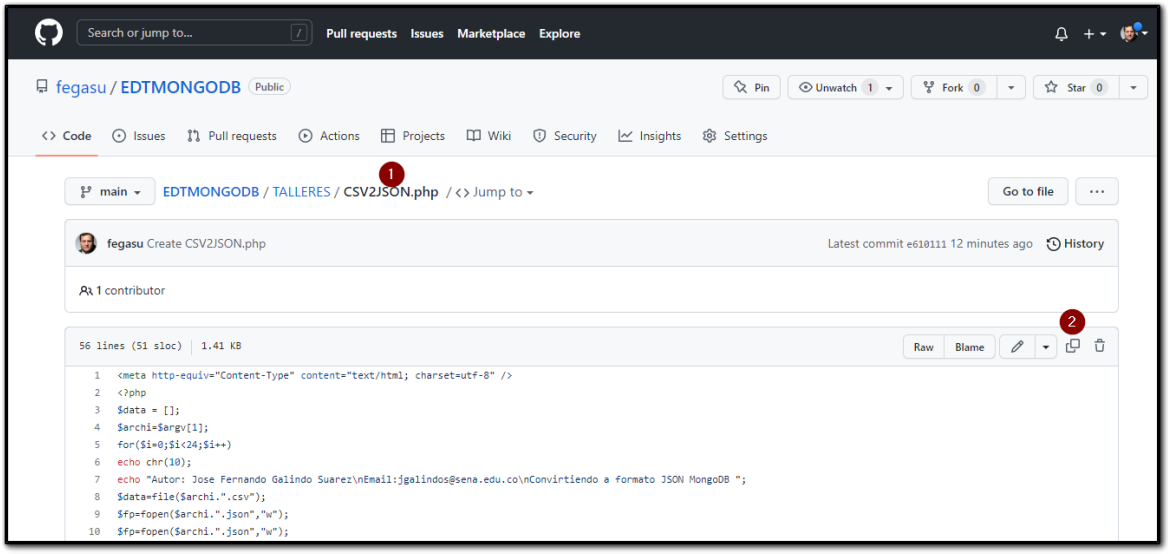
CONVERTIR EXCEL A JSON

Utilizar la página https://products.aspose.app/cells/es/conversion/excel-to-json

Ejercicio: Convertir el archivo "DATOS/NEPTUNO/Neptuno.xls" y migrarlo a una base de datos MONGODB, se recomienda ver el video "InsertManyNeptuno.mp4"

CONVIRTIENDO CSV A JSON

- **Se recomienda ver el video "csv2jsonMongoDB" para conocer el proceso.**
- **Descargar el archivo "COVID19-JULIO2020.zip" que se encuentra en "DATOS/COVID19/COVID19-JULIO2020.zip"**
- **Descomprima**
- **Crear un archivo PHP, llamado "CSV2JSON.php" con el siguiente código ubicado en: "TALLERES/CSV2JSON.php".**

