

# **Ujian Tengah Semester (UTS) Semester Ganjil 20251**

Nama Mahasiswa : SITI MAGFIRATUN WARAHMAH  
NIM : 230104040059  
Kelas : TI23B  
Hari/Tanggal : Senin, 10 November 2025  
Program Studi : Teknologi Informasi  
Mata Kuliah : Web Service Engineering  
Dosen Pengampu : Muhayat, M.IT



## **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. Ketentuan Umum**

1. Gunakan Node.js + Express.js.
2. Tidak menggunakan database — cukup data dummy di file .js / .json.
3. Gunakan struktur folder standar:

```
src/
  └── app.js
  └── routes/
    └── <resource>.routes.js
  └── controllers/
    └── <resource>.controller.js
  └── data/
    └── <resource>.data.js
  └── README.md
```

4. Semua response berbentuk JSON.
5. Gunakan port default 3000.

6. Sertakan endpoint /api/info untuk identitas API.
7. Kode wajib dijalankan dengan npm run dev.

#### E. 7 Prinsip RESTful yang Wajib Diterapkan

No	Prinsip	Implementasi di Soal
1	Resource-Oriented URI	Gunakan kata benda jamak, contoh /api/books
2	Proper HTTP Methods	GET, POST, PUT, DELETE
3	Stateless Communication	Tidak ada session atau state di server
4	Consistent Status Codes	200, 201, 204, 400, 404, 500
5	JSON Representation	Response rapi dan seragam
6	Validation & Error Handling	Periksa field wajib, kirim error 400
7	Discoverability	Tambahkan /api/info sebagai metadata service

#### F. Penentuan Resource Berdasarkan Digit Akhir NIM

Digit Akhir NIM	Resource	Deskripsi	Field Utama (wajib ada)
9	movies	Data film	title, genre, year

#### G. Spesifikasi Endpoint yang Wajib Ada

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

#### H. Contoh Validasi (Wajib Ada)

Field	Syarat	Respon Error
title	Tidak boleh kosong	{ "status": "fail", "message": "Field 'title' wajib diisi" }
author	Tidak boleh kosong	{ "status": "fail", "message": "Field 'author' wajib diisi" }

## I. Screenshot Hasil Uji di Postman

GET /api/info (Berhasil ambil info API) → 200 OK

The screenshot shows the Postman interface with a dark theme. On the left, the sidebar lists collections, environments, flows, and history. The main area shows a collection named "ti Magfiratur Warahmah's Works...". A specific endpoint, "GET /api/info", is selected. The URL is set to "http://localhost:3000/api/info". The "Body" tab is selected, showing a JSON response with the following content:

```
1  {
2      "service": "UTS WSE API - Movies",
3      "author": "Siti Magfiratur Warahmah",
4      "nim": "230104040059",
5      "description": "API untuk mengelola data film (movies) "
6 }
```

The status bar at the bottom indicates a 200 OK response with 7 ms duration and 383 B size.

GET /api/movies (Berhasil ambil semua data film) → 200 OK

The screenshot shows the Postman interface with a dark theme. The sidebar and collection structure are identical to the previous screenshot. The endpoint "GET /api/movies" is selected. The URL is set to "http://localhost:3000/api/movies". The "Body" tab is selected, showing a JSON response with the following content:

```
1  {
2      "status": "success",
3      "message": "Data semua film berhasil diambil",
4      "data": [
5          {
6              "id": 1,
7              "title": "Interstellar",
8              "genre": "Sci-Fi",
9              "year": 2014
10         },
11         {
12             "id": 2,
13             "title": "The Dark Knight",
14             "genre": "Action",
15             "year": 2008
16         },
17         {
18             "id": 3,
19             "title": "Parasite",
20             "genre": "Thriller",
21             "year": 2019
22         }
23     ]
24 }
```

The status bar at the bottom indicates a 200 OK response with 11 ms duration and 495 B size.

GET /api/movies/3 (Berhasil ambil data film by ID) → 200 OK

The screenshot shows the Postman interface with a collection named "Magfiratur Warahmah's Works...". A GET request is made to `http://localhost:3000/api/movies/3`. The response status is 200 OK, and the response body is:

```
1 {
2   "status": "success",
3   "message": "Data film berhasil diambil",
4   "data": [
5     {
6       "id": 3,
7       "title": "Parasite",
8       "genre": "Thriller",
9       "year": 2019
10    }
11 }
```

POST /api/movies (Berhasil tambah data film) → 201 Created

The screenshot shows the Postman interface with the same collection. A POST request is made to `http://localhost:3000/api/movies`. The response status is 201 Created, and the response body is:

```
1 {
2   "status": "success",
3   "message": "Data film berhasil dibuat",
4   "data": [
5     {
6       "id": 4,
7       "title": "Oppenheimer",
8       "genre": "Biography",
9       "year": 2023
10    }
11 }
```

⚠ POST /api/movies (Gagal validasi data) → 400 Bad Request

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

```
1 {
2   "genre": "Biography",
3   "year": 2023
4 }
```

The response status is 400 Bad Request, with the message: "status": "fail", "message": "Field 'title' dan 'genre' wajib diisi".

✓ PUT /api/movies/1 (Berhasil update data film) → 200 OK

The screenshot shows the Postman interface with the same collection. A PUT request is made to `http://localhost:3000/api/movies/1`. The request body is:

```
1 {
2   "title": "Interstellar (Director's Cut)",
3   "genre": "Sci-Fi",
4   "year": 2014
5 }
```

The response status is 200 OK, with the message: "status": "success", "message": "Data film berhasil diperbarui", "data": { "id": 1, "title": "Interstellar (Director's Cut)", "genre": "Sci-Fi", "year": 2014 }.

- DELETE /api/movies/2** (Berhasil hapus data film) → 204 No Content

The screenshot shows the Postman application interface. On the left, there's a sidebar with 'Collections' (containing UTS-WSE-MOVIE-230104040059), 'Environments' (with one entry), 'Flows', and 'History'. The main workspace shows a collection named 'UTS-WSE-MOVIE-230104040059'. A specific request is selected: 'DEL DELETE id:2 movie'. The URL is set to 'http://localhost:3000/api/movies/2'. The 'Params' tab is active, showing a single parameter 'Key' with value 'Value'. The 'Body' tab shows a simple JSON response: '1'. The 'Test Results' tab indicates a '204 No Content' response with a duration of 21 ms and a size of 134 B.

**Checklist Mahasiswa**

No	Komponen	Status
1	CRUD lengkap berjalan	<input type="checkbox"/>
2	Validasi input & error handling	<input type="checkbox"/>
3	Status code konsisten	<input type="checkbox"/>
4	Response JSON rapi	<input type="checkbox"/>
5	Endpoint /api/info aktif	<input type="checkbox"/>
6	Struktur folder sesuai template	<input type="checkbox"/>
7	Screenshot hasil uji CRUD	<input type="checkbox"/>
8	Penerapan 7 RESTful Principles	<input type="checkbox"/>