Kadir Çapkın 21360859023

NodeJS ile Web Programlama Raporu Hafta-14

Bugün derste postman veri işlemlerinde postman uygulamasını kullandık.

İlk olarak dosyamıza girdik, nodemon ve express modülünü ekledik.

Dosyalarımızı içerisinde src adında klasör oluşturduk ve içerisine index.js dosyasını ekledik.

Port numarasını belirlemek için aşağıdaki kod satırını kullandık:

```
const port = process.env.PORT || 3000
```

Bu satırda eğer env dosyasında port için belirli bir değer varsa portun değeri o olacak, aksi halde port değeri 3000 olacaktır. Bunun sebebi de bulut üzerinde çalışırken her zaman istenilen port kullanılamayabilir çünkü bu port başka bir uygulama tarafından kullanılabilir.

app.listen() komutu ile port girerek çalıştırdık. Çalıştığına dair de konsola mesaj döndürdük.

```
const express = require("express")

const app = express()
const port = process.env.PORT || 3000

app.listen(port, () => {
  console.log("Sunucu çalışıyor, port numarası: ", port);
});
```

Package.json dosyası içerisindeki script start ve dev adında iki komut yazdık.

```
▷ Debug
"scripts": {
  "start": "node src/index.js",
  "dev": "nodemon src/index.js"
},
```

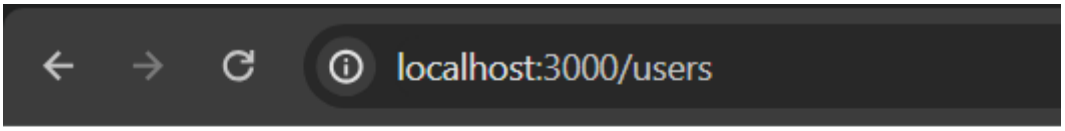
Bu komutlar kodumuzu terminalde çalıştırma işlemlerini tuttu.

Mesela npm run dev dediğimizde nodemon olacak şekilde kodumuz çalıştı:

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

> taskmanagerapp@1.0.0 dev
> nodemon src/index.js

[nodemon] 3.1.1
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,cjs,json
[nodemon] starting `node src/index.js`
Sunucu çalışıyor, port numarası:  3000
```

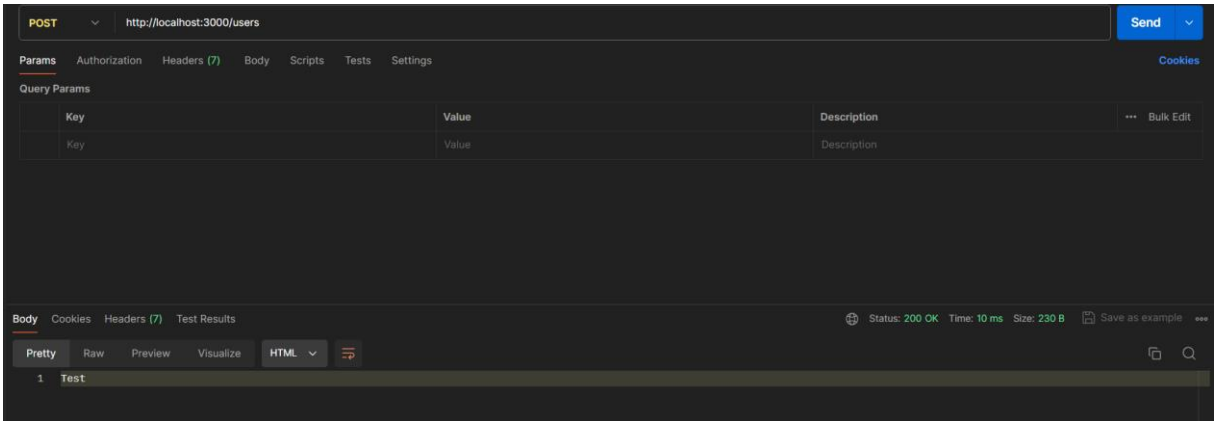
Daha sonra app.post() kullanrak test işlemi sağladık. Bu test işleminde kodumuz /user yolu altında ren.send() komutu ile mesajı döndürmekte.

```
const express = require("express");

const app = express();
const port = process.env.PORT || 3000;

app.listen(port, () => {
  console.log("Sunucu çalışıyor, port numarası: ", port);
});

app.post("/users", (req, res) => {
  res.send("Test");
});
```



Cannot GET /users

Kod çalıştı ve localhost:3000/users bağlantısına gittik fakat hata aldık. Bu hatanın sebebi de bizim kodumuzu post ile göndermemiz. Çünkü bu kısımda get olarak algılanıyor ve kod hata veriyor.

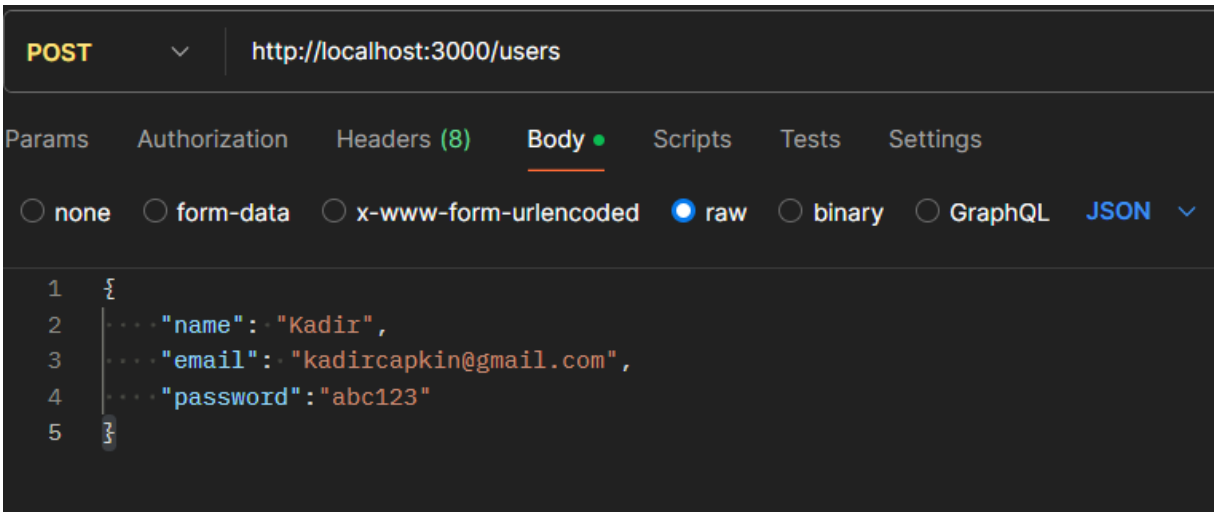


Postman içerisinde post ile yolumuzu yazdığımızda çıktımızı başarılı bir şekilde aldık.

app.use() komutu kullandık. Bu komut içerisine bir parametre girdik. Bu parametre (express.json()) bize gönderilen json formatındaki verinin body kısmından alınarak ulaşılmasını sağladı.

