

# BE NodeJS

## Tugas: Mongoose

Leonardho R. Sitanggang

Soal :

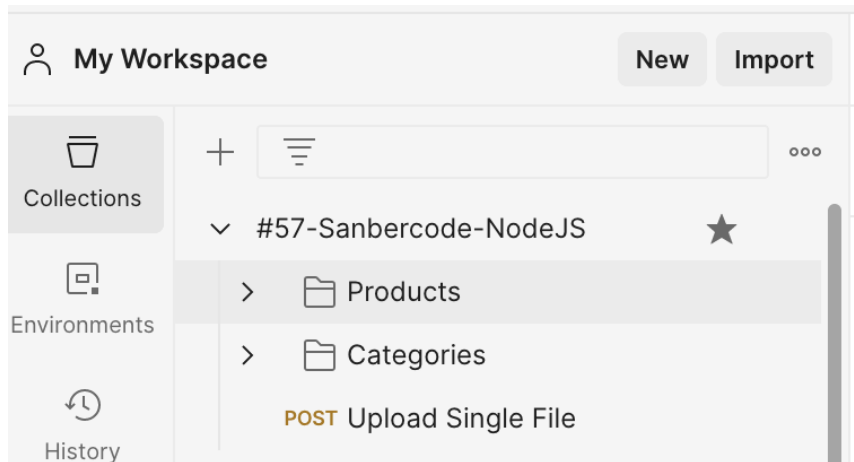
Gunakan [source code materi 7](#),

Buatlah file di folder models, dengan nama **categories.model.ts** , kemudian isi dengan kode berikut ini untuk membuat schema categories

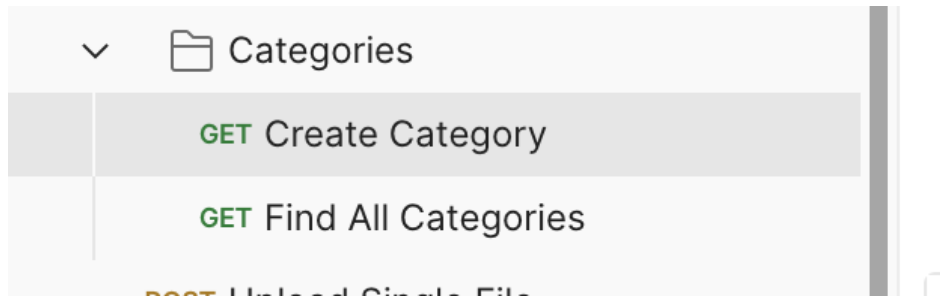
Pastikan sudah import categories.controller di bagian atas file routes.ts

