

# **LAPORAN PRAKTIKUM TEKNOLOGI CLOUD COMPUTING**

## **IMPLEMENTAS CRUD DAN RESTFUL API UNTUK PENGEMBANGAN WEB SERVICE APLIKASI “NOTES”**



Disusun oleh

**Nama : Veyza Pradita Ardhia Putri**  
**NIM : 123220102**

**PROGRAM STUDI INFORMATIKA  
JURUSAN INFORMATIKA  
FAKULTAS TEKNIK INDUSTRI  
UNIVERSITAS PEMBANGUNAN NASIONAL “VETERAN”  
YOGYAKARTA  
2025**

## HALAMAN PENGESAHAN

### LAPORAN PRAKTIKUM

#### IMPLEMENTAS CRUD DAN RESTFUL API UNTUK PENGEMBANGAN WEB SERVICE APLIKASI “NOTES” PLUG IF-E

Disusun Oleh :

Veyza Pradita Ardhia Putri 123220102

Telah diperiksa dan disetujui oleh Asisten Praktikum.....

Pada tanggal : .....

**Menyetujui.**

**Asisten Praktikum**

**Asisten Praktikum**

**Berlyandhica Alam Febriwantoro.**  
NIM 123210060

**Rafli Iskandar Kavarera**  
NIM 123210131

## KATA PENGANTAR

Puji syukur kehadiran Tuhan Yang Maha Esa atas rahmat dan karunia-Nya sehingga laporan praktikum ini dapat diselesaikan dengan baik. Laporan ini disusun sebagai salah satu syarat dalam menyelesaikan tugas praktikum pembuatan web service dengan tema *Notes*.

Dalam proses penyusunan dan penyelesaian laporan ini, penulis mendapatkan banyak bimbingan, arahan, serta dukungan dari berbagai pihak. Oleh karena itu, penulis ingin menyampaikan terima kasih kepada Asisten Praktikum yang telah memberikan bimbingan, masukan, serta arahan yang sangat membantu selama pelaksanaan praktikum ini. Selain itu, penulis juga mengucapkan terima kasih kepada semua pihak yang telah memberikan dukungan dalam penyelesaian tugas ini.

Penulis menyadari bahwa laporan ini masih jauh dari sempurna. Oleh karena itu, kritik dan saran yang membangun sangat diharapkan guna perbaikan di masa yang akan datang. Semoga laporan ini dapat memberikan manfaat bagi pembaca serta menjadi referensi yang berguna bagi pengembangan ilmu pengetahuan dan teknologi.

Yogyakarta, 02 Maret 2025

Penulis

## DAFTAR ISI

<b>HALAMAN PENGESAHAN.....</b>	<b>2</b>
<b>HALAMAN PERSETUJUAN.....</b>	<b>2</b>
<b>KATA PENGANTAR.....</b>	<b>3</b>
<b>DAFTAR ISI.....</b>	<b>4</b>
<b>DAFTAR GAMBAR.....</b>	<b>5</b>
<b>BAB I</b>	
<b>PENDAHULUAN.....</b>	<b>1</b>
1.1 Latar Belakang.....	1
1.2 Rumusan Masalah.....	2
1.3 Tujuan.....	2
1.4 Manfaat.....	3
<b>BAB II</b>	
<b>TINJAUAN LITERATUR.....</b>	<b>5</b>
<b>BAB III</b>	
<b>METODOLOGI.....</b>	<b>8</b>
3.1 Analisis Permasalahan.....	8
3.2 Perancangan Solusi.....	8
<b>BAB IV</b>	
<b>HASIL DAN PEMBAHASAN.....</b>	<b>9</b>
4.1 Hasil.....	9
4.2 Pembahasan.....	10
<b>BAB V</b>	
<b>PENUTUP.....</b>	<b>19</b>
5.1 Kesimpulan.....	19
5.2 Saran.....	19
<b>DAFTAR PUSTAKA.....</b>	<b>21</b>

## **DAFTAR GAMBAR**

Gambar 4.1.1 Halaman Tampilan Seluruh Notes.....	9
Gambar 4.1.2 Tampilan halaman AddNotes.....	10
Gambar 4.1.3 Halaman EditNotes.....	10

