

# **Laboratorium 3: MongoDB — zadanie**

Igor Dzierwa

Grupa: 1

# 1. Wykorzystując bazę danych yelp dataset, wykonaj zapytanie i komendy MongoDB, aby uzyskać następujące rezultaty.

- a) Zwróć bez powtórzeń wszystkie nazwy miast, w których znajdują się firmy (*business*).  
Wynik posortuj na podstawie nazwy miasta alfabetycznie.

```
* db.business.distinct("city").sort()
IgorDzierwa1 localhost:27017 IgorDzierwa1
db.business.distinct("city").sort()
0.072 sec.
/* 1 */
[
  "Ahwatukee",
  "Anthem",
  "Apache Junction",
  "Arcadia",
  "Atlanta",
  "Avondale",
  "Black Canyon City",
```

- b) Zwróć liczbę wszystkich recenzji, które pojawiły się po 2011 roku (*włącznie*).

```
* db.review.find({date: {$gte:'2011-01-01'}})
IgorDzierwa1 localhost:27017 IgorDzierwa1
db.review.find({date: {$gte:'2011-01-01'}}).count()
4.63 sec.
880318
```

- c) Zwróć dane wszystkich zamkniętych (*open*) firm (*business*) z pól: nazwa, adres, gwiazdki (*stars*).

```
* db.business.find({open: false},{name: 1,
IgorDzierwa1 db.business.find({open: false}, {name: 1, full_address: 1, stars: 1})
db.business.find({open: false}, {name: 1, full_address: 1, stars: 1})
business 0.004 sec.
/* 1 */
{
  "_id" : ObjectId("5fa5aaa46af0cf80f36737bc"),
  "full_address" : "4156 County Rd B\nMc Farland, WI 53558",
  "name" : "Charter Communications",
  "stars" : 1.5
}
/* 2 */
{
  "_id" : ObjectId("5fa5aaa46af0cf80f36737c8"),
  "full_address" : "6401 University Ave\nMiddleton, WI 53562",
  "name" : "Crandalls Carryout & Catering",
  "stars" : 4.0
}
/* 3 */
```

d) Zwróć dane wszystkich użytkowników (*user*), którzy nie uzyskali ani jednego pozytywnego głosu z kategorii (*funny lub useful*), wynik posortuj alfabetycznie według imienia użytkownika.

```

* db.user.find({$or: [{"votes.funny": 0}, {"votes.useful": 0}], {name: 1}})
IgorDzierwa1 localhost:27017 IgorDzierwa1
db.user.find({$or: [{"votes.funny": 0}, {"votes.useful": 0}]}).sort({name: 1})
user 0.267 sec.
/* 1 */
{
  "_id" : ObjectId("5fa5ad9fe39777c1e18152f4"),
  "yelping_since" : "2009-08",
  "votes" : {
    "funny" : 0,
    "useful" : 0,
    "cool" : 0
  },
  "review_count" : 1,
  "name" : "Bernard",
  "user_id" : "xP3SPgfgW2vc5Zj5uV8SEA",
  "friends" : [],
  "fans" : 0,
  "average_stars" : 5.0,
  "type" : "user",
  "compliments" : {},
  "elite" : []
}
/* 2 */
{
  "_id" : ObjectId("5fa5ada0e39777c1e1825116"),
  "yelping_since" : "2011-12",
  "votes" : {
    "funny" : 0,
    "useful" : 2,
    "cool" : 2
  },
  "review_count" : 10,
  "name" : "Anastacia",
  "user_id" : "qJlc0rYytqzeVBiTPftfSA",
  "friends" : [],
  "fans" : 0,
  "average_stars" : 4.5,
  "type" : "user",
  "compliments" : {},
  "elite" : []
}
/* 3 */

