

**LAPORAN PRAKTIKUM MATA KULIAH**  
**PEMOGRAMAN WEB**



**Pertemuan 5 Praktikum 4**

**Dosen Pengampu :**

Willdan Aprizal Arifin, S.Pd., M.Kom.

**Disusun Oleh :**

Syifa Fauzia (2309845)

SIK B/3

**PROGRAM STUDI SISTEM INFORMASI KELAUTAN**

**UNIVERSITAS PENDIDIKAN INDONESIA**

**2024**

# 1. Pendahuluan

Pada praktikum ini, kita akan membuat sebuah aplikasi web sederhana dengan fitur CRUD (Create, Read, Update, Delete) menggunakan Node.js dengan framework Express, MySQL sebagai basis data, serta EJS (Embedded JavaScript) sebagai template engine untuk menampilkan data di sisi frontend. CRUD adalah operasi dasar yang biasa digunakan dalam pengelolaan basis data. Dengan implementasi CRUD, kita dapat membuat (create), membaca (read), memperbarui (update), dan menghapus (delete) data dari basis data.

## Tujuan Praktikum

- Mempelajari cara membangun aplikasi web sederhana dengan operasi CRUD.
- Memahami cara menghubungkan aplikasi Node.js dengan database MySQL.
- Menerapkan template engine EJS untuk rendering data dari server ke halaman web.

# 2. Teknologi Utama

## 2.1 Node.js

Node.js adalah lingkungan runtime JavaScript berbasis Chrome V8 yang memungkinkan kita menjalankan JavaScript di sisi server. Node.js sering digunakan untuk membangun aplikasi web yang cepat dan efisien.

## 2.2 Express.js

Express.js adalah framework minimalis untuk Node.js yang memudahkan pengelolaan routing dan middleware, sehingga mempercepat pengembangan aplikasi web. Dalam proyek ini, Express digunakan sebagai server untuk menangani request HTTP dan menyediakan endpoint untuk operasi CRUD.

## 2.3 MySQL

MySQL adalah sistem manajemen basis data relasional (RDBMS) yang banyak digunakan untuk menyimpan dan mengelola data dalam tabel. MySQL dipilih dalam proyek ini untuk menyimpan informasi pengguna seperti nama, email, dan nomor telepon.

## 2.4 EJS (Embedded JavaScript)

EJS adalah template engine untuk Node.js yang memungkinkan kita menyisipkan logika JavaScript ke dalam HTML. EJS akan membantu kita untuk menampilkan data dari server (hasil query MySQL) ke halaman web.

### 3. Penjelasan Kode

#### 3.1 Koneksi ke MySQL

```
9   const connection = mysql.createConnection({
10     host: 'localhost',
11     user: 'root',
12     password: '',
13     database: 'pertemuan5'
14   });
15
```

Kode ini mendefinisikan koneksi ke server MySQL lokal dengan menggunakan paket mysql2. Koneksi dibuat dengan mengatur host, user, password, dan nama database (pertemuan5).

#### 3.2 Pengaturan Middleware

```
5   const app = express();
6   app.use(bodyParser.urlencoded({extended : false}));
7   app.use(bodyParser.json());
8
```

Middleware body-parser digunakan untuk memproses data yang dikirimkan melalui form. urlencoded menangani data form yang dikirimkan dalam format URL, dan json menangani data JSON.

#### 3.3 Routing CRUD

##### 3.3.1 Read (Menampilkan Data)

```
28   //Read
29   app.get('/', (req, res) => {
30     const query = 'SELECT * FROM users';
31     connection.query(query, (err, results) => {
32       res.render('index', {users:results});
33     });
34   });
35
```

Endpoint ini menangani permintaan GET ke root URL (/). Kode ini menjalankan query SQL untuk mengambil semua data dari tabel users dan kemudian menampilkan hasilnya pada halaman index.ejs.

### 3.3.2 Create (Menambah Data)

```
37 //create / input / insert
38 v app.post('/add', (req, res) => {
39   const {nama, email, phone} = req.body;
40   const query = 'INSERT INTO users (nama, email, phone) VALUES (?, ?, ?)';
41 v   connection.query(query, [nama, email, phone], (err, result) => {
42     if (err) throw err;
43     res.redirect('/');
44   });
45 });
46
```

Kode ini menangani permintaan POST untuk menambah data baru. Ketika pengguna mengirimkan data melalui form, nama, email, dan nomor telepon dari body request diambil dan dimasukkan ke dalam tabel users menggunakan perintah SQL INSERT.

