BE NodeJS

Tugas: Routing Rest API

Leonardho R. Sitanggang

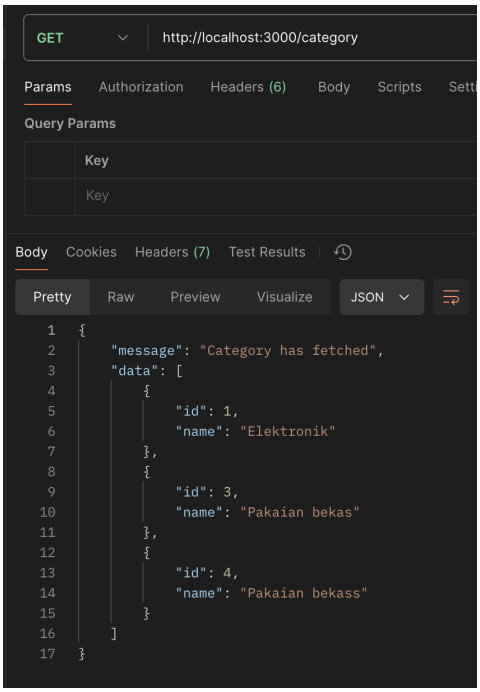
Soal :

1. Buatlah route GET yang mengembalikan daftar semua kategori produk dalam aplikasi e-commerce Anda. Anda bisa asumsikan data kategori disimpan dalam array seperti ini: `[{ id: 1, name: 'Elektronik' }, { id: 2, name: 'Perabotan' }]`.
2. Buatlah route GET yang mengembalikan detail kategori berdasarkan ID. Anda bisa menggunakan array kategori dari soal sebelumnya.
3. Buatlah route POST yang menambahkan kategori baru ke array. Kategori baru harus diberikan melalui body request dalam bentuk JSON, seperti ini: `{ "name": "Pakaian" }`.
4. Buatlah route PUT yang memperbarui kategori berdasarkan ID. Data kategori baru harus diberikan melalui body request dalam bentuk JSON, seperti ini: `{ "name": "Pakaian dan Aksesoris" }`.
5. Buatlah route DELETE yang menghapus kategori berdasarkan ID.
6. Buatlah route GET dengan query string untuk mencari produk berdasarkan nama. Anda bisa asumsikan data produk disimpan dalam array seperti ini: `[{ id: 1, name: 'Laptop', category: 'Elektronik' }, { id: 2, name: 'Meja', category: 'Perabotan' }]`.
7. Buatlah route GET dengan parameter dan query string untuk mendapatkan produk dalam kategori tertentu dan mencari berdasarkan nama. Anda bisa menggunakan array produk dari soal sebelumnya.

Jawab :

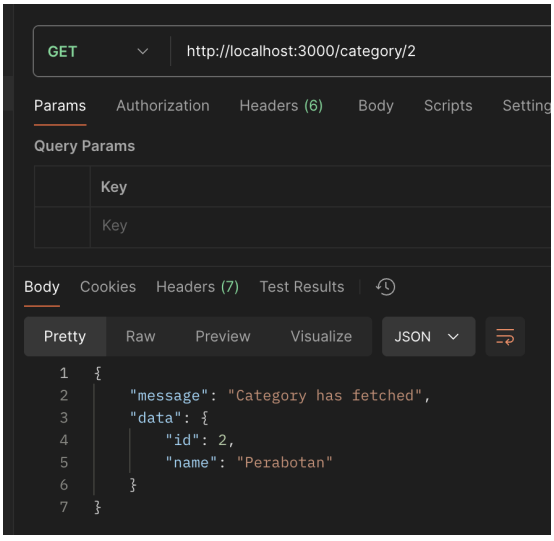
1. Source Code & Output Get Category

```
// Nomor 1 : Buatlah route GET yang mengembalikan daftar
// Anda bisa asumsikan data kategori disimpan dalam array
app.get("/category", (req: Request, res: Response) => {
    try {
        res.status(200).json({
            message: "Category has fetched",
            data: db,
        });
    } catch (error) {
        res.status(500).json({
            message: "error",
            data: "Something wrong happen",
        });
    }
});
```



