

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ  
РОССИЙСКОЙ ФЕДЕРАЦИИ  
Федеральное государственное бюджетное образовательное учреждение  
высшего образования  
«Кемеровский государственный университет»  
Институт цифры

## КУРСОВАЯ РАБОТА

по дисциплине «Базы данных»

Информационная система торговой организации

студента 2 курса

Тимофеева Гордея Евгеньевича

направление подготовки 09.03.03 Прикладная информатика

направленность (профиль) подготовки «Прикладная информатика в экономике»

Научный руководитель:  
кандидат технических наук, доцент  
С.Ю. Завозкин

З

Работа защищена с оценкой

«5» (отлично)

«16» увол. 2025 г.

Кемерово 2025

# **Оглавление**

Введение .....	3
1. Описание предметной области .....	4
2. Анализ предметной области .....	5
3. Создание БД .....	12
4. Выполнение запросов к БД.....	13
5. Реализация графического интерфейса для ИС.....	22
Заключение .....	25

## **Введение**

Курсовая работа направлена на закрепление знаний, полученных на курсе «Базы данных». В ходе выполнения курсовой работы с использованием изученной системы управления базами данных MongoDB и языком программирования Python была реализована информационная система торговой организации в виде web-приложения; были протестированы требуемые запросы и сделан вывод о проделанной работе.

## **1. Описание предметной области**

Торговая организация ведет торговлю в торговых точках разных типов (универмаги, магазины, киоски, лотки и т.д.), в штате которых работают продавцы. Универмаги разделены на отдельные секции. Как универмаги, так и магазины могут иметь несколько залов, универмаги, магазины, киоски могут иметь такие характеристики, как размер торговой точки, платежи за аренду, коммунальные услуги, количество прилавков и т.д. Кроме того, в универмагах и магазинах учет проданных товаров ведется персонифицировано с фиксацией имен и характеристик покупателя, чего в киосках и на лотках сделать не представляется возможным.

Заказы поставщику составляются на основе заявок, поступающих из торговых точек. На основе заявок менеджеры торговой организации выбирают поставщика, формируют заказы, в которых перечисляются наименования товаров и заказываемое их количество, которое может отличаться от запроса из торговой точки. Если указанное наименование товара ранее не поставлялось, оно пополняет справочник номенклатуры товаров. На основе маркетинговых работ постоянно изучается рынок поставщиков, в результате чего могут появляться новые поставщики и исчезать старые. При этом одни и те же товары торговая организация может получать от разных поставщиков и, естественно, по различным ценам.

Поступившие товары распределяются по торговым точкам и в любой момент можно получить такое распределение.

Продавцы торговых точек ведут продажу товаров, учитывая все сделанные продажи, фиксируя номенклатуру и количество проданного товара, а продавцы универмагов и магазинов дополнительно фиксируют имена и характеристики покупателей, что позволяет вести учет покупателей и сделанных ими покупок. В процессе торговли торговые точки вправе менять цены на товары в зависимости от спроса и предложения товаров, а также по согласованию передавать товары в другую торговую точку.

## **2. Анализ предметной области**

Для данной предметной области (розничная торговля с сотрудниками, магазинами, поставщиками, учётом остатков, продаж, а также заявками и заказами на пополнение товара) предложена следующая структура коллекций и схем хранения данных. Для каждой коллекции указаны её назначение, структура и обоснование внесённых данных и архитектурных решений.

### **1. Коллекция employees — сотрудники**

Хранит информацию о сотрудниках магазинов. Вынесение в отдельную коллекцию позволяет централизованно управлять сведениями о работниках (например, при смене ФИО или обновлении зарплаты), а также легко ассоциировать сотрудников с магазинами и продажами через уникальные идентификаторы.

Пример схемы:

- `_id`: уникальный идентификатор сотрудника (`ObjectId`)
- `name`: имя сотрудника (`String`)
- `surname`: фамилия сотрудника (`String`)
- `salary`: размер зарплаты (`Number/Double`)

Обоснование:

Вынесение сотрудников в отдельную коллекцию избавляет от дублирования информации в документах продаж и магазинах. Хранение зарплаты непосредственно здесь облегчает расчёт и обновление выплат.

### **2. Коллекция products — товары**

Хранит полные сведения о каждом товаре, включая описание и категорию. Присутствует поле `supplier_id` для связи с поставщиком, что особенно важно для анализа закупочных процессов.

Пример схемы:

- `_id`: уникальный идентификатор товара (`ObjectId`)
- `name`: наименование товара (`String`)
- `description`: описание товара (`String`)
- `category`: категория товара (`String`)
- `unit`: единица измерения (`String`)
- `price`: розничная цена (`Number/Double`)
- `supplier_id`: идентификатор поставщика (`ObjectId`)

Обоснование:

В отдельной коллекции удобно хранить параметры товаров для последующего поиска, категоризации и анализа (например, остатки товаров по категориям). Наличие `supplier_id` позволяет однозначно определить, кто поставляет данный продукт.

