MONGODB

SESIÓN 1

Escribe la operación necesaria y el resultado para averiguar:

1. Número de ciudades.

```
test> db.ciudades.find().count("$name")
99838
```

2. Datos de la ciudad de Elx.

```
test> db.ciudades.findOne({name:"Elx"})
{
  _id: ObjectId("63d2b0d8da2d5774158ceeb7"),
  name: 'Elx',
  country: 'ES',
  timezone: 'Europe/Madrid',
  population: 230112,
  location: { latitude: 38.26218, longitude: -0.70107 }
}
```

3. Población de la ciudad de Vergel.

```
test> db.ciudades.find({name:"Vergel"}).projection({"poblacion":"$population"})
[ { _id: ObjectId("63d2b0d8da2d5774158cebab"), poblacion: 4372 } ]
```

```
test> db.ciudades.findOne({name:"Vergel"}).population
4372
```

4. Cantidad de ciudades en España ({"country":"ES"}).

```
test> db.ciudades.find({"country":"ES"}).count()
2724
```

5. Datos de las ciudades españolas con más de 1.000.000 de habitantes.

```
test> db.ciudades.find({"country":"ES","population":{$gt:1000000}})
[
  {
    _id: ObjectId("63d2b0d8da2d5774158cf312"),
    name: 'Madrid',
    country: 'ES',
    timezone: 'Europe/Madrid',
    population: 3255944,
    location: { latitude: 40.4165, longitude: -3.70256 }
  },
  {
    _id: ObjectId("63d2b0d8da2d5774158cf537"),
    name: 'Barcelona',
    country: 'ES',
    timezone: 'Europe/Madrid',
    population: 1621537,
    location: { longitude: 41.38879, latitude: 2.15899 }
  }
]
```

6. Cantidad de ciudades de Andorra ({"country":"AD"}) y España.

```
test> db.ciudades.aggregate([{$match:{country:{$in:["AD","ES"]}}},{$group:{_id:"$country",count:{$sum:1}}},{$proj
ect:{_id:0 ,pais:"$_id" ,count:"$count"}}])
[ { pais: 'ES', count: 2724 }, { pais: 'AD', count: 9 } ]
```

7. Listado con el nombre y la población de las 10 ciudades más pobladas.

```
test> db.ciudades.find({},{_id:0,name:1,population:1}).sort({"population":-1}).limit(10)
[
  { name: 'Shanghai', population: 14608512 },
  { name: 'Buenos Aires', population: 13076300 },
  { name: 'Mumbai', population: 12691836 },
  { name: 'Mexico City', population: 12294193 },
  { name: 'Karachi', population: 11624219 },
  { name: 'İstanbul', population: 11174257 },
  { name: 'Delhi', population: 10927986 },
  { name: 'Manila', population: 10444527 },
  { name: 'Moscow', population: 10381222 },
  { name: 'Dhaka', population: 10356500 }
]
```

8. Nombre de las distintas zonas horarias en España.

```
test> db.ciudades.aggregate({$match:{"country":"ES"}},{$group:{_id:"$timezone"}},{$project:{_id:0,"Zona horaria:":"$_id"}})
[
  { 'Zona horaria:': 'Europe/Madrid' },
  { 'Zona horaria:': 'Atlantic/Canary' },
  { 'Zona horaria:': 'Africa/Ceuta' }
]
```

```
test> db.ciudades.distinct("timezone",{country:"ES"})
[ 'Africa/Ceuta', 'Atlantic/Canary', 'Europe/Madrid' ]
```

9. Ciudades españolas que su zona horaria no sea Europe/Madrid.

```
test> db.ciudades.distinct("name",{country:"ES",$nor:[{timezone:"Europe/Madrid"}]})
[
```

```
test> db.ciudades.aggregate([{$match:{country:"ES"}},{$match:{timezone:{$ne:"Europe/Madrid"}}}])
[
```

