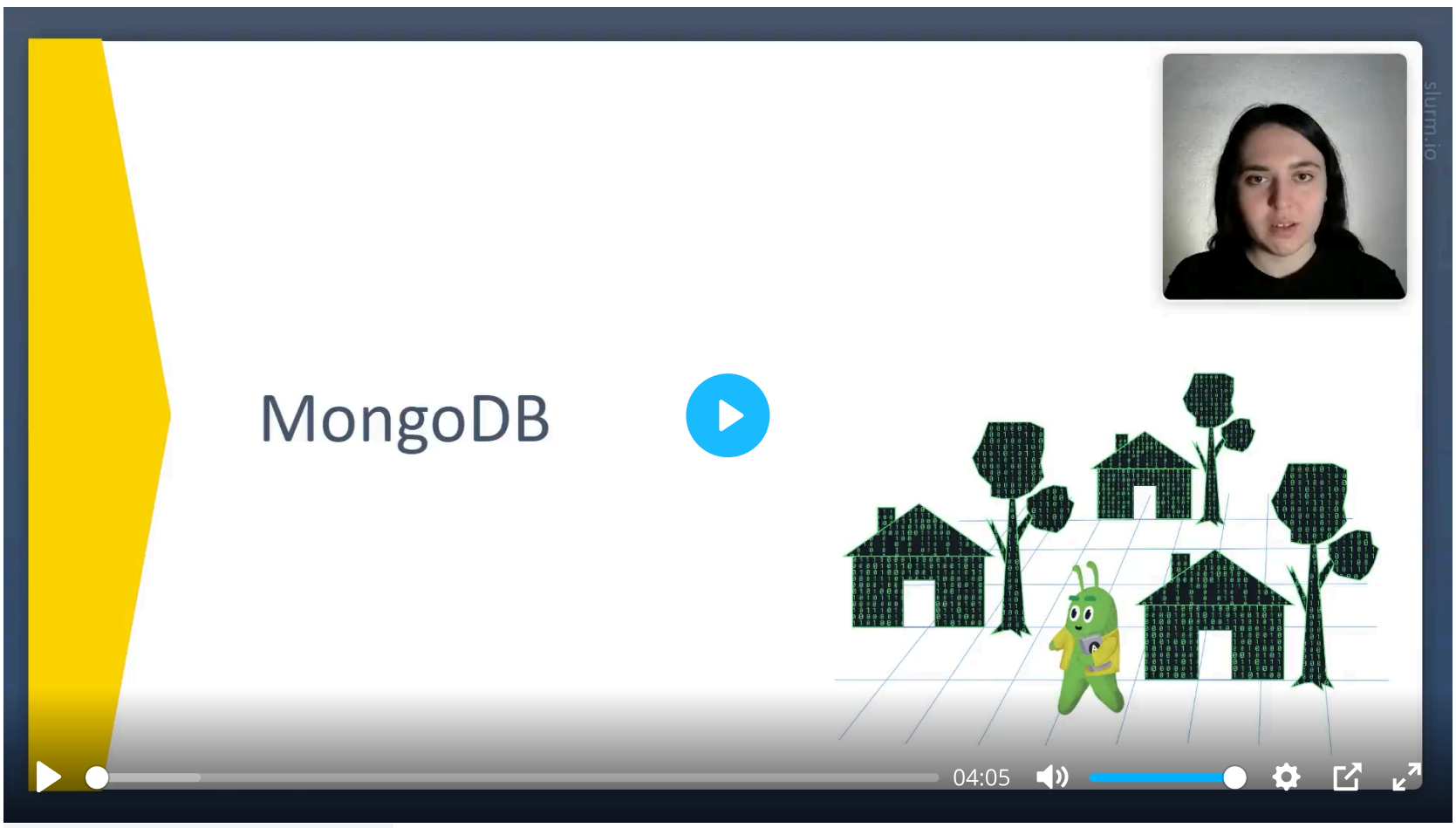
6.3 Практика работы с NoSQL - Clickhouse, Neo4j и Mongo



Презентация к уроку 6.3.5

Код

MongoDB

```
sudo docker run -d --name some-mongo -p27017:27017 -e
MONGO_INITDB_ROOT_USERNAME=mongoadmin -e
MONGO_INITDB_ROOT_PASSWORD=secret mongo:6.0.3
```

> mongodb://[username:password@]host1[:port1][,...hostN[:portN]][/[defaultauthdb][?options]]

Создаем контейнер и подключаемся к БД.