### 3. Коллекция sales — продажи

Документирует все факты продаж; необходима для анализа работы персонала, ассортимента и оборота конкретных магазинов.

Пример схемы:

- `_id`: уникальный идентификатор продажи (`ObjectId`)
- `store_id`: идентификатор магазина (`ObjectId`)
- `employee_id`: идентификатор продавца (`ObjectId`)
- `product_id`: идентификатор товара (`ObjectId`)
- `quantity`: количество проданных единиц (`Integer`)

- date: дата и время продажи (Date)

Обоснование:

Такая структура позволяет легко получать сводные данные по продажам: по магазину, сотруднику, товару или дате. Использование ссылок на сотрудника, магазин и товар исключает дублирование информации и облегчает аналитику.

#### **4. Коллекция stock\_balances — остатки товаров**

Хранит актуальные остатки товаров по каждому магазину. Документ содержит массив объектов с указанием товара и его количества на складе торговой точки.

Пример схемы:

- \_id: уникальный идентификатор (ObjectId)
- store\_id: идентификатор магазина (ObjectId)
- products: массив с объектами { product\_id (ObjectId), quantity (Integer) }

Обоснование:

Вложенность массива по товарам внутри документа магазина обеспечивает быстрый доступ ко всем остаткам в рамках одной торговой точки и значительно ускоряет инвентаризацию и контроль запасов.

#### **5. Коллекция stores — магазины**

Содержит сведения о торговых точках, включая детали структуры (залы, секции), технические параметры и связанные с магазином выплаты. Кроме того, хранит список сотрудников, закреплённых за данной точкой.

Пример схемы:

- \_id: уникальный идентификатор магазина (ObjectId)

- type: тип магазина (String), например, «Универмаг»
- name: наименование (String)
- section\_names: перечень секций (Array of String, опционально)
- hall\_names: перечень залов (Array of String, опционально)
- cash\_registers\_number: количество касс (Integer)
- area: площадь (Integer)
- payments: массив платежей за аренду и коммунальные услуги {  
area\_payment, utilities\_payment, payment\_date }
- employees: массив идентификаторов сотрудников (Array of ObjectId)

Обоснование:

Вся информация, специфичная для магазина, хранится единым документом, что облегчает получение полных сводок по торговым точкам (например, контроль затрат и персонала). Ссылки на объекты сотрудников позволяют быстро получить их данные при необходимости.

## **6. Коллекция suppliers — поставщики**

Хранит базовую информацию о поставщиках. Связывается с товарами через supplier\_id.

Пример схемы:

- \_id: уникальный идентификатор (ObjectId)
- name: название поставщика (String)

Обоснование:

Вынесение в отдельную коллекцию позволяет централизованно управлять поставщиками, вести историю изменения их данных, а также связывать их с товарами.

## **7. Коллекция supply\_orders — заказы на поставку**

Описывает заказы, размещённые магазином у поставщиков (через товары). Документ содержит массив позиций с информацией о количестве каждого товара.

Пример схемы:

- `_id`: уникальный идентификатор заказа (`ObjectId`)
- `store_id`: идентификатор магазина (`ObjectId`)
- `date`: дата заказа (`Date`)
- `products`: массив товаров { `product_id` (`ObjectId`), `quantity` (`Integer`) }

Обоснование:

Данная структура позволяет без избыточности учитывать закупки магазинов. Позволяет получить историю пополнения склада по магазинам и позициям.

## 8. Коллекция `supply_requests` — заявки на поставку

Фиксирует внутренние заявки магазина на пополнение ассортимента. Практически аналогична `supply_orders`, но фиксирует «хотелки» магазина (возможно, для утверждения или анализа несоответствий между заказами и заявками).

Пример схемы:

- `_id`: уникальный идентификатор заявки (`ObjectId`)
- `store_id`: идентификатор магазина (`ObjectId`)
- `date`: дата подачи заявки (`Date`)
- `products`: массив товаров { `product_id` (`ObjectId`), `quantity` (`Integer`) }

Обоснование:

Позволяет хранить историю потребностей магазинов и анализировать процесс пополнения запасов и планирования закупок.

