

Gizem Avcı 21360859071

## NodeJs ile Web Programlama Hafta-11 Raporu

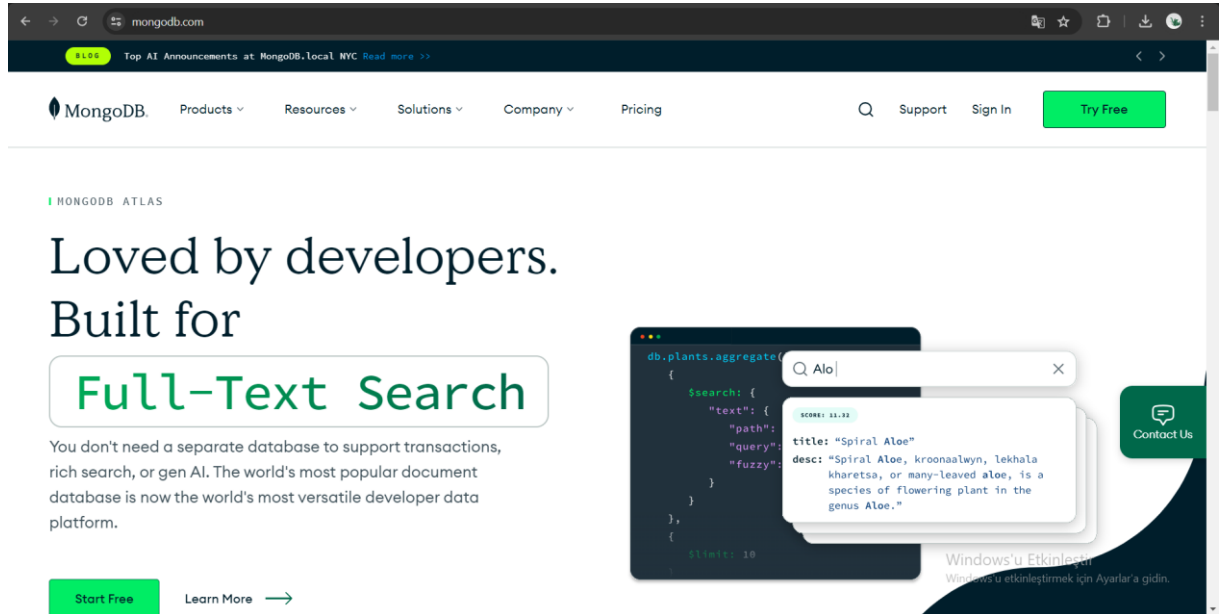
Bugünkü dersimizde veri tabanının nasıl kullanılacağını öğrendik.

Veri tabanı için de MongoDB kullandık. MongoDB, NoSQL veri tabanıdır. SQL veri tabanlarında table denilen bir yapı vardır. O tablolarda her bir veri yeni bir satıra eklenir ve sütun bazlı olarak da veri ile ilişkilendirilir. NoSQL veri tabanlarında ise bu yapı tamamen farklıdır. Mesela satır yerine document bulunmaktadır ve sütun yerine de fields kısımları bulunmaktadır.

MongoDB kendi veri tabanını bulut üzerinde sağlar. Bu yüzden local üzerinden kurup erişmek yerine bulut üzerindeki mongoDB sunucusuna erişip orada işlem yaptık. Bunu yapabilmek için öncelikle mongoDB de bir hesap oluşturmamız gerektirmekte ve ücretsiz sürümleri de mevcuttur.

Kayıt aşamaları içinse ilk olarak <https://www.mongodb.com/> sayfasına girmemiz gerekmektedir. Ardından sağ üste bulunan **Try Free** butonuna tıklayıp hesap bilgilerimizi oluşturmalıyız.

Sayfa ana ekranı şu şekilde:



Google hesabı ile giriş veya orada sorulan soruları cevaplayarak bir hesap oluşturulmaktadır.

Ardından bulut üzerinde oluşturduğu bir cluster ismini soracaktır bunun ismine projenizin ismini verebilirsiniz.




Name  
You cannot change the name once the cluster is created.

