

Laporan Praktikum
Mata Kuliah Pemrograman Berorientasi Objek



Pertemuan 6
“CRUD”

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

Disusun Oleh :
Fadli Kurnia Ramadhan
2307373

PROGRAM STUDI SISTEM INFORMASI KELAUTAN
UNIVERSITAS PENDIDIKAN INDONESIA
2024

I. PENDAHULUAN

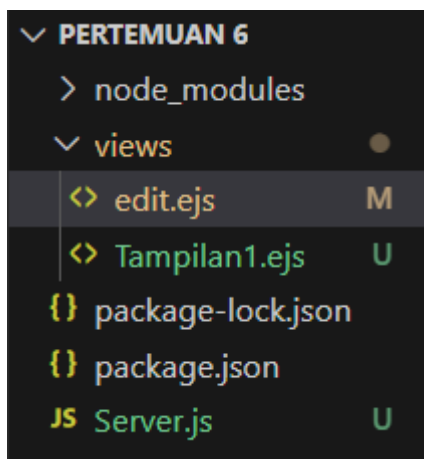
Dalam era digital yang terus berkembang, manipulasi data menjadi tulang punggung dari sebagian besar aplikasi web modern. Salah satu konsep fundamental yang mendasari pengelolaan data ini adalah CRUD (Create, Read, Update, Delete). CRUD merupakan akronim yang merangkum empat operasi dasar dalam interaksi dengan basis data persisten: membuat, membaca, memperbarui, dan menghapus data. Konsep ini, meskipun sederhana dalam definisinya, memiliki implikasi yang luas dalam arsitektur perangkat lunak, desain antarmuka pengguna, dan keamanan sistem.

Penelitian ini bertujuan untuk mengeksplorasi relevansi dan implementasi CRUD dalam konteks pengembangan aplikasi web kontemporer. Dengan memahami nuansa dari setiap operasi CRUD, para pengembang dapat merancang sistem yang lebih efisien, aman, dan mudah dipelihara. Lebih lanjut, studi ini akan menyelidiki bagaimana prinsip-prinsip CRUD berintegrasi dengan paradigma pemrograman modern, kerangka kerja web, dan arsitektur aplikasi yang berkembang.