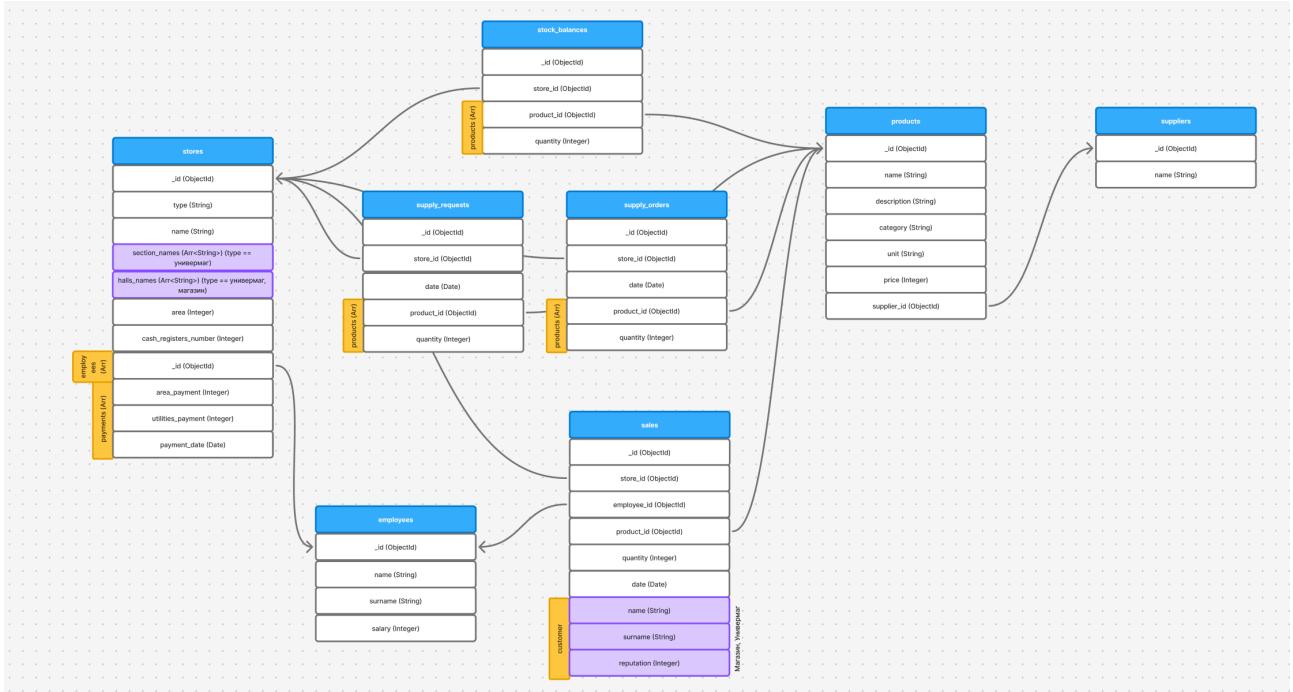
### **Вывод по структуре:**

Декомпозиция схемы на отдельные коллекции по магазинам, сотрудникам, товарам, поставщикам, продажам и операциям с запасами позволяет избежать дублирования информации и избыточного хранения данных, а также обеспечивает быструю масштабируемость и простоту внесения изменений.

Использование массивов поддокументов во всех структурных элементах, где присутствует информация о позициях (номенклатуре поставок, остатках, продажах), оправдано необходимостью частого поиска по списку товаров.

Ссылки между сущностями реализованы через ObjectId, что обеспечивает связность данных и при необходимости может быть быстро заменено на денормализованное хранение (например, дублирование имени товара или продавца) для ускорения отдельных видов аналитических запросов.

Такая организация гарантирует хранение всего объёма бизнес-данных предметной области, их актуальность и возможность последующей аналитики.



### **3. Создание БД**

БД создаётся в полуавтоматическом режиме. Сначала с помощью JS скрипта. Затем с помощью Python.

## 4. Выполнение запросов к БД

### Виды запросов в информационной системе:

- Получить перечень поставщиков, поставляющих указанный вид товара, в объеме, не менее заданного, за указанный период.

```
from datetime import datetime
from pymongo import MongoClient

client = MongoClient("mongodb://localhost:27017/")
db = client["CourseWork"]

start_date = datetime(2025, 5, 1)
end_date = datetime(2025, 5, 31, 23, 59, 59)

match_stage = {
    "$match": {
        "date": {
            "$gte": start_date,
            "$lte": end_date
        },
        "products.quantity": { "$gt": 30 }
    }
}

pipeline = [
    {
        "$unwind": "$products"
    },
    {
        "$lookup": {
            "from": "products",
            "localField": "products.product_id",
            "foreignField": "_id",
            "as": "product",
        }
    },
    match_stage,
    {
        "$lookup": {
            "from": "suppliers",
            "localField": "product.supplier_id",
            "foreignField": "_id",
            "as": "supplier",
        }
    },
    {
        "$group": {
            "_id": None,
            "unique_suppliers": { "$addToSet": "$supplier" }
        }
    },
    {
        "$project": {
            "_id": 0
        }
    }
]
```

```

]

result = db.supply_orders.aggregate(pipeline)
for doc in result:
    print(doc)

Run request1.x
Process finished with exit code 0
{'unique_suppliers': [{"_id": ObjectId('665f1108a1a1a1a1a205'), 'name': 'Gotham Accessories'}, {"_id": ObjectId('665f1108a1a1a1a1a207'), 'name': 'Galactic Trinkets'}]}