Jalankan server dengan perintah `npm run dev`

Buka postman, kemudian buatlah [struktur collection](#) seperti ini



Di dalam folder categories buatlah beberapa request contohnya seperti ini

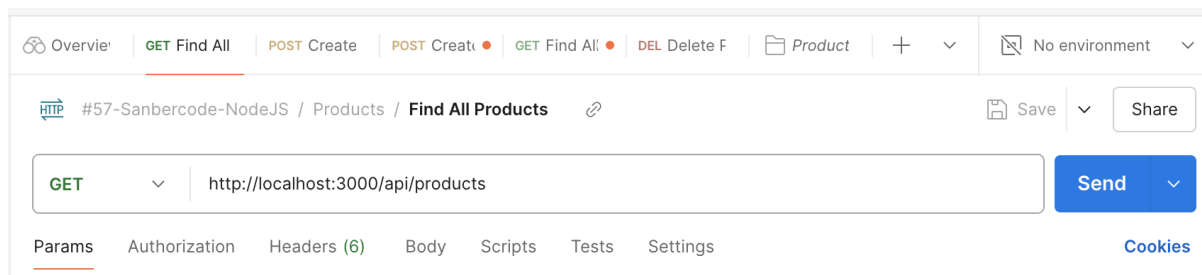


Lakukan proses CRUD pada data Category

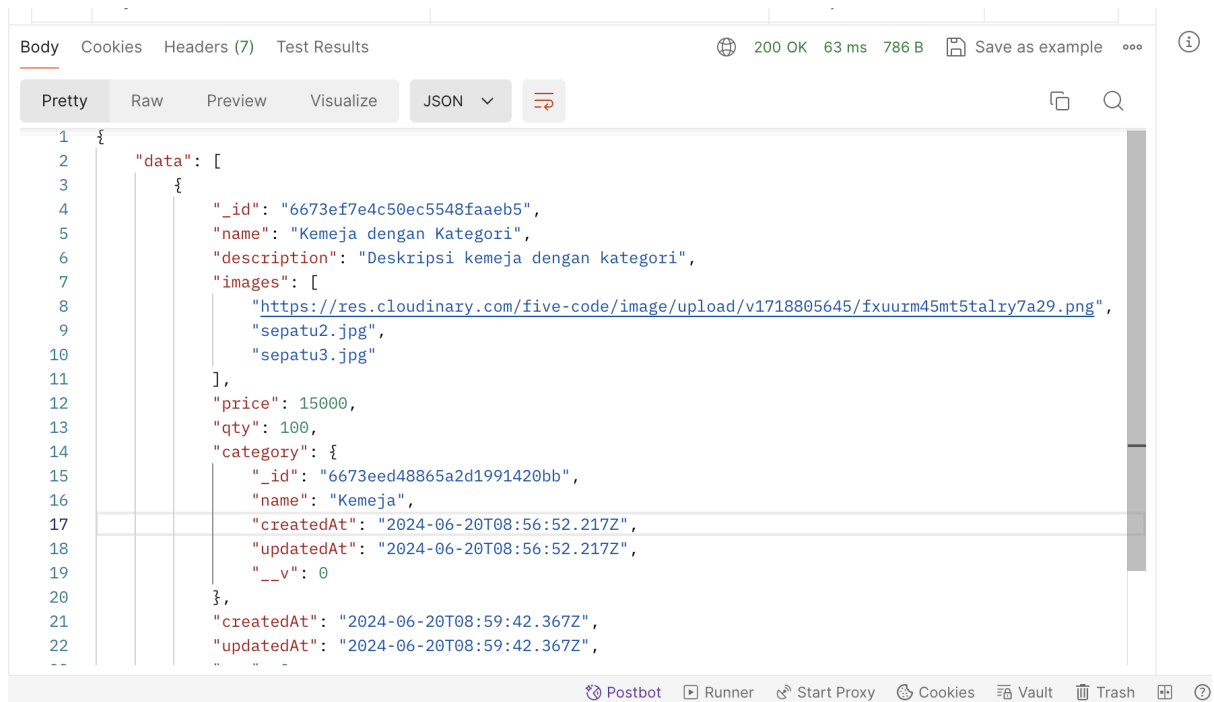
Kemudian, buatlah data **Product** dengan JSON seperti ini :

```
{
  "name": "Kemeja dengan Kategori",
  "description": "Deskripsi kemeja dengan kategori",
  "images":
  ["https://res.cloudinary.com/five-code/image/upload/v1718805645/fxuurm45mt5talry7a29.png", "sepatu2.jpg", "sepatu3.jpg"],
  "price": 15000,
  "qty": 100,
  "category": "-- isi dengan _id dari salah satu category --"
}
```

Gunakan endpoint **Find All Products** untuk menampilkan list data Products



Pastikan hasil dari endpoint ini menampilkan data seperti pada contoh di bawah ini:



di dalam data product dapat menampilkan isi dari **category**.

Jawab :

Source Code

```
16 router.get("/categories", categoriesController.findAll);
17 router.post("/categories", categoriesController.create);
18 router.get("/categories/:id", categoriesController.findOne);
19 router.put("/categories/:id", categoriesController.update);
20 router.delete("/categories/:id", categoriesController.delete);
```

Gambar 1.1 Route Categories

```

4   export default {
5     async create(req: Request, res: Response) {
6       try {
7         const result = await CategoriesModel.create(req.body);
8         res.status(201).json({
9           data: result,
10          message: "Success create product",
11        });
12      } catch (error) {
13        const err = error as Error;
14        res.status(500).json({
15          data: err.message,
16          message: "Failed create product",
17        });
18      }
19    },
20    async findAll(req: Request, res: Response) {
21      try {
22        const result = await CategoriesModel.find();
23        res.status(200).json({
24          data: result,
25          message: "Success get all products",
26        });
27      } catch (error) {
28        const err = error as Error;
29        res.status(500).json({
30          data: err.message,
31          message: "Failed get all products",
32        });
33      }
34    },
35    async findOne(req: Request, res: Response) {
36      try {
37        const result = await CategoriesModel.findOne({
38          _id: req.params.id,
39        });
40        res.status(200).json({
41          data: result,
42          message: "Success get one product",
43        });
44      } catch (error) {
45        const err = error as Error;
46        res.status(500).json({
47          data: err.message,
48          message: "Failed get one product",
49        });
50      }
51    },

