

**Министерство науки и высшего образования Российской Федерации**  
федеральное государственное автономное образовательное учреждение высшего  
образования  
**«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»**

**О Т Ч Е Т**

по лабораторной работе

«Запросы на выборку и модификацию данных, представления и индексы в PostgreSQL»

по дисциплине «Проектирование и реализация баз данных»

Автор: Глушков Кирилл Георгиевич

Факультет: ИКТ

Группа: К32422

Преподаватель: Говорова Марина Михайловна

Дата сдачи: 15.04.23

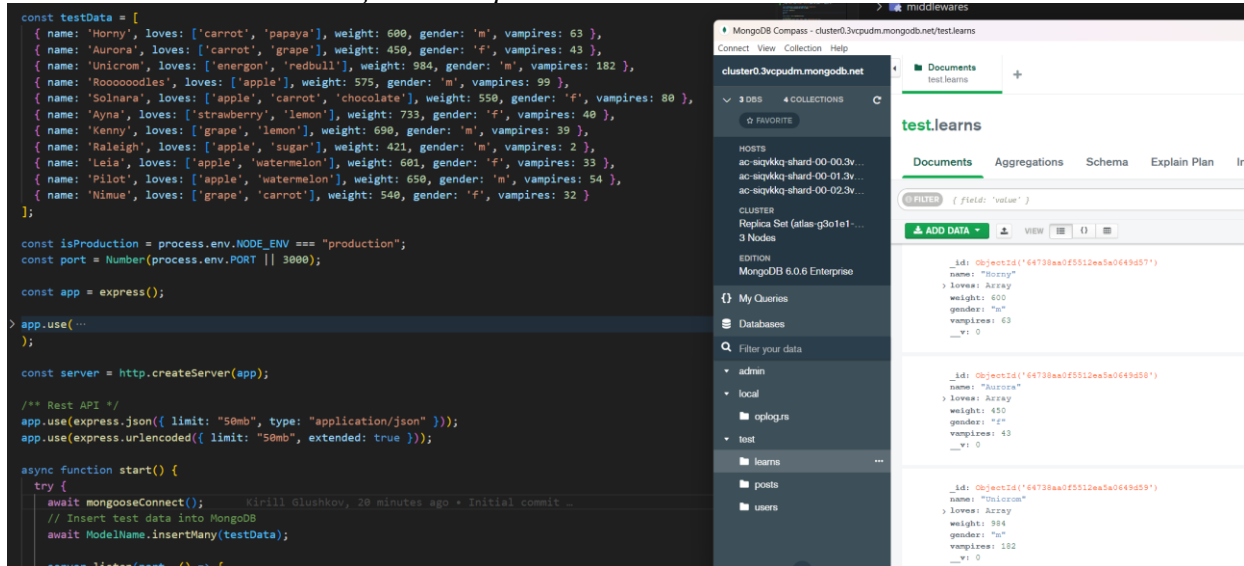


Санкт-Петербург 2023

**СОДЕРЖАНИЕ**

## Практическое задание 8.1.1:

1. Создайте базу данных *learn*.
2. Заполните коллекцию единорогов *unicorns*:



## Практическое задание 8.1.2:

1. Сформируйте запросы для вывода списков самцов и самок единорогов. Ограничьте список самок первыми тремя особями. Отсортируйте списки по имени.
2. Найдите всех самок, которые любят *carrot*. Ограничьте этот список первой особью с помощью функций *findOne* и *limit*.

По умолчанию выборка содержит все поля документа, однако, в том случае, если требуется выбрать лишь конкретные поля, методам *find()* и *findOne()* можно передавать второй аргумент в виде JSON-структуры, с ключами, совпадающими с названиями столбцов и значениями 1, если поле должно попадать в выборку и 0, если его необходимо исключить из выборки.

Вывести только значения полей "age" у все документов, в которых name=Tom:

```
> db.users.find({name: "Tom"}, {age: 1})
```

Использование единицы в качестве параметра {age: 1} указывает, что запрос должен вернуть только содержание свойства age.

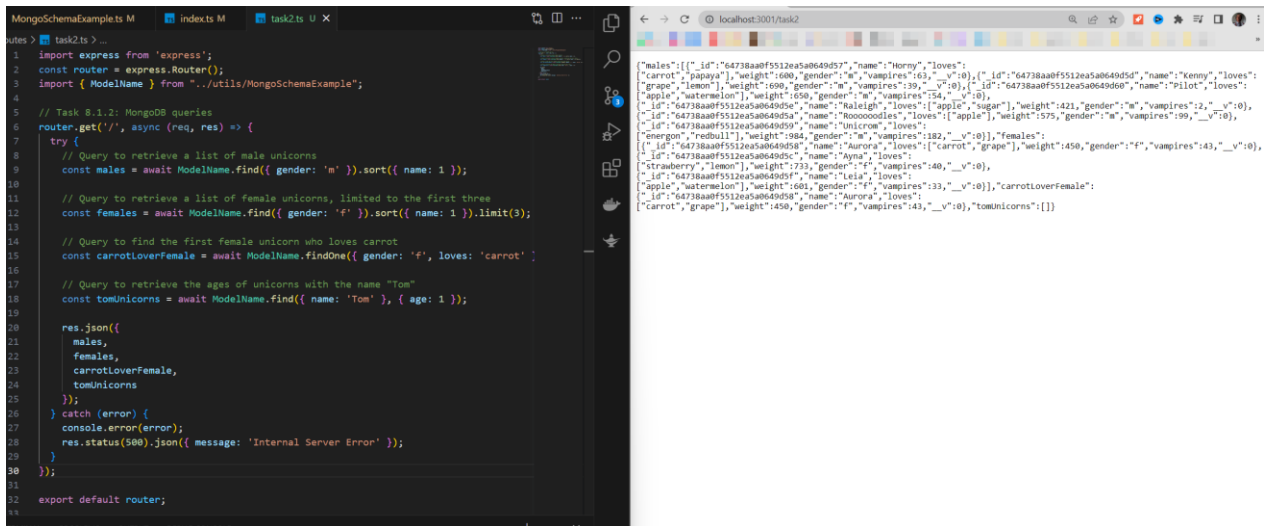
Обратная ситуация: нужно найти все параметры документа, кроме свойства age. В этом случае в качестве параметра указать 0:

```
> db.persons.find({name: "Tom"}, {age: 0})
```

При этом надо учитывать, что даже если мы отметим, что мы хотим получить только поле name, поле *\_id* также будет включено в результирующую выборку. Поэтому, если не нужно видеть данное поле в выборке, то надо явным образом указать: {"\_id": 0}

Альтернативно вместо 1 и 0 можно использовать true и false:

```
> db.users.find({name: "Tom"}, {age: true, _id: false})
```



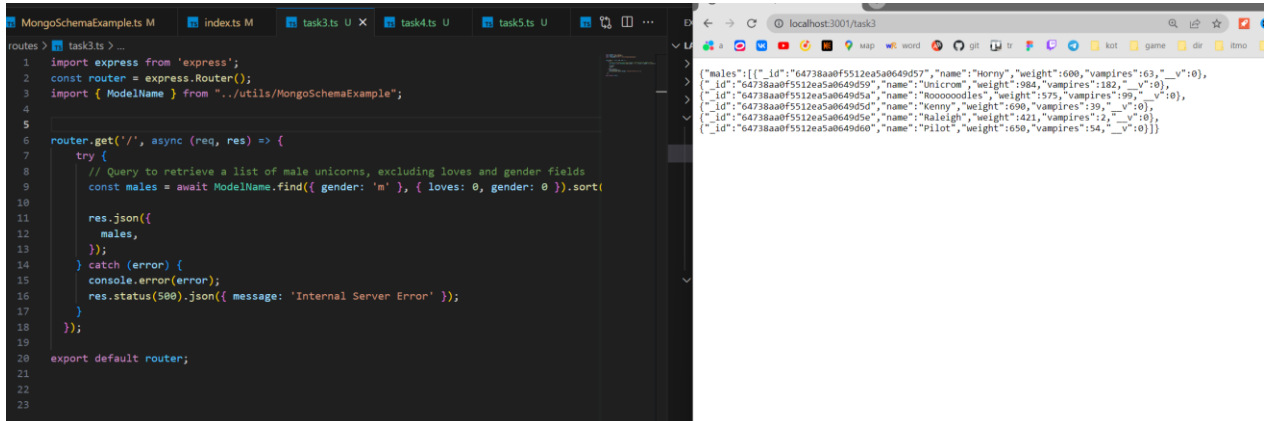
### Практическое задание 8.1.3:

Модифицируйте запрос для вывода списков самцов единорогов, исключив из результата информацию о предпочтениях и поле.

Если нужно отсортировать ограниченную коллекцию, то можно воспользоваться параметром `$natural`. Этот параметр позволяет задать сортировку: документы передаются в том порядке, в каком они были добавлены в коллекцию, либо в обратном порядке.

Например, отобразить последние пять документов:

```
> db.users.find().sort({ $natural: -1 }).limit(5)
```



### Практическое задание 8.1.4:

Вывести список единорогов в обратном порядке добавления.

Для работы с массивами используется оператор `$slice`. Он является в некотором роде комбинацией функций `limit` и `skip`.

Оператор `$slice` принимает два параметра. Первый параметр указывает на общее количество возвращаемых документов. Второй параметр необязательный, но если он

используется, тогда первый параметр указывает на смещение относительно начала (как функция `skip`), а второй - на ограничение количества извлекаемых документов.

