

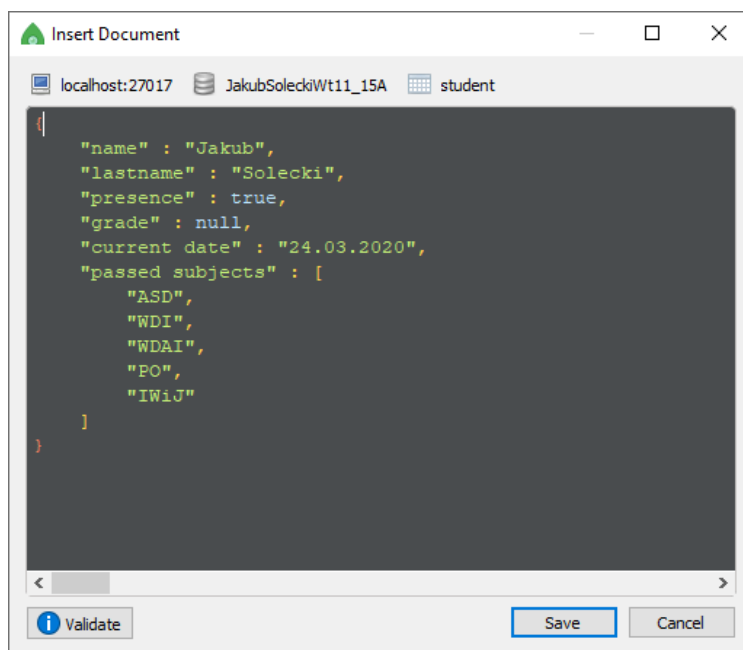
MongoDB – Raport z zajęć

Jakub Solecki

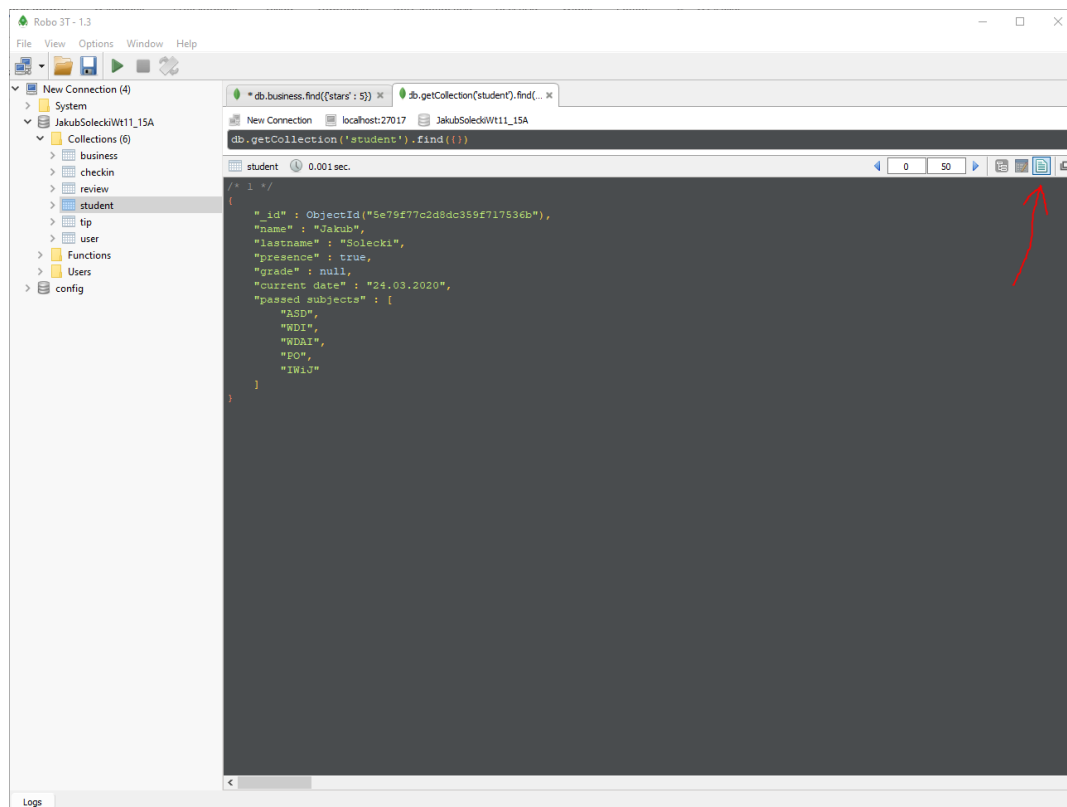
Zadanie 1, 2 oraz 3 zostały przeze mnie pomyślnie wykonane, jednak ze względu na konfiguracyjny chaos, który przy nich przeżyłem, nie posiadam z nich żadnych dowodów w postaci zrzutów ekranu, oprócz wciąż idącego procesu:

```
Wiersz polecenia - mongod
2020-03-24T12:20:37.474+0100 I STORAGE [initandlisten] WiredTiger message [1585048837:473523][7256:140708684258896], txn-recover: Recovering log
2 through 3
2020-03-24T12:20:37.546+0100 I STORAGE [initandlisten] WiredTiger message [1585048837:545499][7256:140708684258896], txn-recover: Recovering log
3 through 3
2020-03-24T12:20:37.613+0100 I STORAGE [initandlisten] WiredTiger message [1585048837:613478][7256:140708684258896], txn-recover: Set global rec
overy timestamp: (0, 0)
2020-03-24T12:20:37.620+0100 I RECOVERY [initandlisten] WiredTiger recoveryTimestamp. Ts: Timestamp(0, 0)
2020-03-24T12:20:37.624+0100 I STORAGE [initandlisten] Timestamp monitor starting
2020-03-24T12:20:37.625+0100 I CONTROL [initandlisten]
2020-03-24T12:20:37.625+0100 I CONTROL [initandlisten] ** WARNING: Access control is not enabled for the database.
2020-03-24T12:20:37.626+0100 I CONTROL [initandlisten] **      Read and write access to data and configuration is unrestricted.
2020-03-24T12:20:37.626+0100 I CONTROL [initandlisten]
2020-03-24T12:20:37.627+0100 I CONTROL [initandlisten] ** WARNING: This server is bound to localhost.
2020-03-24T12:20:37.627+0100 I CONTROL [initandlisten] **      Remote systems will be unable to connect to this server.
2020-03-24T12:20:37.627+0100 I CONTROL [initandlisten] **      Start the server with --bind_ip <address> to specify which IP
2020-03-24T12:20:37.628+0100 I CONTROL [initandlisten] **      addresses it should serve responses from, or with --bind_ip_all to
2020-03-24T12:20:37.628+0100 I CONTROL [initandlisten] **      bind to all interfaces. If this behavior is desired, start the
