

Laporan Praktikum
Mata Pemrograman Web (PEMWEB)



Pertemuan 5. Praktikum 5
“CRUD (Create, Read, Update, Delete)”

Dosen Pengampu:
Willdan Aprizal Aripin, S.Pd., M.Kom.

Disusun Oleh:
Putri Wahyuni
2310271

PROGRAM STUDI SISTEM INFORMASI KELAUTAN
UNIVERSITAS PENDIDIKAN INDONESIA
2024

I. PENDAHULUAN

CRUD adalah akronim dari Create, Read, Update, dan Delete, yang mewakili serangkaian operasi esensial dalam pengelolaan data di aplikasi berbasis database.

1. Create (C) – Menambahkan Data Baru

Operasi Create digunakan untuk menambah data baru ke dalam database. Sebagai contoh, jika aplikasi Anda mengelola data pengguna, operasi ini memungkinkan penambahan pengguna baru. Di SQL, hal ini dilakukan dengan perintah INSERT.

2. Read (R) – Mengambil atau membaca data

Operasi Read berfungsi untuk mengambil atau menampilkan data yang sudah tersimpan di dalam database. Biasanya digunakan untuk melihat data yang tersedia, dan di SQL, operasi ini dilakukan dengan perintah SELECT.

3. Update (U) – Memodifikasi Data

Operasi Update bertujuan untuk mengubah data yang sudah ada dalam database. Sebagai contoh, jika Anda ingin memperbarui informasi pengguna, perintah yang digunakan dalam SQL adalah UPDATE.

4. Delete (D) – Menghapus Data

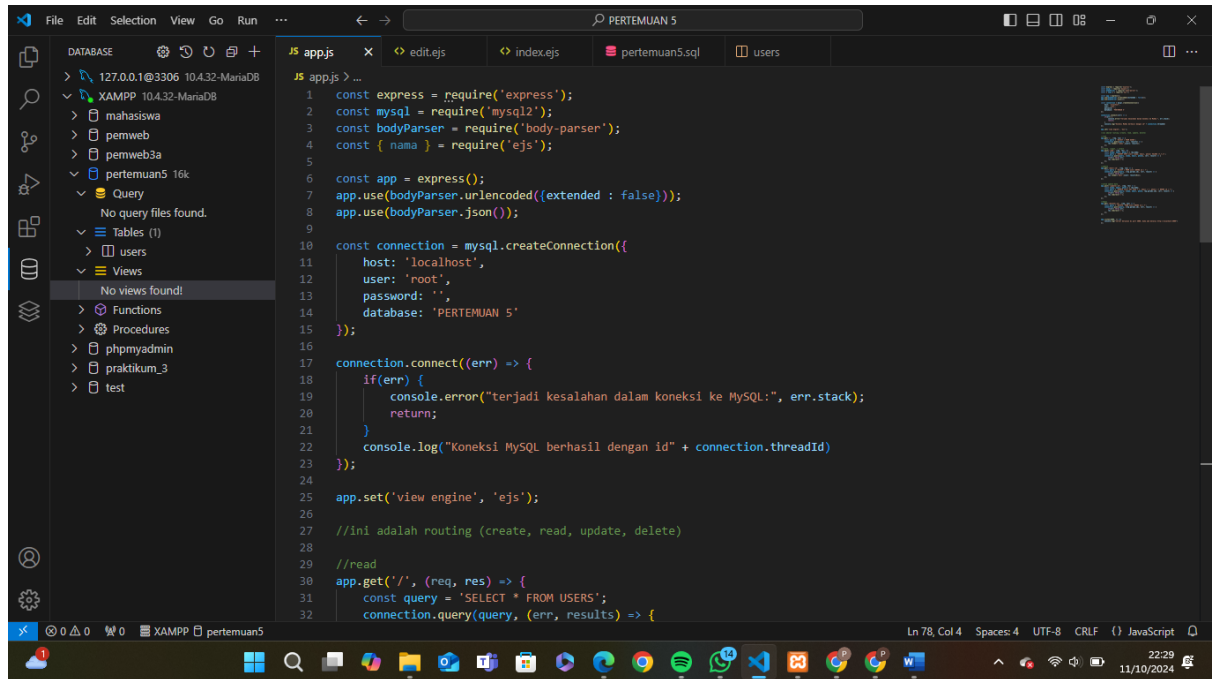
Operasi Delete dipakai untuk menghapus data dari database. Contohnya, ketika ingin menghapus pengguna dari sistem, perintah SQL yang digunakan adalah DELETE.