Cluster0


☒ Automate security setup ⓘ


☒ Preload sample dataset ⓘ

Provider

Region

 Bahrain (me-south-1) ★

★ Recommended ⓘ  Low carbon emissions ⓘ

Cluster ismini default olarak da bırakabiliriz. Maddeler için de **preload sample dataset** kısmı örnek veri setleri içerdiğinden ve yaklaşık 45mb lık yer kapladığından işaretlememize gerek yoktur. Çünkü mongoDB hesabı ücretsiz hesaplar için yalnızca 512mb yer ayırmıştır.

Daha sonra da connection string ve şifre istenecek. Connection string değerini unutabiliriz çünkü connection string adından buton bulunmakta ve istenildiği zaman erişilebilmektedir fakat şifre bilgisi kesinlikle unutulmamalıdır çünkü kodlarımız içerisinde o şifreyi kullanarak erişim sağlayacağız ve unutulan şifreyi bulmak da mümkün değildir.

Artı olarak şu şekilde bir olay var: Mesela bu bağlantıyı okulda kurduk ve çalışmamızı gerçekleştirdik. Daha sonra eve geçince bağlantı sağlayamayabiliriz bunun sebebi de ip adresinin erişimine izin verdiğimiz adresler arasında ev ip adresimizin bulunmamasıdır bu yüzden mekan değişikliğinde bulunan ip adreslerini de güvenli erişime eklemeliyiz.

Ardından vscode da TaskManagerApp adından klasör oluşturduk.

Burada ilk işlemimiz package.json dosyasını yüklemek oldu. Bunu da terminal üzerinde **npm init -y** komutunu çağırarak sağladık.

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\Users\Gizem\Desktop\hafta11\TaskManagerApp> npm init -y
Wrote to C:\Users\Gizem\Desktop\hafta11\TaskManagerApp\package.json:

{
  "name": "taskmanagerapp",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC"
}

PS C:\Users\Gizem\Desktop\hafta11\TaskManagerApp>
```

Ardından **mongodb** modülünü yine terminal üzerinden ekledik: **npm i mongodb**.

```
PS C:\Users\Gizem\Desktop\hafta11\TaskManagerApp> npm i mongodb
added 12 packages, and audited 13 packages in 5s

found 0 vulnerabilities
PS C:\Users\Gizem\Desktop\hafta11\TaskManagerApp> 
```

Bu işlemler sonrasında mongodb.js dosyası oluşturduk. Dosya içerisine mongodb modülünü ekledik.

```
const mongodb = require("mongodb")
```

Kod içerisinde MongoClient oluşturduk bunu da mongodb modülünün içerisinden aldık. Ardından url aldık.

```
const MongoClient = mongodb.MongoClient

const connectionURL =
'mongodb+srv://gizemavci:<password>@cluster0.pxpsfvj.mongodb.net/?retryWrites=
true&w=majority&appName=Cluster0'
```

Bu URL değerini de mongodb sitesinden kopyalayıp aldık:

```
npm install mongodb
```

[View MongoDB Node.js Driver installation instructions.](#)

### 3. Add your connection string into your application code

☐ View full code sample

```
mongodb+srv://gizemavci:<password>@cluster0.pxpsfvj.mongodb.net/?
retryWrites=true&w=majority&appName=Cluster0
```

Replace **<password>** with the password for the **gizemavci** user. Ensure any option params are [URL encoded](#).

#### RESOURCES

[Get started with the Node.js Driver](#)

[Node.js Starter Sample App](#)

[Access your Database Users](#)

[Troubleshoot Connections](#)

[Go Back](#)

[Close](#)

[Review setup steps](#)

<password> yazan yeri oluşturduğumuz şifre değerini girdik. Rapor içinde şifre görsün istemediğimden ben şifreyi eklemediğim versiyonunu ekledim.

Veri tabanı oluşturduğumuz zaman sunucuda birden fazla veri tabanı olabilir. Mesela su anda task-manager uygulaması yapıyoruz fakat başka zaman da başka uygulama için oluşturabiliriz. Farklı uygulamalar için ayrı ayrı sunuculardan hesap açmayıp tek sunucu üzerinde birden fazla veri tabanı oluşturabiliriz. Bu yüzden oluşturacağımız veri tabanına bir isim vermeliyiz. Biz kodumuzda **databaseName** değişkeninde isim değerimizi tuttuk.