```

Gambar 1.2 Controller Create, Read

```

52   async update(req: Request, res: Response) {
53       try {
54           const result = await CategoriesModel.findOneAndUpdate(
55               { _id: req.params.id },
56               req.body,
57               {
58                   new: true,
59               }
60           );
61
62           res.status(200).json({
63               data: result,
64               message: "Success update product",
65           });
66       } catch (error) {
67           const err = error as Error;
68           res.status(500).json({
69               data: err.message,
70               message: "Failed update product",
71           });
72       }
73   },
74   async delete(req: Request, res: Response) {
75       try {
76           const result = await CategoriesModel.findOneAndDelete({
77               _id: req.params.id,
78           });
79
80           res.status(200).json({
81               data: result,
82               message: "Success delete product",
83           });
84       } catch (error) {
85           const err = error as Error;
86           res.status(500).json({
87               data: err.message,
88               message: "Failed delete product",
89           });
90       }
91   },

```

*Gambar 1.3 Controller Update, Delete*

sanber-be-59.categories

STORAGE SIZE: 36KB   LOGICAL DATA SIZE: 203B   TOTAL DOCUMENTS: 2   INDEXES TOTAL SIZE: 36KB

Find

Indexes

Schema Anti-Patterns 0

Aggregation

Search Indexes

Generate queries from natural language in Compass

Filter

Type a query: { field: 'value' }

QUERY RESULTS: 1-2 OF 2

\_id: ObjectId('672e27a3c26057c576eab6ac')

name: "Plastic"

products: Array (empty)

createdAt: 2024-11-08T15:00:51.443+00:00

updatedAt: 2024-11-08T15:00:51.443+00:00

\_\_v: 0

\_id: ObjectId('672e28a6c26057c576eab6b1')

name: "Rubber"

products: Array (empty)

createdAt: 2024-11-08T15:05:10.790+00:00

updatedAt: 2024-11-08T15:05:10.790+00:00

\_\_v: 0

sanber-be-59.products

STORAGE SIZE: 36KB   LOGICAL DATA SIZE: 180B   TOTAL DOCUMENTS: 1   INDEXES TOTAL SIZE: 36KB

Find

Indexes

Schema Anti-Patterns 0

Aggregation

Search Indexes

Generate queries from natural language in Compass

Filter

Type a query: { field: 'value' }

QUERY RESULTS: 1-1 OF 1

\_id: ObjectId('672e28a7c26057c576eab6b7')

name: "Car"

description: "AGP application AI"

images: Array (empty)

price: 779.97

qty: 228

categoryId: ObjectId('672e27a3c26057c576eab6ac')

createdAt: 2024-11-08T15:05:11.128+00:00

updatedAt: 2024-11-08T15:05:11.128+00:00

\_\_v: 0

Gambar 2.1 Evidence Database MongoDB

```

5  const ProductsSchema = new Schema(
6    {
7      name: {
8        type: String,
9        required: true,
10     },
11     description: {
12       type: String,
13       required: true,
14     },
15     images: {
16       type: [String],
17       required: true,
18     },
19     price: {
20       type: Number,
21       required: true,
22     },
23     qty: {
24       type: Number,
25       required: true,
26       min: [1, "Quantity can not be less than 1"],
27     },
28     category: {
29       type: mongoose.Schema.Types.ObjectId,
30       ref: "Categories",
31     },
32   },
33   {
34     timestamps: true,
35   }
36 );
37
38 const ProductsModel = mongoose.model("Products", ProductsSchema);
39

```

```

5  const CategoriesSchema = new Schema(
6    {
7      name: {
8        type: String,
9        required: true,
10     },
11   },
12   {
13     timestamps: true,
14   }
15 );
16
17 const CategoriesModel = mongoose.model("Categories", CategoriesSchema);
18
19 export default CategoriesModel;
20