Полезный материал: https://hub.docker.com/

Взаимодействие с MongoDB Robo 3T

```
db.player.insert({name: "Slurm", age: 108})
show dbs
db.getCollectionNames()
db.player.find({}
```

Mongo Python-драйвер

```
pip install pymongo
import pymongo
```

```
MONGO_CONN="mongodb://mongoadmin:secret@localhost:27017"

if __name__ == "__main__":
    mongo = pymongo.MongoClient(MONGO_CONN)
    print(mongo["testnewdb"]["player"].count_documents({}))

    cursor = mongo["testnewdb"]["player"].aggregate([
        {
            "$match": {"name":{"$in":["Slurm"]}}
        },
        {
            "$project":{"name":1, "age":1}
        }
    ])
    print(list(cursor))
```

Текстовая расшифровка видео:

MONGODB

План:

- MongoDB.

MongoDB

MongoDB – это документоориентированная БД, а также одна из самых популярных в NoSQL.

MongoDB имеет множество драйверов для работы.

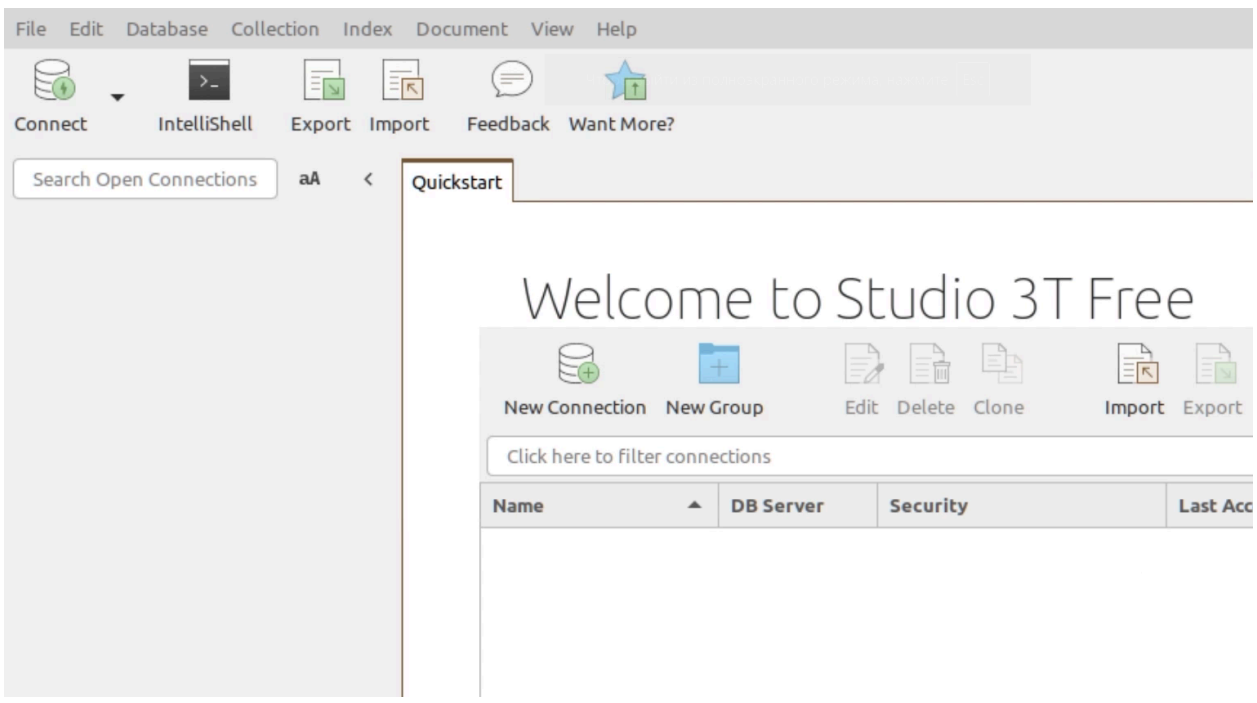
На данном уроке мы попробуем провести Crud-операции напрямую и через Python driver.

