



## Base de Datos II

**Integrantes:** Xavier Calle, Anderson Córdova, Gabriel Ibujés, Elena Pérez, Kevin Seegovia

**Fecha:** 30 – 01 – 2020

**Tema:** Mongo DB

### Instalación y configuración de Mongo DB:

- Nos descargamos MongoDB de la página oficial:  
<https://www.mongodb.com/es>
- Seleccionamos el SO, la versión Current (la más actual), y el tipo de paquete, en ese caso será el paquete MSI ya que es el instalable
- El archivo estará en Downloads, en Archivos de Programas.
- En Windows el servidor de Mongo se llama **mongod.exe** y el cliente o shell **mongo.exe**.
- Ejecutamos el instalador
- Aceptamos términos y condiciones
- Escogemos Custom para saber qué características del programa deseamos tener.
- Corroboramos los programas que necesitamos.
- Seleccionamos que MongoDB se ejecute como servicio (opcional).
- Finalizamos la instalación.
- Creamos una carpeta **data** y dentro de ella otra llamada: **db** en C:\ ya sea mediante interfaz gráfica o la shell.
- Ejecutamos en primer lugar **mongod.exe** para activar el servicio, y quede en segundo plano.
- Luego arrancamos la consola de mongo **mongo.exe**, para poder trabajar con la Base de Datos, dejando ya activo el servicio de Mongo

- Comprobamos que Mongo ya está funcionando insertando comandos en este caso  
: **show dbs** para que nos muestre las bases que nos da por defecto.

### Crear una Base de datos MongoDB

Para crear una base de datos en MongoDB debemos escribir el comando: use [estudiante]. Este comando nos permite seleccionar una base en concreto y si no existe la crea. Cabe recalcar que la base de datos creada no se muestra hasta que se interactúa con la misma.

### Insertar un documento

Dentro de la colección estudiantes, para insertar un documento utilizamos el siguiente comando:  
*dbo.estudiantes.insert({"universidad": "epn", "carreras":["esfot", "sistemas", "mecanica"]})*

La consola nos retorna un *WriteResult* con el número de inserciones.

### Consultar los documentos

Utilizamos el comando find() para encontrar todos los documentos de cada colección:

*Dbo.estudiantes.find()*. Podemos añadir el comando pretty() para mostrar una vista diferente. A cada objeto se le asigna un identificador único por defecto y podemos filtrar los resultados: *db-estudiantes.find({"universidad": "epn"})*.

### Actualizar un documento

Para actualizar un documento usamos el comando *update*. Podemos añadir

una carrera a la lista de modelos del documento existente en la colección;  
`dbo.estudiantes.update({"universidad": "epn"}, {$push: {"carreras": "petroleos"}}).`

```
> db.datos.find({nombre: 'Kevin'}).pretty()
{
  "_id" : ObjectId("5e323e4a682829c816cb94d0"),
  "nombre" : "Kevin",
  "apellido" : "Segovia",
  "edad" : "21"
}
```

### Eliminar un documento

Para borrar un documento usamos el comando `remove`. La consola nos retorna un `WriteResult` con el número de filas eliminadas;  
`dbo.estudiantes.remove({"universidad": "epn"})`

### Ejercicio fácil

Crear una base de datos, en la cual se creará una colección con dos documentos en los que se escribirá el nombre, el apellido y la edad. Además, buscar un documento por el campo nombre.

#### - Creación de la base de datos

Usar el comando “use nombre-base-datos”

```
> use base-datos-expo
switched to db base-datos
>
```

#### - Creación de la colección

Usar el comando “db.nombre-coleccion.save(campos) o a su vez db.nombre-coleccion.insert(campos)”

```
db.datos.save({nombre: 'Kevin',
apellido: 'Segovia',
edad: '21'})
WriteResult({ "nInserted" : 1 })
```

#### - Buscar documento

Usar el comando “db.nombre-coleccion.find(campo por el que se quiere buscar)”

### Ejercicio medio

Los desarrolladores de una aplicación web han contratado nuestros servicios para la captura y almacenamiento de las trazas dejadas por los usuarios de su aplicación. La información que obtenemos del usuario es un identificador de usuario, la URL que ha accedido el usuario, la aplicación a la que pertenece dicha URL (por ejemplo, cliente o gestor) y la fecha y hora en la que se ha producido la acción. Además de almacenar estos datos, se nos solicita una serie de informes respecto a ellos:

- Id: identificador usuario.
- URL: link
- Aplicación: ya sea sistema web o servicio rest, entre otros
- Fecha\_hora: momento en que se ejecuta la acción