II. ALAT DAN BAHAN

- Laptop
- Aplikasi Visual Studio Code
- Bootstrap
- CodeSnap

III. PENJELASAN PROGRAM



1. Hal yang pertama saya lakukan yaitu membuat folder dan file seperti contoh diatas.

```

1  const express = require('express'); // Mengimpor modul Express untuk membuat server
2  const mysql = require('mysql2'); // Mengimpor modul MySQL2 untuk koneksi dengan database MySQL
3  const bodyParser = require('body-parser'); // Mengimpor modul Body Parser untuk mengelola data dari form
4
5  const app = express(); // Membuat instance aplikasi Express
6
7  app.use(bodyParser.urlencoded({ extended: false })); // Menggunakan body-parser untuk menangani data form yang dikirimkan melalui URL-encoded
8  app.use(bodyParser.json()); // Menggunakan body-parser untuk menangani data JSON
9
10 // Koneksi ke database MySQL
11 const connection = mysql.createConnection({
12   host: 'localhost', // Alamat host database (localhost karena database berjalan di komputer lokal)
13   user: 'root', // Username MySQL
14   password: '', // Password MySQL (dikosongkan jika tidak ada)
15   database: 'pertemuan5' // Nama database yang digunakan dalam aplikasi
16 });
17
18 // Cek koneksi MySQL
19 connection.connect((err) => {
20   if (err) { // Jika terjadi kesalahan, tampilkan pesan error
21     console.error("Terjadi kesalahan dalam koneksi ke MySQL:", err.stack);
22     return;
23   }
24   console.log("Koneksi MySQL berhasil dengan id " + connection.threadId); // Jika koneksi berhasil, tampilkan ID thread dari koneksi
25 });
26
27 // Menggunakan EJS sebagai view engine untuk merender halaman HTML
28 app.set('view engine', 'ejs'); // Menyatakan bahwa EJS adalah template engine yang digunakan
29 app.set('views', './views'); // Menyatakan bahwa semua file template HTML berada di folder 'views'
30
31 // Route untuk Read (menampilkan semua user)
32 app.get('/', (req, res) => {
33   connection.query('SELECT * FROM users', (err, result) => { // Query SQL untuk mengambil semua data dari tabel 'users'
34     if (err) throw err; // Jika ada kesalahan dalam query, tampilkan error
35     res.render('Tampilan1', { users: result }); // Render file 'index.ejs' dengan data user yang diambil dari database
36   });
37 });
38
39 // Route untuk Create (menambahkan user baru)
40 app.post('/add', (req, res) => {
41   const { name, email, nim } = req.body; // Mengambil data dari form input user
42   const query = 'INSERT INTO users (name, email, nim) VALUES (?, ?, ?)'; // Query SQL untuk menambahkan user baru ke database
43   connection.query(query, [name, email, nim], (err, result) => { // Eksekusi query dengan data user
44     if (err) throw err; // Jika ada kesalahan, tampilkan error
45     res.redirect('/'); // Setelah berhasil menambahkan, redirect kembali ke halaman utama
46   });
47 });
48
49 // Route untuk Update (mengambil data user berdasarkan ID untuk diedit)
50 app.get('/edit/:id', (req, res) => {
51   const { id } = req.params; // Mengambil ID user dari parameter URL
52   const query = 'SELECT * FROM users WHERE id = ?'; // Query SQL untuk mengambil data user berdasarkan ID
53   connection.query(query, [id], (err, result) => {
54     if (err) throw err; // Jika ada kesalahan, tampilkan error
55     res.render('edit', { user: result[0] }); // Render form edit dengan data user yang diambil
56   });
57 });
58
59 // Route untuk Update (mengirim perubahan user ke database)
60 app.post('/update/:id', (req, res) => {
61   const { name, email, nim } = req.body; // Mengambil data yang telah diubah dari form
62   const query = 'UPDATE users SET name = ?, email = ?, nim = ? WHERE id = ?'; // Query SQL untuk memperbarui data user di database
63   connection.query(query, [name, email, nim, req.params.id], (err, result) => {
64     if (err) throw err; // Jika ada kesalahan, tampilkan error
65     res.redirect('/'); // Setelah berhasil diupdate, redirect kembali ke halaman utama
66   });
67 });
68
69 // Route untuk Delete (menghapus user berdasarkan ID)
70 app.get('/delete/:id', (req, res) => {
71   const query = 'DELETE FROM users WHERE id = ?'; // Query SQL untuk menghapus user dari database berdasarkan ID
72   connection.query(query, [req.params.id], (err, result) => {
73     if (err) throw err; // Jika ada kesalahan, tampilkan error
74     res.redirect('/'); // Setelah berhasil dihapus, redirect kembali ke halaman utama
75   });
76 });
77
78 // Server yang fadli buat nantinya akan berjalan di port 3000
79 app.listen(3000, () => {
80   console.log('Server berjalan di port 3000. Bukalah akses melalui http://localhost:3000'); // Menjalankan server di port 3000
81 });
82
83
84
85

