

Laporan Praktikum
Mata Kuliah Pemograman WEB



Tugas 5

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

Disusun Oleh :
(Nur'aini Dwi Anra)
(2301148)

PROGRAM STUDI SISTEM INFORMASI KELAUTAN
UNIVERSITAS PENDIDIKAN INDONESIA
2024

1. Pendahuluan

Pada praktikum kali ini, kami membuat aplikasi web sederhana yang menerapkan operasi CRUD (Create, Read, Update, Delete) menggunakan Node.js dengan framework Express dan database MySQL. Aplikasi ini dirancang untuk memberikan pengguna kemampuan dalam menambah, melihat, mengedit, dan menghapus data pengguna dengan antarmuka yang mudah digunakan. Tujuan dari praktikum ini adalah untuk memahami dan mengimplementasikan konsep dasar server-side programming dalam pengelolaan data menggunakan basis data. Selain itu, praktikum ini bertujuan untuk memberikan pemahaman mendalam tentang bagaimana berbagai komponen teknologi web bekerja secara bersama-sama, serta bagaimana membangun aplikasi yang responsif dan interaktif.

2. Struktur Program

Aplikasi ini dibangun dengan menggunakan beberapa teknologi utama:

- Node.js: Platform yang digunakan untuk menjalankan server dan aplikasi.
- Express: Framework yang mempermudah routing dan pengelolaan HTTP request.
- MySQL: Sistem manajemen database yang digunakan untuk menyimpan data pengguna.
- EJS (Embedded JavaScript): Template engine yang digunakan untuk menampilkan data dinamis di halaman HTML.
- CSS: Cascading Style Sheets digunakan untuk mengatur tampilan dan nuansa aplikasi.

3. Penjelasan Code

3.1 Koneksi ke Database MySQL

Untuk menghubungkan aplikasi ke MySQL, kami menggunakan library mysql2 dan membuat koneksi dengan mysql.createConnection. Berikut adalah potongan kode untuk membuat koneksi:

```
const express = require('express');
const mysql = require('mysql2');
const bodyParser = require('body-parser');

const app = express();
app.use(bodyParser.urlencoded({extended: false}));
app.use(bodyParser.json());
app.use(express.static('public'));

const connection = mysql.createConnection({
  host: 'localhost',
  user: 'root',
  password: '',
  database: 'pertemuan5'
});

connection.connect((err) => {
  if(err){
    console.error("Terjadi kesalahan dalam koneksi ke MySQL:", err.stack);
    return;
  }
  console.log("Koneksi MySQL berhasil dengan id" + connection.threadId)
});

app.set('view engine', 'ejs')
```

Pada bagian ini, saya menghubungkan aplikasi ke database pertemuan5 yang berisi tabel users.

3.2 Koneksi visual code ke localhost:3000

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

3.3 Rounting (Backend)

Kami membuat beberapa rute (route) untuk mengelola operasi CRUD:

- **Route / (Read):** Mengambil dan menampilkan daftar pengguna dari database. Data pengguna diambil menggunakan query MySQL `SELECT * FROM users`, lalu ditampilkan menggunakan template EJS.

```
// Read  
app.get('/', (req, res) => {  
  const query = 'SELECT * FROM users';  
  connection.query(query, (err, results) => {  
    if (err) throw err;  
    res.render('index', { users: results });  
  });  
});
```

- **Route /add (Create):** Form di halaman utama digunakan untuk menambah pengguna baru. Setelah data di-submit, data akan dikirim melalui metode POST ke server untuk ditambahkan ke database dengan query MySQL `INSERT`.

```
//create /input /insert  
app.post('/add', (req, res) => {  
  const {nama, email, phone} = req.body;  
  const query = 'INSERT INTO users (nama, email, phone) VALUES (?, ?, ?)';  
  connection.query(query, [nama, email, phone], (err, result) => {  
    if (err) throw err;  
    res.redirect('/');  
  });  
});
```