#### - CREACION DE LA DB

```
mydb:trazasApp@localhost x
localhost mydb
1 use mydb
```

#### - CREACION DE COLLECTION

```
mydb:trazasApp@localhost x
localhost mydb
1
2 db.createCollection("trazasApp")
```

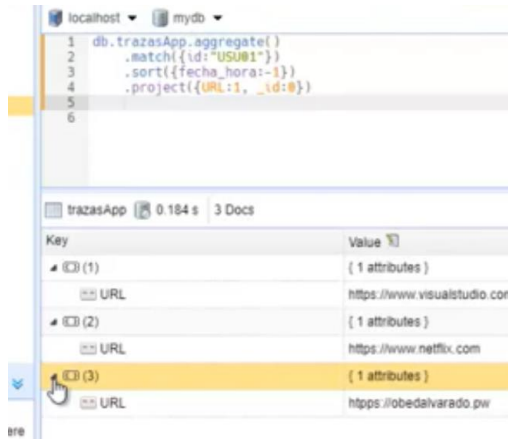
#### - INSERTAR DATOS

```
5 db.trazasApp.insert({id:"00001", URL:"https://www.quechafriends.org/qu/index.html", aplicacion: "Sistema web", fecha_hora: ISODate('')}
6 db.trazasApp.insert({id:"00002", URL:"https://www.melilla.com", aplicacion: "Sistema web", fecha_hora: ISODate('')}
7 db.trazasApp.insert({id:"00003", URL:"https://www.pacheco.com", aplicacion: "Sistema web", fecha_hora: ISODate('')}
8 db.trazasApp.insert({id:"00004", URL:"https://www.facebook.com", aplicacion: "Sistema web", fecha_hora: ISODate('')}
9 db.trazasApp.insert({id:"00005", URL:"https://www.instagram.com", aplicacion: "Sistema web", fecha_hora: ISODate('')}
10 db.trazasApp.insert({id:"00006", URL:"https://www.facebook.com", aplicacion: "Sistema web", fecha_hora: ISODate('')}
11 db.trazasApp.insert({id:"00007", URL:"https://www.instagram.com", aplicacion: "Sistema web", fecha_hora: ISODate('')}
12 db.trazasApp.insert({id:"00008", URL:"https://www.instagram.com", aplicacion: "Sistema web", fecha_hora: ISODate('')}
13
```

Key	Value T1	Type
00001	{ "id": "00001", "URL": "https://www.quechafriends.org/qu/index.html", "aplicacion": "Sistema web", "fecha_hora": "2018-12-01T10:00:00Z" }	Object
00002	{ "id": "00002", "URL": "https://www.melilla.com", "aplicacion": "Sistema web", "fecha_hora": "2018-12-01T10:00:00Z" }	Object

## - CONSULTAS

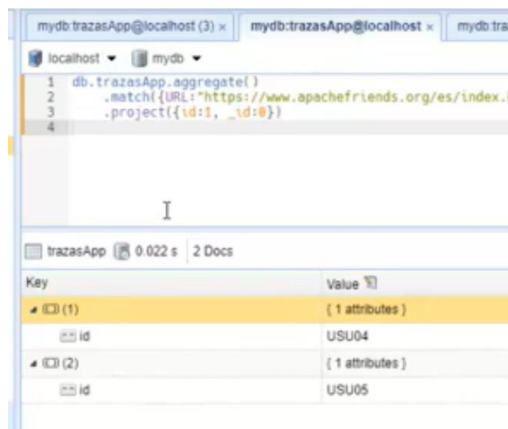
- 5.1.- listado de las URL, accedidas por un usuario en concreto, ordenado por fecha y hora de acceso.



```
1 db.trazasApp.aggregate()
2 .match({id:"USU01"})
3 .sort({fecha_hora:-1})
4 .project({URL:1, _id:0})
```

Key	Value	Type
(1)	{ 1 attributes }	Document
URL	https://www.visualstudio.com	Document
(2)	{ 1 attributes }	Document
URL	https://www.netflix.com	Document
(3)	{ 1 attributes }	Document
URL	https://obedivarado.pw	Document

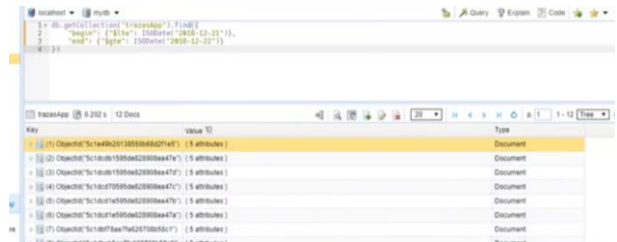
- 5.2.- Listado de todos los usuarios que han accedido a una URL del gestor



```
1 db.trazasApp.aggregate()
2 .match({URL:"https://www.apachefriends.org/es/index.h"})
3 .project({id:1, _id:0})
```

