

Laboratorium 3: MongoDB

Igor Dzierwa

Grupa: 1

2 z 10

1.Uruchom usługę MongoDB:

-> **mongod** — głównym procesem demona systemu MongoDB.

```
➤ mongod --dbpath mongo-data
{"t":{"$date":"2020-11-06T20:12:23.865+01:00"},"s":"I",  "c":"CONTROL",  "id":23285,   "ctx":"","msg":"Automatically disabling TLS 1.0, to force-enable TLS 1.0 specify --sslDisabledProtocols 'none'",
{"t":{"$date":"2020-11-06T20:12:23.867+01:00"},"s":"W",  "c":"NETWORK",  "id":22681,   "ctx":"","msg":"No TransportLayer configured during NetworkInterface startup",
{"t":{"$date":"2020-11-06T20:12:23.867+01:00"},"s":"I",  "c":"CONTROL",  "id":4648682, "ctx":"","msg":"Implicit TCP FastOpen in use.",
{"t":{"$date":"2020-11-06T20:12:23.868+01:00"},"s":"I",  "c":"STORAGE",  "id":4615611, "ctx":"","msg":"MongoDB starting", "attr":{"pid":7166,"port":27017,"dbPath":"mongo-data","architecture":"64-bit","host":"M
acBook-Pro-Igor.local"}}
{"t":{"$date":"2020-11-06T20:12:23.868+01:00"},"s":"I",  "c":"CONTROL",  "id":23403,   "ctx":"","msg":"Build Info", "attr":{"buildInfo":{"version":"4.4.1","gitVersion":"ad91a93a5a31e175f5cbf8c69561e788bc55ce1
","modules":[""],"allocator":"system","environment":{"distarch":"x86_64","target_arch":"x86_64"}}}}
{"t":{"$date":"2020-11-06T20:12:23.868+01:00"},"s":"I",  "c":"CONTROL",  "id":51765,   "ctx":"","msg":"Operating System", "attr":{"os":{"name":"Mac OS X","version":"19.6.0"}}}
{"t":{"$date":"2020-11-06T20:12:23.868+01:00"},"s":"I",  "c":"CONTROL",  "id":21951,   "ctx":"","msg":"Options set by command line", "attr":{"options":{"storage":{"dbPath":"mongo-data"}}}}
{"t":{"$date":"2020-11-06T20:12:23.873+01:00"},"s":"I",  "c":"STORAGE",  "id":22270,   "ctx":"","msg":"Storage engine to use detected by data files", "attr":{"dbPath":"mongo-data","storageEngine":"wiredTiger"}}
{"t":{"$date":"2020-11-06T20:12:23.874+01:00"},"s":"I",  "c":"STORAGE",  "id":22315,   "ctx":"","msg":"Opening WiredTiger", "attr":{"config":{"create,cache_size=7680M,session_max=33000,eviction:(threads_min=4,t
hread_max=4),config_base=false,statistics=(fast),log:(enabled=true,archive=true,path=journal,compressor=snappy),file_manager:(close_idle_time=100000,close_scan_interval=10,close_handle_minimum=250),statistics_log=(wait=6
),verbose=[recovery_progress,checkpoint_progress,compact_progress]}}}
{"t":{"$date":"2020-11-06T20:12:24.314+01:00"},"s":"I",  "c":"STORAGE",  "id":22430,   "ctx":"","msg":"WiredTiger message", "attr":{"message":"[1604689944:313533][7166:0x10efb9dc0]: txn-recover: [WT_VERB_RECOV
ERY_PROGRESS] Recovering log 4 through 5"}}
{"t":{"$date":"2020-11-06T20:12:24.375+01:00"},"s":"I",  "c":"STORAGE",  "id":22430,   "ctx":"","msg":"WiredTiger message", "attr":{"message":"[1604689944:375974][7166:0x10efb9dc0]: txn-recover: [WT_VERB_RECOV
ERY_PROGRESS] Recovering log 5 through 5"}}
{"t":{"$date":"2020-11-06T20:12:24.432+01:00"},"s":"I",  "c":"STORAGE",  "id":22430,   "ctx":"","msg":"WiredTiger message", "attr":{"message":"[1604689944:432973][7166:0x10efb9dc0]: txn-recover: [WT_VERB_RECOV
ERY | WT_VERB_RECOVERY_PROGRESS] Main recovery loop: starting at 4/249728 to 5/256"}}
{"t":{"$date":"2020-11-06T20:12:24.514+01:00"},"s":"I",  "c":"STORAGE",  "id":22430,   "ctx":"","msg":"WiredTiger message", "attr":{"message":"[1604689944:514159][7166:0x10efb9dc0]: txn-recover: [WT_VERB_RECOV
ERY_PROGRESS] Recovering log 4 through 5"}}
{"t":{"$date":"2020-11-06T20:12:24.571+01:00"},"s":"I",  "c":"STORAGE",  "id":22430,   "ctx":"","msg":"WiredTiger message", "attr":{"message":"[1604689944:571602][7166:0x10efb9dc0]: txn-recover: [WT_VERB_RECOV
ERY_PROGRESS] Recovering log 5 through 5"}}
{"t":{"$date":"2020-11-06T20:12:24.625+01:00"},"s":"I",  "c":"STORAGE",  "id":22430,   "ctx":"","msg":"WiredTiger message", "attr":{"message":"[1604689944:625719][7166:0x10efb9dc0]: txn-recover: [WT_VERB_RECOV
ERY | WT_VERB_RECOVERY_PROGRESS] Set global recovery timestamp: (0, 0)}}