II. ALAT DAN BAHAN

- Laptop
- Aplikasi Visual Studio Code
- Google Chrome
- Xampp

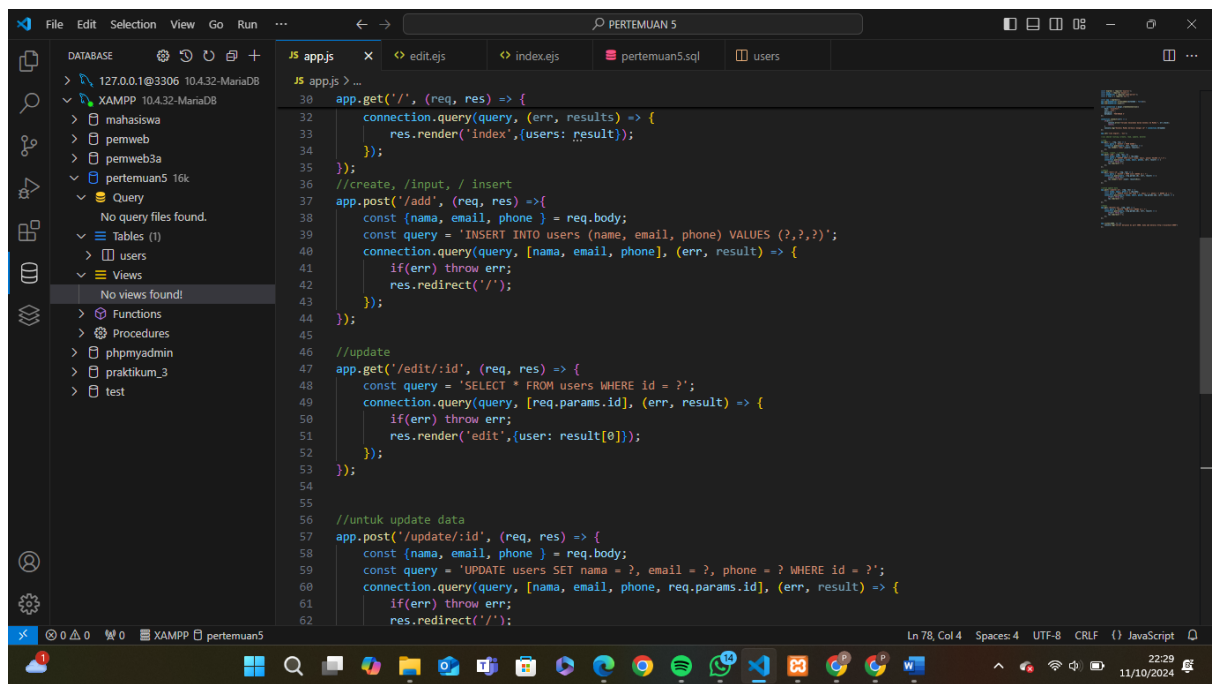
III. PENJELASAN

1. Untuk kodongan app.js



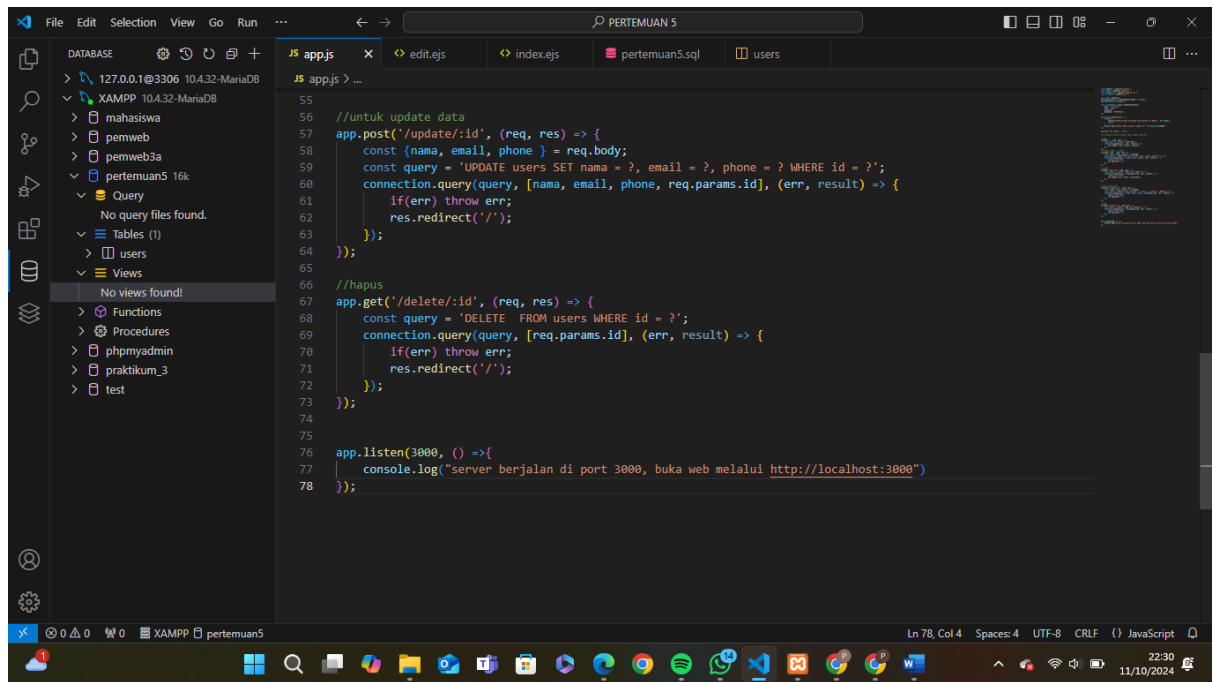
The screenshot shows the VS Code editor with the file explorer on the left displaying the database structure of a XAMPP installation. The main editor window shows the first part of the `app.js` file, which includes the following code:

```
1 const express = require('express');
2 const mysql = require('mysql2');
3 const bodyParser = require('body-parser');
4 const { nama } = require('ejs');
5
6 const app = express();
7 app.use(bodyParser.urlencoded({extended : false}));
8 app.use(bodyParser.json());
9
10 const connection = mysql.createConnection({
11   host: 'localhost',
12   user: 'root',
13   password: '',
14   database: 'PERTEMUAN 5'
15 });
16
17 connection.connect((err) => {
18   if(err) {
19     console.error("terjadi kesalahan dalam koneksi ke MySQL:", err.stack);
20     return;
21   }
22   console.log("Koneksi MySQL berhasil dengan id" + connection.threadId)
23 });
24
25 app.set('view engine', 'ejs');
26
27 //ini adalah routing (create, read, update, delete)
28
29 //read
30 app.get('/', (req, res) => {
31   const query = 'SELECT * FROM users';
32   connection.query(query, (err, results) => {
```