- **Route /edit /:id (Update):** Sistem akan mengambil data berdasarkan id, menampilkan form edit, dan setelah perubahan, akan mengupdate data tersebut ke database.

```
//update  
//untuk akses halaman  
app.get('/edit/:id', (req, res) => {  
  const query = 'SELECT * FROM users WHERE id = ?';  
  connection.query(query, [req.params.id], (err, result) => {  
    if (err) throw err;  
    res.render('edit', { user: result[0] });  
  });  
});  
  
// untuk update dan alert  
app.post('/update/:id', (req, res) => {  
  const {nama, email, phone} = req.body;  
  const query = 'UPDATE users SET nama = ?, email = ?, phone = ? WHERE id = ?';  
  connection.query(query, [nama, email, phone, req.params.id], (err, result) => {  
    if (err) throw err;  
    res.redirect('/?message=update-success');  
  });  
});
```

- **Route /delete/:id (Delete):** Pengguna dapat dihapus dari database dengan query DELETE FROM users WHERE id = ?.

```
// untuk delete dan alert
app.post('/delete/:id', (req, res) => {
  const query = 'DELETE FROM users WHERE id = ?';
  connection.query(query, [req.params.id], (err, result) => {
    if (err) throw err;
    res.redirect('/?message=delete-success');
  });
});
```

3.4 Template Engine (EJS)

Kami menggunakan EJS untuk menampilkan data dinamis di halaman HTML. Contoh penerapan EJS adalah pada halaman daftar pengguna, di mana data pengguna yang diambil dari database ditampilkan dalam tabel.

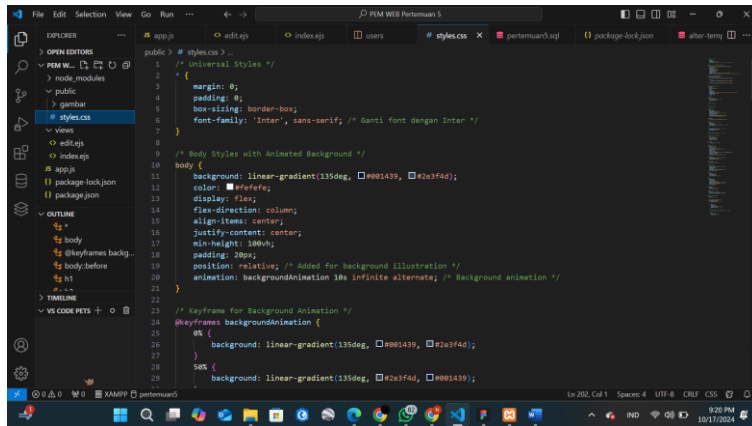
```
<body>
  <h1>Daftar User/Pengguna</h1>
  <table>
    <tr>
      <th>ID</th>
      <th>Nama</th>
      <th>Email</th>
      <th>Telepon</th>
      <th>Aksi</th>
    </tr>
    <% users.forEach(Pengguna => {%)
    <tr>
      <td><%= Pengguna.id %></td>
      <td><%= Pengguna.nama %></td>
      <td><%= Pengguna.email %></td>
      <td><%= Pengguna.phone %></td>
      <td>
        <a href="/edit/<%= Pengguna.id %>">Edit</a>
        <form id="deleteForm-<%= Pengguna.id %>" action="/delete/<%= Pengguna.id %>" method="POST"
          style="display:inline;"
          <input type="hidden" name="_method" value="DELETE">
          <button type="submit" onclick="return confirm('Apakah Anda yakin ingin menghapus pengguna ini?')">Hapus</button>
        </form>
      </td>
    </tr>
    <% })%>
  </table>
```

3.5 Interaksi dan Validasi

Untuk meningkatkan pengalaman pengguna, kami menambahkan beberapa interaksi dan validasi, seperti konfirmasi sebelum menghapus pengguna dan pesan alert yang muncul setelah menambahkan atau mengupdate data pengguna. Fungsi showAlert digunakan untuk menampilkan pesan tersebut.

```
<script>
  function showAlert(message) {
    alert(message);
  }
```