```

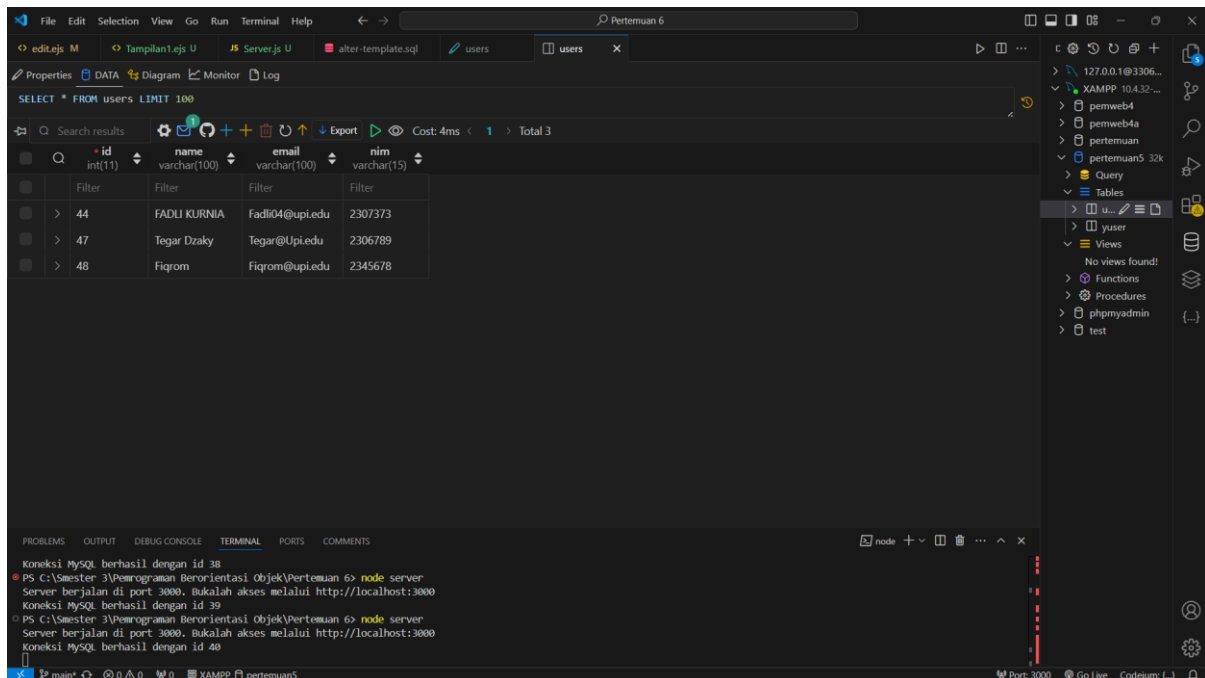
2. Berikut merupakan Server.js yang sudah saya buat yang nanti nya bis agita gunakan untuk memanggil link yang akan kita tuju tanpa menggunakan internet dan tanpa html diujungnya, disini saya membuat beberapa pemanggilan MySQL yaitu untuk menambahkan Mahasiswa

baru, mengedit nama mahasiswa baru, dan menghapus nama mahasiswa tersebut jika terjadi kesalahan dalam penginputan. Disini mahasiswa tersebut bisa memasukkan id untuk primarykey NAMA, NIM dan Gmail.

```
DDL Column Foreign Key Index Check

CREATE TABLE `users` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `name` varchar(100) DEFAULT NULL,
  `email` varchar(100) DEFAULT NULL,
  `nim` varchar(15) DEFAULT NULL,
  PRIMARY KEY (`id`)
) ENGINE = InnoDB AUTO_INCREMENT = 47 DEFAULT CHARSET = utf8mb4 COLLATE = utf8mb4_general_ci
```

3. Berikut merupakan code didalam tabel nya



The screenshot displays a database management interface with the 'users' table selected. The table structure is defined as follows:

id	name	email	nim
44	FADLI KURNIA	Fadli04@upi.edu	2307373
47	Tegar Dzaky	Tegar@Upi.edu	2306789
48	Fiqrom	Fiqrom@upi.edu	2345678

The interface also shows a terminal window at the bottom with the following output:

```
Koneksi MySQL berhasil dengan id 38
PS C:\Semester 3\Pemrograman Berorientasi Objek\Pertemuan 6> node server
Server berjalan di port 3000, bukalah akses melalui http://localhost:3000
Koneksi MySQL berhasil dengan id 39
PS C:\Semester 3\Pemrograman Berorientasi Objek\Pertemuan 6> node server
Server berjalan di port 3000, bukalah akses melalui http://localhost:3000
Koneksi MySQL berhasil dengan id 40
```