10. Ciudades españolas que comiencen por Ben

```
test> db.ciudades.find({country:"ES",name:/^Ben/})
[
```

11. Ciudades que su zona horaria sea Atlantic/Canary o Africa/Ceuta, y que tengan más de 500.000 habitantes.

```
test> db.ciudades.find({"timezone":{$in:["Atlantic/Canary","Africa/Ceuta"]},"population":{$gt:500000}})
```

12. Nombre y población de las tres ciudades europeas más pobladas.

```
test> db.ciudades.find({"timezone":/^Europe/}).sort({"population":-1}).limit(3).projection({_id:0, Nombre:"$name"
,Poblacion:"$population"})
[
  { Nombre: 'İstanbul', Poblacion: 11174257 },
  { Nombre: 'Moscow', Poblacion: 10381222 },
  { Nombre: 'City of London', Poblacion: 7556900 }
]
```

13. Cantidad de ciudades españolas cuya coordenadas de longitud estén comprendidas entre -0.1 y 0.1.

```
test> db.ciudades.find({"country":"ES",$and:[{"location.longitude":{$gt:-0.1}},{"location.longitude":{$lt:0.1}}]}
)
```

14. Modifica la población de Huesca a 1.000.000.

```
test> db.ciudades.updateOne({"name":"Huesca"},{$set:{population:1000000}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
test> db.ciudades.find({"name":"Huesca"})
[
  {
    _id: ObjectId("63d2b0d8da2d5774158cf388"),
    name: 'Huesca',
    country: 'ES',
    timezone: 'Europe/Madrid',
    population: 1000000,
    location: { latitude: 42.13615, longitude: -0.4087 }
  }
]
test>
```

15. Incrementa la población de Elx en 666 personas.

```
test> db.ciudades.find({"name":"Elx"})
[
  {
    _id: ObjectId("63d2b0d8da2d5774158ceeb7"),
    name: 'Elx',
    country: 'ES',
    timezone: 'Europe/Madrid',
    population: 230112,
    location: { latitude: 38.26218, longitude: -0.70107 }
  }
]
test> db.ciudades.updateOne({"name":"Elx"},{$inc:{population:666}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
test> db.ciudades.find({"name":"Elx"})
[
  {
    _id: ObjectId("63d2b0d8da2d5774158ceeb7"),
    name: 'Elx',
    country: 'ES',
    timezone: 'Europe/Madrid',
    population: 230778,
    location: { latitude: 38.26218, longitude: -0.70107 }
  }
]
```

16. Reduce la cantidad de todas las ciudades de Andorra en 5 personas.

```
test> db.ciudades.update({"country":"AD"},{$inc:{population:-5}},{multi:true})
```

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.

```
test> db.ciudades.findAndModify({query:{"name":"ES","timezone":"Europe/Madrid"},update:{name:"Gibraltar",country:
"ES",timezone:"Europe/Madrid",population:30000},new:true})
{
  _id: ObjectId("63d2b0d8da2d5774158d2483"),
  name: 'Gibraltar',
  country: 'ES',
  timezone: 'Europe/Madrid',
  population: 30000
}
```

18. Modifica todas las ciudades y añade un atributo tags que contenga un array vacío.

```
Type "it" for more
test> db.ciudades.update({},{$push:{tags:""}},{multi:true})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 99839,
  modifiedCount: 99839,
  upsertedCount: 0
}
```

19. Modifica todas las ciudades españolas y añade al atributo tags el valor sun.

```
test> db.ciudades.update({country:"ES"},{$set:{tags:"sun"}},{multi:true})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 2724,
  modifiedCount: 2724,
  upsertedCount: 0
}
```

20. Modifica el valor de sun de la ciudad A Coruña y sustituyelo por rain.

```
test> db.ciudades.update({name:"A Coruña"},{$set:{tags:"rain"}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
test> db.ciudades.find({name:"A Coruña"})
[
  {
    _id: ObjectId("63d2b0d8da2d5774158cf369"),
    name: 'A Coruña',
    country: 'ES',
    timezone: 'Europe/Madrid',
    population: 246056,
    location: { latitude: 43.37135, longitude: -8.396 },
    tags: 'rain'
  }
]
```