```

e) Określ, ile każde przedsiębiorstwo otrzymało wskazówek/napiwków (*tip*) w 2012. Wynik posortuj alfabetycznie według liczby (*tip*).

```

* db.tip.aggregate([{$match: {date: {$gte: '2012-01-01', $lte: '2012-12-31'}}},
{$group: {_id: "$business_id", countTips: {$sum: 1}}},
{$sort: {countTips: 1}}])
IgorDzierwa1 localhost:27017 IgorDzierwa1
db.tip.aggregate([{$match: {date: {$gte: '2012-01-01', $lte: '2012-12-31'}}},
{$group: {_id: "$business_id", countTips: {$sum: 1}}},
{$sort: {countTips: 1}}])
tip 0.415 sec.
/* 1 */
{
  "_id" : "d_f2yIJ8mkRXD_BnQ-yA7A",
  "countTips" : 1.0
}
/* 2 */
{
  "_id" : "vP61sXEXF4SssXR5zIhrGA",
  "countTips" : 1.0
}
/* 3 */
{
  "_id" : "zUEy3NGSw9WxcCAN3fL6vg",
  "countTips" : 1.0
}
/* 4 */
{
  "_id" : "UHIL-SpZVtYkxDa3_8HH3Q",
  "countTips" : 1.0
}
/* 5 */
{
  "_id" : "GkCAW9zoECIg7T_HIwCgNA",
  "countTips" : 1.0
}
/* 6 */
{
  "_id" : "4U01z6WgowkETK3YaIT8sg",
  "countTips" : 1.0
}

```

f) Wyznacz, jaką średnią ocen (*stars*) uzyskała każda firma (*business*) na podstawie wszystkich recenzji.  
Wynik ogranicz do recenzji, które uzyskały min. 4.0 gwiazdki.

```

x * db.review.aggregate([ {$group: {_id: db.getCollection('review').find({}})
IgorDzierwa1 localhost:27017 IgorDzierwa1
db.review.aggregate([
  {$group: {_id: "$business_id", averageStars: {$avg: "$stars"}}},
  {$match: {averageStars: {$gte: 4}}}
])

review 2.47 sec.

/* 1 */
{
  "_id" : "ME52N31r904bQ-RQM2SmFg",
  "averageStars" : 4.0
}

/* 2 */
{
  "_id" : "-FK47t3X0ctRNSSmZeFFQA",
  "averageStars" : 4.393939393939393
}

/* 3 */
{
  "_id" : "HpgYqW2SkUKbko_b1N1zCQ",
  "averageStars" : 4.35714285714286
}

/* 4 */
{
  "_id" : "9B5UMyg06E70xwxF1HIG3w",
  "averageStars" : 4.28571428571429
}

/* 5 */
{
  "_id" : "ThP_Q_vASwpIjo5i6axaoQ",
  "averageStars" : 4.083333333333333
}

/* 6 */
{
  "_id" : "08xNB6lB-2kMrj5bqJaEMA",
  "averageStars" : 4.4
}

/* 7 */

```

g) Usun wszystkie firmy (*business*), które posiadają ocenę (*stars*) równą 2.0.

```

x * db.business.deleteMany({stars: 2.0})
IgorDzierwa1 localhost:27017 IgorDzierwa1
db.business.deleteMany({stars: 2.0})

0.061 sec.

/* 1 */
{
  "acknowledged" : true,
  "deletedCount" : 1576.0
}

```

## 2. Zdefiniuj funkcję (*MongoDB*) umożliwiającą dodanie nowej recenzji (*review*).

```

* db.review.find({review_id: "1234567890"})
IgorDzierwa1 localhost:27017 IgorDzierwa1
var addReview = (user_id, review_id, business_id, stars, text) => {
  db.review.insertOne({
    "votes": {
      "funny": 0,
      "useful": 0,
      "cool": 0,
    },
    "user_id": user_id,
    "review_id": review_id,
    "stars": stars,
    "date": Date(),
    "text": text,
    "type": "review",
    "business_id": business_id
  })
}
addReview("123456", "1234567890", "vcNAWiLM4dR7D2nmwJ7nCA", 5, "test");
db.review.find({review_id: "1234567890"})

review 3.45 sec.

/* 1 */
{
  "_id" : ObjectId("5fbc38f69b334c066750462b"),
  "votes" : {
    "funny" : 0.0,
    "useful" : 0.0,
    "cool" : 0.0
  },
  "user_id" : "123456",
  "review_id" : "1234567890",
  "stars" : 5.0,
  "date" : "Wed Nov 25 2020 11:59:02 GMT+0100 (CET)",
  "text" : "test",
  "type" : "review",
  "business_id" : "vcNAWiLM4dR7D2nmwJ7nCA"
}

