

MongoDB

Bartosz Szar

zad1

Po zainstalowaniu oprogramowania i dodaniu odpowiedniej ścieżki do zmiennej środowiskowej PATH mogę użyć polecenia *mongo* z dowolnego katalogu:

```
Wiersz polecenia - mongo
Microsoft Windows [Version 10.0.18362.720]
(c) 2019 Microsoft Corporation. Wszelkie prawa zastrzeżone.

C:\Users\Lenovo>mongo
MongoDB shell version v4.2.5
connecting to: mongodb://127.0.0.1:27017/?compressors=disabled&gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("e9ed5cad-1869-4777-b65f-b5933338dcd1") }
MongoDB server version: 4.2.5
Server has startup warnings:
2020-03-30T18:10:27.686+0200 I CONTROL [initandlisten]
2020-03-30T18:10:27.686+0200 I CONTROL [initandlisten] ** WARNING: Access control is not enabled for the database.
2020-03-30T18:10:27.686+0200 I CONTROL [initandlisten] ** Read and write access to data and configuration is
unrestricted.
2020-03-30T18:10:27.686+0200 I CONTROL [initandlisten]
---
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()
---
>
```

zad2

Każdy z 5 plików *JSON* zaimportowałem do swojej bazy w sposób analogiczny do przedstawionego poniżej, używając komendy *mongoimport*:

```
Wiersz polecenia
C:\Users\Lenovo>cd C:\Users\Lenovo\Desktop\inf\semestr_4\Bazy Danych\bazyDanych\lab2\yelp_dataset

C:\Users\Lenovo\Desktop\inf\semestr_4\Bazy Danych\bazyDanych\lab2\yelp_dataset>mongoimport --db BartoszSzar_wt1115_B --c
ollection business --type json --file yelp_academic_dataset_business.json
2020-03-30T19:27:44.109+0200 Failed: open yelp_academic_dataset_business.json: The system cannot find the file specifi
ed.
2020-03-30T19:27:44.118+0200 0 document(s) imported successfully. 0 document(s) failed to import.

C:\Users\Lenovo\Desktop\inf\semestr_4\Bazy Danych\bazyDanych\lab2\yelp_dataset>mongoimport --db BartoszSzar_wt1115_B --c
ollection business --type json --file yelp_academic_dataset_business.json
2020-03-30T19:28:05.562+0200 connected to: mongodb://localhost/
2020-03-30T19:28:08.563+0200 [#####.....] BartoszSzar_wt1115_B.business 22.6MB/35.5MB (63.6%)
2020-03-30T19:28:10.250+0200 [#####] BartoszSzar_wt1115_B.business 35.5MB/35.5MB (100.0%)
2020-03-30T19:28:10.250+0200 42153 document(s) imported successfully. 0 document(s) failed to import.

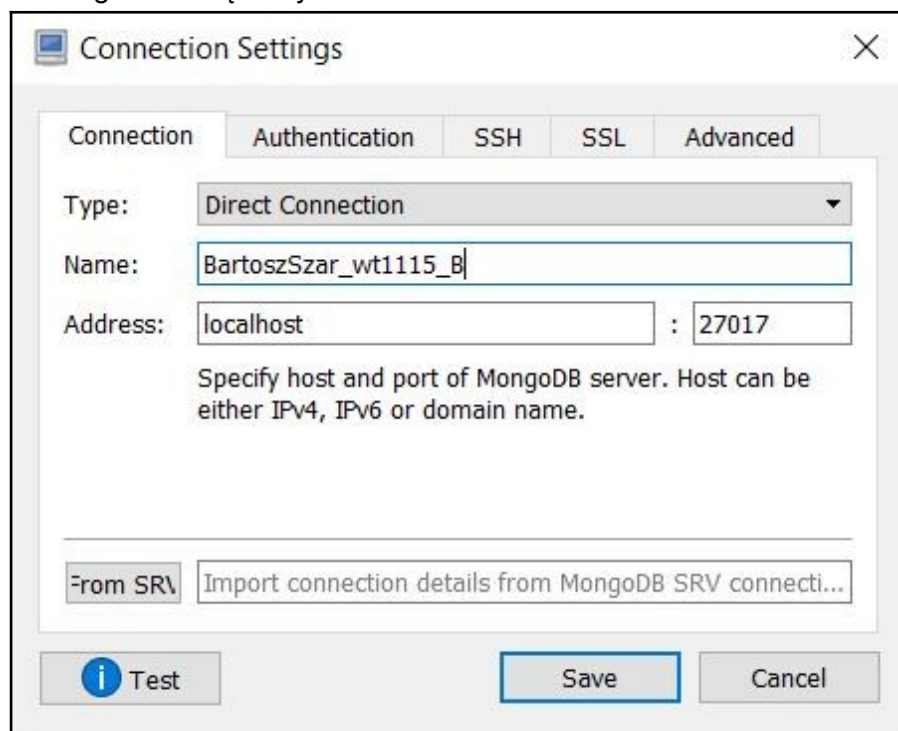
C:\Users\Lenovo\Desktop\inf\semestr_4\Bazy Danych\bazyDanych\lab2\yelp_dataset>
```