Например, в каждом документе определен массив `languages` для хранения языков, на которых говорит человек. Их может быть и 1, и 2, и 3 и более. И допустим, ранее мы добавили следующий объект:

```
> db.users.insert({"name": "Tom", "age": "32", "languages": ["english", "german"]})
```

Если необходимо при выводе документов сделать так, чтобы в выборку попадал только один язык из массива `languages`, а не весь массив, использовать запрос:

```
> db.users.find ({name: "Tom"}, {languages: {$slice : 1}})
```

Данный запрос при извлечении документа оставит в результате только первый язык из массива `languages`, то есть в данном случае `english`.

Обратная ситуация: нужно оставить в массиве также один элемент, но не с начала, а с конца. В этом случае необходимо передать в параметр отрицательное значение:

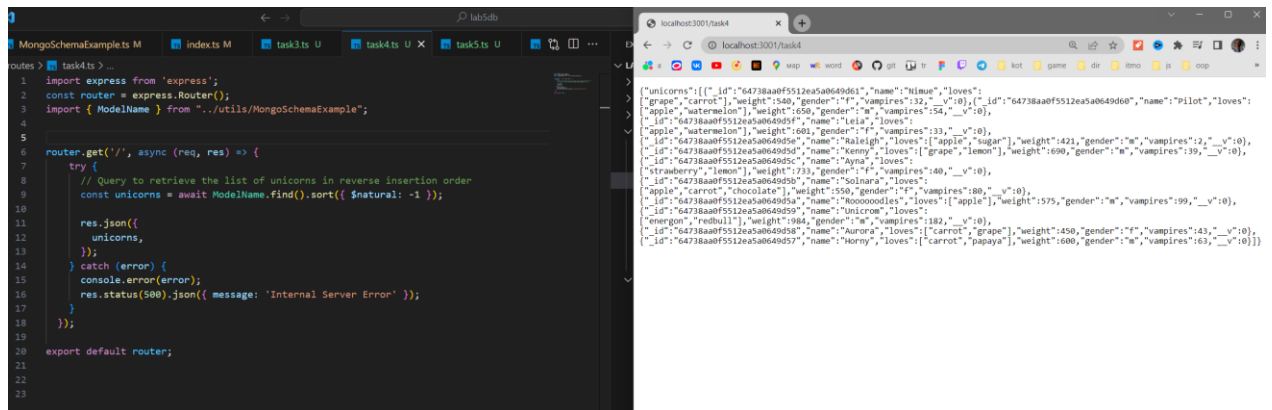
```
> db.users.find ({name: "Tom"}, {languages: {$slice : -1}});
```

Теперь в массиве окажется `german`, так как он первый с конца в добавленном элементе.

Использовать сразу два параметра:

```
> db.users.find ({name: "Tom"}, {languages: {$slice : [-1, 1]}});
```

Первый параметр говорит пропустить один элемент с конца (так как отрицательное значение), а второй параметр указывает на количество возвращаемых элементов массива. В итоге в массиве `language` окажется `"german"`.

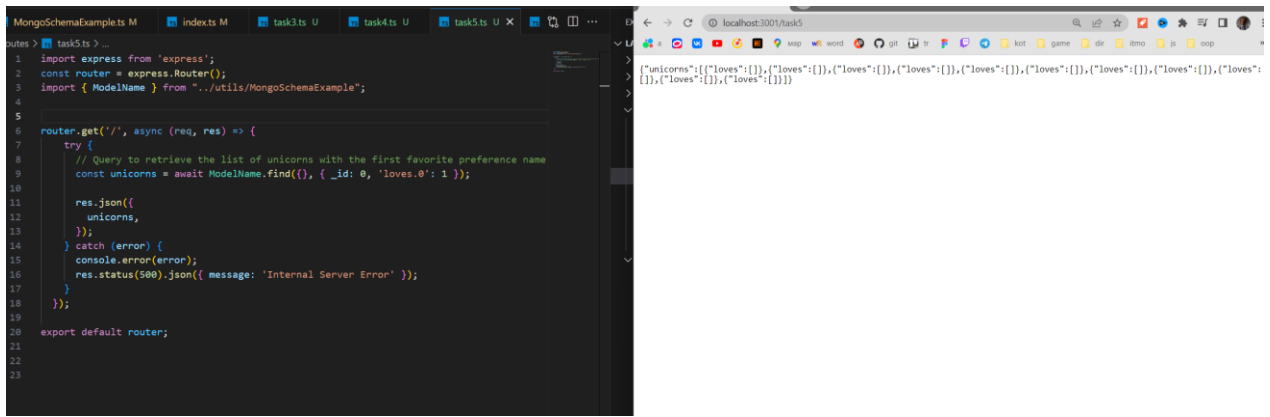


### **Практическое задание 8.1.5:**

*Вывести список единорогов с названием первого любимого предпочтения, исключив идентификатор.*

В качестве селектора могут выступать не только строки, но и регулярные выражения, Например, найти все документы, в которых значение ключа `name` начинается с буквы `T`:

```
> db.users.find({name:/T\w+/i})
```



### Практическое задание 8.1.6:

*Вывести список самок единорогов весом от полутонны до 700 кг, исключив вывод идентификатора.*

Оператор `$ne` извлекает все документы, **не** соответствующие некоторому условию:

```
> db.users.find ({age: {$ne : 22}})
```

Оператор `$in` определяет массив возможных выражений и ищет те ключи, значение которых имеется в массиве:

```
> db.users.find ({age: {$in : [22, 32]}})
```

Противоположным образом действует оператор `$nin`: он определяет массив возможных выражений и ищет те ключи, значение которых отсутствует в этом массиве:

```
> db.users.find ({age: {$nin : [22, 32]}})
```

Оператор `$all` похож на `$in`: он также определяет массив возможных выражений, но требует, чтобы документы имели весь определяемый набор выражений. Например, следующий запрос не вернет нам ни одного результата:

```
> db.users.find ({age: {$all : [22, 32]}})
```

Так как в документе, который представляет человека, может быть только одно значение поля `age` (не может же быть у человека два возраста), то естественно ни в одном документе не будет найдено сразу 22 и 32. Если мы сократим до одного элемента массива, тогда можем получить результат:

```
> db.users.find ({age: {$all : [22]}})
```

Другая ситуация: человек может знать несколько языков, которые присваиваются ключу `languages` в форме массива. Тогда, например, чтобы найти всех людей, говорящих одновременно и по-английски, и по-французски, можно использовать следующее выражение:

```
> db.users.find ({languages: {$all : ["english", "french"]}})
```

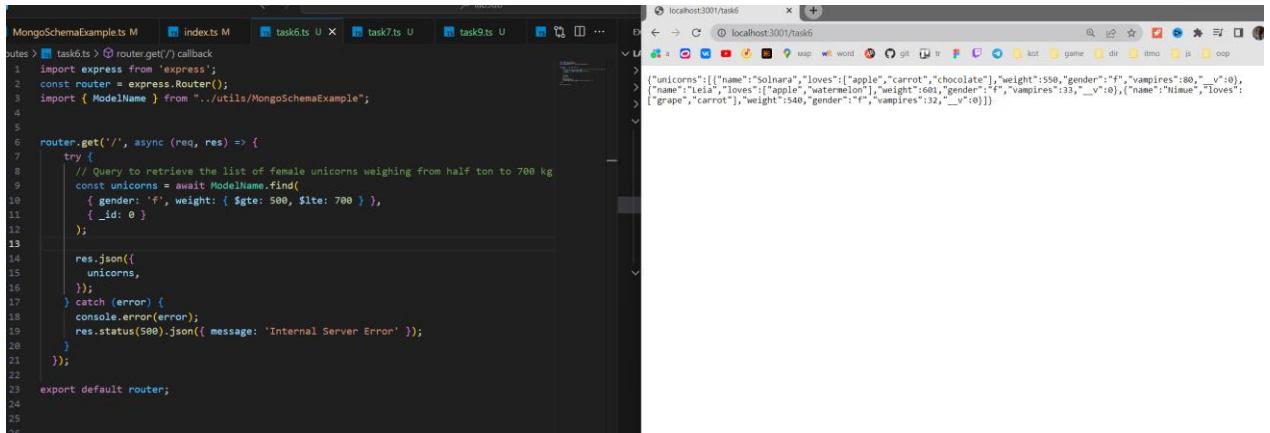
Оператор `$or` определяет набор пар ключ-значение, которые должны иметься в документе. И если документ имеет хоть одну такую пару ключ-значение, то он соответствует данному запросу и извлекается из бд:

```
> db.users.find ({ $or : [{name: "Tom"}, {age: 22}]})
```

Это выражение вернет все документы, в которых либо `name=Tom`, либо `age=22`.