The screenshot shows the VS Code editor with the file explorer on the left displaying the database structure of a XAMPP installation. The main editor window shows the second part of the `app.js` file, which includes the following code:

```
30 app.get('/', (req, res) => {
31   connection.query(query, (err, results) => {
32     res.render('index', {users: results});
33   });
34 });
35
36 //create, /input, / insert
37 app.post('/add', (req, res) =>{
38   const [nama, email, phone] = req.body;
39   const query = 'INSERT INTO users (name, email, phone) VALUES (?, ?, ?)';
40   connection.query(query, [nama, email, phone], (err, result) => {
41     if(err) throw err;
42     res.redirect('/');
43   });
44 });
45
46 //update
47 app.get('/edit/:id', (req, res) => {
48   const query = 'SELECT * FROM users WHERE id = ?';
49   connection.query(query, [req.params.id], (err, result) => {
50     if(err) throw err;
51     res.render('edit', {user: result[0]});
52   });
53 });
54
55 //untuk update data
56 app.post('/update/:id', (req, res) => {
57   const [nama, email, phone] = req.body;
58   const query = 'UPDATE users SET nama = ?, email = ?, phone = ? WHERE id = ?';
59   connection.query(query, [nama, email, phone, req.params.id], (err, result) => {
60     if(err) throw err;
61     res.redirect('/');
62   });
63 });
```



➤ Penjelasan:

1. Impor Modul:

- ``const express = require('express');``: Mengimpor modul Express yang digunakan untuk membuat aplikasi web.
- ``const mysql = require('mysql2');``: Mengimpor modul mysql2 untuk menghubungkan dan berinteraksi dengan database MySQL.
- ``const bodyParser = require('body-parser');``: Mengimpor body-parser untuk menguraikan data yang dikirim dari klien ke server.
- ``const { nama } = require('ejs');``: Mengimpor EJS (Embedded JavaScript) sebagai templating engine, meskipun hanya bagian `nama` tidak digunakan dalam kode yang diberikan.

2. Inisialisasi Aplikasi:

- ``const app = express();``: Membuat instance aplikasi Express.
- ``app.use(bodyParser.urlencoded({extended : false}));``: Menggunakan middleware body-parser untuk menguraikan data URL-encoded.
- ``app.use(bodyParser.json());``: Menggunakan middleware body-parser untuk menguraikan data JSON.

3. Koneksi ke Database:

- ``const connection = mysql.createConnection({...});``: Mengkonfigurasi koneksi ke database MySQL dengan host, user, password, dan nama database.
- ``connection.connect((err) => {...});``: Mencoba untuk terhubung ke database dan menangani kesalahan jika terjadi, serta mencetak ID thread jika koneksi berhasil.

4. Pengaturan View Engine:

- ``app.set('view engine', 'ejs');``: Mengatur EJS sebagai view engine untuk merender tampilan.

5. Rute (Routing):

- Read (Membaca Data):

- `app.get('/', (req, res) => {...});`: Mendefinisikan rute untuk mendapatkan semua pengguna dari database dengan menjalankan query `'SELECT'`. Hasilnya dirender menggunakan template `'index.ejs'`.

- Create (Menambahkan Data):

- `app.post('/add', (req, res) => {...});`: Mendefinisikan rute untuk menambahkan pengguna baru. Data pengguna diambil dari `'req.body'` dan disimpan di database dengan query `'INSERT'`. Setelah berhasil, pengguna akan diarahkan kembali ke halaman utama.

- Update (Memperbarui Data):

- `app.get('/edit/:id', (req, res) => {...});`: Mendefinisikan rute untuk menampilkan formulir edit pengguna berdasarkan ID yang diberikan. Data pengguna diambil dari database menggunakan query `'SELECT'` dan dirender dalam template `'edit.ejs'`.

- `app.post('/update/:id', (req, res) => {...});`: Mendefinisikan rute untuk memperbarui informasi pengguna yang ada. Data yang diperbarui dikirim dari klien, dan query `'UPDATE'` digunakan untuk mengubah data di database.