Для работы инстанса развернем в контейнере докер с помощью команды, где сам **Mongo** – название контейнера, порт – 27017 (по умолчанию), **username** – mongoadmin, пароль – secret:

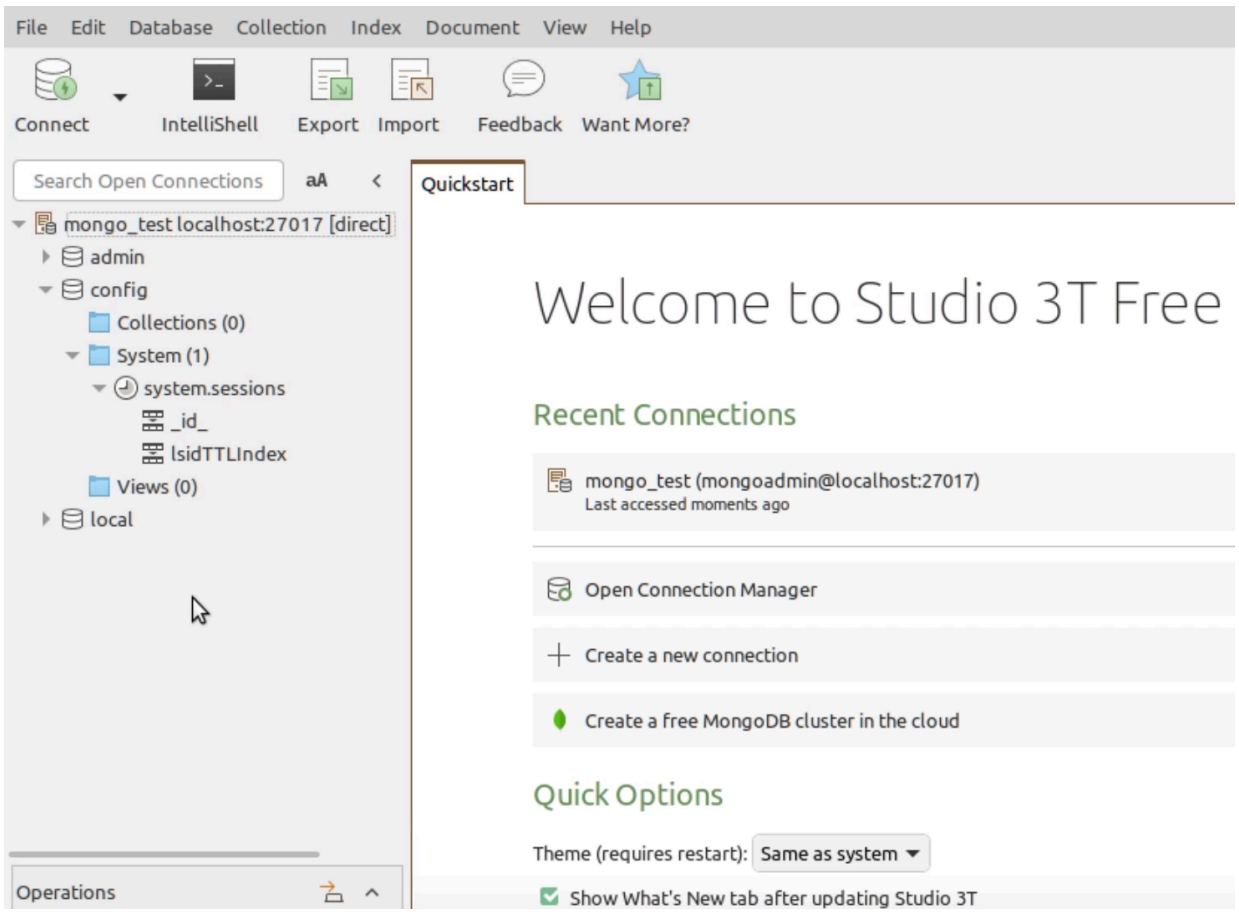
```
asya@asya-HP-Pavilion-dv6-Notebook-PC:~$ sudo docker run -d --name some-mongo -p 27017:27017 -e MONGO_INITDB_ROOT
_USERNAME=mongoadmin -e MONGO_INITDB_ROOT_PASSWORD=secret mongo:6.0.3
```

Для работы с Mongo мы будем использовать robo3t. Также в экосистеме Mongo есть утилита Compass.