Другой пример вернет все документы, в которых name=Tom, а age равно либо 22, либо среди значений languages есть "german":

```
> db.users.find ({name: "Tom", $or : [{age: 22}, {languages: "german"}]})
```



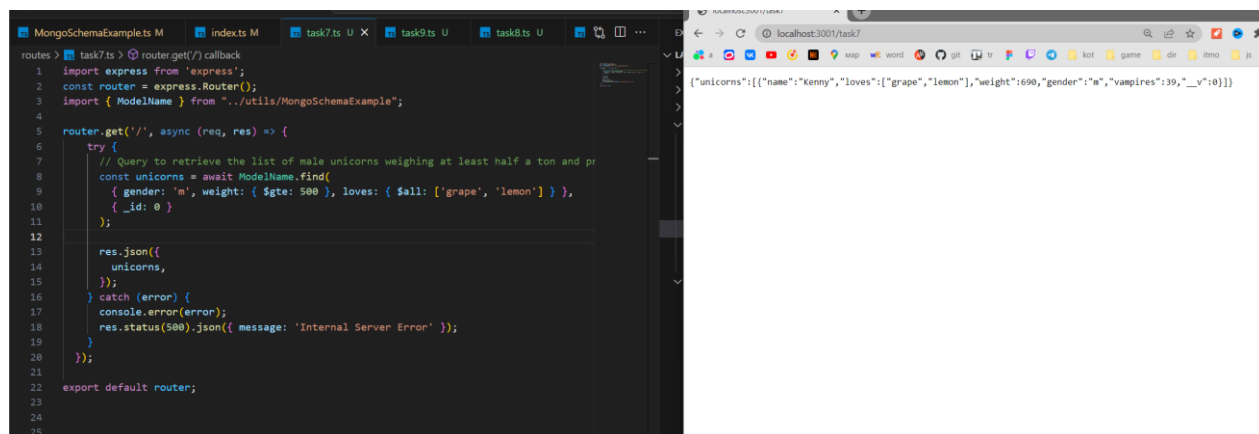
### Практическое задание 8.1.7:

*Вывести список самцов единорогов весом от полутонны и предпочитающих grape и lemon, исключив вывод идентификатора.*

Оператор \$exists позволяет извлечь только те документы, в которых определенный ключ присутствует или отсутствует. Например, вернуть все документы, в которых есть ключ company:

```
> db.users.find ({company: {$exists:true}})
```

Если указать у оператора \$exists в качестве параметра false, то запрос вернет только те документы, в которых не определен ключ company.



### Практическое задание 8.1.8:

*Найти всех единорогов, не имеющих ключ vampires.*

Оператор \$regex задает регулярное выражение, которому должно соответствовать значение поля. Например, пусть поле name обязательно имеет букву "b":

```
> db.users.find ({name: {$regex:"b"}})
```

Важно понимать, что \$regex принимает не просто строки, а именно регулярные выражения, например: `name: { $regex: "om$" }` - значение `name` должно оканчиваться на "om".

Оператор \$slice является в некотором роде комбинацией функций `limit` и `skip`. Но в отличие от них \$slice может работать с массивами.

Оператор \$slice принимает два параметра. Первый параметр указывает на общее количество возвращаемых документов. Второй параметр необязательный, но если он используется, тогда первый параметр указывает на смещение относительно начала (как функция `skip`), а второй - на ограничение количества извлекаемых документов.

Например, в каждом документе определен массив `languages` для хранения языков, на которых говорит человек. Их может быть и 1, и 2, и 3 и более. И допустим, ранее добавили следующий объект:

```
> db.users.insert({"name": "Tom", "age": "32", "languages": ["english", "german"]})
```

Если нужно в выводе документов сделать так, чтобы в выборку попадал только один язык из массива `languages`, а не весь массив, то:

```
1 > db.users.find ({name: "Tom"}, {languages: {$slice : 1}})
```

Данный запрос при извлечении документа оставит в результате только первый язык из массива `languages`, то есть в данном случае `english`.

Обратная ситуация: нужно оставить в массиве также один элемент, но не с начала, а с конца. В этом случае необходимо передать в параметр отрицательное значение:

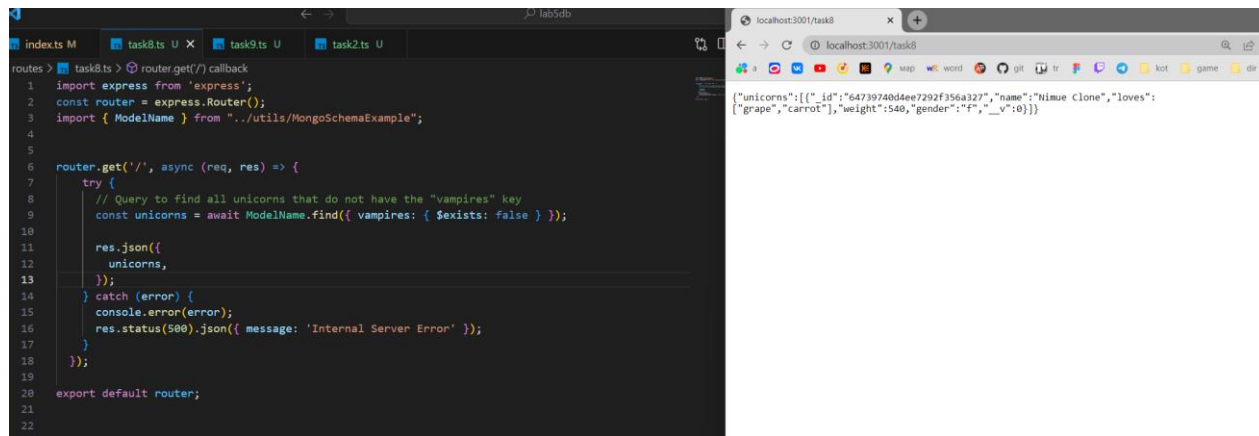
```
1 > db.users.find ({name: "Tom"}, {languages: {$slice : -1}});
```

Теперь в массиве окажется `german`, так как он первый с конца в добавленном элементе.

Использовать сразу два параметра:

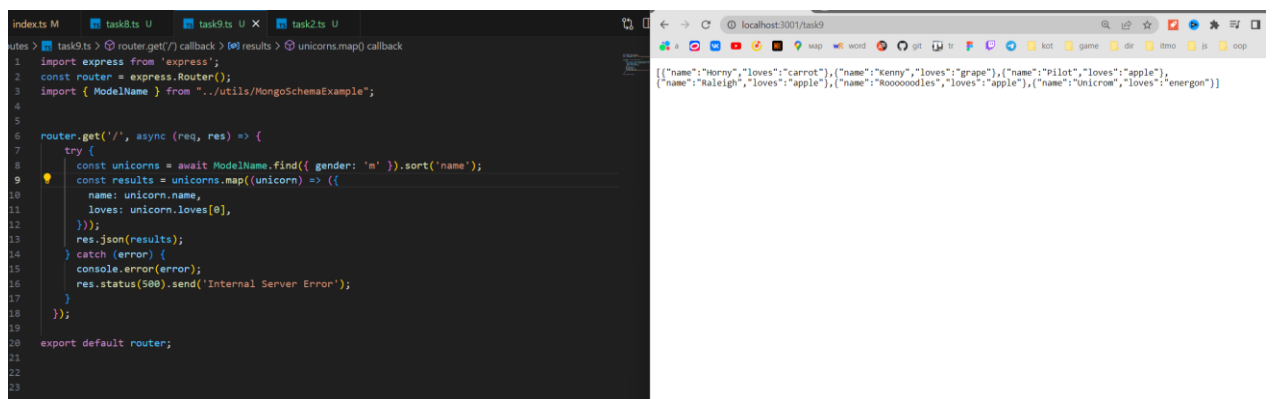
```
1 > db.users.find ({name: "Tom"}, {languages: {$slice : [-1, 1]}});
```

Первый параметр говорит пропустить один элемент с конца (так как отрицательное значение), а второй параметр указывает на количество возвращаемых элементов массива. В итоге в массиве `language` окажется `"german"`.



### **Практическое задание 8.1.9:**

*Вывести список упорядоченный список имен самцов единорогов с информацией об их первом предпочтении.*



2 часть

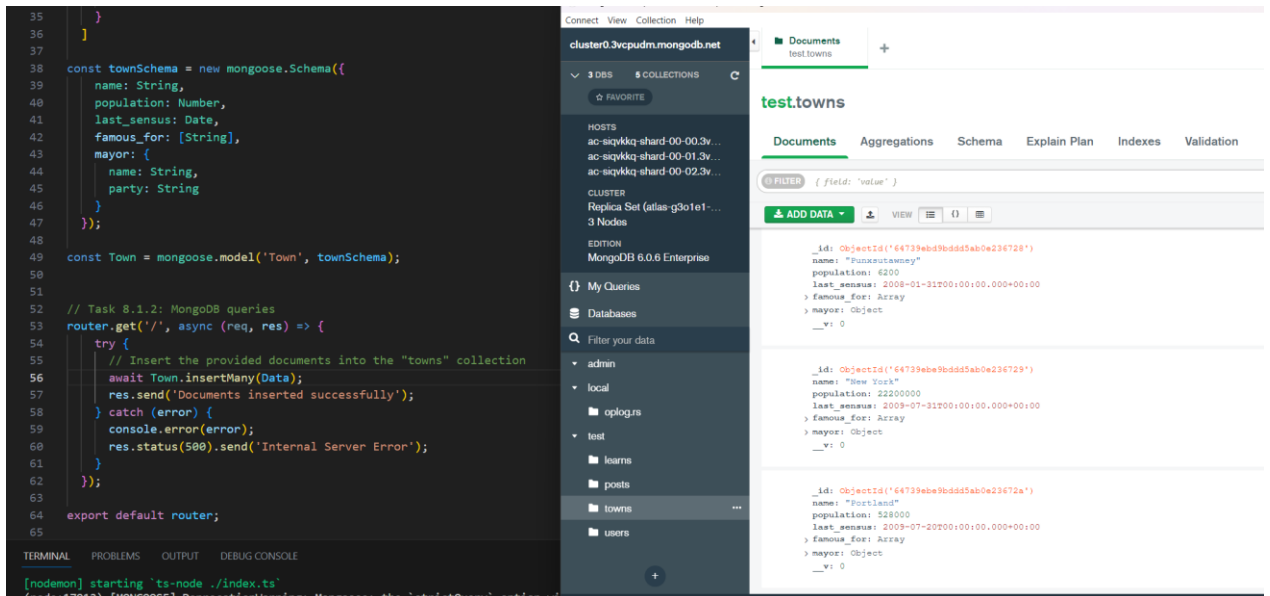
### **Практическое задание 8.2.1:**

*1. Создайте коллекцию towns, включающую следующие документы:*

```
{name: "Punxsutawney ",
  populatiuon: 6200,
  last_sensus: ISODate("2008-01-31"),
  famous_for: [""],
  mayor: {
    name: "Jim Wehrle"
  }}

{name: "New York",
  populatiuon: 22200000,
  last_sensus: ISODate("2009-07-31"),
  famous_for: ["status of liberty", "food"],
```

