

Laporan Praktikum
Mata Kuliah Pemograman WEB



Tugas 4. Pertemuan 5

Dosen Pengampu :

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

Disusun Oleh :

(Novia Ramadhani)

(2305968)

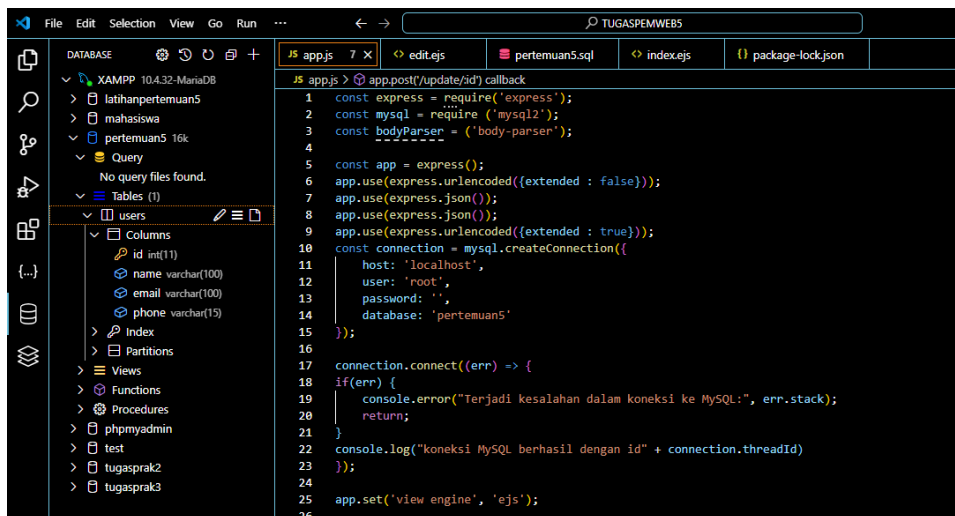
PROGRAM STUDI SISTEM INFORMASI KELAUTAN
UNIVERSITAS PENDIDIKAN INDONESIA
2024

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. 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.

PENJELASAN CODE



```
1 const express = require('express');
2 const mysql = require('mysql2');
3 const bodyParser = ('body-parser');
4
5 const app = express();
6 app.use(express.urlencoded({extended : false}));
7 app.use(express.json());
8 app.use(express.json());
9 app.use(express.urlencoded({extended : true}));
10 const connection = mysql.createConnection({
11   host: 'localhost',
12   user: 'root',
13   password: '',
14   database: 'pertemuan5'
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');
```

- **express:** adalah framework web minimalis untuk Node.js yang memudahkan pembuatan server web. Dengan Express bisa membuat rute, menangani permintaan HTTP, dan menyajikan konten statis.
- **mysql2:** Ini adalah pustaka untuk berkomunikasi dengan database MySQL dari aplikasi Node.js. Digunakan untuk membuat koneksi, menjalankan kueri, dan berinteraksi dengan database MySQL.
- **body-parser:** Middleware untuk Express yang digunakan untuk memproses data **request body** dalam format JSON atau URL-encoded yang dikirimkan melalui form HTML .
- **connection.connect():** Fungsi ini mencoba membuat koneksi dengan database MySQL menggunakan konfigurasi yang telah ditetapkan.
- Route ini menangani request ke halaman utama (/).

- **app.set('view engine', 'ejs');** Mengatur **EJS** sebagai view engine untuk merender halaman HTML. EJS memungkinkan kamu untuk menyisipkan logika JavaScript dalam template HTML dan mengirimkan data dari server ke tampilan.
- Jika ada error dalam query, server akan mengirimkan status **500 Internal Server Error**.
- Jika berhasil, hasil query (results) dikirim ke template index dengan variabel users

```

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) => {
33     if (err) {
34       console.error(err);
35       res.status(500).send("Internal Server Error");
36       return;
37     }
38     res.render('index', {users: results});
39   });
40 });
41
42 //create / input / insert
43 app.post('/add', (req, res) => {
44   const {name, email, phone} = req.body;
45   const query = 'INSERT INTO users (name, email, phone) VALUES (?, ?, ?)';
46   connection.query(query, [name, email, phone], (err, result) => {
47     if (err) throw err;
48     res.redirect('/');
49   });
50 });

```

- **connection.query(query, (err, results)):** Menjalankan query SQL dan menangani hasilnya:
Jika terjadi error selama eksekusi query, akan ditampilkan error di console dan mengirimkan **500 Internal Server Error** ke klien. Jika berhasil, data hasil query (results) dikirimkan ke template index.ejs dan ditampilkan dengan variabel users.