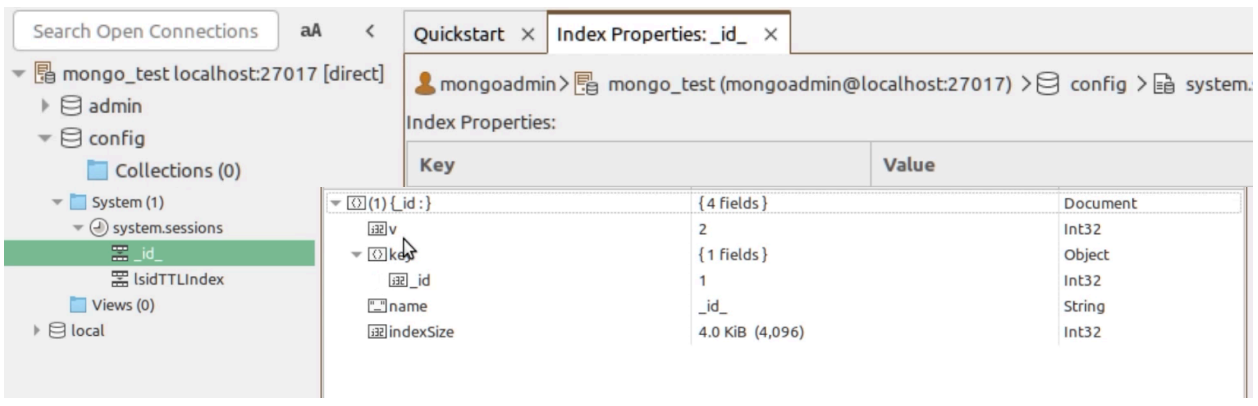
Рассмотрим пример:



Для того, чтобы добавить новое соединение, нажимаем New Connection. Connection Mongo пишется в виде обычной строки, где указывается username, пароль, сервер, порт. Далее мы оставляем все без изменений и создаем подключение, после чего жмем «connect». Здесь отобразилась БД:



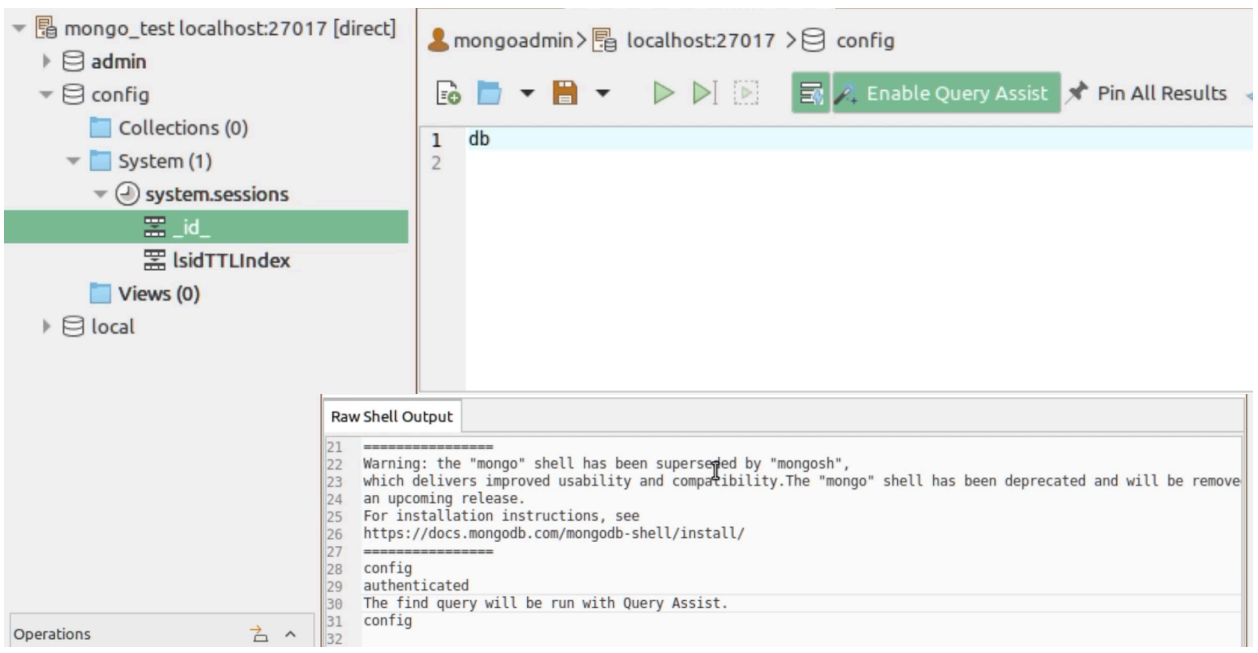
Для того, чтобы посмотреть, как выглядят документы в базе, нажимаем на одну из коллекций документов. Здесь видим структуру документов:



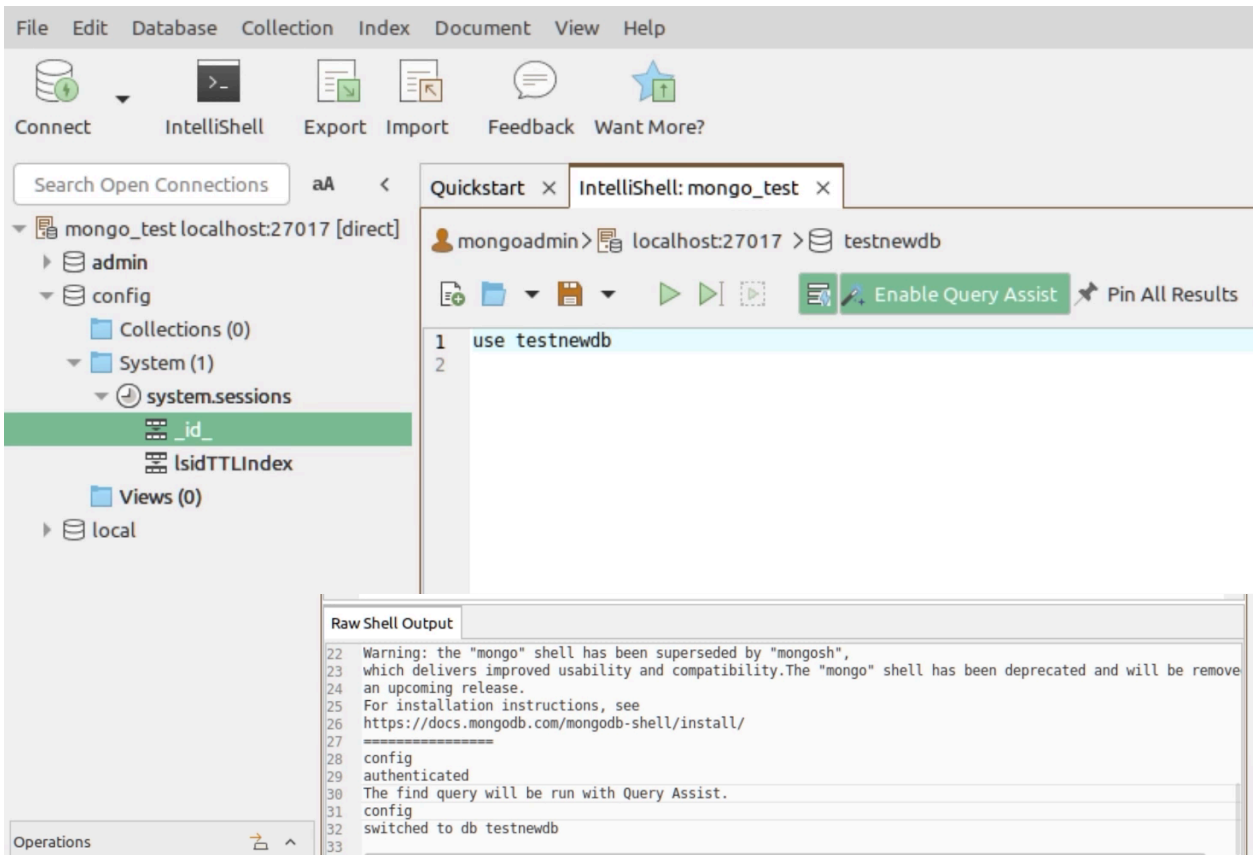
Структура, по сути, и есть JSON. Здесь находятся ключи и разные типы.