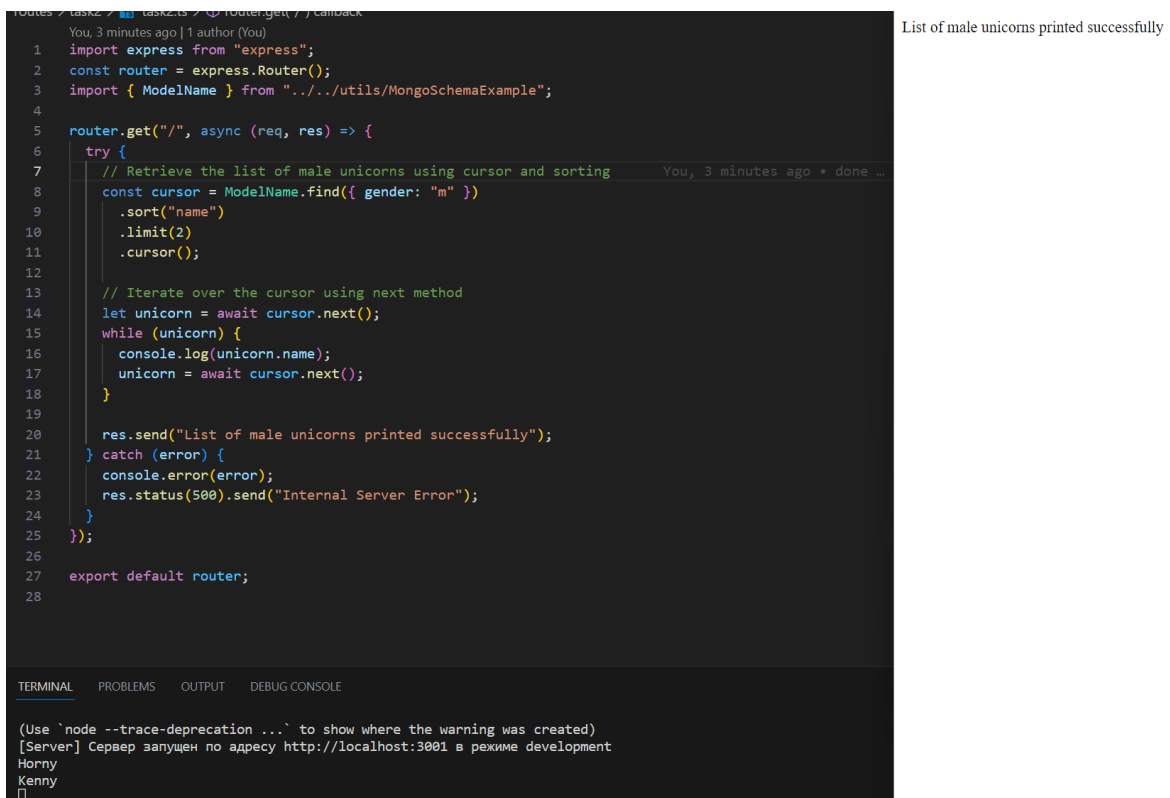




## Практическое задание 8.2.2:

1. Сформировать функцию для вывода списка самцов единорогов.
2. Создать курсор для этого списка из первых двух особей с сортировкой в лексикографическом порядке.
3. Вывести результат, используя `forEach`.

Содержание коллекции единорогов `unicorns`:



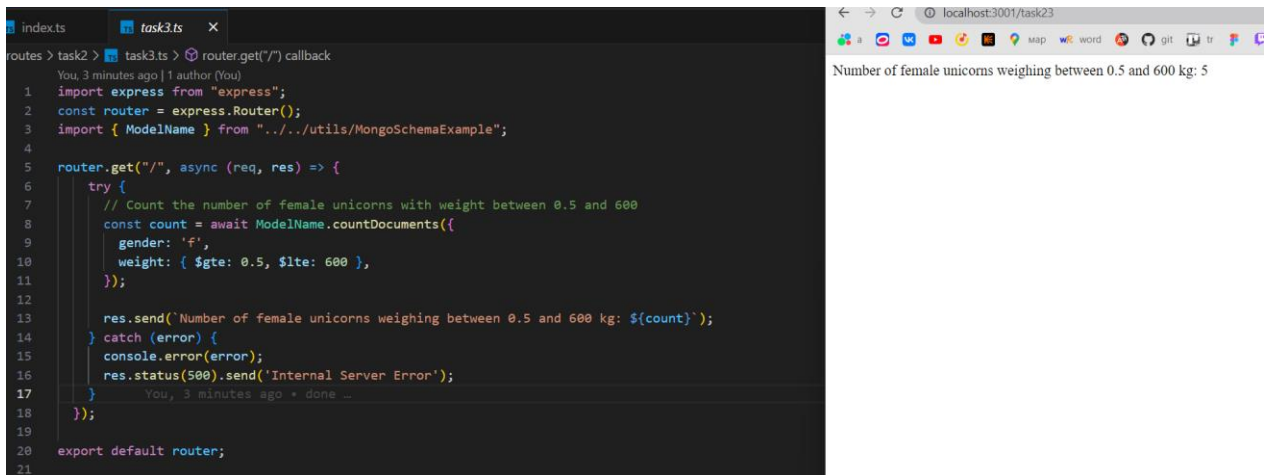
### Практическое задание 8.2.3:

*Вывести количество самок единорогов весом от полутонны до 600 кг.*

Функция **distinct()** позволяет найти уникальные различающиеся значения для одного или нескольких полей документа.

Например, в нескольких документах определено name: "Tom". Нужно найти только уникальные различающиеся значения для одного из полей документа. Для этого можно воспользоваться функцией distinct:

```
> db.users.distinct("name")
["Tom", "Bill", "Alex"]
```



The screenshot shows a code editor on the left and a web browser on the right. The code editor displays a TypeScript file named `task3.ts` with the following content:

```
1 import express from "express";
2 const router = express.Router();
3 import { ModelName } from "../../utils/MongoSchemaExample";
4
5 router.get("/", async (req, res) => {
6   try {
7     // Count the number of female unicorns with weight between 0.5 and 600
8     const count = await ModelName.countDocuments({
9       gender: 'f',
10      weight: { $gte: 0.5, $lte: 600 },
11    });
12
13    res.send(`Number of female unicorns weighing between 0.5 and 600 kg: ${count}`);
14  } catch (error) {
15    console.error(error);
16    res.status(500).send('Internal Server Error');
17  }
18 });
19
20 export default router;
```

The web browser on the right shows the URL `localhost:3001/task23` and the response text: `Number of female unicorns weighing between 0.5 and 600 kg: 5`.

### Практическое задание 8.2.4:

*Вывести список предпочтений.*

Использование метода `aggregate` аналогично применению выражения `GROUP BY` в SQL. Метод `group` принимает три параметра:

- `$group`: агрегатор, который вернет новый документ
- `_id`: указывает на ключ, по которому надо проводить группировку (`$+название поля`)
- `$sum`: оператор для вычисления.

```
index.ts task5.ts X
routes > task2 > task5.ts > router.get("/") callback > preferences
You, 4 minutes ago | author (You)
1 import express from "express";
2 const router = express.Router();
3 import { ModelName } from "../../utils/MongoSchemaExample";
4
5 router.get("/", async (req, res) => {
6   try {
7     // Group documents by loves and count the number of occurrences for each love
8     const preferences = await ModelName.aggregate([
9       { $unwind: '$loves' },
10      {
11        $group: {
12          _id: '$loves',
13          count: { $sum: 1 },
14        },
15      },
16    ]);
17    res.json(preferences);
18  } catch (error) {
19    console.error(error);
20    res.status(500).send('Internal Server Error');
21  }
22 });
23
24 export default router;
```

```
{ "_id": "grape", "count": 5 }, { "_id": "energon", "count": 1 }, { "_id": "watermelon", "count": 2 }, { "_id": "carrot", "count": 6 },
{ "_id": "apple", "count": 5 }, { "_id": "chocolate", "count": 1 }, { "_id": "strawberry", "count": 1 }, { "_id": "papaya", "count": 1 },
{ "_id": "redbull", "count": 1 }, { "_id": "sugar", "count": 1 }, { "_id": "lemon", "count": 2 } ]
```

## Практическое задание 8.2.5:

*Посчитать количество особей единорогов обоих полов.*

```
index.ts task5.ts X
routes > task2 > task5.ts > router.get("/") callback > genderCounts > $group
You, 4 minutes ago | 1 author (You)
1 import express from "express";
2 const router = express.Router();
3 import { ModelName } from "../../utils/MongoSchemaExample";
4
5 router.get("/", async (req, res) => {
6   try {
7     // Count the number of individuals for each gender
8     const genderCounts = await ModelName.aggregate([
9       {
10        $group: {
11          _id: '$gender',
12          count: { $sum: 1 },
13        },
14      },
15    ]);
16    res.json(genderCounts);
17  } catch (error) {
18    console.error(error);
19    res.status(500).send('Internal Server Error');
20  }
21 });
22
23 export default router;
```

```
localhost:3001/task25
[ { "_id": "m", "count": 6 }, { "_id": "f", "count": 7 } ]
```

