

Node Js Ödevi

10th Week

Farhan Ahmad

20360859096

Geçen hafta **Tarayıcı Üzerinden Express API'leri Oluşturma ve Kullanma Rehberi** gibi konuları incelemiştik. Bu hafta ise **MongoDB ve Promiseler** gibi konuları ele alacağız.

MongoDB ve Promiselerle Başlamak:

Görev yöneticisi projesine başlayacağız. Kimlik doğrulama sistemi gerekmektedir.

MongoDB Kullanımı:

Çoğu Node geliştiricisi MongoDB kullanıyor. Siz de alternatif olarak MySQL veya PostgreSQL kullanabilirsiniz. MongoDB, diğerlerinin aksine NoSQL bir veri tabanıdır. SQL veri tabanlarında tablolar bulunurken, NoSQL veri tabanlarında tabloların yerini koleksiyonlar alır. SQL'deki tablolarda girişler için satırlar veya kayıtlar, farklı veri parçaları için ise sütunlar bulunurken, NoSQL'de belgeler satırların yerine, alanlar ise sütunların yerine geçer.

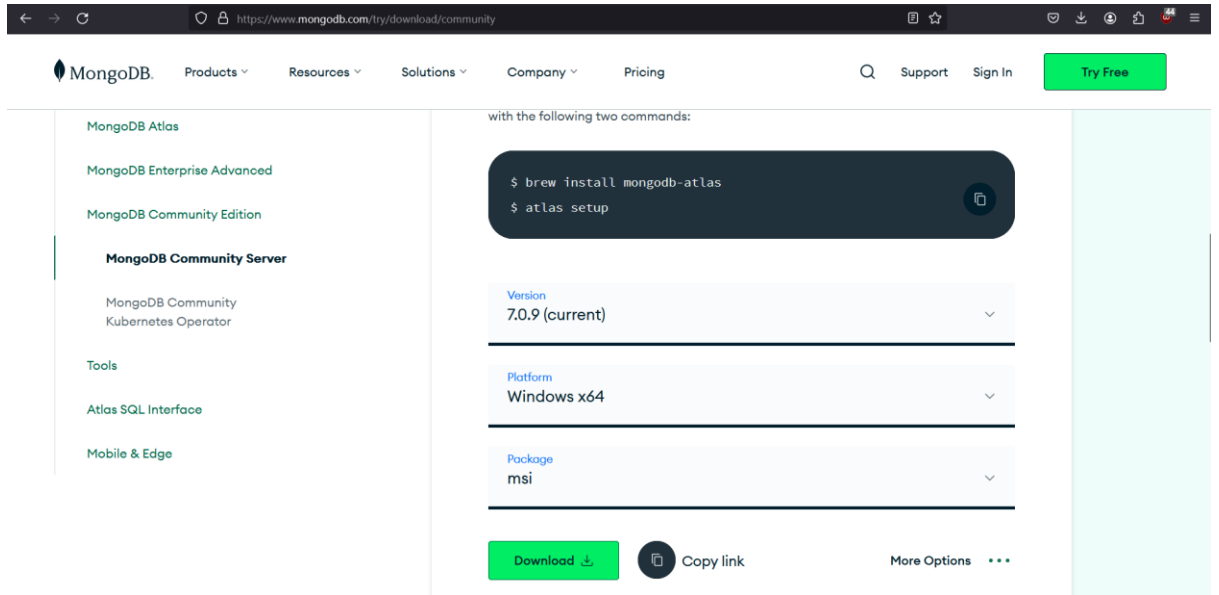
Kurulum İşlemleri:

- **Mac ve Linux İçin:**
 - MongoDB web sitesine gidin ve sunucu sekmesini seçin.

