

UTS WSE

Mata Kuliah : Web Service Engineering
Dosen Pengampu : Muhayat, M.IT
Proyek : UTS
Nama Mahasiswa : Husna Norgina
NIM : 230104040056
Kelas : TI23B
Digit Akhir NIM : 6
Resource : events
Deskripsi : Data kegiatan kampus
Field Utama : title, date, location
Link Repo Github : <https://github.com/husna-norgina/UTS-WSE-230104040056>
Tanggal Praktikum : 10-11-2025

A. Soal Ujian

“Membangun RESTful API dengan CRUD Lengkap dan 7 RESTful Principles”

B. Deskripsi Singkat

Membangun RESTful API berbasis Express.js untuk satu resource baru, dengan operasi CRUD lengkap, validasi input, serta penerapan 7 Prinsip RESTful API seperti yang sudah dipelajari pada Praktikum 5 & 6.

API harus menampilkan data dalam format JSON, menggunakan status code yang tepat, serta memiliki endpoint dokumentasi (/api/info).

C. Tujuan

1. Mendesain endpoint RESTful untuk resource baru.
2. Mengimplementasikan CRUD lengkap dengan Express.js.
3. Menggunakan metode HTTP dan status code sesuai standar REST.
4. Menerapkan validasi dan error handling yang tepat.
5. Menghasilkan representasi data JSON yang konsisten.
6. Menyusun struktur folder modular dan mudah dibaca.
7. Menerapkan 7 RESTful Principles secara eksplisit.

D. Spesifikasi Endpoint

Method	Endpoint	Deskripsi	Status Code
GET	/api/<resource>	Ambil semua data	200
GET	/api/<resource>/:id	Ambil data berdasarkan id	200 / 404
POST	/api/<resource>	Tambah data baru	201 / 400
PUT	/api/<resource>/:id	Update data	200 / 400 / 404
DELETE	/api/<resource>/:id	Hapus data	204 / 404
GET	/api/info	Informasi service	200

E. Hasil & Bukti

Screenshot Hasil Uji Endpoint di Postman

- ✓ GET /api/events → Menampilkan seluruh events (200 OK)

The screenshot shows the Postman interface with the following details:

- Collection:** Husna Norgina's Workspace
- Request:** GET /api/events
- Response Status:** 200 OK
- Response Body (JSON):**

```
1 {
2     "success": true,
3     "data": [
4         {
5             "id": 1,
6             "title": "Seminar Teknologi",
7             "date": "10-11-2025",
8             "location": "Labkom 3"
9         },
10        {
11            "id": 2,
12            "title": "Workshop Bahasa Pemrograman",
13            "date": "15-11-2025",
14            "location": "Ruang Serbaguna"
15        }
16    ]
17}
```

- ✓ GET /api/events/1 → Menampilkan events dengan ID tertentu (200 OK)

The screenshot shows the Postman interface with the following details:

- Collection:** Husna Norgina's Workspace
- Request:** GET /api/events/1
- Response Status:** 200 OK
- Response Body (JSON):**

```
1 {
2     "success": true,
3     "data": {
4         "id": 1,
5         "title": "Seminar Teknologi",
6         "date": "10-11-2025",
7         "location": "Labkom 3"
8     }
9 }
```

✓ POST /api/events (Berhasil tambah events) → 201 Created

The screenshot shows the Postman interface with a successful API call. The collection 'Husna Norgina's Workspace' is selected. A new request 'POST POST new events' is being made to the URL `http://localhost:3000/api/events`. The request body contains the following JSON:

```
1 {
2   "title": "Seminar Keamanan Siber",
3   "date": "05-12-2025",
4   "location": "Labkom 1"
5 }
```

The response status is 201 Created, with a response time of 31 ms and a response size of 374 B. The response body is:

```
1 {
2   "success": true,
3   "message": "Event created",
4   "data": [
5     {
6       "id": 6,
7       "title": "Seminar Keamanan Siber",
8       "date": "05-12-2025",
9       "location": "Labkom 1"
10    }
11 }
```

⚠ POST /api/events (Gagal tambah events) → 400 Bad Request

The screenshot shows the Postman interface with an unsuccessful API call. The collection 'Husna Norgina's Workspace' is selected. A new request 'POST POST new events' is being made to the URL `http://localhost:3000/api/events`. The request body contains the following JSON:

```
1 {
2   "title": "Seminar Keamanan Siber",
3   "location": "Labkom 1"
4 }
```

The response status is 400 Bad Request, with a response time of 7 ms and a response size of 356 B. The response body is:

```
1 {
2   "status": "fail",
3   "message": "Validation error",
4   "errors": [
5     {
6       "field": "date",
7       "message": "Field 'date' wajib diisi"
8     }
9   ]
10 }
```

✓ PUT /api/events/1 (Berhasil update events) → 200 OK

The screenshot shows the Postman application interface. 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, P6-RESTFUL-BEST-230104040056, UTS-WSE-230104040056), 'Environments', 'Flows', and 'History'. The main area shows a collection named 'UTS-WSE-230104040056' with methods: 'GET GET ALL', 'GET GET by ID', 'POST POST new events', 'PUT PUT ID:1' (which is selected), 'DEL DEL ID:2', and 'GET GET info'. The 'PUT ID:1' request is being viewed, with the URL set to `http://localhost:3000/api/events/1`. The 'Body' tab contains the following JSON payload:

```
1 {
  "title": "Seminar Teknologi Terbaru",
  "date": "12-11-2025",
  "location": "Labkom 3"
}
```

