MODUL 05

PUT METHOD

A. Capaian Pembelajaran

Setelah mengikuti praktikum ini, mahasiswa diharapkan mampu:

1. Menangani permintaan HTTP PUT pada sisi server menggunakan PHP.
2. Menerapkan penggunaan prepared statement untuk mencegah SQL Injection.
3. Mengembalikan respons HTTP yang sesuai berdasarkan hasil eksekusi operasi database.
4. Menghasilkan output dalam format JSON untuk komunikasi API.

B. Review Materi PUT Method

Salah satu prinsip CRUD yang digunakan RESTful API adalah fungsi PUT atau biasa dinamakan metode PUT. PUT digunakan ketika ingin mengubah nilai data yang ada dalam sistem. PUT pada RESTful API akan mengambil data yang ditentukan berdasarkan id untuk dilakukan perubahan data dan mengirimkannya kembali ke server untuk disimpan.

C. Deskripsi Singkat

Modul ini membimbing mahasiswa untuk membuat endpoint PUT yang mengambil data pengguna dari dalam database users lalu mengubah dan menyimpannya kembali menggunakan PHP. Praktikum ini juga membahas pengolahan input, pengamanan melalui prepared statement, dan pengiriman respons dalam format JSON.

D. Persiapan

1. Pastikan telah memiliki web server lokal (XAMPP, Laragon, dsb).
2. Database MySQL sudah aktif.
3. Database **test_db** dan tabel **users** telah dibuat.
4. Telah membuat folder praktikum-api dengan file **db.php** yang berisi koneksi ke database

E. Membuat API GET Method

1. Buat file baru untuk membuat API dengan nama **put.php** di dalam folder praktikum-api yang telah dibuat sebelumnya.
2. Set Header response

   Tambahkan instruksi PHP untuk memastikan response dari server berupa JSON:

   ```
   header('Content-Type: application/json');
   ```

3. Panggil file koneksi database menggunakan **`require`** dan diikuti **nama file koneksi**
4. Deklarasi variabel untuk menyimpan data yang dikirim oleh client menggunakan **$_POST**, karena PHP tidak otomatis menyediakan variabel superglobal seperti **$_PUT**

```php
$id = $_POST['id'] ?? null;
```

5. Buat deklarasi untuk variabel **`$name`** dan **`$email`** dengan $_POST seperti variabel **`$id`** di atas
6. Buat validasi input untuk variabel memastikan id, name dan email tidak kosong

```php
if (!$id || (!$name && !$email)) {
    // response error
}
```

7. Pada bagian untuk respons error tulislah **`http_response_code(400);`** untuk memberikan kode **400 Bad Request**.
8. Lalu buat kode untuk mengembalikan pesan gagal seperti di bawah ini

```php
echo json_encode([
    // pesan error
]);
```

9. Isi pesan error dengan
    - **status = error**
    - **message = ID dan minimal satu field update harus diisi**
10. Setelah itu tambahkan **`exit;`** dibawah kode **`echo ...([...])`**
11. Selanjutnya buat array kosong **`$updates = []`** untuk meyimpan data yang akan di-update tepat di bawah fungsi validasi input (poin nomor 6)
12. Buat validasi input variabel name untuk kondisi name tidak null sehingga data akan dimasukkan ke dalam array updates. **`$conn->real_escape_string($name)`** akan menghapus karakter spesial dalam string (seperti ', \, dll) sehingga melindungi query dari serangan **SQL Injection**.

```php
if ($name) $updates[] = "name = '".$conn->real_escape_string($name)."'";
```

13. Buat validasi input seperti contoh di atas untuk variabel **`$email`**
14. Buat variabel untuk menampung query yang diperlukan untuk update data seperti di bawah ini

```php
$sql = "UPDATE users SET ".implode(',', $updates)." WHERE id = ".intval($id);
```

