

BURSA TEKNİK ÜNİVERSİTESİ

**Mühendislik ve Doğa Bilimleri Fakültesi – Bilgisayar
Mühendisliği Bölümü**



BLM0470 NodeJS İle Web Programlama

2023-2024 Bahar Dönemi

10.Hafta Raporu

Berkan SERBES – 22360859353

İçindekiler

1. NoSQL Veritabanları	3
1.1 NoSQL ve SQL Veritabanları Arasındaki Farklar	4
1.2 MongoDB.....	5
1.2.1 MongoDB Atlas Hesabı Oluşturma.....	6
2. Görev Yönetim Uygulaması Geliştirme	10

1. NoSQL Veritabanları

NoSQL, “**Not only SQL**” kavramının kısaltması olup, geleneksel SQL tabanlı ilişkisel veritabanlarına alternatif olarak geliştirilmiş bir veritabanı türüdür. NoSQL veritabanları, yapısal olmayan (JSON, XML, metin, grafik vb.) ve genellikle dağıtık sistemlerde büyük veri kümelerini depolamak, işlemek ve yönetmek için tasarlanmıştır. NoSQL veritabanları, çeşitli veri modellerini destekleyebilir, dağıtık ve ölçeklenebilir mimarilere uyum sağlayabilir, yüksek performans sağlayabilir ve esneklik sunabilir. Bu özellikler, özellikle büyük miktarda yapısal olmayan verinin hızlı bir şekilde işlenmesi gereken modern uygulama geliştirme senaryolarında tercih edilmesine olanak tanır.

NoSQL veritabanları, SQL tabanlı ilişkisel veritabanlarının tipik kısıtlamalarını aşmak için tasarlanmıştır. Örneğin, ilişkisel veritabanlarında sıkça görülen şemaya bağlılık ve katı veri yapısı kısıtlamaları NoSQL veritabanlarında genellikle bulunmaz. Bu sayede NoSQL veritabanları, daha esnek veri modellemesi ve depolama seçenekleri sunabilir.

Bu veritabanları, çeşitli veri modellerini (doküman tabanlı, anahtar-değer tabanlı, sütun tabanlı, grafik tabanlı vb.) destekler.

NoSQL veritabanları, çeşitli kullanım senaryolarına ve gereksinimlere göre farklı türlerde gelir:

- 1. Anahtar-Değer (Key-Value) tabanlı depolama:** Verileri anahtar-değer çiftleriyle depolarlar ve yüksek performanslı veri depolama ve önbellekleme için idealdirler. Örneğin, Redis, Amazon DynamoDB.
- 2. Doküman tabanlı depolama:** Belge tabanlı veri modelini destekler. JSON veya BSON gibi yapısal olmayan verilerle çalışır. Örneğin, MongoDB.
- 3. Sütun tabanlı depolama:** Verileri sütunlar halinde depolarlar ve çok büyük ölçekli veriler için uygundur. Örneğin, Apache Cassandra.
- 4. Grafik Veritabanları:** Grafik yapılarına dayalı veri modellemesi sağlarlar ve ilişkisel verileri temsil etmek için kullanılırlar. Örneğin, Neo4j.

NoSQL veritabanlarının bazı örnekleri arasında MongoDB, Cassandra, Couchbase, Redis, Amazon DynamoDB, Apache HBase ve Apache CouchDB gibi popüler sistemler bulunmaktadır. Her bir NoSQL veritabanı farklı kullanım senaryolarına, veri modellerine ve performans gereksinimlerine hitap eder. Örneğin, MongoDB belgelere dayalı bir veritabanıdır ve ölçeklenebilir, yüksek performanslı uygulamalar için idealdir. Cassandra dağıtık bir sütun tabanlı veritabanıdır ve özellikle büyük ölçekli, yüksek yoğunluklu yazma işlemleri gerektiren uygulamalarda kullanılır. Redis, anahtar-değer tabanlı bir veritabanıdır ve yüksek performanslı veri depolama ve önbellekleme için idealdir.

1.1 NoSQL ve SQL Veritabanları Arasındaki Farklar

NoSQL ve SQL veritabanları, farklı veri depolama ve yönetim yaklaşımlarını temsil eder. NoSQL, yapısal olmayan ve büyük ölçekli verilerle çalışmak için esnek bir çözüm sunar. SQL ilişkisel veritabanlarında daha yapılandırılmış bir yaklaşım sunar. Bu farklılıklar, veri işleme gereksinimlerine ve kullanım senaryolarına göre tercih edilen veritabanı türünü belirlemede etkilidir. Bu başlık altında, NoSQL ve SQL veritabanları arasındaki temel farkları inceliyoruz.

SQL veritabanları:

1.Yapılandırılmış Veri Modeli: SQL (Structured Query Language) tabanlı veritabanları, geleneksel olarak yapılandırılmış veri modeline dayanır. Veri, tablolar halinde saklanır ve bu tablolar arasında ilişkiler (relationship) bulunur. Örneğin, bir ilişkisel veritabanında kullanıcılar ve siparişler arasında bir ilişki olabilir.

2. Şema Tabanlı: SQL veritabanları, şema tabanlıdır. Bu, veri yapısının önceden belirlenmiş bir şemaya göre oluşturulduğu ve bu şemanın veri türlerini, sınırlayıcıları (constraints) ve ilişkileri tanımladığı anlamına gelir. Her bir tablo için belirli bir yapı önceden belirlenmiştir.

3. ACID Uyumluluğu: SQL veritabanları, ACID (Atomicity, Consistency, Isolation, Durability) özelliklerine dayanarak veri bütünlüğünü sağlar. Bu, veritabanında gerçekleşen işlemlerin güvenilirliğini ve tutarlılığını sağlar. Örneğin, bir işlem ya tamamıyla gerçekleşir ya da hiç gerçekleşmez (atomicity).

NoSQL veritabanları:

1. Yapılandırılmamış veya Yarı Yapılandırılmış Veri Modeli: NoSQL (Not Only SQL) veritabanları, yapılandırılmamış veya yarı yapılandırılmış verileri saklamak için tasarlanmıştır. Bu, verilerin JSON (JavaScript Object Notation), XML veya benzeri formatlarda olduğu anlamına gelir. Veriler arasında bir ilişki olabilir ancak bu ilişki genellikle daha esnek.

2. Esnek Şema: NoSQL veritabanları, esnek bir şemaya sahiptir. Bu, veri yapısının önceden tanımlanmış bir şemaya bağlı olmadığı ve gerektiğinde değiştirilebildiği anlamına gelir. Veri modeli, uygulama gereksinimlerine ve veri türüne göre kolayca adapte edilebilir veya genişletilebilir.

3. Çeşitli Veri Modelleri: NoSQL veritabanları, farklı veri modellerini destekler. Bunlar arasında anahtar-değer tabanlı, belge tabanlı, sütun tabanlı ve grafik tabanlı veri modelleri bulunur. Bu, çeşitli kullanım senaryolarına uygun çözümler sunar.

4. Ölçeklenebilirlik ve Performans: NoSQL veritabanları, ölçeklenebilirlik ve yüksek performans üzerine odaklanır. Büyük hacimli verilerle ve dağıtılmış sistemler üzerinde çalışabilirler. NoSQL veritabanları, karmaşık sorgulara ve işlemlere olanak tanıırken hızlı okuma ve yazma işlemleri sunar.

5. ACID Uyumluluğu: NoSQL veritabanları, ACID özelliklerinin tam olarak uygulanmasını gerektirmez. Bu nedenle, bazı durumlarda esneklik sağlarlar ve performansı artırabilirler.

1.2 MongoDB

MongoDB, belge tabanlı bir NoSQL veritabanı yönetim sistemidir. MongoDB'nin temel özelliği, BSON (Binary JSON) formatında belgelerle çalışmasıdır. Bu, verilerin JSON benzeri bir formatta depolanmasını sağlar ve uygulamaların verileri daha doğal bir şekilde işlemesine imkan tanır. Ayrıca, MongoDB dinamik şemalara olanak tanır, yani her belge farklı bir yapıya sahip olabilir. Bu özellik, uygulamalarda esneklik sağlar ve veri modellemesini kolaylaştırır.

MongoDB'nin dağıtılmış mimarisi, yüksek ölçeklenebilirlik ve performans sunar. Bu sayede, büyük veri kümelerini kolayca yönetebilir ve büyüyen veri miktarlarına hızlı bir şekilde yanıt verebilir. Veriler, birden çok sunucu üzerinde dağıtılarak yük dengesi sağlanır ve yüksek kullanılabilirlik sağlanır. Ayrıca, MongoDB'nin güçlü sorgu diline sahip olması ve karmaşık sorguları desteklemesi, veri analitiği ve iş zekası uygulamaları için ideal bir tercih haline getirir.