Для работы используется IntelliShell. Shell – оболочка, где можно писать запросы в БД Mongo.

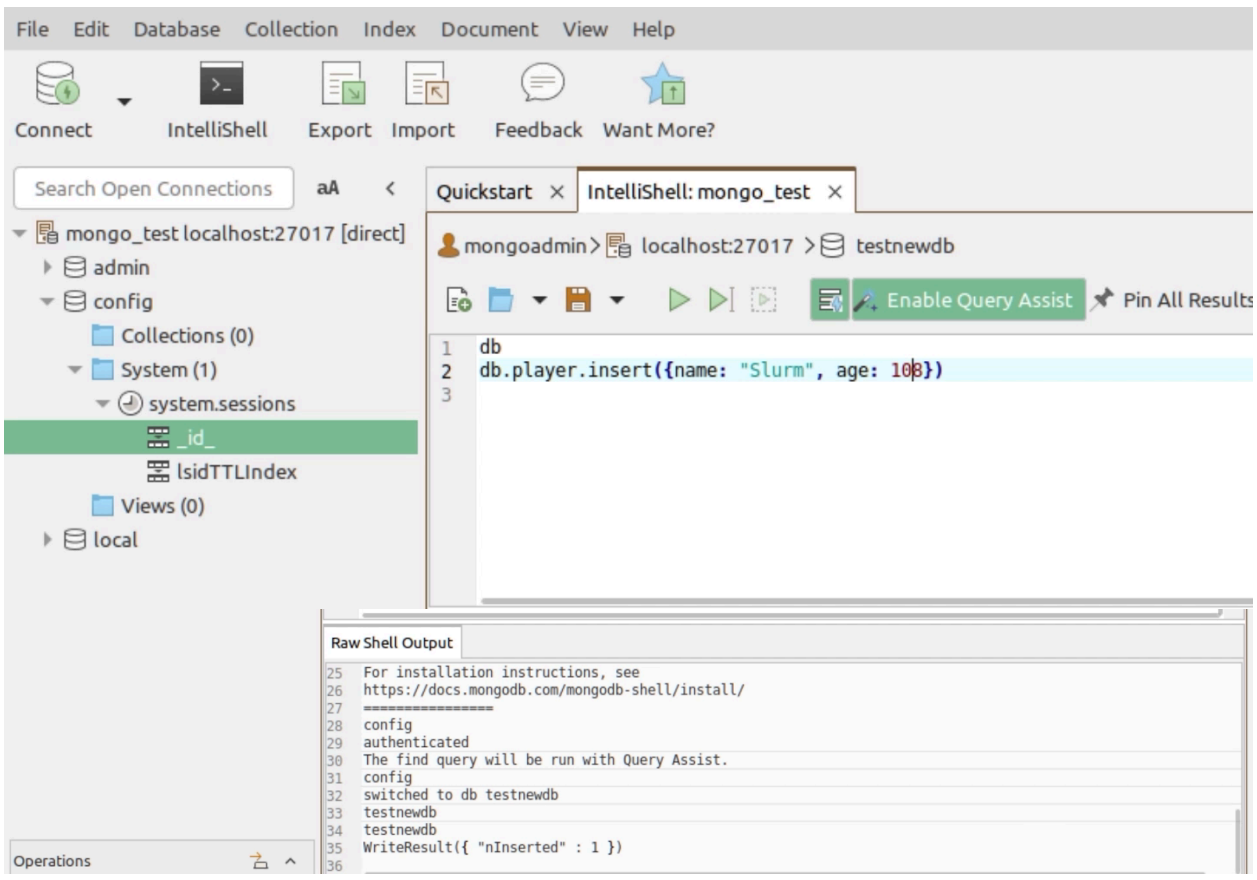
Для того, чтобы посмотреть, какую БД мы используем, пишем «db» (это база-конфиг):



Для того, чтобы создавать свои новые таблицы, лучше использовать не ту базу, указанную по умолчанию, а создать новую. Делается это через «use название БД» (оно автоматически высвечивается):



Проверяем, написав «**testnewdb**». База появляется после того, как в ней появятся какие-либо данные. В NoSQL БД нет строгой схемы. Эти схемы могут меняться от документа к документу. Поэтому мы задаем схему документа, вставляя данные:



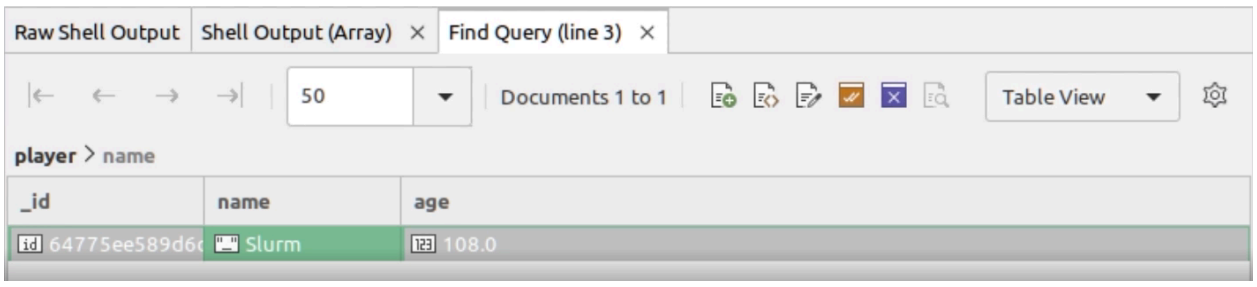
Вставляем данные в виде JSON, где имя – «Slurm», age – 108.

Для того, чтобы найти все элементы коллекции, вначале мы получаем название коллекции, потом делаем следующее:

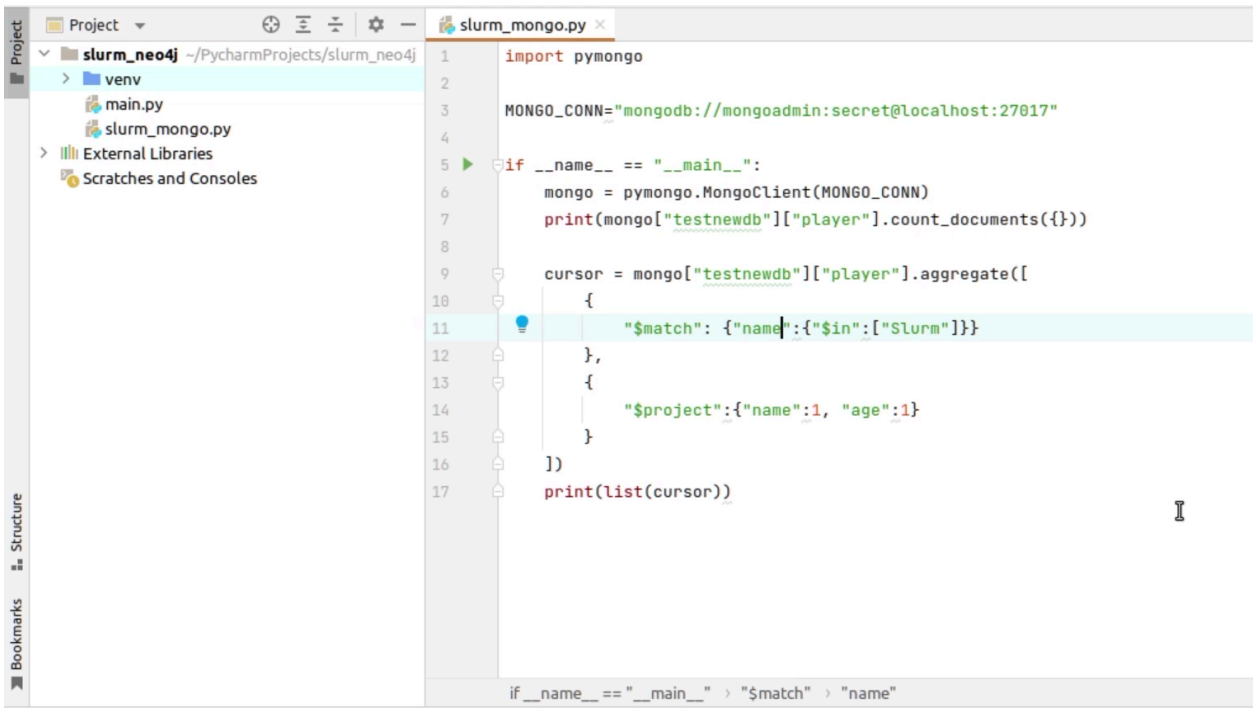
```
1  db
2  db.getCollectionNames()
3  db.player.find({})
4
```

Это означает, что у нас нет фильтров. Также нам даются все значения коллекции. Коллекции Mongo являются аналогами таблиц.