21. Renombra en las ciudades de Andorra, el atributo population por población.

```
test> db.ciudades.update({country:"AD"},{$rename:{"Población":'Poblacion'}},{multi:true})
```

22. Elimina las coordenadas de Gibraltar (el "español", no el americano).

```
test> db.ciudades.update({name:"Gibraltar",timezone:"Europe/Madrid"},{$unset:{tags:1}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
test> db.ciudades.find({name:"Gibraltar",timezone:"Europe/Madrid"})
[
  {
    _id: ObjectId("63d2b0d8da2d5774158d2483"),
    name: 'Gibraltar',
    country: 'ES',
    timezone: 'Europe/Madrid',
    population: 30000
  }
```

23. Elimina la población de Huesca

```
test> db.ciudades.drop({name:"Huesca"})
true
test> db.ciudades.find({name:"Huesca"})

test>
```

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.

```
private static boolean insertaCiudad(Ciudad ciudad) {

try {
    Document d = new Document("name",ciudad.getName())
            .append("country",ciudad.getCountry())
            .append("timezone",ciudad.getTimezone())
            .append("location",ciudad.getLongitude())
            .append("latitude",ciudad.getLatitude())
            .append("Poblacion",ciudad.getPopulation());


    collection.insertOne(d);
    return true;

} catch (Exception e) {
    return false;
}
```

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

Sant Julià de Lòria
Pas de la Casa
.....
Epworth
Chitungwiza

```
private static void listarCiudades() {

    ArrayList<Document> lista = new ArrayList<>();
    Document documento1 = new Document("$project",new Document("nombre","$name")
                                .append("_id", 0));

    lista.add(documento1);

    AggregateIterable<Document> imprimir = collection.aggregate(lista);
    Iterator it = imprimir.iterator();
    while(it.hasNext()) {
        System.out.println(it.next());
    }
    mongo.close();
}
```

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

```
private static void listarCiudadesPais(String pais) {

    ArrayList<Document> lista = new ArrayList<>();
    Document d = new Document("$match",new Document("country",pais));
    Document d1 = new Document("$project",new Document("_id",0).append("ciudad","$name"));

    lista.add(d);
    lista.add(d1);

    AggregateIterable<Document> aggregate = collection.aggregate(lista);
    Iterator it = aggregate.iterator();
    while(it.hasNext()) {
        System.out.println(it.next());
    }
    mongo.close();
}
```

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

AD
AE
...
ZM
ZW

```
private static void listarPaises() {

    ArrayList<Document> lista = new ArrayList<>();
    Document d = new Document("$group",new Document("_id","$country"));
    Document d1 = new Document("$project",new Document("_id",1));
    Document d2 = new Document("$sort",new Document("_id",1));
    lista.add(d);
    lista.add(d1);
    lista.add(d2);

    AggregateIterable<Document> ag = collection.aggregate(lista);
    Iterator it = ag.iterator();

    while (it.hasNext()) {
        System.out.println(it.next());
    }


}
```

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.

ANTES	DESPUÉS

db.ciudades.find({name:"Huesca"}).explain("executionStats")
{
  explainVersion: '1',
  queryPlanner: {
  namespace: 'test.ciudades',
  indexFilterSet: false,
  parsedQuery: { name: { '$eq': 'Huesca' }
},
  maxIndexedOrSolutionsReached: false,
  maxIndexedAndSolutionsReached: false,
  maxScansToExplodeReached: false,
  winningPlan: {
  stage: 'COLLSCAN',
  filter: { name: { '$eq': 'Huesca' } },
  direction: 'forward'
  },
  rejectedPlans: []
  },
  executionStats: {
  executionSuccess: true,
  nReturned: 1,
  executionTimeMillis: 33,
  totalKeysExamined: 0,
  totalDocsExamined: 99838,
  executionStages: {
  stage: 'COLLSCAN',
  filter: { name: { '$eq': 'Huesca' } },
  nReturned: 1,
  executionTimeMillisEstimate: 0,
  works: 99840,
  advanced: 1,
  needTime: 99838,
  needYield: 0,
  saveState: 99,
  restoreState: 99,
  isEOF: 1,
	db.ciudades.createIndex({name:-1})
	