Po wykonaniu importu sprawdziłem zawartość kolekcji w bazie:

```
> show dbs
BartoszSzar_wt1115_B  0.653GB
admin                  0.000GB
config                 0.000GB
local                  0.000GB
> use BartoszSzar_wt1115_B
switched to db BartoszSzar_wt1115_B
> show collections
business
checkin
review
tip
user
>
```

zad3

Po zainstalowaniu oprogramowania Robo3T podłączyłem do systemu utworzoną uprzednio w MongoDB bazę danych:



zad4

The image shows two screenshots from the Robo 3T application. The top screenshot is the 'Insert Document' dialog, and the bottom screenshot is the main application window showing a database query.

Insert Document Dialog:

- Location: localhost:27017
- Database: BartoszSzar_wt1115_B
- Collection: student
- Document to insert:

```
{
  "name:" : "Jerzy",
  "surname" : "Kiler",
  "presence" : false,
  "grade" : null,
  "date" : "31.03.2020",
  "passed subjects" : [
    "lipski",
    "ryba",
    "siara"
  ]
}
```

Buttons: Validate, Save, Cancel

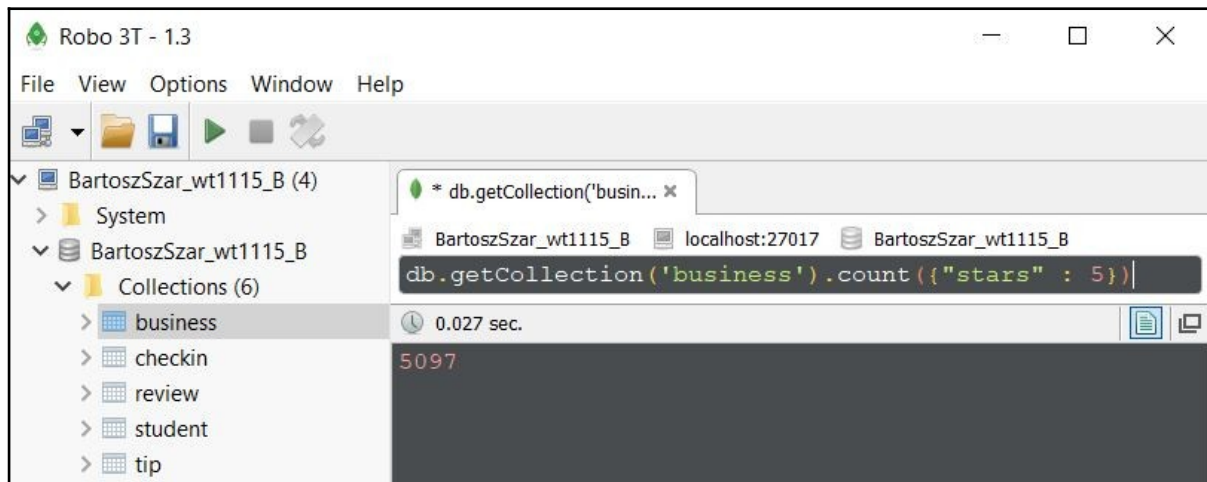
Robo 3T - 1.3 Main Window:

- Database: BartoszSzar_wt1115_B (4)
- Collection: student (selected)
- Query: `db.getCollection('student').find({})`
- Execution time: 0.001 sec.
- Results (1 document):

