**Федеральное государственное бюджетное образовательное учреждение**

**высшего образования**

**«Саратовский государственный технический университет**

**имени Гагарина Ю.А.»**

Институт прикладных информационных технологий и коммуникаций

Направление «Информационные системы и технологии»

Кафедра «Прикладные информационные технологии»

**КОНТРОЛЬНАЯ РАБОТА**

по дисциплине «Нереляционные базы данных»

Выполнил студент группы

б1-ИФСТипу-21

заочной формы обучения

Рамазанов Павел Андреевич

Номер зачетной книжки: 202583

Проверил: к.т.н., доцент кафедры ПИТ

Шварц А.Ю.

Саратов, 2022

**Задание № 1**

**Назначение базы данных. Анализ предметной области. Техническое задание на разработку БД.**

**Учебный пример разработки базы данных**

Рассмотрим основные моменты проектирования базы данных "Сотрудники офисного отдела".

**Назначение базы данных и анализ предметной области**

База данных предназначена для автоматизации основных информационных процессов в офисе. Для каждого сотрудника в базе данных будут созданы следующие поля:

* Id.
* Имя.
* Фамилия.
* Возраст.
* Должность.
* Заработная плата.
* Личный номер сотрудника.
* Интересы/хобби.
* Наличие автомобиля, корпоративного ПК (необязательное поле).
* Домашний адрес сотрудника.

**Составление технического задания**

На основе проведённого анализа предметной области составляем техническое задание (ТЗ) в соответствие с ЕСПД ГОСТ 19.201-78.

––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––

**Оформление ТЗ - в соответствии с ГОСТ 19.201-78**

Техническое задание

ВВЕДЕНИЕ

Наименование - база данных «Сотрудники офиса)» (далее - БД).

Краткая характеристика - нереляционная база данных для обеспечения деятельности сотрудников офиса и автоматизации поиска сотрудников по необходимым параметрам.

1. ОСНОВАНИЕ ДЛЯ РАЗРАБОТКИ

Задание преподавателя на выполнение учебной лабораторной работы.

1. НАЗНАЧЕНИЕ РАЗРАБОТКИ

БД предназначена для автоматизации основных видов работ сотрудников офиса. Она должна представлять собой информационную систему обеспечения деятельности сотрудников компании, работающих как в офисе, так и удаленно.

**Задание № 2**

**Разработка базы данных**

Разработка БД будет производиться в экосистеме нереляционных баз данных MongoDB. Для наглядности будет продемонстрировано два примера работы с данными в проектируемой БД: в консольном терминале Mongosh и графической среде работы с БД MongoDB Compass.

После установки необходимого ПО нужно сделать следующее:

1. Запустить локальный сервер Mongod (рис. 2.1).

