

Laporan Praktikum WSE #5

Mata Kuliah : Web Service Engineering
Dosen Pengampu : Muhayat, M.IT
Praktikum : P5 - **Membangun RESTful CRUD API dengan Express**
Nama Mahasiswa : Husna Norgina
NIM : 230104040056
Kelas : TI23B
Link Repo Github : [husna-norgina/P5-CRUD-REST-230104040056](https://github.com/husna-norgina/P5-CRUD-REST-230104040056)
Tanggal Praktikum : 03-11-2025

A. Tujuan Praktikum

1. Mampu membangun RESTful API sederhana menggunakan Express.js.
2. Menerapkan operasi dasar CRUD (Create, Read, Update, Delete).
3. Menggunakan HTTP method dan status code secara benar.
4. Mengetahui struktur dasar server API modular dengan Node.js.

B. Lingkungan & Tools

- ✓ Node.js 18+ & npm
- ✓ Express.js
- ✓ VS Code
- ✓ Nodemon (dev dependency)
- ✓ Postman / Thunder Client
- ✓ Git & GitHub (opsional)

C. Arsitektur Singkat

- ✓ Client (Postman / Thunder Client) → mengirim HTTP request.
- ✓ API Server (Express) → menerima request dan menentukan route handler.
- ✓ Router (products.routes.js) → mendefinisikan endpoint CRUD.
- ✓ Data (products.data.js) → menyimpan data produk sementara (array).
- ✓ Response JSON → dikembalikan ke client dengan status code yang sesuai.

D. Langkah Implementasi (ringkas)

1. Membuat folder project bernama praktikum5_restful_crud/
2. Inisialisasi project menggunakan npm init -y
3. Menginstal Express dan Nodemon (npm install express & npm install nodemon --save-dev)
4. Membuat struktur folder src/routes/, src/data/, src/app.js
5. Menulis kode CRUD (GET, POST, PUT, DELETE) di products.routes.js
6. Menghubungkan route ke app.js menggunakan app.use('/api/products', productRoutes)

7. Menjalankan server dengan npm run dev dan menguji endpoint di Postman

E. Hasil & Bukti

Lampirkan Screenshot Hasil Uji Endpoint di Postman

- GET /api/products → Menampilkan seluruh produk (200 OK)

The screenshot shows the Postman interface with a collection named "P5-CRUD-REST-230104040056". A GET request is made to "http://localhost:3000/products". The response is a 200 OK status with a response time of 5 ms and a size of 372 B. The JSON body contains the following data:

```
1: {  
2:   "status": "success",  
3:   "data": [  
4:     {  
5:       "id": 1,  
6:       "name": "Pensil 2B",  
7:       "price": 5000,  
8:       "stock": 120  
9:     },  
10:    {  
11:      "id": 2,  
12:      "name": "Buku Tulis",  
13:      "price": 12000,  
14:      "stock": 50  
15:    }  
16:  ]  
17: }
```

- GET /api/products/1 → Menampilkan produk dengan ID tertentu (200 OK / 404 Not Found)

The screenshot shows the Postman interface with the same collection. A GET request is made to "http://localhost:3000/products/1". The response is a 200 OK status with a response time of 5 ms and a size of 315 B. The JSON body contains the following data:

```
1: {  
2:   "status": "success",  
3:   "data": [  
4:     {  
5:       "id": 1,  
6:       "name": "Pensil 2B",  
7:       "price": 5000,  
8:       "stock": 120  
9:     }  
10:  ]  
11: }
```

✓ POST /api/products → Menambah produk baru (201 Created)

The screenshot shows the Postman interface with a dark theme. On the left, there's a sidebar with 'Collections' (My Collection, P1-WSE-230104040056, P2-WSE-230104040056, P3-WSE-230104040056, P4-AGILE-230104040056, P5-CRUD-REST-230104040056), 'Environments', 'Flows', 'History', and a 'POST POST new product' section. The main area shows a request for 'P5-CRUD-REST-230104040056 / POST new product'. The method is 'POST' and the URL is 'http://localhost:3000/products'. The 'Body' tab is selected, showing raw JSON:

```
1 {  
2   "name": "Pulpen Pilot Biru",  
3   "price": 8000,  
4   "stock": 60  
5 }  
6
```