```
const databaseName = "task-manager"
```

Ardından mongodb üzerindeki aşağıda bulunan kodu sayfamıza ekledik. Bu sayfayı bağlan kısmından **driver** butonunun içerisinden aldık.

### 3. Add your connection string into your application code

 View full code sample

```
const { MongoClient, ServerApiVersion } = require('mongodb');
const uri = "mongodb+srv://gizemavci:<password>@cluster0.pxpsfvj.mongodb.net/?retryWrite

// Create a MongoClient with a MongoClientOptions object to set the Stable API version
const client = new MongoClient(uri, {
  serverApi: {
    version: ServerApiVersion.v1,
    strict: true,
    deprecationErrors: true,
  }
});

async function run() {
  try {
    // Connect the client to the server (optional starting in v4.7)
    await client.connect();
    // Send a ping to confirm a successful connection
    await client.db("admin").command({ ping: 1 });
    console.log("Pinged your deployment. You successfully connected to MongoDB!");
  } finally {
    // Ensures that the client will close when you finish/error
    await client.close();
  }
}
run().catch(console.dir);
```

Replace <password> with the password for the **gizemavci** user. Ensure any option params are [URL encoded](#).

Kodun son haline bakmamız gerekirse:

```
const { MongoClient, ServerApiVersion } = require("mongodb");
const uri =
  "mongodb+srv://gizemavci:<password>@cluster0.pxpsfvj.mongodb.net/?retryWrites=true&w=majority&appName=Cluster0";

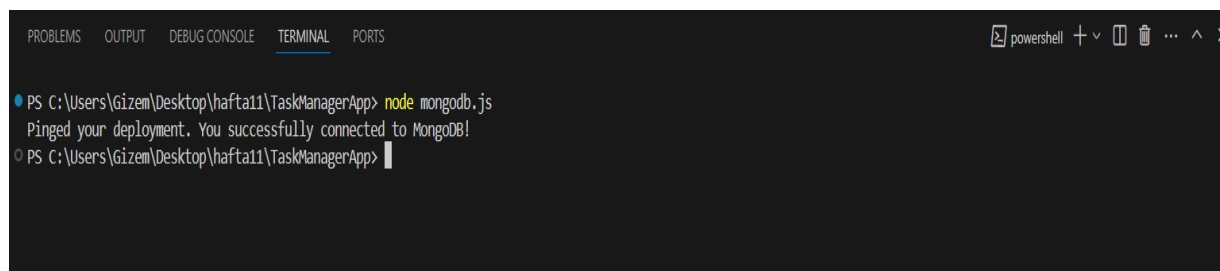
const dbName = "task-manager"

// Create a MongoClient with a MongoClientOptions object to set the Stable API version
const client = new MongoClient(uri, {
  serverApi: {
    version: ServerApiVersion.v1,
    strict: true,
    deprecationErrors: true,
  },
});

async function run() {
  try {
    // Connect the client to the server (optional starting in v4.7)
    await client.connect();
    // Send a ping to confirm a successful connection
    await client.db("admin").command({ ping: 1 });
    console.log(
      "Pinged your deployment. You successfully connected to MongoDB!"
    );
  } finally {
    // Ensures that the client will close when you finish/error
    await client.close();
  }
}
run().catch(console.dir);
```

Bu şekilde oldu ve yine password kısmını çalıştırırken değiştirdim.

Çalıştırdığımda da şu çıktıyı aldım:



The screenshot shows a PowerShell terminal window with the following content:

```
PS C:\Users\Gizem\Desktop\hafta11\TaskManagerApp> node mongodb.js
Pinged your deployment. You successfully connected to MongoDB!
PS C:\Users\Gizem\Desktop\hafta11\TaskManagerApp>
```

Yani başarılı bir şekilde mongodb sunucusuna bağlantı gerçekleştirdim.

Ardından veri tabanına kayıt atmaya baktık. Daha önceden task-manager adında oluşturmuştuk.

Client üzerinden database in isim bilgisini aldık ve db değişkeninde tuttuk. Sonra o database isimi altında bir collection oluşturduk ismine de users değerini girdik. Burada kullanıcıya ait verileri tuttuk. Tek bir değişken eklediğimizden insertOne komutunu kullandık. Ve kullanıcımıza name ve age bilgileri ekledik.

