



**Raihan Eka Pramudya**

**3122600011**

POST TEST Praktikum Basis Data Lanjut  
membuat API

**Source Code (GitHub):**

**Click Here**



**INSTALASI**

- Buat folder baru:

```
Express_Basdat
```

- Buka terminal dan inisialisasi dengan perintah berikut:

```
npm init
```

- Enter dan isi seperti perintah berikut:

```
package name: (express_basdat)
version: (1.0.0)
description: Express Basdat
entry point: (index.js) server.js
test command:
git repository: github.com/qiau/Express-Basdat
keywords:
author: qiau
license: (ISC)
```



- Lanjutkan instalasi dengan perintah berikut:

```
PS D:\APPLICATION\Express\Express_Basdat> npm install express mongoose cors --save
```

- Jika muncul seperti gambar di bawah maka instalasi berhasil:

```
added 86 packages, and audited 87 packages in 35s
```

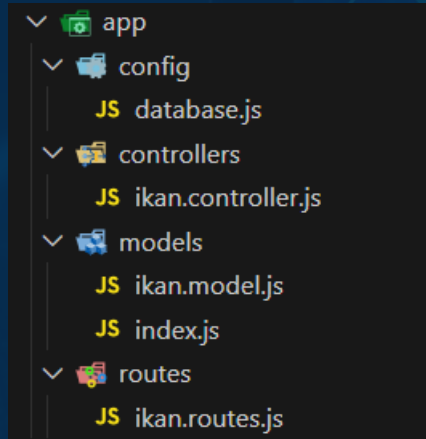
```
12 packages are looking for funding  
  run 'npm fund' for details
```

```
found 0 vulnerabilities
```



# KONFIGURASI

- Buka project pada code editor, lalu buat folder bernama app beserta isinya sebagai berikut:



- Buat file baru bernama server.js:

JS server.js

# 1. File **database.js**

## - Source Code

```
module.exports = {  
  url: 'mongodb://localhost:27017/express_basdat'  
};
```



## 2. File `ikan.controller.js`

### - Source Code

```
const db = require('../models');  
const Ikan = db.ikan
```

#### a. create

```
exports.create = (req, res) => {  
  req.body.id_penjual = new Number(req.body.id_penjual);  
  
  Ikan.create(req.body)  
    .then(() => {  
      res.send({message: "Data ikan berhasil disimpan"});  
    }).catch((err) => {  
      res.status(500).send({  
        message: err.message || "Error menginputkan data ikan."  
      });  
    });  
}
```

## 2. File `ikan.controller.js`

### b. `findAll`

```
exports.findAll = (req, res) => {  
  Ikan.find()  
    .then((result) => {  
      res.send(result);  
    }).catch((err) => {  
      res.status(500).send({  
        message: err.message || "Error mendapatkan data Ikan."  
      });  
    });  
}
```

## 2. File `ikan.controller.js`

### c. update

```
exports.update = (req, res) => {
  const id = req.params.id;

  req.body.id_penjual = new Number(req.body.id_penjual);

  Ikan.findByIdAndUpdate(id, req.body, { useFindAndModify: false })
    .then((result) => {
      if (!result) {
        res.status(404).send({
          message: "Ikan tidak ditemukan dengan id " + id
        });
      } else {
        res.send({ message: "Data ikan berhasil diupdate" });
      }
    }).catch((err) => {
      res.status(500).send({
        message: err.message || "Error memperbarui data ikan dengan id " + id
      });
    });
}
```

## 2. File `ikan.controller.js`

### d. delete

```
exports.delete = (req, res) => {  
  const id = req.params.id;  
  
  Ikan.findByIdAndDelete(id)  
    .then((result) => {  
      if (!result) {  
        res.status(404).send({  
          message: "Ikan tidak ditemukan dengan id " + id  
        });  
      }  
      res.send({ message: "Ikan berhasil dihapus!" });  
    }).catch((err) => {  
      res.status(500).send({  
        message: err.message || "Gagal menghapus ikan dengan id " + id  
      });  
    });  
}
```