2020-03-24T12:20:37.628+0100 I CONTROL [initandlisten] **      server with --bind_ip 127.0.0.1 to disable this warning.
2020-03-24T12:20:37.629+0100 I CONTROL [initandlisten]
2020-03-24T12:20:37.631+0100 I SHARDING [initandlisten] Marking collection local.system.replset as collection version: <unsharded>
2020-03-24T12:20:37.632+0100 I STORAGE [initandlisten] Flow Control is enabled on this deployment.
2020-03-24T12:20:37.633+0100 I SHARDING [initandlisten] Marking collection admin.system.roles as collection version: <unsharded>
2020-03-24T12:20:37.633+0100 I SHARDING [initandlisten] Marking collection admin.system.version as collection version: <unsharded>
2020-03-24T12:20:37.633+0100 I SHARDING [initandlisten] Marking collection local.startup_log as collection version: <unsharded>
2020-03-24T12:20:37.639+0100 I SHARDING [initandlisten] Marking collection local.startup_log as collection version: <unsharded>
2020-03-24T12:20:37.895+0100 W FTDC [initandlisten] Failed to initialize Performance Counters for FTDC: WindowsPdhError: PdhExpandCounterPath
W failed with 'The specified object was not found on the computer.' for counter '\\Processor(_Total)\\% Idle Time'
2020-03-24T12:20:37.895+0100 I FTDC [initandlisten] Initializing full-time diagnostic data capture with directory 'C:/data/db/diagnostic.data
'
2020-03-24T12:20:37.900+0100 I SHARDING [LogicalSessionCacheRefresh] Marking collection config.system.sessions as collection version: <unsharded>
2020-03-24T12:20:37.900+0100 I SHARDING [LogicalSessionCacheReap] Marking collection config.transactions as collection version: <unsharded>
2020-03-24T12:20:37.900+0100 I NETWORK [listener] Listening on 127.0.0.1
2020-03-24T12:20:37.901+0100 I NETWORK [listener] waiting for connections on port 27017
2020-03-24T12:20:38.001+0100 I SHARDING [ftdc] Marking collection local.oplog.rs as collection version: <unsharded>
```

