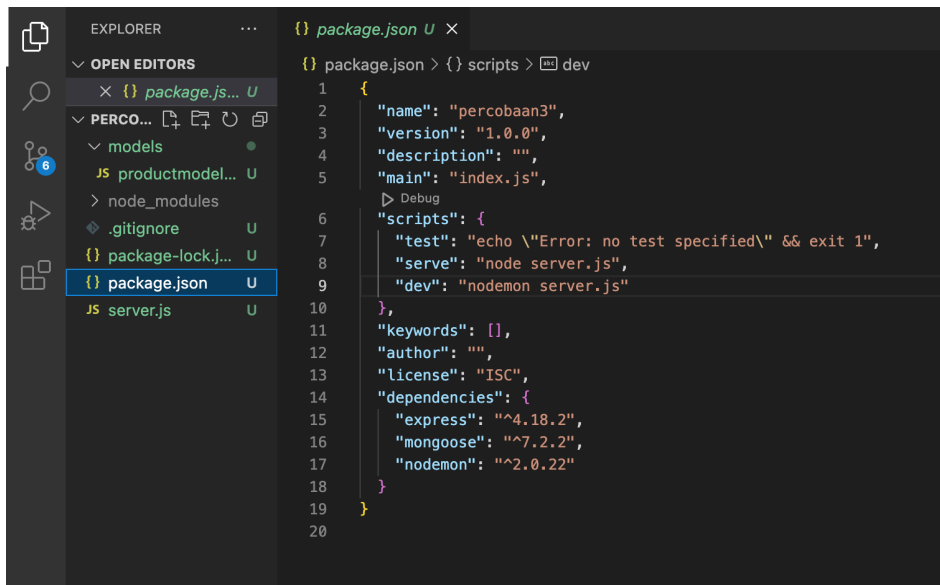


Laporan Praktikum Teknologi Web Lanjut

Pengenalan Node.js, Express dan MongoDB

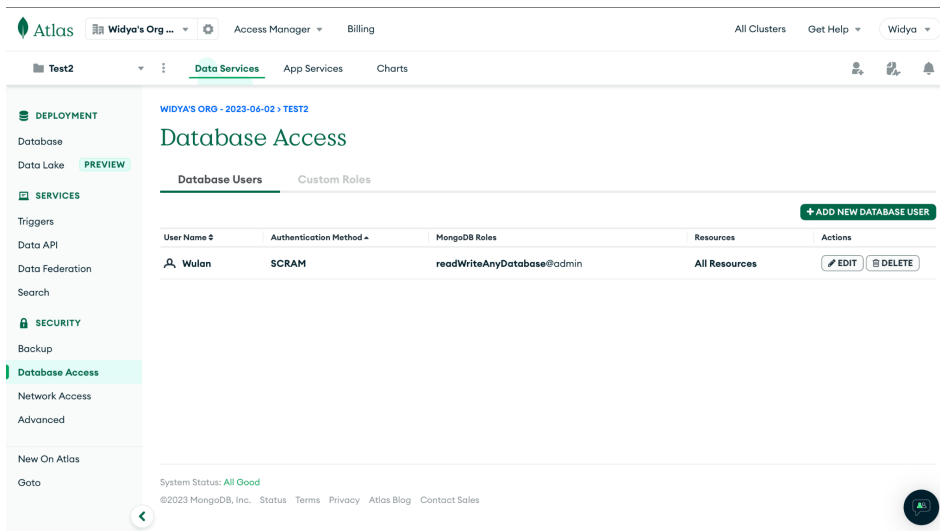
Praktikum pertemuan 6

1. Menyiapkan lingkungan pengembangan Node.js dan express dengan cara Menginstall node.js, Express, Mongoose, nodemon.



```
1 {
2   "name": "percobaan3",
3   "version": "1.0.0",
4   "description": "",
5   "main": "index.js",
6   "scripts": {
7     "test": "echo \"Error: no test specified\" && exit 1",
8     "serve": "node server.js",
9     "dev": "nodemon server.js"
10  },
11  "keywords": [],
12  "author": "",
13  "license": "ISC",
14  "dependencies": {
15    "express": "^4.18.2",
16    "mongoose": "^7.2.2",
17    "nodemon": "^2.0.22"
18  }
19 }
```