WriteResult({ "nInserted" : 1 })

## Практическое задание 8.2.6:

*1. Выполнить команду:*

```
> db.unicorns.save({name: 'Barny', loves: ['grape'],
weight: 340, gender: 'm'})
```

*2. Проверить содержимое коллекции unicorns.*

В данном случае результат говорит о том, что найден один документ, удовлетворяющий условию, и один документ был обновлен.

```
task6.ts
You 5 minutes ago | 1 author (You)
import express from "express";
const router = express.Router();
import { ModelName } from "../../utils/MongoSchemaExample";

router.get("/", async (req, res) => {
  try {
    // Create a new unicorn document using the Model
    const barny = new ModelName({
      name: 'Barny',
      loves: ['grape'],
      weight: 340,
      gender: 'm'
    });

    // Save the unicorn document to the database
    await barny.save();

    // Fetch and display the contents of the "unicorns" collection
    const unicorns = await ModelName.find();

    res.json({
      unicorns,
    });
  } catch (error) {
    console.error(error);
    res.status(500).send("An error occurred.");
  }
});
```

```
localhost:3001/task26
[{"_id":"64738aa0f5512ea5a0649d57","name":"Horny","loves":["carrot","papaya"],weight:1600,gender:"m",vampires:63,v:"0"},{"_id":"64738aa0f5512ea5a0649d58","name":"Aurora","carrot","grape"],weight:450,gender:"f",vampires:43,v:"0"},{"_id":"64738aa0f5512ea5a0649d59","name":"Unicorn","emeron","redbull"],weight:984,gender:"m",vampires:182,v:"0"},{"_id":"64738aa0f5512ea5a0649d5a","name":"Rocoooodles","loves":["apple"],weight:575,gender:"m",vampires:99,v:"0"},{"_id":"64738aa0f5512ea5a0649d5b","name":"Solnara","loves":["apple","carrot","chocolate"],weight:550,gender:"f",vampires:80,v:"0"},{"_id":"64738aa0f5512ea5a0649d5c","name":"Ayna","loves":["strawberry","lemon"],weight:733,gender:"f",vampires:40,v:"0"},{"_id":"64738aa0f5512ea5a0649d5d","name":"Kenny","loves":["grape","lemon"],weight:690,gender:"m",vampires:39,v:"0"},{"_id":"64738aa0f5512ea5a0649d5e","name":"Raleigh","loves":["apple","sugar"],weight:421,gender:"m",vampires:12,v:"0"},{"_id":"64738aa0f5512ea5a0649d5f","name":"Leia","loves":["apple","watermelon"],weight:601,gender:"f",vampires:33,v:"0"},{"_id":"64738aa0f5512ea5a0649d60","name":"Pilot","loves":["apple","watermelon"],weight:650,gender:"m",vampires:54,v:"0"},{"_id":"64738aa0f5512ea5a0649d61","name":"Nimue","loves":["grape","carrot"],weight:540,gender:"f",vampires:32,v:"0"},{"_id":"64739740d4ee7292f356a327","name":"Nimue Clone","loves":["grape","carrot"],weight:540,gender:"f",v:"0"},{"_id":"6473978797723b84b9b9a654","name":"Nimue Clone","loves":["grape","carrot"],weight:540,gender:"f",v:"0"},{"_id":"6473a7e33d7808b5bde14519","name":"Barny","loves":["grape"],weight:340,gender:"m",v:"0"}]
```

## Практическое задание 8.2.7:

1. Для самки единорога Ayna внести изменения в БД: теперь ее вес 800, она убила 51 вампира.

Проверить содержимое коллекции unicorns.

```
task7.ts
You 6 minutes ago | 1 author (You)
import express from "express";
const router = express.Router();
import { ModelName } from "../../utils/MongoSchemaExample";

router.get("/", async (req, res) => {
  try {
    // Update the unicorn document with name "Ayna"
    const updateResult = await ModelName.updateOne(
      { name: 'Ayna' },
      { weight: 800, vampires: 51 }
    );

    // Check the contents of the "unicorns" collection
    const unicorns = await ModelName.find();

    res.json({
      unicorns,
    });
  } catch (error) {
    console.error(error);
    res.status(500).send("An error occurred.");
  }
});

export default router;
```

```
localhost:3001/task27
[{"_id":"64738aa0f5512ea5a0649d57","name":"Horny","loves":["carrot","papaya"],weight:1600,gender:"m",vampires:63,v:"0"},{"_id":"64738aa0f5512ea5a0649d58","name":"Aurora","carrot","grape"],weight:450,gender:"f",vampires:43,v:"0"},{"_id":"64738aa0f5512ea5a0649d59","name":"Unicorn","emeron","redbull"],weight:984,gender:"m",vampires:182,v:"0"},{"_id":"64738aa0f5512ea5a0649d5a","name":"Rocoooodles","loves":["apple"],weight:575,gender:"m",vampires:99,v:"0"},{"_id":"64738aa0f5512ea5a0649d5b","name":"Solnara","loves":["apple","carrot","chocolate"],weight:550,gender:"f",vampires:80,v:"0"},{"_id":"64738aa0f5512ea5a0649d5c","name":"Ayna","loves":["strawberry","lemon"],weight:733,gender:"f",vampires:40,v:"0"},{"_id":"64738aa0f5512ea5a0649d5d","name":"Kenny","loves":["grape","lemon"],weight:690,gender:"m",vampires:39,v:"0"},{"_id":"64738aa0f5512ea5a0649d5e","name":"Raleigh","loves":["apple","sugar"],weight:421,gender:"m",vampires:12,v:"0"},{"_id":"64738aa0f5512ea5a0649d5f","name":"Leia","loves":["apple","watermelon"],weight:601,gender:"f",vampires:33,v:"0"},{"_id":"64738aa0f5512ea5a0649d60","name":"Pilot","loves":["apple","watermelon"],weight:650,gender:"m",vampires:54,v:"0"},{"_id":"64738aa0f5512ea5a0649d61","name":"Nimue","loves":["grape","carrot"],weight:540,gender:"f",vampires:32,v:"0"},{"_id":"64739740d4ee7292f356a327","name":"Nimue Clone","loves":["grape","carrot"],weight:540,gender:"f",v:"0"},{"_id":"6473978797723b84b9b9a654","name":"Nimue Clone","loves":["grape","carrot"],weight:540,gender:"f",v:"0"},{"_id":"6473a7e33d7808b5bde14519","name":"Barny","loves":["grape"],weight:340,gender:"m",v:"0"}]
```

## Практическое задание 8.2.8:

1. Для самца единорога Raleigh внести изменения в БД: теперь он любит рэббул.
2. Проверить содержимое коллекции unicorns.

```
index.ts task8.ts X
routes > task2 > task8.ts > router.get("/") callback
You, 7 minutes ago | 1 author (You)
1 import express from "express";
2 const router = express.Router();
3 import { ModelName } from "../utils/MongoSchemaExample";
4
5 router.get("/", async (req, res) => {
6   try {
7     // Update the unicorn document with name "Raleigh"
8     const updateResult = await ModelName.updateOne(
9       { name: "Raleigh" },
10      { $push: { loves: 'Redbull' } }
11    );
12
13    // Check the contents of the "unicorns" collection
14    const unicorns = await ModelName.find();
15    res.json(unicorns);
16  } catch (error) {
17    console.error(error);
18    res.status(500).send('An error occurred.');
```

```
localhost:3001/task28 Raleigh
1/1
[{"_id":"64738aa0f5512ea5a0649d57","name":"Horny","loves":["carrot","papaya"],"weight":600,"gender":"m","vampires":63,"_v":0}, {"_id":"64738aa0f5512ea5a0649d58","name":"Aurora","loves":["carrot","grape"],"weight":450,"gender":"f","vampires":43,"_v":0}, {"_id":"64738aa0f5512ea5a0649d59","name":"Unicorn","loves":["emergon","redbull"],"weight":984,"gender":"m","vampires":182,"_v":0}, {"_id":"64738aa0f5512ea5a0649d5a","name":"Roooonoodles","loves":["apple"],"weight":575,"gender":"m","vampires":99,"_v":0}, {"_id":"64738aa0f5512ea5a0649d5b","name":"Solnara","loves":["apple","carrot","chocolate"],"weight":550,"gender":"f","vampires":80,"_v":0}, {"_id":"64738aa0f5512ea5a0649d5c","name":"Ayna","loves":["strawberry","lemon"],"weight":800,"gender":"f","vampires":51,"_v":0}, {"_id":"64738aa0f5512ea5a0649d5d","name":"Kenney","loves":["grape","lemon"],"weight":690,"gender":"m","vampires":39,"_v":0}, {"_id":"64738aa0f5512ea5a0649d5e","name":"Raleigh","loves":["apple","carrot","chocolate"],"weight":550,"gender":"f","vampires":80,"_v":0}, {"_id":"64738aa0f5512ea5a0649d5f","name":"Leila","loves":["apple","watermelon"],"weight":601,"gender":"f","vampires":33,"_v":0}, {"_id":"64738aa0f5512ea5a0649d60","name":"Pilot","loves":["apple","watermelon"],"weight":650,"gender":"m","vampires":154,"_v":0}, {"_id":"64738aa0f5512ea5a0649d61","name":"Nilue","loves":["grape","carrot"],"weight":540,"gender":"f","vampires":32,"_v":0}, {"_id":"64739740d4ee725f356a327","name":"Nilue Clone","loves":["grape","carrot"],"weight":540,"gender":"f","_v":0}, {"_id":"6473978797723b84b0b9a654","name":"Nilue Clone","loves":["grape","carrot"],"weight":540,"gender":"f","_v":0}, {"_id":"6473a7e33d780b5b0e14519","name":"Barny","loves":["grape"],"weight":340,"gender":"m","_v":0}]
```