## 2. File `ikan.controller.js`

### e. show

```
exports.show = (req, res) => {  
  const id = req.params.id;  
  
  Ikan.findById(id)  
    .then((result) => {  
      if (!result) {  
        res.status(404).send({  
          message: "Ikan tidak ditemukan dengan id " + id  
        });  
      }  
      res.send(result);  
    }).catch((err) => {  
      res.status(500).send({  
        message: err.message || "Error mendapatkan ikan dengan id " + id  
      });  
    });  
}
```

### 3. File `ikan.model.js`

#### - Source Code

```
module.exports = mongoose => {  
  ...  
  const schema = mongoose.Schema(  
    {  
      nama_ikan: String,  
      harga_ikan: Number,  
      stok_ikan: Number,  
      deskripsi_ikan: String,  
      kategori_ikan: String,  
      id_penjual: Number  
    },  
    { timestamps: true }  
  );  
  
  schema.method("toJSON", function () {  
    const { __v, _id, ...object } = this.toObject();  
    object.id_ikan = _id;  
    return object;  
  });  
  
  return mongoose.model("ikan", schema);  
}
```

## 4. File `index.js`

### - Source Code

```
const dbConfig = require("../config/database");
const mongoose = require("mongoose");

module.exports = {
  mongoose,
  url: dbConfig.url,
  ikan: require("./ikan.model.js")(mongoose)
}
```

## 5. File `ikan.routes.js`

### - Source Code

```
module.exports = app => {  
  const ikan = require("../controllers/ikan.controller")  
  const router = require("express").Router();  
  
  router.get("/", ikan.findAll);  
  router.get("/:id", ikan.show);  
  router.post("/", ikan.create);  
  router.put("/:id", ikan.update);  
  router.delete("/:id", ikan.delete);  
  
  app.use('/ikan', router);  
}
```



## 6. File server.js

### - Source Code

```
const express = require('express');
const cors = require('cors');
const db = require('./app/models');
const app = express();

const corsOptions = {
  origin: '*'
};

app.use(cors(corsOptions));
app.use(express.json());

db.mongoose.connect(db.url, { useNewUrlParser: true, useUnifiedTopology: true })
  .then(() => {
    console.log('Connected to the database!');
  }).catch(err => {
    console.log('Cannot connect to the database!', err);
    process.exit();
  });

require('./app/routes/ikan.routes')(app);

const PORT = process.env.PORT || 8000;

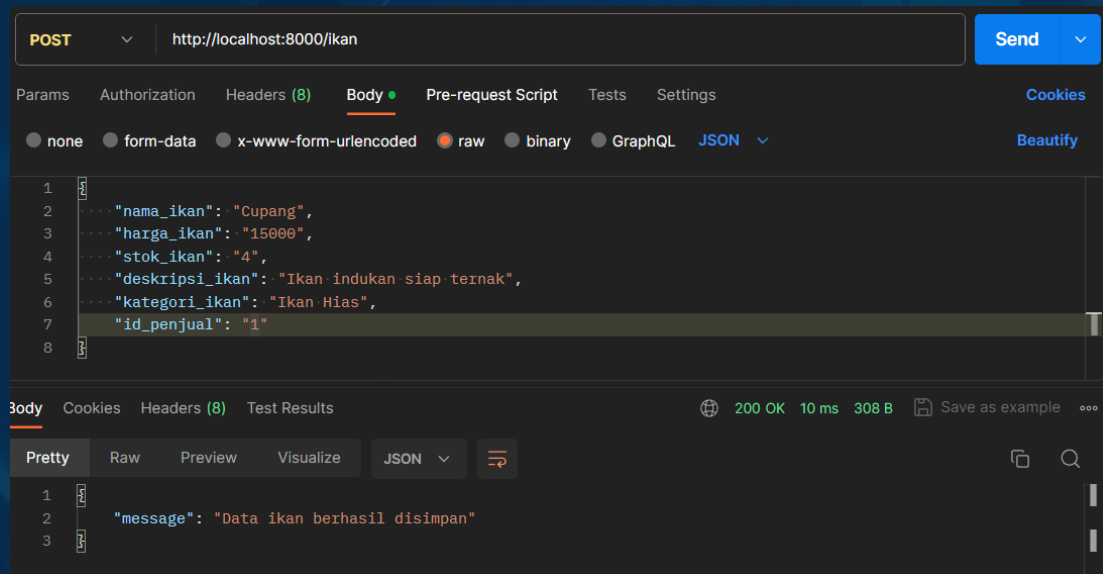
app.listen(PORT, () => {
  console.log(`Server is running on port ${PORT}.`);
});
```



