

Materi praktek minggu 7: REST API Sederhana (CRUD) menggunakan PHP

Struktur Project

```
rest-api/  
    ├── db.php  
    ├── api.php  
    └── .htaccess (jika pakai Apache)
```

db.php

```
<?php  
$host = "localhost";  
$user = "root";  
$pass = "";  
$db = "kampus";  
  
$koneksi = new mysqli($host, $user, $pass, $db);  
  
if ($koneksi->connect_error) {  
    die("Koneksi gagal: " . $koneksi->connect_error);  
}  
?>
```

api.php

```
<?php  
header("Content-Type: application/json");  
require 'db.php';
```

```
$method = $_SERVER['REQUEST_METHOD'];

switch ($method) {
    case 'GET':
        $sql = "SELECT * FROM mahasiswa";
        $result = $koneksi->query($sql);
        $data = [];
        while ($row = $result->fetch_assoc()) {
            $data[] = $row;
        }
        echo json_encode($data);
        break;
    case 'POST':
        $input = json_decode(file_get_contents("php://input"), true);
        $nama = $input['nama'];
        $nim = $input['nim'];
        $jurusan = $input['jurusan'];
        $sql = "INSERT INTO mahasiswa (nim, nama, jurusan) VALUES ('$nim', '$nama', '$jurusan')";
        echo json_encode(['status' => $koneksi->query($sql)]);
        break;
    case 'PUT':
        $input = json_decode(file_get_contents("php://input"), true);
        $id = $input['id'];
        $nama = $input['nama'];
        $jurusan = $input['jurusan'];
        $sql = "UPDATE mahasiswa SET nama='$nama', jurusan='$jurusan' WHERE id=$id";
        echo json_encode(['status' => $koneksi->query($sql)]);
        break;
    case 'DELETE':
        $input = json_decode(file_get_contents("php://input"), true);
        $id = $input['id'];
        $sql = "DELETE FROM mahasiswa WHERE id=$id";
```

```
echo json_encode(['status' => $koneksi->query($sql)]);
break;
default:
http_response_code(405);
echo json_encode(['error' => 'Method Not Allowed']);
break;
}
?>
```

📌 Testing dengan Postman

1. GET

- URL: <http://localhost/rest-api/api.php> (sesuaikan dengan URL punyanya masing-masing)
- Method: GET

2. POST

- URL: <http://localhost/rest-api/api.php>
- Method: POST
- Body → Raw → JSON:

```
{
    "nim": "12345",
    "nama": "Dewi",
    "jurusan": "Informatika"
}
```

3. PUT

- URL: <http://localhost/rest-api/api.php>
- Method: PUT
- Body → Raw → JSON:

```
{
```

```
"id": 1,  
"nama": "Dewi Update",  
"jurusan": "SI"  
}
```

4. DELETE

- URL: <http://localhost/rest-api/api.php>
- Method: DELETE
- Body → Raw → JSON:

```
{  
    "id": 1  
}
```

🔧 .htaccess (optional untuk Apache clean URL)

```
RewriteEngine On  
  
RewriteCond %{REQUEST_FILENAME} !f  
  
RewriteRule ^(.*)$ api.php [QSA,L]
```

Langkah-Langkah Testing REST API Menggunakan Postman

1. Install Postman

Kalau belum punya, unduh dari:

 <https://www.postman.com/downloads>

Setelah selesai install, buka aplikasi Postman.

2. Persiapan Endpoint API

Pastikan file api.php sudah bisa diakses lewat browser. Misalnya:

<http://localhost/rest-api/api.php>

Kalau dibuka dan keluar data JSON, berarti ready untuk dites di Postman.