```
const db = client.db(databaseName);
await db.collection("users").insertOne({
  name: "Gizem",
  age: 21,
});
```

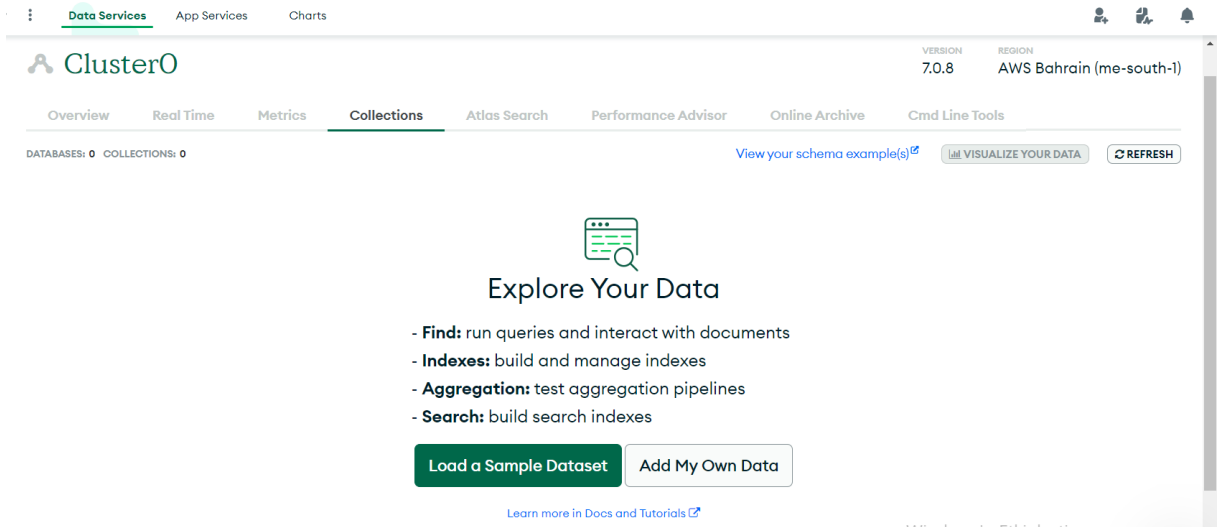
Bu kod parçasını da try bloğunun alt kısmına ekledik. Yani şu kısma:

```
13
14 async function run() {
15   try {
16     // Connect the client to the server (optional starting in v4.7)
17     await client.connect();
18     // Send a ping to confirm a successful connection
19     await client.db("admin").command({ ping: 1 });
20     console.log(
21       "Pinged your deployment. You successfully connected to MongoDB!"
22     );
23     const db = client.db(databaseName);
24     await db.collection("users").insertOne({
25       name: "Gizem",
26       age: 21,
27     });
28   } finally {
29     // Ensures that the client will close when you finish/error
30     await client.close();
31   }
32 }
```

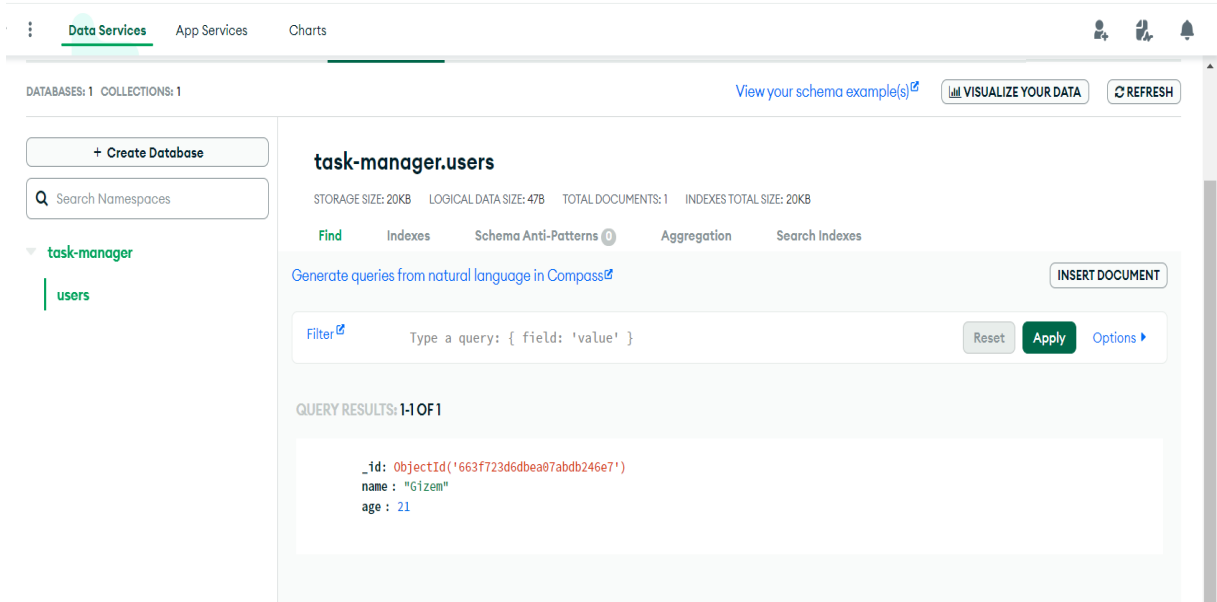
Kodu çalıştırdığımızda da:

```
PS C:\Users\Gizem\Desktop\hafta11\TaskManagerApp> node mongodb.js
Pinged your deployment. You successfully connected to MongoDB!
PS C:\Users\Gizem\Desktop\hafta11\TaskManagerApp>
```

Mongodb ye bakarsak da sayfa içerisinde collection kısmının içerisinde bulunmakta:



Eğer sayfa içerisindeyse böyle bir ekran karşımıza gelir. Verileri görmek için güncellememiz yani refresh etmemiz gerekir. Bu yüzden sayfadaki sağ üstte bulunan refresh butonuna tıklamalıyız. Ardından karşımıza sayfa şu şekilde çıkar:



Sayfayı incelersek üst solda bulunan database ve collection değeri güncellenip 1 değerini aldı. Task-manager adında database oluşturulup altından users adında kullanıcı bilgilerinin bulunduğu veri collection u oluşturuldu. Ve gönderdiğimiz name ve age bilgileri de sayfamızın içerisinde mevcuttur.

Ardından ilk verimizi ve bağlantı kısmını yorum satırına alıp yeni veri eklemeyi denedik.

O da şu şekilde gerçekleşti:

```

15  async function run() {
16    try {
17      // // Connect the client to the server (optional starting in v4.7)
18      // await client.connect();
19      // // Send a ping to confirm a successful connection
20      // await client.db("admin").command({ ping: 1 });
21      // console.log(
22      //   "Pinged your deployment. You successfully connected to MongoDB
23      // );
24      // const db = client.db(databaseName);
25      // await db.collection("users").insertOne({
26      //   name: "Gizem",
27      //   age: 21,
28      // });
29
30      const db = client.db(databaseName);
31      await db.collection("users").insertOne({
32        name: "Kadir",
33        age: 24,
34      });
35    } finally {
36      // Ensures that the client will close when you finish/error
37      await client.close();
38    }

```

Burada da sayfaya girip refresh ettiğimizde şu çıktıyı aldık:

The screenshot shows the MongoDB Compass interface. On the left, the 'task-manager' database is selected, and the 'users' collection is highlighted. The main panel displays the 'task-manager.users' collection with the following statistics: STORAGE SIZE: 36KB, LOGICAL DATA SIZE: 94B, TOTAL DOCUMENTS: 2, INDEXES TOTAL SIZE: 36KB. Below the statistics, there are tabs for 'Find', 'Indexes', 'Schema Anti-Patterns', 'Aggregation', and 'Search Indexes'. The 'Find' tab is active, showing a query filter bar with the text 'Type a query: { field: 'value' }'. Below the filter bar, the query results are displayed, showing two documents:

```

{
  "_id": ObjectId('663f723d6d6ea97abdb246e7'),
  "name": "Gizem",
  "age": 21
}