2. Membuat new project dengan nama “Test2” dan membuat database user di dalam MongoDB.



3. Membuat cluster pada mongoDB

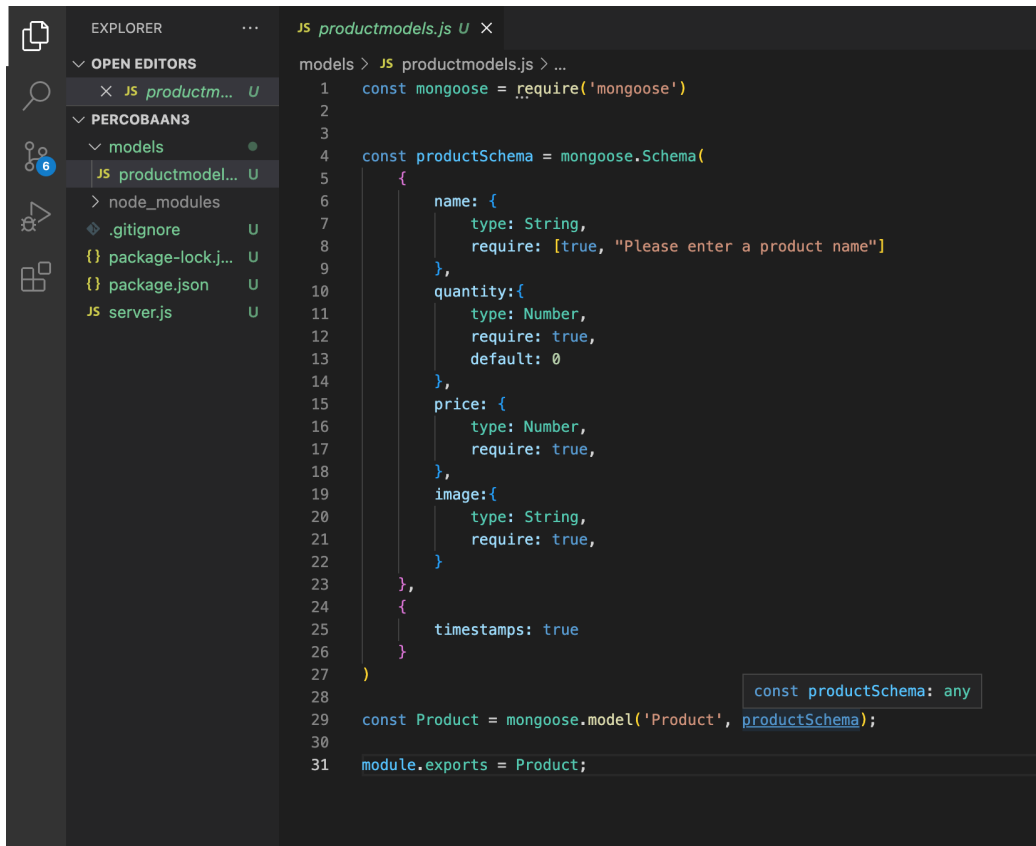
The screenshot shows the MongoDB Atlas interface for a deployment named 'Test2'. The left sidebar contains navigation links for Deployment, Database, Data Lake, Services, Security, and Backup. The main content area is titled 'Database Deployments' and includes a search bar, a '+ Create' button, and a 'Load sample datasets to ClusterAPI' button. Below this, there's a 'Visualize Your Data' section with several charts: Read (R) at 0.007, Write (W) at 0, Connections at 10.0, In/Out traffic at 147.8 B/s and 801.2 B/s, and Data Size at 36.6 KB. A table at the bottom provides details about the deployment, including version (6.0.6), region (AWS / Singapore), cluster tier (M0 Sandbox), type (Replica Set - 3 nodes), backups (Inactive), linked app services (None Linked), and atlas sql (Connect).