```

## 3. Zdefiniuj funkcję (*MongoDB*), która zwróci wszystkie biznesy (*business*), w których w kategorii znajduje się podana przez użytkownika cecha. Wartość kategorii należy przekazać do funkcji jako parametr. Wykonaj przykładowe wywołanie zdefiniowanej funkcji.

```

* var findByCategory = (category) => {
db.getCollection('business').find({})
IgorDzierwa1 localhost:27017 IgorDzierwa1
var findByCategory = (category) => {
  return db.business.find({categories: category})
}
findByCategory("Television Stations")

business 0.466 sec.

/* 1 */
{
  "_id" : ObjectId("5fa5aaa46af0cf80f36737bc"),
  "business_id" : "oLctHIA1Axmsg0uu4dM6Vw",
  "full_address" : "4156 County Rd B\nMc Farland, WI 53558",
  "hours" : {},
  "open" : false,
  "categories" : [
    "Television Stations",
    "Mass Media"
  ],
  "city" : "Mc Farland",
  "review_count" : 10,
  "name" : "Charter Communications",
  "neighborhoods" : [],
  "longitude" : -89.3229199,
  "state" : "WI",
  "stars" : 1.5,
  "latitude" : 42.9685074,
  "attributes" : {},
  "type" : "business"
}
/* 2 */

```

4. Zdefiniuj funkcję (*MongoDB*), która umożliwi modyfikację nazwy użytkownika (*user*) na podstawie podanego id. Id oraz nazwa mają być przekazywane jako parametry.

```

* db.user.find({name: "igordzie"}) db.getCollection('user').find({})
IgorDzierwa1 localhost:27017 IgorDzierwa1

var modifyUser = (userID, userName) => {
  db.user.updateOne(
    {"user_id": userID},
    {$set: {"name": userName}}
  )
};

modifyUser("5Xh4Qc3rxhAQ_NcNtxLssQ", "igordzie");

db.user.find({name: "igordzie"})

user 0.18 sec.

/* 1 */
{
  "_id" : ObjectId("5fa5ad9de39777c1e18092db"),
  "yelping_since" : "2012-01",
  "votes" : {
    "funny" : 0,
    "useful" : 1,
    "cool" : 0
  },
  "review_count" : 1,
  "name" : "igordzie",
  "user_id" : "5Xh4Qc3rxhAQ_NcNtxLssQ",
  "friends" : [],
  "fans" : 0,
  "average_stars" : 1.0,
  "type" : "user",
  "compliments" : {},
  "elite" : []
}

```

5. Zwróć średnią ilość wszystkich wskazówek/napiwków dla każdego z biznesów, wykorzystaj map reduce.

```

* var mapf = function() { emit(this.bus db.getCollection('tip').find({})
IgorDzierwa1 localhost:27017 IgorDzierwa1

var mapf = function() {
  emit(this.business_id, 1);
};

var reducef = function(key, value) {
  return value.length;
};

db.tip.mapReduce(
  mapf,
  reducef,
  {out: "tip_count"}
)

db.getCollection("tip_count").find({}).sort({value: -1})

3.81 sec.

result ok _id _keys _db _coll
1 tip_count 1.0 { 2 fields } [ 2 ... { 3 fields } { 4 fields }

tip_count 0.107 sec.

/* 1 */
{
  "_id" : "jf67Z1pwmElRSX1lpQHiJg",
  "value" : 2678.0
}

/* 2 */
{
  "_id" : "hW0Ne_JTHIEAgGF1rAdnR-g",
  "value" : 1778.0
}

/* 3 */
{
  "_id" : "AtjsjFzalWqJ7S9DUFQ4bw",
  "value" : 1043.0
}

/* 4 */

```

## 6. Odwzoruj wszystkie zadania z punktu 1 w języku programowania (np. JAVA) z pomocą API do MongoDB. Wykorzystaj dla każdego zadania odrębną metodę.