# BAB I

## PENDAHULUAN

### 1.1 Latar Belakang

Dalam era digital saat ini, kebutuhan akan sistem pencatatan yang efisien semakin meningkat. Banyak individu maupun organisasi memerlukan cara yang lebih praktis untuk menyimpan dan mengelola catatan mereka secara digital. Metode pencatatan manual seperti menggunakan buku atau kertas sering kali memiliki keterbatasan, seperti risiko hilang, rusak, atau sulit diakses kembali. Oleh karena itu, pengembangan aplikasi berbasis web untuk pencatatan (*Notes*) menjadi solusi yang relevan untuk meningkatkan kemudahan dan efisiensi dalam menyimpan informasi.

Untuk memenuhi kebutuhan tersebut, pengembangan sistem berbasis *web service* menjadi salah satu solusi yang dapat diterapkan. Dengan memanfaatkan konsep *RESTful API*, sistem ini memungkinkan pengelolaan data catatan secara terstruktur melalui operasi *CRUD* (Create, Read, Update, Delete). Pada bagian *back-end*, sistem akan mengelola penyimpanan data dan menyediakan layanan berbasis *API* yang dapat diakses oleh *front-end*. Sementara itu, *front-end* berperan sebagai antarmuka pengguna yang memudahkan interaksi dengan sistem, sehingga pengguna dapat menambahkan, mengedit, menghapus, serta melihat catatan mereka dengan lebih mudah.

Pemilihan pendekatan ini didasarkan pada beberapa keunggulan utama. Pertama, dengan menerapkan konsep *RESTful API*, integrasi antara *front-end* dan *back-end* dapat berjalan dengan lebih fleksibel dan efisien. Kedua, sistem berbasis web memungkinkan akses dari berbagai perangkat tanpa perlu instalasi tambahan, sehingga memberikan kenyamanan bagi pengguna. Ketiga, solusi ini dapat dikembangkan lebih lanjut dengan fitur tambahan seperti pencarian, pengkategorian, atau sinkronisasi data dengan layanan penyimpanan lainnya. Dengan demikian, implementasi *web service* untuk aplikasi *Notes* ini diharapkan

dapat memberikan pengalaman pencatatan digital yang lebih terorganisir, aman, dan mudah digunakan.

## **1.2 Rumusan Masalah**

Berdasarkan latar belakang yang telah diuraikan, pengembangan aplikasi Notes berbasis web service memerlukan sistem yang dapat menyimpan, mengelola, dan menampilkan catatan secara efisien. Untuk mewujudkan hal tersebut, terdapat beberapa permasalahan utama yang perlu dirumuskan, yaitu:

1. Bagaimana merancang dan mengembangkan back-end yang mampu menangani operasi CRUD (Create, Read, Update, Delete) dengan menerapkan konsep RESTful API agar dapat mengelola data catatan dengan baik?
2. Bagaimana membangun front-end berbasis web yang dapat berkomunikasi dengan back-end secara efektif serta memberikan antarmuka yang intuitif dan mudah digunakan oleh pengguna?
3. Bagaimana memastikan bahwa front-end dan back-end dapat terhubung dengan baik sehingga seluruh fitur pencatatan dapat berfungsi dengan optimal?

Rumusan masalah ini menjadi dasar dalam perancangan dan implementasi aplikasi Notes agar dapat berfungsi sesuai dengan kebutuhan pengguna serta memenuhi standar pengembangan web service yang baik.

## **1.3 Tujuan**

Tujuan dari pengembangan *web service* untuk aplikasi *Notes* ini adalah sebagai berikut:

1. Mengembangkan *back-end* yang mampu menangani operasi *CRUD* (Create, Read, Update, Delete) dengan menerapkan konsep *RESTful API* sehingga data catatan dapat dikelola dengan baik.

2. Membangun *front-end* berbasis web yang dapat berkomunikasi dengan *back-end* secara efektif serta menyediakan antarmuka yang intuitif dan mudah digunakan oleh pengguna.
3. Memastikan integrasi yang baik antara *front-end* dan *back-end* sehingga seluruh fitur pencatatan dapat berfungsi secara optimal dan sesuai dengan standar pengembangan *web service*.