{
  "_id": ObjectId('663f740752ae53ce51164816'),
  "name": "Kadir",
  "age": 24
}

```

Kod aynı şekilde çalıştı çünkü client kısmı fonksiyonel. Onu yazmasak da gerçekleşiyor. Çünkü biz yazmasak bile kapalı bir mekanizma olarak arka kısmında o bağlantıyı gerçekleştiriyor. Bu yüzden kodun güncel halinde o kısmı silerek devam ettim.

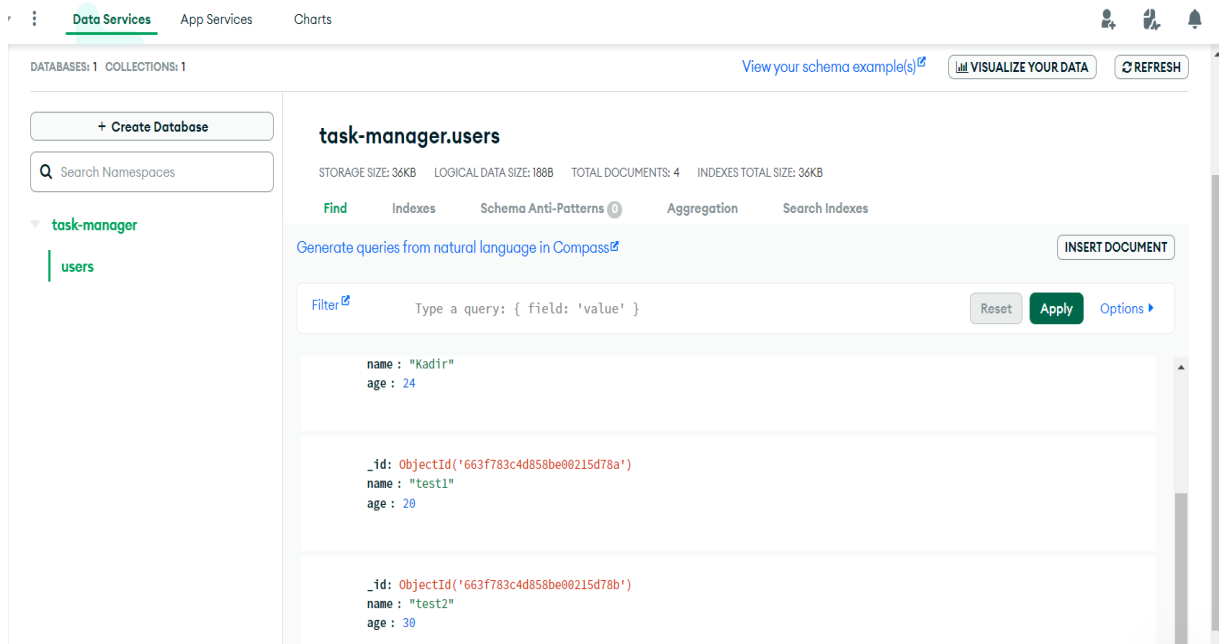
Az önce InsertOne ile tek bir veri ekleme işlemi gerçekleştirmiştik şimdi de InsertMany ile birden fazla veri ekleme kısmını gerçekleştirmeye çalıştık.



InsertMany dizi alabilir bu şekilde birden fazla veri gönderebiliriz.