VERSION	REGION	CLUSTER TIER	TYPE	BACKUPS	LINKED APP SERVICES	ATLAS SQL
6.0.6	AWS / Singapore (ap-southeast-1)	M0 Sandbox (General)	Replica Set - 3 nodes	Inactive	None Linked	Connect

4. Mengkoneksikan MongoDB dengan Node.js

```
93 //mengoneksikan ke mongoDB
94 mongoose.connect('mongodb+srv://Wulan:admin1234@clusterapi.thuzaaff.mongodb.net/users?retryWrites=true&w=majority')
95 .then(()=>{
96   console.log('connected to MongoDB')
97   app.listen(3000, ()=>{
98     console.log('Node API app is running on port 3000')
99   })
100 }).catch((error) =>{
101   console.log(error)
102 })
103 }
```

5. Membuat Schema MongoDB

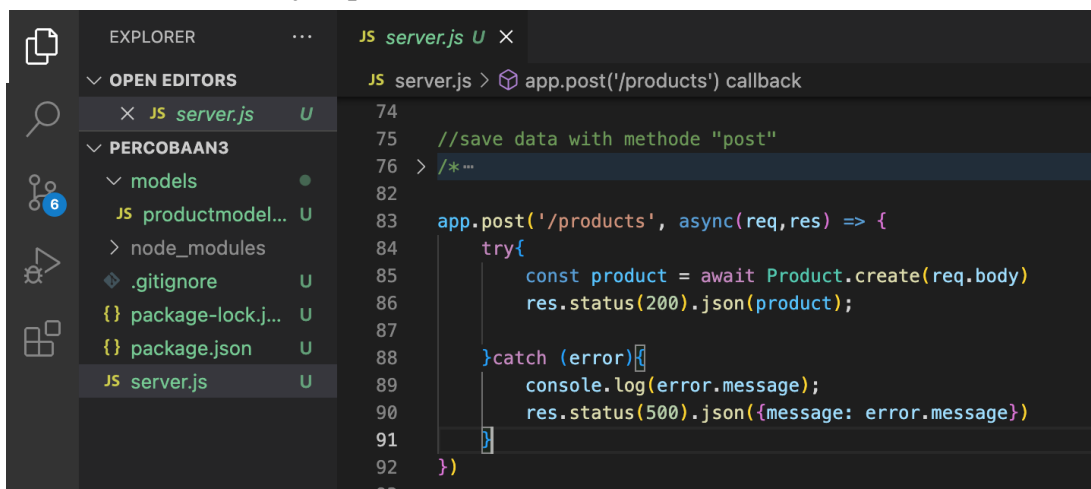


The screenshot shows a VS Code editor with a file explorer on the left and a code editor on the right. The file explorer shows a project named 'PERCOBAAN3' with a 'models' folder containing 'productmodel.js'. The code editor shows the content of 'productmodels.js' with the following code:

```
1 const mongoose = require('mongoose')
2
3
4 const productSchema = mongoose.Schema(
5   {
6     name: {
7       type: String,
8       require: [true, "Please enter a product name"]
9     },
10    quantity:{
11      type: Number,
12      require: true,
13      default: 0
14    },
15    price: {
16      type: Number,
17      require: true,
18    },
19    image:{
20      type: String,
21      require: true,
22    }
23  },
24  {
25    timestamps: true
26  }
27 )
28
29 const Product = mongoose.model('Product', productSchema);
30
31 module.exports = Product;
```