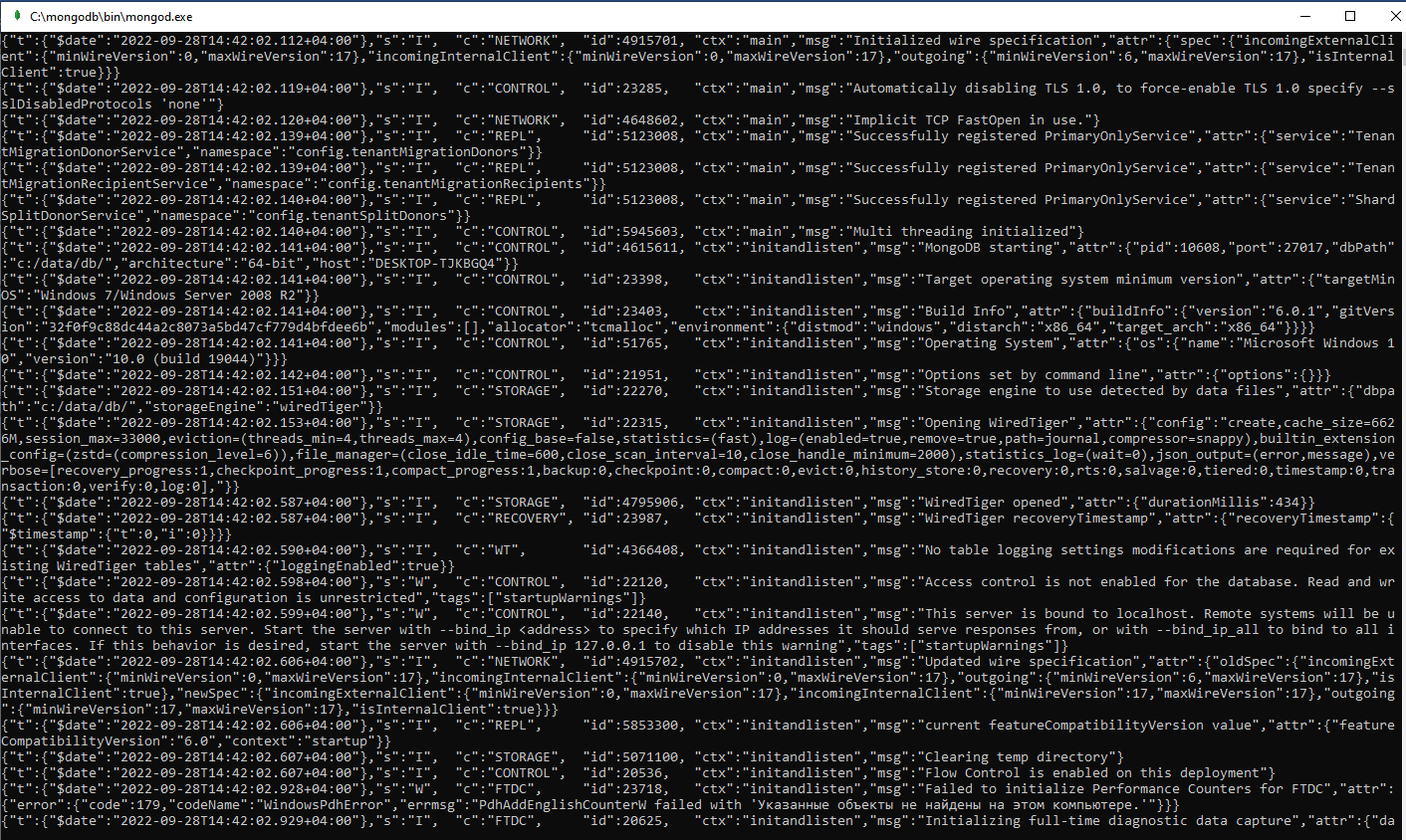


Рисунок 2.1 – Запуск локального сервера MongoDB Mongod

1. Запустить консольный терминал Mongosh, в котором будет проходить процесс создания БД (рис. 2.2, 2.3).