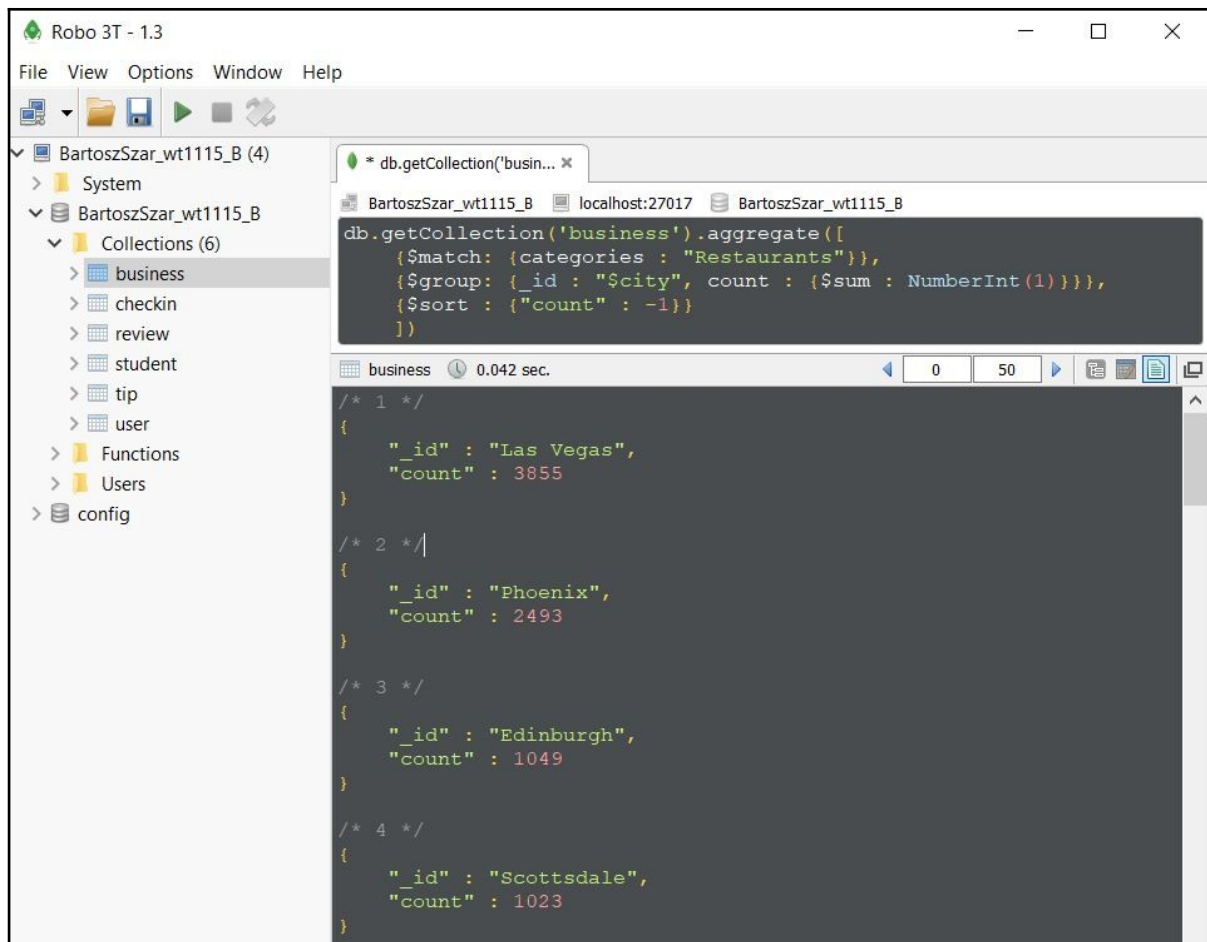
```
{
  "_id" : ObjectId("5e8230f96e8bc6ca11d4d514"),
  "name:" : "Jerzy",
  "surname" : "Kiler",
  "presence" : false,
  "grade" : null,
  "date" : "31.03.2020",
  "passed subjects" : [
    "lipski",
    "ryba",
    "siara"
  ]
}
```