MongoDB'nin sahip olduğu bazı özellikler MongoDB'nin sıkça kullanılan bir veritabanı olmasında oldukça önem taşır. Bu özellikler şunlardır:

Sürücüler (Drivers): Sürücüler, sunucularda bulunan ve MongoDB ile etkileşime girebilen araçlar ve yazılımlardır. MongoDB bir çok programlama dilini desteklemektedir. Bu diller aşağıdaki görseldeki gibidir.

Client Libraries



C



C++



C#



Go



Java



Kotlin



Node.js



PHP



Python



Ruby



Rust



Scala



Swift



TypeScript

Depolama Motoru (Storage Engine): Depolama motoru, MongoDB'nin diskte ve bellekte depolanan veri miktarıyla ilgilenen bölümüdür. Ayrıca verileri aramak için kullanılan arama motorlarıyla da ilgilidir. WiredTiger, MongoDB'nin varsayılan arama motorudur ancak bellek içi depolama motoru ve MongoDB'nin kullandığı şifreli depolama motoru gibi başka depolama motorları da vardır.

MongoDB Shell: MongoDB Shell, MongoDB veritabanıyla etkileşimde bulunmak için kullanılan MongoDB'nin resmi komut satırı arabirimidir (CLI). Komut satırı tabanlı bir arabirimdir ve MongoDB veritabanı yönetimi için bir dizi komut ve işlem sunar. Kullanıcılar, MongoDB Shell'i kullanarak veritabanına bağlanabilir, sorgular çalıştırabilir, veritabanı yapılandırmasını yönetebilir ve veri manipülasyonu yapabilir.

MongoDB, birçok farklı endüstride geniş bir kullanım alanına sahiptir. Web uygulamaları, mobil uygulamalar, e-ticaret platformları, büyük veri analitiği ve bulut tabanlı uygulamalar gibi birçok alanda tercih edilmektedir. Ayrıca, MongoDB'nin zengin bir ekosisteme sahip olması, geliştiricilere daha verimli bir şekilde çalışma imkanı sunar.

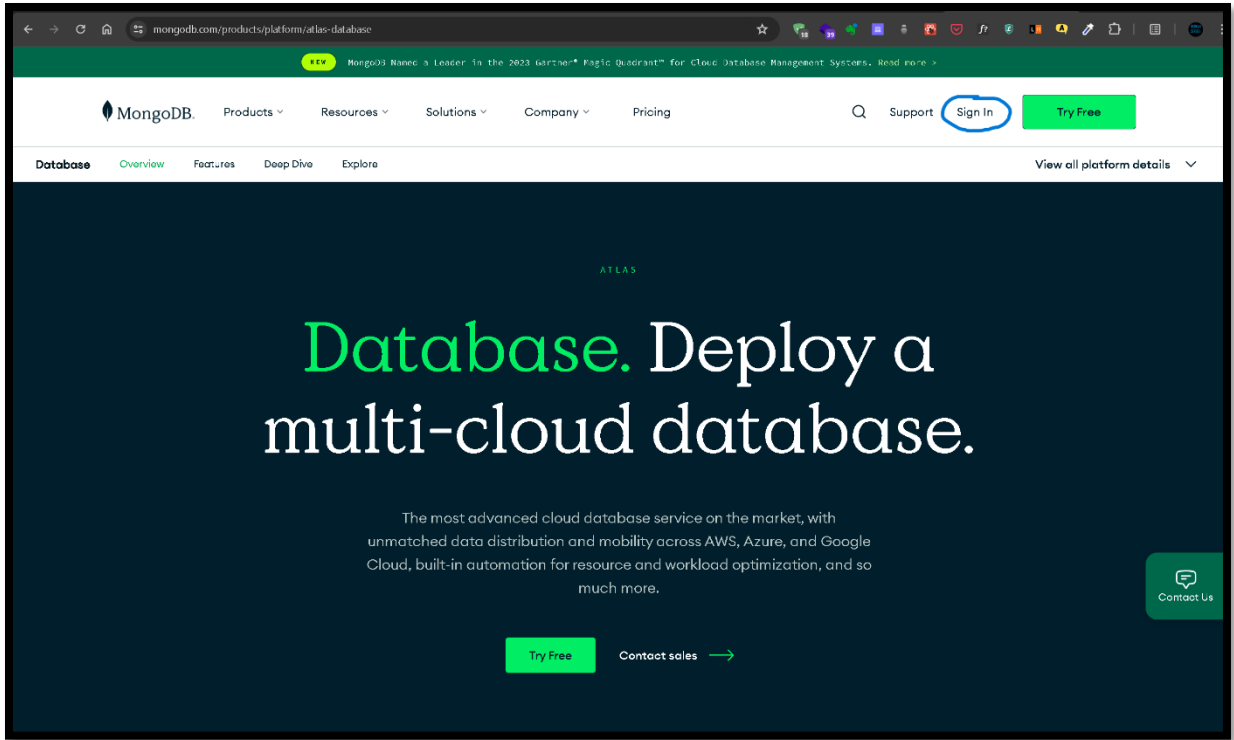
Sonuç olarak, MongoDB esnekliği, ölçeklenebilirliği ve performansı modern uygulamalar için ideal bir veritabanı çözümüdür. Gelişmiş belge tabanlı mimarisi ve güçlü sorgu yetenekleriyle, MongoDB, veri yönetimi ihtiyaçlarını karşılamak için güvenilir bir seçenek sunar.

1.2.1 MongoDB Atlas Hesabı Oluşturma

MongoDB Atlas, MongoDB'nin bulut tabanlı veritabanı hizmetidir. Bu hizmet, MongoDB'nin esneklik, ölçeklenebilirlik ve güvenilirliğini bulut ortamında sunar. Atlas, kullanıcıların MongoDB veritabanılarını kolayca dağıtmasına, yönetmesine ve ölçeklemesine olanak tanır. Atlas, birden çok bulut sağlayıcısında (AWS, Azure, Google Cloud) ve farklı bölgelerde veri merkezlerinde çalışabilir. Ayrıca Atlas, otomatik yedekleme, güvenlik kontrolleri ve yüksek kullanılabilirlik gibi özellikler sunar, böylece kullanıcılar veritabanılarını güvenli ve işlevsel bir şekilde işletebilirler. Bu, geliştiricilere ve işletmelere uygulama geliştirme süreçlerinde ve operasyonlarında önemli ölçüde kolaylık sağlar.

Windows işletim sisteminde MongoDB Atlas hesabı oluşturmak için aşağıdaki adımları takip edebilirsiniz:

- Web tarayıcınızda aşağıdaki bağlantıya gidin.
<https://www.mongodb.com/products/platform/atlas-database>
- Bulunduğunuz sayfada yer alan sağ üst köşedeki Sign In butonuna tıklayın.




- Eğer herhangi bir kayıtlı hesabınız yoksa veya yeni bir hesap açmak istiyorsanız sayfanın solunda yer alan formda Sign Up seçeneğine tıklayın. Bu seçeneğe tıkladığınızda karşınıza aşağıdaki görseldeki gibi bir hesap oluşturma formu gelecektir.

MongoDB Atlas

- ✓ **Work with your data as code**
Documents in MongoDB map directly to objects in your programming language. Modify your schema as your apps grow over time.
- ✓ **Focus on building, not managing**
Let MongoDB Atlas take care of the infrastructure operations you need for performance at scale, from always-on security to point-in-time recovery.
- ✓ **Simplify your data dependencies**
Leverage application data for full-text search, real-time analytics, rich visualizations and more with a single API and minimal data movement.