{"t":{"$date":"2020-11-06T20:12:24.625+01:00"},"s":"I",  "c":"STORAGE",  "id":22430,   "ctx":"","msg":"WiredTiger message", "attr":{"message":"[1604689944:625779][7166:0x10efb9dc0]: txn-recover: [WT_VERB_RECOV
ERY | WT_VERB_RECOVERY_PROGRESS] Set global oldest timestamp: (0, 0)}}
{"t":{"$date":"2020-11-06T20:12:24.689+01:00"},"s":"I",  "c":"STORAGE",  "id":4795906,   "ctx":"","msg":"WiredTiger opened", "attr":{"durationMillis":815}}
{"t":{"$date":"2020-11-06T20:12:24.689+01:00"},"s":"I",  "c":"RECOVERY",  "id":23987,   "ctx":"","msg":"WiredTiger recoveryTimestamp", "attr":{"recoveryTimestamp":{"t":"0","i":"0"}}}
{"t":{"$date":"2020-11-06T20:12:24.707+01:00"},"s":"I",  "c":"STORAGE",  "id":22262,   "ctx":"","msg":"Timestamp monitor starting",
{"t":{"$date":"2020-11-06T20:12:24.709+01:00"},"s":"W",  "c":"CONTROL",  "id":22120,   "ctx":"","msg":"Access control is not enabled for the database. Read and write access to data and configuration is unres
tricted", "tags":["startupWarnings"]}
{"t":{"$date":"2020-11-06T20:12:24.709+01:00"},"s":"W",  "c":"CONTROL",  "id":22140,   "ctx":"","msg":"This server is bound to localhost. Remote systems will be unable to connect to this server. Start the ser
ver with --bind_ip <address> to specify which IP addresses it should serve responses from, or with --bind_ip_all to bind to all interfaces. If this behavior is desired, start the server with --bind_ip 127.0.0.1 to disable
this warning", "tags":["startupWarnings"]}
{"t":{"$date":"2020-11-06T20:12:24.709+01:00"},"s":"W",  "c":"CONTROL",  "id":22184,   "ctx":"","msg":"Soft rlimits too low", "attr":{"currentValue":256,"recommendedMinimum":64000},"tags":["startupWarnings"]}
{"t":{"$date":"2020-11-06T20:12:24.743+01:00"},"s":"I",  "c":"STORAGE",  "id":20536,   "ctx":"","msg":"Flow Control is enabled on this deployment",
{"t":{"$date":"2020-11-06T20:12:24.754+01:00"},"s":"I",  "c":"FTDC",  "id":20625,   "ctx":"","msg":"Initializing full-time diagnostic data capture", "attr":{"dataDirectory":"mongo-data/diagnostic.data"}}
{"t":{"$date":"2020-11-06T20:12:24.758+01:00"},"s":"I",  "c":"NETWORK",  "id":23015,   "ctx":"","msg":"Listening on", "attr":{"address":"/tmp/mongod-27017.sock"}}
{"t":{"$date":"2020-11-06T20:12:24.758+01:00"},"s":"I",  "c":"NETWORK",  "id":23015,   "ctx":"","msg":"Listening on", "attr":{"address":"127.0.0.1"}}
{"t":{"$date":"2020-11-06T20:12:24.758+01:00"},"s":"I",  "c":"NETWORK",  "id":23016,   "ctx":"","msg":"Waiting for connections", "attr":{"port":27017,"ssl":"off"}}
{"t":{"$date":"2020-11-06T20:13:21.423+01:00"},"s":"I",  "c":"NETWORK",  "id":22943,   "ctx":"","msg":"Connection accepted", "attr":{"remote":"127.0.0.1:51962","connectionId":1,"connectionCount":1}}
{"t":{"$date":"2020-11-06T20:13:21.424+01:00"},"s":"I",  "c":"NETWORK",  "id":51800,   "ctx":"","msg":"client metadata", "attr":{"remote":"127.0.0.1:51962","client":"conn1","doc":{"application":{"name":"MongoDB Shell"
},"driver":{"name":"MongoDB Internal Client","version":"4.4.1"},"os":{"type":"Darwin","name":"Mac OS X","architecture":"x86_64","version":"19.6.0"}}}}
{"t":{"$date":"2020-11-06T20:13:21.424+01:00"},"s":"I",  "c":"NETWORK",  "id":22944,   "ctx":"","msg":"Connection ended", "attr":{"remote":"127.0.0.1:51962","connectionId":1,"connectionCount":0}}
{"t":{"$date":"2020-11-06T20:13:21.424+01:00"},"s":"I",  "c":"NETWORK",  "id":22943,   "ctx":"","msg":"Connection accepted", "attr":{"remote":"127.0.0.1:52252","connectionId":2,"connectionCount":1}}
{"t":{"$date":"2020-11-06T20:13:21.424+01:00"},"s":"I",  "c":"NETWORK",  "id":51800,   "ctx":"","msg":"client metadata", "attr":{"remote":"127.0.0.1:52252","client":"conn2","doc":{"application":{"name":"MongoDB Shell"
},"driver":{"name":"MongoDB Internal Client","version":"4.4.1"},"os":{"type":"Darwin","name":"Mac OS X","architecture":"x86_64","version":"19.6.0"}}}}