**TESTING**

# 1. Create Data

## - Postman

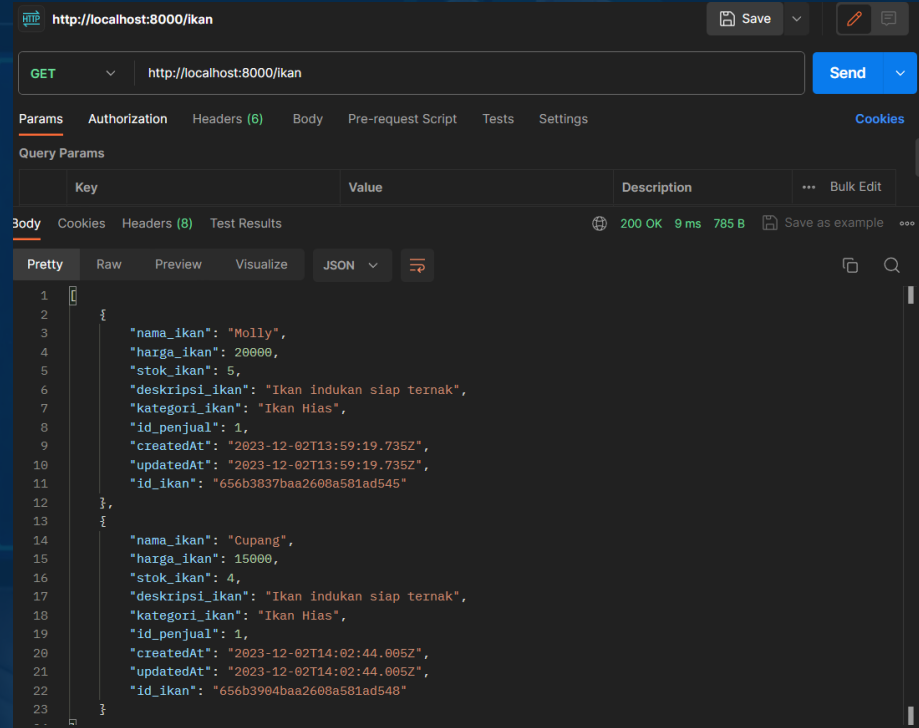


## - MongoDB



## 2. findAll Data

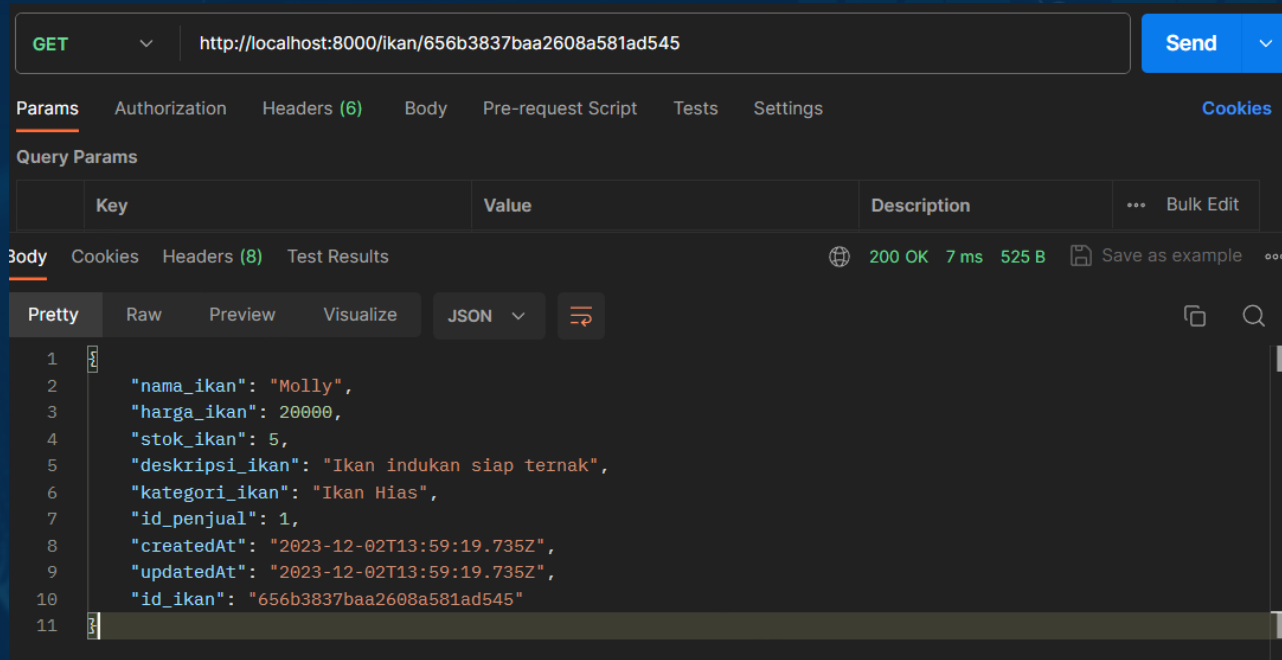
- Postman





### 3. Show Data **by ID**

#### - Postman



The screenshot shows a Postman interface with a GET request to the endpoint `http://localhost:8000/ikan/656b3837baa2608a581ad545`. The response is a JSON object with the following data:

Key	Value
nama_ikan	"Molly",
harga_ikan	20000,
stok_ikan	5,
deskripsi_ikan	"Ikan indukan siap ternak",
kategori_ikan	"Ikan Hias",
id_penjual	1,
createdAt	"2023-12-02T13:59:19.735Z",
updatedAt	"2023-12-02T13:59:19.735Z",
id_ikan	"656b3837baa2608a581ad545"

## 4. Update Data

### - Postman

The Postman interface shows a PUT request to the URL `http://localhost:8000/ikan/656b3837bba2608a581ad545`. The request body is a JSON object with the following fields:

```
1 {
2   "nama_ikan": "Molly Tiger",
3   "harga_ikan": 29000,
4   "stok_ikan": 3,
5   "deskripsi_ikan": "Ikan indukan siap ternak",
6   "kategori_ikan": "Ikan Hias",
7   "id_penjual": 1
8 }
```

The response status is 200 OK. The response body is a JSON object with the following fields:

```
1 {
2   "message": "Data ikan berhasil diupdate"
3 }
```