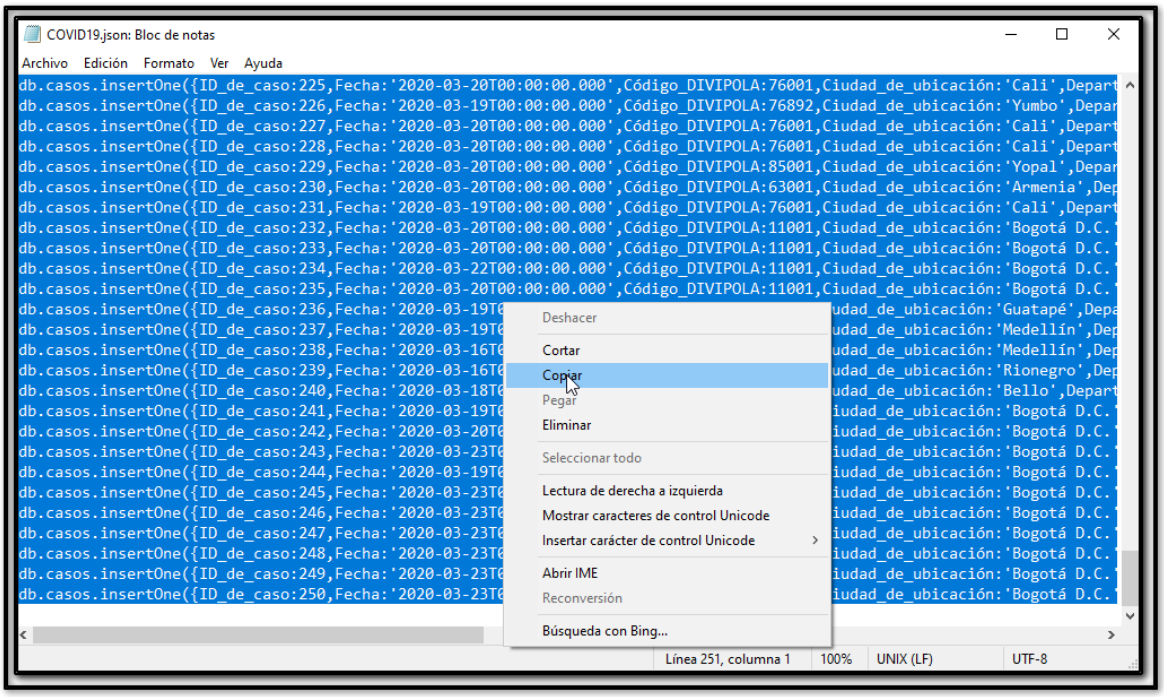


- **Ejecutar el programa anterior donde lo creo, desde la línea de comando así:**

```
php -f CSV2JSON.php COVID19
```

- **Creará un archivo llamado "COVID19.json" al terminar la ejecución.**
- **Ábralo, realizar una instrucción para insertar muchos documentos.**

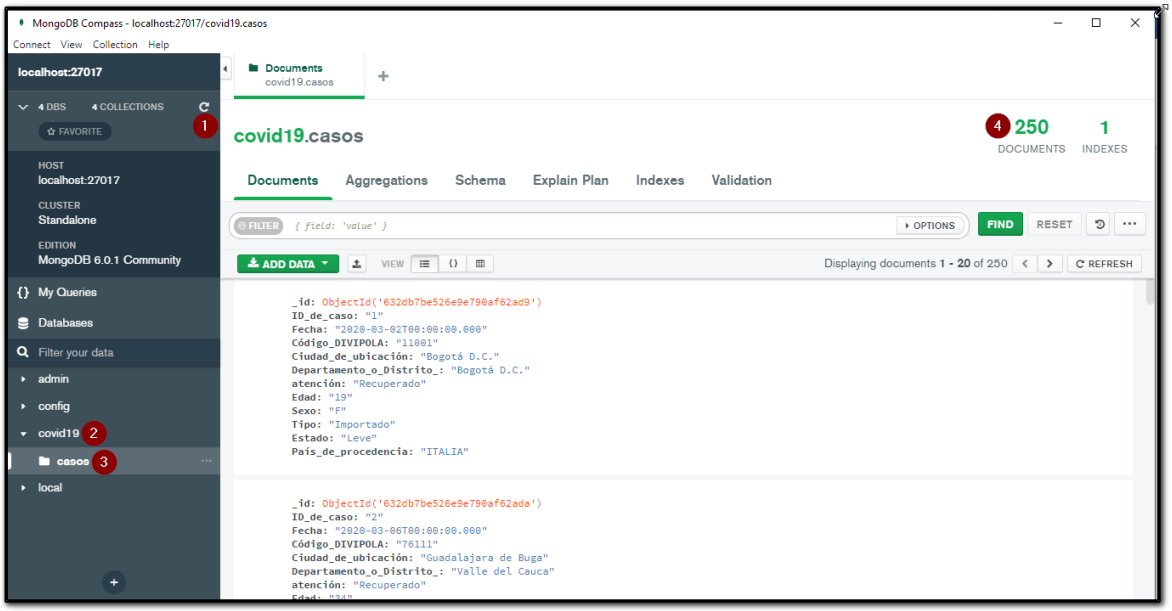
- **Copie al portapapeles todo el contenido.**