```

*Gambar 2.2 Model Products & Categories*

tugas-mongoose-odm - Run results

Ran today at 07:41:25

[View all runs](#)

Source	Environment	Iterations	Duration	All tests	Avg. Resp. Time
Runner	none	1	460ms	0	69 ms

All Tests

Passed (0)

Failed (0)

Skipped (0)

Iteration 1

1

POST

tugas-mongoose-odm / Categories / Create Category

Response

Headers

Request

Pretty

1

2

3

4

5

6

7

8

9

10

{

"data": {

"name": "Metal",

"\_id": "672eafb5530ea4f33040e8e4",

"createdAt": "2024-11-09T00:41:25.786Z",

"updatedAt": "2024-11-09T00:41:25.786Z",

"\_\_v": 0

}

,

"message": "Success create product"

}

POST Create Category

http://localhost:3000/api/categor... 201 Created

No tests found

GET Find All Categories

http://localhost:3000/api/categories 200 OK

No tests found

POST Product

http://localhost:3000/api/products 201 Created

No tests found

GET All Products

http://localhost:3000/api/products 200 OK

No tests found

Gambar 3.1 Test Result Post Category

tugas-mongoose-odm - Run results

Ran today at 07:41:25

[View all runs](#)

Source	Environment	Iterations	Duration	All tests	Avg. Resp. Time
Runner	none	1	460ms	0	69 ms

All Tests

Passed (0)

Failed (0)

Skipped (0)

Iteration 1

1

GET

tugas-mongoose-odm / Categories / Find All Categories

Response

Headers

Request

Pretty

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

{

"data": [

{

"\_id": "672ea82c81755c5b4924a9f4",

"name": "Soft",

"createdAt": "2024-11-09T00:09:16.313Z",

"updatedAt": "2024-11-09T00:09:16.313Z",

"\_\_v": 0

},

{

"\_id": "672eafb5530ea4f33040e8e4",

"name": "Metal",

"createdAt": "2024-11-09T00:41:25.786Z",

"updatedAt": "2024-11-09T00:41:25.786Z",

"\_\_v": 0

}

],

"message": "Success get all products"