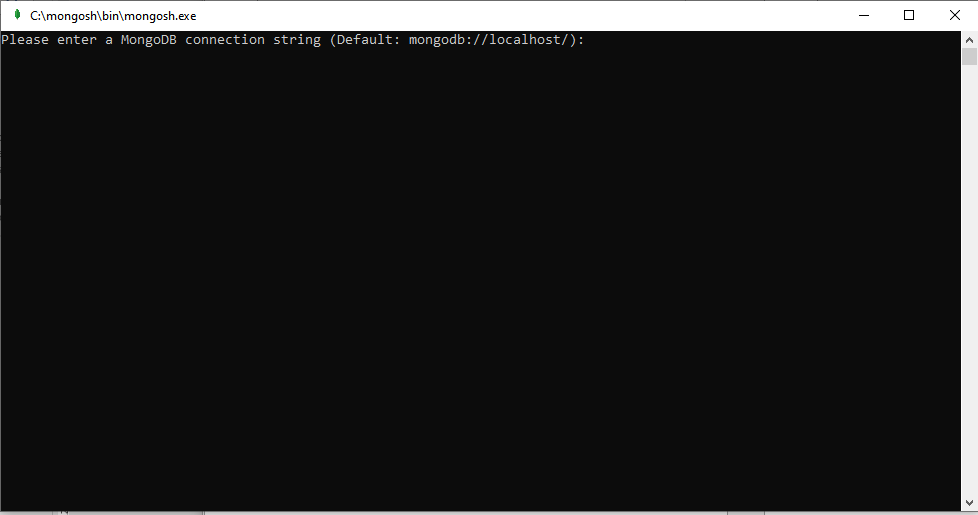


Рисунок 2.2 – Подключение к запущенному локальному серверу MongoDB localhost

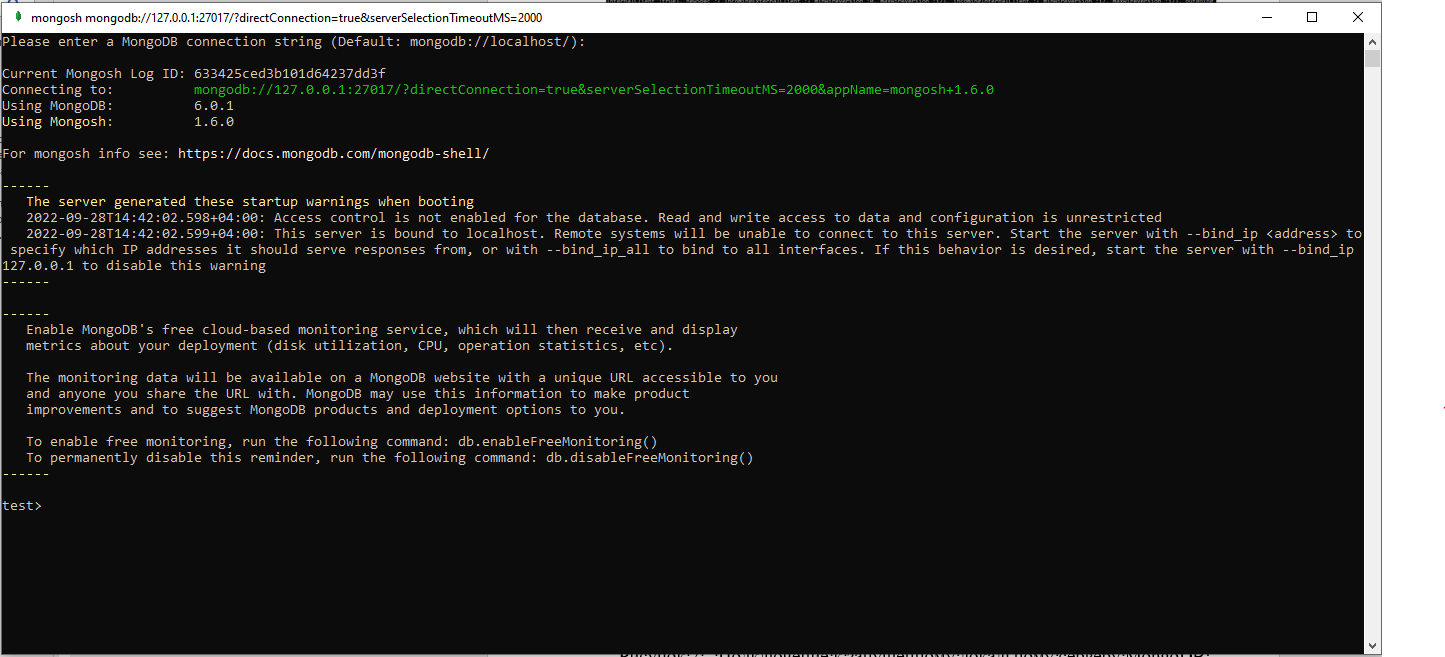


Рисунок 2.3 – Интерфейс рабочего окна консольного терминала Mongosh

Для того чтобы работать с данными в проектируемой БД, необходимо сначала её создать, используя команду **use \*имя БД\***. В случае данной контрольной работы база данных будет названа **officedb**. На рисунке 2.4 показан этап создания БД officedb на локальном сервере.

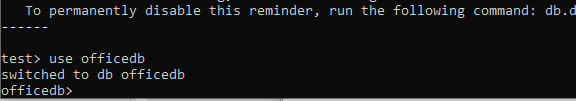


Рисунок 2.4 – Создание БД на локальном сервере

Далее, требуется заполнить базу данных необходимыми сведениями о сотрудниках офиса. На основании задания № 1 данные о сотрудниках будут вноситься по следующему шаблону на языке BSON (Binary JSON):

{

"\_id": \*int\*,

"name": \*string\*,

"surname": \*string\*,

"age": \*int\*,

"position": \*string\*,

"salary": \*int\*,

"personal\_number": \*int\*,

"interests": \*string\_array["",""…]\*,

"items": \*document{}\*, // (optional)

"address": {

"city": \*string\*,

"street": \*string\*,

"building": \*string\*,

"apartment": \*int\*

}

},

где \*слово\* - тип данных присваиваемого значения ключу документа.

По заданию требуется создать не менее 30-ти документов в одной БД. MongoDB предоставляет два метода создания документов в БД:

1. **db.\*dbname\*.insertOne({\*document\*})** – создание одновременно одного документа в БД.
2. **db.\*dbname\*.insertMany([{\*document\*}, {\*document\*}, …])** – создание одновременно двух и более документов в БД.

Так как нецелесообразно создавать 30 документов поочередно, будет использован метод insertMany() добавления документов в БД. Для удобства документы сначала будут внесены в файл с расширением .js в программе Notepad++ в качестве шаблона, после чего загружены в БД при помощи команды **load("путь\_к\_файлу")**.

Добавление документов шаблоном продемонстрировано на рисунке 2.5.