**\*Wybrany język programowania: Java**

- Pobrane pliki jar:

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

```
public class Mongo {
    private MongoClient mongoClient;
    private MongoDB db;

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

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

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

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

```
public class Main {

    public static void main(String[] args) {
        Mongo mongoLab = new Mongo( hostname: "localhost", port: 27017, dbName: "IgorDzierwa1");

        mongoLab.showCollections();
    }
}
```

```
collection name: user
collection name: students
collection name: checkin
collection name: votes_summary
collection name: business
collection name: tip
collection name: review
collection name: sum_tips
```

a) Zwróć bez powtórzeń wszystkie nazwy miast, w których znajdują się firmy (*business*).  
Wynik posortuj na podstawie nazwy miasta alfabetycznie.

```
public void pointA () {
    MongoCollection<Document> collection = this.db.getCollection( s: "business");
    List<String> results = new ArrayList<>();
    collection.distinct( s: "city", String.class).into(results);

    Collections.sort(results);

    for (String city : results) {
        System.out.println(city);
    }
}
```

```
Ahwatukee
Anthem
Apache Junction
Arcadia
Atlanta
Avondale
Black Canyon City
```

b) Zwróć liczbę wszystkich recenzji, które pojawiły się po 2011 roku (*włącznie*).

```
public void pointB () {
    MongoCollection<Document> collection = this.db.getCollection( s: "review");
    List<Document> results = new ArrayList<>();
    BasicDBObject whereQuery = new BasicDBObject();
    whereQuery.put("date", new Document("$gte", "2011-01-01"));
    collection.find(whereQuery).into(results);

    System.out.println("All reviews after 2011-01-01: " + results.size());
}
```

All reviews after 2011-01-01: 880320

c) Zwróć dane wszystkich zamkniętych (*open*) firm (*business*) z pól: nazwa, adres, gwiazdki (*stars*).

```
public void pointC () {
    MongoCollection<Document> collection = this.db.getCollection( s: "business");
    List<Document> results = new ArrayList<>();
    BasicDBObject whereQuery = new BasicDBObject();
    whereQuery.put("open", false);
    collection.find(whereQuery)
        .projection(Projections.include( ...fieldNames: "name", "full_address", "stars"))
        .into(results);

    for (Document document : results) {
        System.out.println(document);
    }
}
```

```
Document{{_id=5fbd9ca82f691c51b61f2367, full_address=10430 S Eastern Ave
Anthem
Henderson, NV 89052, name=Carmines, stars=3.0}}
Document{{_id=5fbd9ca82f691c51b61f2377, full_address=3000 W Ann Rd
Ste 109
North Las Vegas, NV 89031, name=Super No1 Chinese Restaurant, stars=3.5}}
Document{{_id=5fbd9ca82f691c51b61f23f0, full_address=13014 N Saguardo Blvd
Fountain Hills, AZ 85268, name=Moxie Modern Fare, stars=3.5}}
```

d) Zwróć dane wszystkich użytkowników (*user*), którzy nie uzyskali ani jednego pozytywnego głosu z kategorii (*funny lub useful*), wynik posortuj alfabetycznie według imienia użytkownika.

```
public void pointD () {
    MongoCollection<Document> collection = this.db.getCollection( s: "user");
    List<Document> results = new ArrayList<>();

    BasicDBObject orQuery = new BasicDBObject();
    List<BasicDBObject> orArguments = new ArrayList<>();
    orArguments.add(new BasicDBObject("votes.funny", 0));
    orArguments.add(new BasicDBObject("votes.useful", 0));
    orQuery.put("$or", orArguments);

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

    for (Document document : results) {
        System.out.println(document);
    }
}
```