Tujuan ini diharapkan dapat menghasilkan sistem pencatatan digital berbasis *web service* yang dapat digunakan untuk mencatat, menyimpan, mengedit, dan menghapus catatan dengan mudah serta memberikan pengalaman pengguna yang lebih efisien dan praktis.

#### **1.4 Manfaat**

Pengembangan *web service* untuk aplikasi *Notes* ini memberikan berbagai manfaat, baik secara teknis maupun non-teknis. Secara teknis, aplikasi ini memungkinkan pengguna untuk menyimpan, mengedit, menghapus, dan mengelola catatan secara digital dengan lebih aman dan terstruktur. Dengan menerapkan operasi *CRUD* berbasis *RESTful API*, komunikasi antara *front-end* dan *back-end* dapat berjalan secara optimal, sehingga meningkatkan efisiensi dalam pengelolaan data. Selain itu, karena berbasis web, aplikasi ini dapat diakses dari berbagai perangkat tanpa memerlukan instalasi tambahan, memberikan fleksibilitas lebih bagi pengguna. Antarmuka yang intuitif dan responsif juga dirancang untuk meningkatkan pengalaman pengguna, sehingga pencatatan menjadi lebih praktis dan mudah digunakan.

Secara non-teknis, aplikasi ini membantu pengguna dalam mengorganisir catatan secara lebih efisien dibandingkan dengan metode pencatatan manual. Dengan sistem digital, risiko kehilangan atau kerusakan catatan yang sering terjadi pada pencatatan konvensional dapat diminimalisir. Selain itu, aplikasi ini juga berkontribusi dalam meningkatkan produktivitas pengguna dengan memberikan kemudahan dalam mencatat dan mengakses informasi kapan saja dan di mana



saja. Dengan adanya solusi ini, diharapkan pencatatan digital dapat menjadi lebih terorganisir, aman, dan fleksibel sesuai dengan kebutuhan pengguna.

## BAB II

### TINJAUAN LITERATUR

#### 2.1 Web Service

Web service adalah suatu layanan yang memungkinkan komunikasi antara aplikasi yang berbeda melalui jaringan, menggunakan protokol standar seperti HTTP dan format pertukaran data seperti JSON atau XML. Web service memungkinkan aplikasi *front-end* dan *back-end* untuk saling berinteraksi tanpa harus terikat pada satu platform tertentu. Salah satu implementasi web service yang umum digunakan adalah *RESTful API*, yang mengadopsi prinsip *Representational State Transfer* (REST) untuk memungkinkan pertukaran data secara efisien dan terstruktur.

#### 2.2 RESTful API

RESTful API adalah salah satu jenis web service yang memanfaatkan protokol HTTP untuk mengelola sumber daya dalam suatu sistem. API ini dirancang berdasarkan enam prinsip utama REST, yaitu *statelessness*, *client-server architecture*, *cacheability*, *layered system*, *code on demand* (opsional), dan *uniform interface*. Dalam implementasinya, RESTful API memungkinkan penggunaan metode HTTP seperti *GET* untuk mengambil data, *POST* untuk menambahkan data, *PUT* untuk memperbarui data, dan *DELETE* untuk menghapus data. Dengan pendekatan ini, aplikasi *Notes* dapat memiliki sistem *back-end* yang modular dan fleksibel untuk dikelola serta dikembangkan lebih lanjut.

#### 2.3 CRUD (Create, Read, Update, Delete)

CRUD merupakan konsep dasar dalam pengelolaan data yang diterapkan dalam sistem berbasis database dan web service.

- **Create:** Operasi yang digunakan untuk menambahkan data baru ke dalam sistem.
- **Read:** Operasi yang memungkinkan pengguna mengambil atau melihat data yang telah tersimpan.

- **Update:** Operasi yang digunakan untuk mengubah atau memperbarui data yang ada dalam sistem.
- **Delete:** Operasi yang digunakan untuk menghapus data yang tidak lagi diperlukan.