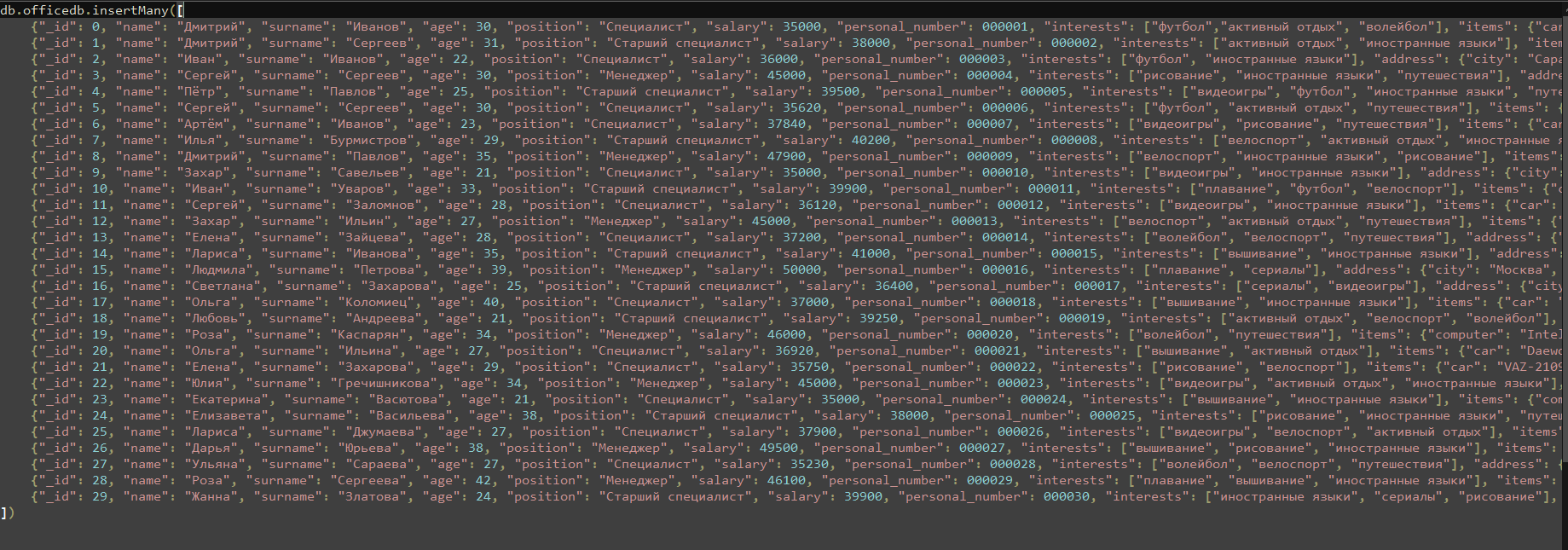


Рисунок 2.5 – Создание документов для добавления в БД во внешнем файле officedb\_docs.js

Шаблон для внесения документов в коллекцию (БД) заполнен. На рисунке 2.6 показано применение метода load() для добавления документов в БД из внешнего файла.



Рисунок 2.6 – Загрузка документов коллекции из внешнего файла

Документы внесены в БД. Для того чтобы удостовериться в том, что БД заполнена, нужно воспользоваться методом **db.\*dbname\*.find()**, который отображает все документы коллекции, что изображено на рисунке 2.7.



Рисунок 2.7 – Проверка наличия документов в коллекции

Теперь проверка документов в БД будет осуществлена в графической среде MongoDB Compass (второй способ, описываемый в начале контрольной работы). На рисунке 2.8 изображен стартовый экран программы. Нажав на кнопку Connect, администратор БД подключается к локальному серверу (тому, который запущен через Mongod).