Create/Input/Insert

- **Tujuan:** Route ini digunakan untuk menambahkan data baru ke tabel users dari form yang dikirimkan oleh klien.
- **const { name, email, phone } = req.body:** Mengambil data name, email, dan phone dari body request (dikirimkan oleh form HTML).
- **INSERT INTO users (name, email, phone) VALUES (?, ?, ?):** Query SQL untuk menambahkan data baru ke tabel users. Tanda ? akan digantikan dengan nilai yang diambil dari body request (name, email, phone).
- **connection.query(query, [name, email, phone]):** Menjalankan query insert ke database, mengisi nilai name, email, dan phone.
- Jika berhasil, pengguna akan diarahkan kembali ke halaman utama (/), yang akan memuat data terbaru yang baru ditambahkan.

DATABASE

XAMPP 10.4.32-MariaDB

latihanpertemuan5

mahasiswa

pertemuan5 16k

Query

No query files found.

Tables (1)

users

Views

Functions

Procedures

phpmyadmin

test

tugasprak2

tugasprak3

JS app.js 7

edit.js

pertemuan5.sql

users

index.js

package-lock.json

views > edit.js > html CAUsers\novia\OneDrive\Documents\TUGASPEMWEB5\views\edit.js

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Edit Data Pengguna</title>

</head>

<body>

<h2>Edit Data Pengguna</h2>

<form action="/update/<%= user.id %>" method="post">

<label for="name">Nama:</label>

<input type="text" id="name" name="name" required value="<%= user.name %>">

<label for="email">Email:</label>

<input type="email" name="email" id="email" value="<%= user.email %>">

<label for="phone">Telepon:</label>

<input type="text" name="phone" id="phone" value="<%= user.phone %>">

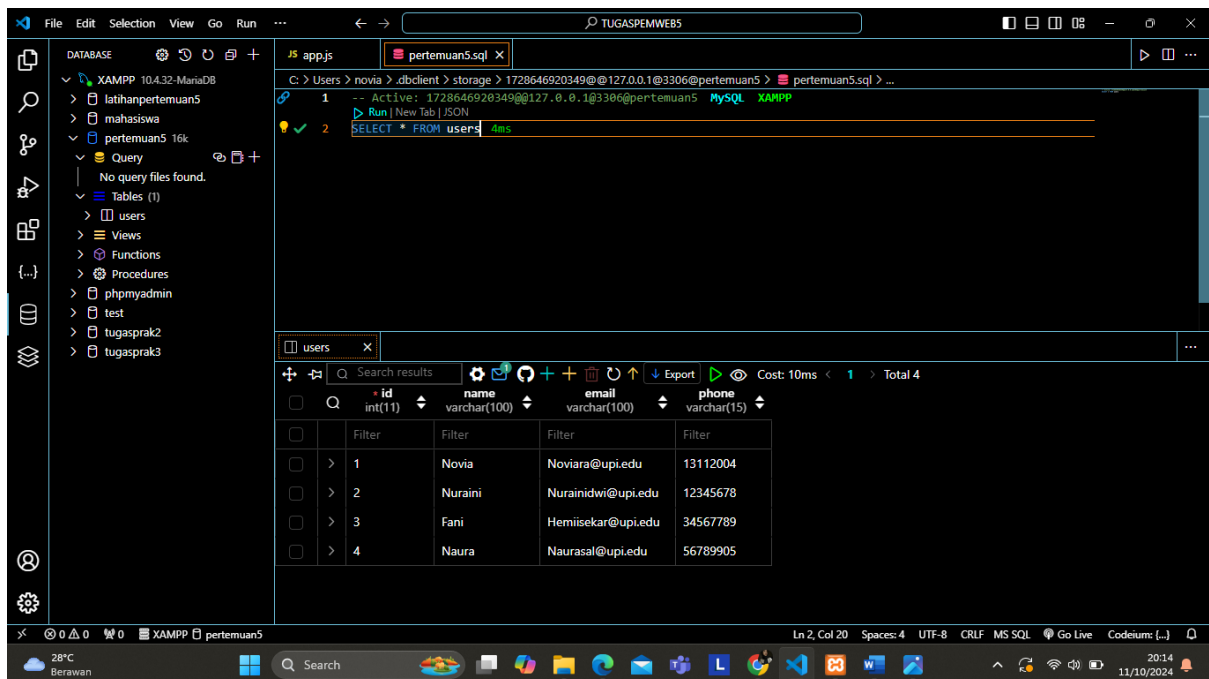
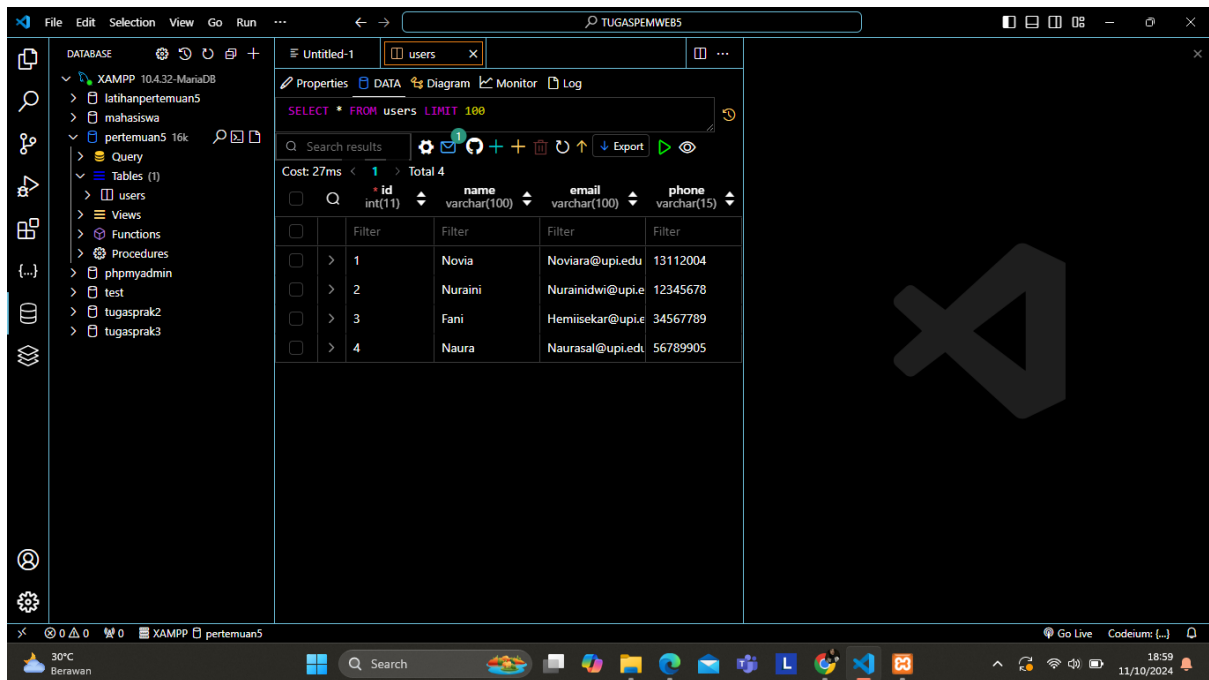
<button type="submit">Tambah</button>