```
app.use(express.json());
```

Postman uygulamasında denemek adında post olacak şekilde yolumuzu girdik. Ardından body kısmında row seçeneğini tıkladık ve json formatını ayarladık. Name, email ve password bilgileri olacak şekilde veri girip gönderme denemesi yaptık.

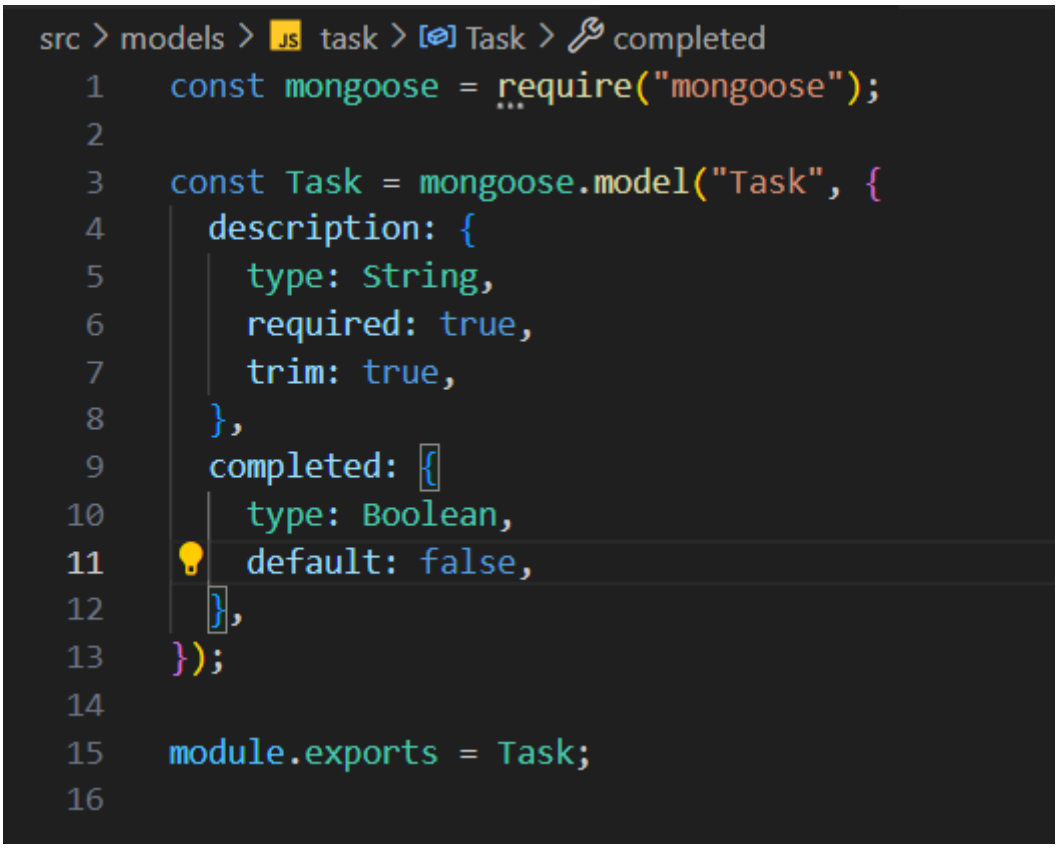


Src klasörü altında db ve models klasörleri açtık. Models klasörü içerisine task ve user adında kullanıcı ve görevleri tutacağımız dosyalar açtık. Db klasörü altına mongodb.js ve mongoose.js dosyasını taşıdık.

Mongoose.js dosyasının içerisinde daha önceden kullanıcı ve görevlere ait kod blokları bulunmaktaydı. Bu blokları ve modülleri user ve task dosyalarımıza taşıdık.

User dosyası:

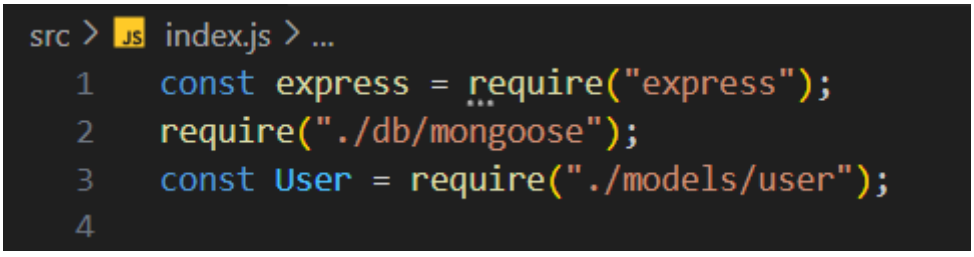
Task dosyası:



Mongoose.js:

```
const mongoose = require("mongoose");
mongoose.connect(
  "mongodb+srv://kadircapkinn:<password>@cluster0.pguczr6.mongodb.net/?retryWrites=true&w=majority&appName=Cluster0&dbName=mydb"
);
```

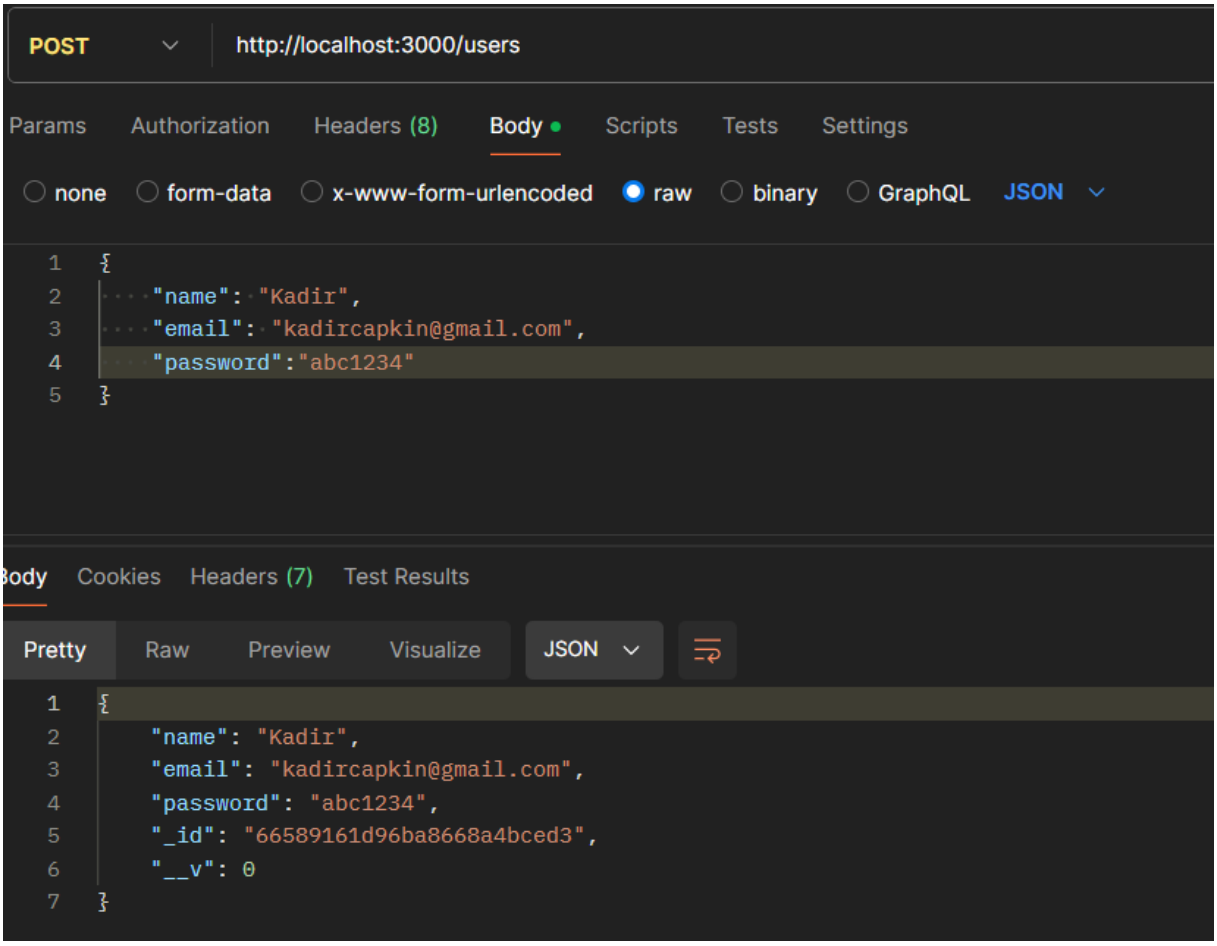
İndex.js dosyasında mongoose ve user dosyasını ekledik.



Artı olarak /users yolu içerisinde user adında yeni bir kullanıcı oluşturarak veri alma işlemini gerçekleştirdik.

Başarılı olma durumunda durum kodu 400 olurken başarısız olma durumunda da 201 oldu.

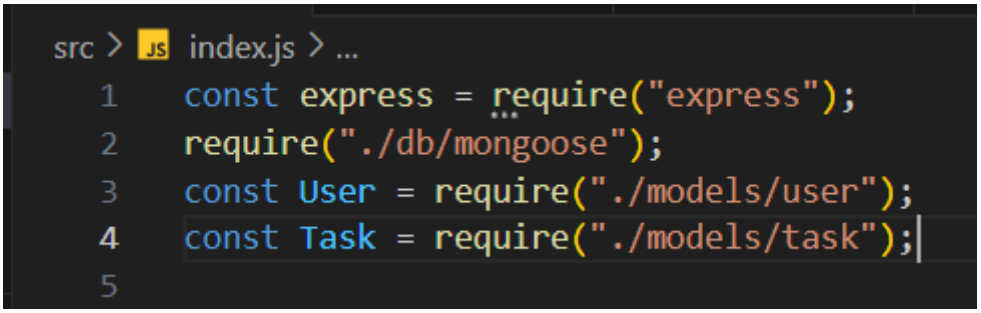
Daha sonra postman üzerinden verilerimizi tekrardan gönderdik. Fakat şifrem 6 haneli olduğundan hata aldım. Bunun sebebi de user oluştururken password kısmında belirlediğim koşullardandı. Bu yüzden şifremi değiştirdim ve tekrar gönderdim.



Mongodb içerinde de verim başarılı bir şekilde eklendi.