Sign up

See what Atlas is capable of for free

 Sign up with Google

First Name*
Berkan

Last Name*
Serbes

Company

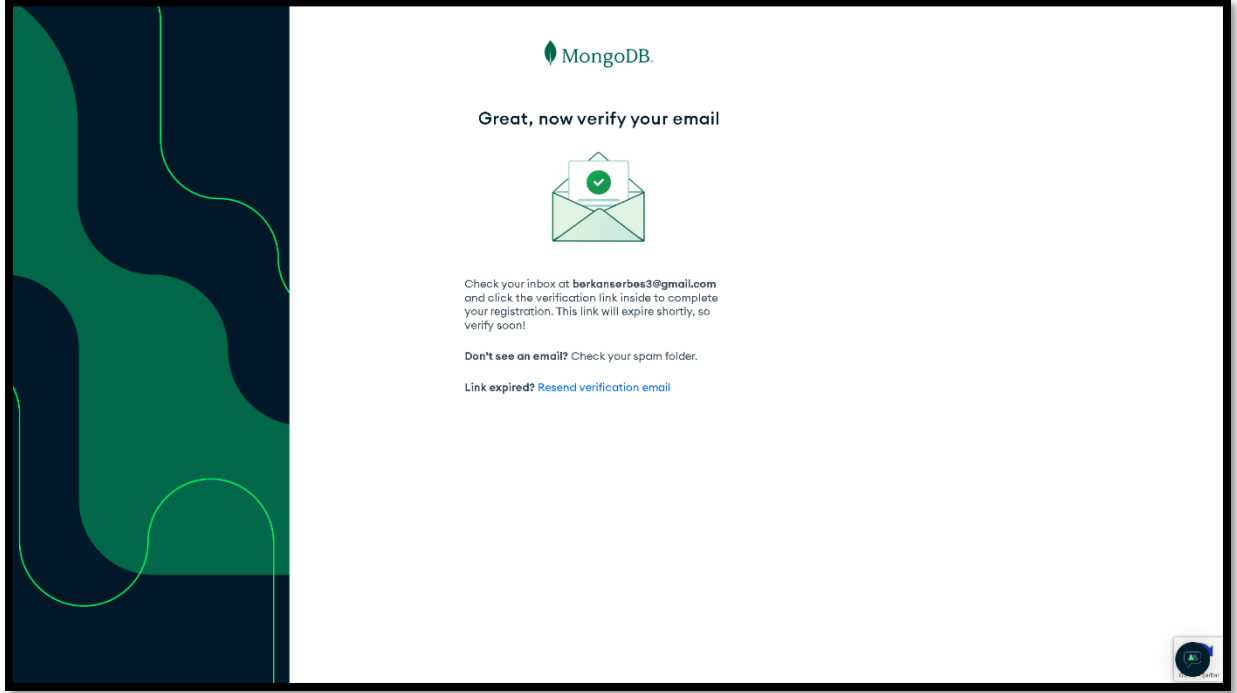
Email*
berkanserbes3@gmail.com

Password*
••••••••••

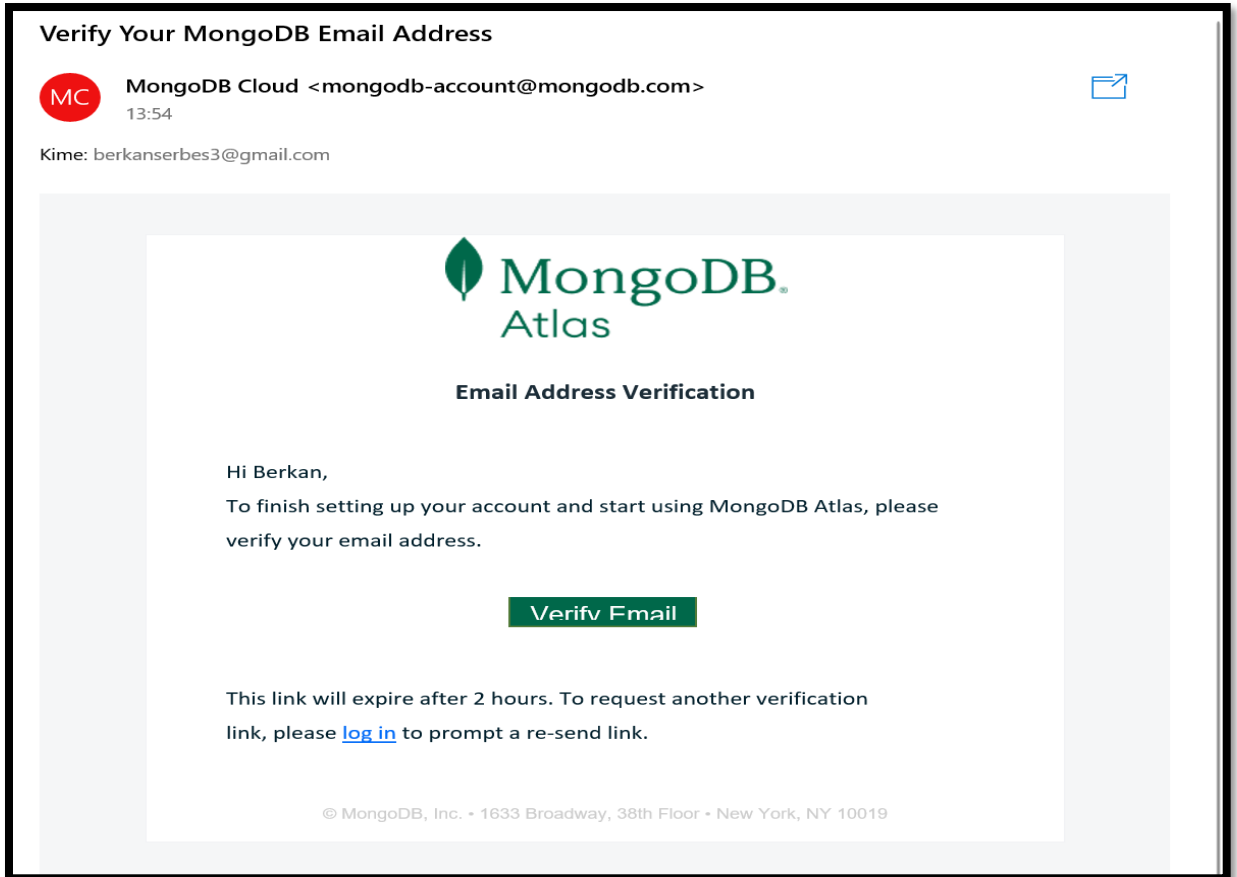
☒ I agree to the [Terms of Service](#) and [Privacy Policy](#).

Create your Atlas account

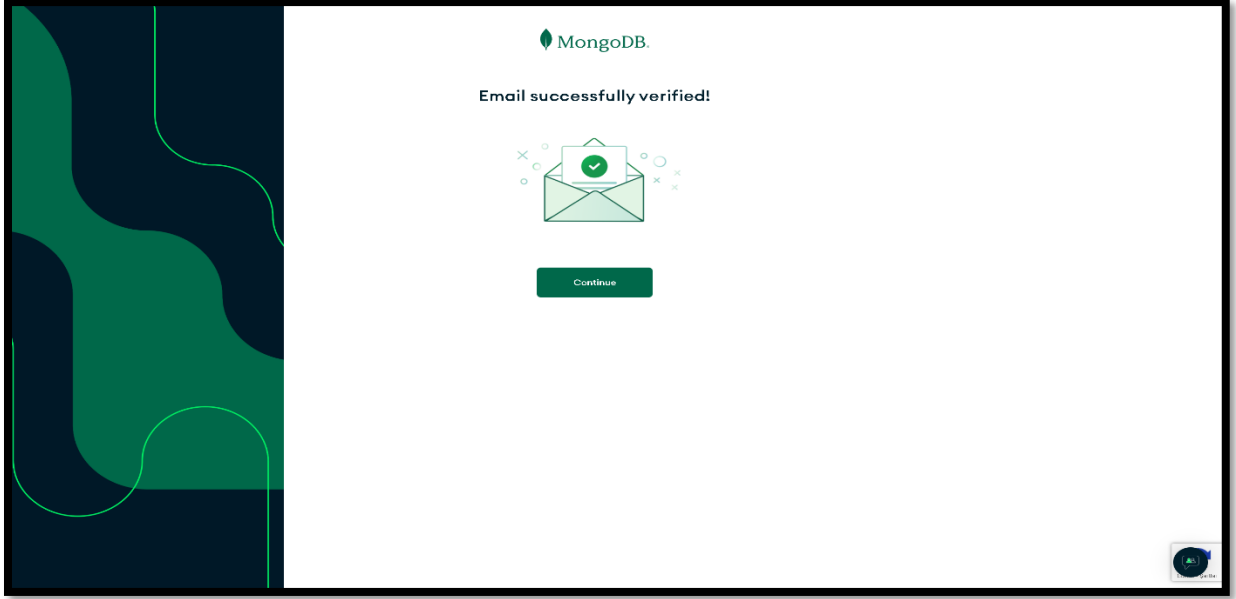
- Gerekli bilgileri girdikten sonra **Create your Atlas account** butonuna tıklayın.
- Sonrasında kaydolduđunuz email hesabına bir aktivasyon bađlantısı g nderilecektir.



- Epostanıza gelen dođrulama mailinde yer alan Verify Email butonuna tıklayın ve hesabınızı dođrulayın.