- **Abra el programa MONGOSH**
- **Elija la base de datos**

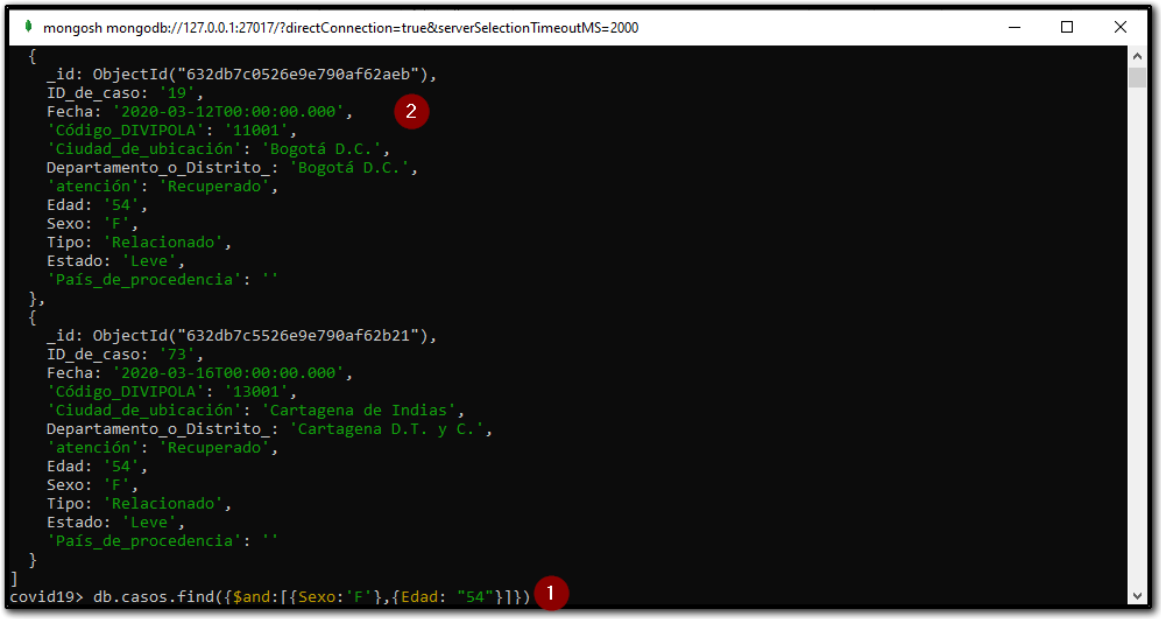
```
test> use covid19
```

- **Copiar del portapapeles, usando la tecla derecha del mouse.**
- **Ingrese al programa Mongo COMPASS y verifique el cargue**



- **Realizar las siguientes consultas para verificar y sacar registro gráfico.**

```
db.casos.find({$and:[{Sexo:'F'},{Edad: 54}]})
db.casos.find({Edad:{$lte:50}})
```



CONECTANDO DESDE PYTHON A MONGODB

Se necesita instalar la biblioteca pymongo así:

```
Pip install pymongo
```

Realizar el siguiente código en un archivo llamado: "MongoCliente.py"

```
from pymongo import MongoClient
MONGO_URI='mongodb://localhost'
cliente=MongoClient(MONGO_URI)
db=cliente['prueba']
coleccion=db['emp']
eps1=coleccion.find()
for r in eps1:
    print(r['idemp']+"-"+r['nom'])
```

Ejecute el programa

```
Python MongoCliente.py
```

CONECTANDO DESDE PHP A MONGO[iii]

- **Para descargar el controlador MongoDB para Windows, visita** pecl.php.net
- **Descargar el controlador** 7.3 Thread Safe (TS) x86**, este es para la versión de PHP 7.3.x y si se utiliza versión 8 o posterior utilice el drive correspondiente**
- **Descomprima el archivo descargado y copie los archivos "php_mongodb.dll" y "php_mongodb.pdb" en "C:\xampp\php\ext"**
- **Abrir el archivo "php.ini"**
- **Adicionar la línea:**

```
extension=php_mongodb.dll
```