4. Ini merupakan tampilan dari tabel yang sudah saya buat supaya nanti Ketika melakukan pemanggilan MySQL sudah ada data yang terinput.

```

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>Daftar Pengguna</title>
7    <!-- Bootstrap CSS CDN -->
8    <link href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css" rel="stylesheet">
9  </head>
10 <style>
11 /* Custom Notification Styling */
12 .notification {
13   display: none;
14   margin-top: 10px;
15   padding: 10px;
16   background-color: #28a745;
17   color: white;
18   border-radius: 5px;
19 }
20 </style>
21 <body class="container mt-5">
22
23 <h1 class="text-center">Daftar Mahasiswa Sistem Informasi Kelautan</h1>
24
25 <div class="notification" id="notification">Mahasiswa sukses ditambahkan!</div>
26
27 <table class="table table-bordered mt-4">
28   <thead class="thead-dark">
29     <tr>
30       <th>NO</th>
31       <th>Nama</th>
32       <th>Email</th>
33       <th>NIM</th>
34       <th>Aksi</th>
35     </tr>
36   </thead>
37   <tbody>
38     <% users.forEach(pengguna => { %>
39       <tr>
40         <td><%= pengguna.id %></td>
41         <td><%= pengguna.name %></td>
42         <td><%= pengguna.email %></td>
43         <td><%= pengguna.nim %></td>
44         <td>
45           <a href="#" class="btn btn-warning btn-sm" onclick="confirmEdit('/edit/<%= pengguna.id %>');">Edit</a>
46           <a href="#" class="btn btn-danger btn-sm" onclick="confirmDelete('/delete/<%= pengguna.id %>');">Hapus</a>
47         </td>
48       </tr>
49     <% }) %>
50   </tbody>
51 </table>
52
53 <h2 class="mt-5">Tambah Pengguna Baru</h2>
54 <form action="/add" method="post" class="mt-4" onsubmit="showNotification()">
55   <div class="form-group">
56     <label for="name">Nama:</label>
57     <input type="text" class="form-control" id="name" name="name" required>
58   </div>
59   <div class="form-group">
60     <label for="email">Email:</label>
61     <input type="email" class="form-control" name="email" id="email" required>
62   </div>
63   <div class="form-group">
64     <label for="nim">NIM:</label>
65     <input type="text" class="form-control" name="nim" id="nim" required>
66   </div>
67   <button type="submit" class="btn btn-primary">Submit</button>
68 </form>
69
70 <!-- Bootstrap JS and dependencies (Popper.js and jQuery) -->
71 <script src="https://code.jquery.com/jquery-3.5.1.slim.min.js"></script>
72 <script src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.5.2/dist/umd/popper.min.js"></script>
73 <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js"></script>
74
75 <script>
76   function confirmDelete(url) {
77     if (confirm('Apakah kamu yakin ingin menghapus data ini?')) {
78       window.location.href = url;
79     }
80   }
81
82   function confirmEdit(url) {
83     if (confirm('Apakah kamu yakin ingin mengedit data ini?')) {
84       window.location.href = url;
85     }
86   }
87
88   function showNotification() {
89     var notification = document.getElementById('notification');
90     notification.style.display = 'block';
91     setTimeout(function() {
92       notification.style.display = 'none';
93     }, 3000);
94   }
95 </script>
96 </body>
97 </html>
98

