

Laporan Praktikum WSE #6

Mata Kuliah : Web Service Engineering
Dosen Pengampu : Muhayat, M.IT
Praktikum : P6 – **RESTful API Best Practice (Express)**
Nama Mahasiswa : SITI MAGFIRATUN WARAHMAH
NIM : 230104040059
Kelas : TI23B
Tanggal Praktikum : 03-11-2025

A. Tujuan Praktikum

1. Memahami penerapan prinsip RESTful pada API Express.
2. Menggunakan HTTP Method dan Status Code secara tepat.
3. Mengimplementasikan 7 RESTful Principles dalam desain endpoint.
4. Menangani validasi input dan error secara terstandar.
5. Menyiapkan dokumentasi endpoint yang mudah dibaca dan diuji..

B. Lingkungan & Tools

- ✓ Node.js 18+ & npm
- ✓ Express.js
- ✓ VS Code / Postman / Thunder Client
- ✓ Nodemon (dev dependency)
- ✓ morgan → logging request
- ✓ Middleware: `validateProduct.js`, `errorHandler.js`

C. Arsitektur Singkat

- ✓ Client (Postman / Thunder Client / Frontend nanti) → mengirim HTTP request untuk CRUD + uji validasi + uji error.
- ✓ API Server (Express) → jadi pusat kontrol, menerima request, mem-parsing JSON, meneruskan ke router sesuai path /api/....
- ✓ Router (src/routes/products.routes.js) → mendefinisikan endpoint RESTful lengkap: GET, POST, PUT, PATCH, DELETE, plus /api/health.
- ✓ Controller / Handler → di dalam route, berisi logika ambil data, update, hapus, lalu membentuk response dalam format standar ({ success, message, data }).
- ✓ Middleware Validasi (src/middlewares/validateProduct.js) → dipasang khusus di POST dan PUT untuk menolak request yang tidak punya name atau price → balas 400 Bad Request.
- ✓ Middleware Error Global (src/middlewares/errorHandler.js) → menangkap error tak terduga (typo variabel, throw manual) dan mengembalikan 500 Server error tanpa menjatuhkan server.
- ✓ Data Layer (src/data/products.data.js) → masih pakai array in-memory sebagai sumber data sementara, tapi sudah dipisah agar nanti gampang diganti DB.
- ✓ Utility Response (src/utils/apiResponse.js) → (opsional) menyamakan bentuk response sukses dan error supaya API konsisten.
- ✓ Logging (morgan) → mencatat setiap request (method, path, status) untuk kebutuhan observability di praktikum berikutnya.

- ✓ Response JSON → semua endpoint mengembalikan JSON terstandar + status code yang sesuai (200, 201, 400, 404, 500).

D. Langkah Implementasi (ringkas)

1. Membuat struktur project Express di folder src/
2. Membuat file products.routes.js dengan endpoint CRUD + PATCH
3. Menambahkan middleware validateProduct untuk validasi POST dan PUT
4. Menambahkan middleware errorHandler untuk menangani error global
5. Menguji endpoint di Postman: GET, POST, PUT, PATCH, DELETE
6. Menguji validasi input dan simulasi error 500
7. Menulis dokumentasi endpoint di README.md

E. Hasil & Bukti

Lampirkan Screenshot Hasil Uji Endpoint di Postman

- POST /api/products (Berhasil tambah produk) → 201 Created

The screenshot shows the Postman interface with the following details:

- Collection:** Siti Magfiratun Warahmah's Works...
- Request:**
 - Method: POST
 - URL: <http://localhost:3000/api/products>
 - Body (raw JSON):

```
{
  "name": "Keyboard RGB",
  "price": 250000,
  "stock": 20
}
```
- Response:**
 - Status: 201 Created
 - Time: 377 ms
 - Size: 350 B
 - Content:

```
{
  "success": true,
  "message": "Product created",
  "data": {
    "id": 1762695685554,
    "name": "Keyboard RGB",
    "price": 250000
  }
}
```

PUT /api/products/1 (Berhasil update produk) → 200 OK

The screenshot shows the Postman application interface. On the left, the sidebar displays a collection named "Siti Magfiratur Warahmah's Works..." containing several items, including "P6-RESTFUL-BEST-230104040059" which is expanded to show "POST POST new product" and "PUT PUT update product". The main workspace shows a "PUT" request to "http://localhost:3000/api/products/1" with the following JSON body:

```
1 {
2   "name": "Laptop Pro X",
3   "price": 12000000,
4   "stock": 5
}
```

The response status is "200 OK" with a duration of "7 ms" and a size of "334 B". The response body is:

```
1 {
2   "success": true,
3   "message": "Product updated",
4   "data": {
5     "id": 1,
6     "name": "Laptop Pro X",
7     "price": 12000000
8   }
9 }
```

PATCH /api/products/1 (Ubah sebagian data) → 200 OK

The screenshot shows the Postman application interface. On the left, the sidebar displays a collection named "Siti Magfiratur Warahmah's Works..." containing several items, including "P6-RESTFUL-BEST-230104040059" which is expanded to show "POST POST new product" and "PUT PUT update product". The main workspace shows a "PATCH" request to "http://localhost:3000/api/products/1" with the following JSON body:

```
1 {
2   "stock": 12
3 }
```