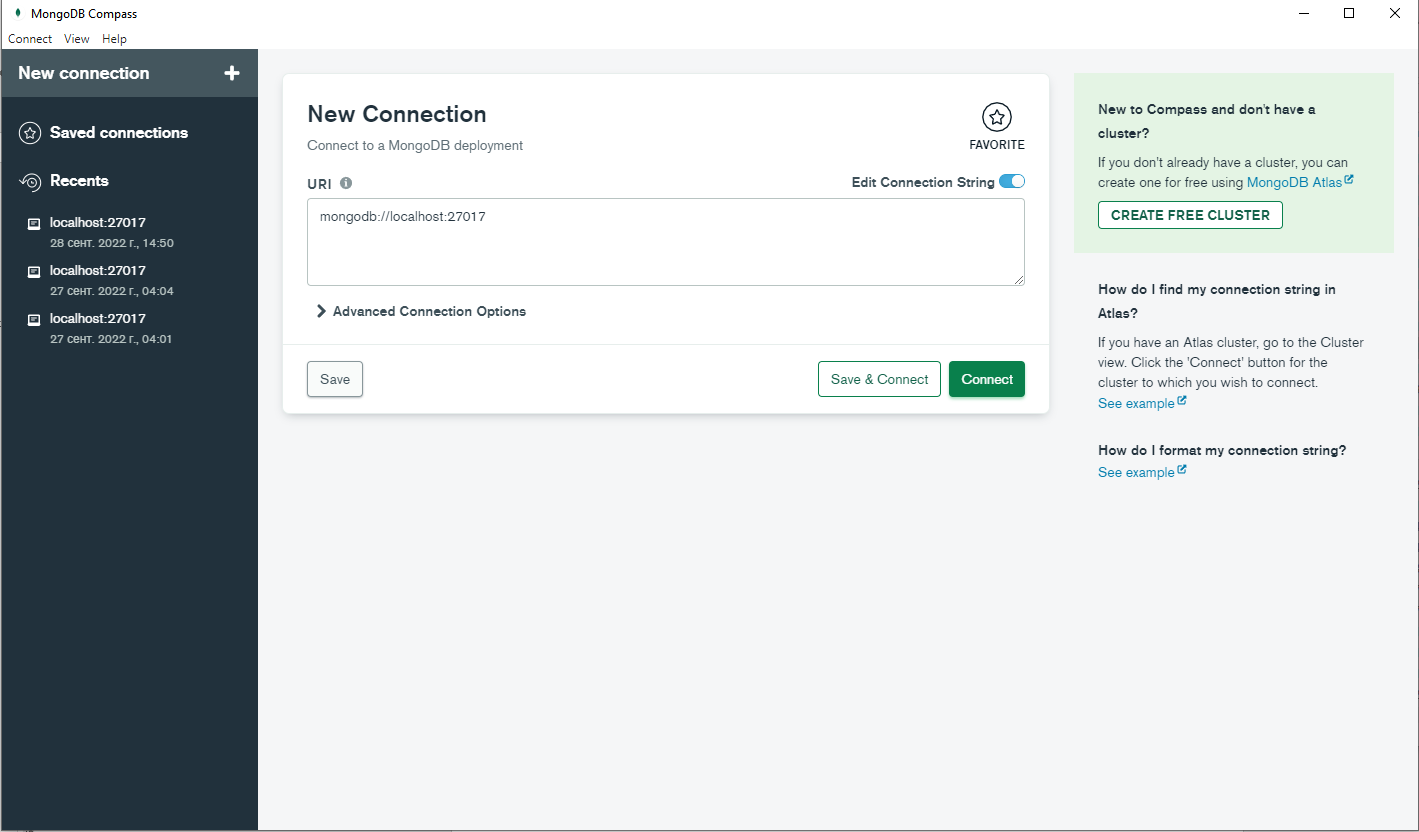


Рисунок 2.8 – Заглавный экран MongoDB Compass

В разделе Databases (рис. 2.9) можно посмотреть все активные базы данных (admin, config и local – автоматически созданные БД).

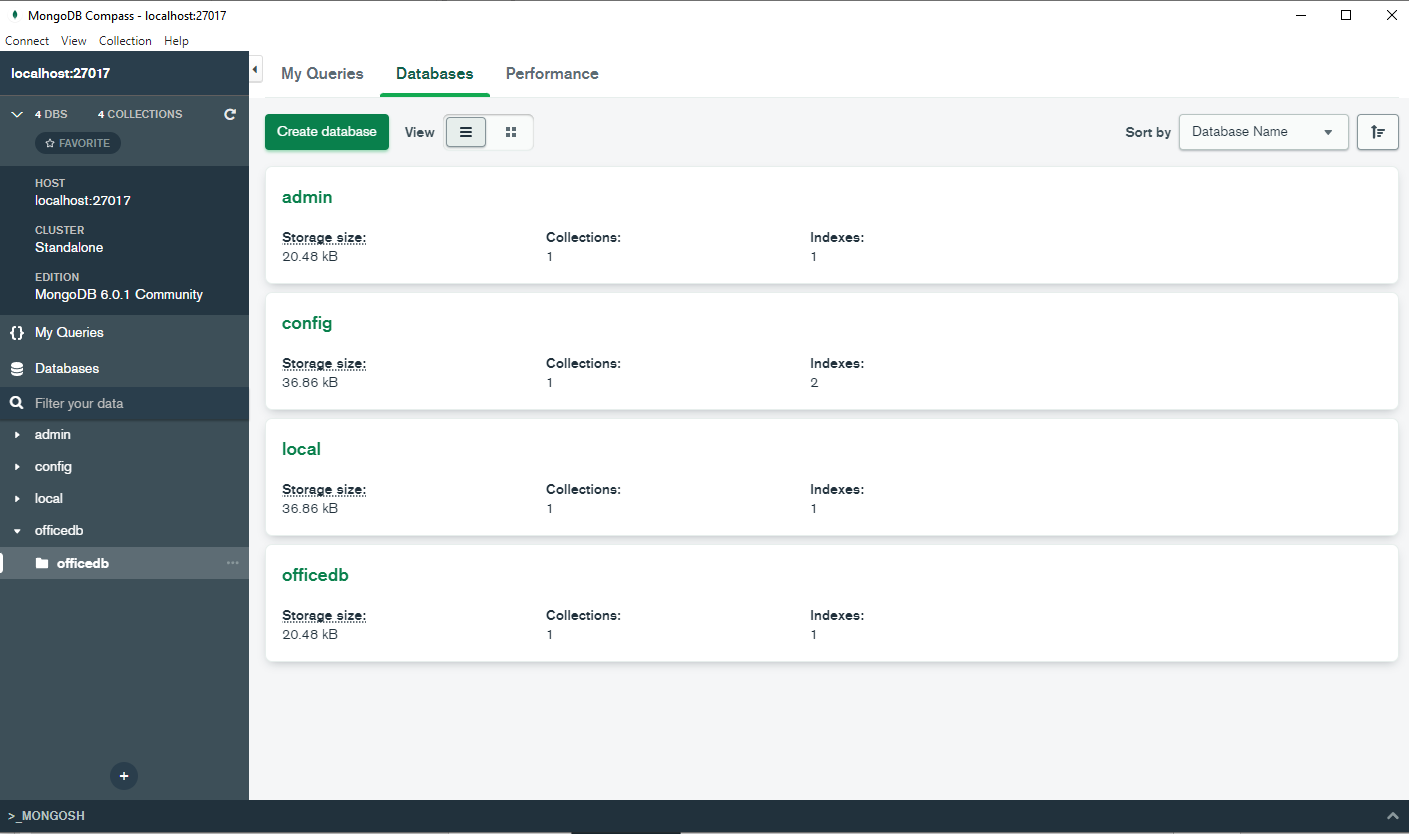


Рисунок 2.9 – Список активных БД на локальном сервере MongoDB

Открыв БД officedb, можно увидеть все добавленные документы, загруженные из внешнего файла (рис. 2.10).