## Практическое задание 8.2.9:

1. Всем самцам единорогов увеличить количество убитых вампиров на 5.
2. Проверить содержимое коллекции unicorns.

```
index.ts task9.ts X
routes > task2 > task9.ts > router.get("/") callback
You, 8 minutes ago | 1 author (You)
1 import express from "express";
2 const router = express.Router();
3 import { ModelName } from "../utils/MongoSchemaExample";
4
5 router.get("/", async (req, res) => {
6   try {
7     // Update all male unicorn documents
8     const updateResult = await ModelName.updateMany(
9       { gender: 'm' },
10      { $inc: { vampires: 5 } }
11    );
12
13    // Check the contents of the "unicorns" collection
14    const unicorns = await ModelName.find();
15    res.json(unicorns);
16  } catch (error) {
17    console.error(error);
18    res.status(500).send('An error occurred.');
```

```
localhost:3001/task29
[{"_id":"64738aa0f5512ea5a0649d57","name":"Horny","loves":["carrot","papaya"],"weight":600,"gender":"m","vampires":68,"_v":0}, {"_id":"64738aa0f5512ea5a0649d58","name":"Aurora","loves":["carrot","grape"],"weight":450,"gender":"f","vampires":43,"_v":0}, {"_id":"64738aa0f5512ea5a0649d59","name":"Unicorn","loves":["emergon","redbull"],"weight":984,"gender":"m","vampires":187,"_v":0}, {"_id":"64738aa0f5512ea5a0649d5a","name":"Roooonoodles","loves":["apple"],"weight":575,"gender":"m","vampires":104,"_v":0}, {"_id":"64738aa0f5512ea5a0649d5b","name":"Solnara","loves":["apple","carrot","chocolate"],"weight":550,"gender":"f","vampires":85,"_v":0}, {"_id":"64738aa0f5512ea5a0649d5c","name":"Ayna","loves":["strawberry","lemon"],"weight":800,"gender":"f","vampires":51,"_v":0}, {"_id":"64738aa0f5512ea5a0649d5d","name":"Kenney","loves":["grape","lemon"],"weight":690,"gender":"m","vampires":44,"_v":0}, {"_id":"64738aa0f5512ea5a0649d5e","name":"Raleigh","loves":["apple","sugar","redbull"],"weight":421,"gender":"m","vampires":7,"_v":0}, {"_id":"64738aa0f5512ea5a0649d5f","name":"Leila","loves":["apple","watermelon"],"weight":601,"gender":"f","vampires":33,"_v":0}, {"_id":"64738aa0f5512ea5a0649d60","name":"Pilot","loves":["apple","watermelon"],"weight":650,"gender":"m","vampires":159,"_v":0}, {"_id":"64738aa0f5512ea5a0649d61","name":"Nilue","loves":["grape","carrot"],"weight":540,"gender":"f","vampires":32,"_v":0}, {"_id":"64739740d4ee725f356a327","name":"Nilue Clone","loves":["grape","carrot"],"weight":540,"gender":"f","_v":0}, {"_id":"6473978797723b84b0b9a654","name":"Nilue Clone","loves":["grape","carrot"],"weight":540,"gender":"f","_v":0}, {"_id":"6473a7e33d780b5b0e14519","name":"Barny","loves":["grape"],"weight":340,"gender":"m","_v":0,"vampires":5}]
```

## Практическое задание 8.2.10:

1. Изменить информацию о городе Портланд: мэр этого города теперь беспартийный.

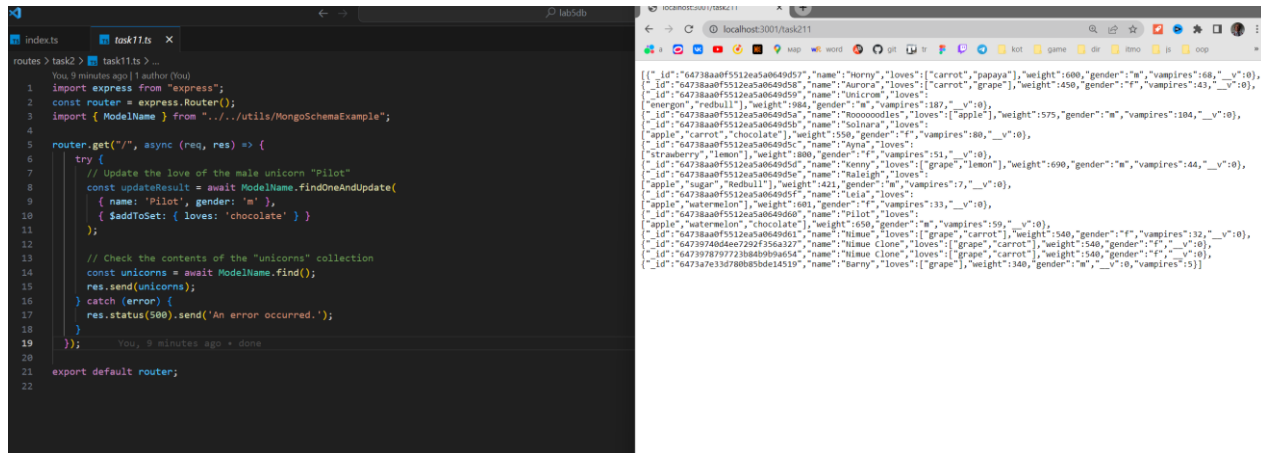
Проверить содержимое коллекции towns.

```
> task2 > task10.ts > router.get("/") callback > updateResult
You, 8 minutes ago | 1 author (You)
1 import express from "express";
2 import { Town } from "../task1";
3 const router = express.Router();
4 Town
5
6 router.get("/", async (req, res) => {
7   try {
8     // Update the unicorn document with name "Portland"
9     const updateResult = await Town.updateOne(
10      { name: 'Portland' },
11      { $unset: { 'mayor.party': '' } }
12    );
13
14    // Check the contents of the "towns" collection
15    const towns = await Town.find();
16
17    res.json({
18      towns,
19    });
20  } catch (error) {
21    console.error(error);
22    res.status(500).send('An error occurred.');
```

```
[{"towns":[{"mayor":{"name":"Jim Wehrle"},"_id":"64739eb9dbdd5a80e236728","name":"Punxsutawney","population":6200,"last_sensus":"2008-01-31T00:00:00.000Z","famous_for":[""],"_v":0}, {"mayor":{"name":"Michael Blomberg","party":"I"},"_id":"64739eb9dbdd5a80e236729","name":"New York","population":22200000,"last_sensus":"2009-07-31T00:00:00.000Z","famous_for":["status of liberty","food"],_v":0}, {"mayor":{"name":"Sam Adams"},"_id":"64739eb9dbdd5a80e23672a","name":"Portland","population":520000,"last_sensus":"2009-07-20T00:00:00.000Z","famous_for":["beer","food"],_v":0}]}]
```

## Практическое задание 8.2.11:

1. Изменить информацию о самце единорога *Pilot*: теперь он любит и шоколад.
2. Проверить содержимое коллекции *unicorns*.



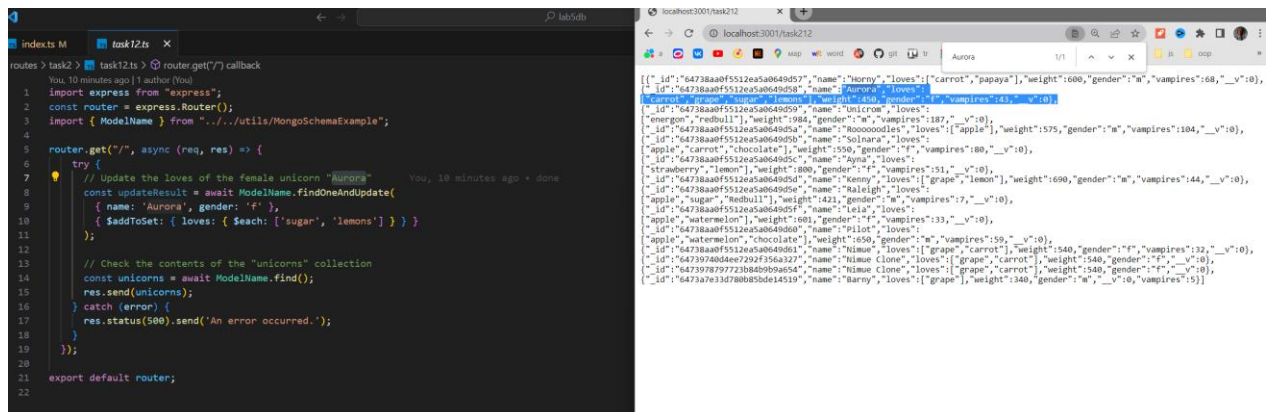
The screenshot shows a VS Code editor on the left with a file named `task11.js`. The code is an Express.js application that connects to a MongoDB database. It has a GET route for `/` that updates the `loves` array of a unicorn named `Pilot` by adding `'chocolate'` to it. It then finds all documents in the `unicorns` collection and sends them as a JSON response. The browser on the right shows the response at `localhost:3001/task11`, displaying a large array of unicorn objects. Each object has fields like `_id`, `name`, `loves` (an array), `weight`, `gender`, and `vampires`.