```

8. Code program diatas merupakan tampilan 1 dari halaman website yang sudah saya buat disini saya menggunakan beberapa framework bootstrap untuk mempercantik tampilan website saya.

```

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>Daftar Pengguna</title>
7    <!-- Bootstrap CSS CDN -->
8    <link href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css" rel="stylesheet">
9  </head>
10 <style>
11 /* Custom Notification Styling */
12 .notification {
13   display: none;
14   margin-top: 10px;
15   padding: 10px;
16   background-color: #28a745;
17   color: white;
18   border-radius: 5px;
19 }
20 </style>
21 <body class="container mt-5">
22
23 <h1 class="text-center">Daftar Mahasiswa Sistem Informasi Kelautan</h1>
24
25 <div class="notification" id="notification">Mahasiswa sukses ditambahkan!</div>
26
27 <table class="table table-bordered mt-4">
28   <thead class="thead-dark">
29     <tr>
30       <th>NO</th>
31       <th>Nama</th>
32       <th>Email</th>
33       <th>NIM</th>
34       <th>Aksi</th>
35     </tr>
36   </thead>
37   <tbody>
38     <% users.forEach(pengguna => { %>
39       <tr>
40         <td><%= pengguna.id %></td>
41         <td><%= pengguna.name %></td>
42         <td><%= pengguna.email %></td>
43         <td><%= pengguna.phone %></td>
44         <td>
45           <a href="#" class="btn btn-warning btn-sm" onclick="confirmEdit('/edit/<%= pengguna.id %>');">Edit</a>
46           <a href="#" class="btn btn-danger btn-sm" onclick="confirmDelete('/delete/<%= pengguna.id %>');">Hapus</a>
47         </td>
48       </tr>
49     <% }) %>
50   </tbody>
51 </table>
52
53 <h2 class="mt-5">Tambah Pengguna Baru</h2>
54 <form action="/add" method="post" class="mt-4" onsubmit="showNotification()">
55   <div class="form-group">
56     <label for="name">Nama:</label>
57     <input type="text" class="form-control" id="name" name="name" required>
58   </div>
59   <div class="form-group">
60     <label for="email">Email:</label>
61     <input type="email" class="form-control" name="email" id="email" required>
62   </div>
63   <div class="form-group">
64     <label for="nim">NIM:</label>
65     <input type="text" class="form-control" name="nim" id="nim" required>
66   </div>
67   <button type="submit" class="btn btn-primary">Submit</button>
68 </form>
69
70 <!-- Bootstrap JS and dependencies (Popper.js and jQuery) -->
71 <script src="https://code.jquery.com/jquery-3.5.1.slim.min.js"></script>
72 <script src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.5.2/dist/umd/popper.min.js"></script>
73 <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js"></script>
74
75 <script>
76   function confirmDelete(url) {
77     if (confirm('Apakah kamu yakin ingin menghapus data ini?')) {
78       window.location.href = url;
79     }
80   }
81
82   function confirmEdit(url) {
83     if (confirm('Apakah kamu yakin ingin mengedit data ini?')) {
84       window.location.href = url;
85     }
86   }
87
88   function showNotification() {
89     var notification = document.getElementById('notification');
90     notification.style.display = 'block';
91     setTimeout(function() {
92       notification.style.display = 'none';
93     }, 3000);
94   }
95 </script>
96 </body>
97 </html>
98

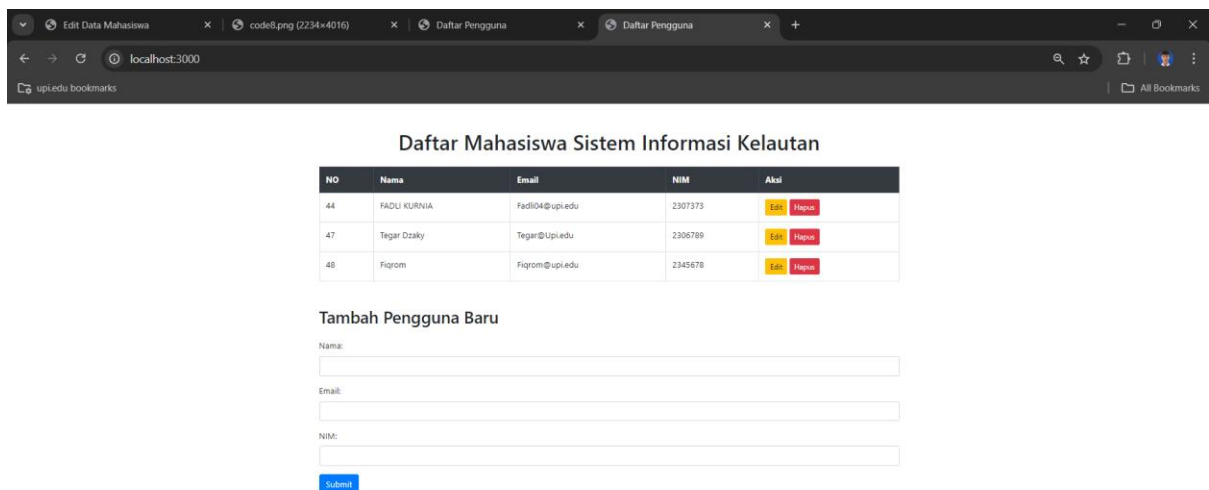
```

