

**Laporan Praktikum**  
**Mata Kuliah Pemograman Berbasis Object**



**Tugas 5. Pertemuan 6**

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 memungkinkan pengguna untuk menambah, melihat, mengedit, dan menghapus data pengguna. Tujuan dari praktikum ini adalah untuk memahami cara kerja server-side programming dalam mengelola data dengan database.

## 2. Struktur Program

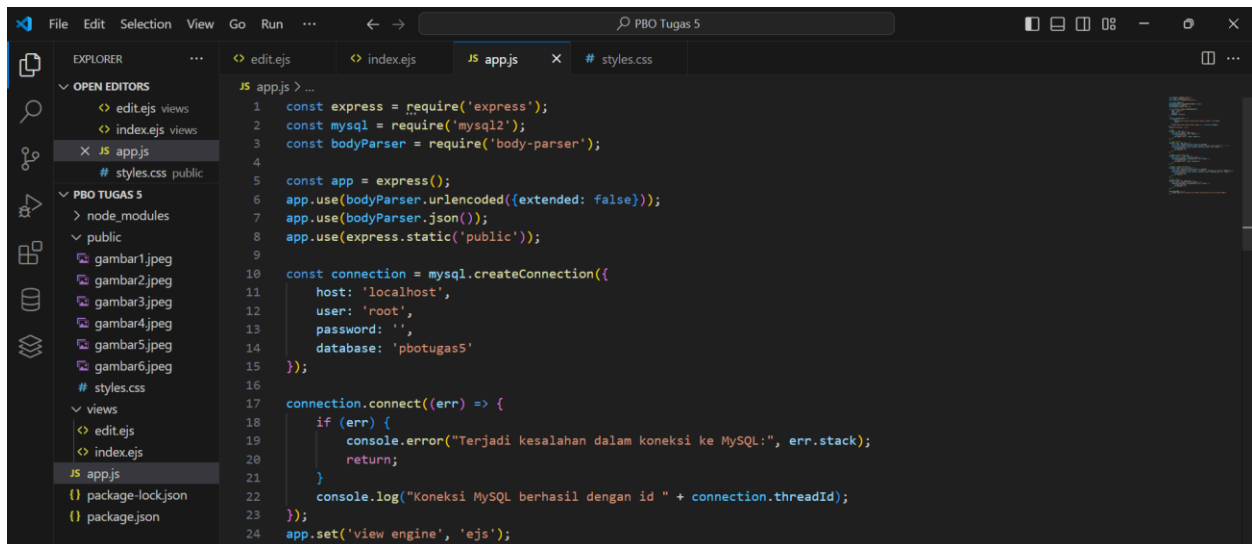
Aplikasi ini dibangun dengan menggunakan beberapa teknologi utama:

- **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.

## 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:



```
1  const express = require('express');
2  const mysql = require('mysql2');
3  const bodyParser = require('body-parser');
4
5  const app = express();
6  app.use(bodyParser.urlencoded({extended: false}));
7  app.use(bodyParser.json());
8  app.use(express.static('public'));
9
10 const connection = mysql.createConnection({
11   host: 'localhost',
12   user: 'root',
13   password: '',
14   database: 'pbotugas5'
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 app.set('view engine', 'ejs');
```

Pada bagian ini, saya menghubungkan aplikasi ke database PBO TUGAS 5 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

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 (Insert)
app.post('/add', (req, res) => {
  const { name, datetime, partysize, phone } = req.body;
  const query = 'INSERT INTO users (name, datetime, partysize, phone) VALUES (?, ?, ?, ?)';
  connection.query(query, [name, datetime, partysize, phone], (err, result) => {
    if (err) throw err;
    res.redirect('/');
  });
});
```

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

```
// Update - Akses halaman edit
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] });
  });
});

// Update - Lakukan update data
app.post('/update/:id', (req, res) => {
  const { name, datetime, partysize, phone } = req.body;
  const query = 'UPDATE users SET name = ?, datetime = ?, partysize = ?, phone = ? WHERE id = ?';
  connection.query(query, [name, datetime, partysize, phone, req.params.id], (err, result) => {
    if (err) throw err;
    res.redirect('/');
  });
});
```

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