3. Test Method: GET

Langkah:

1. Buka Postman
 2. Pilih method: GET
 3. Masukkan URL:
<http://localhost/rest-api/api.php>
 4. Klik tombol **Send**
 5. Lihat hasilnya di tab **Body (bawah)** → JSON
-

4. Test Method: POST (menambah data)

Langkah:

1. Pilih method: POST
2. URL tetap:
<http://localhost/rest-api/api.php>
3. Klik tab **Body (atas)**
4. Pilih opsi **raw** dan ubah tipe ke **JSON**
5. Masukkan data JSON seperti ini:

```
{  
    "nim": "12345",  
    "nama": "Dewi",  
    "jurusan": "Informatika"  
}
```

6. Klik tombol **Send**
 7. Lihat hasilnya di tab **Body (bawah) → JSON**
 8. Hasil akan muncul: {"status":true} jika berhasil
-

📌 5. Test Method: PUT (mengedit data)

Langkah:

1. Pilih method: PUT
2. URL tetap:
<http://localhost/rest-api/api.php>
3. Tab **Body (atas) → raw → JSON**
4. Masukkan data (misalnya update data dengan id 1):

```
{  
    "id": 1,  
    "nama": "Dewi Update",  
    "jurusan": "Sistem Informasi"
```

```
}
```

5. Klik **Send**
 6. Lihat hasilnya di tab **Body (bawah)** → JSON
 7. Lihat respon {"status":true} jika sukses
-

📌 6. Test Method: DELETE (menghapus data)

Langkah:

1. Pilih method: DELETE
 2. URL:
<http://localhost/rest-api/api.php>
 3. Tab **Body (atas)** → **raw** → **JSON**
 4. Masukkan:

```
{
  "id": 1
}
```
 5. Klik **Send**
 6. Respon sukses akan: {"status":true}
-

🔍 Tips Tambahan

- Kalau muncul error 405 / 500 di Postman, periksa kembali struktur JSON dan method-nya.
- Kita bisa aktifkan tab **Console** (Ctrl + Alt + C) di Postman untuk lihat log request.
- Saat menggunakan postman pastikan xampp sudah aktif

Tutorial Tes REST API dengan Postman

1. Buka Postman

Pastikan aplikasi Postman sudah terbuka dan bisa diakses.

2. Masukkan URL Endpoint

Contoh:

<http://localhost/project-api/api.php>

Atau untuk GET satu data (dengan parameter):

<http://localhost/project-api/api.php?id=1>

3. Pilih Method

Di sebelah kiri URL, pilih salah satu metode:

- GET → Untuk menampilkan data
- POST → Untuk menambah data
- PUT → Untuk mengedit data
- DELETE → Untuk menghapus data

4. Buka Tab “Body” (atas)

- Pilih **raw**
- Pilih format: **JSON** (klik dropdown di kanan)

5. Masukkan Data (jika POST/PUT)

Misalnya untuk **POST**:

```
{  
    "nim": "12345678",  
    "nama": "Rina Saputri",  
    "jurusan": "Sistem Informasi"  
}
```

6. Klik Send

Klik tombol **Send** di pojok kanan atas.

Dimana Melihat Hasil Response?