9. Ini merupakan Tampilan kedua website yang sudah saya buat yaitu menu edi yang menggunakan system “Crud”. Yang nantinya akan bisa mengedit data yang sudah dikirimkan.

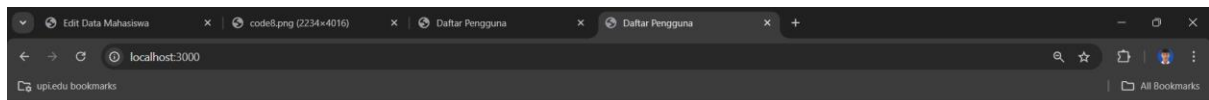
III. Penjelasan Website

```
30
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS COMMENTS
PS C:\Smester 3\Pemrograman Berorientasi Objek\Pertemuan 6> node Server.js
Server berjalan di port 3000. Bukalah akses melalui http://localhost:3000
Koneksi MySQL berhasil dengan id 41
```

1. Buka terminal dan ketik “node Server.js” yang nantinya akan memanggil link yang sudah saya sediakan di file Server.js tadi yaitu dibagian app.listen3000, dan console.log.



2. Ini merupakan halaman awal yang sudah saya buat dengan format Tampilan1 tadi. Disini saya menggunakan beberapa framework bootstrap untuk mempercantik tampilannya.



NO	Nama	Email	NIM	Aksi
44	FADU KURNIA	Fadi04@upi.edu	2307373	Edit Hapus
47	Tegar Dzaky	Tegar@Upi.edu	2306789	Edit Hapus
48	Fitrom	Fitrom@upi.edu	2345678	Edit Hapus

Tambah Pengguna Baru

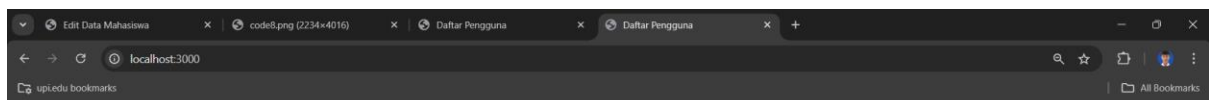
Nama:

Email:

NIM:

[Submit](#)

3. Bisa juga menambahkan mahasiswa baru.



NO	Nama	Email	NIM	Aksi
44	FADU KURNIA	Fadi04@upi.edu	2307373	Edit Hapus
47	Tegar Dzaky	Tegar@Upi.edu	2306789	Edit Hapus
48	Fitrom	Fitrom@upi.edu	2345678	Edit Hapus
50	Nelita	nelitamaharani54@gmail.com	23789779	Edit Hapus