```

-> **mongo** — interaktywny interfejs powłoki JavaScript.

```
➤ mongo
MongoDB shell version v4.4.1
connecting to mongod://127.0.0.1:27017/?compressors=disabled&gssapiServiceName=mongod
Implicit session: session { "id" : UUID("eb510ce0-02d0-4185-bb32-9592c975f82d") }
MongoDB server version: 4.4.1

The server generated these startup warnings when booting:
2020-11-06T20:12:24.709+01:00: Access control is not enabled for the database. Read and write access to data and configuration is unrestricted
2020-11-06T20:12:24.709+01:00: This server is bound to localhost. Remote systems will be unable to connect to this server. Start the server with --bind_ip <address> to specify which IP addresses it should serve r
esponses from, or with --bind_ip_all to bind to all interfaces. If this behavior is desired, start the server with --bind_ip 127.0.0.1 to disable this warning
2020-11-06T20:12:24.709+01:00: Soft rlimits too low
2020-11-06T20:12:24.709+01:00:   currentValue: 256
2020-11-06T20:12:24.709+01:00:   recommendedMinimum: 64000

---

Enable MongoDB's free cloud-based monitoring service, which will then receive and display
metrics about your deployment (disk utilization, CPU, operation statistics, etc).

The monitoring data will be available on a MongoDB website with a unique URL accessible to you
and anyone you share the URL with. MongoDB may use this information to make product
improvements and to suggest MongoDB products and deployment options to you.

To enable free monitoring, run the following command: db.enableFreeMonitoring()
To permanently disable this reminder, run the following command: db.disableFreeMonitoring()

---
```