Здесь находятся данные: айдишник, имя, возраст:

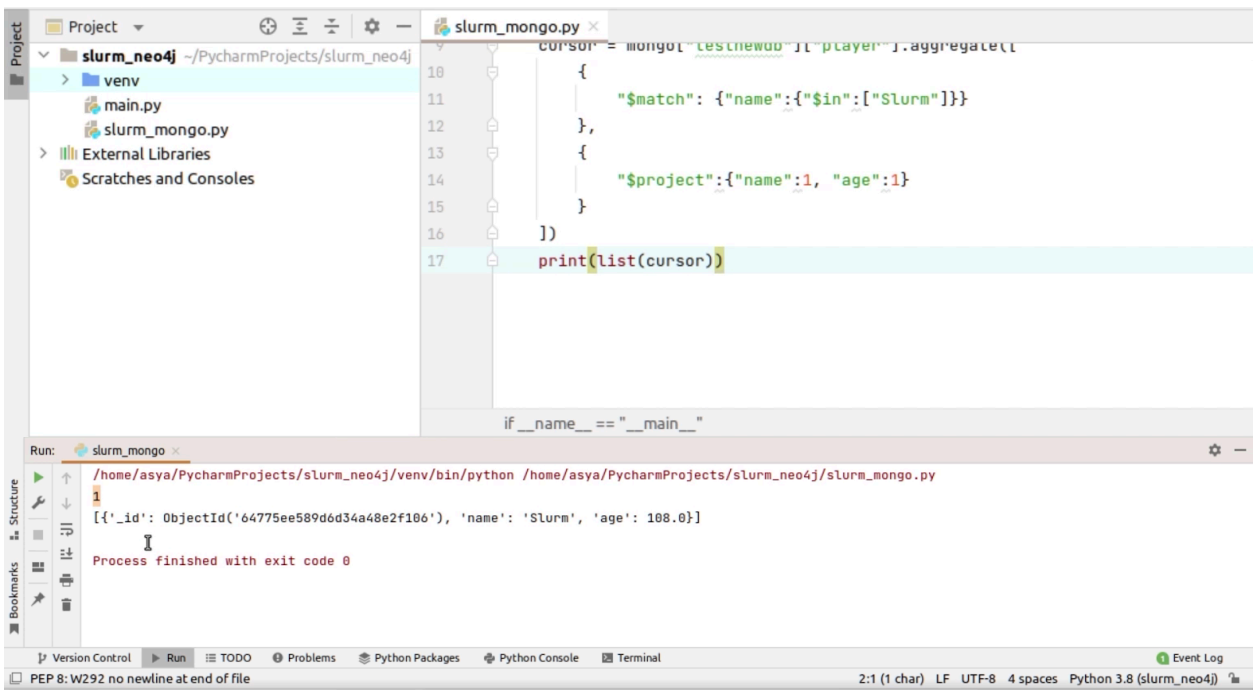


Дальше мы будем работать через драйвер Python:



В коде есть import pymongo для того, чтобы работать с Mongo. Далее мы пишем connection к БД. Видим, что **if _ _name_ _ = "_ _ main_ _"**. Инициализируем клиент для работы, а также вначале считаем количество документов в коллекции «Player». Далее делаем более сложные действия с агрегационными запросами, то есть это пайплайн-агрегации, которые позволяют преобразовывать данные по аналогии с Group By в SQL-запросах.

Вначале данные фильтруются по значению «name», потом через project строится определенная структура, то есть «name» должно быть во множестве «Slurm». Мы можем добавить еще какие-либо фильтры, однако в данный момент мы вывели только те, которые содержат «Slurm». Project будет выводить имя и возраст. Попробуем запустить:



1 – это количество аргументов.

Здесь выводится ID, имя (как было в project) и возраст:

```
[{'_id': ObjectId('64775ee589d6d34a48e2f106'), 'name': 'Slurm', 'age': 108.0}]
```

Как вам урок?

Изучил, далее ›