- Hesabınız başarılı bir şekilde doğrulandığında aşağıdaki görseldeki gibi bir sayfaya yönlendirileceksiniz.

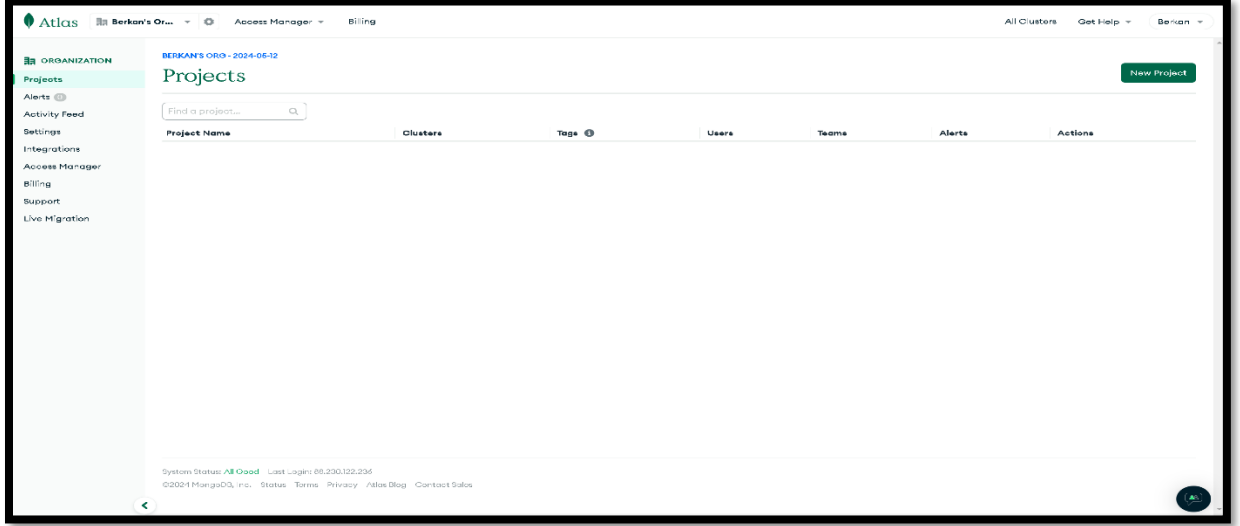


- Continue butonuna bastıktan sonra karşınıza şu şekilde bir form sayfası çıkacaktır. Bu formu kendi kullanım şeklinize göre doldurun ve Finish butonuna basın ve üyelik oluşturma işlemi başarıyla tamamlanacaktır.

A screenshot of the MongoDB Atlas onboarding form. The page has a light gray background. At the top left is the Atlas logo, followed by the text "Welcome to Atlas. Let's build something great." and a subtext "Help us tailor your experience by taking a minute to answer the questions below." The form is divided into two main sections: "GETTING TO KNOW YOU" and "GETTING TO KNOW YOUR PROJECT". The "GETTING TO KNOW YOU" section contains two questions: "What is your primary goal?" with a dropdown menu showing "Build a project I have in mind", and "How long have you been developing software with MongoDB?" with a dropdown menu showing "1-6 months experience". The "GETTING TO KNOW YOUR PROJECT" section contains three questions: "What programming language are you primarily building on MongoDB with?" with a dropdown menu showing "JavaScript / Node.js", "What type(s) of data will your project use?" with a dropdown menu showing "Not sure" and a subtext "You can choose as many as you want", and "Will your application include any of the following architectural models?" with a dropdown menu showing "Not sure" and a subtext "You can choose as many as you want". At the bottom right of the form is a green "Finish" button.

2. Görev Yönetim Uygulaması Geliştirme

Uygulamamızı geliştirmeye ilk olarak MongoDB Atlas üzerinden sahip olduğumuz hesaba bir proje ve veritabanı oluşturmakla başlayacağız. Aşağıdaki görselde yer alan sağ üstte bulunan New Project butonuna basalım.



Açılan iki aşamalı sayfada bizden projemizin adını, projemiz hakkında bilgi içeren etiketler ekleme, bu projede çalışacak kişileri ekleme gibi sorular yer alacaktır. Bunları aşağıdaki görsellerdeki gibi doldurabilirsiniz veya kendinize göre doldurabilirsiniz.

BERKAN'S ORG - 2024-05-12 > PROJECTS

Create a Project

Name Your Project

Add Members

Name Your Project

Project names have to be unique within the organization (and other restrictions).

task-manager

Add Tags (Optional)

Use tags to efficiently label and categorize your projects. A project can have a maximum of 50 tags. You can modify tags for the project later. [Learn more](#)

Key	Value	Actions
Select a key or enter your own	:	Select a value or enter your own
+ Add tag		

0 TAGS

Cancel

Next

BERKAN'S ORG - 2024-05-12 > PROJECTS

Create a Project

✓ Name Your Project

Add Members

Add Members and Set Permissions

Invite new or existing users via email address...

Give your members access permissions below.

berkanserbes3@gmail.com
(you)

Project Owner

Back

Cancel

Create Project

Bir sonraki adımımız projemiz için bir Cluster oluşturmak bunun için aşağıdaki görselde yer alan sayfada sol panelde yer alan Database sekmesine tıklayalım ve ilgili sayfada yer alan Build a Cluster butonuna tıklayalım.

task-manager

Data Services

App Services

Charts

Overview

DEPLOYMENT

Database

Data Lake

SERVICES

Device & Edge Sync

Triggers

Data API

Data Federation

Atlas Search

Stream Processing

Migration

SECURITY

Backup

Database Access

Network Access

Advanced

New On Atlas

Goto

BERKAN'S ORG - 2024-05-12 > TASK-MANAGER

Clusters

Create a cluster

Choose your cloud provider, region, and specs.

Build a Cluster

Once your database is up and running, live migrate an existing MongoDB database into Atlas with our [Live Migration Service](#).

Bir sonraki adımımıza geçmeden önce yazılımda “**Cluster**” teriminin ne olduğunu anlayalım.

Bir yazılım ortamında "cluster", genellikle birlikte çalışan ve birbiriyle etkileşimde bulunan birden fazla bilgisayarın veya sunucunun bir araya gelmesiyle oluşan bir grup olarak tanımlanır. Bu grup, işlem gücünü artırmak, yüksek kullanılabilirlik sağlamak, hata toleransı oluşturmak ve büyük ölçekli uygulamaları desteklemek için kullanılır.

Cluster'lar, dağıtık sistemlerin temel yapı taşlarından biridir ve genellikle ağ üzerinde birbirleriyle iletişim kurarlar. Bir cluster, genellikle benzer donanım ve yazılım yapılandırmalarına sahip olan ve aynı işlevleri yerine getiren sunucu veya bilgisayarlardan oluşur. Ancak, farklı donanım ve yazılım yapılandırmalarına sahip olan sunucuları içerebilir ve hatta farklı coğrafi konumlarda bulunabilirler.

Cluster'lar genellikle aşağıdaki alanlarda kullanılır:

Yüksek kullanılabilirlik: Birden fazla sunucunun veya bilgisayarın bir araya gelmesiyle, bir sunucunun veya bileşenin başarısız olması durumunda hizmetin kesintisiz olarak devam etmesi sağlanabilir.

İşlem gücünün artırılması: Bir cluster, iş yükünü paralel olarak dağıtarak, bir uygulamanın daha fazla işlem gücüne sahip olmasını sağlar.

Hata toleransı: Bir cluster, bir bileşenin başarısız olması durumunda sistemdeki diğer bileşenlerin hizmeti sürdürmesine olanak tanır. Bu sayede, sistem genelinde hata toleransı sağlanır.

Büyük ölçekli veri depolama ve işleme: Büyük veri setlerini depolamak ve işlemek için kullanılabilirler. Veri parçalarını farklı sunucular arasında dağıtarak ve paralel işlem yaparak yüksek performans sağlayabilirler.

Paralel hesaplama: Cluster'lar, karmaşık hesaplamaları paralel olarak gerçekleştirebilirler, bu da hesaplama süresini önemli ölçüde azaltabilir.

Sonuç olarak, bir cluster, yazılım uygulamalarında yüksek performans, yüksek kullanılabilirlik ve hata toleransı gibi önemli avantajlar sağlayan bir yapıdır.

Cluster kavramını anladığımızıza göre kaldığımız yerden projemiz için bir cluster oluşturalım.

Deploy your cluster

Use a template below or set up advanced configuration options. You can also edit these configuration options once the cluster is created.