2.Zaimportuj do MongoDB pliki yelp_academic_data

a) wykorzystaj komendę:

mongoimport --db <db-name> --collection <coll- name> --type json --file <file>

```
~/Desktop/yelp_dataset ls
yelp_academic_dataset_business.json yelp_academic_dataset_checkin.json yelp_academic_dataset_review.json yelp_academic_dataset_tip.json yelp_academic_dataset_user.json
~/Desktop/yelp_dataset mongoimport --db IgorDzierwal --collection business --file yelp_academic_dataset_business.json
2020-11-06T20:57:24.423+0100 connected to: mongodb://localhost/
2020-11-06T20:57:26.167+0100 42153 document(s) imported successfully. 0 document(s) failed to import.
~/Desktop/yelp_dataset mongoimport --db IgorDzierwal --collection checkin --file yelp_academic_dataset_checkin.json
2020-11-06T21:01:14.225+0100 connected to: mongodb://localhost/
2020-11-06T21:01:15.352+0100 31617 document(s) imported successfully. 0 document(s) failed to import.
~/Desktop/yelp_dataset mongoimport --db IgorDzierwal --collection checkin --file yelp_academic_dataset_checkin.json
2020-11-06T21:08:06.294+0100 connected to: mongodb://localhost/
2020-11-06T21:08:07.370+0100 31617 document(s) imported successfully. 0 document(s) failed to import.
```

```
~/Desktop/yelp_dataset mongoimport --db IgorDzierwa1 --collection review --file yelp_academic_dataset_review.json
connected to: mongodb://localhost/
2020-11-06T21:08:37.513+0100 [##.....] IgorDzierwa1.review 96.0MB/997MB (9.6%)
2020-11-06T21:08:40.516+0100 [####.....] IgorDzierwa1.review 192MB/997MB (19.3%)
2020-11-06T21:08:43.515+0100 [#####.....] IgorDzierwa1.review 297MB/997MB (29.8%)
2020-11-06T21:08:46.518+0100 [#####.....] IgorDzierwa1.review 400MB/997MB (40.1%)
2020-11-06T21:08:49.514+0100 [#####.....] IgorDzierwa1.review 492MB/997MB (49.4%)
2020-11-06T21:08:52.514+0100 [#####.....] IgorDzierwa1.review 591MB/997MB (59.3%)
2020-11-06T21:08:55.516+0100 [#####.....] IgorDzierwa1.review 683MB/997MB (68.5%)
2020-11-06T21:08:58.517+0100 [#####.....] IgorDzierwa1.review 777MB/997MB (77.9%)
2020-11-06T21:09:01.517+0100 [#####.....] IgorDzierwa1.review 869MB/997MB (87.1%)
2020-11-06T21:09:04.515+0100 [#####.....] IgorDzierwa1.review 964MB/997MB (96.6%)
2020-11-06T21:09:07.518+0100 [#####.....] IgorDzierwa1.review 997MB/997MB (100.0%)
2020-11-06T21:09:08.711+0100 [#####.....] IgorDzierwa1.review 997MB/997MB (100.0%)
2020-11-06T21:09:08.711+0100 1125458 document(s) imported successfully. 0 document(s) failed to import.
~/Desktop/yelp_dataset mongoimport --db IgorDzierwa1 --collection tip --file yelp_academic_dataset_tip.json
connected to: mongodb://localhost/
2020-11-06T21:09:43.862+0100 [#####.....] IgorDzierwa1.tip 43.0MB/75.7MB (56.8%)
2020-11-06T21:09:46.865+0100 [#####.....] IgorDzierwa1.tip 75.7MB/75.7MB (100.0%)
2020-11-06T21:09:49.047+0100 [#####.....] IgorDzierwa1.tip 75.7MB/75.7MB (100.0%)
2020-11-06T21:09:49.047+0100 403210 document(s) imported successfully. 0 document(s) failed to import.
~/Desktop/yelp_dataset mongoimport --db IgorDzierwa1 --collection user --file yelp_academic_dataset_user.json
connected to: mongodb://localhost/
2020-11-06T21:10:05.815+0100 [#####.....] IgorDzierwa1.user 50.5MB/113MB (44.6%)
2020-11-06T21:10:08.815+0100 [#####.....] IgorDzierwa1.user 100MB/113MB (88.5%)
2020-11-06T21:10:11.815+0100 [#####.....] IgorDzierwa1.user 100MB/113MB (88.5%)
2020-11-06T21:10:12.584+0100 [#####.....] IgorDzierwa1.user 113MB/113MB (100.0%)
2020-11-06T21:10:12.584+0100 252898 document(s) imported successfully. 0 document(s) failed to import.
```