6. Mengimplementasikan operasi CRUD (Create, Read, Update, Delete) untuk sumber daya sederhana.

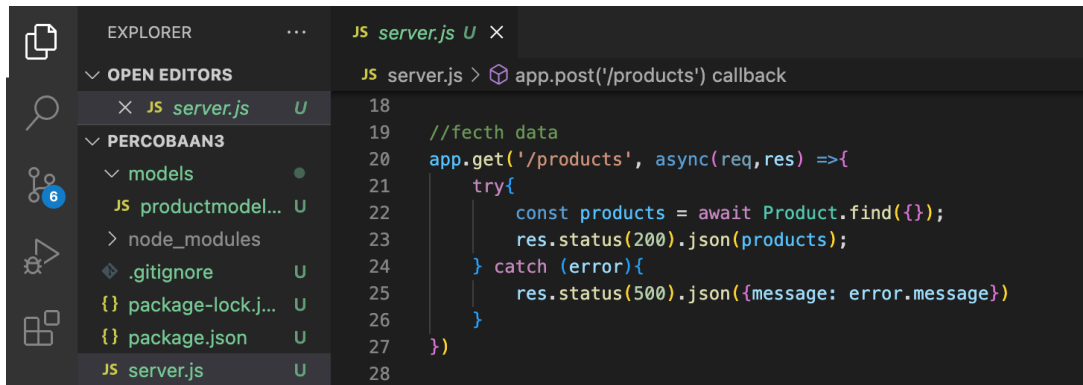
-Membuat dan menyimpan data



The screenshot shows a VS Code editor with a file explorer on the left and a code editor on the right. The file explorer shows a project named 'PERCOBAAN3' with a 'models' folder containing 'productmodel.js' and a 'server.js' file. The code editor shows the content of 'server.js' with the following code:


```
74 JS server.js > app.post('/products') callback
75 //save data with methode "post"
76 > /* ...
82
83 app.post('/products', async(req,res) => {
84   try{
85     const product = await Product.create(req.body)
86     res.status(200).json(product);
87   }catch (error){
88     console.log(error.message);
89     res.status(500).json({message: error.message})
90   }
91 }
92 })
```

-Fetch seluruh data



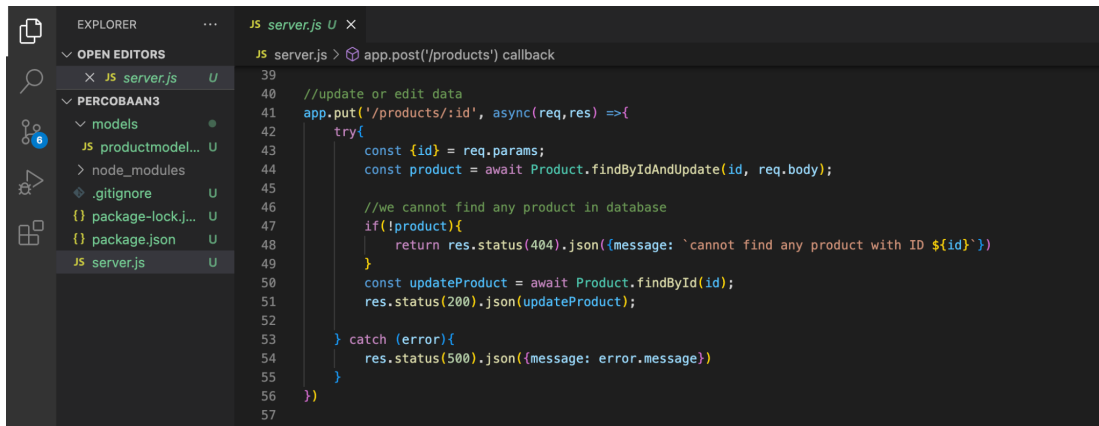
```
JS server.js > app.post('/products') callback
18
19 //fetch data
20 app.get('/products', async(req,res) =>{
21     try{
22         const products = await Product.find({});
23         res.status(200).json(products);
24     } catch (error){
25         res.status(500).json({message: error.message});
26     }
27 }
28
```

-Fetch data berdasarkan "ID"



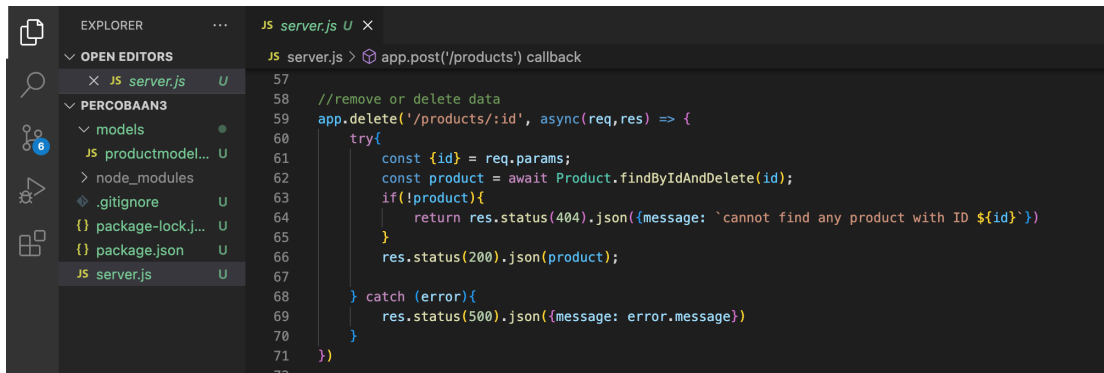
```
JS server.js > app.post('/products') callback
29 //fetch data from id
30 app.get('/products/:id', async(req,res) =>{
31     try{
32         const {id} = req.params;
33         const product = await Product.findById(id);
34         res.status(200).json(product);
35     } catch (error){
36         res.status(500).json({message: error.message});
37     }
38 }
39
```