Tambah Pengguna Baru

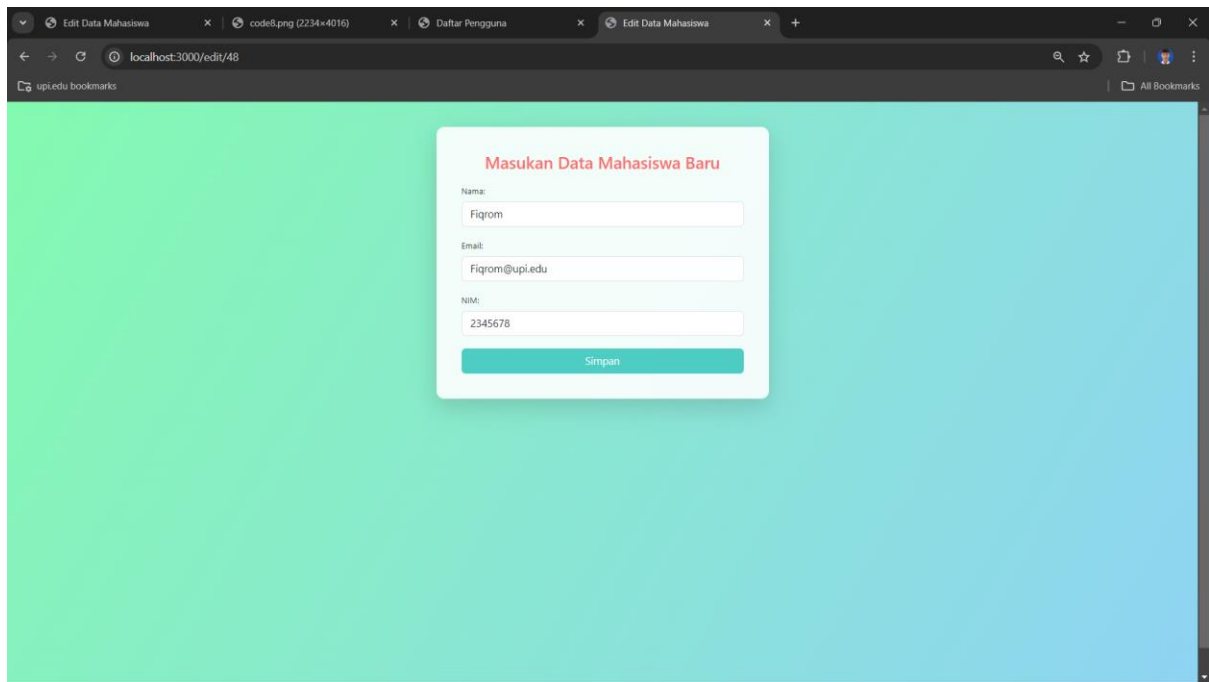
Nama:

Email:

NIM:

[Submit](#)

4. Berikut hasilnya. Kalian juga bisa menghapus data yang kalian input dengan klik tombol hapus.



5. Ini merupakan tampilan menu untuk meng edit data nya.

IV. KESIMPULAN

Setelah mempelajari CRUD lebih dalam, kita bisa melihat betapa pentingnya konsep ini dalam dunia pengembangan web. Meskipun kedengarannya sederhana - hanya Create, Read, Update, dan Delete - ternyata CRUD ini seperti fondasi rumah yang kokoh untuk hampir semua aplikasi web yang kita gunakan sehari-hari.

Yang menarik, meski teknologi terus berubah dengan cepat, CRUD tetap relevan. Ini seperti pisau Swiss dalam toolbox pengembang web - selalu berguna dalam berbagai situasi. Kita lihat bagaimana CRUD bisa menyesuaikan diri dengan tren terbaru seperti API RESTful atau cara-cara coding yang lebih modern.

Penerapan CRUD yang baik bukan cuma membuat aplikasi berjalan lancar, tapi juga membuat pengalaman pengguna lebih menyenangkan. Plus, ini membantu menjaga keamanan data dan membuat aplikasi lebih mudah dikembangkan di masa depan. Tapi perlu diingat, menerapkan CRUD di dunia web yang kompleks sekarang ini butuh pemikiran ekstra. Kita perlu memikirkan hal-hal seperti bagaimana menangani banyak pengguna sekaligus atau menjaga data tetap akurat.