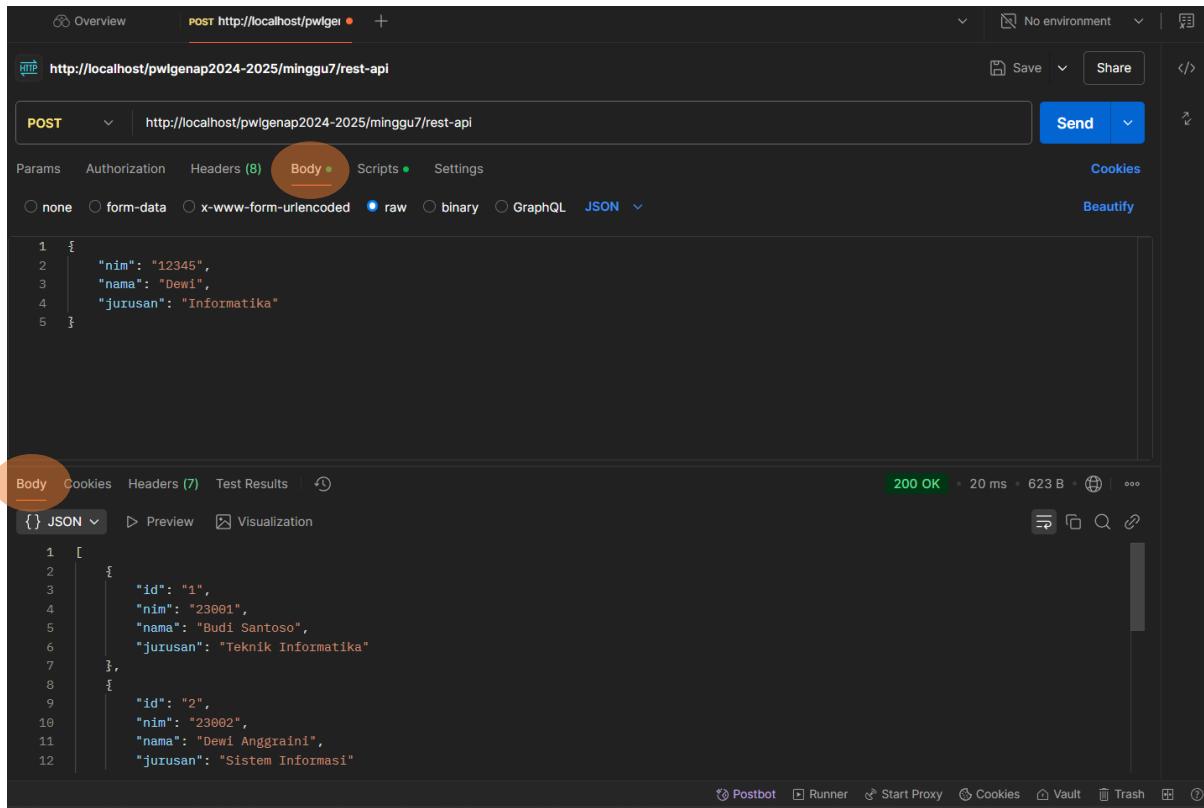
Di tab body bagian bawah, ada beberapa pilihan tampilan hasil:

- **JSON**: menampilkan JSON dengan rapi (ada beberapa pilihan: xml, html, javascript, dsb)
- **Preview**: tampilan seperti di browser
- **Visualize**: melihat hasil dalam bentuk tabel atau grafik

Tips Tambahan

- Kalau ingin uji **DELETE** atau **PUT**, kadang kita perlu menambahkan plugin eksternal untuk bisa uji di browser, tapi dengan **Postman**, semua metode HTTP bisa langsung diuji.
- Pastikan header Content-Type: application/json aktif (biasanya otomatis saat kita pakai JSON di Body).
- Cek juga tab **Headers** kalau perlu debug masalah.

Penjelasan Dua Tab Body di Postman



✓ 1. Tab Body (atas) – bagian pengiriman data (Request Body)

📍 Lokasi: Di bagian atas, setelah tab "Headers"

📌 Fungsinya:

Tab ini digunakan untuk **mengirim data** ke server, biasanya saat kita menggunakan method **POST** atau **PUT**.

◆ Kita bisa pilih formatnya:

- raw → digunakan untuk kirim data **JSON**, seperti di gambar
- form-data → biasanya untuk upload file
- x-www-form-urlencoded → seperti form HTML biasa

💡 **Contoh pada gambar (POST):**

{

```
"nim": "12345",
"nama": "Dewi",
"jurusan": "Informatika"
}
```

Ini adalah **data request** yang dikirim ke server via endpoint POST
<http://localhost/.../rest-api>.

✓ 2. Tab Body (bawah) – bagian hasil respon dari server (Response Body)

💡 Lokasi: Di bagian bawah, sebelum tab "Cookies", "Headers", dan "Test Results"

📌 Fungsinya:

Menampilkan **balasan dari server** setelah kita klik tombol Send. Isinya bisa bermacam-macam: JSON, HTML, teks biasa, atau error message.