	db.ciudades.find({name:"Huesca"}).explain("executionStats")
	{
	  explainVersion: '1',
	  queryPlanner: {
	  namespace: 'test.ciudades',
	  indexFilterSet: false,
	  parsedQuery: { name: { '$eq': 'Huesca' }
	},
	  maxIndexedOrSolutionsReached: false,
	  maxIndexedAndSolutionsReached: false,
	  maxScansToExplodeReached: false,
	  winningPlan: {
	  stage: 'FETCH',
	  inputStage: {
	  stage: 'IXSCAN',
	  keyPattern: { name: -1 },
	  indexName: 'name_-1',
	  isMultiKey: false,
	  multiKeyPaths: { name: [] },
	  isUnique: false,
	  isSparse: false,
	  isPartial: false,
	  indexVersion: 2,
	  direction: 'forward',
	  indexBounds: { name: ['["Huesca", "Huesca"]'] }
	  }
	  },
	  rejectedPlans: []
	  },
	  executionStats: {
	  executionSuccess: true,
	  nReturned: 1,
	  executionTimeMillis: 2,

```
    direction: 'forward',
    docsExamined: 99838
   }
  },
  command: { find: 'ciudades', filter: { name:
'Huesca' }, '$db': 'test' },
  serverInfo: {
   host: 'DESKTOP-6Q7JQ8O',
   port: 27017,
   version: '5.0.5',
   gitVersion:
'd65fd89df3fc039b5c55933c0f71d647a5451
0ae'
  },
```

```
   totalKeysExamined: 1,
   totalDocsExamined: 1,
   executionStages: {
    stage: 'FETCH',
    nReturned: 1,
    executionTimeMillisEstimate: 0,
    works: 2,
    advanced: 1,
    needTime: 0,
    needYield: 0,
    saveState: 0,
    restoreState: 0,
    isEOF: 1,
    docsExamined: 1,
    alreadyHasObj: 0,
```

```
db.ciudades.find({country:"ES"}).sort({popul
ation:-1}).limit(5).explain("executionStats")
{
  explainVersion: '1',
  queryPlanner: {
   namespace: 'test.ciudades',
   indexFilterSet: false,
   parsedQuery: { country: { '$eq': 'ES' } },
   maxIndexedOrSolutionsReached: false,
   maxIndexedAndSolutionsReached: false,
   maxScansToExplodeReached: false,
   winningPlan: {
    stage: 'SORT',
    sortPattern: { population: -1 },
    memLimit: 104857600,
    limitAmount: 5,
    type: 'simple',
    inputStage: {
     stage: 'COLLSCAN',
     filter: { country: { '$eq': 'ES' } },
     direction: 'forward'
    }
   },
   rejectedPlans: []
  },
  executionStats: {
   executionSuccess: true,
   nReturned: 5,
   executionTimeMillis: 30,
   totalKeysExamined: 0,
   totalDocsExamined: 99838,
   executionStages: {
    stage: 'SORT',
    nReturned: 5,
    executionTimeMillisEstimate: 1,
    works: 99846,
```