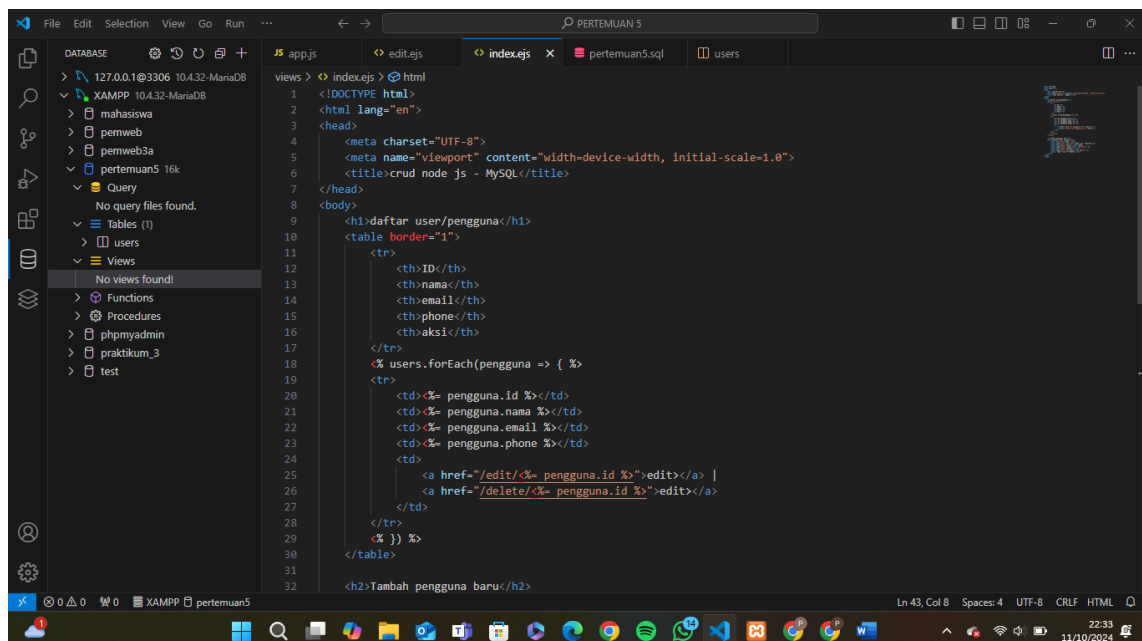
- Delete (Menghapus Data):

- `app.get('/delete/:id', (req, res) => {...});`: Mendefinisikan rute untuk menghapus pengguna berdasarkan ID yang diberikan. Query `'DELETE'` digunakan untuk menghapus data dari database dan setelah itu pengguna diarahkan kembali ke halaman utama.

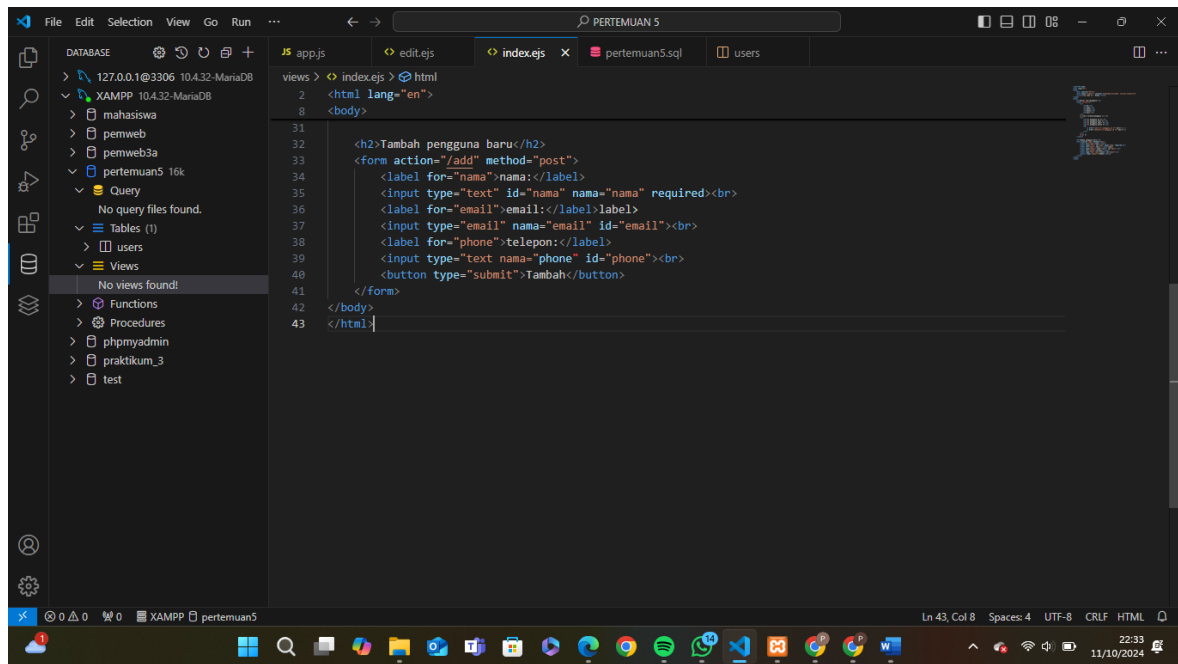
- Menjalankan Server:

- `app.listen(3000, () => {...});`: Menjalankan server pada port 3000 dan mencetak pesan yang memberitahukan bahwa server aktif dan dapat diakses di `'http://localhost:3000'`.

2. Kodingan index.ejs



```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4 <meta charset="UTF-8">
5 <meta name="viewport" content="width=device-width, initial-scale=1.0">
6 <title>crud node js - MySQL</title>
7 </head>
8 <body>
9 <h1>daftar user/pengguna</h1>
10 <table border="1">
11 <tr>
12 <th>ID</th>
13 <th>nama</th>
14 <th>email</th>
15 <th>phone</th>
16 <th>aksi</th>
17 </tr>
18 <% users.forEach(pengguna => { %>
19 <tr>
20 <td>%= pengguna.id %></td>
21 <td>%= pengguna.nama %></td>
22 <td>%= pengguna.email %></td>
23 <td>%= pengguna.phone %></td>
24 <td>
25 <a href="/edit/%= pengguna.id %>">edit</a> |
26 <a href="/delete/%= pengguna.id %>">delete</a>
27 </td>
28 </tr>
29 <% } %>
30 </table>
31
32 <h2>Tambah pengguna baru</h2>
```



➤ Penjelasan:

1. Deklarasi Doctype dan Elemen HTML: Kode dimulai dengan `<!DOCTYPE html>` yang menandakan bahwa ini adalah dokumen HTML5. Elemen `<html lang="en">` menunjukkan bahwa halaman ini menggunakan bahasa Inggris.

2. Bagian Head

- `<meta charset="UTF-8">` untuk menentukan karakter encoding halaman.
- `<meta name="viewport" content="width=device-width, initial-scale=1.0">` untuk memastikan tampilan responsif pada perangkat mobile.
- `<title>crud node js - MySQL</title>` menetapkan judul halaman yang akan ditampilkan di tab browser.

3. Bagian Body

- `<h1>daftar user/pengguna</h1>` menampilkan judul utama halaman.
- `<table border="1">` digunakan untuk membuat tabel dengan garis batas. Tabel ini berfungsi untuk menampilkan data pengguna, dengan kolom untuk ID, nama, email, telepon, dan aksi.
- Baris pertama tabel `<tr>` berisi `<th>` yang mendefinisikan judul kolom.

4. Pengulangan Data Pengguna:

- Kode `<% users.forEach(pengguna => { %>` digunakan untuk melakukan iterasi terhadap array `users`, di mana setiap elemen dari array tersebut akan dirender ke dalam baris tabel baru (`<tr>`).
- Setiap kolom dalam baris diisi dengan data dari objek `pengguna`, menggunakan sintaks `<%= pengguna.id %>`, `<%= pengguna.nama %>`, `<%= pengguna.email %>`, dan `<%= pengguna.phone %>`, yang diambil dari properti objek pengguna.