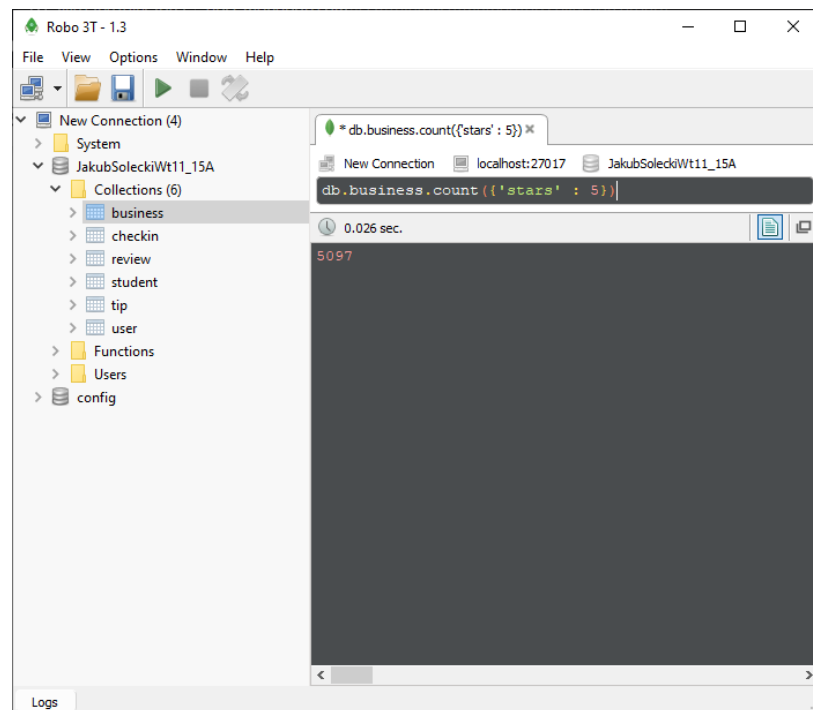
Zad 4a



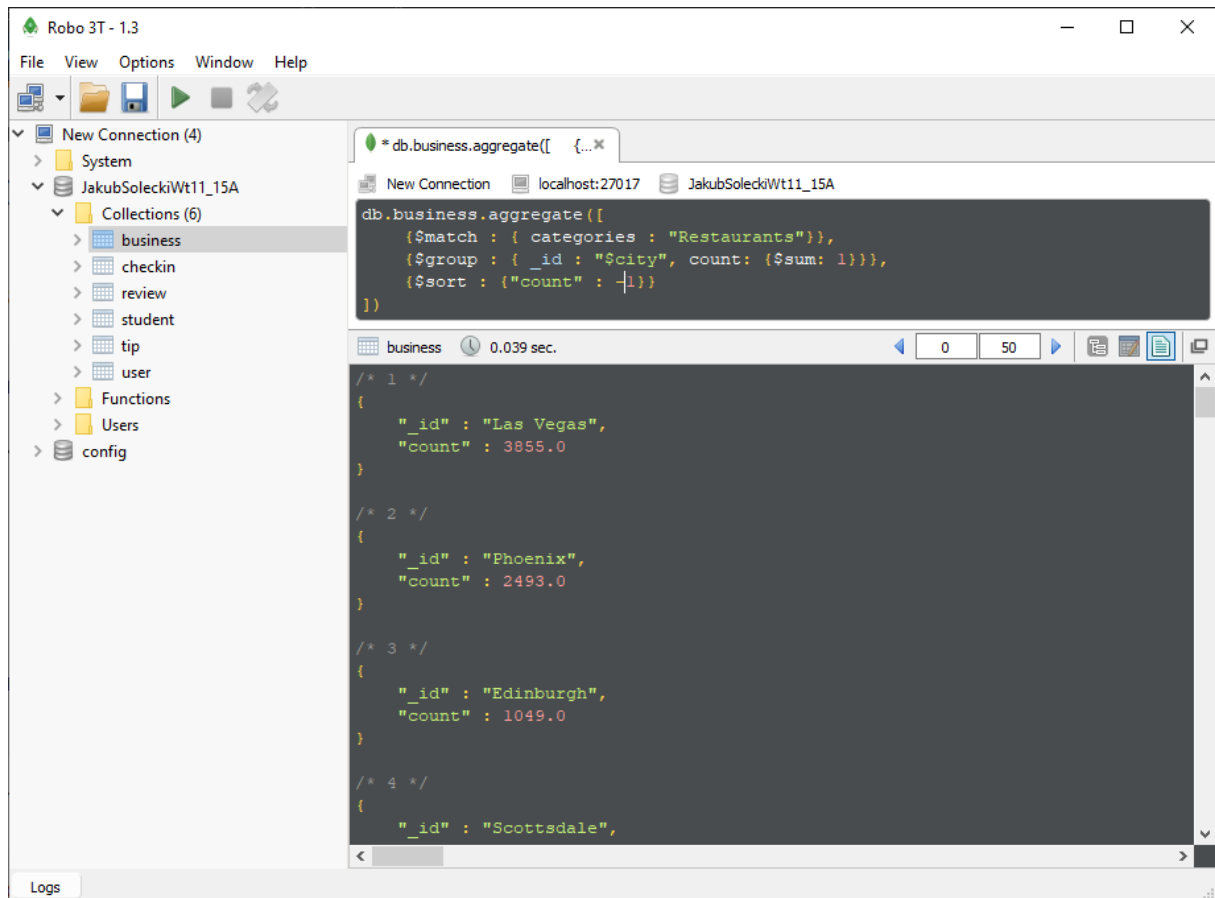
Zad 4b



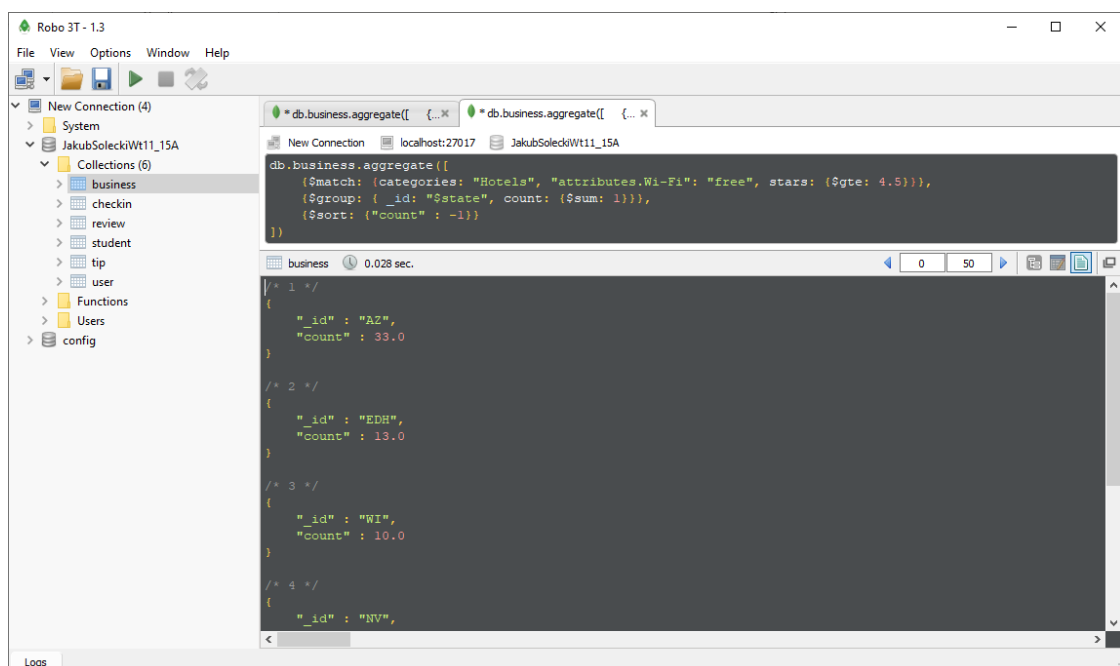
Zad 5a



Zad 5b



Zad 5c



Zad 6

```
private long zad6_5a(){
    MongoClient<Document> businessColl = db.getCollection( s: "business");
    BasicDBObject whereQuery = new BasicDBObject();
    whereQuery.put("stars", 5);
    return businessColl.countDocuments(whereQuery);
}
```

5097

```
private void zad6_5b() {
    MongoClient<Document> businessColl = db.getCollection( s: "business");
    AggregateIterable agg = businessColl.aggregate(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"))))
    ));
    for(Object res : agg) {
        System.out.println((res));
    }
}
```

```
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=Wickenburg, count=31}}
Document{{_id=Sun City, count=31}}
Document{{_id=Litchfield Park, count=27}}
Document{{_id=Paradise Valley, count=26}}
Document{{_id=Laveen, count=25}}
```

(i dużo więcej)

```

private void zad6_5c() {
    MongoClient<Document> businessColl = db.getCollection( s: "business");
    AggregateIterable agg = businessColl.aggregate(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)),
        Aggregates.sort(Sorts.descending( ...fieldNames: "count"))
    )
);

    for(Object res : agg) {
        System.out.println((res));
    }
}

```

```

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

```

Zad 8

```

private List<String> zad8() {
    MongoClient<Document> revs = db.getCollection( s: "review");
    List<String> count = new ArrayList<>();

    count.add("funny: " + revs.countDocuments(Filters.gt( fieldName: "votes.funny", value: 0)));
    count.add("cool: " + revs.countDocuments(Filters.gt( fieldName: "votes.cool", value: 0)));
    count.add("useful: " + revs.countDocuments(Filters.gt( fieldName: "votes.useful", value: 0)));

    return count;
}

```

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