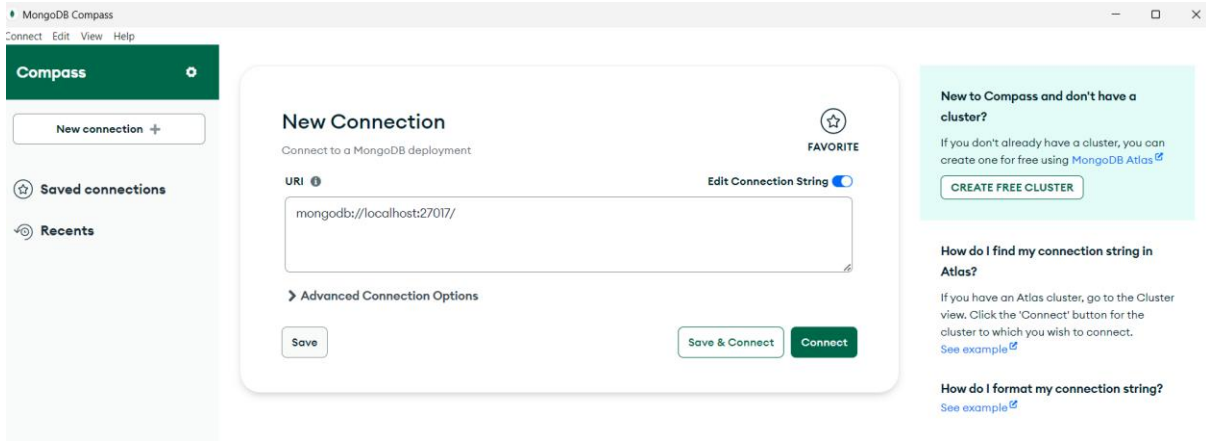
- Topluluk sürümünü indirin ve arşivi çift tıklayarak açın.
- Bin dizinine gidin ve içindeki dosyaları bir klasöre taşıyın.
- Sabit sürücünüzde veri saklamak için "mongodb-data" adında bir klasör oluşturun.
- Terminalde **~/mongodb/bin/mongod --dbpath=~/mongodb-data** komutunu çalıştırarak sunucuyu başlatın.

- **Windows için:**

- MongoDB web sitesine gidin ve sunucu sekmesini seçin.



- Topluluk sürümünü indirin ve arşivi çift tıklayarak açın.
- Bin dizinine gidin ve içindeki dosyaları bir klasöre taşıyın.
- Sabit sürücünüzde veri saklamak için "mongodb-data" adında bir klasör oluşturun.
- Terminalde **~/mongodb/bin/mongod --dbpath=~/mongodb-data** komutunu çalıştırarak sunucuyu başlatın.



GUI Görüntüleyici Kurulumu:

Robo 3T gibi bir veritabanı yönetim aracı kurabilirsiniz. İndirdikten sonra bağlantı oluşturup MongoDB'ye erişebilirsiniz.

Belge Ekleme ve Bağlantı:

MongoDB'ye bağlanmak için native sürücüyü kullanabilirsiniz. Daha fazla bilgi için MongoDB belgelerine bakabilirsiniz.

MongoDB CRUD İşlemleri:

```
const { MongoClient } = require('mongodb');

// MongoDB'ye bağlanılacak URL ve veritabanı adı
const connectionURL = 'mongodb://127.0.0.1:27017';
const dbName = 'task-manager';

// MongoClient ile MongoDB'ye bağlanma
const client = new MongoClient(connectionURL);

(async () => {
  try {
```

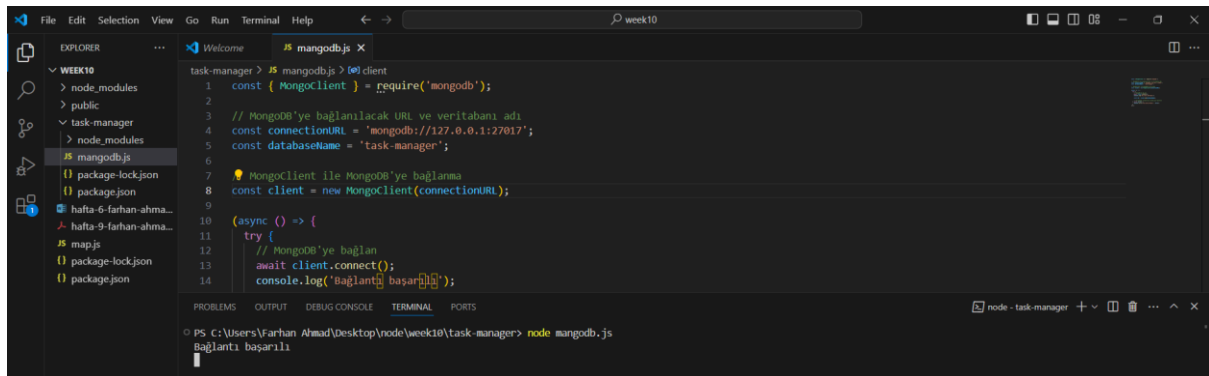
```

// MongoDB'ye bağlan
await client.connect();
console.log('Bağlantı başarılı');

const db = client.db(databaseName);

// Veritabanı işlemlerini burada gerçekleştir
} catch (error) {
  console.error('Bağlantı hatası:', error);
}
})();