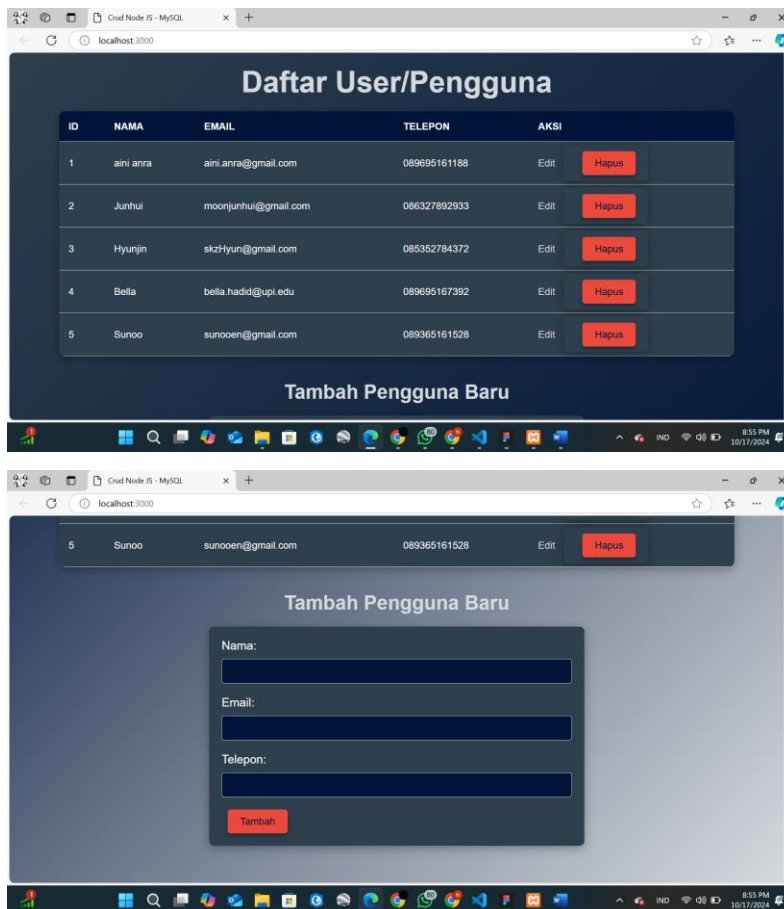
3.6 CSS

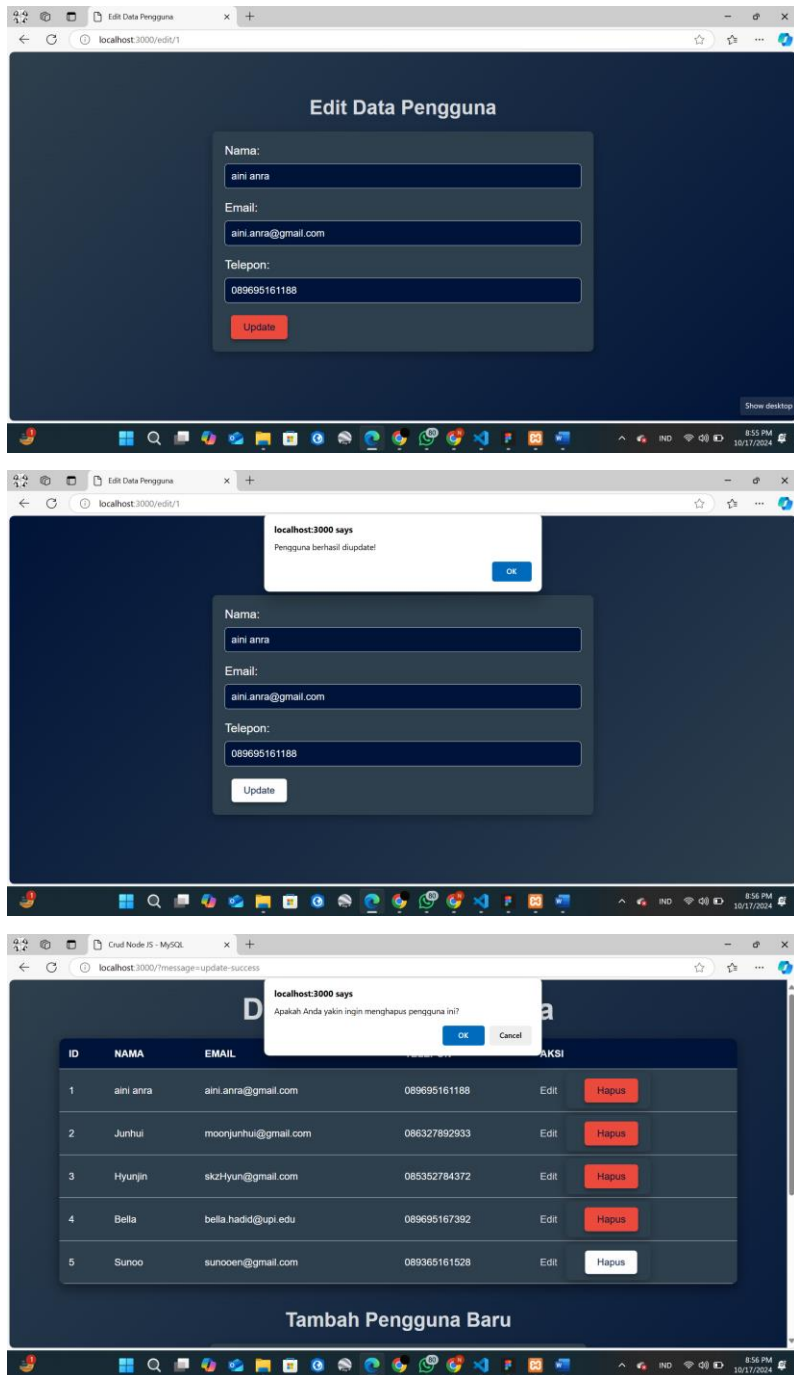


```
1 /* Universal Styles */
2 {
3   margin: 0;
4   padding: 0;
5   box-sizing: border-box;
6   font-family: 'Inter', sans-serif; /* Ganti font dengan Inter */
7 }
8
9 /* Body Styles with Animated Background */
10 body {
11   background: linear-gradient(135deg, #001439, #e2e3f4);
12   color: #001439;
13   display: flex;
14   flex-direction: column;
15   align-items: center;
16   justify-content: center;
17   min-height: 100vh;
18   padding: 20px;
19   position: relative; /* Added for background illustration */
20   animation: backgroundAnimation 10s infinite alternate; /* Background animation */
21 }
22
23 /* Keyframes for Background Animation */
24 @keyframes backgroundAnimation {
25   0% {
26     background: linear-gradient(135deg, #001439, #e2e3f4);
27   }
28   50% {
29     background: linear-gradient(135deg, #e2e3f4, #001439);
30   }
31 }
```