```
// Delete (Hapus)
app.get('/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('/');
  });
});
```

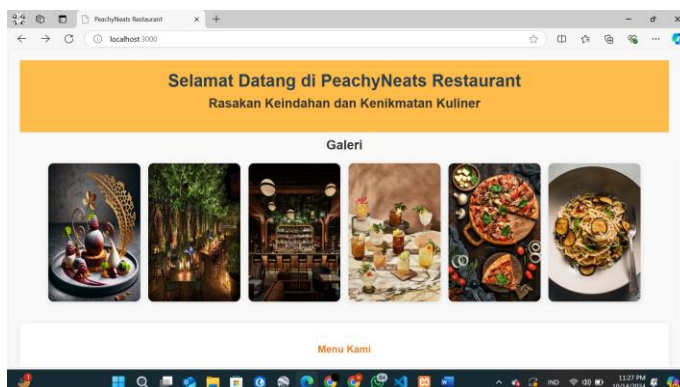
### 3.4 Template Engine (EJS)

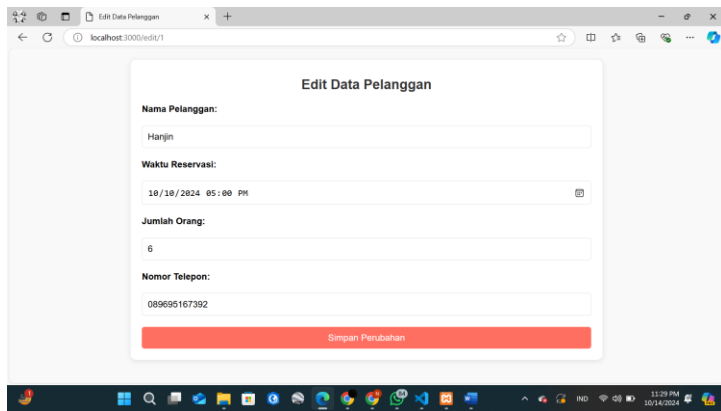
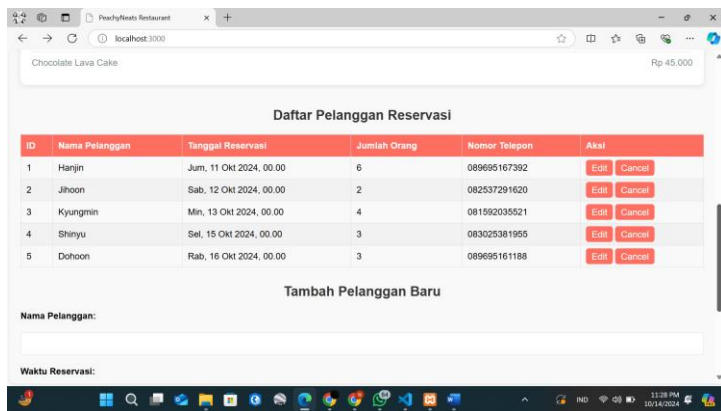
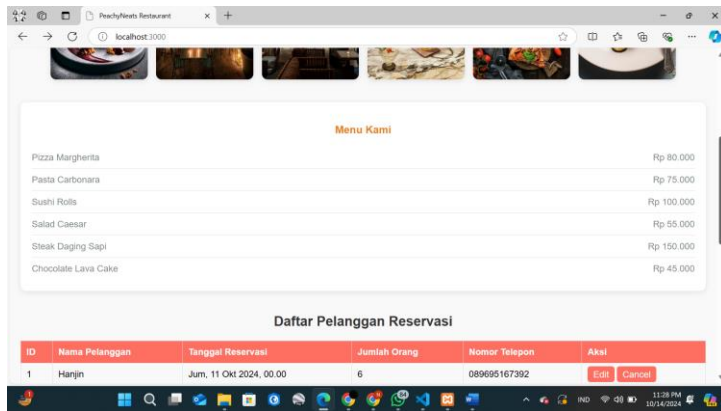
Kami menggunakan EJS untuk menampilkan data dinamis di halaman HTML. Contoh penerapan EJS adalah pada halaman daftar pengguna, di mana data pengguna yang diambil dari database ditampilkan dalam tabel.

```
<tbody>
  <% users.forEach(function(User) { %>
    <tr>
      <td><%= User.id %></td>
      <td><%= User.name %></td>
      <td>
        <%= new Date(User.datetime).toLocaleString('id-ID', {
          weekday: 'short',
          year: 'numeric',
          month: 'short',
          day: 'numeric',
          hour: '2-digit',
          minute: '2-digit'
        }) %>
      </td>
      <td><%= User.partysize %></td>
      <td><%= User.phone %></td>
      <td>
        <a href="/edit/<%= User.id %>" class="button">Edit</a>
        <a href="/delete/<%= User.id %>" class="button delete">Cancel</a>
      </td>
    </tr>
  <% }) %>
</tbody>
```

## 4. Hasil

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





Pada tampilan ini, terdapat informasi lengkap tentang restoran kami, termasuk menu, galeri foto, serta sistem reservasi pelanggan yang mudah dan efisien. Kami memiliki galeri foto yang menampilkan berbagai hidangan dan suasana restoran. Gambar-gambar ini memberikan gambaran tentang pengalaman kuliner yang bisa Anda nikmati di PeachyNeats.

Pengunjung dapat melihat daftar reservasi pelanggan yang sudah ada. Data yang ditampilkan mencakup ID, nama pelanggan, waktu reservasi, jumlah orang, dan nomor telepon. Fitur ini memudahkan pelanggan untuk melakukan reservasi dan memastikan tempat duduk yang nyaman.

Website ini dilengkapi dengan fitur CRUD untuk manajemen data pelanggan. Pengunjung dapat menambah, mengedit, dan menghapus data reservasi dengan mudah. 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 dibangun dengan Node.js, Express, dan MySQL memungkinkan pengelolaan data pengguna dengan mudah melalui browser. Operasi CRUD (Create, Read, Update, Delete) dapat dilakukan secara efektif menggunakan server-side programming dan database yang terintegrasi dengan baik. Aplikasi ini dapat dikembangkan lebih lanjut untuk penggunaan yang lebih kompleks, misalnya dengan menambahkan fitur validasi data atau autentikasi pengguna.