Proyecto Final 1 evaluación Mongo DB

Alumno: María Prados

En este proyecto vamos a crear una base de datos con inserciones que incluyan campos de todos los tipos y consultas que utilicen todos los operadores vistos en clase aplicados a la práctica.

Mi base de datos trata sobre los menú de una Pizzeria, una colección con elementos simples y bien conocidos, que da pie a realizar las consultas oportunas.

Partimos de mongo DB ya instalado desde la practica anterior.

Lo primero es escribir en la consola **mongo** para activar los comandos de mongo.

Escribimos **use pizza** para crear y usar nuestra nueva colección.

```
---
> use pizza
switched to db pizza
>
```

Para editar los textos, he utilizado Visual Code y Sublime Text.

Estructura de la colección

Han sido un total de 15 documentos insertados con la siguiente estructura:

```
db.pizza.insertOne(
                    {
                        _id: 1,
                        pizza: "Desierta",
                        ingredientes:   [
                                            "tomate",
                                            "mozzarella"
                                                ],
                        tamaño: [
                                        "pequeña",
                                        "mediana",
                                        "familiar"
                                        ],
                        precio:
                                        {
                                            pequeña: 4.90,
                                            mediana: 6.90,
                                            familiar: 9.90},
                        complementos: {
                                extra_queso: false,
                                masa_fina: true,
                                salsa_texmex: false,
                                bordes_rellenos: false
                                            },
                    opiniones:[
                                        {
                                            hombres:{
                                                    valoración: 7.9, favorita:false},
                                        mujeres: {valoración: 7.3, favorita:false},
                                        niños: {valoración: 9, favorita:true},
                                        ancianos: {valoración: 9.5, favorita:true}
                                        }
                                    ],
                    numero_porciones: [1,2,4,6,8],
                    ultimo_pedido: new Date ("2021-11-01")
                    }
                )
```

Incluye varios campos de tipo string, arrays (string y numéricos), campos tipo documento, campo fecha y array de documentos.

Introducimos la colección utilizando los comandos:

db.pizza.drop({})
db.pizza.insertMany([])

Utilizamos el drop para borrar toda la colección cada vez que insertamos elementos nuevos para evitar duplicados.

He separado las consultas en 2 archivos js, uno para las simples y otro para las compuestas, donde he incluido ejemplos aplicados con cada una de ellas y unos supuestos aplicados.

Consultas simples: Una búsqueda con cada uno de los operadores con un enunciado que explica lo que hace cada consulta:

```
/*CONSULTAS SIMPLES*/

/*por campo*/

db.pizza.find({pizza:"Llorona"})
db.pizza.find({_id:3})

/*por campo dentro de otro*/

db.pizza.find({"precio.pequeña":{$lte:8}}).pretty()
db.pizza.find({"precio.familiar":{$gte:12}})

/*por fecha*/

db.pizza.find({ultimo_pedido: {$gt: new Date ("2021-11-10")}})
db.pizza.find({ultimo_pedido: {$lt: new Date ("2021-11-15")}})

/*Por elemento exacto dentro de un array*/

db.pizza.find({ingredientes: {$elemMatch:{$eq:"jalapeños"}}})
db.pizza.find({ingredientes: {$elemMatch:{$ne:"cheddar"}}})
db.pizza.find({tamaño: {$elemMatch: {$eq:"familiar"}}})

/*Por caracteres especiales*/

db.pizza.find({pizza: {$regex:/^M/i}})

db.pizza.find({pizza: {$regex:/a$/i}})
```

```
/*Mi amigo es vegetariano, encuentra todas las pizzas que no lleven
nada de carne ni pescado, usando el comando $Nin*/

db.pizza.find({ingredientes: {$nin: ["bacon", "Jamon York", "chorizo", "jamon serrano",
                                     "ternera", "pepperoni", "carne picada", "salsa barbacoa"]}})

/*Contener elementos, no excluyente*/

db.pizza.find({ingredientes: {$in: ["cebolla", "guacamole", "espinacas"]}})

/*Buscar las pizzas que tengan los campos existentes*/

db.pizza.find({"complementos.extra_queso": {$exists:true}})
db.pizza.find({"complementos.masa_fina": {$exists:false}})

/*Buscar las pizzas que tengas esos complementos disponibles o no*/

db.pizza.find({"complementos.extra_queso":{$eq:true}}).pretty()
db.pizza.find({"complementos.masa_fina":{$eq:false}}).pretty()

/*Negar un condicionante*/
db.pizza.find({ingredientes: {$not: {$eq:"ternera"}}})
db.pizza.find({"precio.pequeña":{$not: {$lte:7}}})
```