☐ M10 \$0.09/hour

For production applications with sophisticated workload requirements.

STORAGE	RAM	vCPU
10 GB	2 GB	2 vCPUs

☐ Serverless

For application development and testing, or workloads with variable traffic.

STORAGE	RAM	vCPU
Up to 1TB	Auto-scale	Auto-scale

☒ M0 Free

For learning and exploring MongoDB in a cloud environment.

STORAGE	RAM	vCPU
512 MB	Shared	Shared

✓ **Free forever!** Your M0 cluster is ideal for experimenting in a limited sandbox. You can upgrade to a production cluster anytime.

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

☒ Automate security setup ⓘ
☒ Preload sample dataset ⓘ

Provider

☒ aws ☐ Google Cloud ☐ Azure

Region

ⓘ

★ Recommended ⓘ ☒ Low carbon emissions ⓘ

Aşağıdaki adımda veritabanımız için veritabanı kullanıcısı oluşturmamız gerekiyor. Kullanıcı adı ve şifrenizi yazıp bir veritabanı kullanıcısı oluşturabilirsiniz.

Connect to my-cluster

1

2

3

Set up connection security Choose a connection method Connect

You need to secure your MongoDB Atlas cluster before you can use it. Set which users and IP addresses can access your cluster now. [Read more](#) ⓘ

1. Add a connection IP address

✓ Your current IP address (88.230.122.236) has been added to enable local connectivity. Add another later in [Network Access](#) ⓘ.

2. Create a database user

✓ A database user has been added to this project. Create another user later in [Database Access](#) ⓘ.

You'll need your database user's credentials in the next step.

Bu adımda sahip olduğumuz cluster'a hangi yoldan bağlantı kuracağımızı seçmemiz gerekiyor. Biz NodeJS projemiz üzerinden bağlantı kuracağımız için Drivers seçeneğini seçiyoruz.

Connect to my-cluster


✓

2

3

Set up connection securityChoose a connection methodConnect


Connect to your application



Drivers
Access your Atlas data using MongoDB's native drivers (e.g. Node.js, Go, etc.)


>

Access your data through tools




Compass
Explore, modify, and visualize your data with MongoDB's GUI

>




Shell
Quickly add & update data using MongoDB's Javascript command-line interface

>



MongoDB for VS Code
Work with your data in MongoDB directly from your VS Code environment

>



Atlas SQL
Easily connect SQL tools to Atlas for data analysis and visualization

>

Go Back

Close

Bu adım, bize seçmiş olduğumuz programlama dilinde mongodb ile nasıl bağlantı kurabileceğimizi gösteriyor. Driver olarak Node.js seçeneğini seçiyoruz. Aşağıdaki görselde yer alan 3.adımda bize bir connection string verilmiş. Bu connection stringi daha sonra uygulamamızda kullanmak için boş bir metin dosyasına kaydedelim. Connection stringi kaydederken **<password>** kısmını oluşturduğumuz veritabanı kullanıcısının şifresiyle değiştirmeyi unutmayalım.

Connect to my-cluster

✓

✓

3

Set up connection securityChoose a connection methodConnect

Connecting with MongoDB Driver

1. Select your driver and version

We recommend installing and using the latest driver version.

Driver	Version
Node.js	5.5 or later

2. Install your driver

Run the following on the command line

```
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://berkanserbes:<password>@my-cluster.avof9tn.mongodb.net/?
retryWrites=true&w=majority&appName=my-cluster
```

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

RESOURCES

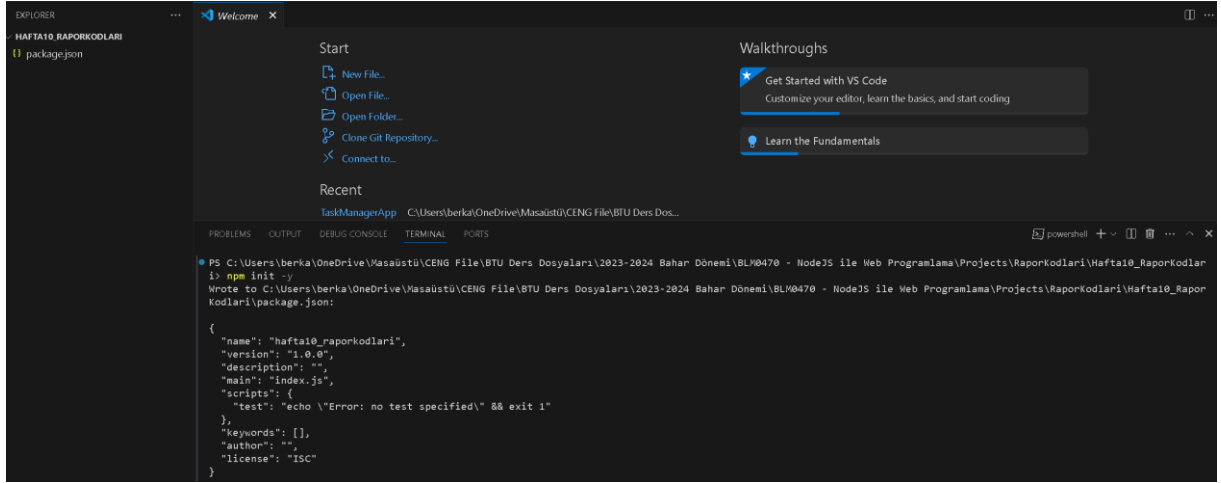
[Get started with the Node.js Driver](#)[Node.js Starter Sample App](#)

[Access your Database Users](#)[Troubleshoot Connections](#)

Go Back

Done

İlk olarak boş bir dizine bir klasör açıp o klasörü kod editörümüzde açalım ve komut satırında **npm init -y** komutunu kullanarak projemizi oluşturalım.



```
PS C:\Users\berka\OneDrive\Masaüstü\CENG File\BTU Ders Dosyaları\2023-2024 Bahar Dönemi\BLM0470 - NodeJS ile Web Programlama\Projects\RaporKodlari\Hafta10_RaporKodlari> npm init -y
Wrote to C:\Users\berka\OneDrive\Masaüstü\CENG File\BTU Ders Dosyaları\2023-2024 Bahar Dönemi\BLM0470 - NodeJS ile Web Programlama\Projects\RaporKodlari\Hafta10_RaporKodlari\package.json:

{
  "name": "hafta10_raporKodlari",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC"
}
```

Sonrasında ise uygulama boyunca kullanacağımız paketleri **npm install** komutu aracılığıyla yüklememiz gerekmektedir. Uygulamamızda kullanacağımız paketler **dotenv** ve **mongoose** paketleridir. **dotenv** paketi, çevre değişkenlerini kolayca projemize yüklememizi ve kullanmamızı sağlar. Bu sayede hassas verileri kodlarımıza sabitlemeden güvenli bir şekilde kullanabiliriz. **mongoose** paketi ise MongoDB veritabanı ile etkileşimde bulunmak için gerekli olan işlevleri sağlar. Veritabanıyla bağlantı kurmayı, veri eklemeyi, güncellemeyi, silmeyi ve sorgulamayı sağlar.



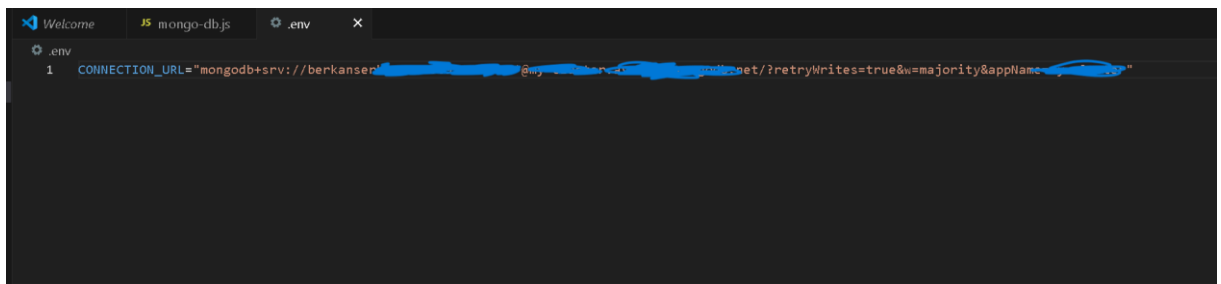
```
PS C:\Users\berka\OneDrive\Masaüstü\CENG File\BTU Ders Dosyaları\2023-2024 Bahar Dönemi\BLM0470 - NodeJS ile Web Programlama\Projects\RaporKodlari\Hafta10_RaporKodlari> npm i dotenv mongoose
added 13 packages, and audited 14 packages in 11s

1 package is looking for funding
  run 'npm fund' for details

found 0 vulnerabilities
```

Sonraki adımda, ana çalışma dosyamızı belirlemek için **mongo-db.js** adında bir dosya oluşturalım. Bu dosya, projemizdeki temel dosya olacak ve MongoDB ile ilgili işlemleri bu dosya üzerinden gerçekleştireceğiz.