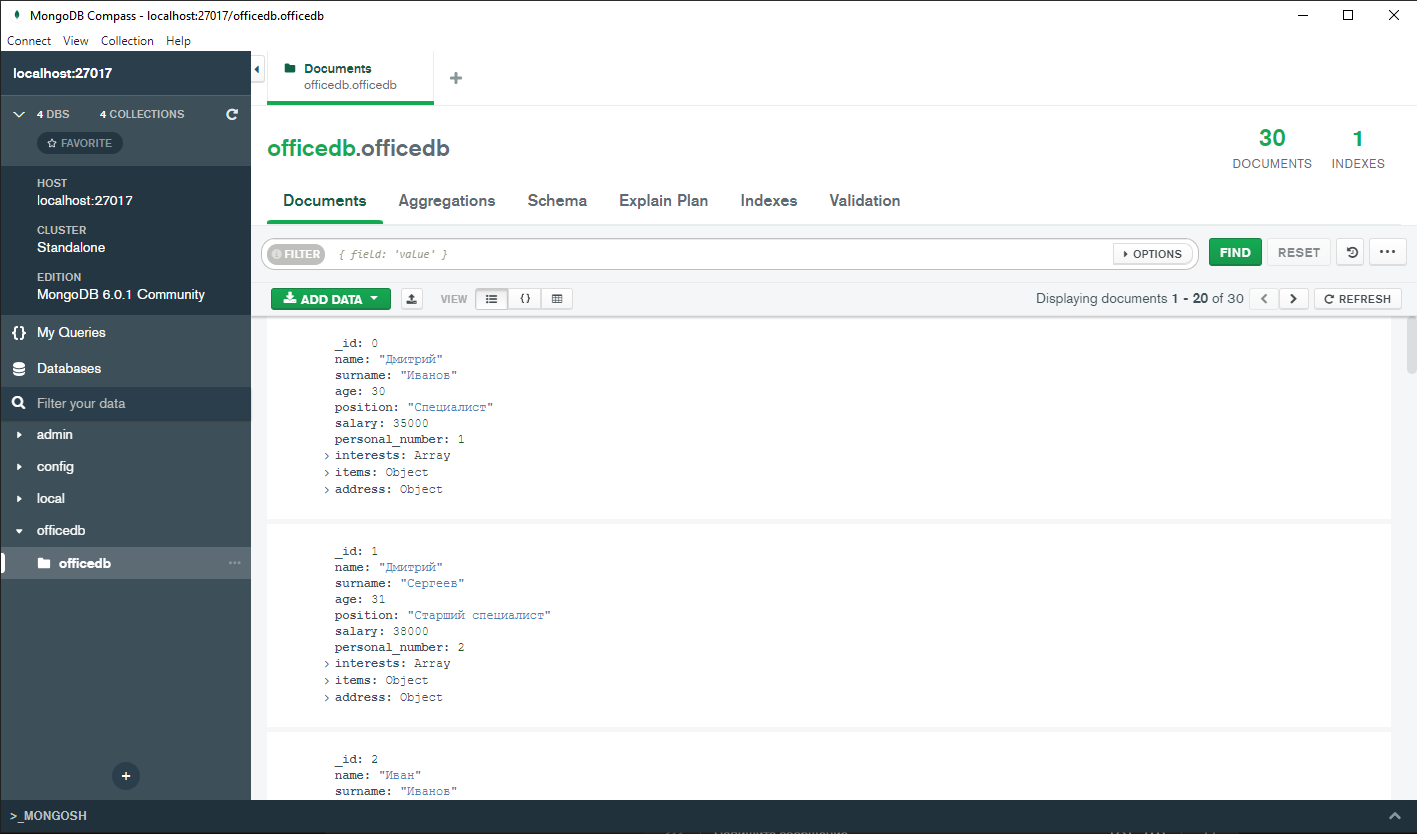


Рисунок 2.10 – Отображение существующих документов в коллекции officedb

База данных создана. Теперь нужно создать несколько запросов к БД для проверки её работоспособности.

**Задание № 3**

**Тестирование базы данных**

Для демонстрации работы созданной базы данных напишем несколько запросов на языке JavaScript в формате JSON, в том числе с использованием метода агрегации MongoDB.

Запрос № 1. В компании скоро состоится корпоративный матч по футболу. Руководителям отделов поручено выяснить, кто из сотрудников имеет спортивный разряд по данному виду спорта (в массиве **interests** элемент **футбол** стоит на первом месте). Сделать это можно следующим запросом:

**db.officedb.find({"interests.0": "футбол"}** **, {name: 1, surname: 1, position: 1}).**

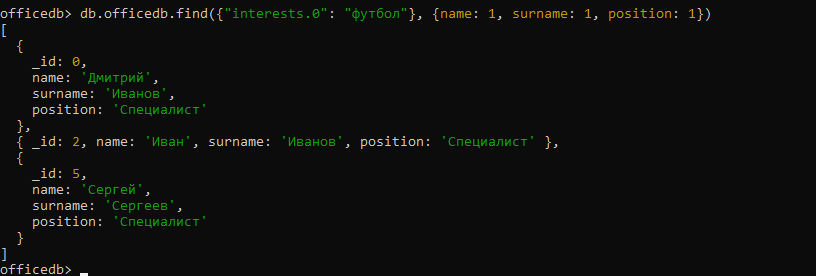


Рисунок 3.1 – Результат запроса № 1 (3 сотрудника имеют спортивный разряд по футболу)