-Update or edit data



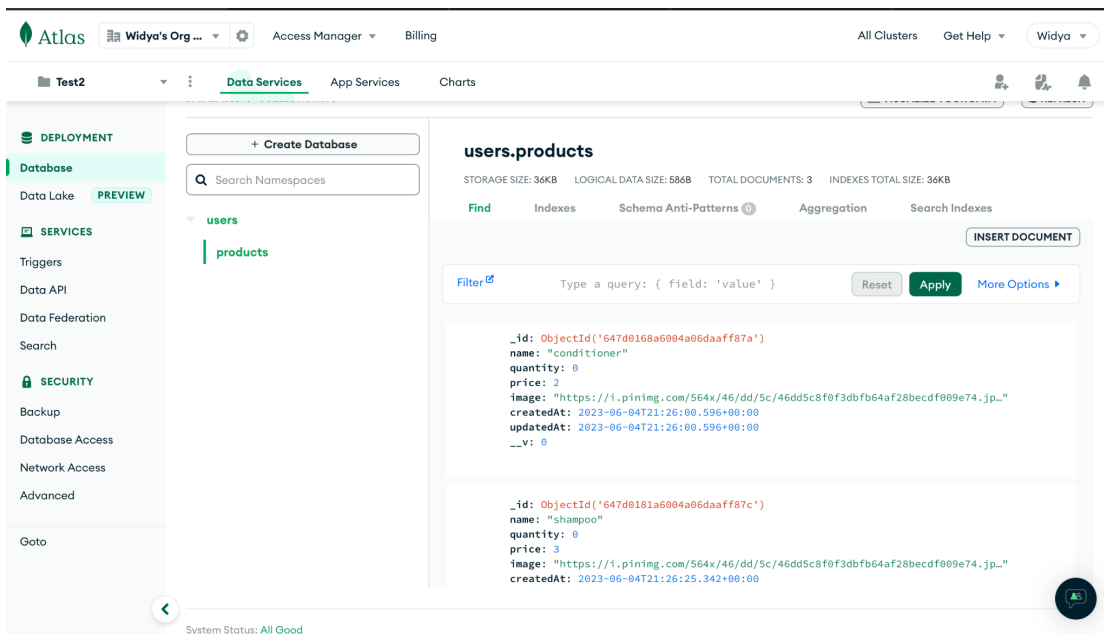
```
JS server.js > app.post('/products') callback
39
40 //update or edit data
41 app.put('/products/:id', async(req,res) =>{
42     try{
43         const {id} = req.params;
44         const product = await Product.findByIdAndUpdate(id, req.body);
45
46         //we cannot find any product in database
47         if(!product){
48             return res.status(404).json({message: `cannot find any product with ID ${id}`});
49         }
50         const updateProduct = await Product.findById(id);
51         res.status(200).json(updateProduct);
52     } catch (error){
53         res.status(500).json({message: error.message});
54     }
55 }
56 }
57
```