Key	Value	Type
(1)	{ 1 attributes }	Document
id	USU04	Document
(2)	{ 1 attributes }	Document
id	USU05	Document

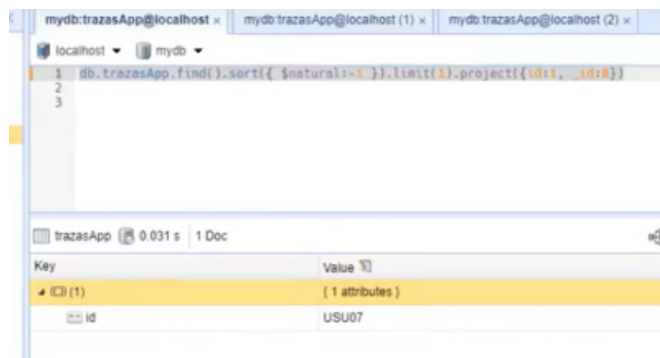
- 5.3.- Listado de todos los usuarios que han accedido a la aplicación en un rango de tiempo específico.



```
1 db.trazasApp.find().sort({ _id:-1 }).limit(1).project({id:1, _id:0})
```

Key	Value	Type
(1)	{ 1 attributes }	Document
id	USU07	Document

- 5.4.- Identificador del ultimo usuario que ha utilizado la aplicación.



```
1 db.trazasApp.find().sort({ _id:-1 }).limit(1).project({id:1, _id:0})
```

Key	Value	Type
(1)	{ 1 attributes }	Document
id	USU07	Document

- 5.5.- Listado de todas las URL almacenadas y el número de accesos que ha tenido cada una.



```
1 db.trazasApp.aggregate({
2   $group: {
3     _id: {
4       URL: "$URL"
5     },
6     count: {
7       $sum: 1
8     }
9   })
```

Key	Value	Type
(1) { URL: "https://www.instagram.com" }	{ 2 attributes }	Document
count	2	Document
(2) { URL: "https://www.facebook.com" }	{ 2 attributes }	Document
(3) { URL: "https://www.netflix.com" }	{ 2 attributes }	Document
(4) { URL: "https://www.apachefriends.org/es" }	{ 2 attributes }	Document
(5) { URL: "https://www.mongodb.com" }	{ 2 attributes }	Document



## Evaluación

- Se precisa diseñar un blog de noticias donde los usuarios registrados pueda publicar sus comentarios:
  - Cada autor tiene un nombre, un nombre de usuario, una cuenta de Twitter y una descripción. Además, de forma opcional, los usuarios pueden proporcionar como datos su dirección postal (calle, número, puerta, C.P., ciudad) o sus teléfonos de contacto (pueden tener varios).
  - Las noticias tienen un título, un cuerpo y una fecha de publicación. Son publicadas por un autor y pueden contener o no, una lista de tags.
  - Las noticias reciben comentarios, quedando registrado la persona que lo escribió, el comentario escrito y el momento en el que lo hizo.
- **Consultas**  
Consultas de los datos del usuario por nombre de usuario y por cuenta de Twitter:
  - Agrupación por código postal (contar el número de usuarios de cada C.P).
  - Consultas por número de teléfono.Consultas de noticias publicadas por usuarios
  - 10 últimas noticias publicadas ordenadas por fecha (de más reciente a más antigua).Número de comentarios por noticia, por día o por usuario.

- **Crear una colección**

- Recordemos:  
“no es necesario crear una

### *colección de forma explícita”*

- Se pueden insertar documentos y crear índices sobre campos sobre colecciones que aún no tengan documento alguno.
- En MongoDB, no hay esquemas.
  - En cada colección, los documentos pueden tener diferentes campos.
  - Control por parte del cliente: la aplicación que accede a la base de datos MongoDB es responsable de cómo evolucione el esquema.
- **Modelo físico de usuarios**
- Cuestiones a tener en cuenta en su “diseño” (por ahora nos olvidamos de que las noticias y los comentarios existen):
  - Teléfonos de contacto:
    - ¿Varios campos en un sub-documento?
    - ¿Array de teléfonos?.
  - Dirección postal con varios campos:
    - ¿Sub-documento?.