Dalam aplikasi *Notes*, operasi CRUD diterapkan untuk memungkinkan pengguna menambahkan, membaca, memperbarui, dan menghapus catatan mereka secara digital. Dengan adanya fitur CRUD, sistem menjadi lebih dinamis dan dapat menyesuaikan dengan kebutuhan pengguna.

## 2.4 Front-end dan Back-end

Dalam pengembangan aplikasi berbasis web, sistem umumnya terdiri dari dua bagian utama, yaitu *front-end* dan *back-end*.

- **Front-end** adalah bagian dari aplikasi yang berinteraksi langsung dengan pengguna. Biasanya, *front-end* dikembangkan menggunakan teknologi seperti HTML, CSS, dan JavaScript, serta berbagai framework seperti React, Vue.js, atau Angular. Dalam aplikasi *Notes*, *front-end* bertanggung jawab dalam menampilkan antarmuka pengguna yang intuitif dan responsif agar pengguna dapat mengelola catatan dengan mudah.
- **Back-end** adalah bagian dari aplikasi yang berfungsi sebagai server untuk menangani proses bisnis dan manajemen data. Back-end biasanya dikembangkan menggunakan bahasa pemrograman seperti Node.js, Python, PHP, atau Java, dan terhubung dengan database untuk menyimpan data. Dalam aplikasi *Notes*, *back-end* bertanggung jawab untuk mengelola penyimpanan catatan serta memastikan integrasi data dengan *front-end* melalui RESTful API.

## 2.5 Database

Database adalah komponen penting dalam aplikasi *Notes* yang digunakan untuk menyimpan catatan pengguna secara permanen. Sistem manajemen database (DBMS) yang digunakan dapat berupa MySQL. Dalam konteks aplikasi *Notes*,

database berfungsi untuk menyimpan data catatan secara terstruktur sehingga pengguna dapat dengan mudah mengakses, memperbarui, atau menghapus catatan mereka sesuai kebutuhan.

## **BAB III**

### **METODOLOGI**

#### **3.1 Analisis Permasalahan**

Dalam tugas praktikum ini, penugasan yang diberikan adalah mengembangkan sebuah *web service* dengan tema *Notes* yang mencakup sistem *back-end* dan *front-end* yang saling terhubung. *Back-end* harus menerapkan konsep *CRUD* (Create, Read, Update, Delete) dan menggunakan pendekatan *RESTful API* agar dapat berkomunikasi dengan *front-end* secara efisien. Sementara itu, *front-end* harus mampu menampilkan antarmuka pengguna yang intuitif serta memanfaatkan API yang disediakan oleh *back-end* untuk mengelola catatan secara dinamis.

Masalah utama yang dihadapi dalam penugasan ini adalah bagaimana membangun sistem pencatatan digital yang memungkinkan pengguna untuk membuat, membaca, memperbarui, dan menghapus catatan dengan mudah dan efisien. Selain itu, sistem harus dirancang agar memiliki integrasi yang baik antara *front-end* dan *back-end*, sehingga komunikasi data berjalan lancar dan responsif.

#### **3.2 Perancangan Solusi**


Dalam rangka menyelesaikan permasalahan yang telah diidentifikasi, solusi yang diusulkan adalah membangun sebuah aplikasi pencatatan digital berbasis web dengan arsitektur *client-server* yang terdiri dari *front-end* dan *back-end*. Perancangan solusi ini mencakup beberapa aspek utama, yaitu arsitektur sistem, desain *database*, pembuatan *RESTful API*, serta pengembangan antarmuka pengguna.