İndex.js dosyasında task adında da işlem oluşturduk. Task adındaki değişkende verileri tutup gönderme işlemi sağladık. Bunu da /tasks yolu altında yaptık.

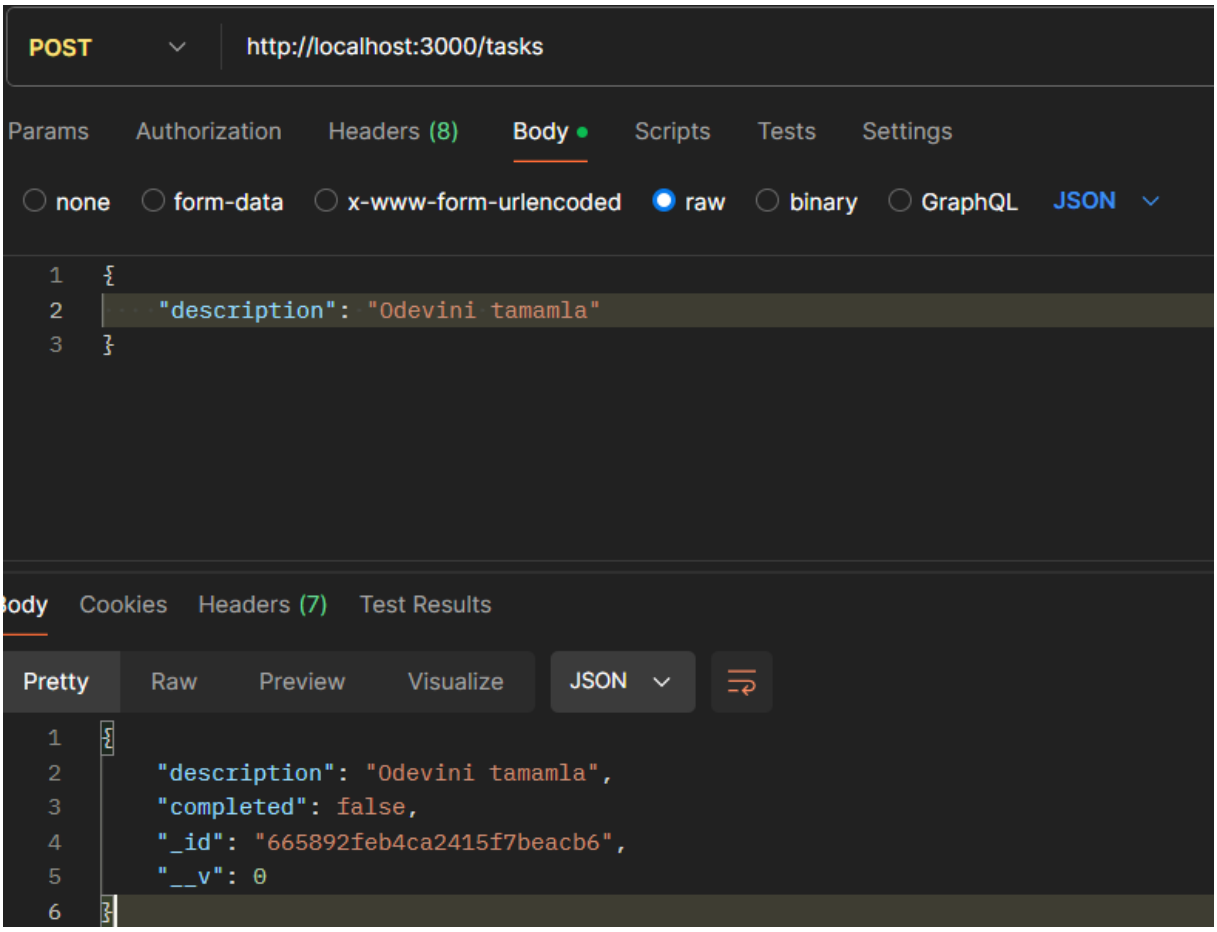
Ondan önce task dosyamıza require ile erişim sağladık:



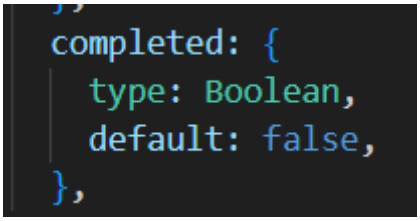
Task bloğu da user bloğuna benzer şekilde oldu:

Yine işlemin başarılı bir şekilde kayıt olması durumunda durum kodunu 201 e ayarlarken hata olma durumunda 400 e ayarladık.

Postman üzerinden görev ismi girerek bir veri gönderme işlemi gerçekleştirdik.



Completed kısmı default olarak false girdi çünkü task kısmında completed ifadesinin girilmesini zorunlu kılmadık ve girilmeme durumunda da default olarak false ifadesi girilmesine dair kod yazdık. O kod da şudur:



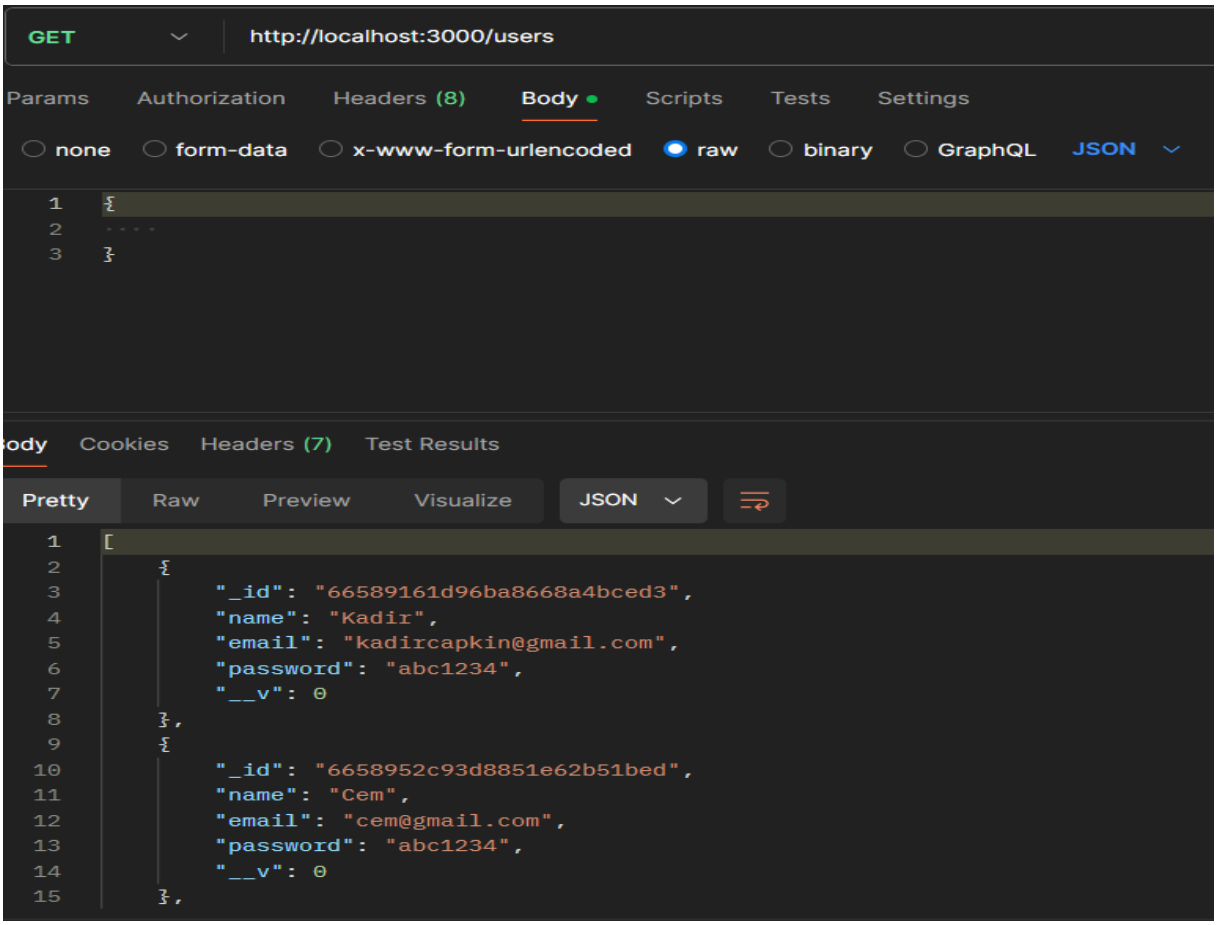
Mongodb içerisine de verimiz başarılı bir şekilde yüklendi.

Daha sonra users için get isteği oluşturduk. Bu istek ile veri çekme işlemini gerçekleştirmeyi amaç edindik.

