
mongoDB.
node
JS

ПОДКЛЮЧЕНИЕ

```
let mongoClient = require('mongodb').MongoClient;
// Строка подключения
let url = 'mongodb://localhost:27017';
// Подключение
mongoClient.connect(url, { useNewUrlParser: true }, (err, client)=>{
        if (err){
                return console.log(err);
        }
        // взаимодействие с базой данных
        console.log('Подключились к базе данных!');
        let db = client.db('test');
        client.close();
});
```

ВСТАВКА

```
let collection = db.collection("users");
let user = {name: "Sasha", age: 32};
collection.insertOne(user, (err, result)=>{
        if (err) return console.log(err);
        console.log(result.ops);
        let users = [{name: "Bob", age: 34} ,
                {name: "Alice", age: 21}, {name: "Tom", age: 45} ];
        collection.insertMany(users, (err, result)=>{
                if (err) return console.log(err);
                        console.log(result.ops);
                        client.close();
        });
});
```

ПОИСК

```
let collection = db.collection("users");
//поиск без параметров
collection.find().toArray((err, results)=>{
        console.log(results);
        //поиск с параметрами
        collection.find({name: "Tom"}).toArray((err, results)=>{
                console.log(results);
                client.close();
        });
});
```

ОБНОВЛЕНИЕ

```
let collection = db.collection("users");
collection.updateMany(
        {name: "Bob"}, // критерий фильтрации
        { $set: {name: "Sam"}}, // параметр обновления
        (err, result)=>{
                console.log(result);
                collection.find().toArray((err, results)=>{
                        console.log(results);
                        client.close();
                });
        }
);
```

УДАЛЕНИЕ

```
let collection = db.collection("users");
collection.deleteMany({name: "Tom"}, (err, result)=>{
        console.log(result);
        collection.find().toArray((err, results)=>{
                console.log(results);
                client.close();
        });
});
```

ЗАДАЧА

Доработать одностраничный todoList с занятия по Vue.js. Создать серверную часть, которая позволяет получать все задачи, удалять задачи и создавать новую задачу. Организовать хранение задач с использованием MongoDB.

ДОПОЛНИТЕЛЬНО

Mongoose - это библиотека JavaScript, часто используемая в приложении Node.js с базой данных MongoDB.

Mongoose - это ODM (Object Document Mapper - объектно-документный отобразитель). Это означает, что Mongoose позволяет вам определять объекты со строго-типизированной схемой, соответствующей документу MongoDB.

Mongoose предоставляет огромный набор функциональных возможностей для создания и работы со схемами. На данный момент Mongoose содержит восемь SchemaTypes (* типы данных схемы), которые может иметь свойство, сохраняемое в MongoDB. Эти типы следующие: String, Number, Date, Buffer, Boolean, Mixed, ObjectId (* уникальный идентификатор объекта, первичный ключ, _id), Array.

ДОПОЛНИТЕЛЬНО

Для каждого типа данных можно:

- задать значение по умолчанию;
- задать пользовательскую функцию проверки данных;
- указать, что поле необходимо заполнить;
- задать get-функцию (геттер), которая позволяет вам проводить манипуляции с данными до их возвращения в виде объекта;
- задать set-функцию (* сеттер), которая позволяет вам проводить манипуляции с данными до их сохранения в базу данных;
- определить индексы для более быстрого получения данных.

ДОПОЛНИТЕЛЬНО

Официальный сайт библиотеки, где можно посмотреть всю необходимую документацию: http://mongoosejs.com

Описание с примерами: https://metanit.com/web/nodejs/6.6.php

ПОЛЕЗНАЯ ССЫЛКА

https://metanit.com/nosql/mongodb/
- Руководство по MongoDB

http://mongodb.github.io/node-mongodb-native/3.1/api/
- Документация по API mongodb driver

http://mongodb.github.io/node-mongodb-native/3.1/reference/connecting/connection-settings/
- Настройки соединения mongodb driver