Sonrasında **.env** adında bir dosya oluşturalım. Bu dosyada MongoDB ile bağlantı kurmak için **connection string'imizi** saklayacağız. Connection string, bir veritabanına erişmek için gereken bilgileri içeren bir metinsel ifadedir. Bu bilgiler genellikle sunucu adı, veritabanı adı, kullanıcı adı, parola gibi öğeleri içerir. Connection string, uygulamanın veritabanıyla iletişim kurmasını sağlar ve veritabanına bağlanma sürecini tanımlar.



```
.env
1 CONNECTION_URL="mongodb+srv://berkansen:berkansen@berkansen.mongodb.net/?retryWrites=true&w=majority&appName=berkansen"
```

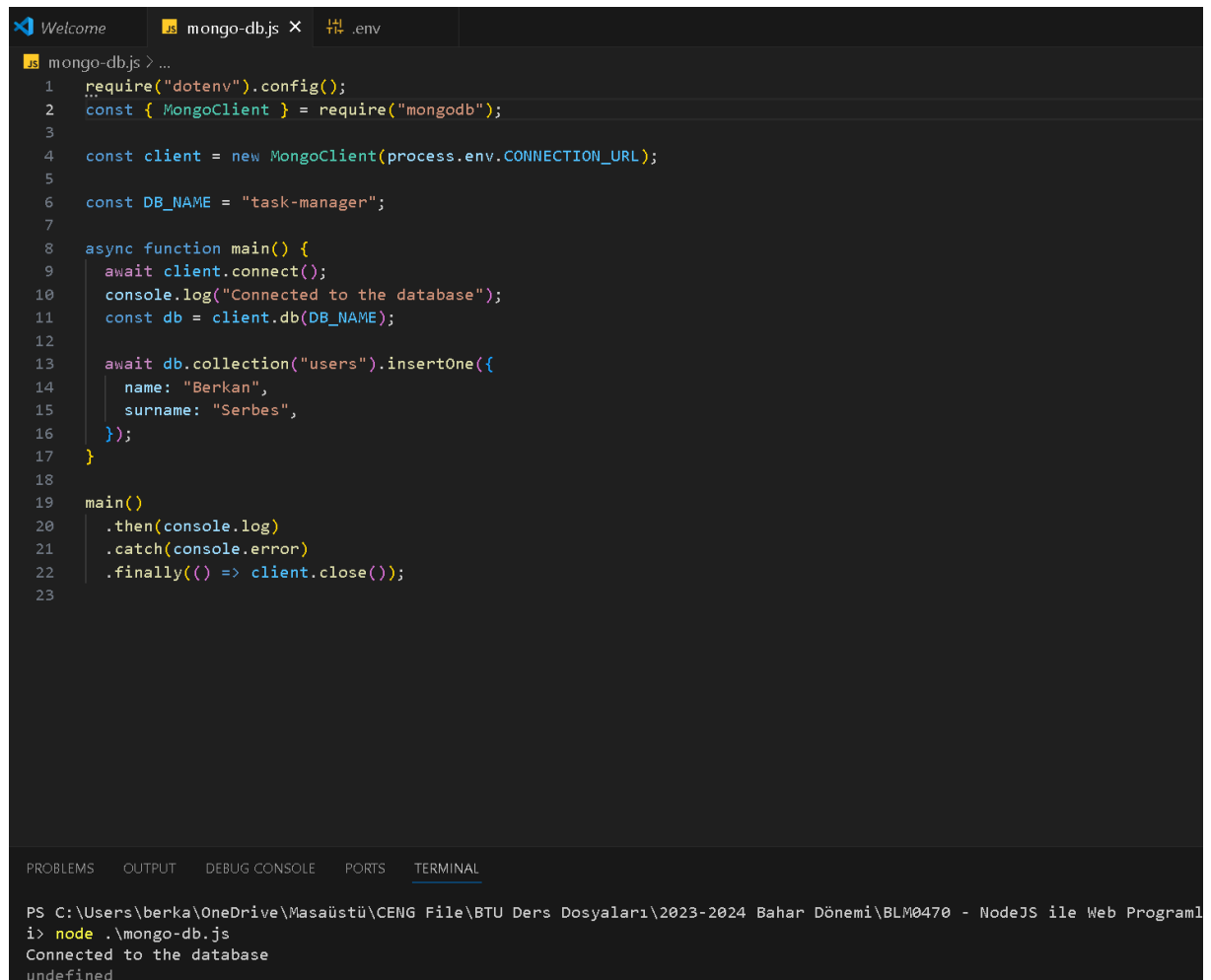

Aşağıdaki görselde yer alan kod parçasında, MongoDB veritabanına bağlanmak ve belirli bir veritabanı içindeki bir koleksiyona veri eklemek için kullanılır. İlk olarak, dotenv paketi kullanılarak .env dosyasından gelen bağlantı bilgileri alınır. Daha sonra, mongodb paketinden MongoClient modülü alınır. Bu modül, MongoDB veritabanına bağlanmak için gereklidir.

Sonraki adımda, **MongoClient** sınıfından bir örnek oluşturulur ve connect metodu kullanılarak MongoDB veritabanına bağlanılır. Bağlantı başarılı olduğunda, "**Connected to the database**" mesajı konsola yazdırılır.

DB_NAME sabiti aracılığıyla hedeflenen veritabanı adı belirlenir ve **client.db(DB_NAME)** yöntemiyle bu veritabanına erişim sağlanır.

insertOne metodu kullanılarak "**users**" koleksiyonuna yeni bir belge eklenir. Eklenen belge, adı "Berkan" ve soyadı "Serbes" olan bir kullanıcıyı temsil eder.

Kodun son kısmında, main fonksiyonu then ve catch bloklarıyla birlikte çağrılır. Bu, asenkron işlemin başarılı veya başarısız olması durumunda uygun bir geri çağırma fonksiyonunun çalıştırılmasını sağlar. Son olarak, finally bloğu kullanılarak **client.close()** metodu çağrılarak MongoDB bağlantısı kapatılır. Bu, uygulamanın veritabanıyla olan bağlantısını güvenli bir şekilde sonlandırır.

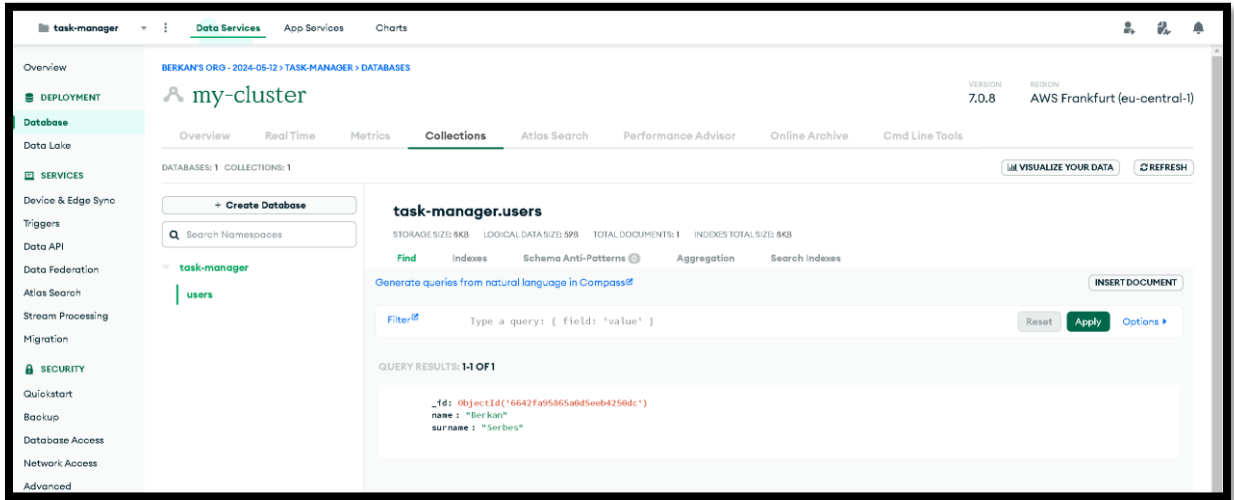


```
1 require("dotenv").config();
2 const { MongoClient } = require("mongodb");
3
4 const client = new MongoClient(process.env.CONNECTION_URL);
5
6 const DB_NAME = "task-manager";
7
8 async function main() {
9   await client.connect();
10  console.log("Connected to the database");
11  const db = client.db(DB_NAME);
12
13  await db.collection("users").insertOne({
14    name: "Berkan",
15    surname: "Serbes",
16  });
17 }
18
19 main()
20   .then(console.log)
21   .catch(console.error)
22   .finally(() => client.close());
23
```

PROBLEMS OUTPUT DEBUG CONSOLE PORTS TERMINAL

```
PS C:\Users\berka\OneDrive\Masaüstü\CENG File\BTU Ders Dosyaları\2023-2024 Bahar Dönemi\BLM0470 - NodeJS ile Web Programl
i> node .\mongo-db.js
Connected to the database
undefined
```

MongoDB Atlas üzerinden veritabanına baktığımızda eklediğimiz verinin başarılı bir şekilde veritabanına eklendiğini aşağıdaki görselden görmekteyiz.