## BAB IV

### HASIL DAN PEMBAHASAN

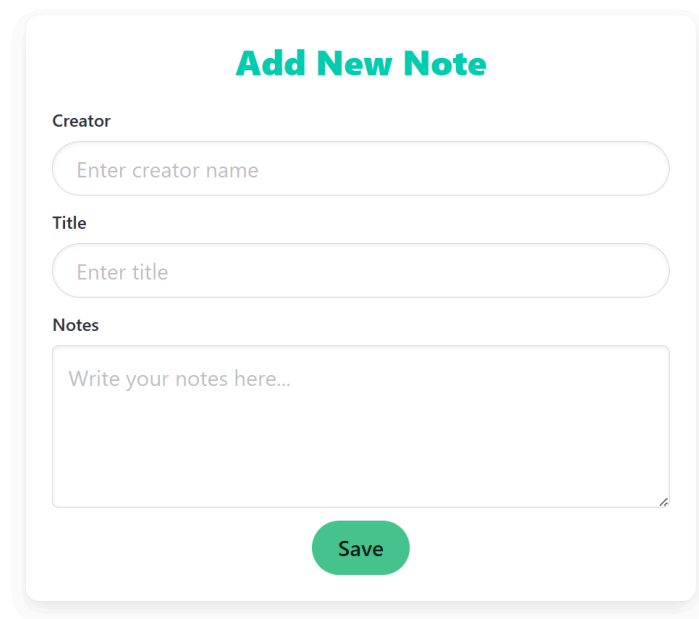
#### 4.1 Hasil

Berikut ditampilkan hasil dari dari program yang sudah dibuat berupa halaman untuk menampilkan seluruh notes, halaman untuk menambahkan notes dan halaman untuk edit notes.



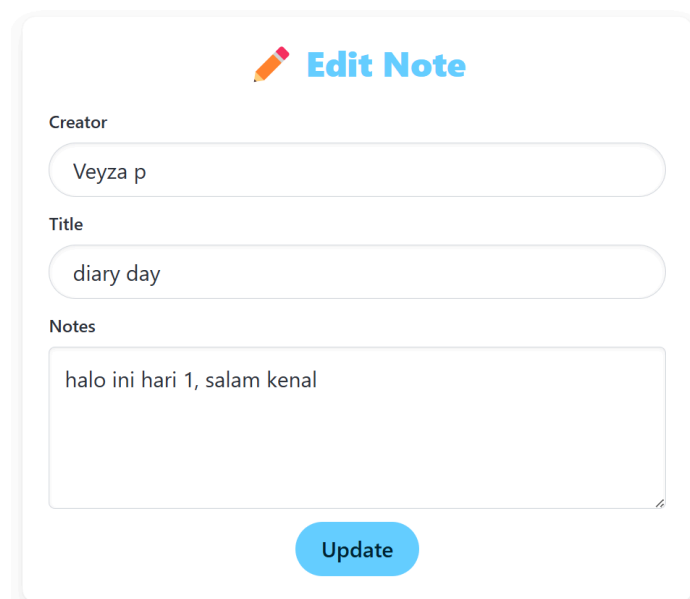
Notes				
Add New				
No	Creator	Title	Notes	Actions
1	Veyza p	diary day	halo ini hari 1, salam kenal	<a href="#">Edit</a> <a href="#">Delete</a>
2	Putri	hari ini ngapain ya	aku hari ini bla nla	<a href="#">Edit</a> <a href="#">Delete</a>
3	Veyyy	diary day	halo ini hari 1, salam kenal	<a href="#">Edit</a> <a href="#">Delete</a>

**Gambar 4.1.1 Halaman Tampilan Seluruh Notes**



The image shows a web form titled "Add New Note" in a teal font. It contains three input fields: "Creator" with the placeholder "Enter creator name", "Title" with the placeholder "Enter title", and "Notes" with the placeholder "Write your notes here...". A green "Save" button is located at the bottom right of the form.

**Gambar 4.1.2 Tampilan halaman AddNotes**



The image shows a web form titled "Edit Note" with a pencil icon. It contains three input fields: "Creator" with the value "Veyza p", "Title" with the value "diary day", and "Notes" with the value "halo ini hari 1, salam kenal". A blue "Update" button is located at the bottom right of the form.

**Gambar 4.1.3 Halaman EditNotes**

## 4.2 Pembahasan

Dalam proses pengembangan sistem, beberapa komponen utama dikembangkan untuk memenuhi kebutuhan fungsionalitas CRUD pada RESTful API. Berikut adalah pembahasan terkait implementasi dari masing-masing fungsionalitas.