b) stwórz nową bazę danych o nazwie: ImieNazwiskoNrGrupy:

```
> use IgorDzierwa1
switched to db IgorDzierwa1
```

```
> show dbs
IgorDzierwa1 0.766GB
admin         0.000GB
config        0.000GB
forum         0.000GB
local         0.000GB
> use IgorDzierwa1
```

c) Przyporządkuj kolekcjom opartych na importowanych plikach .json odpowiednie nazwy.

```
> show collections
business
checkin
review
tip
user
> |
```

3. Połącz się z bazą z użyciem narzędzia Robo3T:

The screenshot shows the Robo3T interface with the database 'IgorDzierwa1' selected. The left sidebar shows the database structure with collections: business, checkin, review, tip, and user. The main panel displays the 'db.stats()' command results, showing various statistics for the database.

Key	Value	Type
db	{ 14 fields }	Object
collections	5	Int32
views	0	Int32
objects	1886953	Int32
avgObjSize	705.496178760149	Double
dataSize	1331238131.0	Double
storageSize	804642816.0	Double
indexes	5	Int32
indexSize	18272256.0	Double
totalSize	822915072.0	Double
scaleFactor	1.0	Double
fsUsedSize	162777829376.0	Double
fsTotalSize	250685575168.0	Double
ok	1.0	Double

4. Za pomocą narzędzia Robo 3T wykonaj polecenie dodające do stworzonej bazy kolekcję „student”

- wprowadź własne dane do kolekcji: imię, nazwisko, obecność (typ bool), ocena z lab. (null), aktualna data, zaliczone przedmioty (*min 3 przykładowe*).
- wyświetl wynik dodania danej w formie `.json` `txt`.



The screenshot shows the Robo 3T interface with three tabs: 'IgorDzierwa1', 'localhost:27017', and 'IgorDzierwa1'. The active tab shows a MongoDB shell session. The first command is `db.students.insert()` with a document containing: `"imie": "Igor", "nazwisko": "Dzierwa", "obecność": true, "ocena": null, "aktualna data": Date(), "zaliczone przedmioty": ["AISD", "Technika cyfrowa", "Sieci komputerowe"]`. The second command is `db.students.find({})`. The execution time for the first command is 0.005 sec. The result shows 'Inserted 1 record(s) in 5ms'. The third command is `students` with an execution time of 0.002 sec. The result is a JSON document: `{ "_id" : ObjectId("5fa5be1d2987268befa480a0"), "imie" : "Igor", "nazwisko" : "Dzierwa", "obecność" : true, "ocena" : null, "aktualna data" : "Fri Nov 06 2020 22:20:29 GMT+0100 (CET)", "zaliczone przedmioty" : ["AISD", "Technika cyfrowa", "Sieci komputerowe"] }`

```
db.students.insert(  
  {  
    "imie": "Igor",  
    "nazwisko": "Dzierwa",  
    "obecność": true,  
    "ocena": null,  
    "aktualna data": Date(),  
    "zaliczone przedmioty": [  
      "AISD",  
      "Technika cyfrowa",  
      "Sieci komputerowe"  
    ]  
  }  
)  
  
db.students.find({})
```

0.005 sec.