### 3.3.3 Update (Memperbarui Data)

- **Menampilkan Form Update:**

```
46
47 //update
48 //untuk akses halaman
49 app.get('/edit/:id', (req, res) => {
50   const query = 'SELECT * FROM users WHERE id = ?';
51   connection.query(query, [req.params.id], (err, result) => {
52     if (err) throw err;
53     res.render('edit', {user:result[0]});
54   });
55 })
56
```

Permintaan GET ke /edit/:id akan menampilkan data pengguna berdasarkan id yang dikirimkan di URL. Data ini kemudian akan ditampilkan pada form di halaman edit.ejs.

- **Mengirim Data untuk Diperbarui:**

```
56
57 //untuk update data
58 app.post('/update/:id', (req, res) =>{
59   const {nama, email, phone} = req.body;
60   const query = 'UPDATE users SET nama = ?, email = ?, phone = ? WHERE id = ?';
61   connection.query(query, [nama, email, phone, req.params.id], (err, result) =>{
62     if (err) throw err;
63     res.redirect('/');
64   });
65 })
66
```

Kode ini menangani permintaan POST untuk memperbarui data pengguna di database berdasarkan id yang dikirimkan. Query UPDATE akan mengubah data pengguna sesuai input baru yang dimasukkan ke dalam form.

### 3.3.4 Delete (Menghapus Data)

```
67 //hapus
68 v app.get('/delete/:id', (req, res) => {
69     const query = 'DELETE FROM users WHERE id = ?';
70 v     connection.query(query, [req.params.id], (err, result) => {
71         if (err) throw err;
72         res.redirect('/');
73     });
74 })
75
```

Endpoint ini menghapus data dari tabel users berdasarkan id yang dikirimkan melalui URL. Query SQL DELETE digunakan untuk menghapus data tersebut.

### 3.4 Menjalankan Server

```
76 app.listen(3000, () => {
77     console.log("Server berjalan di prt 3000, buka web melalui http://localhost:3000 ")
78 });
```

## Penjelasan Detail mengenai index.ejs

### 1. Deklarasi Standar HTML

```
views > <> index.ejs > html > body > table > tr > th
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>
```

Bagian ini adalah deklarasi standar HTML. Kode ini menetapkan tipe dokumen HTML, pengaturan bahasa (dalam hal ini "en" untuk bahasa Inggris), dan pengaturan meta untuk charset (UTF-8), serta viewport agar halaman responsif. Judul halaman diatur menjadi Crud Node JS - MySQL.

## 2. Judul Halaman dan Tabel Pengguna

```
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>Telephon</th>
16      <th>Aksi</th>
17    </tr>
```

Bagian ini membuat judul halaman "Daftar User/Pengguna" dan sebuah tabel HTML untuk menampilkan daftar pengguna. Baris pertama tabel (header) menggunakan tag <th> untuk menampilkan judul kolom seperti ID, Nama, Email, Telephon, dan Aksi.

## 3. Rendering Data Pengguna dengan EJS

```
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 %>">Hapus</a>
27     </td>
28   </tr>
29 <% })%>
30 </table>
```

Bagian ini adalah kode EJS yang digunakan untuk menampilkan data pengguna yang berasal dari server.

- `users.forEach` adalah metode yang digunakan untuk melakukan iterasi (loop) pada array `users` yang berisi daftar pengguna yang didapat dari database.
- Setiap kali loop, variabel `Pengguna` akan merujuk ke satu data pengguna, dan kita bisa mengakses properti seperti `id`, `nama`, `email`, dan `phone` menggunakan sintaks EJS (`<%= ... %>`).
- Pada kolom "Aksi", terdapat dua tautan:
  - Edit: Mengarahkan ke halaman `/edit/<%= Pengguna.id %>` untuk mengedit data pengguna berdasarkan ID.
  - Hapus: Mengarahkan ke halaman `/delete/<%= Pengguna.id %>` untuk menghapus data pengguna berdasarkan ID.

#### 4. Form Tambah Pengguna

```
31
32     <h2>Tambah Pengguna Baru </h2>
33     <form action="/add" method="post">
34         <label for="nama">Nama:</label>
35         <input type="text" id="nama" name="nama" required><br>
36         <label for="email">Email:</label>
37         <input type="email" name="email" id="email"><br>
38         <label for="phone">Telephon:</label>
39         <input type="text" name="phone" id="phone"><br>
40         <button type="submit">Tambah</button>
41     </form>
42
43 </body>
44 </html>
```

Bagian ini berfungsi untuk menampilkan form yang memungkinkan pengguna menambahkan data pengguna baru.

- Form ini menggunakan metode POST dan akan mengirimkan data ke endpoint /add.
- Form berisi tiga input yaitu nama, email dan telephon

Setiap input dilengkapi dengan label dan atribut name yang digunakan untuk mengirimkan data ke server. Setelah data diisi, pengguna bisa menekan tombol Tambah untuk menambahkan data ke database.