```



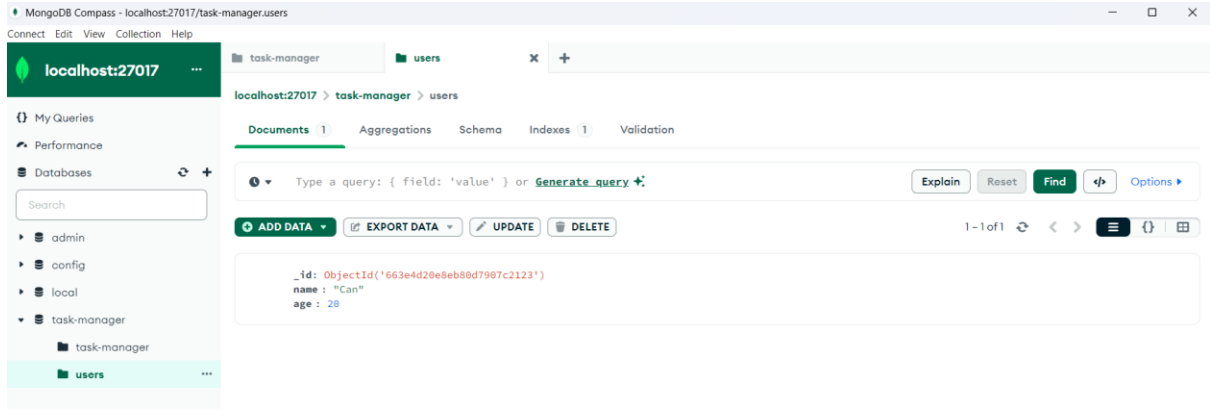
Bu kod parçası, MongoDB'ye bağlanmayı sağlar. Bağlantı başarılı olduğunda Başarılı bir şekilde bağlandı yazısı konsola yazdırılır. Bağlantıyı test etmek için terminalde node mongodb.js komutunu çalıştırabilirsiniz. Bağlantıyı sonlandırmak için CTRL + C tuşlarına basabilirsiniz.

Şimdi, belge eklemeyi deneyelim:

```

db.collection('users').insertOne({
  name: 'Can',
  age: 20
}, (error, result) => {
  if (error) {
    return console.log('Kullanıcı eklenemedi');
  }
  console.log(result.ops);
});

```



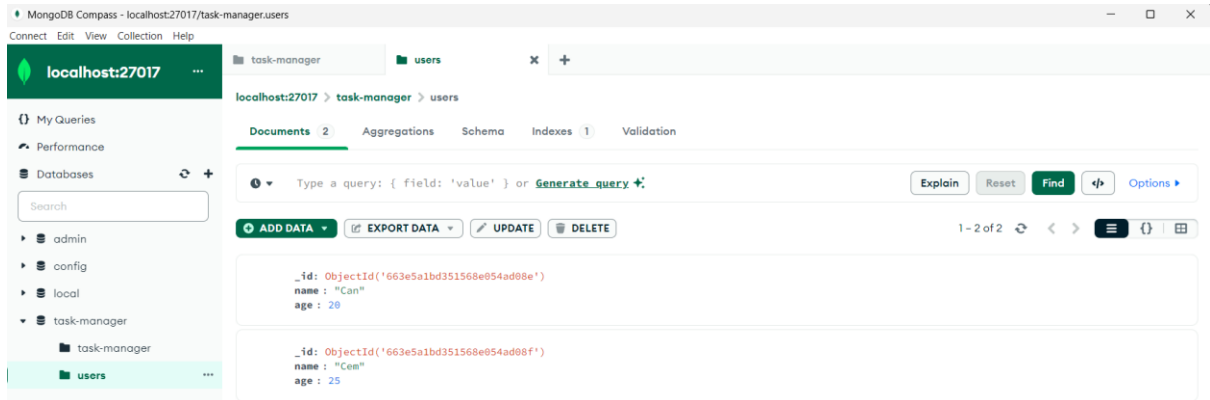
Bu kod parçası, users koleksiyonuna yeni bir belge ekler. Ekleme işlemi başarılı olduğunda, eklenen belgenin detaylarını konsola yazdırır.

Bu işlemleri gerçekleştirdikten sonra, dosyayı kaydedin ve terminalde node mongodb.js komutunu çalıştırarak kodu çalıştırın. Sonuçları Robo 3T gibi bir araçla da kontrol edebilirsiniz. Bağlantıyı oluşturduktan sonra, belgelerin eklenip eklenmediğini görüntüleyebiliriz.