- Remove or Delete data

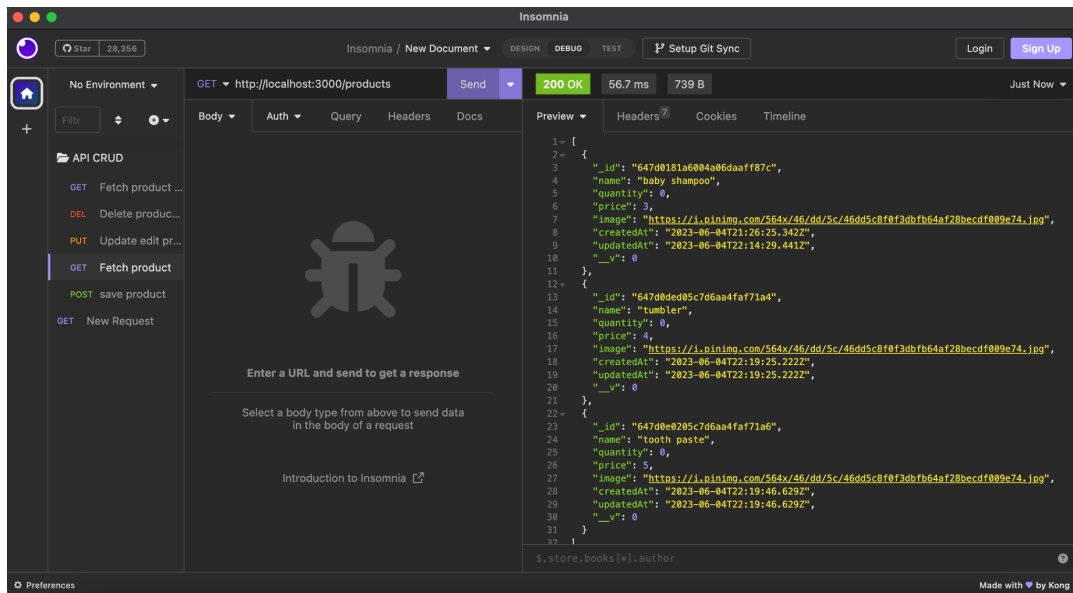


```
JS server.js > app.post('/products') callback
57
58 //remove or delete data
59 app.delete('/products/:id', async(req,res) => {
60   try{
61     const {id} = req.params;
62     const product = await Product.findByIdAndDelete(id);
63     if(!product){
64       return res.status(404).json({message: `cannot find any product with ID ${id}`});
65     }
66     res.status(200).json(product);
67   } catch (error){
68     res.status(500).json({message: error.message});
69   }
70 }
71 })
72
```