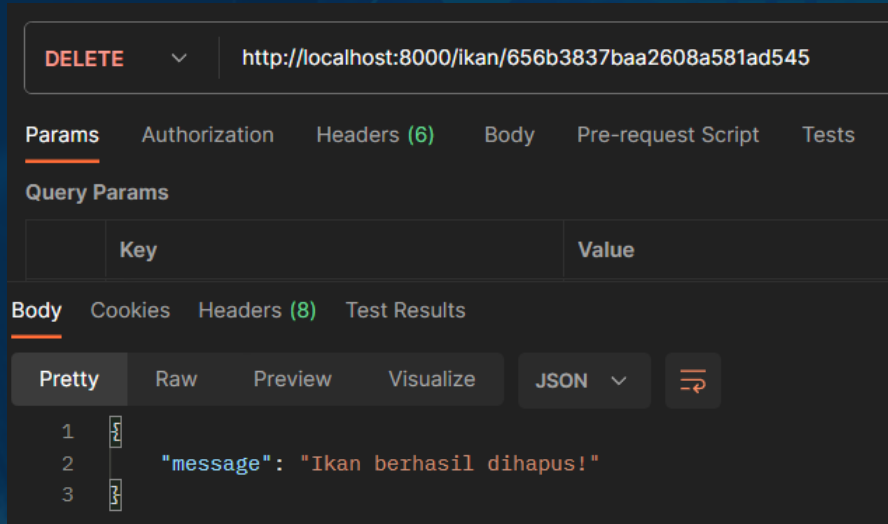
### - MongoDB

The MongoDB interface shows the updated document with the following fields:

```
_id: ObjectId('656b3837bba2608a581ad545')
nama_ikan: "Molly Tiger"
harga_ikan: 29000
stok_ikan: 3
deskripsi_ikan: "Ikan indukan siap ternak"
kategori_ikan: "Ikan Hias"
id_penjual: 1
createdAt: 2023-12-02T13:59:19.735+00:00
updatedAt: 2023-12-02T14:12:13.934+00:00
__v: 0
```

## 5. Delete Data

### - Postman



The image shows the Postman interface for a DELETE request. The URL is `http://localhost:8000/ikan/656b3837bba2608a581ad545`. The 'Body' tab is selected, showing a JSON response: `{ "message": "Ikan berhasil dihapus!" }`. The response is formatted as 'Pretty' JSON.

**DELETE** `http://localhost:8000/ikan/656b3837bba2608a581ad545`

Params Authorization Headers (6) Body Pre-request Script Tests

Query Params

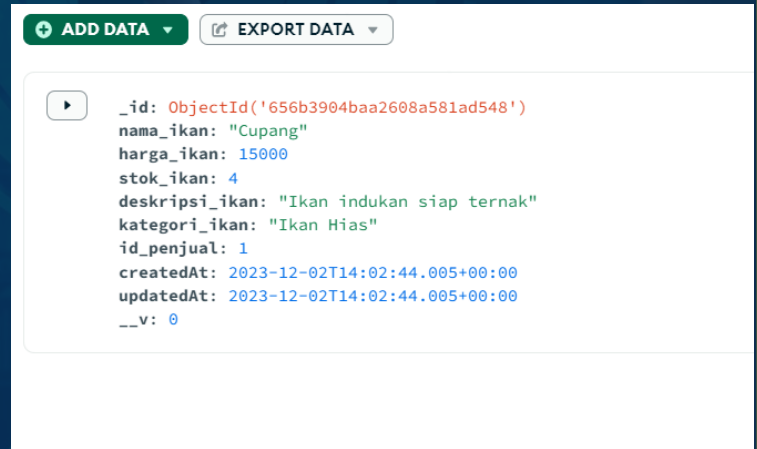
Key	Value
-----	-------

Body Cookies Headers (8) Test Results

Pretty Raw Preview Visualize JSON `{ }`

```
1 {  
2   "message": "Ikan berhasil dihapus!"  
3 }
```

### - MongoDB



The image shows the MongoDB data view interface. It displays a single document with the following fields: `_id`, `nama_ikan`, `harga_ikan`, `stok_ikan`, `deskripsi_ikan`, `kategori_ikan`, `id_penjual`, `createdAt`, `updatedAt`, and `__v`.

ADD DATA EXPORT DATA

```
{  
  "_id": ObjectId('656b3904bba2608a581ad548'),  
  "nama_ikan": "Cupang",  
  "harga_ikan": 15000,  
  "stok_ikan": 4,  
  "deskripsi_ikan": "Ikan indukan siap ternak",  
  "kategori_ikan": "Ikan Hias",  
  "id_penjual": 1,  
  "createdAt": "2023-12-02T14:02:44.005+00:00",  
  "updatedAt": "2023-12-02T14:02:44.005+00:00",  
  "__v": 0  
}
```

The background is a dark blue gradient with various faint, light blue icons related to technology and business, such as a cloud, a person with a headset, a laptop with a gear, a shield, a microchip, and a globe. A yellow border frames the central content area, with a yellow line extending from the top left and another from the bottom right, each ending in a small yellow dot.

# Thanks!

Dosen Pembimbing:  
Tessy Badriyah, S.Kom., M.T., Ph.D.