{

He añadido el operador **$te**xt, no visto en clase, para aportar algo adicional, es un operador simple pero muy util para buscar dentro de las búsquedas, cualquier información independientemente del campo en el que esté, tiene varias opciones de búsqueda, yo lo he utilizado para busquedas de tipo string.

Tiene una particularidad y es que hay que indexar primero el campo en el que se va a buscar para poder realizar la consulta.

```
/*Operador Text para buscar cualquier texto*/

db.pizza.createIndex({ingredientes:"text"})

db.pizza.find({$text:{$search:"champiñones"}})
```

Consultas avanzadas: En este apartado he metido aquellas consultas en las que interviene más de un documneto, como **$and, $or, $nor,** así como algunos supuestos con las búsquedas combinadas.

```
/*Encontrar las que cumplan más de una condición*/

db.pizza.find({$and: [{pizza: {$ne:"Iberica"}},
                      {ingredientes: {$elemMatch: {$eq:"guacamole"}}}]})

/*Encontrar pizzas que no tengan la opcion de extra de queso, O que si tengan texmex*/

db.pizza.find({$or: [{"complementos.extra_queso": {$ne:true}},
                     {"complementos.salsa_texmex":{$ne:false}}]}).pretty()

/*Que no se cumplan una o las dos condiciones*/
db.pizza.find({$nor: [{"precio.familiar": {$gt: 15}},
                      {"complementos.bordes_rellenos": {$eq:true}}]})
```

```
/*Carlos ha descubierto, una promocion especial por el dia de su cumpleaños, las pizzas que empiecen por su inicial tienen 5 dolares de descuento,
quiere pedir una con queso cheddar y que tenga salsa texmex como complemento*/

db.pizza.find({$and: [
          {pizza: {$regex:/^C/i}},
          {ingredientes: {$elemMatch:{$eq:"cheddar"}}},
          {"complementos.salsa_texmex":{$eq:true}}]})


/*Pizzas pedidas entre el 8 de noviembre y el 14 de noviembre, que contengan chorizo, y un precio inferior a 10 para el tamaño mediano*/
db.pizza.find({$and: [
              {ultimo_pedido:{$gte: new Date ("2021-11-08")}},
              {ultimo_pedido:{$lte: new Date ("2021-11-14")}},
              {ingredientes: {$in: ["chorizo"]}},
              {"precio.mediana": {$lt:10}}]})


/*A silvia le toca este fin de semana quedarse con su hijo, ha decidido llevarselo a una pizzeria por primera vez y no sabe lo que le puede gustar,
pide las recomendaciones segun las estadísticas del restaurante*/

db.pizza.find({opiniones: {$all: [{"$elemMatch":{"niños.valoración":{$gt:9}}},
                                  {"$elemMatch":{"mujeres.favorita":true}}]
                           }})

/*Voy a llevarme a comer a mi jefe, para recibir un ascenso, la cuenta la paga el, buscame las pizzas mas valoradas y las mas caras*/

db.pizza.find({$and: [{opiniones: {$all: [{"$elemMatch":{"hombres.valoración":{$gt:9}}}]}},
                      {"precio.mediana":{$gte:10}}]}).sort({"precio.mediana":-1}).pretty()
```

En este apartado también he incluido algo no visto en clase, el metodo **.sort()** que sirve para ordenar los resultados de la busqueda en orden ascendente o descendente.