Запрос № 2. На компьютере руководителя экономического отдела компании сломалась программа документооборота, в связи с чем список сотрудников не обновляется. Нужно восстановить порядок, отсортировав фамилии работников по алфавиту:

**db.officedb.find({}, {surname: 1, name: 1, \_id: 0}).sort({surname: 1}).**

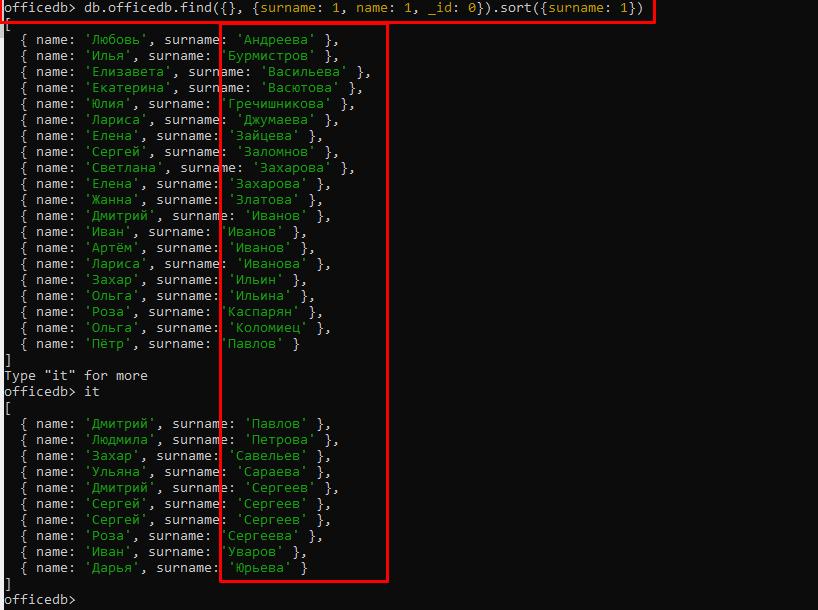


Рисунок 3.2 – Результат запроса № 2 (список сотрудников отсортирован по алфавиту)

Запрос № 3. В связи с планируемой инвентаризацией корпоративной техники требуется вызвать в офис тех сотрудников, кому на удаленную работу выданы компьютеры от производителя «Intel». Для того, чтобы их вызвать, требуется сделать запрос к базе данных:

**db.officedb.find({"items.computer": "Intel"}, {\_id: 0, name: 1, surname: 1, personal\_number: 1, "items.computer": 1}).**

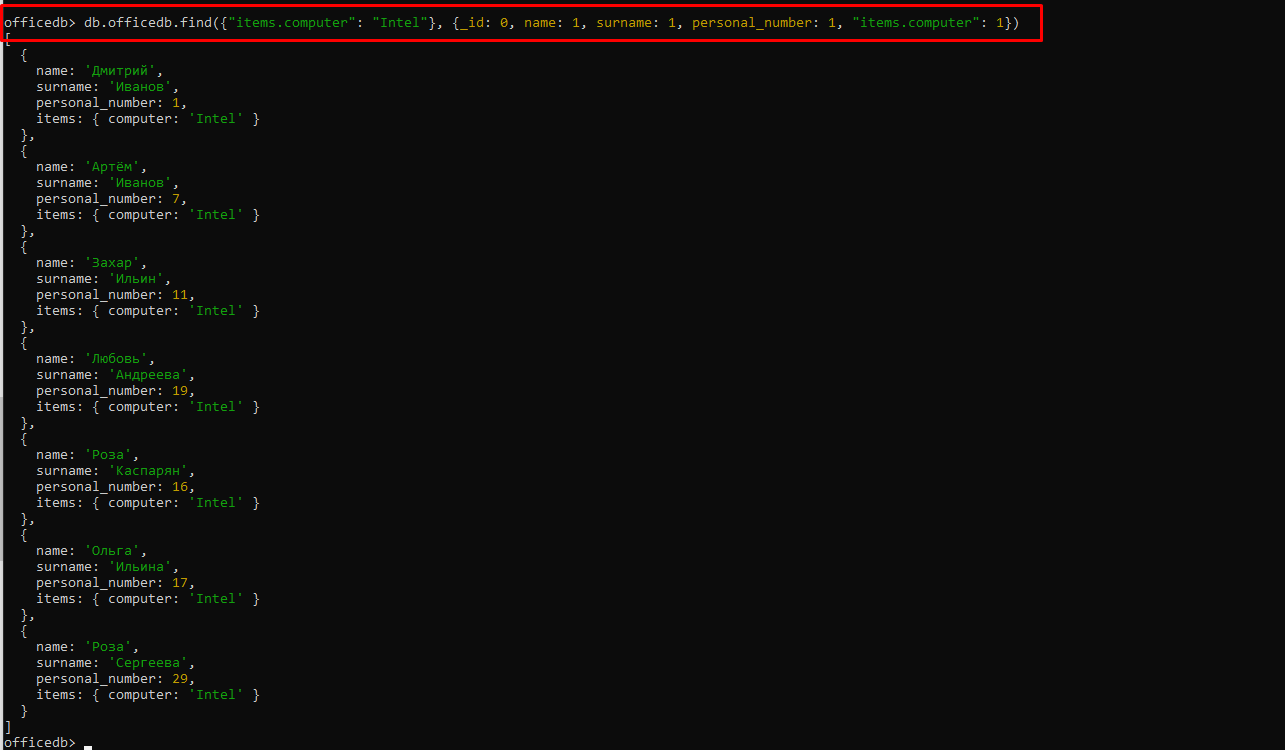


Рисунок 3.3 – Результат запроса № 3 (выборка сотрудников с компьютерами марки «Intel»)