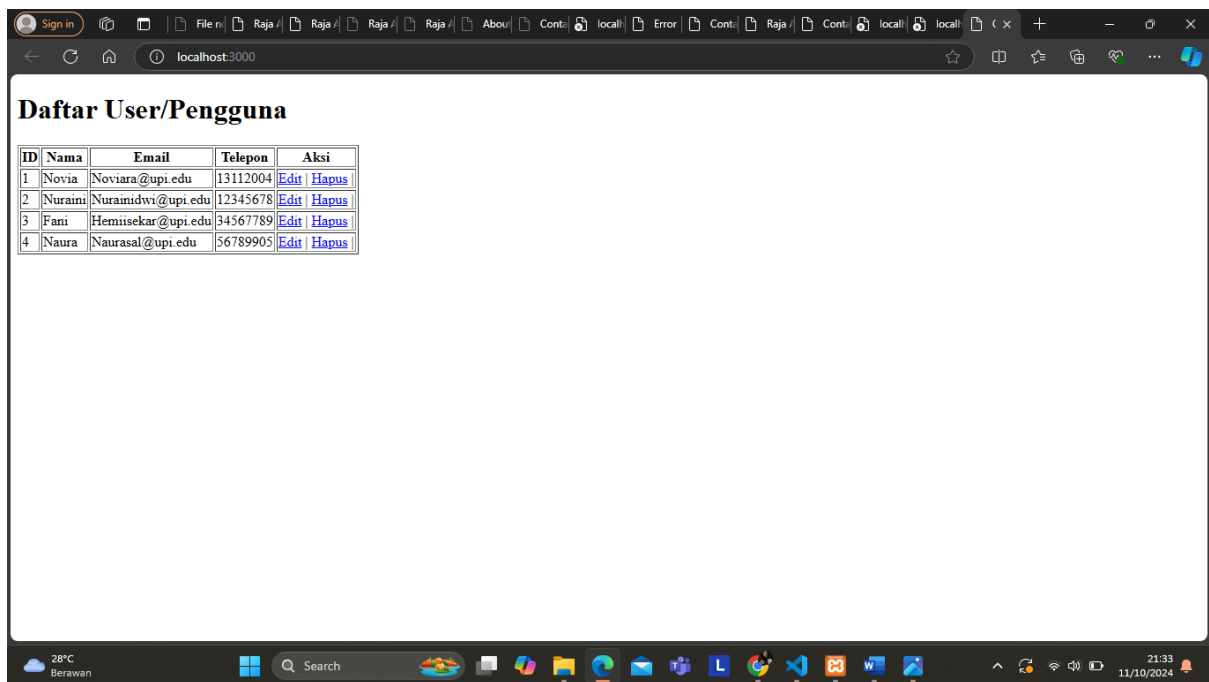
</form>