Belge Ekleme (Inserting Documents):

Veritabanına belge eklemek için insertOne ve insertMany gibi yöntemler kullanılır. insertOne, verilen bir belgeyi eklerken, insertMany birden fazla belgeyi aynı anda ekler. Örneğin, aşağıdaki kod bir kullanıcı belgesi ekler:

```
db.collection('users').insertMany([
  {
    name: 'Can',
    age: 20
  },
  {
    name: 'Cem',
    age: 25
  }
])
```



Object ID:

Object ID, MongoDB'de belirli bir belgeyi benzersiz bir şekilde tanımlamak için kullanılan bir türdür. GUID (Global Unique Identifier) olarak da bilinir. Bu tanımlayıcılar, belirli bir belgeyi diğerlerinden ayırt etmek için kullanılır ve genellikle `_id` alanında saklanır.

MongoDB'de `ObjectId` sınıfı, bu benzersiz tanımlayıcıları oluşturmak ve işlemek için kullanılır. `ObjectId` oluşturmak için `new ObjectId()` kullanılır ve bu şekilde her seferinde farklı bir benzersiz kimlik elde edilir. Örneğin:

```
const { MongoClient, ObjectId } = require('mongodb');
const connectionURL = 'mongodb://127.0.0.1:27017';
const databaseName = 'task-manager';
const client = new MongoClient(connectionURL);

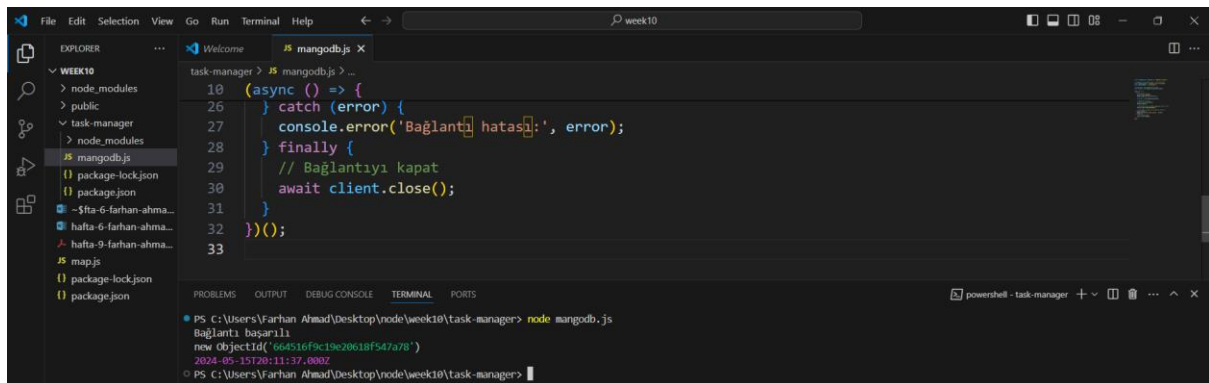
(async () => {
  try {
    await client.connect();
    console.log('Bağlantı başarılı');

    const db = client.db(databaseName);
    const id = new ObjectId();
    console.log(id);
  } catch (error) {
    console.error(error);
  }
})();
```

```
    console.log(id.getTimestamp());

} catch (error) {
    console.error('Bağlantı hatası:', error);
} finally {

    await client.close();
}
})();
```



The screenshot shows the Visual Studio Code interface. The Explorer sidebar on the left displays a project structure for 'WEEK10' with folders 'node_modules', 'public', 'task-manager', and 'node_modules'. Inside 'task-manager', there is a file 'mangodb.js' and several 'package.json' files. The main editor window shows the content of 'mangodb.js', which is a JavaScript file with the following code:

```
10 (async () => {
26 } catch (error) {
27     console.error('Bağlantı hatası:', error);
28 } finally {
29     // Bağlantıyı kapat
30     await client.close();
31 }
32 })();
33
```

Below the editor, the 'TERMINAL' panel shows the output of running 'node mangodb.js' in a PowerShell terminal. The output indicates a successful connection:

```
PS C:\Users\Farhan Ahmad\Desktop\node\week10\task-manager> node mangodb.js
Bağlantı başarılı
new ObjectId('664516f9c19e20618f547a78')
2024-05-15T20:11:37.000Z
PS C:\Users\Farhan Ahmad\Desktop\node\week10\task-manager>
```