```
app.get("/users", (req, res) => {
  User.find({})
    .then((users) => {
      res.send(users);
    })
    .catch((e) => {
      res.status(400).send(e);
    });
});
```

Bu kod ile beraber /users yolu ile postman üzerinden get adı altında gönderdiğimiz isteklerde mongodb veri seti içerisindeki users altındaki verileri getirme işlemi sağlanır. User.find() komutu içerisinde herhangi bir koşul girmediğimizden tüm kullanıcıları döndürür. İşlem istenildiği gibi başarılı bir şekilde gerçekleşmez ise 400 durum kodu ile hata döndürülür.

Bu işlem için user altına 2 veri daha gönderip postman üzerinden çekmeyi denedim ve şu sonucu aldım.



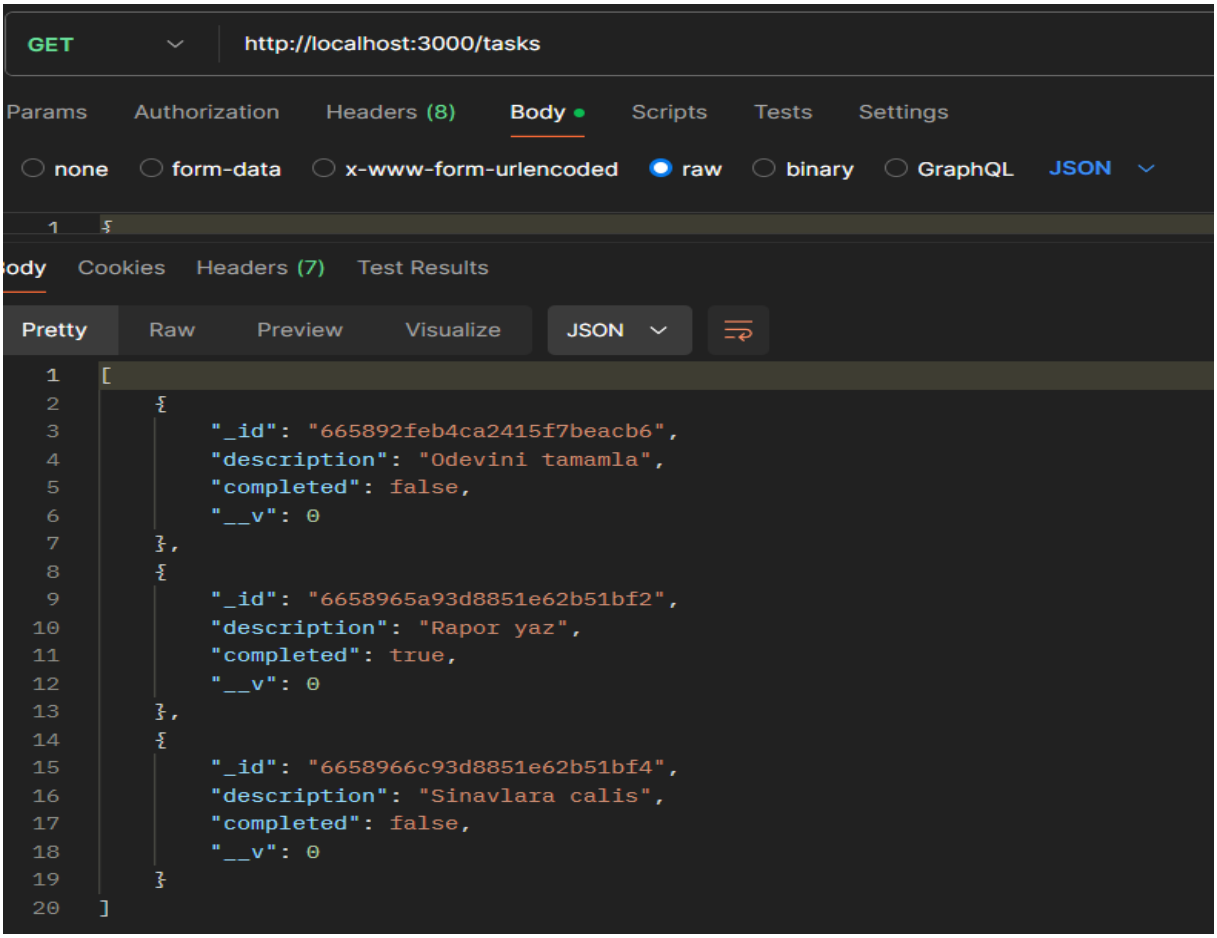
İstenildiği gibi veri çekme işlemi başarılı bir şekilde tamamlanıyordu.

Daha sonra aynı işlemi tasks için de denedik.