5. Aksi untuk Edit dan Hapus: Di bagian aksi (`<td>`), terdapat dua tautan:

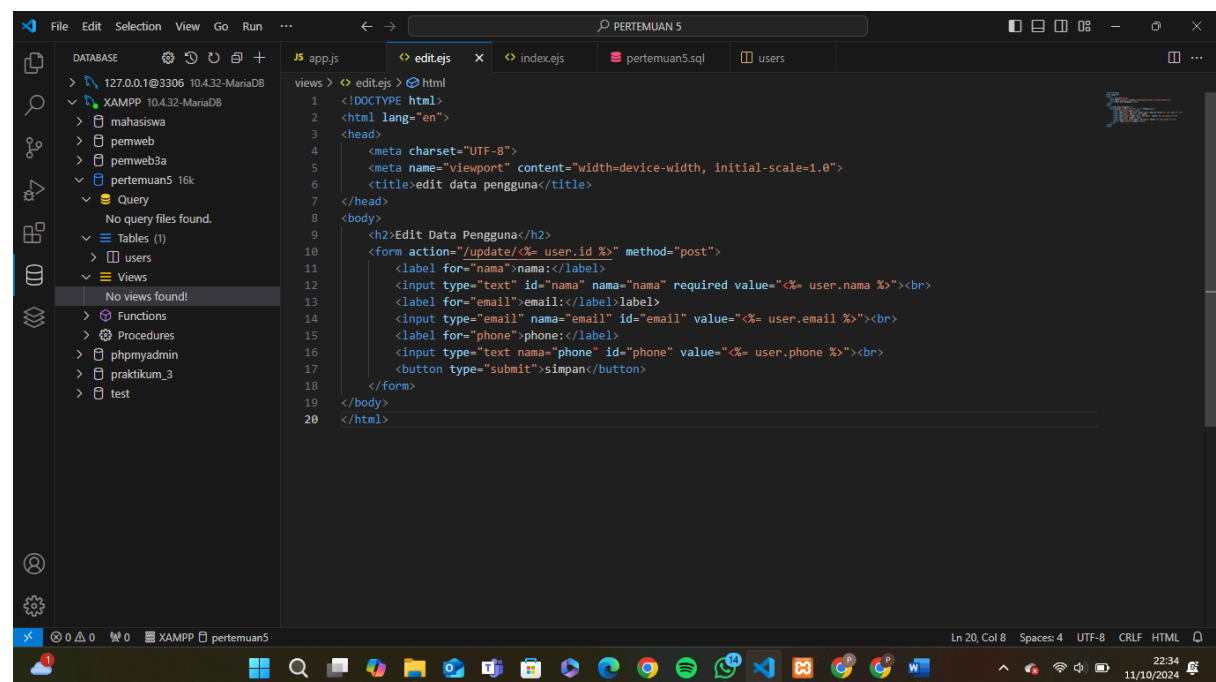
- Tautan untuk mengedit data pengguna (`/edit/<%= pengguna.id %>`).
- Tautan untuk menghapus data pengguna (`/delete/<%= pengguna.id %>`).

6. Formulir Tambah Pengguna:

- Setelah tabel, terdapat formulir untuk menambahkan pengguna baru. Formulir ini memiliki metode pengiriman POST dan akan mengarah ke `/add` saat disubmit.
- Setiap input dalam formulir dilengkapi dengan label dan atribut `required` untuk memastikan bahwa nama harus diisi sebelum mengirimkan formulir.
- Input untuk nama, email, dan telepon masing-masing memiliki atribut `id` dan `name`.

7. Tombol Submit: Terakhir, ada tombol submit bertuliskan "Tambah" yang memungkinkan pengguna untuk mengirimkan data yang telah diisi dalam formulir.

3. Kodingan edit.ejs



```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport" content="width=device-width, initial-scale=1.0">
6   <title>edit data pengguna</title>
7 </head>
8 <body>
9   <h2>Edit Data Pengguna</h2>
10  <form action="/update/<%= user.id %%" method="post">
11    <label for="nama">nama:</label>
12    <input type="text" id="nama" name="nama" required value="<%= user.nama %%"><br>
13    <label for="email">email:</label>
14    <input type="email" name="email" id="email" value="<%= user.email %%"><br>
15    <label for="phone">phone:</label>
16    <input type="text" name="phone" id="phone" value="<%= user.phone %%"><br>
17    <button type="submit">simpan</button>
18  </form>
19 </body>
20 </html>
```

➤ Penjelasan:

1. Deklarasi Tipe Dokumen:

- `<!DOCTYPE html>`: Menunjukkan bahwa dokumen ini adalah dokumen HTML5.