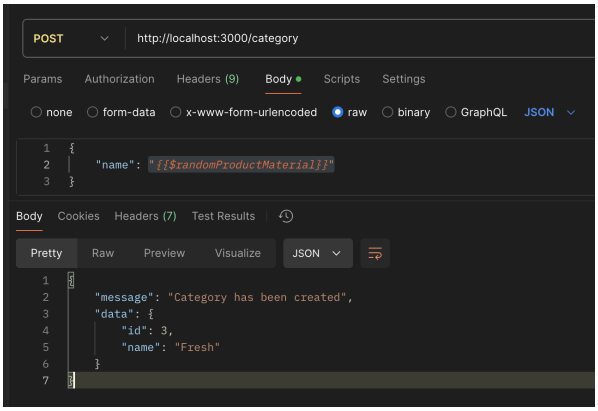
2. Source Code & Output Get Category By Id

```
// Nomor 2 : Buatlah route GET yang mengembalikan detail kategori berdasarkan
app.get("/category/:id", (req: Request, res: Response) => {
    try {
        const id = parseInt(req.params.id)
        const result = db.find(item => item.id === id)
        res.status(result ? 200 : 404).json({
            message: result ? "Category has fetched" : "Category not found",
            data: result,
        });
    } catch (error) {
        res.status(500).json({
            message: "error",
            data: "Something wrong happen",
        });
    }
});
```



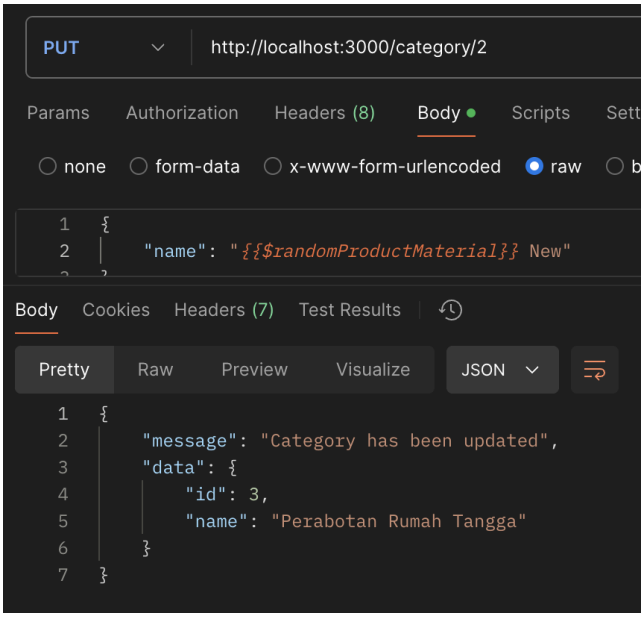
3. Source Code & Output Post Category

```
// Nomor 3 : Buatlah route POST yang menambahkan kategori baru ke array. Kategori baru haru
app.post("/category", (req: Request, res: Response) => {
    try {
        const name = req.body.name
        const check = db.find(item => item.name === name)
        let category = null

        if(!check){
            category = {
                id: db.length + 1,
                name: name
            }
            db.push(category)
        }

        res.status(!check ? 201 : 409).json({
            message: !check ? "Category has been created" : "Category name has been used",
            data: category,
        });
    } catch (error) {
        res.status(500).json({
            message: "error",
            data: "Something wrong happen",
        });
    }
});
```



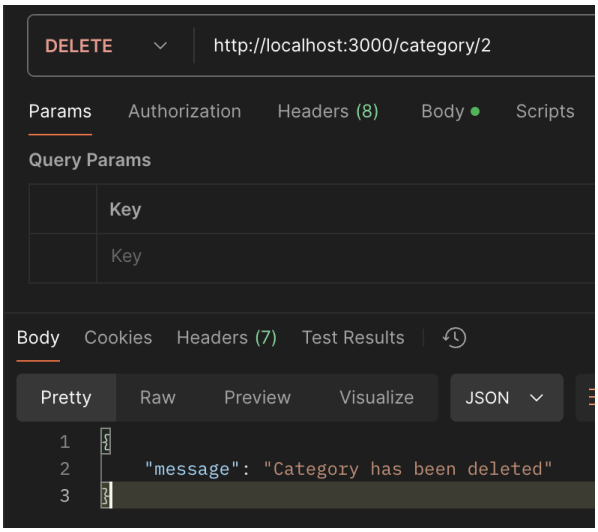
4. Source Code & Output Put Category By Id

```
// Nomor 4 : Buatlah route PUT yang memperbarui kategori berdas
app.put("/category/:id", (req: Request, res: Response) => {
    try {
        const id = parseInt(req.params.id)
        const name = req.body.name
        const check_name = db.find(item => item.name === name)
        const check_id = db.findIndex(item => item.id === id)
        let category = null

        if(check_name){
            res.status(409).json({
                message: "Category name has been used",
                data: category,
            });
        } else {
            if(check_id == -1){
                res.status(404).json({
                    message: "Category not found",
                    data: category,
                });
            } else {
                db[check_id].name = name

                res.status(200).json({
                    message: "Category has been updated",
                    data: db[check_id],
                });
            }
        }
    } catch (error) {
        res.status(500).json({
            message: "error",
            data: "Something wrong happen",
        });
    }
});
```