```
const db = client.db(databaseName);
await db.collection("users").insertMany([
  {
    name: "ABC",
    age: 1,
  },
  {
    name: "DEF",
    age: 2,
  },
]);
```

Deneme olacak şekilde yukarıdaki gibi kod yazdık ve kodu çalıştırdık. Sayfaya girip refresh ettiğimizde şu şekilde oluştu:



Deneme testimiz başarılı bir şekilde çalıştı. Daha önce tek tek nasıl veri eklediyseniz şimdi de onu tek seferde gerçekleştirdik.

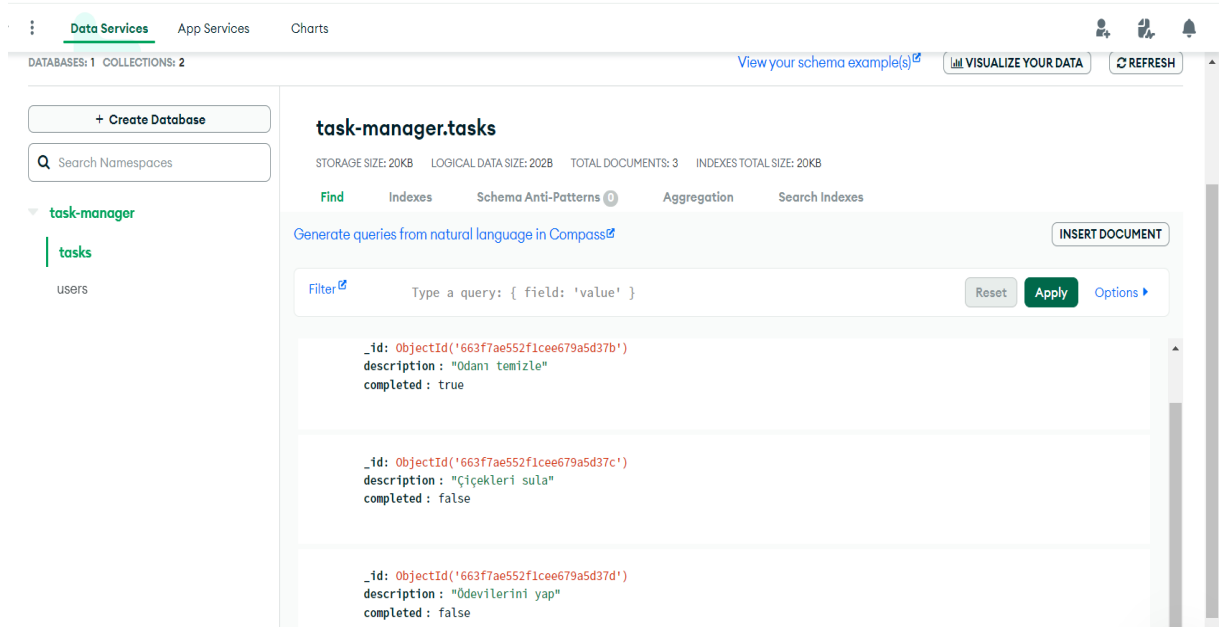
Bu projenin amacı taskleri yani görevleri tutmak bu yüzden de şimdi gerçek işimiz olan taskleri tutan bir collection oluşturduk.

Bunun içerisinde de üç tane görev tuttuk ve her birinin bir description değeri var. Burada görev isimlerini string olacak şekilde girdik.

Daha sonra bir de completed kısımları var. Bu da görevin yapıp yapılmadığını belirten bir değişken. Tür olarak da boolean değer aldı.

```
async function run() {
  try {
    const db = client.db(databaseName);
    await db.collection("tasks").insertMany([
      {
        description: "Odanı temizle",
        completed: true,
      },
      {
        description: "Çiçekleri sula",
        completed: false,
      },
      {
        description: "Ödevlerini yap",
        completed: false,
      },
    ]);
  } finally {
    // Ensures that the client will close when you finish/error
    await client.close();
  }
}
```

Kodumuzu çalıştırdığımızda sayfamız şöyle bir hal aldı:



The screenshot shows the MongoDB Compass interface. On the left sidebar, the 'task-manager' database is expanded, showing the 'tasks' collection. The main panel displays the 'task-manager.tasks' collection with three documents. The documents are as follows:

_id	description	completed
ObjectId('663f7ae552f1cee679a5d37b')	"Odanı temizle"	true
ObjectId('663f7ae552f1cee679a5d37c')	"Çiçekleri sula"	false
ObjectId('663f7ae552f1cee679a5d37d')	"Ödevlerini yap"	false

Sayfamıza baktığımızda yine sol üste bulunan collection değeri bir arttı çünkü artık users dan başka tasks adında collection da mevcut. Artı olarak da tüm verilerimiz sayfaya eklenmiş durumda.

Ardından unique bir id değeri almayı gördük.

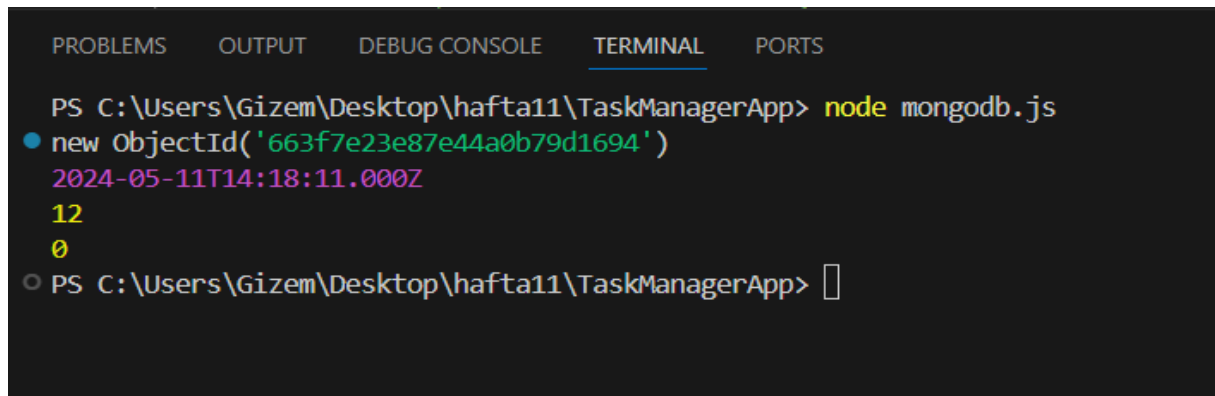
```
const { MongoClient, ServerApiVersion, ObjectId } = require("mongodb");