```

2. Получить общее число покупателей, купивших указанный вид товара за некоторый период в объеме, не менее заданного.

```

from datetime import datetime
from pymongo import MongoClient

client = MongoClient("mongodb://localhost:27017/")
db = client["CourseWork"]

start_date = datetime(2025, 3, 1)
end_date = datetime(2025, 5, 31, 23, 59, 59)

match_stage = {
    "$match": {
        "date": {
            "$gte": start_date,
            "$lte": end_date
        }
    }
}

# попались без фиксированных покупателей, поэтому просто число продаж
pipeline = [
    {
        "$match": {
            "quantity": { "$gt": 1 }
        }
    },
    {
        "$lookup": {
            "from": "products",
            "localField": "product_id",
            "foreignField": "_id",
            "as": "product",
        }
    },
    match_stage,
    {
        "$match": {
            "product.category": "Продукты"
        }
    }
]

result = db.sales.aggregate(pipeline)

```

```
result_list = list(result)

for doc in result_list:
    print(doc)

print("Число продаж: " + str(len(result_list)))
```

3. Получить номенклатуру и количество товаров в указанной торговой точке.

```
from datetime import datetime
from pymongo import MongoClient

client = MongoClient("mongodb://localhost:27017/")
db = client["CourseWork"]

store_id = db.stores.find_one({"name": "Магазин на Рекордной"})["_id"]
print(store_id)

pipeline = [
    {
        "$match": {
            "store_id": store_id
        }
    },
    {
        "$project": {
            "products": 1
        }
    }
]

result = db.stock_balances.aggregate(pipeline)
result_list = list(result)

for doc in result_list:
    print(doc)
```

```
Run request3
[...]
/C:/Users/princeedelen/PycharmProjects/CouseWorkMongoDB/.venv/bin/python /Users/princeedelen/PycharmProjects/CouseWorkMongoDB/requests/request3.py
665f201aa1a1a1a1a1a1a05
{"_id": ObjectId('6646754578912b964bc8c2d'), "products": [{"product_id": ObjectId('675f301aa1a1a1a1a1a309'), "quantity": 586}, {"product_id": ObjectId('675f301aa1a1a1a1a1a310'), "quantity": 786}]}
Process finished with exit code 0
```

4. Получить сведения о количестве и ценах на указанный товар по торговым точкам заданного типа.

```
from datetime import datetime

from bson import ObjectId
from pymongo import MongoClient

client = MongoClient("mongodb://localhost:27017/")
db = client["CourseWork"]
```

```

# обязательно указывать какого поставщика
product_id = db.products.find_one({
    "name": "Футболка 'I Am Groot'",
    "supplier_id": ObjectId("665f1100a1a1a1a1a1a201")
})["_id"]
# print(product_id)

product_price = db.products.find_one({
    "name": "Футболка 'I Am Groot'",
    "supplier_id": ObjectId("665f1100a1a1a1a1a1a201")
})["price"]
# print(product_price)

# указанного типа
stores_cursor = db.stores.find({"type": "Универмаг"})
stores_id = [store["_id"] for store in stores_cursor]
# print(stores_id)

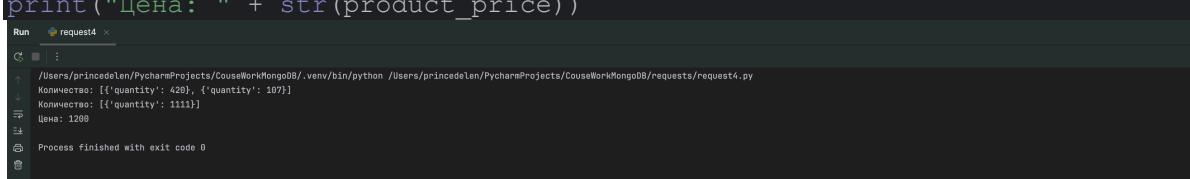
pipeline = [
    {
        "$match": {
            "products.product_id": product_id,
            "store_id": {"$in": stores_id}
        }
    },
    {
        "$project": {
            "products.quantity": 1
        }
    }
]

result = db.stock_balances.aggregate(pipeline)
result_list = list(result)

for doc in result_list:
    print("Количество: " + str(doc['products']))

print("Цена: " + str(product_price))