💡 Contoh yang muncul pada gambar:

```
{
  "status": true
}
```

■ Jadi, kalau kita ingin melihat hasilnya pakai tab Body yang bawah, karena:

- Itu adalah **response dari server**.
- Kita bisa tahu apakah request kita berhasil atau gagal.

📌 Kesimpulan:

Tab	Posisi	Fungsi	Digunakan Saat
Body (atas)	Atas	Untuk mengirim data ke server (Request Body)	POST, PUT, PATCH
Body (bawah)	Bawah	Untuk melihat balasan dari server (Response Body)	Semua request (GET, POST, dst)

Template Uji CRUD Postman

Berikut adalah **template dokumentasi uji CRUD lengkap di Postman** untuk REST API dengan PHP & JSON. Template ini untuk pegangan mahasiswa agar bisa:

- Mengerti tiap jenis request (Create, Read, Update, Delete)
- Tahu cara mengetesnya di Postman
- Menyertakan contoh URL, body, dan response

Dokumentasi Uji CRUD REST API (Postman)

Base URL

<http://localhost/nama-folder/rest-api.php>

Ganti nama-folder dengan nama folder lokal masing-masing.

1. GET (Read All Data)

Tujuan:

Menampilkan semua data mahasiswa

Konfigurasi di Postman:

- Method: GET
- URL:

<http://localhost/nama-folder/rest-api.php>

Response (Contoh):

```
[  
 {  
   "id": "1",  
   "nim": "23001",  
   "nama": "Budi Santoso",
```

```
        "jurusan": "Teknik Informatika"  
    },  
    {  
        "id": "2",  
        "nim": "23002",  
        "nama": "Dewi Lestari",  
        "jurusan": "Sistem Informasi"  
    }  
]
```

🟡 2. POST (Create Data)

✓ Tujuan:

Menambahkan data mahasiswa baru

🔧 Konfigurasi di Postman:

- Method: POST
- URL:
<http://localhost/nama-folder/rest-api.php>
- Pilih tab Body (atas)
- Pilih raw → pilih JSON
- Masukkan JSON seperti ini:

```
{  
    "nim": "12345",  
    "nama": "Putri Ayu",  
    "jurusan": "Informatika"  
}
```

🌐 Response (Contoh):

```
{  
    "message": "Data berhasil ditambahkan"
```

```
}
```

● 3. PUT (Update Data)

✓ Tujuan:

Mengubah data mahasiswa berdasarkan ID

🔧 Konfigurasi di Postman:

- Method: PUT

- URL:

`http://localhost/nama-folder/rest-api.php`

- Pilih tab Body (atas)
- Pilih raw → pilih JSON
- Masukkan data JSON:

```
{
  "id": "2",
  "nim": "23002",
  "nama": "Dewi Lestari Update",
  "jurusan": "Teknik Komputer"
}
```

✉ Response (Contoh):

```
{
  "message": "Data berhasil diupdate"
}
```

● 4. DELETE (Delete Data)

✓ Tujuan:

Menghapus data berdasarkan ID

Konfigurasi di Postman:

- Method: DELETE
- URL:
`http://localhost/nama-folder/rest-api.php`

- Pilih tab Body (atas)
- Pilih raw → pilih JSON
- Masukkan data ID:

```
{  
    "id": "2"  
}
```

Response (Contoh):

```
{  
    "message": "Data berhasil dihapus"  
}
```

Tips Tambahan:

- Pastikan XAMPP aktif dan folder aplikasi API berada di htdocs.
- Format data yang dikirim dan diterima selalu dalam bentuk **JSON**.
- Selalu pilih Body (atas) > raw > JSON di Postman untuk POST, PUT, DELETE.

Apa Itu Endpoint?

Definisi Sederhana:

Endpoint adalah URL (alamat) yang digunakan untuk mengakses data atau layanan tertentu dari sebuah API.

Bisa dibilang, endpoint itu seperti pintu masuk ke suatu data atau fungsi di server.