}

POST Create Category

http://localhost:3000/api/categor... 201 Created

No tests found

GET Find All Categories

http://localhost:3000/api/categories 200 OK

No tests found

POST Product

http://localhost:3000/api/products 201 Created

No tests found

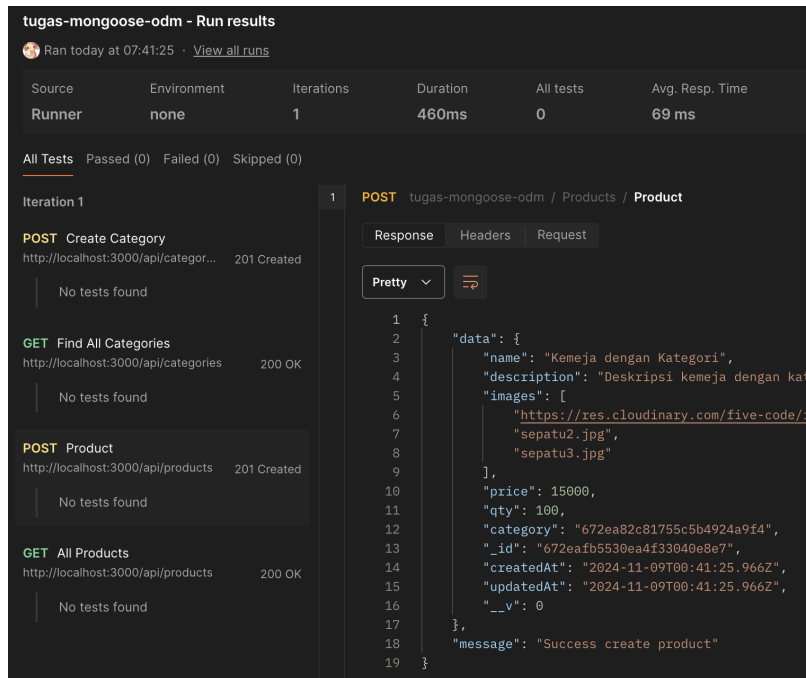
GET All Products

http://localhost:3000/api/products 200 OK

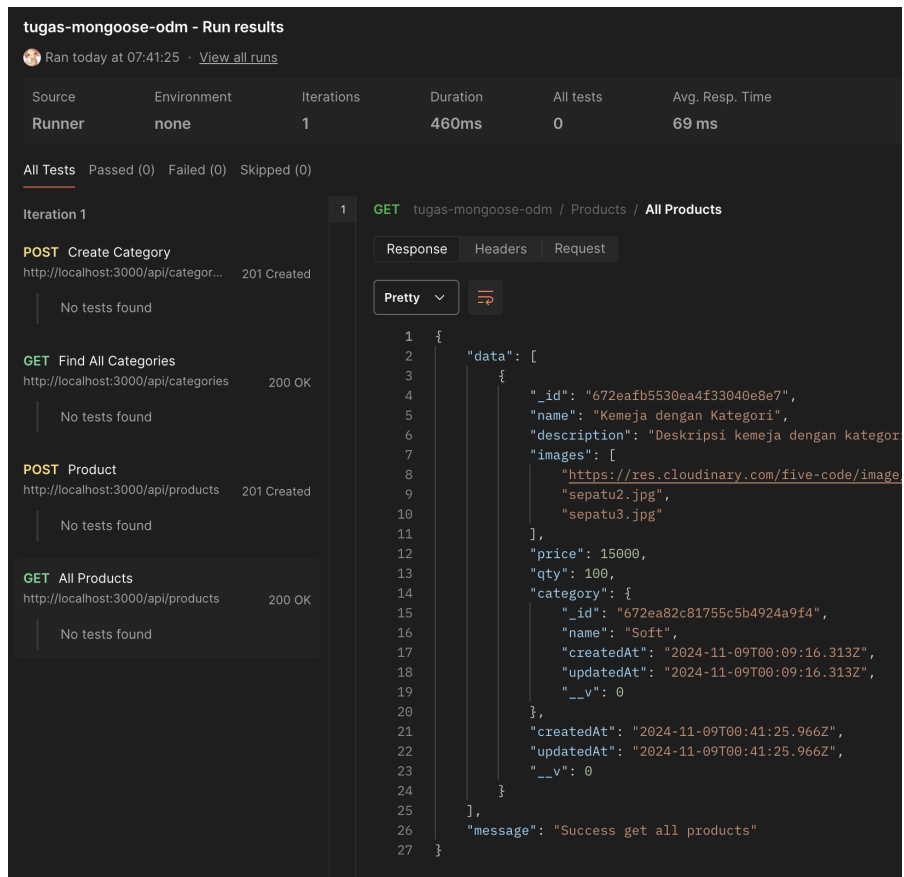
No tests found

Gambar 3.2 Test Result Get All Categories





Gambar 3.3 Test Result Post Product with Category Relation



Gambar 3.4 Test Result Get All Product with Category Relation

Repository :

- Github <https://github.com/FlazeFy/sanbercode-nodejs-xpress/tree/main/tugas-mongoose-odm>
- Postman Documentation <https://documenter.getpostman.com/view/18882129/2sAY52bz5m>

Refrensi :

- Materi Pekan 2.1 Framework Express JS  
<https://sanbercode.com/bootcamp/class/684/materi/2-1-framework-expressjs/dc0618b7-fd52-4eec-a578-5ed88c4d5b3c>
- Materi Pekan 2.2 Routing Rest API  
<https://sanbercode.com/bootcamp/class/684/materi/routing-rest-api/de70a7be-9056-40e0-bed0-c280c6bc1de0>
- Materi Pekan 2.3 Middleware Express  
<https://sanbercode.com/bootcamp/class/684/materi/2-3-middleware-expressjs/c33672de-a33d-4916-b07c-9221b9e836b5>
- Materi Pekan 2.4 Database MongoDB  
<https://sanbercode.com/bootcamp/class/684/materi/2-4-database-mongodb/ae319daf-6cf4-4282-8fe2-0c21c4da0d13>
- Materi Pekan 2.5 Mongoose  
<https://sanbercode.com/bootcamp/class/684/materi/2-6-video-mongoose/f919e6e2-ec3e-4888-aa76-1fb85c552df2>