- **Reinicie de nuevo el apache**
- **Verifique con el phpinfo (http://localhost/dashboard/phpinfo.php) que xampp soporta MongoDB**

mongodb

MongoDB support	enabled
MongoDB extension version	1.11.0
MongoDB extension stability	stable
libbson bundled version	1.19.1
libmongoc bundled version	1.19.1
libmongoc SSL	enabled
libmongoc SSL library	OpenSSL
libmongoc crypto	enabled
libmongoc crypto library	libcrypto
libmongoc crypto system profile	disabled
libmongoc SASL	enabled
libmongoc ICU	disabled
libmongoc compression	disabled
libmongocrypt bundled version	1.2.1
libmongocrypt crypto	enabled
libmongocrypt crypto library	libcrypto

Directive	Local Value	Master Value
mongodb.debug	*no value*	*no value*
mongodb.mock_service_id	Off	Off

- **Escriba un archivo llamado "cnxMongoDb.php" en c:\xampp\htdocs"**

```php
<?php
echo "_____<br>";
$manager = new \MongoDB\Driver\Manager('mongodb://localhost:27017');
$options = ['limit' => 1];
$filter = ['_id' => new \MongoDB\BSON\ObjectID('63222b6f4c5aac0c8e499a9c')];
$query = new \MongoDB\Driver\Query($filter, $options);
echo "_____<br>";
$cursor = $manager->executeQuery('prueba.emp', $query);
$cursorArray = $cursor->toArray();
if(isset($cursorArray[0])) {
    var_dump($cursorArray[0]);
}
echo "_____<br>";
```

- **Cambiar el ObjectId por uno _id válido**
- **Regresa un arreglo BSON**

RETO POR DESARROLLAR

- **Cargar los datos que se encuentran en "DATOS/COVID19/COVID19-JULIO2020.csv" en una base de datos MONGODB llamada "COVID", en la colección llamada "CASOS".**
- **Realizar las siguientes consultas:**
  - **Cuantos casos se registraron en Bogotá**
  - **Cuantos casos se recuperaron en Bogotá**
  - **Cual fue la ciudad que más se recuperaron**
  - **Cual fue la ciudad que más se recuperaron siendo mujeres.**
  - **Cuantos fallecidos que fueron mujeres y mayores de 50 años.**

  - **Cuantos fallecidos que fueron hombres y menores de 50 años.**
  - **Cuantos casos fueron por personas del extranjero**
  - **Cuantos casos se registraron en Antioquía, menores de 30 años y por persona extranjeras**
  - **Cuantos casos se registraron por personas de España**
  - **Cuantos casos se registraron por personas de edad entre 20 y 50 años**
- **Realizar un programa que muestre cada una de las consultadas solicitadas en el punto anterior.**

PRODUCTOS SOLICITADOS.

- **Documento de tipo texto con las consultas solicitadas**
- **Documento fuente del programa Python**
- **Informe grafico de desarrollo del taller.**
- **Ensayo sobre que son las bases de datos NOSQL y aplicaciones existentes**.

MODO DE ENTREGA.

Se debe empaquetar los productos solicitados en un archivo llamado "TallerNoSQL.zip"

Realizado por el instructor José Fernando Galindo Suarez
Centro de Gestión de Mercados, Logística y Tecnologías de la información
Jgalindos @sena.edu.co ®2022

[i] que es bson - Búsqueda (bing.com)
[ii] https://www.codehoven.com/mongodb-aggregate-operaciones-agregacion-pipeline/#:~:text=Para%20utilizar%20la%20tuber%C3%ADa%20de%20agregaci%C3%B3n%20de%20MongoDB%2C,un%20objeto%20que%20representa%20una%20operaci%C3%B3n%20a%20realizar.

[iii] PHP => Utilizando MongoDB (learntutorials.net)