The response status is '201 Created' with a response time of '8 ms' and a size of '356 B'. The response body is:

```
1 {  
2   "status": "success",  
3   "message": "Product created",  
4   "data": {  
5     "id": 3,  
6     "name": "Pulpen Pilot Biru",  
7     "price": 8000,  
8     "stock": 60  
9   }  
10 }
```

✓ PUT /api/products/1 → Memperbarui data produk (200 OK / 404 Not Found)

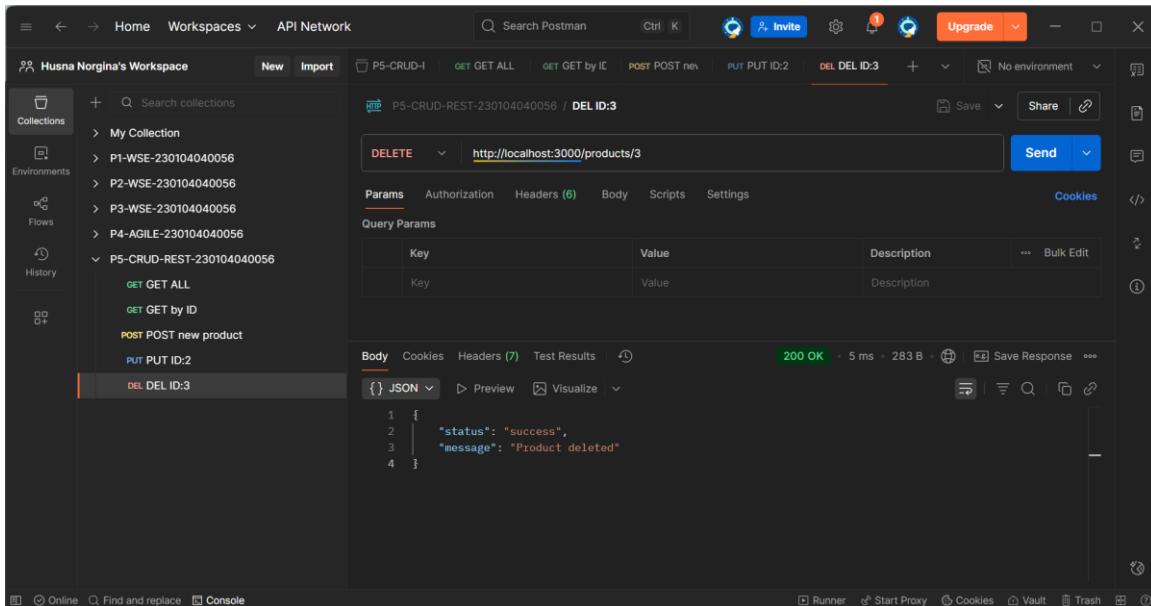
The screenshot shows the Postman interface with a dark theme. The sidebar is identical to the previous one. The main area shows a request for 'P5-CRUD-REST-230104040056 / PUT ID:2'. The method is 'PUT' and the URL is 'http://localhost:3000/products/2'. The 'Body' tab is selected, showing raw JSON:

```
1 {  
2   "name": "Buku Catatan Kuliah",  
3   "price": 15000,  
4   "stock": 75  
5 }  
6
```

The response status is '200 OK' with a response time of '7 ms' and a size of '354 B'. The response body is:

```
1 {  
2   "status": "success",  
3   "message": "Product updated",  
4   "data": {  
5     "id": 2,  
6     "name": "Buku Catatan Kuliah",  
7     "price": 15000,  
8     "stock": 75  
9   }  
10 }
```

- ✓ DELETE /api/products/1 → Menghapus produk (200 OK / 404 Not Found)



F. Analisis

API telah berfungsi sesuai prinsip CRUD dasar. Setiap endpoint berhasil merespons dengan status code yang sesuai. Namun, validasi input dan penanganan error masih sederhana dan akan dikembangkan lebih lanjut pada Praktikum 6 (RESTful API Best Practices).

G. Kesimpulan

Mahasiswa berhasil membangun RESTful CRUD API sederhana dengan Express.js. Hasil pengujian menunjukkan endpoint berfungsi dengan benar untuk operasi dasar Create, Read, Update, dan Delete.

H. Checklist Praktikum

- ✓ Struktur project Express sudah sesuai
- ✓ CRUD endpoint berjalan dengan benar
- ✓ Status code sesuai dengan operasi
- ✓ Response JSON terstruktur rapi
- ✓ Pengujian berhasil di Postman