### 4.2.1 Implementasi Backend

#### - Konfigurasi Database

```
import {Sequelize} from "sequelize";
const db = new Sequelize ('notes','root','',{
  host: 'localhost',
  dialect: 'mysql'
});

export default db;
```

#### - Implementasi Controller API

```
import { where } from "sequelize";
import User from "../models/UserModel.js";

export const getUsers = async(req, res) =>{
  try {
    const response = await User.findAll();
    res.status(200).json(response);
  } catch (error) {
    console.log(error.message);
  }
}

export const getUsersById = async(req, res) =>{
  try {
    const response = await User.findOne({
      where:{
        id: req.params.id
      }
    });
    res.status(200).json(response);
  } catch (error) {
    console.log(error.message);
  }
}

export const createUser = async(req, res) =>{
  try {
    await User.create(req.body);
    res.status(201).json({msg:"User Created"});
  } catch (error) {
    console.log(error.message);
  }
}

export const updateUser = async(req, res) =>{
  try {
    await User.update(req.body,{
      where:{
        id: req.params.id
      }
    });
    res.status(200).json({msg:"User Updated"});
  } catch (error) {
    console.log(error.message);
  }
}
```



```

}

export const deleteUser = async(req, res) =>{
  try {
    await User.destroy({
      where:{
        id: req.params.id
      }
    });
    res.status(200).json({msg:"User Deleted"});
  } catch (error) {
    console.log(error.message);
  }
}

```

## - Implementasi Routing

```

import express from "express";
import {
  createUser,
  deleteUser,
  getUsers,
  getUsersById,
  updateUser,
} from "../controller/UserController.js";

const router = express.Router();
router.get('/users', getUsers);
router.get('/users/:id', getUsersById);
router.post('/users', createUser);
router.patch('/users/:id', updateUser);
router.delete('/users/:id', deleteUser);

export default router;

```

## - Pemodelan Tabel

```

import { Sequelize } from "sequelize";
import db from "../config/Database.js";

const {DataTypes} = Sequelize;

const User = db.define('users',{
  creator : DataTypes.STRING,
  title : DataTypes.STRING,
  notes:DataTypes.STRING
},{
  freezeTableName:true
});

export default User;

(async ()=> {

```

```
    await db.sync();
  }) ();
```

#### 4.2.2 Implementasi Frontend

- Konfigurasi Routing Frontend

```
import { BrowserRouter, Routes, Route } from "react-router-dom";
import UserList from "../components/UserList";
import AddUser from "../components/AddUser";
import EditUser from "../components/EditUser";
import React from "react";
function App() {
  return (
    <BrowserRouter>
      <Routes>
        <Route path="/" element={<UserList/>} />
        <Route path="add" element={<AddUser/>} />
        <Route path="edit/:id" element={<EditUser/>} />
      </Routes>
    </BrowserRouter>
  );
}
export default App;
```

- Render aplikasi React ke dalam HTML

```
import React from 'react';
import ReactDOM from 'react-dom';
import App from './App';
import "bulma/css/bulma.css";

ReactDOM.render(<App />, document.getElementById('root'));
```

- Membuat bagian UserList untuk menampilkan seluruh Notes

```
import React, { useState, useEffect } from "react";
import axios from "axios";
import { Link } from "react-router-dom";
const UserList = () => {
  const [users, setUser] = useState([]);

  useEffect(() => {
    getUsers();
  }, []);

  const getUsers = async () => {
    const response = await
    axios.get('http://localhost:5000/users');
    setUser(response.data);
  };

  const deleteUser = async (id) => {
```

```

    try {
      await axios.delete(`http://localhost:5000/users/${id}`);
      getUsers();
    } catch (error) {
      console.log(error);
    }
  }

  return (
    <div className="columns mt-5 is-centered">
      <div className="column is-half">
        <h1 className="title has-text-centered has-text-primary">
Notes</h1>
        <div className="is-flex is-justify-content-space-between
mb-4">
          <Link to={`add`} className="button is-success
is-rounded">Add New</Link>
        </div>
        <table className="table is-striped is-fullwidth
is-hoverable">
          <thead>
            <tr className="has-background-primary-light">
              <th className="has-text-centered">No</th>
              <th>Creator</th>
              <th>Title</th>
              <th>Notes</th>
              <th className="has-text-centered">Actions</th>
            </tr>
          </thead>
          <tbody>
            {users.map((user, index) => (
              <tr key={user.id}>
                <td className="has-text-centered">{index + 1}</td>
                <td>{user.creator}</td>
                <td>{user.title}</td>
                <td>{user.notes}</td>
                <td className="is-flex is-justify-content-center">
                  <Link to={`edit/${user.id}`} className="button
is-small is-info is-light is-rounded mr-2">Edit</Link>
                  <button
                    onClick={() => deleteUser(user.id)}
                    className="button is-small is-danger is-light
is-rounded"
                  >
                    Delete
                  </button>
                </td>
              </tr>
            ))}
          </tbody>
        </table>
      </div>
    </div>
  );
};
export default UserList;