```
Document{{_id=5fa5ada4e39777c1e1846eac, yelping_since=2010-11, votes=Document{{funny=0, useful=1, cool=0}}, review_count=3, name=Allister, user_id=ZRzhH3_33EkbinjsWYHRNw, friends=[], fans=0, ave
Document{{_id=5fa5ada4e39777c1e1846eaf, yelping_since=2013-06, votes=Document{{funny=0, useful=0, cool=0}}, review_count=1, name=Bibek, user_id=hmPP1xvAv9b-TY8bman85w, friends=[], fans=0, averag
Document{{_id=5fa5ada4e39777c1e1846eb1, yelping_since=2013-08, votes=Document{{funny=0, useful=4, cool=1}}, review_count=1, name=Ben, user_id=eurVEQKminFFm0Z6j1ULJw, friends=[], fans=0, average_
Document{{_id=5fa5ada4e39777c1e1846eb2, yelping_since=2012-08, votes=Document{{funny=0, useful=3, cool=3}}, review_count=2, name=Jennifer, user_id=cm20U1KU7sE51bsCfN8jzw, friends=[YHbrdVajp-WYsj
Document{{_id=5fa5ada4e39777c1e1846eb4, yelping_since=2013-10, votes=Document{{funny=1, useful=0, cool=0}}, review_count=1, name=Anamika, user_id=CYKWssZpM3JVE6uXzydKGUA, friends=[], fans=0, aver
Document{{_id=5fa5ada4e39777c1e1846eb5, yelping_since=2014-04, votes=Document{{funny=0, useful=0, cool=0}}, review_count=2, name=Stacey, user_id=lyq6kwACo7GGWVs-4VKPHQ, friends=[], fans=0, avera
Document{{_id=5fa5ada4e39777c1e1846eb7, yelping_since=2007-05, votes=Document{{funny=0, useful=1, cool=0}}, review_count=1, name=Eric, user_id=gXeEBcrUAA6mMamDRGW78A, friends=[], fans=0, average_
Document{{_id=5fa5ada4e39777c1e1846eb8, yelping_since=2011-07, votes=Document{{funny=0, useful=3, cool=0}}, review_count=1, name=Ronny, user_id=xgtaIn3d8d1UW4kCVdnGcg, friends=[], fans=0, averag
Document{{_id=5fa5ada4e39777c1e1846eb9, yelping_since=2011-01, votes=Document{{funny=0, useful=0, cool=0}}, review_count=6, name=Fiana Sachl, user_id=qQBLVT7eJKD1aC2KoY4fSw, friends=[], fans=0, ...}}
```



e) Określ, ile każde przedsiębiorstwo otrzymało wskazówek/napiwków (*tip*) w 2012. Wynik posortuj alfabetycznie według liczby (*tip*).

```
public void pointE () {
    MongoCollection<Document> collection = this.db.getCollection( s: "tip");

    List<Bson> results = Arrays.asList(
        Aggregates.match(Filters.and(
            Filters.gte( fieldName: "date", value: "2012-01-01"),
            Filters.lte( fieldName: "date", value: "2012-12-31"))),
        Aggregates.group( id: "$business_id", Accumulators.sum( fieldName: "count", expression: 1))
    );

    AggregateIterable iterable = collection.aggregate(results);

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

```
Document{{_id=BRcREoex-p9WLnscUjN2sQ, count=4}}
Document{{_id=Xr0SQ1i6vHgT0RBZfyzJ8Q, count=1}}
Document{{_id=_2HFbrYZFp2p59hptZeDbw, count=2}}
Document{{_id=SMUVbjzqLXEXJaGZkG134Q, count=1}}
Document{{_id=@wK--zW6Bgqgvnob8Qc77g, count=4}}
Document{{_id=CDY7AE5HWBy0TJXCakqLBQ, count=1}}
Document{{_id=7mNPYqJYtIR26Y4KQa2g1Q, count=1}}
Document{{_id=N7jFyn8Pgetj0VeJG122vw, count=3}}
Document{{_id=Lc3-XLmobyijQp0xHdEPFQ, count=2}}
Document{{_id=VLEWdvhdZS_F9Lwxn67IOQ, count=10}}
Document{{_id=sLRPUBoSA70a0YjW1q7gMA, count=3}}
Document{{_id=wx2EJUCNOCPmC0DtKb98A, count=2}}
Document{{_id=83C1eqPfmQHTy_8MC4Ug6w, count=6}}
```

