UT6 Bases de datos NoSQL . **EJERCICIOS**

SESIÓN 0

- Instalación gestor + cliente gráfico (MongoDb Compass).
- Instalación MongoDB Shell.
- Instalación (voluntaria) de la extensión "MongoDB for VS Code " para el Visual Studio Code.

SESIÓN 1

Para realizar estos ejercicios utilizar la base de datos test y crear la colección ciudades:
test> db.createCollection("ciudades")

e importar en ella los documentos que se encuentran en el fichero *"mongo_cities1000_valido.json"*

Escribe la operación necesaria y el resultado para averiguar:

1. Número de ciudades.
2. Datos de la ciudad de Elx.
3. Población de la ciudad de Vergel.
4. Cantidad de ciudades en España ({"country":"ES"}).
5. Datos de las ciudades españolas con más de 1.000.000 de habitantes.
6. Cantidad de ciudades de Andorra ({"country":"AD"}) y España.
7. Listado con el nombre y la población de las 10 ciudades más pobladas.
8. Nombre de las distintas zonas horarias en España.
9. Ciudades españolas que su zona horaria no sea Europe/Madrid.
10. Ciudades españolas que comiencen por Ben
11. Ciudades que su zona horaria sea Atlantic/Canary o Africa/Ceuta, y que tengan más de 500.000 habitantes.
12. Nombre y población de las tres ciudades europeas más pobladas.
13. Cantidad de ciudades españolas cuya coordenadas de longitud estén comprendidas entre -0.1 y 0.1.

Escribe la operación necesaria para:

14. Modifica la población de Huesca a 1.000.000.
15. Incrementa la población de Elx en 666 personas.
16. Reduce la cantidad de todas las ciudades de Andorra en 5 personas.
17. Modifica la ciudad de Gibraltar para que sea española (tanto el país como la zona horaria). Ten en cuenta que hay una ciudad americana con ese mismo nombre que no debe modificarse.
18. Modifica todas las ciudades y añade un atributo tags que contenga un array vacío.
19. Modifica todas las ciudades españolas y añade al atributo tags el valor sun.
20. Modifica el valor de sun de la ciudad A Coruña y sustitúyelo por rain.
21. Renombra en las ciudades de Andorra, el atributo population por poblacion.
22. Elimina las coordenadas de Gibraltar (el "español", no el americano).
23. Elimina la población de Huesca

---

24. Crea un proyecto nuevo de Java y los siguientes procedimientos dentro de una clase que contenga el método *main()*. Utiliza la clase *Ciudad.java* que encontrarás en los materiales

a. *private static boolean insertaCiudad(Ciudad ciudad)* para insertar la ciudad que se le pase como argumento. El método devolverá true/false según se haya podido realizar la operación o no.

b. *private static void listarCiudades()* para listar el nombre de todas las ciudades:

```
Sant Julià de Lòria
Pas de la Casa
……
Epworth
Chitungwiza
```

c. *private static void listarCiudadesPais(String pais)* para listar el nombre de todas las ciudades del país que se le pasa como argumento ordenadas albabéticamente. Ej. `listarCiudadesPais("ES"):`

```
A Coruña
A Estrada
Abadín
...
...
Órjiva
Úbeda
```

d. *private static void listarPaises()* para listar todos los países ordenados alfabéticamente:

```
AD
AE
…
ZM
ZW
```

SESIÓN 2

25.- En este ejercicio se va a optimizar la base de datos utilizada en sesiones anteriores. Las consultas que más se realizan sobre la colección ciudades son:

a. Recuperar una ciudad por su nombre.
b. Recuperar las 5 ciudades más pobladas de un determinado país.
c. Recuperar las 3 ciudades menos pobladas de una zona horaria.

Se pide crear los índices adecuados para que estas consultas se ejecuten de manera óptima. Anota y guarda los comandos empleados para comprobar los planes de ejecución **antes y después** de crear los índices necesarios para analizar si ha merecido la pena su utilización. Anota tus conclusiones y guarda también los comandos necesarios para crear los índices elegidos.

SESIÓN 3

Utilizando la colección ciudades de ejercicios anteriores, escribe los comandos para obtener la información de las siguientes consultas mediante el pipeline de agregación:

26.- Nombre y población de las tres ciudades españolas con más habitantes. Resultado:

```
[
  { nombre: 'Madrid', poblacion: 3255944 },
  { nombre: 'Barcelona', poblacion: 1621537 },
  { nombre: 'Valencia', poblacion: 814208 }
]
```

27.- País, población y cantidad de ciudades de dicho país, ordenados de mayor a menor población. Resultado:

```
[
  { 'población': 282839031, ciudades: 5568, 'país': 'CN' },
  { 'población': 272149640, ciudades: 3350, 'país': 'IN' },
  …
 { 'población': 99, ciudades: 1, 'país': 'GS' },
  { 'población': 46, ciudades: 1, 'país': 'PN' }
]
```

28.- País, ratio entendido como el resultado de dividir la población del país entre el número de ciudades de dicho país, ordenados por el ratio de población/ciudades:

```
  { 'país': 'SG', ratio: 3547809 },
  { 'país': 'HK', ratio: 2260092.75 },
  { 'país': 'MO', ratio: 520400 },
  { 'país': 'TW', ratio: 452805.67741935485 },
….
```

29.- Codifica una clase con su método *main()* para realizar la consulta del ejercicio 26 desde Java. Resultado:

```
Document{{nombre=Madrid, poblacion=3255944}}
Document{{nombre=Barcelona, poblacion=1621537}}
Document{{nombre=Valencia, poblacion=814208}}
```