```



```

Run request4.x
C:\Users\prinedelen\PycharmProjects\CouseWorkMongoDB\venv\bin\python /Users/prinedelen/PycharmProjects/CouseWorkMongoDB/requests/request4.py
Количество: [{"quantity": 420}, {"quantity": 107}]
Количество: [{"quantity": 1111}]
Цена: 1288
Process finished with exit code 0

```

## 5. Получить данные о выработке отдельно взятого продавца отдельно взятой торговой точки за указанный период.

```

from datetime import datetime
from pymongo import MongoClient

client = MongoClient("mongodb://localhost:27017/")
db = client["CourseWork"]

employee_id = db.employees.find_one({
    "name": "Людмила",
    "surname": "Синицына",
})["_id"]

```

```

print(employee_id)

store_id = db.stores.find_one({"name": "Универмаг на Соборной"}) ["_id"]
print(store_id)

start_date = datetime(2025, 1, 1)
end_date = datetime(2025, 5, 31, 23, 59, 59)

match_stage = {
    "$match": {
        "date": {
            "$gte": start_date,
            "$lte": end_date
        },
        "store_id": store_id,
        "employee_id": employee_id
    }
}

pipeline = [
    match_stage,
]

result = db.sales.aggregate(pipeline)
result_list = list(result)

for doc in result_list:
    print(doc)

```

Run requestS ×

Process finished with exit code 0

## 6. Получить данные об объеме продаж указанного товара за некоторый период по торговым точкам заданного типа.

```

from datetime import datetime

from bson import ObjectId
from pymongo import MongoClient

client = MongoClient("mongodb://localhost:27017/")
db = client["CourseWork"]

product_id = db.products.find_one({
    "name": "Чайник 'Гарри'",
}) ["_id"]
# print(product_id)

# указанного типа
stores_cursor = db.stores.find({"type": "Универмаг"})
stores_id = [store["_id"] for store in stores_cursor]
# print(stores_id)

```

```

start_date = datetime(2025, 3, 1)
end_date = datetime(2025, 5, 31, 23, 59, 59)

match_stage = {
    "$match": {
        "date": {
            "$gte": start_date,
            "$lte": end_date
        },
        "store_id": {"$in": stores_id},
        "product_id": product_id
    }
}

pipeline = [
    match_stage,
    {
        "$group": {
            "_id": None, # Все документы в одну группу
            "total_quantity": {"$sum": "$quantity"}
        }
    }
]

result = db.sales.aggregate(pipeline)
result_list = list(result)

for doc in result_list:
    print(doc)

```

```

Run request6
Process finished with exit code 0

```

## 7. Получить данные о заработной плате продавцов по конкретной торговой точке.

```

from datetime import datetime

from bson import ObjectId
from pymongo import MongoClient

from InsertSales import employee_id

client = MongoClient("mongodb://localhost:27017/")
db = client["CourseWork"]

pipeline = [
    {
        "$match": {
            "name": "Универмаг на Соборной"
        }
    },
    {
        "$unwind": "$employees"
    },
    {
        "$lookup": {
            "from": "employees",
            "localField": "employees",

```

```

        "foreignField": "_id",
        "as": "employee",
    }
},
{
    "$project": {
        "employee.salary": 1
    }
}
]

result = db.stores.aggregate(pipeline)
result_list = list(result)

for doc in result_list:
    print(doc)

```

The screenshot shows the PyCharm IDE with a code editor and a terminal window. The terminal output displays the results of the aggregation query, showing two documents from the 'stores' collection, each containing a single document from the 'employees' collection with the salary field projected.

## 8. Получить сведения о поставках определенного товара указанным поставщиком за некоторый период.

```

from datetime import datetime
from bson import ObjectId
from pymongo import MongoClient

client = MongoClient("mongodb://localhost:27017/")
db = client["CourseWork"]

start_date = datetime(2025, 3, 1)
end_date = datetime(2025, 5, 31, 23, 59, 59)

match_date_stage = {
    "$match": {
        "date": {
            "$gte": start_date,
            "$lte": end_date
        }
    }
}

# обязательно указывать какого поставщика
product_id = db.products.find_one({
    "name": "Футболка 'I Am Groot'",
    "supplier_id": ObjectId("665f1100a1a1a1a1a1a201")
})["_id"]
print(product_id)

# попались без фиксированных покупателей, поэтому просто число продаж
pipeline = [
    match_date_stage,
    {
        "$match": {
            "products.product_id": product_id,
        }
    }
]

```

```

]

result = db.supply_orders.aggregate(pipeline)
result_list = list(result)

for doc in result_list:
    print(doc)

```

```

Process finished with exit code 0

```

## 9. Получить сведения о поставках товаров по указанному номеру заказа.

```

from datetime import datetime

from bson import ObjectId
from pymongo import MongoClient

client = MongoClient("mongodb://localhost:27017/")
db = client["CourseWork"]

# номер заказа - его id
pipeline = [
    {
        "$match": {
            "_id": ObjectId("6846754141f00dde3bb185d3"),
        }
    }
]

result = db.supply_orders.aggregate(pipeline)
result_list = list(result)

for doc in result_list:
    print(doc)

