

Laporan Praktikum
Mata Kuliah Pemograman WEB



Tugas 4. Pertemuan 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 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

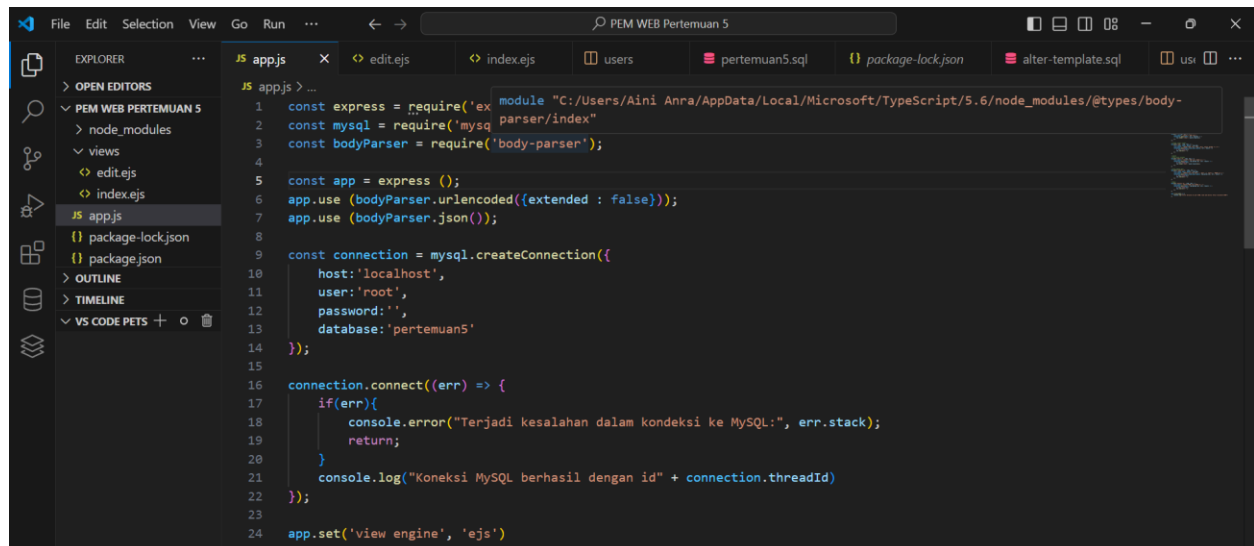
Aplikasi ini dibangun dengan menggunakan beberapa teknologi utama:

- **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:

A screenshot of the Visual Studio Code editor interface. The Explorer sidebar on the left shows a project named 'PEM WEB PERTEMUAN 5' with files like 'node_modules', 'views', 'edit.ejs', 'index.ejs', 'app.js', 'package-lock.json', and 'package.json'. The main editor window displays the 'app.js' file. The code in 'app.js' includes imports for 'express', 'mysql2' (aliased as 'mysql'), and 'body-parser'. It sets up an Express app with body-parser middleware and creates a MySQL connection using 'mysql.createConnection' with the following configuration: host: 'localhost', user: 'root', password: '', and database: 'pertemuan5'. The connection is established using 'connection.connect', and an error handler is provided. A log message is added upon successful connection. Finally, the view engine is set to 'ejs'.

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

3.2 Routing

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) => {
    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):** 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
//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]});
  });
});
```

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

```
//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('/');
  });
});
```

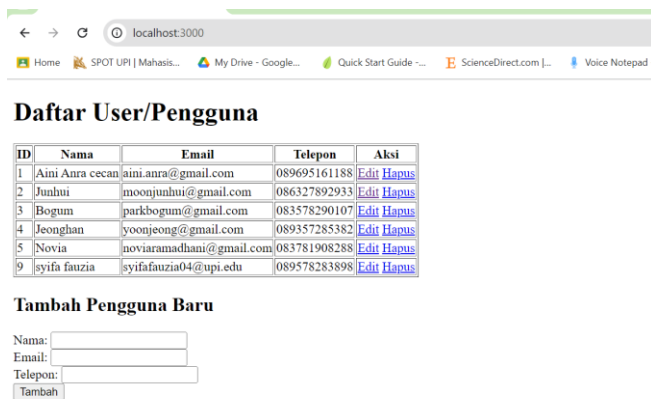
3.3 Template Engine (EJS)

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.

```
<% 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>
      <a href="/delete/<%= Pengguna.id %>">Hapus</a>
    </td>
  </tr>
<% })%>
```

4. Hasil

Berikut adalah tampilan dari aplikasi CRUD setelah dijalankan di browser:



Daftar User/Pengguna

ID	Nama	Email	Telepon	Aksi
1	Aini Anra cecan	aini.anra@gmail.com	089695161188	Edit Hapus
2	Junhui	moonjunhui@gmail.com	086327892933	Edit Hapus
3	Bogum	parkbogum@gmail.com	083578290107	Edit Hapus
4	Jeonghan	yoonjeong@gmail.com	089357285382	Edit Hapus
5	Novia	noviaramadhani@gmail.com	083781908288	Edit Hapus
9	syifa fauzia	syifafauzia04@upi.edu	089578283898	Edit Hapus

Tambah Pengguna Baru

Nama:

Email:

Telepon:

Pada tampilan ini, terdapat tabel berisi daftar pengguna, form untuk menambah pengguna baru, dan aksi edit serta hapus yang dapat dilakukan langsung melalui tampilan web.

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.