Inserted 1 record(s) in 5ms

students 0.002 sec.

```
{  
  "_id" : ObjectId("5fa5be1d2987268befa480a0"),  
  "imie" : "Igor",  
  "nazwisko" : "Dzierwa",  
  "obecność" : true,  
  "ocena" : null,  
  "aktualna data" : "Fri Nov 06 2020 22:20:29 GMT+0100 (CET)",  
  "zaliczone przedmioty" : [  
    "AISD",  
    "Technika cyfrowa",  
    "Sieci komputerowe"  
  ]  
}
```

5. Za pomocą narzędzia Robo 3T wykonaj zapytania, które pozwolą uzyskać następujące wyniki:

a) ilość miejsc ocenianych na 5 gwiazdek (pole stars, kolekcja business)

```
* db.business.find({stars: 5}).count()
```

IgorDzierwa1 localhost:27017 IgorDzierwa1

```
db.business.find({stars: 5}).count()
```

0.057 sec.

5097

b) ilość restauracji w każdym mieście, wynik posortuj malejąco na podstawie liczby. Pole categories w dokumencie business musi zawierać wartość Restaurants.

IgorDzierwa1 localhost:27017 IgorDzierwa1

```
db.business.aggregate([
  { $match: { "categories": "Restaurants" } },
  { $group: { _id: "$city", count: { $sum: 1 } } },
  { $sort: { count: -1 } }
])
```

business 0.056 sec.

	_id	count
1	Las Vegas	3855.0
2	Phoenix	2493.0
3	Edinburgh	1049.0
4	Scottsdale	1023.0
5	Mesa	693.0
6	Madison	679.0
7	Tempe	672.0
8	Henderson	564.0
9	Chandler	548.0
10	Glendale	422.0
11	Gilbert	317.0
12	Peoria	221.0
13	North La...	198.0
14	Surprise	144.0
15	Goodyear	119.0
16	Waterloo	117.0
17	Avondale	100.0
18	Kitchener	96.0
19	Queen ...	82.0
20	Middleton	66.0

c) Ilość hoteli (atrybut categories powinien mieć wartość Hotels) w każdym stanie/okręgu (state), które posiadają darmowe Wi-fi (pole attributes, klucz-wartość 'Wi-Fi': 'free') oraz ocenę co najmniej 4.5 gwiazdki. Wykorzystaj funkcję group.

```
IgorDzierwa1 localhost:27017 IgorDzierwa1
db.business.aggregate([
  {$match:
    {$and : [
      {"categories": "Hotels"},
      {"attributes.Wi-Fi": "free"},
      {"stars": {$gte: 4.5}}
    ]}
  },
  {$group : {_id: "$state", count: {$sum: 1}}},
])

business 0.042 sec.

/* 1 */
{
  "_id" : "NV",
  "count" : 10.0
}

/* 2 */
{
  "_id" : "EDH",
  "count" : 13.0
}

/* 3 */
{
  "_id" : "MLN",
  "count" : 1.0
}

/* 4 */
{
  "_id" : "AZ",
  "count" : 33.0
}

/* 5 */
{
  "_id" : "WI",
  "count" : 10.0
}

/* 6 */
{
  "_id" : "ON",
  "count" : 2.0
}
```