Analogi Gampang:

Bayangkan kita sedang masuk ke rumah teman kita:

- Alamat rumahnya: Jalan API No. 1
- Tapi kita nggak langsung masuk semua ruangan kan?
- Kita bisa pilih:
 - masuk Dapur → untuk ambil makanan
 - masuk Ruang belajar → untuk baca buku
 - masuk Kamar → untuk tidur

Nah, dalam API:

- `http://example.com/mahasiswa` adalah endpoint untuk data mahasiswa
 - `http://example.com/dosen` adalah endpoint untuk data dosen
-



Contoh Endpoint & HTTP Method:

Endpoint	HTTP Method	Kegunaan
/mahasiswa	GET	Ambil semua data mahasiswa
/mahasiswa/5	GET	Ambil data mahasiswa dengan ID 5

/mahasiswa	POST	Tambah data mahasiswa baru
/mahasiswa/5	PUT	Edit data mahasiswa dengan ID 5
/mahasiswa/5	DELETE	Hapus data mahasiswa dengan ID 5

Jadi, endpoint adalah **alamat spesifik** yang digunakan client (misalnya Postman, browser, frontend app) untuk **berkomunikasi dengan API**.

📌 **Kesimpulan:**

Endpoint = Alamat API + Resource Contoh: <http://localhost/api/mahasiswa>

Di situlah kita bisa GET, POST, PUT, atau DELETE data mahasiswa 

Latihan Praktikum

💡 **Latihan 1: Buat REST API CRUD untuk Tabel mahasiswa**

Langkah-langkah:

1. Buat database kampus dan tabel mahasiswa (gunakan SQL yang sudah diberikan).
 2. Buat file PHP rest-api.php yang mendukung operasi:
 - GET: Menampilkan semua data mahasiswa
 - POST: Menambahkan data mahasiswa
 - PUT: Mengubah data mahasiswa berdasarkan id
 - DELETE: Menghapus data mahasiswa berdasarkan id
 3. Uji semua metode tersebut dengan **Postman**.
-

💡 **Latihan 2: Buat File Dokumentasi API**

Buat file README.md atau file Word yang berisi:

- Penjelasan singkat API yang dibuat
- Contoh request dan response (format JSON)
- Contoh URL API:
 - GET `http://localhost/project/rest-api.php`
 - POST `http://localhost/project/rest-api.php`
 - dst.

Tugas Minggu 7

Deskripsi:

Buat **REST API sederhana untuk data buku di perpustakaan**, dengan operasi CRUD. Setiap buku memiliki:

1. id
2. judul
3. penulis
4. tahun_terbit

Target:

- Mahasiswa membuat API yang bisa diakses via Postman
 - Mahasiswa menyertakan dokumentasi sederhana
-

Keluaran yang Dikumpulkan:

1. File project (PHP + SQL)
 2. Screenshot hasil uji Postman untuk setiap metode (GET, POST, PUT, DELETE)
 3. Dokumentasi singkat (boleh PDF, atau Word)
-

Tips:

- Gunakan struktur kode yang sama seperti contoh mahasiswa, hanya ganti data.
- Cek error dengan membaca response JSON dan Network tab jika ada.

Template Laporan dan Dokumentasi REST API (digunakan saat membuat tugas)

Laporan & Dokumentasi REST API

1. Identitas Mahasiswa

- **Nama:**
 - **NIM:**
 - **Kelompok:**
 - **Tanggal Pengumpulan:**
-

2. Deskripsi API

Tuliskan penjelasan singkat tentang API yang dibuat:

Contoh:

REST API ini digunakan untuk mengelola data buku perpustakaan. API mendukung operasi CRUD (Create, Read, Update, Delete) menggunakan metode HTTP: GET, POST, PUT, dan DELETE. Format pertukaran data menggunakan JSON.