Eğer aynı anda birden fazla veriyi koleksiyonumuza eklemek istersek o zaman **insertMany** fonksiyonunu kullanmamız gerekmektedir. Koleksiyona eklemek istediğimiz kullanıcı verilerini içeren bir dizi belgeyi insertMany yöntemiyle iletilir. Her bir veri ögesi, JSON formatında bir nesne olarak tanımlanır ve koleksiyona eklenir. Bu sayede, tek bir işlemde birden fazla kullanıcı verisini veritabanına eklemek mümkün olur. Örnekte, üç farklı kullanıcı verisi eklenmektedir: Linus Torvalds, Dan Abramov ve Ryan Dahl.

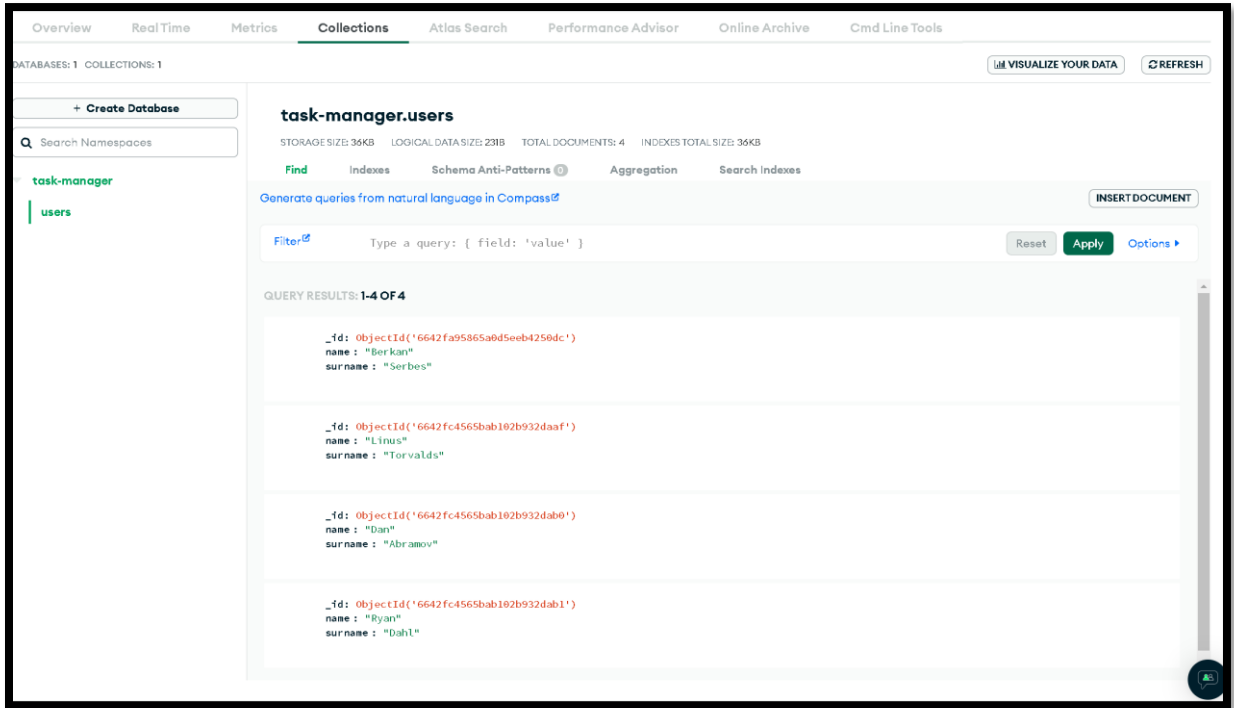
```
1 require("dotenv").config();
2 const { MongoClient } = require("mongodb");
3
4 const client = new MongoClient(process.env.CONNECTION_URL);
5
6 const DB_NAME = "task-manager";
7
8 async function main() {
9   await client.connect();
10  console.log("Connected to the database");
11  const db = client.db(DB_NAME);
12
13  await db.collection("users").insertMany([
14    {
15      name: "Linus",
16      surname: "Torvalds",
17    },
18    {
19      name: "Dan",
20      surname: "Abramov",
21    },
22    {
23      name: "Ryan",
24      surname: "Dahl",
25    },
26  ]);
27 }
28
29 main()
30   .then(console.log)
31   .catch(console.error)
32   .finally(() => client.close());
33
```

PS C:\Users\berka\OneDrive\Kasaüstü\CENG File\BTU Ders Dosyaları\2023-2024 Bahar Dönemi\BL\0470 - NodeJS ile Web Programlama\Projects\RaporKodları\Hafta10_RaporKodları> node .\mongo-db.js

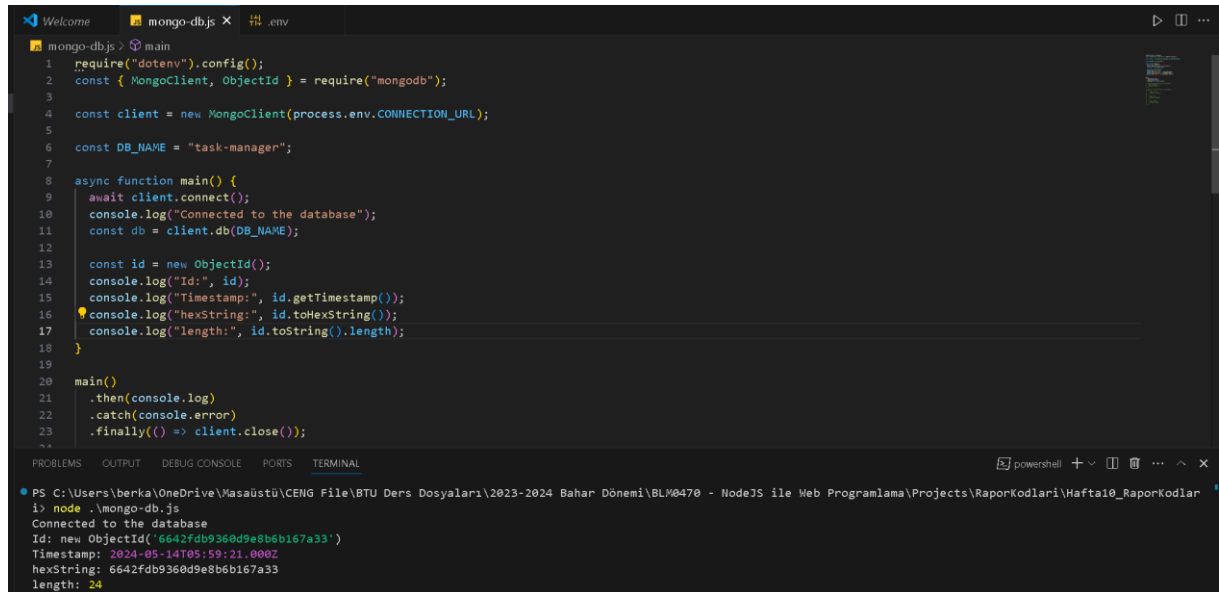
Connected to the database

undefined

Aşağıdaki görselde işlemimizin başarılı bir şekilde gerçekleştiğini görmekteyiz.