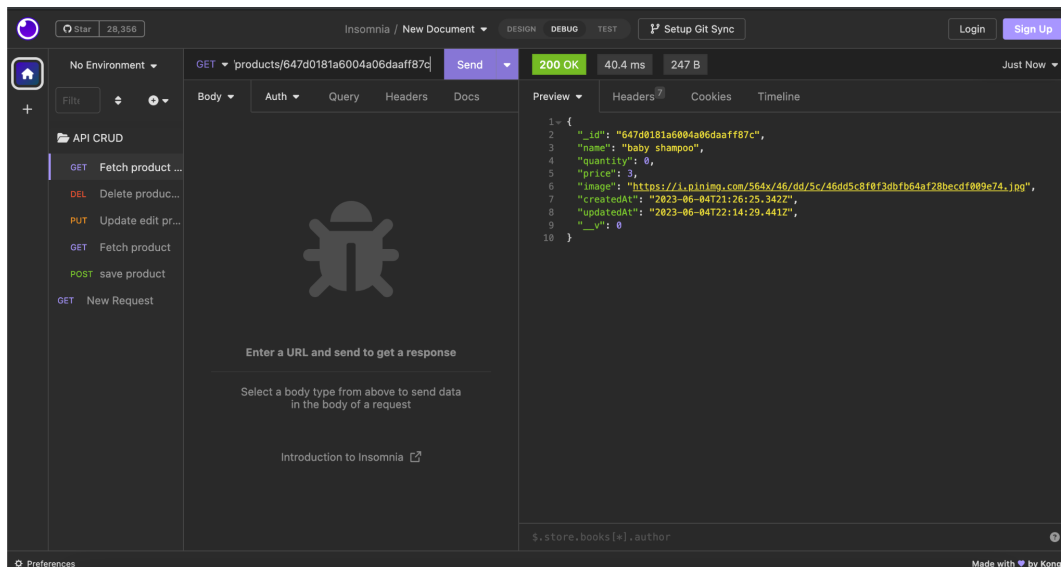
7. Tampilan data yang ada di MongoDB



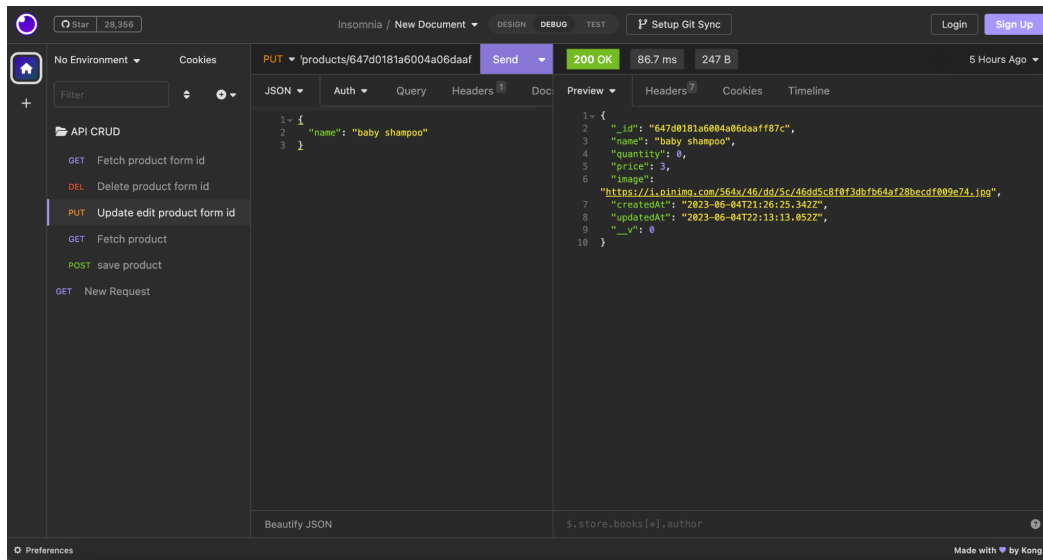
8. Pengujian dan debugging API menggunakan Insomnia, dengan url (http://localhost:3000/products)
- Get/fetch data:



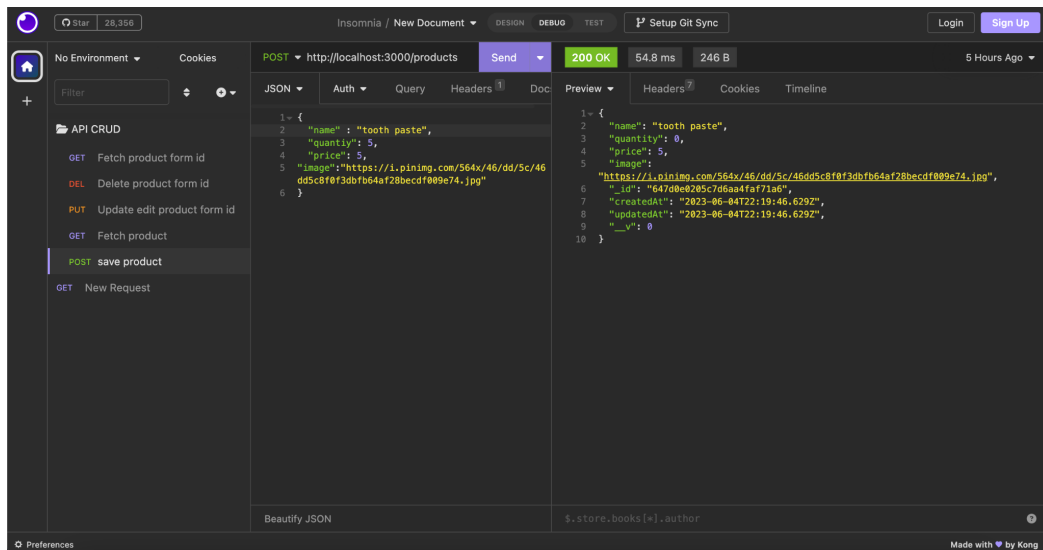
- Get/fetch data from 'id'



- Update or Edit data from 'id' with "PUT"



- Save data with "POST"



- Delete data from 'id' with "DELETE"