f) Wyznacz, jaką średnią ocen (*stars*) uzyskała każda firma (*business*) na podstawie wszystkich recenzji. Wynik ogranicz do recenzji, które uzyskały min. 4.0 gwiazdki.

```
public void pointF () {
    MongoCollection<Document> collection = this.db.getCollection( s: "review");

    List<Bson> results = Arrays.asList(
        Aggregates.group( id: "$business_id", Accumulators.avg( fieldName: "averageStars", expression: "$stars")),
        Aggregates.match(Filters.gte( fieldName: "averageStars", value: 4.0))
    );

    AggregateIterable iterable = collection.aggregate(results);

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

```
Document{{_id=yGK0UuDg9V0rNYNakIFRw, averageStars=4.4375}}
Document{{_id=Ls7XGu8JFYfG80HaGzp-_A, averageStars=4.2}}
Document{{_id=Kx3ITctVIWmLxQ3gj6C0TQ, averageStars=4.8}}
Document{{_id=7whi5KTXSNkZaSMZ_NRAmw, averageStars=5.0}}
Document{{_id=NnoEbaGnQa44CZE_A5tBTQ, averageStars=5.0}}
Document{{_id=hALLNh4q7n6E5PTrtJZ8Q, averageStars=4.333333333333333}}
Document{{_id=Lw74h6AbGU-ze0YpAe56PA, averageStars=4.318181818181818}}
Document{{_id=tE_ytDK-ox7a-cWX7txvIA, averageStars=4.5}}
Document{{_id=Ajhc8sooWSX5cKz4LVUXA, averageStars=4.142857142857143}}
Document{{_id=B15tTS0Tie3dWA2hWn53Jw, averageStars=4.666666666666667}}
Document{{_id=07-054xA54g_tyP3A9pXZQ, averageStars=5.0}}
Document{{_id=w4zPPr-MOkqNq4zC11o8FA, averageStars=4.0}}
Document{{_id=TXysymbMWuKD_U2LI8yVbA, averageStars=4.666666666666667}}
Document{{_id=nMHuYan8e3c0No3PornJA, averageStars=4.294117647058823}}
```

g) Usuń wszystkie firmy (*business*), które posiadają ocenę (*stars*) równą 2.0.