d) Zwróć, ile recenzji posiadają oceny z każdej kategorii: funny, cool, useful.
Przypisanie recenzji do kategorii oznacza, że przynajmniej jedna osoba zagłosowała na recenzję w tej kategorii).
Wykorzystaniem mechanizmu map-reduce.

```
//funkcja mapujaca
var map = function() {
  emit("votes", this.votes);
};

//funkcja redukujaca
var reduce = function(voteName, array) {
  var valueFunny = 0;
  var valueCool = 0;
  var valueUseful = 0;

  for(var i = 0; i < array.length; i++) {
    if(array[i].funny > 0)
      valueFunny += 1;
    if(array[i].cool > 0)
      valueCool += 1;
    if(array[i].useful > 0)
      valueUseful += 1;
  }
  return {funny: valueFunny, cool: valueCool, useful: valueUseful};
};

db.user.mapReduce(
  map,
  reduce,
  {
    out: "votes_summary"
  }
)

db.votes_summary.find({})
```

3.07 sec.

```
/* 1 */
{
  "result" : "votes_summary",
  "ok" : 1.0,
  "_o" : {
    "result" : "votes_summary",
    "ok" : 1.0
  },
  "_keys" : [
    "result",
    "ok"
  ],
  "_db" : {
```

votes_summary 0.142 sec.

```
/* 1 */
{
  "_id" : "votes",
  "value" : {
    "funny" : 153130.0,
    "cool" : 159951.0,
    "useful" : 211034.0
  }
}
```

6. Wykonaj zadania punktu 4 z poziomu języka Java:

*Pobrane pliki JAR:

- mongo-java-driver-3.12.7.jar
- mongodb-driver-core-4.1.1.jar

```
import com.mongodb.BasicDBObject;
import com.mongodb.client.MongoCollection;
import com.mongodb.client.MongoDatabase;
import com.mongodb.MongoClient;
import org.bson.Document;

public class Mongo {
    private MongoClient mongoClient;
    private MongoDatabase db;

    public Mongo(String hostname, int port, String databaseName) {
        this.mongoClient = new MongoClient(hostname, port);
        this.db = mongoClient.getDatabase(databaseName);
    }

    public MongoClient getMongoClient() {
        return this.mongoClient;
    }

    public MongoDatabase getDb() {
        return this.db;
    }

    public void showCollections() {
        for (String name : db.listCollectionNames()) {
            System.out.println("collection name: " + name);
        }
    }
}
```

a) Odwzorowanie punktu 5a:

```
public void exercise5a() {
    ArrayList<Document> results = new ArrayList<>();
    MongoCollection<Document> collection = db.getCollection(s: "business");
    BasicDBObject whereQuery = new BasicDBObject();
    whereQuery.put("stars", 5);
    collection.find(whereQuery).into(results);
    System.out.println("Count: " + results.size());
}
```

Count: 5097

b) Odwzorowanie punktu 5b:

```
public void exercise5b() {
    MongoCollection<Document> collection = db.getCollection(s: "business");
    List<Bson> results = Arrays.asList(
        Aggregates.match(Filters.eq( fieldName: "categories", value: "Restaurants")),
        Aggregates.group( id: "$city", Accumulators.sum( fieldName: "count", expression: 1)),
        Aggregates.sort(Sorts.descending( ...fieldNames: "count"))
    );

    AggregateIterable iterable = collection.aggregate(results);

    for(Object i : iterable) {
        System.out.println(i);
    }
}
```