async function run() {
  try {
    const id = new ObjectId();
    console.log(id);
    console.log(id.getTimestamp());
    console.log(id.id.length);
    console.log(id.toHexString().length);
    const db = client.db(databaseName);
    await db.collection("users").insertOne({
      _id: id,
      name: "Özlem",
      age: 23,
    });

  } finally {
    // Ensures that the client will close when you finish/error
    await client.close();
  }
}
```

ObjectId üzerinden id oluşturduk ve id değerini oluşturulduğu tarihi uzunluğu gibi bilgileri konsolda yazdırdık, artı olarak da yeni bir kullanıcı verisi ekleyip bu değerle id oluşturabildik.

Kodu çalıştırdığımızda id hakkında bilgileri bize verdi:



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS C:\Users\Gizem\Desktop\hafta11\TaskManagerApp> node mongodb.js
● new ObjectId('663f7e23e87e44a0b79d1694')
  2024-05-11T14:18:11.000Z
  12
  0
○ PS C:\Users\Gizem\Desktop\hafta11\TaskManagerApp> 
```

Ardından sayfaya baktığımızda:

```
_id: ObjectId('663f7e23e87e44a0b79d1694')
name : "Özlem"
age : 23
```

Tıpkı konsolda da çıktığı gibi 663f7e23e87e44a0b79d1694 id numaralı veri oluşturuldu.

İd değerini manuel olarak da girebiliriz onda da yine aynı şekilde girip id yazmak yerine istediğimiz bir değer gireriz. O da şu şekilde olur:

```
const db = client.db(databaseName);
await db.collection("users").insertOne({
  _id: 10,
  name: "Deneme",
  age: 10,
});
```

Sayfamızda da şu şekilde gözükür:

```
_id: 10
name : "Deneme"
age : 10
```

Son olarak da veriler içerisinde kullanıcı arama kısmına baktık.

Bunu findOne ile sağladık. findOne içerisinde girdiğimiz kullanıcıya ait bilgilerle bu isimde veya yaşta kullanıcı var mı sorgusu yaptık. Var olma durumunda ne döndüreceğini hata olma durumunda ile hangi çıktıyı alacağını belirledik. Mesela Gizem isminde bir kullanıcı var mı sorgusu şu kod ile sağlandı.

```
Await db.collection("users").findOne({ name: "Gizem" });
```

Ardından bulup bulmama durumuna neler olacağını da şu kod ile sağladık:

```
const db = client.db(databaseName);

const user = await db
  .collection("users")
  .findOne({ name: "Gizem" }, (error, user) => {
    if (error) {
      console.log("Kullanıcı bulunamadı");
    }
  });
console.log(user);
```

Mesela ismi gizem olan birden fazla kişi varsa kod bize ilk bulduğu veriyi döndürecektir.

Yukarıdaki kodu çalıştırdığımızda şu çıktıyı elde ederiz:

```
● PS C:\Users\Gizem\Desktop\hafta11\TaskManagerApp> node mongodb.js
{
  _id: new ObjectId('663f723d6dbea07abdb246e7'),
  name: 'Gizem',
  age: 21
}
○ PS C:\Users\Gizem\Desktop\hafta11\TaskManagerApp> █
```

Şimdi de sayfada kontrol etmek amacıyla verimize bakalım:

```
_id: ObjectId('663f723d6dbea07abdb246e7')
name : "Gizem"
age : 21
```

İki age değeri de id değeri de aynı çıkmış bulunmakta, yani doğru veriyi buldu.