Buttons: Logs

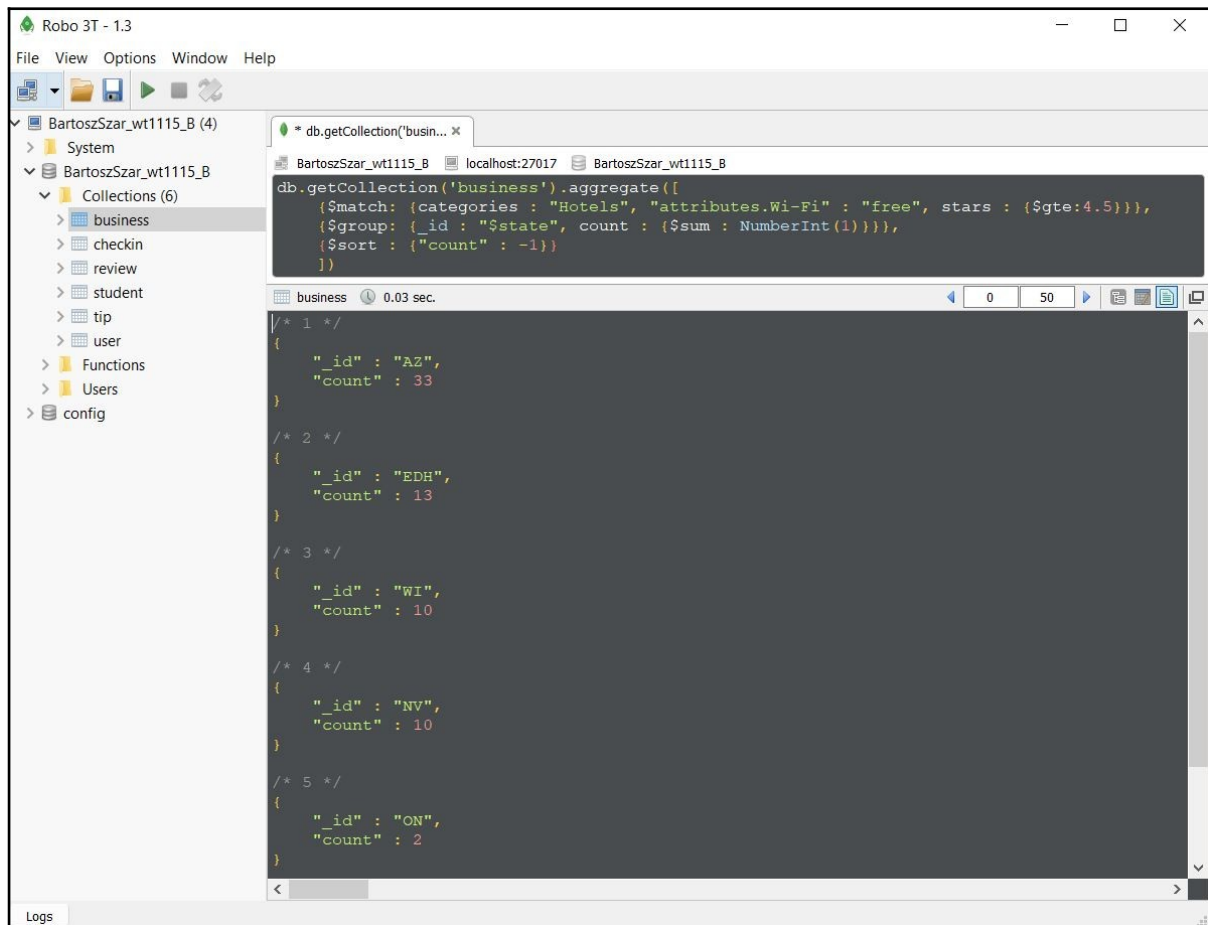
zad5 a)



zad5 b)



zad5 c)



zad6

```
protected long zad_6_5a(){
    MongoClient<Document> business =
        database.getCollection("business");
    Document query = new Document("stars", 5);
    return business.count(query);
}
```