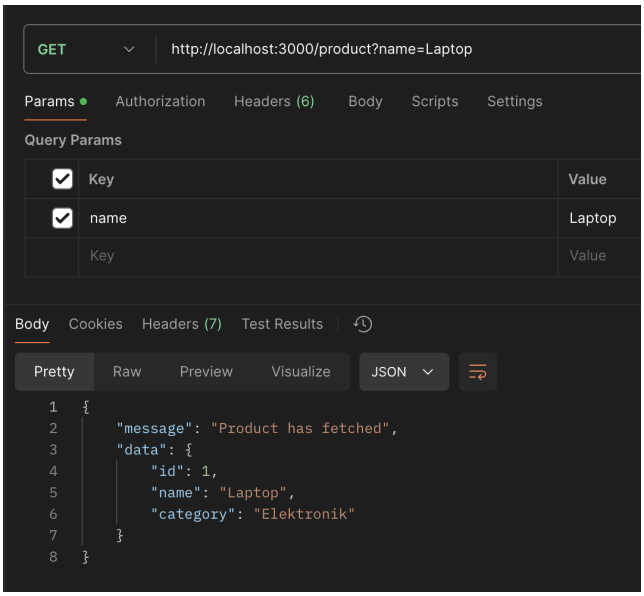
5. Source Code & Output Delete Category By Id

```
// Nomor 5 : Buatlah route DELETE yang menghapus kategori berdasarkan ID.
app.delete("/category/:id", (req: Request, res: Response) => {
    try {
        const id = parseInt(req.params.id)
        const check_id = db.findIndex(item => item.id === id)

        if(check_id == -1){
            res.status(404).json({
                message: "Category not found",
            });
        } else {
            db.splice(check_id, 1)

            res.status(200).json({
                message: "Category has been deleted",
            });
        }
    } catch (error) {
        res.status(500).json({
            message: "error",
            data: "Something wrong happen",
        });
    }
});
```



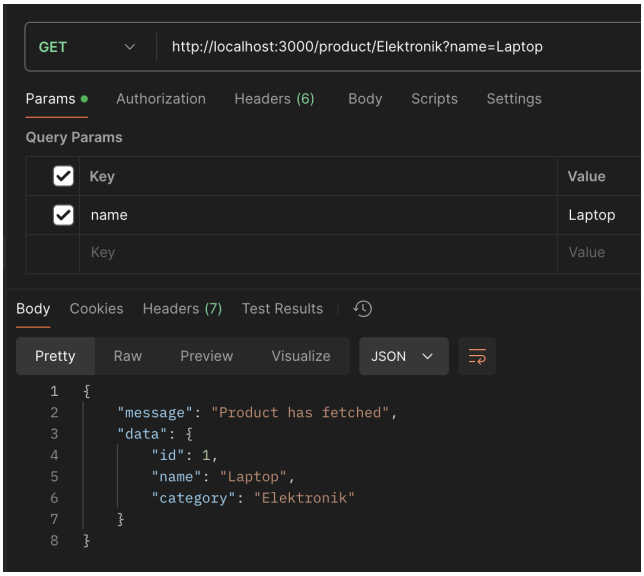
6. Source Code & Output Get Product By Name

```
// Nomor 6 : Buatlah route GET dengan query string untuk mencari produk ber
app.get("/product", (req: Request, res: Response) => {
    try {
        const name = req.query.name
        const result = db_product.find(item => item.name === name)
        res.status(result ? 200 : 404).json({
            message: result ? "Product has fetched" : "Product not found",
            data: result,
        });
    } catch (error) {
        res.status(500).json({
            message: "error",
            data: "Something wrong happen",
        });
    }
});
```



7. Source Code & Output Get Product By Name & Category

```
// Nomor 7 : Buatlah route GET dengan parameter dan query string untuk mendapatkan produk dalam ka
app.get("/product/:category", (req: Request, res: Response) => {
    try {
        const category = req.params.category
        const name = req.query.name
        const result = db_product.find(item => item.name === name && item.category === category)
        res.status(result ? 200 : 404).json({
            message: result ? "Product has fetched" : "Product not found",
            data: result,
        });
    } catch (error) {
        res.status(500).json({
            message: "error",
            data: "Something wrong happen",
        });
    }
});
```



Repository :

- *Github https://github.com/FlazeFy/sanbercode-nodejs-xpress/tree/main/tugas-routing-rest-api*
- *Postman Documentation https://documenter.getpostman.com/view/18882129/2sAY517fPm*

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*