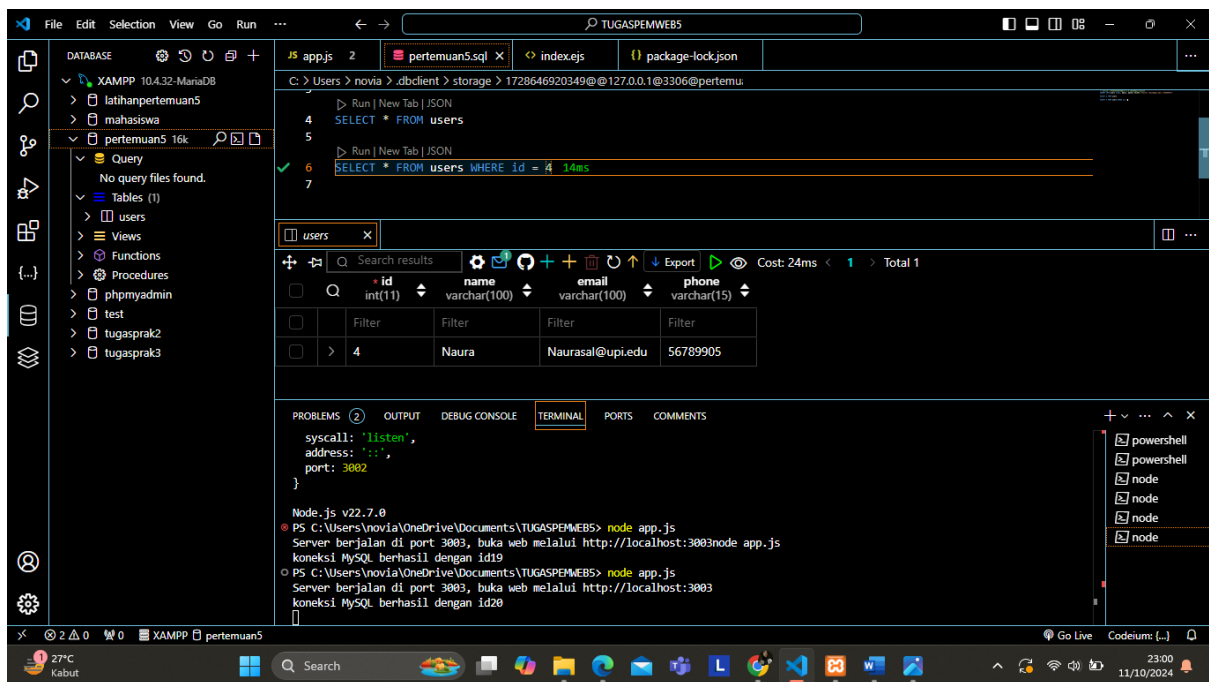
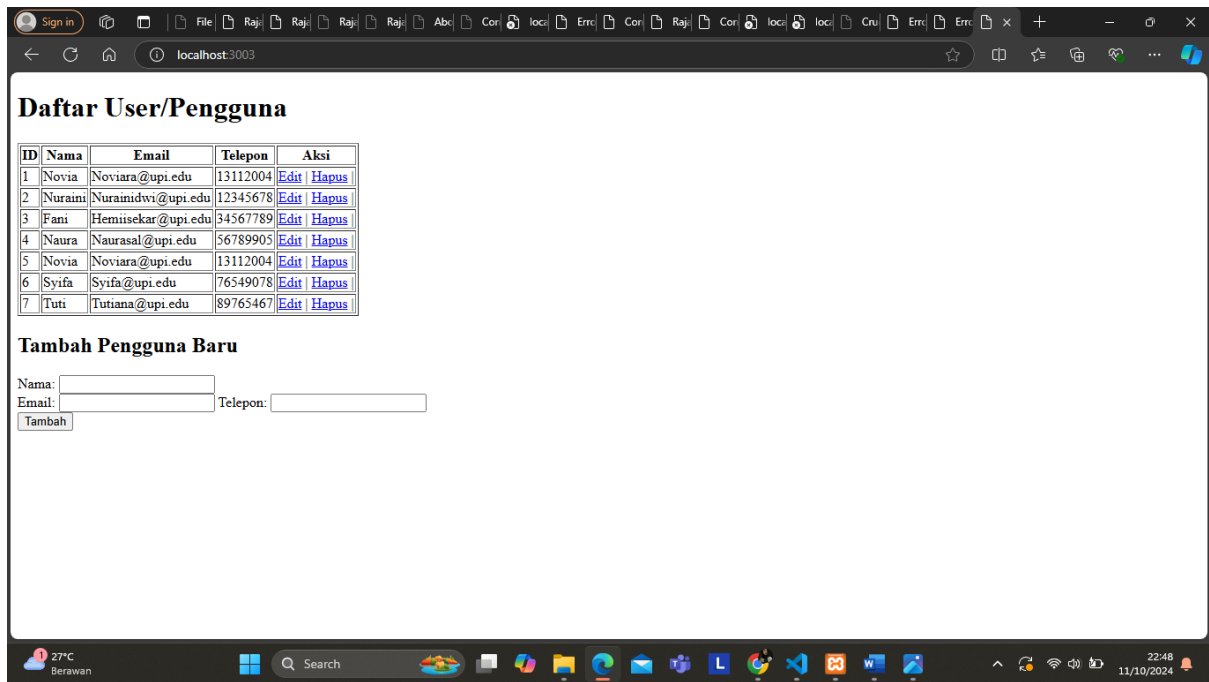
</body>