**db.ciudades.createIndex({population:-1},
{partialFilterExpression:{country:"ES"}})**

```
db.ciudades.find({country:"ES"}).sort({popul
ation:-1}).limit(5).explain("executionStats")
{
  explainVersion: '1',
  queryPlanner: {
   namespace: 'test.ciudades',
   indexFilterSet: false,
   parsedQuery: { country: { '$eq': 'ES' } },
   maxIndexedOrSolutionsReached: false,
   maxIndexedAndSolutionsReached: false,
   maxScansToExplodeReached: false,
   winningPlan: {
    stage: 'LIMIT',
    limitAmount: 5,
    inputStage: {
     stage: 'FETCH',
     filter: { country: { '$eq': 'ES' } },
     inputStage: {
      stage: 'IXSCAN',
      keyPattern: { population: -1 },
      indexName: 'population_-1',
      isMultiKey: false,
      multiKeyPaths: { population: [] },
      isUnique: false,
      isSparse: false,
      isPartial: true,
      indexVersion: 2,
      direction: 'forward',
      indexBounds: { population: [
'[MaxKey, MinKey]' ] }
     }
    }
   },
```

```
    advanced: 5,
    needTime: 99840,
    needYield: 0,
    saveState: 99,
    restoreState: 99,
    isEOF: 1,
    sortPattern: { population: -1 },
    memLimit: 104857600,
    limitAmount: 5,
    type: 'simple',
    totalDataSizeSorted: 7286,
    usedDisk: false,
    inputStage: {
      stage: 'COLLSCAN',
```

```
    rejectedPlans: []
  },
  executionStats: {
    executionSuccess: true,
    nReturned: 5,
    executionTimeMillis: 7,
    totalKeysExamined: 5,
    totalDocsExamined: 5,
    executionStages: {
      stage: 'LIMIT',
      nReturned: 5,
      executionTimeMillisEstimate: 0,
      works: 6,
      advanced: 5,
      needTime: 0,
```

```
db.ciudades.find({country:"ES"}).sort({population:1}).limit(5).explain("executionStats")
{
  explainVersion: '1',
  queryPlanner: {
    namespace: 'test.ciudades',
    indexFilterSet: false,
    parsedQuery: { country: { '$eq': 'ES' } },
    maxIndexedOrSolutionsReached: false,
    maxIndexedAndSolutionsReached: false,
    maxScansToExplodeReached: false,
    winningPlan: {
      stage: 'SORT',
      sortPattern: { population: 1 },
      memLimit: 104857600,
      limitAmount: 5,
      type: 'simple',
      inputStage: {
        stage: 'COLLSCAN',
        filter: { country: { '$eq': 'ES' } },
        direction: 'forward'
      }
    },
    rejectedPlans: []
  },
  executionStats: {
    executionSuccess: true,
    nReturned: 5,
    executionTimeMillis: 47,
    totalKeysExamined: 0,
    totalDocsExamined: 99839,
    executionStages: {
      stage: 'SORT',
      nReturned: 5,
      executionTimeMillisEstimate: 2,
      works: 99847,
```

db.ciudades.createIndex({population:1})

```
db.ciudades.find({}).sort({population:1}).limit(3).explain("executionStats")
{
  explainVersion: '1',
  queryPlanner: {
    namespace: 'test.ciudades',
    indexFilterSet: false,
    parsedQuery: {},
    maxIndexedOrSolutionsReached: false,
    maxIndexedAndSolutionsReached: false,
    maxScansToExplodeReached: false,
    winningPlan: {
      stage: 'LIMIT',
      limitAmount: 3,
      inputStage: {
        stage: 'FETCH',
        inputStage: {
          stage: 'IXSCAN',
          keyPattern: { population: 1 },
          indexName: 'population_1',
          isMultiKey: false,
          multiKeyPaths: { population: [] },
          isUnique: false,
          isSparse: false,
          isPartial: false,
          indexVersion: 2,
          direction: 'forward',
          indexBounds: { population: [ '[MinKey, MaxKey]' ] }
        }
      }
    },
    rejectedPlans: []
  },
```

advanced: 5,
needTime: 99841,
needYield: 0,
saveState: 99,
restoreState: 99,
isEOF: 1,
sortPattern: { population: 1 },
memLimit: 104857600,
limitAmount: 5,
type: 'simple',
totalDataSizeSorted: 6932,
usedDisk: false,
inputStage: {
stage: 'COLLSCAN',
	executionStats: {
	executionSuccess: true,
	nReturned: 3,
	executionTimeMillis: 0,
	totalKeysExamined: 3,
	totalDocsExamined: 3,
	executionStages: {
	stage: 'LIMIT',
	nReturned: 3,
	executionTimeMillisEstimate: 0,
	works: 4,
	advanced: 3,
	needTime: 0,
	needYield: 0,
	saveState: 0,
	restoreState: 0,