2. Tag HTML dan Elemen Dasar:

- `<html lang="en">`: Membuka tag HTML dan menetapkan bahasa dokumen sebagai bahasa Inggris.
- `<head>`: Bagian ini berisi informasi metadata tentang dokumen, seperti karakter set dan judul.

- `<meta charset="UTF-8">`: Menetapkan pengkodean karakter dokumen ke UTF-8, yang mendukung banyak karakter.
- `<meta name="viewport" content="width=device-width, initial-scale=1.0">`: Memastikan bahwa halaman dapat ditampilkan dengan baik pada perangkat seluler dengan mengatur lebar tampilan sesuai dengan lebar perangkat.
- `<title>edit data pengguna</title>`: Menetapkan judul halaman yang akan ditampilkan di tab browser.

3. Bagian Konten:

- `<body>`: Bagian ini berisi konten utama dari halaman.
- `<h2>Edit Data Pengguna</h2>`: Menampilkan judul yang menjelaskan bahwa halaman ini digunakan untuk mengedit data pengguna.

4. Formulir Pengeditan:

- `<form action="/update/<%= user.id %>" method="post">`: Membuat formulir dengan metode POST untuk mengirimkan data ke server. `action` menunjukkan URL tujuan yang berisi ID pengguna yang sedang diedit, di mana `<%= user.id %>` diisi dengan ID pengguna yang bersangkutan.
- Label dan Input:
 - `<label for="nama">nama:</label>`: Label untuk kolom nama.
 - `<input type="text" id="nama" nama="nama" required value="<%= user.nama %>">`: Input teks untuk nama pengguna. Atribut `value` diisi dengan nama pengguna yang sudah ada, menggunakan EJS untuk menampilkan data yang sesuai dari objek pengguna (`user.nama`). Atribut `required` menunjukkan bahwa kolom ini harus diisi.
 - `<label for="email">email:</label>`: Label untuk kolom email.
 - `<input type="email" nama="email" id="email" value="<%= user.email %>">`: Input email untuk alamat email pengguna. Atribut `value` diisi dengan email pengguna yang sudah ada.
 - `<label for="phone">phone:</label>`: Label untuk kolom nomor telepon.
 - `<input type="text" nama="phone" id="phone" value="<%= user.phone %>">`: Input teks untuk nomor telepon pengguna. Atribut `value` diisi dengan nomor telepon pengguna yang sudah ada. (Catatan: Ada kesalahan penulisan pada `input` ini, seharusnya `name` dan bukan `nama`).

5. Tombol Kirim:

- `<button type="submit">simpan</button>`: Tombol untuk mengirimkan formulir. Ketika tombol ini ditekan, data yang diisi dalam formulir akan dikirimkan ke server untuk diperbarui.

➤ **Link GitHub**

IV. KESIMPULAN

Kode dalam `app.js` membangun logika server dengan mengatur koneksi ke database dan menetapkan berbagai rute untuk operasi CRUD, yang mencakup membaca, menambah, mengedit, dan menghapus data pengguna. Sementara itu, `index.ejs` berperan sebagai tampilan utama yang menyajikan daftar pengguna dalam bentuk tabel dan menyediakan formulir untuk menambah pengguna baru. Di sisi lain, `edit.ejs` memungkinkan pengguna untuk memperbarui informasi mereka dengan menampilkan data yang telah diisi sebelumnya. Struktur keseluruhan ini menawarkan antarmuka yang intuitif dan efisien bagi pengguna dalam mengelola data mereka melalui aplikasi berbasis web.