3. Base URL

http://localhost/nama_folder_project/rest-api.php

(Ganti sesuai nama folder project masing-masing di XAMPP/htdocs)

4. Endpoint dan HTTP Methods

Method	Endpoint	Deskripsi
GET	/rest-api.php	Ambil semua data
GET	/rest-api.php?id=1	Ambil data berdasarkan id
POST	/rest-api.php	Tambah data baru
PUT	/rest-api.php?id=1	Ubah data berdasarkan id
DELETE	/rest-api.php?id=1	Hapus data berdasarkan id

5. Request dan Response

Contoh:

- ◆ GET (All Data)

Request:

```
GET http://localhost/rest-api.php
```

Response:

```
[  
 {  
   "id": 1,  
   "nama": "Andi",  
   "nim": "12345678",  
   "jurusan": "Informatika"  
,  
 {  
   "id": 2,  
   "nama": "Budi",  
   "nim": "87654321",  
   "jurusan": "Sistem Informasi"  
}  
]
```

- ◆ **POST (Tambah Data)**

Request:

POST http://localhost/rest-api.php

Body (raw JSON):

```
{  
    "nama": "Citra",  
    "nim": "13579024",  
    "jurusan": "Teknik Komputer"  
}
```

Response:

```
{  
    "status": "success",  
    "message": "Data berhasil ditambahkan"  
}
```

- ◆ **PUT (Update Data)**

Request:

PUT http://localhost/rest-api.php?id=1

Body:

```
{  
    "nama": "Andi Update",  
    "nim": "12345678",  
    "jurusan": "Data Science"  
}
```

Response:

```
{  
    "status": "success",
```

```
        "message": "Data berhasil diubah"  
    }  


---


```

◆ **DELETE (Hapus Data)**

Request:

DELETE http://localhost/rest-api.php?id=1

Response:

```
{  
    "status": "success",  
    "message": "Data berhasil dihapus"  
}  


---


```

Buat request dan response untuk program API nya masing-masing

6. **Screenshot Postman**

Lampirkan:

- Screenshot hasil GET
 - Screenshot POST (beserta isi Body)
 - Screenshot PUT
 - Screenshot DELETE
-

7. **Kesimpulan**

Tuliskan kesimpulan singkat:

- Apa yang dipelajari?
- Apa kesulitan yang dihadapi?
- Bagaimana REST API bisa digunakan dalam aplikasi nyata?

Tujuan Pemberian Materi Praktik REST API dan JSON

Materi praktik yang diberikan pada minggu ke-7 disusun untuk **mendekatkan konsep REST API dan JSON ke dunia nyata** yang langsung bisa dirasakan mahasiswa. Tujuannya bukan cuma supaya mahasiswa bisa “menjalankan kode”, tapi **mengerti apa itu API, bagaimana bekerja, dan kenapa penting di aplikasi web modern.**

Tujuan Praktik Secara Khusus

Komponen Praktik	Tujuan	Outcome untuk Mahasiswa
Membuat REST API CRUD dengan PHP & JSON	Menunjukkan bagaimana server menyediakan data dan menerima data dari client	Mahasiswa paham struktur endpoint API, cara kerja GET/POST/PUT/DELETE
Menggunakan JSON sebagai format data	Membiasakan penggunaan JSON dalam pertukaran data	Mahasiswa memahami format JSON, kelebihannya, dan penggunaannya
Testing dengan Postman	Melatih menguji API tanpa frontend, dan mengenali response/request API	Mahasiswa terbiasa membaca response server, status code, dan debugging
Melihat alur komunikasi antara client (Postman) dan server (PHP API)	Membangun pemahaman bahwa API adalah “jembatan” frontend dan backend	Mahasiswa mengerti konsep client-server dan pertukaran data melalui HTTP

Praktek API & REST API ini sangat krusial karena:

- **Teori REST API** itu abstrak (stateless, resource, method HTTP)
- Praktik **membumikan teori**: agar mahasiswa bisa “merasakan langsung” peran API

Agar lebih mudah memahami

- Anggap Postman itu frontend/client dan PHP nya itu API Server