Запрос № 4. В рамках корпоративной традиции руководство дарит теплые пледы тем сотрудникам, кто живет на низких этажах. Необходимо сделать выборку по таким сотрудникам (найти минимум среди элементов **apartment** объектов **address**):

**db.officedb.aggregate([{$group: {\_id: "Least\_Apartment", apartment: {$min: "$address.apartment"}}}]);**

// [ { \_id: 'Least\_Apartment', apartment: 12 } ]

**db.officedb.find({"address.apartment": 12}, {\_id: 0, name: 1, surname: 1, position: 1, address: 1}).**

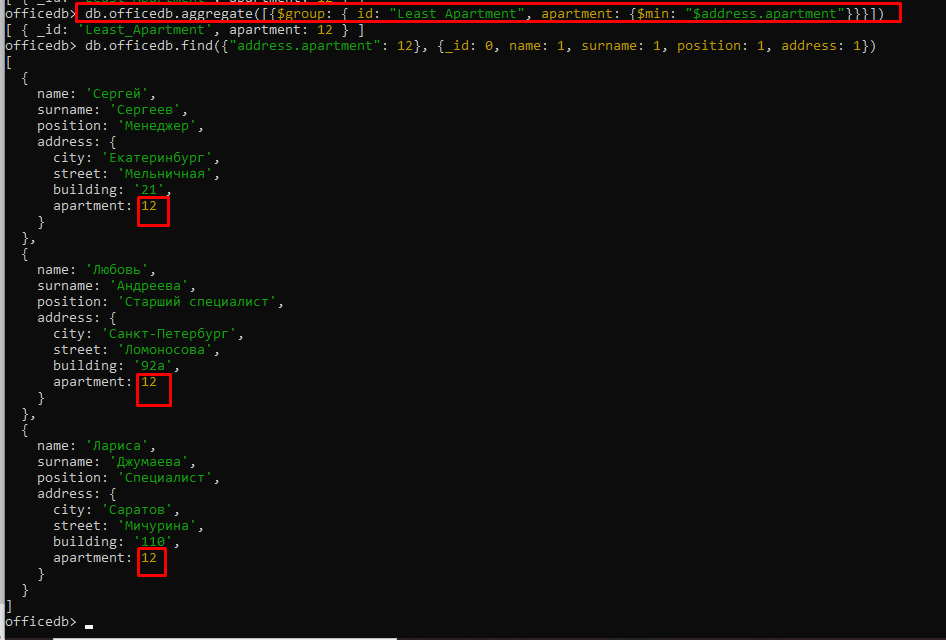


Рисунок 3.4 – Результат запроса № 4 (Выборка сотрудников при помощи метода агрегации)

Запрос № 5. Для ведения бухгалтерской отчетности отделу экономистов поручили суммировать ежемесячный доход всех сотрудников. Сделать это можно следующим запросом:

**db.officedb.aggregate([{$group: {\_id: "Employees", total\_salary: {$sum: "$salary"}}}]).**

Результат запроса представлен на рисунке 3.5.

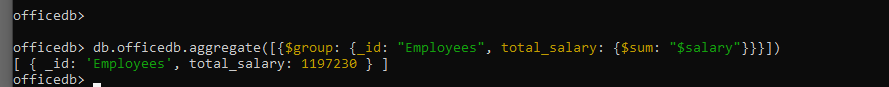


Рисунок 3.5 - Результат запроса № 5 (подсчитана сумма зарплат всех сотрудников)

Запрос № 6. Отдел кадров собирает статистику по отделу. Сотрудникам необходимо вычислить средний возраст всех работников. Для этого будет написан запрос:

**db.officedb.aggregate([{$group: {\_id: "Employees", avg\_age: {$avg: "$age"}}}]).**



Рисунок 3.6 – Результат запроса № 6 (средний возраст сотрудников составил ~30 лет)

Запрос № 7. Руководство поручило отделу бухгалтерии высчитать минимальную заработную плату среди сотрудников:

**db.officedb.aggregate([{$group: {\_id: "Employees", lst\_salary: {$min: "$salary"}}}]).**



Рисунок 3.7 – Результат запроса № 7 (вычислена наименьшая заработная плата среди сотрудников)

Запрос № 8. Теперь требуется вычислить наибольшую заработную плату среди сотрудников компании:

**db.officedb.aggregate([{$group: {\_id: "Employees", lst\_salary: {$max: "$salary"}}}])**



Рисунок 3.8 – Результат запроса № 8 (вычислена наибольшая заработная плата среди сотрудников)