Оператор `$addToSet` подобно оператору `$push` добавляет объекты в массив. Отличие состоит в том, что `$addToSet` добавляет данные, если их еще нет в массиве:

```
> db.users.update({name : "Tom"}, {$addToSet: {languages: "russian"}})
```

## Практическое задание 8.2.12:

1. Изменить информацию о самке единорога *Aurora*: теперь она любит еще и сахар, и лимоны.
2. Проверить содержимое коллекции *unicorns*.



The screenshot shows a VS Code editor on the left with a file named `task12.js`. The code is similar to the previous one, but it updates the `loves` array of a unicorn named `Aurora` by adding `'sugar'` and `'lemons'` to it. The browser on the right shows the response at `localhost:3001/task12`, displaying the same array of unicorn objects, but with `Aurora`'s `loves` array now containing `'grape', 'lemon', 'sugar', 'lemons'`.

## Практическое задание 8.2.13:

1. Создайте коллекцию *towns*, включающую следующие документы:

```
{name: "Punxsutawney ",
popujatiuon: 6200,
last_sensus: ISODate("2008-01-31"),
famous_for: ["phil the groundhog"],
mayor: {
  name: "Jim Wehrle"
}
```

```

    }}

    {name: "New York",
    popujatiuon: 22200000,
    last_sensus: ISODate("2009-07-31"),
    famous_for: ["status of liberty", "food"],
    mayor: {
        name: "Michael Bloomberg",
        party: "I"}}

    {name: "Portland",
    popujatiuon: 528000,
    last_sensus: ISODate("2009-07-20"),
    famous_for: ["beer", "food"],
    mayor: {
        name: "Sam Adams",
        party: "D"}}

```

2. *Удалите документы с беспартийными мэрами.*
3. *Проверьте содержание коллекции.*
4. *Очистите коллекцию.*
5. *Просмотрите список доступных коллекций.*



```
routes > task2 > task13.ts > router.get("/") callback
You, 10 minutes ago | 1 author (You)
1 import express from "express";
2 const router = express.Router();
3 import { Town } from "../task1";
4
5 router.get("/", async (req, res) => {
6   try {
7     // Delete documents with mayors who don't belong to any party
8     await Town.deleteMany({ 'mayor.party': { $exists: false } });
9
10    // Check the contents of the collection
11    const towns = await Town.find();
12    console.log('Contents of the towns collection:', towns);
13
14    // Clear the collection
15    await Town.deleteMany();
16
17    res.send('Tasks completed successfully.');

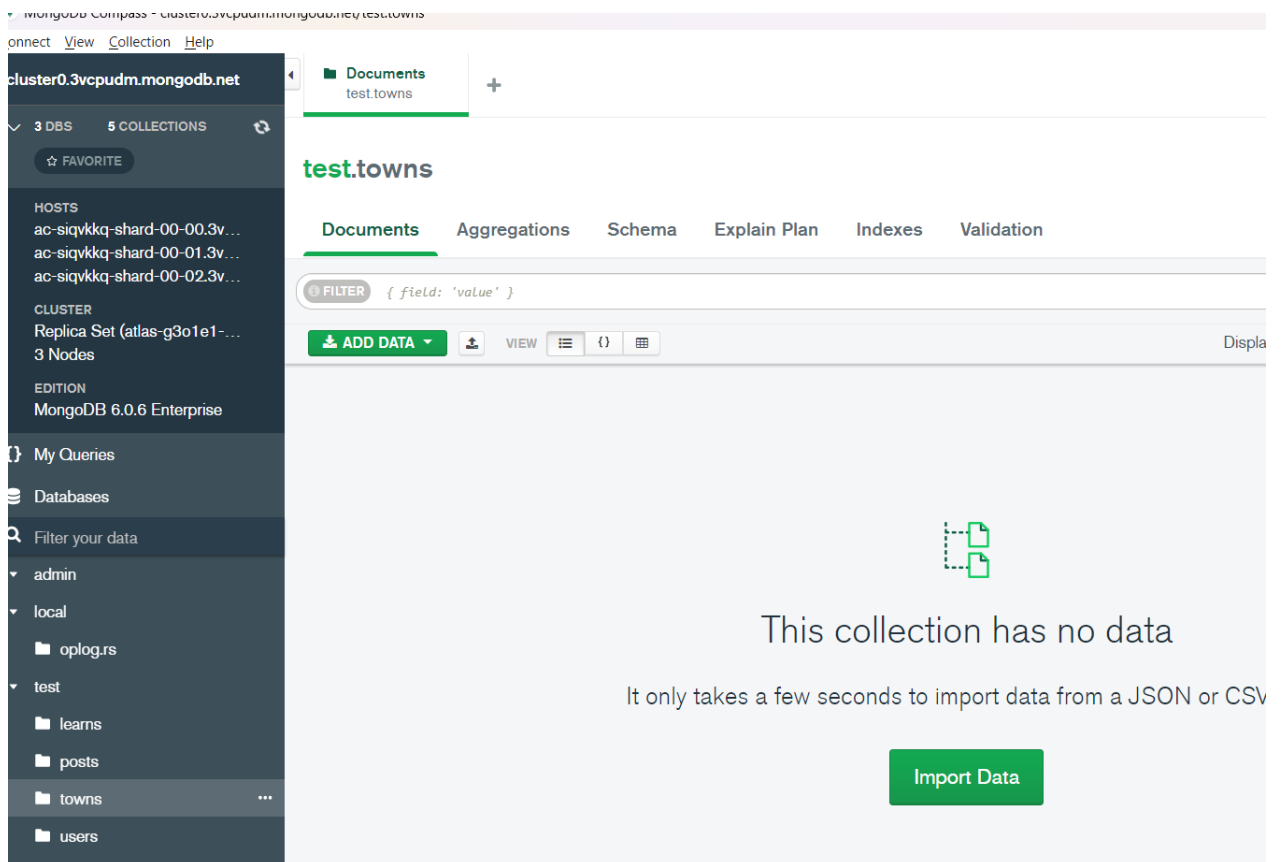
TERMINAL PROBLEMS OUTPUT DEBUG CONSOLE



```
name: 'New York',
population: 22200000,
last_sensus: 2009-07-31T00:00:00.000Z,
famous_for: [ 'status of liberty', 'food' ],
__v: 0
}
```


```

Tasks completed successfully.





### Практическое задание 8.3.1:

1. Создайте коллекцию зон обитания единорогов, указав в качестве идентификатора кратко название зоны, далее включив полное название и описание.
2. Включите для нескольких единорогов в документы ссылку на зону обитания, используя второй способ автоматического связывания.
3. Проверьте содержание коллекции единорогов.

```
index.ts \  index.ts \task3  task1.ts routes  task4.ts routes  task1.ts \task3  task2.ts
> task3 > task1.ts > [e] HabitatsSchema > _id
{
  required: true,
},
description: {
  type: String,
  required: true,
},
});
export const Habitat = mongoose.model('Habitat', HabitatsSchema);

// Task 8.3.2: MongoDB queries
router.get('/', async (req, res) => {
  try {
    // Insert the provided documents into the "towns" collection
    await Habitat.insertMany(habitatsData);
    await ModelName.updateMany(
      { name: { $in: ['Horny', 'Aurora', 'Unicorn'] } },
      { $set: { habitat: { $ref: 'habitats', $id: 'forest' } } }
    );
    const unicorns = await ModelName.find();
    res.json(unicorns);
  } catch (error) {
    console.error(error);
    res.status(500).send('Internal Server Error');
  }
});
export default router;
```

```
[{"_id": "64738aa0f5512ea5a0649d52", "name": "Horny", "loves": ["carrot", "papaya"], "weight": 600, "gender": "m", "vampires": 68, "vampire_loves": ["carrot", "grape", "sugar", "lemons"], "weight": 450, "gender": "f", "vampires": 143, "v": 0}, {"_id": "64738aa0f5512ea5a0649d59", "name": "Unicorn", "loves": ["emeron", "redbull"], "weight": 984, "gender": "m", "vampires": 187, "v": 0}, {"_id": "64738aa0f5512ea5a0649d5a", "name": "Rococooodles", "loves": ["apple"], "weight": 575, "gender": "m", "vampires": 104, "v": 0}, {"_id": "64738aa0f5512ea5a0649d5b", "name": "Soliana", "loves": ["apple", "carrot", "chocolate"], "weight": 550, "gender": "f", "vampires": 80, "v": 0}, {"_id": "64738aa0f5512ea5a0649d5c", "name": "Ayna", "loves": ["strawberry", "lemon"], "weight": 800, "gender": "f", "vampires": 51, "v": 0}, {"_id": "64738aa0f5512ea5a0649d5d", "name": "Kenny", "loves": ["grape", "lemon"], "weight": 690, "gender": "m", "vampires": 44, "v": 0}, {"_id": "64738aa0f5512ea5a0649d5e", "name": "Raleigh", "loves": ["apple", "sugar", "redbull"], "weight": 421, "gender": "m", "vampires": 7, "v": 0}, {"_id": "64738aa0f5512ea5a0649d5f", "name": "Leia", "loves": ["apple", "watermelon"], "weight": 601, "gender": "f", "vampires": 33, "v": 0}, {"_id": "64738aa0f5512ea5a0649d60", "name": "Pilot", "loves": ["apple", "watermelon", "chocolate"], "weight": 650, "gender": "m", "vampires": 59, "v": 0}, {"_id": "64738aa0f5512ea5a0649d61", "name": "Nima", "loves": ["grape", "carrot"], "weight": 540, "gender": "f", "vampires": 32, "v": 0}, {"_id": "64739740d4e7292f356a327", "name": "Nima Clone", "loves": ["grape", "carrot"], "weight": 540, "gender": "f", "v": 0}, {"_id": "6473978797723084b9b9a654", "name": "Nima Clone", "loves": ["grape", "carrot"], "weight": 540, "gender": "f", "v": 0}, {"_id": "6473a7e3d7808650de145139", "name": "Barry", "loves": ["grape"], "weight": 340, "gender": "m", "v": 0, "vampires": 15}]
```