```

```

Process finished with exit code 0

```

## 10.Получить сведения о наиболее активных покупателях по всем торговым точкам.

```

from pymongo import MongoClient

client = MongoClient("mongodb://localhost:27017/")
db = client["CourseWork"]

# номер заказа - его id
pipeline = [
    {
        "$match": {
            "customer": {"$exists": True} # Только документы, где есть customer
        }
    },
    {

```

```
        "$group": {
            "_id": {
                "name": "$customer.name",
                "surname": "$customer.surname"
            },
            "count": {"$sum": 1}
        },
    {
        "$sort": {"count": -1} # Сортировка по убыванию количества
    }
]

result = db.sales.aggregate(pipeline)
result_list = list(result)

for doc in result_list:
    print(f'{doc["_id"]['name']} {doc["_id"]['surname']}: {doc['count']}')

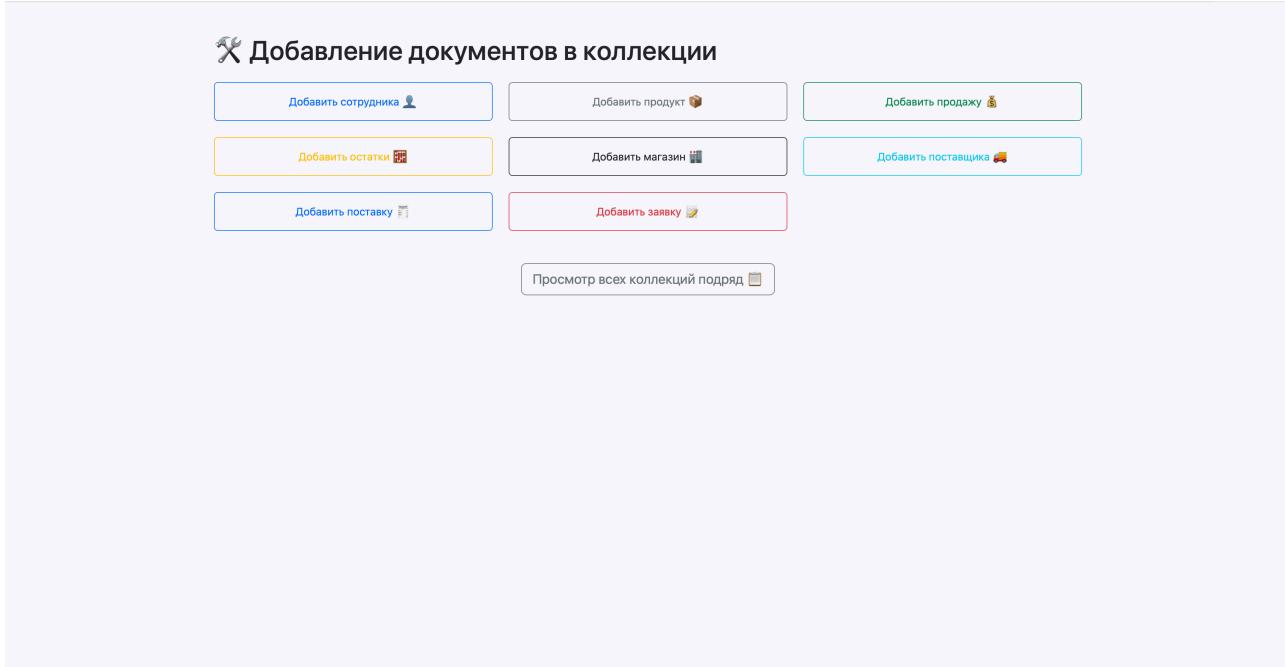
Run request10
```



The screenshot shows the PyCharm IDE interface. At the top, there's a toolbar with icons for Run, Stop, and others. Below it is a navigation bar with 'Run' and 'request10'. The main area is a code editor containing the provided MongoDB aggregation pipeline and a Python script to execute it. Below the code editor is a terminal window titled 'request10' showing the command run and its output. The output lists names and surnames along with their counts, sorted by count in descending order.

Name	Surname	Count
Дмитрий	Смирнов(а)	4
Анна	Попов(а)	4
Илья	Кузнецов(а)	4
Анна	Кузнецова(а)	4
Дмитрий	Попов(а)	3
Мария	Попова(а)	3
Мария	Смирнов(а)	3
Илья	Иванов(а)	2

## 5. Реализация графического интерфейса для ИС



Главная страница выглядит следующим образом. Сразу представлены кнопки добавления документов в коллекции. Использован Python + Flask и HTML + bootstrap.