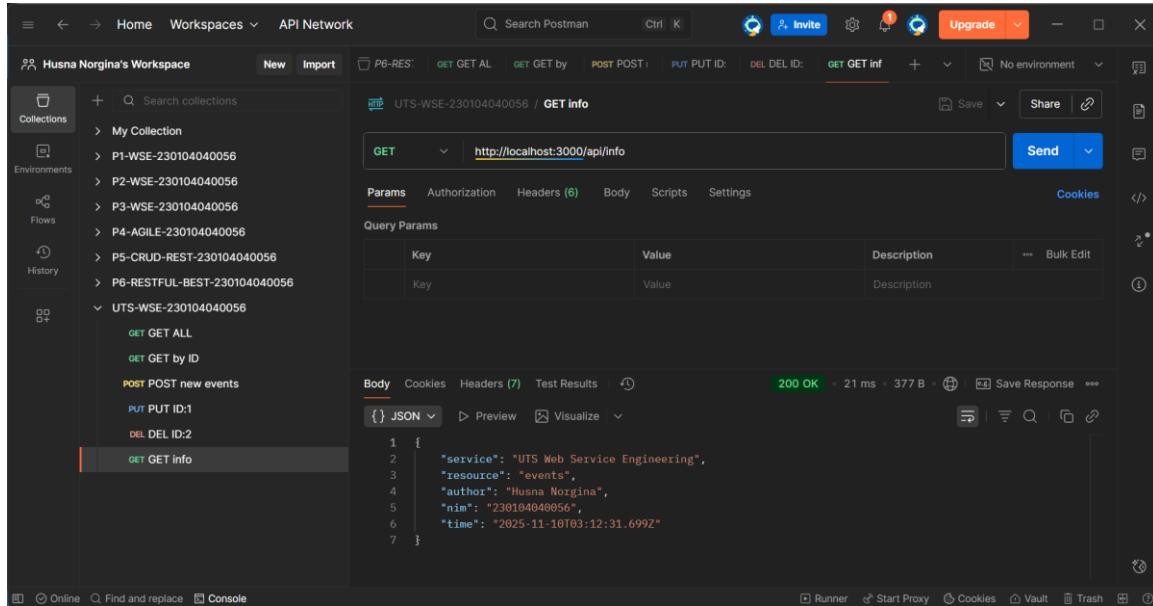
The response status is 200 OK, with a response time of 3 ms and a body size of 372 B. The response body is:

```
1 {
  "success": true,
  "message": "Event updated",
  "data": [
    {
      "id": 1,
      "title": "Seminar Teknologi Terbaru",
      "date": "12-11-2025",
      "location": "Labkom 3"
    }
  ]
}
```

✓ DELETE /api/events/2 (Events terhapus) → 204

The screenshot shows the Postman application interface, similar to the previous one but with a different collection. The sidebar shows 'Collections' (My Collection, P1-WSE-230104040056, P2-WSE-230104040056, P3-WSE-230104040056, P4-AGILE-230104040056, P5-CRUD-REST-230104040056, P6-RESTFUL-BEST-230104040056, UTS-WSE-230104040056), 'Environments', 'Flows', and 'History'. The main area shows the same collection structure as before. The 'PUT ID:1' method is still selected. The 'DELETE' request is being viewed, with the URL set to `http://localhost:3000/api/events/2`. The 'Headers' tab is selected, showing 'Content-Type: application/json'. The response status is 204 No Content, with a response time of 17 ms and a body size of 134 B. The response body is empty.

 GET info → http://localhost:3000/api/info → 200 OK



The screenshot shows the Postman application interface. On the left, the sidebar displays 'Husna Norgina's Workspace' with various collections, environments, flows, and history. In the center, a collection named 'P6-RES' is selected, containing several API endpoints: 'GET GET ALL', 'GET GET by ID', 'POST POST new events', 'PUT PUT ID:1', 'DEL DEL ID:2', and 'GET GET info'. The 'GET GET info' endpoint is highlighted with a red border. The main workspace shows a 'GET' request to 'http://localhost:3000/api/info'. The 'Headers' tab is selected, showing '(6)' entries. The 'Body' tab is selected, showing a JSON response with the following content:

```
1 {  
2   "service": "UTS Web Service Engineering",  
3   "resource": "events",  
4   "author": "Husna Norgina",  
5   "nim": "230104040056",  
6   "time": "2025-11-10T03:12:31.699Z"  
7 }
```

F. Kesimpulan

RESTful API untuk resource events telah berhasil dibuat menggunakan Express.js. Semua operasi CRUD (GET, POST, PUT, PATCH, DELETE) dapat berjalan dengan baik sesuai spesifikasi, dilengkapi validasi input dan penanganan error. Status code HTTP yang digunakan konsisten dengan standar REST, dan semua response disajikan dalam format JSON yang rapi. Endpoint /api/info juga telah tersedia sebagai metadata service. Dengan demikian, semua 7 RESTful Principles telah diterapkan secara lengkap pada proyek ini.