```
app.get("/tasks", (req, res) => {
  Task.find({})
    .then((tasks) => {
      res.send(tasks);
    })
    .catch((e) => {
      res.status(400).send(e);
    });
});
```

Bu kod ile beraber /tasks yolu ile postman üzerinden get adı altında gönderdiğimiz isteklerde mongodb veri seti içerisindeki tasks altındaki verileri getirme işlemi sağlanır. User.find() komutu içerisinde herhangi bir koşul girmediğimizden tüm tasklari döndürür. İşlem istenildiği gibi başarılı bir şekilde gerçekleşmez ise 400 durum kodu ile hata döndürülür.

Bu işlemi denemek için de birkaç veri gönderip çekme işlemine baktım.

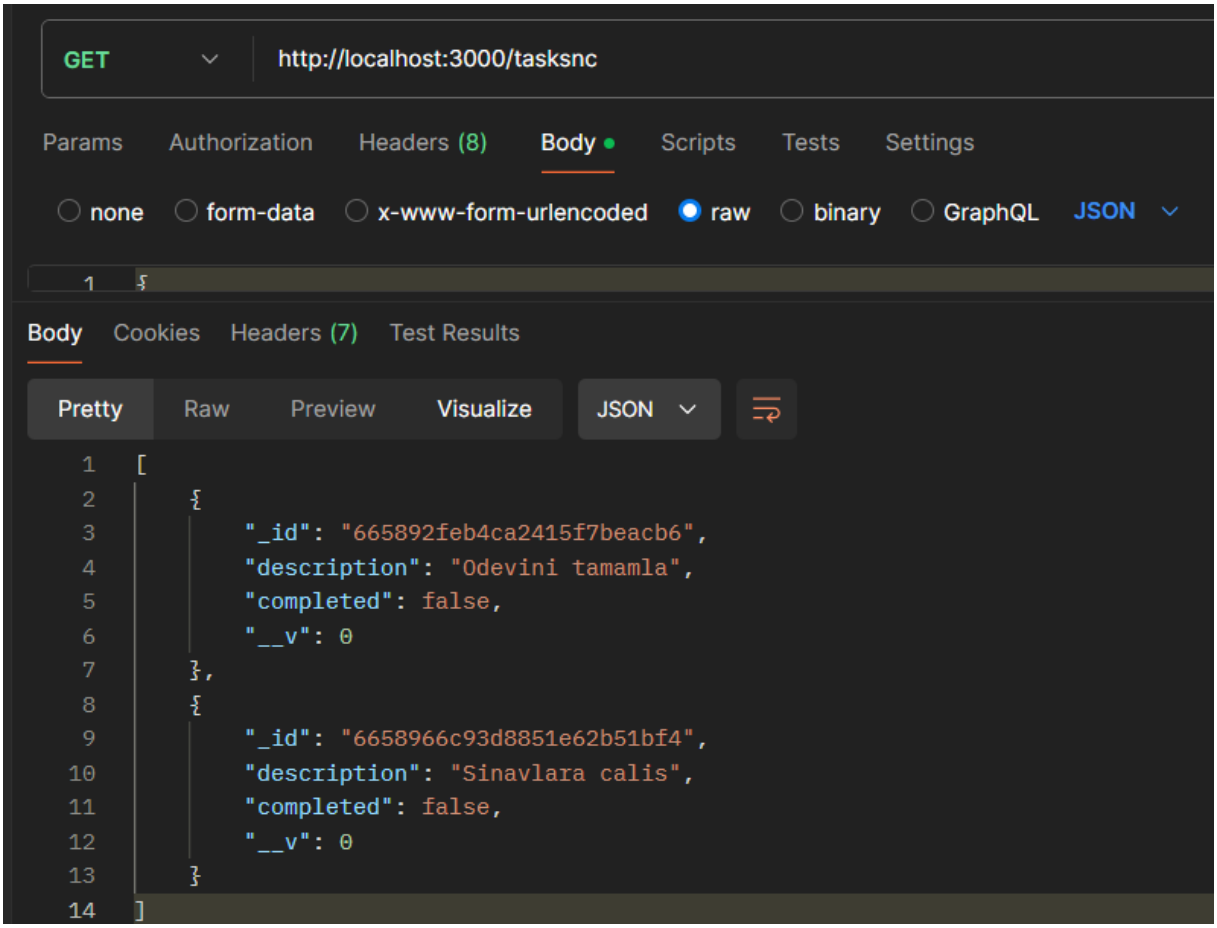


Daha sonra koşul sağlayarak veri çekmeyi gerçekleştirdik. Bu koşulumuzda tamamlanmayan görevleri çekmek oldu.

/usersnc yolu ile bir blok oluşturduk. Bu kod bloğunda task.find() komutu içerisinde competed:false girerek tamamlanmayan görevleri alıp yazdırmasını sağladık.