4. Hasil

Berikut adalah tampilan dari aplikasi CRUD yang telah di tambahkan CSS setelah dijalankan di localhost:





Pada tampilan ini, pengguna dapat melihat daftar data pengguna yang sudah ada. Data yang ditampilkan mencakup ID, nama pengguna, email, dan nomor telepon. Website ini dilengkapi dengan fitur CRUD di mana pengguna dapat menambahkan data baru melalui formulir yang tersedia, mengedit informasi yang ada, serta menghapus data dengan konfirmasi untuk mencegah

kesalahan. Aplikasi ini juga dilengkapi dengan notifikasi yang memberikan umpan balik kepada pengguna setelah setiap operasi, meningkatkan interaksi secara keseluruhan.

Proses ini dibuat sederhana agar semua pengguna, bahkan yang tidak berpengalaman sekalipun, dapat menggunakan sistem ini. Website ini dibangun menggunakan Node.js dan Express untuk backend, dengan MySQL sebagai database untuk menyimpan data pelanggan. Untuk tampilan antarmuka, kami menggunakan EJS sebagai template engine dan CSS untuk styling, menciptakan pengalaman visual yang menarik dan fungsional.

5. Kesimpulan

Aplikasi CRUD yang dikembangkan menggunakan Node.js, Express, dan MySQL memungkinkan pengguna untuk mengelola data dengan efektif melalui antarmuka web. Melalui penerapan operasi CRUD, pengguna dapat melakukan berbagai tindakan terhadap data, seperti menambah, membaca, memperbarui, dan menghapus data pengguna dengan mudah. Implementasi template engine EJS memungkinkan tampilan data yang dinamis dan responsif, sementara CSS digunakan untuk menciptakan pengalaman visual yang menarik. Selain itu, praktikum ini menunjukkan pentingnya integrasi antara server-side programming dan database dalam pengembangan aplikasi web yang fungsional. Ke depan, aplikasi ini dapat dikembangkan lebih lanjut dengan penambahan fitur-fitur seperti validasi data yang lebih kompleks, autentikasi pengguna, serta interaksi yang lebih baik, untuk meningkatkan pengalaman pengguna secara keseluruhan.