```

- Membuat AddNotes untuk menambahkan notes

```

import React, { useState } from 'react';
import axios from 'axios';
import { useNavigate } from "react-router-dom";

function AddUser() {
  const [creator, setCreator] = useState("");
  const [title, setTitle] = useState("");
  const [notes, setNotes] = useState("");
  const navigate = useNavigate();

  const saveUser = async (e) =>{
    e.preventDefault();
    try {
      await axios.post('http://localhost:5000/users',{
        creator,
        title,
        notes
      });
      navigate("/");
    } catch (error) {
      console.log(error);
    }
  };

  return (
    <div className="columns mt-5 is-centered">
      <div className="column is-half">
        <div className="box p-5">
          <h1 className="title has-text-centered has-text-primary">
Add New Note</h1>
          <form onSubmit={saveUser}>
            <div className="field">
              <label className="label">Creator</label>
              <div className="control">
                <input
                  type="text"
                  className="input is-medium is-rounded"
                  value={creator}
                  onChange={ (e) => setCreator(e.target.value) }
                  placeholder="Enter creator name"
                  required
                />
              </div>
            </div>

            <div className="field">
              <label className="label">Title</label>
              <div className="control">
                <input
                  type="text"
                  className="input is-medium is-rounded"
                  value={title}
                  onChange={ (e) => setTitle(e.target.value) }
                  placeholder="Enter title"
                  required
                />
              </div>
            </div>
          </form>
        </div>
      </div>
    </div>
  );
}

```

```

        <div className="field">
          <label className="label">Notes</label>
          <div className="control">
            <textarea
              className="textarea is-medium is-rounded"
              value={notes}
              onChange={ (e) => setNotes (e.target.value) }
              rows="4"
              placeholder="Write your notes here..."
              required
            ></textarea>
          </div>
        </div>

        <div className="field has-text-centered">
          <button type="submit" className="button is-success
is-medium is-rounded px-5">
            Save
          </button>
        </div>
      </form>
    </div>
  </div>
</div>

)
}
export default AddUser

```

## - Membuat EditNotes untuk mengedit Notes

```

import React, { useState } from 'react';
import axios from 'axios';
import { useNavigate } from "react-router-dom";

function AddUser() {
  const [creator, setCreator] = useState("");
  const [title, setTitle] = useState("");
  const [notes, setNotes] = useState("");
  const navigate = useNavigate();

  const saveUser = async (e) =>{
    e.preventDefault();
    try {
      await axios.post('http://localhost:5000/users',{
        creator,
        title,
        notes
      });
      navigate("/");
    } catch (error) {
      console.log(error);
    }
  };
}

