

Laporan Praktikum WSE #5

Mata Kuliah : Web Service Engineering
Dosen Pengampu : Muhayat, M.IT
Praktikum : P5 – **Membangun RESTful CRUD**
Nama Mahasiswa : SITI MAGFIRATUN WARAHMAH
NIM : 230104040059
Kelas : TI23B
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 successful response for the GET /api/products endpoint. The response body is a JSON object containing a status key ('success') and a data key, which is an array of three product objects. Each product has an id, name, price, and stock.

```
1 {  
2   "status": "success",  
3   "data": [  
4     {  
5       "id": 1,  
6       "name": "Buku Catatan Kuliah",  
7       "price": 15000,  
8       "stock": 75  
9     },  
10    {  
11      "id": 2,  
12      "name": "Buku Tulis",  
13      "price": 12000,  
14      "stock": 50  
15    },  
16    {  
17      "id": 3,  
18      "name": "Pulpen Pilot Biru",  
19      "price": 8000,  
20      "stock": 60  
21  ]  
22}
```

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

The screenshot shows the Postman interface with a successful response for the GET /api/products/1 endpoint. The response body is a JSON object containing a status key ('success') and a data key, which is an array with one element. This element is a product object with an id, name, price, and stock.

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

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

The screenshot shows the Postman interface with a collection named "P5-CRUD-REST-230104040059". A POST request is made to `http://localhost:3000/products`. The request body contains the following JSON:

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

The response status is 201 Created, with a response time of 448 ms and a response 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/2 → Memperbarui data produk (200 OK / 404 Not Found)

The screenshot shows the Postman interface with a collection named "P5-CRUD-REST-230104040059". A PUT request is made to `http://localhost:3000/products/2`. The request body contains the following JSON:

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

The response status is 200 OK, with a response time of 10 ms and a response 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/3 → Menghapus produk (200 OK / 404 Not Found)

The screenshot shows the Postman interface with a collection named "P5-CRUD-REST-230104040059". A DELETE request is made to `http://localhost:3000/products/3`. The Headers tab shows a Content-Type header set to `application/json`. The Body tab displays a JSON response with two fields: `status` and `message`, both of which have the value `"Product deleted"`.

Key	Value	Description
Content-Type	application/json	

Key	Value	Description

```
1  {
2    "status": "success"
3    "message": "Product deleted"
4 }
```

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