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 }
]

```
test> db.ciudades.aggregate([{$match:{country:"ES"}},{$sort:{population:-1}},{$limit:3},{$project:{_id:0,nombre:"$name",poblacion:"$population"}}])
[
  { 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' }
]

```
test> db.ciudades.aggregate([{$group:{_id:"$country",ciudades:{$sum:1},poblacion:{$sum:"$population"}}},{$sort:{poblacion:-1}},{$project:{_id:0,poblacion:"$poblacion",ciud
ades:"$ciudades",pais:"$_id"}}])
```

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 },
....

```
test> db.ciudades.aggregate([{$group:{_id:"$country",poblacion:{$sum:"$population"},ciudades:{$sum:1}}},{$project:{_id:0,pais:"$_id",ratio:{$divide:["$poblacion","$ciudade
s"]}}},{$sort:{ratio:-1}}])
```

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}}

```java
public class Ejercicio29 {
    public static void main(String[] args) {
        /*
         db.ciudades.aggregate([{$match:{country:"ES"}},
                                {$sort:{population:-1}},
                                {$limit:3},
                                {$project:{_id:0,nombre:"$name",poblacion:"$population"}}])


        */
        MongoClient mongo = new MongoClient( "localhost" , 27017 );
        MongoDatabase database = mongo.getDatabase("test");
        MongoCollection<Document> collection = database.getCollection("ciudades");

        ArrayList<Document> lista = new ArrayList<>();
        Document d = new Document("$match",new Document("country","ES"));
        Document d1 = new Document("$sort",new Document("population",-1));
        Document d2 = new Document("$limit",3);
        Document d3 = new Document("$project",new Document("_id",0).append("nombre","$name").append("poblacion", "$population"));

        lista.add(d);
        lista.add(d1);
        lista.add(d2);
        lista.add(d3);

        AggregateIterable<Document> documento = collection.aggregate(lista);
        Iterator iterador = documento.iterator();
        while(iterador.hasNext()) {
            System.out.println(iterador.next());
        }
        mongo.close();

    }
}
```

```
<terminated> Ejercicio29 (1) [Java Application] C:\Program Files\Java\jre1.8.0_301\bin\javaw.exe  (6 feb 2023 22:03:42 – 22:03:43) [pid: 5412]
feb 06, 2023 10:03:43 PM com.mongodb.diagnostics.logging.JULLogger log
INFORMACIÓN: Cluster created with settings {hosts=[localhost:27017], mode=SINGLE, requiredClusterType=UNKNOWN, serverSelectionTimeout='300
feb 06, 2023 10:03:43 PM com.mongodb.diagnostics.logging.JULLogger log
INFORMACIÓN: Cluster description not yet available. Waiting for 30000 ms before timing out
feb 06, 2023 10:03:43 PM com.mongodb.diagnostics.logging.JULLogger log
INFORMACIÓN: Opened connection [connectionId{localValue:1, serverValue:677}] to localhost:27017
feb 06, 2023 10:03:43 PM com.mongodb.diagnostics.logging.JULLogger log
INFORMACIÓN: Monitor thread successfully connected to server with description ServerDescription{address=localhost:27017, type=STANDALONE,
feb 06, 2023 10:03:43 PM com.mongodb.diagnostics.logging.JULLogger log
INFORMACIÓN: Opened connection [connectionId{localValue:2, serverValue:678}] to localhost:27017
Document{{nombre=Madrid, poblacion=3255944}}
Document{{nombre=Barcelona, poblacion=1621537}}
Document{{nombre=Valencia, poblacion=814208}}
feb 06, 2023 10:03:43 PM com.mongodb.diagnostics.logging.JULLogger log
INFORMACIÓN: Closed connection [connectionId{localValue:2, serverValue:678}] to localhost:27017 because the pool has been closed.
```