```

```

return (
  <div className="columns mt-5 is-centered">
    <div className="column is-half">
      <div className="box p-5">
        <h1 className="title has-text-centered has-text-primary">
Add New Note</h1>
        <form onSubmit={saveUser}>
          <div className="field">
            <label className="label">Creator</label>
            <div className="control">
              <input
                type="text"
                className="input is-medium is-rounded"
                value={creator}
                onChange={ (e) => setCreator(e.target.value)}
                placeholder="Enter creator name"
                required
              />
            </div>
          </div>

          <div className="field">
            <label className="label">Title</label>
            <div className="control">
              <input
                type="text"
                className="input is-medium is-rounded"
                value={title}
                onChange={ (e) => setTitle(e.target.value)}
                placeholder="Enter title"
                required
              />
            </div>
          </div>

          <div className="field">
            <label className="label">Notes</label>
            <div className="control">
              <textarea
                className="textarea is-medium is-rounded"
                value={notes}
                onChange={ (e) => setNotes(e.target.value)}
                rows="4"
                placeholder="Write your notes here..."
                required
              ></textarea>
            </div>
          </div>

          <div className="field has-text-centered">
            <button type="submit" className="button is-success
is-medium is-rounded px-5">
              Save
            </button>
          </div>
        </form>
      </div>
    </div>
  </div>
)

```

```
    </div>

  )
}

export default AddUser
```

- Button untuk menghapus notes

```
<button
      onClick={() => deleteUser(user.id)}
      className="button is-small is-danger is-light
is-rounded"
    >
      Delete
    </button>
```

Diatas dilampirkan beberapa *source code* program. Untuk lebih lengkapnya dapat di lihat pada link github berikut:

[https://github.com/Veyzaputri/Tugas1\\_Prak\\_TCC.git](https://github.com/Veyzaputri/Tugas1_Prak_TCC.git)

## **BAB V**

### **PENUTUP**

#### **5.1 Kesimpulan**

Dari pembahasan ini, dapat disimpulkan bahwa sistem yang dikembangkan telah berhasil mencapai tujuan utama dalam membangun RESTful API untuk pengelolaan catatan. Metode dan praktik perancangan menggunakan arsitektur MVC dan teknologi berbasis Node.js, Express.js, serta Sequelize ORM terbukti efektif dalam membangun API yang modular dan scalable.

Namun, masih ada beberapa keterbatasan yang perlu diperbaiki, terutama dalam aspek keamanan, validasi input, dan efisiensi database. Pemahaman baru yang diperoleh dari proyek ini mencakup pentingnya middleware dalam RESTful API, efisiensi penggunaan ORM, serta kaitannya dengan teori yang telah dikemukakan dalam literatur terkait.

#### **5.2 Saran**

Berdasarkan hasil yang telah diperoleh dalam pengembangan sistem RESTful API untuk pengelolaan catatan, terdapat beberapa rekomendasi yang dapat dijadikan sebagai langkah perbaikan dan pengembangan lebih lanjut. Saran ini bertujuan untuk meningkatkan kualitas sistem, memperbaiki kekurangan yang ditemukan, serta memastikan bahwa sistem dapat digunakan secara lebih optimal di masa mendatang.

1. Meningkatkan Keamanan API dengan autentikasi JWT, validasi input, dan perlindungan dari serangan seperti SQL Injection dan XSS.
2. Optimasi Database dengan indexing, caching, dan penggunaan raw query untuk meningkatkan performa.
3. Peningkatan Error Handling dan Logging menggunakan Winston atau Morgan agar error lebih mudah dianalisis.



4. pengujian dengan Jest atau Mocha.
5. Pengembangan Fitur Tambahan seperti kategori catatan, notifikasi pengingat, dan sistem multi-user untuk meningkatkan fungsionalitas.

## DAFTAR PUSTAKA

- Bell, J. (2017). *Machine learning: Hands-on for developers and technical professionals*. Wiley.
- Connolly, T., & Begg, C. (2015). *Database systems: A practical approach to design, implementation, and management* (6th ed.). Pearson.
- Fielding, R. T. (2000). *Architectural styles and the design of network-based software architectures* (Doctoral dissertation). University of California, Irvine.
- Flanagan, D. (2020). *JavaScript: The definitive guide* (7th ed.). O'Reilly Media.
- Stallings, W. (2017). *Cryptography and network security: Principles and practice* (7th ed.). Pearson.
- Tilkov, S., & Vinoski, S. (2010). Node.js: Using JavaScript to build high-performance network applications. *IEEE Internet Computing*, 14(6), 80-83. <https://doi.org/10.1109/MIC.2010.136>