Motor

Рекомендация по работе с Motor:

- Motor, по сути — обертка над библиотекой pymongo. Поэтому если с документации motor чего-то не хватает, можно посмотреть в документации pymongo.
- Самая полная документация к библиотеке — ее исходный код. Код библиотеки Motor неочевидный. Чтобы убедиться в этом, достаточно посмотреть на реализацию AsyncIOMotorClient. Но так как motor — обертка над pymongo, всегда можно посмотреть на код pymongo. Пример того же клиента MongoClient в pymongo.
- Библиотека во многом повторяет язык запросов MongoDB. Если возникают вопросы, как сделать тот или иной запрос, по MongoDB есть полная документация.

Примеры запросов в MongoDB

connectors/mongo/crud_one.py

```
import asyncio
import os
from typing import Optional

from bson import ObjectId
from motor.motor_asyncio import AsyncIOMotorClient


def get_client():
    return AsyncIOMotorClient(os.getenv("MONGO_URL"))


def get_db():
    return get_client().metaclass


def get_user_collection():
    return get_db().user


async def insert_user() -> ObjectId:
    document = {'first_name': 'Alexander', 'last_name': 'Opryshko'}
    result = await get_user_collection().insert_one(document)
    return result.inserted_id


async def find_user_by_id(user_id: ObjectId) -> Optional[dict]:
    document = await get_user_collection().find_one({'_id': user_id})
    return document


async def update_by_id(user_id: ObjectId) -> int:
    result = await get_user_collection().update_one({'_id': user_id}, {'$set': {'is_tutor': True}})
    return result.modified_count


async def delete_by_id(user_id: ObjectId) -> int:
    result = await get_user_collection().delete_one({'_id': user_id})
    return result.deleted_count


async def run():
    user_id = await insert_user()
    print('insert_user', user_id)
    user = await find_user_by_id(user_id)
    print('find_user_by_id', user)
    res = await update_by_id(user_id)
    print('update_by_id', res)
    user = await find_user_by_id(user_id)
    print('after update_by_id', user)
    # res = await delete_by_id(user_id)
    # print('delete_by_id', res)
    user = await find_user_by_id(user_id)
    print('after delete_by_id', user)


if __name__ == "__main__":
    asyncio.run(run())
```

Запустить пример можно в mercury:

```
python3 connectors/mongo/crud_one.py
```

Рассмотрим функции по порядку:

get_client создает client к MongoDB, с помощью которого нужно выполнять запросы к базе. В момент создания клиента подключения к базе не происходит.
Принимает url вида: mongodb://<user>:<pass>@<host>:port/.
Внутри mercury url можно получить по команде: echo $MONGO_URL.

get_db и **get_user_collection.** В Mongo существует иерархия сущностей: есть база данных, внутри которой находятся коллекции (как таблицы в реляционной БД). В motor можно писать client.db_does_not_exist.coll_does_not_exist, и ошибок не будет. Если коллекции не существует, вы узнаете об этом при запросе.

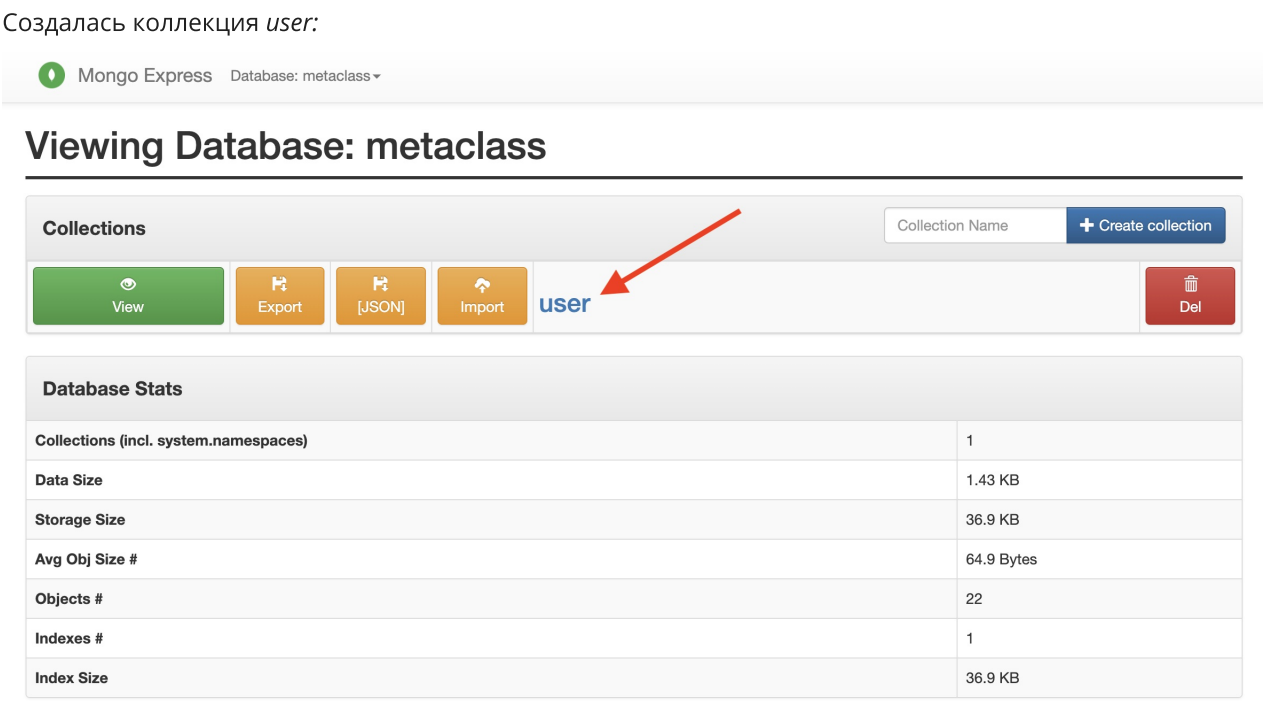
Когда вы получили коллекцию, можно выполнять к ней запросы. Методы, доступные у коллекции, повторяют методы, доступные у коллекции внутри MongoDB.

В этом примере приведены методы (insert_one, find_one, update_one, delete_one), которые выполняют действие только над одним найденным объектом, даже если заданному условию удовлетворяет больше одной записи.

Есть запустить пример и открыть mongo express, можно увидеть результаты работы скрипта.

Создалась база данных *metaclass:*

Создалась коллекция *user:*



Создался *документ:*