</html>

Foto Hasil:







//Hasil Update Data

The screenshot shows the VS Code interface with the Database Explorer on the left, displaying the 'users' table. The main editor shows the SQL editor with the following queries:

```
SELECT * FROM users
SELECT * FROM users WHERE id = 4
UPDATE users SET name = "Naura Cantik", email = "Naurasasa@gmail.com", phone = "1243568976" WHERE id = 4
```

The 'users' table structure is shown in the Database Explorer:

id	name	email	phone
int(11)	varchar(100)	varchar(100)	varchar(15)

The terminal output shows the execution of the update query and the resulting data:

```
Node.js v22.7.0
PS C:\Users\novia\OneDrive\Documents\TUGASPEMWEB5> node app.js
Server berjalan di port 3003, buka web melalui http://localhost:3003node app.js
koneksi MySQL berhasil dengan id19
PS C:\Users\novia\OneDrive\Documents\TUGASPEMWEB5> node app.js
Server berjalan di port 3003, buka web melalui http://localhost:3003
koneksi MySQL berhasil dengan id20
```

//Hasil Fitur Hapus

The screenshot shows the VS Code interface with the Database Explorer on the left, displaying the 'users' table. The main editor shows the SQL editor with the following query:

```
SELECT * FROM users LIMIT 100
```

The 'users' table structure is shown in the Database Explorer:

id	name	email	phone
int(11)	varchar(100)	varchar(100)	varchar(15)

The terminal output shows the execution of the delete query and the resulting data:

```
Node.js v22.7.0
PS C:\Users\novia\OneDrive\Documents\TUGASPEMWEB5> node app.js
Server berjalan di port 3003, buka web melalui http://localhost:3003node app.js
koneksi MySQL berhasil dengan id19
PS C:\Users\novia\OneDrive\Documents\TUGASPEMWEB5> node app.js
Server berjalan di port 3003, buka web melalui http://localhost:3003
koneksi MySQL berhasil dengan id20
```