### customers

[Добавить документ](#)

Name	Characteristics	Действия
Иван Иванов	Постоянный клиент, предпочитает дорогие товары	<a href="#">Изменить</a> <a href="#">Удалить</a>
Мария Смирнова	Часто участвует в акциях и распродажах	<a href="#">Изменить</a> <a href="#">Удалить</a>
Алексей Петров	Оптовый покупатель, делает заказы на группу людей	<a href="#">Изменить</a> <a href="#">Удалить</a>
Екатерина Соколова	Любит экзотические товары, редко совершает покупки	<a href="#">Изменить</a> <a href="#">Удалить</a>
Дмитрий Кузнецов	Совершает частые мелкие покупки, всегда проверяет качество	<a href="#">Изменить</a> <a href="#">Удалить</a>
Андрей Львович	Постоянный покупатель	<a href="#">Изменить</a> <a href="#">Удалить</a>

[Назад к главной](#)

Так выглядит пример добавления

### 👤 Добавление сотрудника

Имя

Test

Фамилия

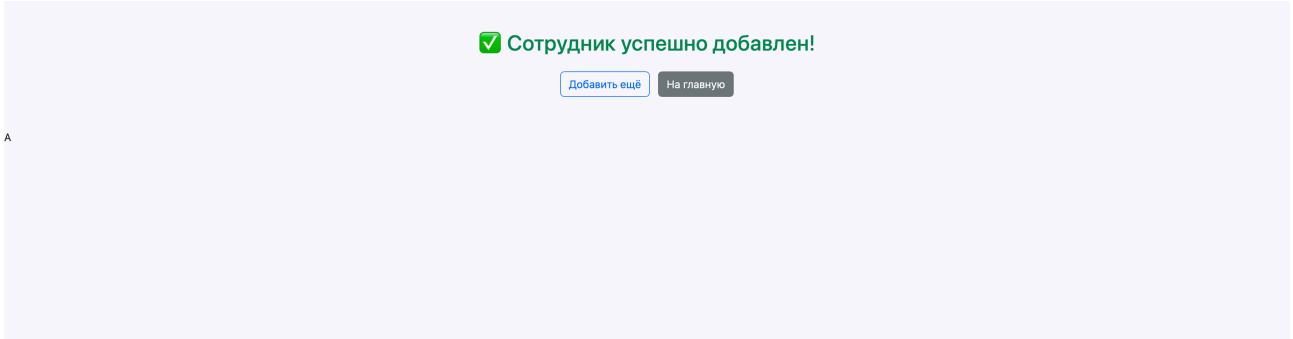
Testtest

Зарплата

2000

[Сохранить](#)

[Назад](#)



Вот так выглядит переход на «все коллекции»:

[← Назад](#)

## Все коллекции

**Сотрудники (employees)**

ID	name	surname	salary	Действие
665f101aa1a1a1a1a1a1a101	Иван	Жмых	40000	<a href="#">Изменить</a> <a href="#">Удалить</a>
665f101aa1a1a1a1a1a1a102	Василий	Петров	25000	<a href="#">Изменить</a> <a href="#">Удалить</a>
665f101aa1a1a1a1a1a1a103	Роман	Бондарь	60000	<a href="#">Изменить</a> <a href="#">Удалить</a>
665f101aa1a1a1a1a1a1a104	Питер	Паркер	20000	<a href="#">Изменить</a> <a href="#">Удалить</a>
665f101aa1a1a1a1a1a1a105	Сергей	Смит	35000	<a href="#">Изменить</a> <a href="#">Удалить</a>
665f101aa1a1a1a1a1a1a106	Георгий	Филимонов	50000	<a href="#">Изменить</a> <a href="#">Удалить</a>
665f101aa1a1a1a1a1a1a107	Анна	Евтушенко	20000	<a href="#">Изменить</a> <a href="#">Удалить</a>
665f101aa1a1a1a1a1a1a108	Марина	Гоголь	25000	<a href="#">Изменить</a> <a href="#">Удалить</a>
665f101aa1a1a1a1a1a1a109	Олег	Козлов	27000	<a href="#">Изменить</a> <a href="#">Удалить</a>
665f101aa1a1a1a1a1a1a10a	Андрей	Калинин	33000	<a href="#">Изменить</a> <a href="#">Удалить</a>
665f101aa1a1a1a1a1a1a10b	Алексей	Миронов	45000	<a href="#">Изменить</a> <a href="#">Удалить</a>
665f101aa1a1a1a1a1a1a10c	Людмила	Синицина	30000	<a href="#">Изменить</a> <a href="#">Удалить</a>
665f101aa1a1a1a1a1a1a10d	Татьяна	Ларина	27000	<a href="#">Изменить</a> <a href="#">Удалить</a>
68469d9d2cd7ae12faa6d7	Test	Testtest	2000	<a href="#">Изменить</a> <a href="#">Удалить</a>