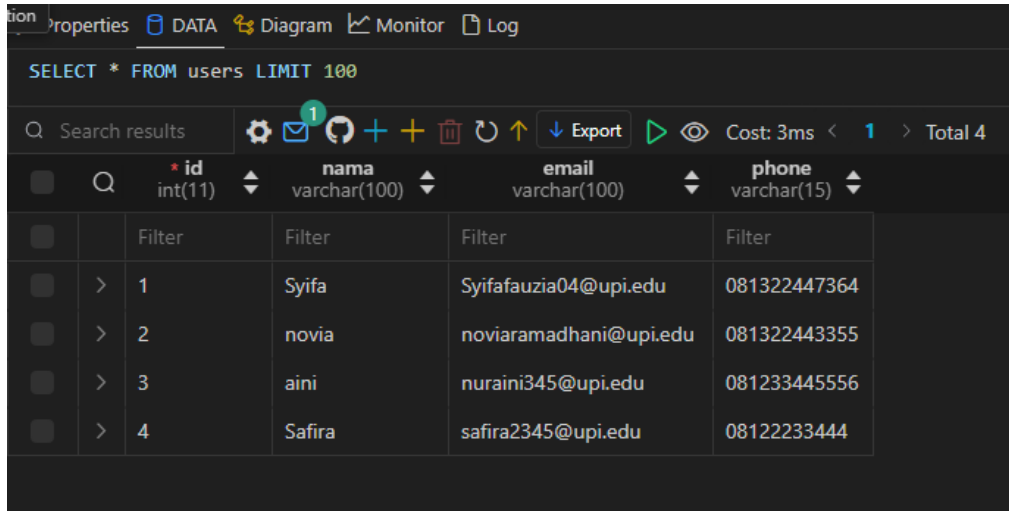
#### Penjelasan Detail mengenai edit.ejs

```
views > edit.ejs > html
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">Telephon:</label>
16         <input type="text" name="phone" id="phone" value="<%= user.phone %%"><br>
17         <button type="submit">Tambah</button>
18     </form>
19
20 </body>
21 </html>
```

Bagian ini menampilkan judul "Edit Data Pengguna" dan mendefinisikan formulir dengan metode POST yang akan mengirimkan data ke endpoint di sini merujuk pada ID pengguna yang datanya akan diperbarui. Setiap field input menampilkan nilai pengguna yang sudah ada

(nama, email, dan telepon) dari objek user. Nilai ini dirender dengan EJS (<%= user.nama %> dan seterusnya), sehingga saat halaman ditampilkan, field sudah terisi dengan data lama pengguna, siap untuk diperbarui. Tombol submit memungkinkan pengguna mengirimkan perubahan yang telah dilakukan di form ke server untuk diperbarui di database.

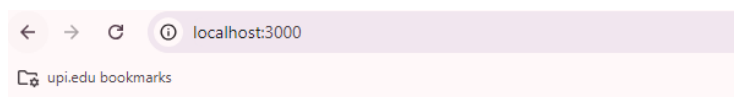
Ini beberapa contoh dari data yang di buat di table lalu di users



SELECT \* FROM users LIMIT 100

id	nama	email	phone
1	Syifa	Syifafauzia04@upi.edu	081322447364
2	novia	noviaramadhani@upi.edu	081322443355
3	aini	nuraini345@upi.edu	081233445556
4	Safira	safira2345@upi.edu	08122233444

Hasil akhir setelah di coba di web



## Daftar User/Pengguna

ID	Nama	Email	Telephon	Aksi
1	Syifa	syifafauzia04@upi.edu	081322447364	<a href="#">Edit</a> <a href="#">Hapus</a>
2	Novia	noviaramadhani@upi.edu	081322443355	<a href="#">Edit</a> <a href="#">Hapus</a>
4	Safira	safira2345@upi.edu	08122233444	<a href="#">Edit</a> <a href="#">Hapus</a>
7	Sarah	sarahalfira@upi.edu	1234567890	<a href="#">Edit</a> <a href="#">Hapus</a>
8	Aini	nuraini@upi.edu	08132345678	<a href="#">Edit</a> <a href="#">Hapus</a>

## Tambah Pengguna Baru

Nama:

Email:

Telephon:

## 3. Kesimpulan

Form HTML yang tersedia untuk input dan pembaruan data memberikan pengalaman pengguna yang intuitif dalam mengelola informasi. Operasi CRUD dalam aplikasi ini berjalan sesuai harapan, mulai dari pengambilan data, penyimpanan, hingga pembaruan dan penghapusan data. Secara keseluruhan, kombinasi teknologi yang digunakan menciptakan aplikasi yang sederhana namun efektif untuk mengelola data, dan dapat menjadi dasar untuk pengembangan aplikasi yang lebih kompleks di masa depan.