```
47  app.get("/tasksnc", (req, res) => {
48    Task.find({ completed: false })
49      .then((tasks) => {
50        res.send(tasks);
51      })
52      .catch((e) => {
53        res.status(400).send(e);
54      });
55  });
```

Postman de deneme yaptık.



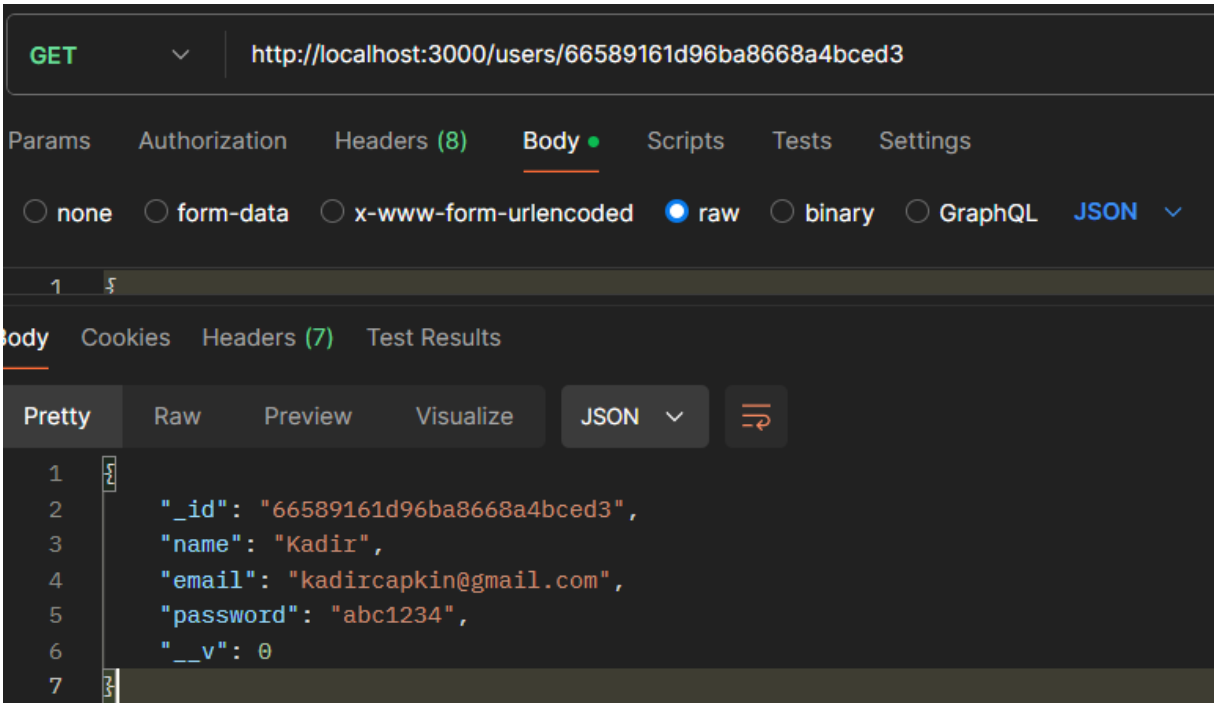
/tasks ile çektiğimiz verilerden farklı olarak yalnızca 2 veriyi yazdırdı. Rapor yaz adındaki veriyi yazdırmadı çünkü o görev tamamlanmış bir görevdi. Yani başarılı bir şekilde çalıştı.

Son olarak da bilinen id numarasına göre veri çekme işlemi sağladık. Bu users için de tasks için de sağlandı kod olarak şu şekilde oldu:

```
app.get("/users/:id", (req, res) => {
  const _id = req.params.id;
  User.findById({ _id })
    .then((user) => {
      if (!user) {
        return res.status(404).send();
      }
      res.send(user);
    })
    .catch((e) => {
      res.status(400).send(e);
    });
});
```

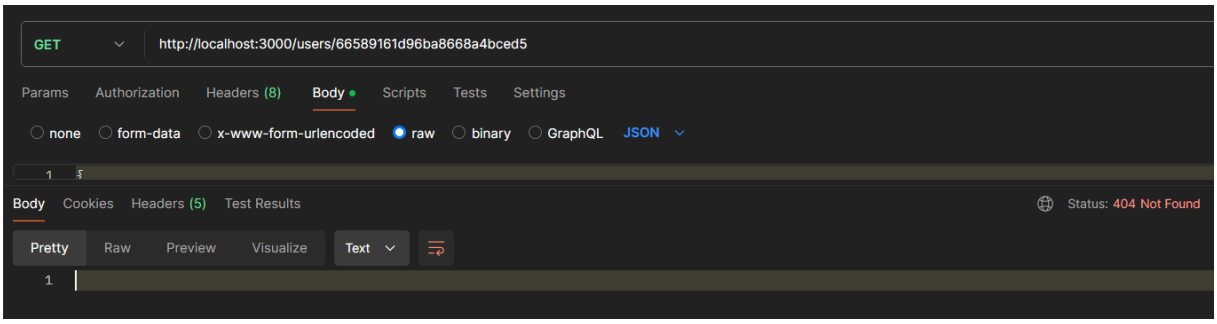
Bu kod ile /users/id ile yaptığımız aratmalarda id yazan kısma bilinen kullanıcı id numarası yazmamız gerekmekte. Kod içerisinde girdiğimiz id alınıp _id adındaki bir değişkene atanır. Daha sonra findByID() komutu ile girilen id veri setinde aranılır. Bulunma durumunda verimiz başarılı şekilde bize ulaşır fakat bulunmazsa yani öyle bir kullanıcı yoksa 404 durum kodu ile hata verir. Blok içerisinde herhangi bir hata olursa da 400 durum hata kodu ile döner.

Bu ihtimalleri deneyelim.



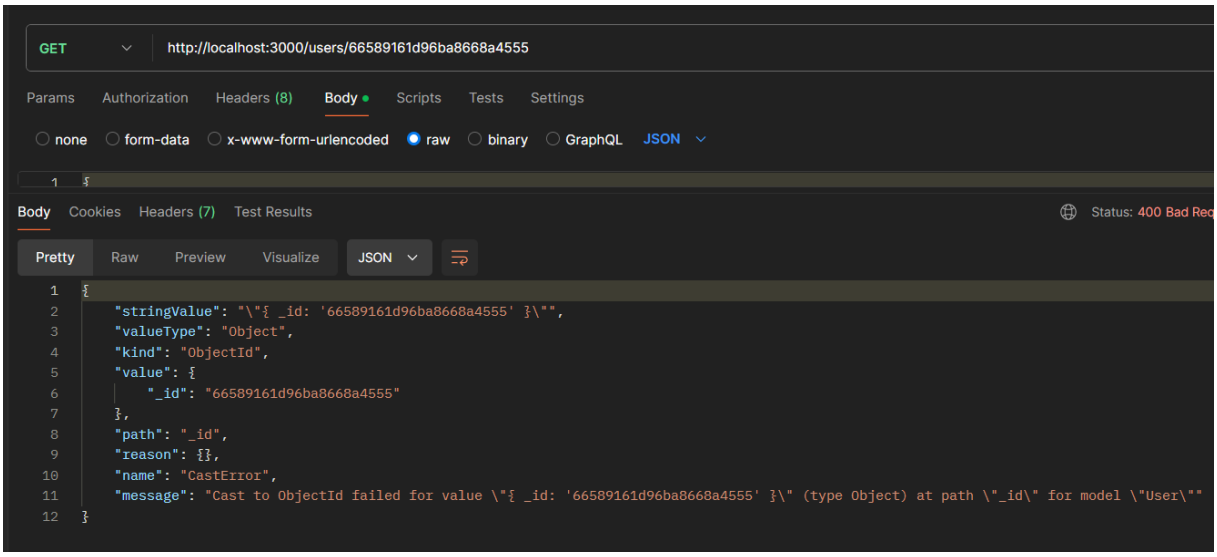
İlk olarak doğru id numarası ile denedik ve başarılı bir şekilde sonuç aldık. Daha sonra yanlış yani olmayan bir id ile giriş yaptık.

Daha sonra girdiğimiz id değerinin son rakamını değiştirerek giriş yaptım.



Burada 404 hatası verdi. Çünkü girilen id değeri bulunmamaktaydı.

Son olarak da id boyutundan daha kısa bir değer girdirip denedim.



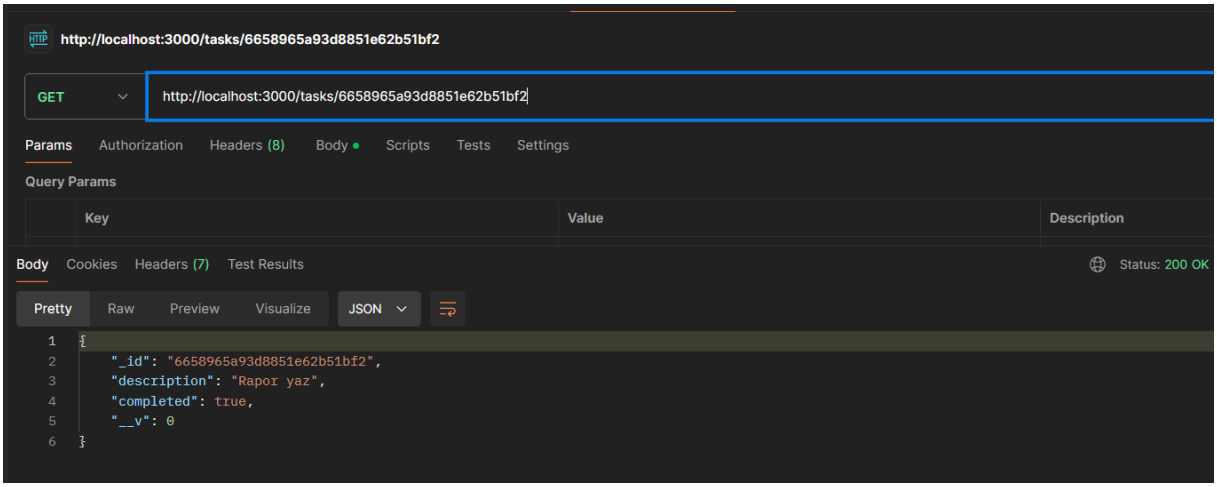
Ve objectID hatası aldık. Bu da 400 hata durum koduna girdi.

Bu işlemleri tasks için de tekrarladım.

Aşağıda bulunan kod ile /tasks/id ile yaptığımız aratmalarda id yazan kısma bilinen task id yazmamız gerekmekte. Kod içerisinde girdiğimiz id alınıp _id adındaki bir değişkene atanır. Daha sonra findByID() komutu ile girilen id veri setinde aranılır. Bulunma durumunda verimiz başarılı şekilde bize ulaşır fakat bulunmazsa yani öyle bir tasks yoksa 404 durum kodu ile hata verir. Blok içerisinde herhangi bir hata olursa da 400 durum hata kodu ile döner.

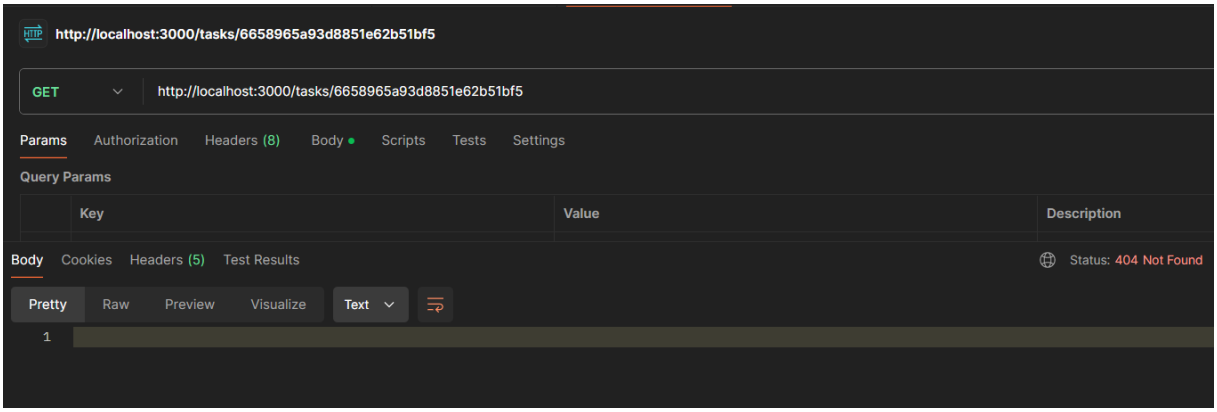
```
app.get("/tasks/:id", (req, res) => {
  const _id = req.params.id;
  Task.findById({ _id })
    .then((task) => {
      if (!task) {
        return res.status(404).send();
      }
      res.send(task);
    })
    .catch((e) => {
      res.status(400).send(e);
    });
});
```

Postman içerisinde bu işlemi de denedik.



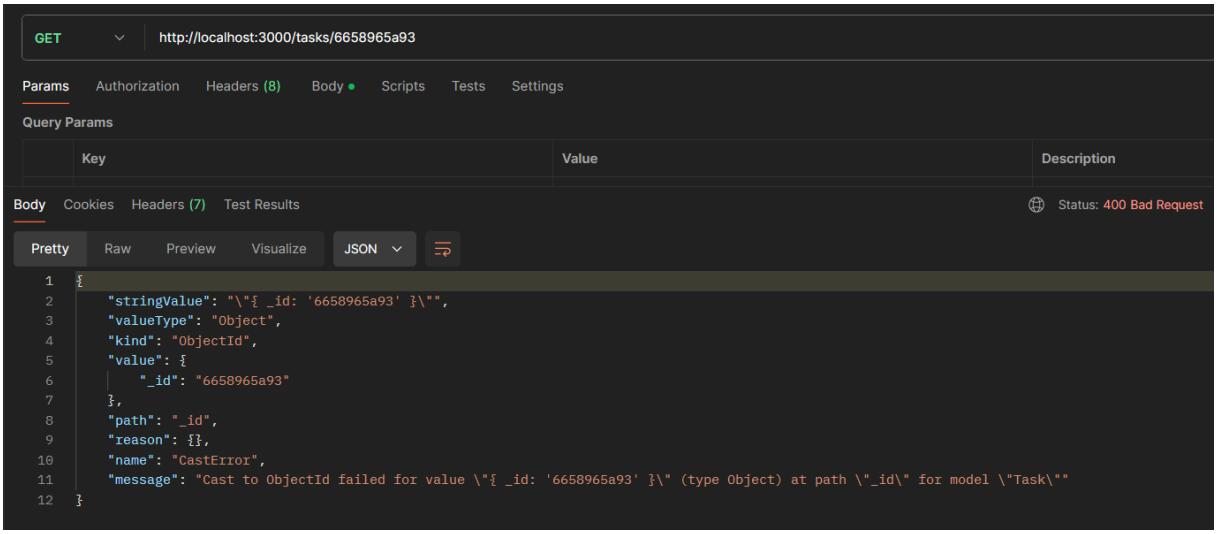
Yine doğru bir id girildiğinde başarılı bir şekilde veriyi döndürdü. Durum kodu:200

Daha sonra son haneyi değiştirdim.



Girilen id değerinin bulunmadığına dair kod gönderdi. Durum kod: 404

Son olarak da boyutu da farklı girdim.



Burada da objectId hatası aldım. Durum kodu: 400