### Практическое задание 8.3.2:

1. Проверьте, можно ли задать для коллекции unicorns индекс для ключа name с флагом unique.

```
index.ts \  index.ts \task3  task1.ts routes  task4.ts routes  task1.ts \task3  task2.ts M X
> task3 > task2.ts > router.get('/') callback
gender: {
  type: String,
  enum: ['m', 'f'],
  required: true
},
vampires: {
  type: Number,
  required: false
}
});
// Define the model
export const unicorns = mongoose.model('unicorns', UnicornsSchema);

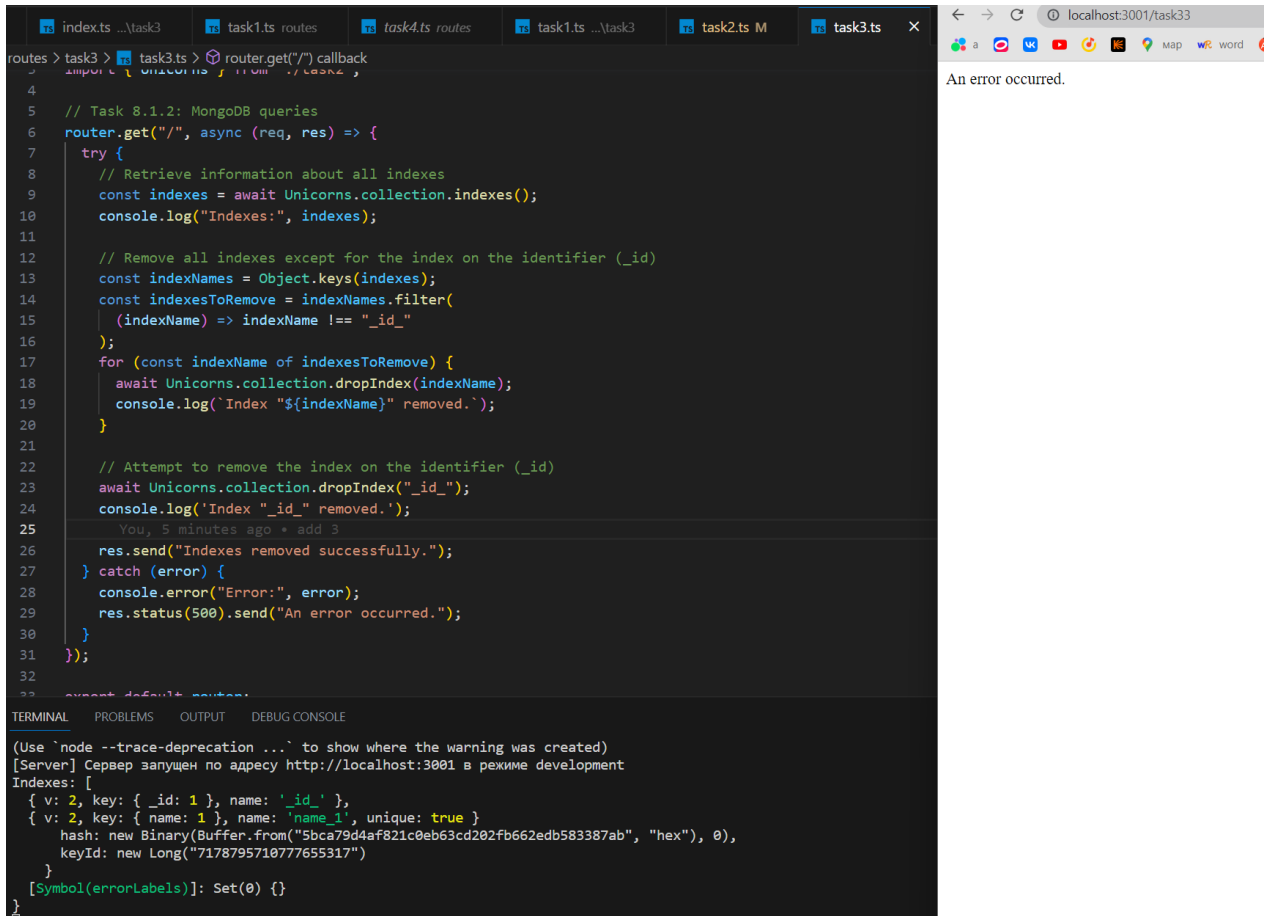
router.get('/', async (req, res) => {
  try {
    // insert
    // await Unicorns.insertMany(unicornData);
    // Create a unique index for the "name" key in the "unicorns" collection
    await Unicorns.collection.createIndex({ name: 1 }, { unique: true });

    // Check if the index was successfully created
    const indexInfo = await Unicorns.collection.indexInformation();
    const isIndexCreated = indexInfo.hasOwnProperty('name_1') && indexInfo.name_1.unique;

    res.send({ isIndexCreated });
  } catch (error) {
    res.status(500).send('An error occurred.');
```

### Практическое задание 8.3.3:

1. Получите информацию о всех индексах коллекции *unicorns*.
2. Удалите все индексы, кроме индекса для идентификатора.
3. Попробуйте удалить индекс для идентификатора.



```
routes > task3 > task3.ts > router.get("/") callback
4
5 // Task 8.1.2: MongoDB queries
6 router.get("/", async (req, res) => {
7   try {
8     // Retrieve information about all indexes
9     const indexes = await Unicorns.collection.indexes();
10    console.log("Indexes:", indexes);
11
12    // Remove all indexes except for the index on the identifier (_id)
13    const indexNames = Object.keys(indexes);
14    const indexesToRemove = indexNames.filter(
15      (indexName) => indexName !== "_id_"
16    );
17    for (const indexName of indexesToRemove) {
18      await Unicorns.collection.dropIndex(indexName);
19      console.log(`Index "${indexName}" removed.`);
20    }
21
22    // Attempt to remove the index on the identifier (_id)
23    await Unicorns.collection.dropIndex("_id_");
24    console.log(`Index "_id_" removed.`);
25
26    res.send("Indexes removed successfully.");
27  } catch (error) {
28    console.error("Error:", error);
29    res.status(500).send("An error occurred.");
30  }
31 });
32
33 export default routes;
```

TERMINAL PROBLEMS OUTPUT DEBUG CONSOLE

(Use `node --trace-deprecation ...` to show where the warning was created)

[Server] Сервер запущен по адресу http://localhost:3001 в режиме development

Indexes: [

```
{ v: 2, key: { _id: 1 }, name: '_id_',
  { v: 2, key: { name: 1 }, name: 'name_1', unique: true }
  hash: new Binary(Buffer.from("5bca79d4af821c0eb63cd202fb662edb583387ab", "hex"), 0),
  keyId: new Long("7178795710777655317")
}
```

[Symbol(errorLabels)]: Set(0) {}

An error occurred.

### Практическое задание 8.3.4:

1. Создайте объемную коллекцию *numbers*, задействовав курсор:  

```
for(i = 0; i < 100000; i++){db.numbers.insert({value: i})}
```
2. Выберите последних четыре документа.
3. Проанализируйте план выполнения запроса 2. Сколько потребовалось времени на выполнение запроса? (по значению параметра *executionTimeMillis*)
4. Создайте индекс для ключа *value*.
5. Получите информацию о всех индексах коллекции *numbers*.
6. Выполните запрос 2.
7. Проанализируйте план выполнения запроса с установленным индексом. Сколько потребовалось времени на выполнение запроса?
8. Сравните время выполнения запросов с индексом и без. Дайте ответ на вопрос: какой запрос более эффективен?

```
task1.ts routes task4.ts routes task1.ts ...task3 task2.ts M task3.ts task4.ts ...task3 M X
routes > task3 > task4.ts > router.get("/") callback
13 // Create a mongoose model for numbers collection
14 const Numbers = mongoose.model("Numbers", numbersSchema);
15
16 // Task 8.1.2: MongoDB queries
17 router.get("/", async (req, res) => {
18   try {
19     // Query the last four documents
20     const query = Numbers.find().sort({ _id: -1 }).limit(4);
21     const executionTimeWithoutIndex = await measureExecutionTime(
22       query.exec(),
23       "Execution Time (Without Index)"
24     );
25
26     // Create an index on the "value" key
27     await Numbers.createIndexes();
28
29     // Get information about all indexes
30     const indexes = await Numbers.collection.indexes();
31     console.log("Indexes:", indexes);
32
33     // Query the last four documents with the index
34     const queryWithIndex = Numbers.find().sort({ _id: -1 }).limit(4);
35     const executionTimeWithIndex = await measureExecutionTime(
36       queryWithIndex.exec(),
37       "Execution Time (With Index)"
38     );
39
40     // Compare execution times
41     const comparisonResult =
42       executionTimeWithoutIndex < executionTimeWithIndex
43       ? "Without Index is more efficient"
44       : "With Index is more efficient";
45
46     res.send(`Comparison Result: ${comparisonResult}`);
47   } catch (error) {
48     console.error("Error:", error);
49     res.status(500).send("An error occurred.");
50   }
51 }
52 }
```

localhost:3001/task34  
Comparison Result: Without Index is more efficient

Выводы:

Прокачал свои навыки по монгодб