15. Selanjutnya tulis kode di bawah ini untuk menjalankan query yang telah dibuat

```php
$res = $conn->query($sql);
```

16. Buat pengecekan update data dengan **if** untuk mengetahui nilai **`$res`** apakah **true** atau **false** dan mengembalikan respons sukses atau gagal.

```php
if ($res && $conn->affected_rows > 0) {
    // response sukses
} else {
    // response gagal
}
```

17. Pada bagian untuk respons sukses tulislah `http_response_code(200);` untuk memberikan kode **200 OK**.
18. Lalu buat kode untuk mengembalikan pesan sukses seperti di bawah ini

```php
echo json_encode([
    // pesan sukses
]);
```

19. Isi pesan sukses dengan
    - **status = success**
    - **message = User berhasil diperbarui**
20. Setelah itu tulislah `http_response_code(404);` pada bagian respons gagal untuk mengembalikan kode **404 Not Found**. Lalu buat kode untuk mengembalikan pesan gagal seperti di bawah ini.

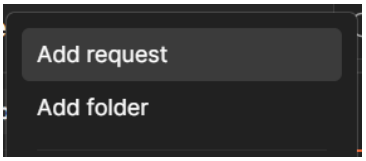
```php
echo json_encode([
    // pesan error
]);
```

21. Isi pesan error dengan
    - **status = error**
    - **message = User tidak ditemukan atau data sama**
22. Terakhir tulis kode untuk menutup koneksi dengan database

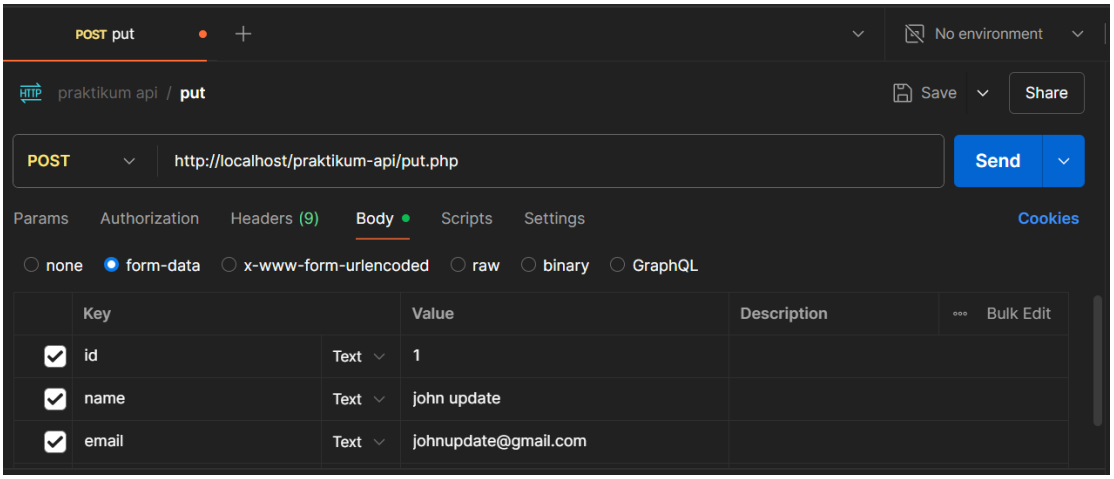
```php
$conn->close();
```

F. Pengujian Postman

1. Buka aplikasi postman
2. Buka **Collection** Praktikum API yang telah dibuat sebelumnya
3. Klik icon titik tiga pada collection dan pilih **Add Request**



4. Beri nama **put** untuk request yang baru saja dibuat.
5. Selanjutnya atur **method** menjadi **POST** dan ketikkan **url** lokal diikuti lokasi file **put.php**
6. Pilih **Body → form-data**, lalu isi bagian **Key** dengan **nama variabel** yang sesuai dengan API yang dibuat. Isi bagian **Value** dengan data yang ingin dimasukkan ke database



7. Setelah mengisi **Value**, klik tombol **Send** di pojok kanan atas. Respons dari API akan muncul seperti gambar berikut