- Fragment zwróconej zawartości:

```
Document{{_id=Las Vegas, count=3855}}
Document{{_id=Phoenix, count=2493}}
Document{{_id=Edinburgh, count=1049}}
Document{{_id=Scottsdale, count=1023}}
Document{{_id=Mesa, count=693}}
Document{{_id=Madison, count=679}}
Document{{_id=Tempe, count=672}}
Document{{_id=Henderson, count=564}}
Document{{_id=Chandler, count=548}}
Document{{_id=Glendale, count=422}}
Document{{_id=Gilbert, count=317}}
Document{{_id=Peoria, count=221}}
Document{{_id=North Las Vegas, count=198}}
Document{{_id=Surprise, count=144}}
Document{{_id=Goodyear, count=119}}
Document{{_id=Waterloo, count=117}}
Document{{_id=Avondale, count=100}}
Document{{_id=Kitchener, count=96}}
Document{{_id=Queen Creek, count=82}}
Document{{_id=Middleton, count=66}}
Document{{_id=Cave Creek, count=63}}
Document{{_id=Casa Grande, count=61}}
Document{{_id=Fountain Hills, count=47}}
Document{{_id=Apache Junction, count=44}}
Document{{_id=Buckeye, count=42}}
Document{{_id=Sun Prairie, count=39}}
Document{{_id=Fitchburg, count=38}}
Document{{_id=Maricopa, count=37}}
Document{{_id=Monona, count=32}}
Document{{_id=Sun City, count=31}}
Document{{_id=Wickenburg, count=31}}
Document{{_id=Litchfield Park, count=27}}
Document{{_id=Paradise Valley, count=26}}
Document{{_id=Laveen, count=25}}
Document{{_id=Anthem, count=24}}
Document{{_id=Verona, count=19}}
Document{{_id=San Tan Valley, count=16}}
```

c) Odwzorowanie punktu 5c:

```
public void exercise5c() {
    MongoCollection<Document> collection = db.getCollection("business");
    List<Bson> results = Arrays.asList(
        Aggregates.match(Filters.and(
            Filters.eq(fieldName: "categories", value: "Hotels"),
            Filters.eq(fieldName: "attributes.Wi-Fi", value: "free"),
            Filters.gte(fieldName: "stars", value: 4.5)),
        Aggregates.group(id: "$state", Accumulators.sum(fieldName: "count", expression: 1)));

    AggregateIterable iterable = collection.aggregate(results);

    for(Object i : iterable) {
        System.out.println(i);
    }
}
```

```
Document{{_id=NV, count=10}}
Document{{_id=EDH, count=13}}
Document{{_id=MLN, count=1}}
Document{{_id=AZ, count=33}}
Document{{_id=WI, count=10}}
Document{{_id=ON, count=2}}
```

d) Odwzorowanie punktu 5d:

```

public void exercise5d() {
    ArrayList<Document> results = new ArrayList<>();
    Map<String, Integer> counter = new HashMap<>();
    int funnyCount = 0;
    int coolCount = 0;
    int usefulCount = 0;
    MongoCollection<Document> collection = db.getCollection( s: "user");

    BasicDBObject whereQuery = new BasicDBObject();
    whereQuery.put("votes", new Document("$exists", true));

    collection.find(whereQuery).into(results);

    for(Document user : results) {
        Document votes = (Document) user.get("votes");
        if(votes.getInteger( key: "funny") > 0)
            funnyCount++;
        if(votes.getInteger( key: "cool") > 0)
            coolCount++;
        if(votes.getInteger( key: "useful") > 0)
            usefulCount++;
    }

    counter.put("Funny", funnyCount);
    counter.put("Cool", coolCount);
    counter.put("Useful", usefulCount);

    System.out.println(counter);
}

```

```
{Cool=159951, Useful=211034, Funny=153130}
```

7. Napisz kod w języku Java (metoda), który zwróci użytkownika (nazwa użytkownika) o największej liczbie pozytywnych recenzji (ocena co najmniej 4.5).

```

public void exercise7() {
    MongoCollection<Document> collection = db.getCollection( s: "user");

    List<Bson> results = Arrays.asList(
        Aggregates.match(Filters.gte( fieldName: "average_stars", value: 4.5)),
        Aggregates.sort(Sorts.descending( ...fieldNames: "review_count")
    ));

    AggregateIterable iterable = collection.aggregate(results);
    Document user = (Document) iterable.first();
    System.out.println("Name: " + user.getString( key: "name") +
        "\n" +
        "Review count: " + user.getInteger( key: "review_count") +
        "\n" +
        "Average rate: " + user.getDouble( key: "average_stars"));
}

```