The response status is "200 OK" with a duration of "15 ms" and a size of "356 B". The response body is:

```
1 {
2   "success": true,
3   "message": "Product partially updated",
4   "data": {
5     "id": 1,
6     "name": "laptop Pro X",
7     "price": 12000000,
8     "stock": 12
9   }
10 }
```

DELETE /api/products/1 (Produk terhapus) → 200 OK

The screenshot shows the Postman interface with a dark theme. On the left, the sidebar lists collections: P1-WSE-230104040059, P2-WSE-230104040059, P3-WSE-230104040059, P4-AGILE-KELOMPOK1-230104040059, P5-CRUD-REST-230104040059, and P6-RESTFUL-BEST-230104040059. The P6 collection is expanded, showing POST new product, PUT PUT update product, PATCH PATCH update sebagian, and GET DEL product id 1. The current request is "P6-RESTFUL-BEST-230104040059 / DEL product id 1". The method is set to DELETE, and the URL is http://localhost:3000/api/products/1. The response status is 200 OK, with a response time of 10 ms, a body size of 279 B, and a preview showing a JSON object with "success": true and "message": "Product deleted".

POST tanpa name → 400 Bad Request (Validation error)

The screenshot shows the Postman interface with a dark theme. The sidebar lists the same collections as the previous screenshot. The current request is "P6-RESTFUL-BEST-230104040059 / POST tanpa name". The method is set to POST, and the URL is http://localhost:3000/api/products. The response status is 400 Bad Request, with a response time of 54 ms, a body size of 348 B, and a preview showing a JSON object with "success": false, "message": "Validation error", and an "errors" array containing one entry: { "field": "name", "message": "Name is required"}. The "Body" tab shows the raw JSON input: { "price": 250000, "stock": 20 }.

⚠ POST tanpa price → 400 Bad Request (Validation error)

The screenshot shows the Postman interface with a collection named "Siti Magfiratur Warahmah's Works...". A POST request is made to `http://localhost:3000/api/products`. The request body is:

```
1 {
2   "name": "Keyboard RGB",
3   "stock": 20
4 }
```

The response status is **400 Bad Request** with a duration of 10 ms and a size of 350 B. The response body is:

```
1 {
2   "success": false,
3   "message": "Validation error",
4   "errors": [
5     {
6       "field": "price",
7       "message": "Price is required"
8     }
9   ]
10 }
```

⚠ POST tanpa name/price → 400 Bad Request (Validation error)

The screenshot shows the Postman interface with a collection named "Siti Magfiratur Warahmah's Works...". A POST request is made to `http://localhost:3000/api/products`. The request body is:

```
1 {
2   "stock": 20
3 }
```

The response status is **400 Bad Request** with a duration of 49 ms and a size of 396 B. The response body is:

```
1 {
2   "success": false,
3   "message": "Validation error",
4   "errors": [
5     {
6       "field": "name",
7       "message": "Name is required"
8     },
9     {
10       "field": "price",
11       "message": "Price is required"
12     }
13   ]
14 }
```

❖ Simulasi typo variabel → 500 Internal Server Error (Error handler aktif)

The screenshot shows the Postman interface with a collection named "P6-RESTFUL-BEST-230104040059". A GET request is made to `http://localhost:3000/api/products/crash/test`. The response status is 500 Internal Server Error, with a response body containing:

```
1 {  
2   "success": false,  
3   "message": "Server error"  
4 }
```

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

The screenshot shows the Postman interface with the same collection. A GET request is made to `http://localhost:3000/api/products`. The response status is 200 OK, with a response body containing:

```
1 {  
2   "success": true,  
3   "data": [  
4     {  
5       "id": 1,  
6       "name": "Laptop",  
7       "price": 8000000  
8     },  
9     {  
10      "id": 2,  
11      "name": "Headset",  
12      "price": 250000  
13    }  
14  ]  
15 }
```

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

The screenshot shows the Postman application interface. On the left, there's a sidebar with collections, environments, flows, and history. The main area shows a collection named "P6-RESTFUL-BEST-230104040059". A specific POST request named "GET Id 1" is selected. The request URL is "http://localhost:3000/api/products/1". The response tab shows a 200 OK status with a response body in JSON format:

```

1 {
2   "success": true,
3   "data": [
4     {
5       "id": 1,
6       "name": "Laptop",
7       "price": 8000000
8     }
9   ]

```

F. Analisis

API telah memenuhi 7 RESTful Principles, memiliki struktur modular, dan menampilkan status code yang konsisten. Middleware validasi berhasil mencegah input kosong, dan error handler menampilkan pesan 500 tanpa mematikan server.

G. Kesimpulan

Melalui praktikum ini, dapat dipahami bahwa RESTful API tidak hanya tentang CRUD, tetapi juga tentang penerapan prinsip desain yang konsisten, aman, dan mudah digunakan oleh client.

H. Checklist Praktikum

- ✓ Endpoint CRUD lengkap
- ✓ PATCH berfungsi
- ✓ Middleware validasi aktif
- ✓ Error handler berjalan
- ✓ Status code konsisten
- ✓ Dokumentasi README.md selesai