Aşağıdaki görselde bulunan kod parçasında, **ObjectId** sınıfı kullanılarak benzersiz kimlik numarası oluşturulmaktadır. MongoDB’de her belge birincil anahtar olarak bir kimlik alanına sahiptir ve bu kimlik alanı genellikle ‘_id’ olarak adlandırılır. ‘_id’ alanı, belgenin benzersiz bir şekilde tanımlanmasını sağlar ve veritabanında bir belgeyi tekil bir şekilde temsil eder. ObjectId sınıfı, MongoDB tarafından otomatik olarak oluşturulan benzersiz kimlikleri oluşturmak için kullanılır. Bu kimlikler genellikle **hexadecimal** formatında ve **12 byte** uzunluğundadır. **ObjectId oluştururken kullanılan zaman damgası, bilgisayarın saatine dayalı olduğu için her ObjectId nesnesi benzersizdir.**

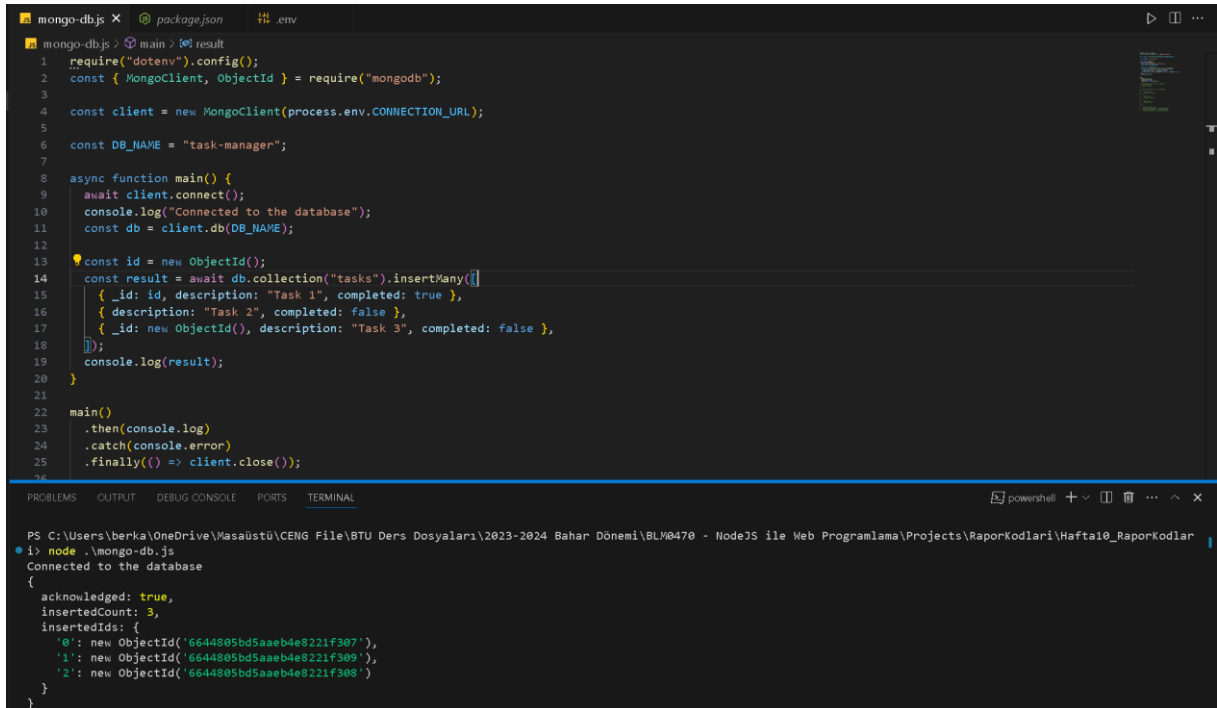


Bu kod bloğunda, ObjectId sınıfı kullanılarak yeni bir kimlik oluşturulmakta ve bu kimlik çeşitli özellikleriyle birlikte konsola yazdırılmaktadır. Bu işlem, MongoDB'de yeni bir belge

oluştururken veya mevcut bir belgeyi güncellerken benzersiz kimliklerin nasıl oluşturulduğunu anlamak için yapılır. Ayrıca, oluşturulan kimliklerin zaman damgası ve benzersizlik özelliklerini test etmek için kullanılabilir.

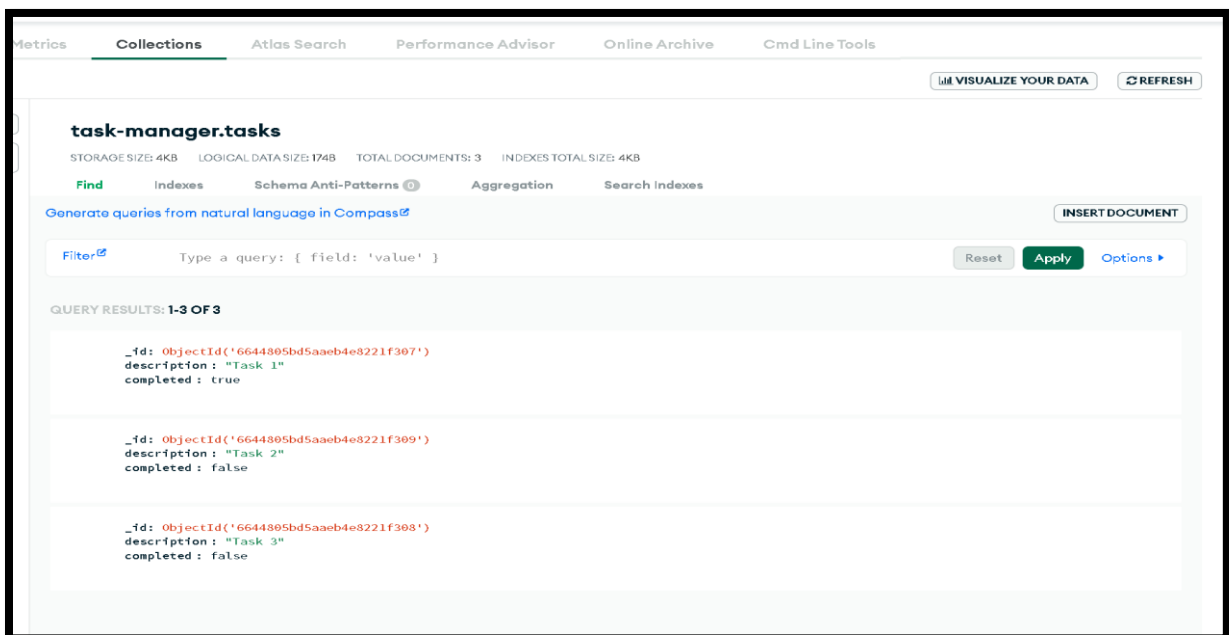
Özellikle ‘_id’ alanı belirtilmediğinde, MongoDB bir belge eklerken otomatik olarak bir _id oluşturur. Ancak, bazı durumlarda, özellikle belirli bir _id değeri atamak gerektiğinde, **new ObjectId()** ifadesiyle manuel olarak bir _id oluşturulur ve belgeye atanır.

Bu örnekte, insertMany() yöntemi kullanılarak bir dizi belge eklenirken, ilk belgeye ve üçüncü belgeye özel bir _id atanırken, ikinci belgeye otomatik olarak oluşturulan _id değerleri atanır. Bu şekilde, belirli bir belgeyi benzersiz bir şekilde tanımlamak için ObjectId kullanılmış olur.



```
1 require("dotenv").config();
2 const { MongoClient, ObjectId } = require("mongodb");
3
4 const client = new MongoClient(process.env.CONNECTION_URL);
5
6 const DB_NAME = "task-manager";
7
8 async function main() {
9   await client.connect();
10  console.log("Connected to the database");
11  const db = client.db(DB_NAME);
12
13  const id = new ObjectId();
14  const result = await db.collection("tasks").insertMany([
15    { _id: id, description: "Task 1", completed: true },
16    { description: "Task 2", completed: false },
17    { _id: new ObjectId(), description: "Task 3", completed: false },
18  ]);
19  console.log(result);
20 }
21
22 main()
23   .then(console.log)
24   .catch(console.error)
25   .finally(() => client.close());
```

```
PS C:\Users\berka\OneDrive\Masaüstü\CENG File\BTU Ders Dosyaları\2023-2024 Bahar Dönemi\BLM0470 - NodeJS ile Web Programlama\Projects\RaporKodlari\Hafta10_RaporKodlar
i> node .\mongo-db.js
Connected to the database
{
  acknowledged: true,
  insertedCount: 3,
  insertedIds: {
    '0': new ObjectId('6644805bd5aaeb4e8221f307'),
    '1': new ObjectId('6644805bd5aaeb4e8221f309'),
    '2': new ObjectId('6644805bd5aaeb4e8221f308')
  }
}
```



KAYNAKÇA

<https://www.spiceworks.com/tech/cloud/articles/what-is-mongodb/>

<https://chatgpt.com/>