**Продукты (products)**

ID	name	description	category	unit	price	supplier_id	Действие
675f301aa1a1a1a1a1a1a301	Футболка 'I Am Groot'	Футболка с принтом из Стражей Галактики	Одежда	шт	1200	665f1100a1a1a1a1a1a201	<a href="#">Изменить</a> <a href="#">Удалить</a>

Сразу же можем видеть добавленного сотрудника

### Редактировать документ (employees)

name

surname

salary

[Сохранить](#) [Отмена](#)

68469d9d2cd7ae12faa6d7	ExampleTest	Testtest	2000	<a href="#">Изменить</a> <a href="#">Удалить</a>
И удалим документ из коллекции.				
Удалить этот документ?				
<a href="#">Отменить</a> <a href="#">OK</a>				
665f101aa1a1a1a1a1a1a107	Олег	Козлов	27000	<a href="#">Изменить</a> <a href="#">Удалить</a>
665f101aa1a1a1a1a1a1a108	Андрей	Калинин	33000	<a href="#">Изменить</a> <a href="#">Удалить</a>
665f101aa1a1a1a1a1a1a109	Алексей	Миронов	45000	<a href="#">Изменить</a> <a href="#">Удалить</a>
665f101aa1a1a1a1a1a1a10a	Людмила	Синицина	30000	<a href="#">Изменить</a> <a href="#">Удалить</a>
665f101aa1a1a1a1a1a1a10b	Татьяна	Ларина	27000	<a href="#">Изменить</a> <a href="#">Удалить</a>
665f101aa1a1a1a1a1a1a10c				
665f101aa1a1a1a1a1a1a10d				
68469d9d2cd7ae12faa6d7	ExampleTest	Testtest	2000	<a href="#">Изменить</a> <a href="#">Удалить</a>

## Коллекция: Сотрудники

Id	name	surname	salary	Действия
665f101aa1a1a1a1a1a1a1a101	Иван	Жмыых	40000	<button>Изменить</button> <button>Удалить</button>
665f101aa1a1a1a1a1a1a1a102	Василий	Петров	25000	<button>Изменить</button> <button>Удалить</button>
665f101aa1a1a1a1a1a1a1a103	Роман	Бондарь	60000	<button>Изменить</button> <button>Удалить</button>
665f101aa1a1a1a1a1a1a1a104	Питер	Паркер	20000	<button>Изменить</button> <button>Удалить</button>
665f101aa1a1a1a1a1a1a1a105	Сергей	Смит	35000	<button>Изменить</button> <button>Удалить</button>
665f101aa1a1a1a1a1a1a1a106	Георгий	Филимонов	50000	<button>Изменить</button> <button>Удалить</button>
665f101aa1a1a1a1a1a1a1a107	Анна	Евтушенко	20000	<button>Изменить</button> <button>Удалить</button>
665f101aa1a1a1a1a1a1a1a108	Марина	Гоголь	25000	<button>Изменить</button> <button>Удалить</button>
665f101aa1a1a1a1a1a1a1a109	Олег	Козлов	27000	<button>Изменить</button> <button>Удалить</button>
665f101aa1a1a1a1a1a1a1a10a	Андрей	Калинин	33000	<button>Изменить</button> <button>Удалить</button>
665f101aa1a1a1a1a1a1a1a10b	Алексей	Миронов	45000	<button>Изменить</button> <button>Удалить</button>
665f101aa1a1a1a1a1a1a1a10c	Людмила	Синицина	30000	<button>Изменить</button> <button>Удалить</button>
665f101aa1a1a1a1a1a1a1a10d	Татьяна	Ларина	27000	<button>Изменить</button> <button>Удалить</button>

[Назад](#)

## **Заключение**

Разработанная структура базы данных и сопутствующая архитектура проекта обеспечивают удобное и масштабируемое хранение информации для системы розничной торговли. Чёткое разделение сущностей на отдельные коллекции позволяет избежать дублирования данных, упрощает внесение изменений и поддерживает высокую производительность при работе с большими объёмами информации.

Использование связей через ObjectId обеспечивает целостность данных и гибкость для дальнейшего развития проекта. Подготовленные модели, тестовые данные и автоматизированные скрипты наполнения базы данных создают удобную инфраструктуру для разработки, тестирования и последующего внедрения системы.

Такое решение гарантирует актуальность данных, прозрачность бизнес-процессов и возможность эффективной аналитики, что особенно важно для оптимизации работы розничной сети.

## **Источники**

- Метанит. MongoDB. [Электронный ресурс]. — Режим доступа: <https://metanit.com/nosql/mongodb/>, свободный. — Дата обращения: 08.06.2025.