5097


```

protected void zad_6_5b(){
    MongoCollection<Document> business =
        database.getCollection("business");
    AggregateIterable<Document> result = business.aggregate(
        Arrays.asList(
            Aggregates.match(Filters.eq("categories",
                "Restaurants")),
            Aggregates.group("$city",
                Accumulators.sum("no_of_restaurants", 1)),
            Aggregates.sort(Sorts.descending("no_of_restaurants"))
        )
    );
    for (Object tuple : result){
        System.out.println(tuple);
    }
}

```

```

Document{{_id=Las Vegas, no_of_restaurants=3855}}
Document{{_id=Phoenix, no_of_restaurants=2493}}
Document{{_id=Edinburgh, no_of_restaurants=1049}}
Document{{_id=Scottsdale, no_of_restaurants=1023}}
Document{{_id=Mesa, no_of_restaurants=693}}
Document{{_id=Madison, no_of_restaurants=679}}
Document{{_id=Tempe, no_of_restaurants=672}}
Document{{_id=Henderson, no_of_restaurants=564}}
Document{{_id=Chandler, no_of_restaurants=548}}
Document{{_id=Glendale, no_of_restaurants=422}}
Document{{_id=Gilbert, no_of_restaurants=317}}
Document{{_id=Peoria, no_of_restaurants=221}}
Document{{_id=North Las Vegas, no_of_restaurants=198}}
Document{{_id=Surprise, no_of_restaurants=144}}
Document{{_id=Goodyear, no_of_restaurants=119}}
Document{{_id=Waterloo, no_of_restaurants=117}}
Document{{_id=Avondale, no_of_restaurants=100}}
Document{{_id=Kitchenner, no_of_restaurants=96}}
Document{{_id=Queen Creek, no_of_restaurants=82}}
Document{{_id=Middleton, no_of_restaurants=66}}
Document{{_id=Cave Creek, no_of_restaurants=63}}
Document{{_id=Casa Grande, no_of_restaurants=61}}
Document{{_id=Fountain Hills, no_of_restaurants=47}}
Document{{_id=Apache Junction, no_of_restaurants=44}}
Document{{_id=Buckeye, no_of_restaurants=42}}

```

```

protected void zad_6_5c(){
    MongoClient<Document> business =
        database.getCollection("business");
    AggregateIterable<Document> result = business.aggregate(
        Arrays.asList(
            Aggregates.match(Filters.and(
                Filters.eq("categories",
                    "Hotels"),
                Filters.eq("attributes.Wi-Fi",
                    "free"),
                Filters.gte("stars", 4.5))),
            Aggregates.group("$state",
                Accumulators.sum("no_of_hotels", 1)),
            Aggregates.sort(Sorts.descending("no_of_hotels"))
        )
    );
    for (Object tuple : result){
        System.out.println(tuple);
    }
}

```

```

Document{{_id=AZ, no_of_hotels=33}}
Document{{_id=EDH, no_of_hotels=13}}
Document{{_id=Nv, no_of_hotels=10}}
Document{{_id=WI, no_of_hotels=10}}
Document{{_id=ON, no_of_hotels=2}}
Document{{_id=MLN, no_of_hotels=1}}

```

zad 7

```
protected String zad7(){
    MongoClient<Document> reviews =
        database.getCollection(("review"));
    MongoClient<Document> users =
        database.getCollection(("user"));
    Object bestRatedUser = reviews.aggregate(
        Arrays.asList(
            Aggregates.match(Filters.gte("stars", 4.5)),
            Aggregates.group("$user_id",
                Accumulators.sum("no_of_reviews", 1)),
            Aggregates.sort(Sorts.descending("no_of_reviews"))
        )).first().get("_id");
    return users.find(Filters.eq("user_id",
        bestRatedUser.toString())).first().get("name").toString();
}
```

Rand

zad8

```
List<String> zad8(){
    MongoClient<Document> reviews =
        database.getCollection(("review"));
    List<String> no_of_votes = new ArrayList<>();

    no_of_votes.add("funny: " + reviews.countDocuments(
        Filters.gt("votes.funny", 0)
    ));
    no_of_votes.add("cool: " + reviews.countDocuments(
        Filters.gt("votes.cool", 0)
    ));
    no_of_votes.add("useful: " + reviews.countDocuments(
        Filters.gt("votes.useful", 0)
    ));

    return no_of_votes;
}
```

```
[funny: 269256, cool: 346519, useful: 549519]
```