```
public void pointG () {
    MongoCollection<Document> collection = this.db.getCollection( s: "business");
    collection.deleteMany(eq( fieldName: "stars", value: 2.0));
}
```

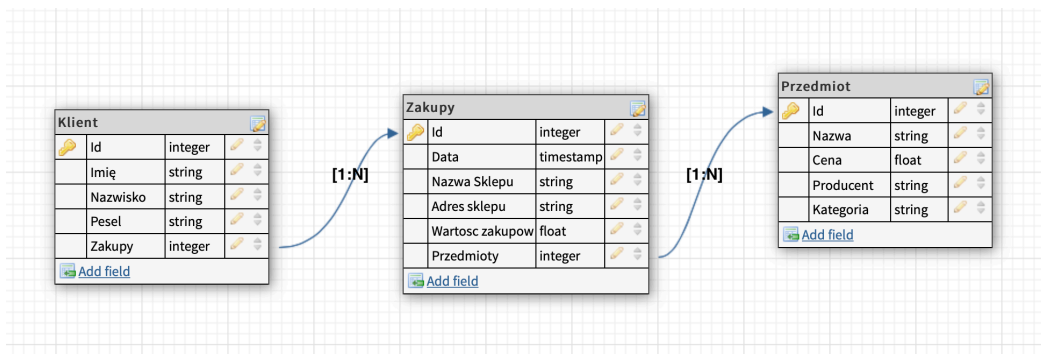
```
* db.getCollection('business').find({stars: 2.0})
IgorDzierwa1 localhost:27017 IgorDzierwa1
db.getCollection('business').find({stars: 2.0})
0.054 sec.
Fetched 0 record(s) in 54ms
```

7. Zaproponuj bazę danych składającą się z 3 kolekcji pozwalającą przechowywać dane dotyczące: klientów, zakupu oraz przedmiotu zakupu. W bazie wykorzystaj: pola proste, złożone i tablice. Zaprezentuj strukturę dokumentów w formie JSON dla przykładowych danych.

Uzasadnij swoją propozycję.

\***Dokument** — odpowiednik rekordu z podejścia relacyjnego

\***Kolekcja** — odpowiednik tabel z podejścia relacyjnego.



\***Dokument Klient** — zawiera 5 pól:

- id — klucz główny
- Imię (*string*), Nazwisko (*string*), Pesel (*string*) — pozwalają na jednoznaczne zdefiniowanie klienta.
- Zakupy (*integer*) — identyfikator paragonu. W relacyjnym modelu bazy danych byłby kluczem obcym łączącym tabelę Klient z tabelą Zakupy (*relacja jeden do wielu*).

\***Dokument Zakupy** — zawiera 5 pól:

- id — klucz główny.
- Data (*timestamp*), Nazwa Sklepu (*string*), Adres Sklepu (*string*), Wartosc zakupow (*float*) — pozwalają na jednoznaczne zdefiniowanie paragonu.
- Przedmioty (*integer*) — Identyfikator przedmiotu. W relacyjnym modelu bazy danych byłby kluczem obcym łączącym tabelę Zakupy z tabelą Przedmiot (*relacja jeden do wielu*).

\***Dokument Przedmiot** — zawiera 5 pól:

- id — klucz główny
- Nazwa (*string*), Cena (*float*), Producent (*string*), Kategoria (*string*) — pozwalają na jednoznaczne zdefiniowanie produktu.

\***Utworzenie bazy danych składającej się z 3 kolekcji (Klient, Zakupy, Przedmiot)**

```
> use ShoppingDB
switched to db ShoppingDB
```

```
> db.createCollection("Klient")
{ "ok" : 1 }
> db.createCollection("Zakupy")
{ "ok" : 1 }
> db.createCollection("Przedmiot")
{ "ok" : 1 }
```

**\*Funkcja dodająca Klienta:**

```
var addClient = (imie, nazwisko, pesel) => {
  db.Klient.insertOne({
    "imie": imie,
    "nazwisko": nazwisko,
    "pesel": pesel,
    "zakupy": []
  })
}

addClient("Jan", "Kowalski", "97021002999");
addClient("Robert", "Nowak", "99023001222");
```

**\*Funkcja dodająca Zakupy:**

```
var addShopping = (nazwaSklepu, adresSklepu, wartoscZakupow) => {
  db.Zakupy.insertOne({
    "date": new Date(),
    "nazwaSklepu": nazwaSklepu,
    "adresSklepu": adresSklepu,
    "wartoscZakupow": wartoscZakupow,
    "przedmioty": []
  })
}

addShopping("Zabka", "Dobrego Pasterza 103", 100);
addShopping("Lidl", "Czarnowiejska 90", 50);
```

**\*Funkcja dodająca Przedmiot:**

```
var addProduct = (nazwa, cena, producent, kategoria) => {
  db.Przedmiot.insertOne({
    "nazwa": nazwa,
    "cena": cena,
    "producent": producent,
    "kategoria": kategoria,
  })
}

addProduct("mleko", 10, "mullermilch", "przetwory mleczne");
addProduct("bombonierka", 40, "Lindt lindor", "slodycze");
```

**\*Funkcja dodająca Przedmioty do Zakupów:**

```
var addProductToShopping = (shoppingID, productID) => {
  db.Zakupy.update(
    { _id: ObjectId(shoppingID) },
    { $push: { przedmioty: new ObjectId(productID) } }
  )
}

addProductToShopping("5fc039fdb1dc57df7ff7def", "5fc03ce6be1dc57df7ff7df1");
addProductToShopping("5fc039fdb1dc57df7ff7def", "5fc03ce6be1dc57df7ff7df1");
addProductToShopping("5fc039fdb1dc57df7ff7def", "5fc03ce6be1dc57df7ff7df2");
addProductToShopping("5fc039fdb1dc57df7ff7def", "5fc03ce6be1dc57df7ff7df2");

addProductToShopping("5fc039fdb1dc57df7ff7df0", "5fc03ce6be1dc57df7ff7df1");
addProductToShopping("5fc039fdb1dc57df7ff7df0", "5fc03ce6be1dc57df7ff7df0");
```

**\*Funkcja dodająca Zakupy do Klientów:**

```
var addShoppingToClient = (clientID, shoppingID) => {
  db.Klient.update(
    { _id: ObjectId(clientID) },
    { $push: { zakupy: new ObjectId(shoppingID) } }
  )
};

addShoppingToClient("5fc03650be1dc57df7ff7ded", "5fc039fdb1dc57df7ff7def");
addShoppingToClient("5fc03650be1dc57df7ff7dee", "5fc039fdb1dc57df7ff7df0");
```

**\*Przykładowe dane kolekcji Klient:**

```

db.getCollection('Klient').find({})
IgorDzierwa1 localhost:27017 ShoppingDB
db.getCollection('Klient').find({})

Klient 0.001 sec.

/* 1 */
{
  "_id" : ObjectId("5fc03650be1dc57df7ff7ded"),
  "imie" : "Jan",
  "nazwisko" : "Kowalski",
  "pesel" : "97021002999",
  "zakupy" : [
    ObjectId("5fc039fdbbe1dc57df7ff7def")
  ]
}

/* 2 */
{
  "_id" : ObjectId("5fc03650be1dc57df7ff7dee"),
  "imie" : "Robert",
  "nazwisko" : "Nowak",
  "pesel" : "99023001222",
  "zakupy" : [
    ObjectId("5fc039fdbbe1dc57df7ff7df0")
  ]
}

```

**\*Przykładowe dane kolekcji Zakupy:**

```

db.getCollection('Zakupy').find({})
IgorDzierwa1 localhost:27017 ShoppingDB
db.getCollection('Zakupy').find({})

Zakupy 0.001 sec.

/* 1 */
{
  "_id" : ObjectId("5fc039fdbbe1dc57df7ff7def"),
  "date" : ISODate("2020-11-26T23:27:57.408Z"),
  "nazwaSklepu" : "Zabka",
  "adresSklepu" : "Dobrego Pasterza 103",
  "wartoscZakupow" : 100.0,
  "przedmioty" : [
    ObjectId("5fc03ce6be1dc57df7ff7df1"),
    ObjectId("5fc03ce6be1dc57df7ff7df1"),
    ObjectId("5fc03ce6be1dc57df7ff7df2"),
    ObjectId("5fc03ce6be1dc57df7ff7df2")
  ]
}

/* 2 */
{
  "_id" : ObjectId("5fc039fdbbe1dc57df7ff7df0"),
  "date" : ISODate("2020-11-26T23:27:57.412Z"),
  "nazwaSklepu" : "Lidl",
  "adresSklepu" : "Czarnowiejska 90",
  "wartoscZakupow" : 50.0,
  "przedmioty" : [
    ObjectId("5fc03ce6be1dc57df7ff7df1"),
    ObjectId("5fc03ce6be1dc57df7ff7df2")
  ]
}

```

**\*Przykładowe dane kolekcji Przedmiot:**

```

db.getCollection('Przedmiot').find({})
Close Tabierwa1 localhost:27017 ShoppingDB
db.getCollection('Przedmiot').find({})

Przedmiot 0.001 sec.

/* 1 */
{
  "_id" : ObjectId("5fc03ce6be1dc57df7ff7df1"),
  "nazwa" : "mleko",
  "cena" : 10.0,
  "producent" : "mullermilch",
  "kategoria" : "przetwory mleczne"
}

/* 2 */
{
  "_id" : ObjectId("5fc03ce6be1dc57df7ff7df2"),
  "nazwa" : "bombonierka",
  "cena" : 40.0,
  "producent" : "Lindt lindor",
  "kategoria" : "slodycze"
}

```