LAPORAN PRAKTIKUM TEKNOLOGI CLOUD COMPUTING

WEB SERVICE (BASIC REST)

Disusun oleh

Nama : Muhammad Hafizh Maulana

NIM : 123210194

PROGRAM STUDI TEKNIK INFORMATIKA

JURUSAN INFORMATIKA

FAKULTAS TEKNIK INDUSTRI

UNIVERSITAS PEMBANGUNAN NASIONAL "VETERAN"

YOGYAKARTA

2025

HALAMAN PENGESAHAN

LAPORAN PRAKTIKUM

WEB SERVICE (BASIC REST)

Disusun Oleh :

Muhammad Hafizh Maulana 123210194

Telah diperiksa dan disetujui oleh Asisten Praktikum…… ……

Pada tanggal : ……

.

Menyetujui.

Asisten Praktikum	**Asisten Praktikum**
Muhammad Rafli	**Raditya Haikal**
NIM 123210078	**NIM 123210062**

KATA PENGANTAR

Puji dan syukur penulis panjatkan ke hadirat Tuhan Yang Maha Esa atas rahmat dan karunia-Nya sehingga laporan praktikum ini dapat diselesaikan dengan baik. Laporan ini disusun sebagai salah satu bentuk pertanggungjawaban dalam menyelesaikan tugas praktikum Praktikum Cloud Computing.

Dalam penyusunan laporan ini, penulis mendapat banyak bantuan, bimbingan, serta dukungan dari berbagai pihak. Oleh karena itu, penulis ingin mengucapkan terima kasih yang sebesar-besarnya kepada Bapak Rudi Cahyadi S.Si., M.T. selaku Dosen Teori yang telah memberikan ilmu dan bimbingan dalam memahami konsep dasar cloud computing, serta kepada Muhammad Rafli dan Raditya Haikal selaku Asisten Praktikum yang telah memberikan arahan, masukan, serta bantuan teknis selama proses pengerjaan proyek ini.

Penulis menyadari bahwa laporan ini masih jauh dari kesempurnaan. Oleh karena itu, penulis sangat mengharapkan kritik dan saran yang membangun demi perbaikan di masa mendatang. Semoga laporan ini dapat bermanfaat bagi pembaca serta menjadi referensi dalam memahami konsep pembuatan web service dan bacic rest.

Yogyakarta, 5 Maret 2025

Penulis

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

BAB I
PENDAHULUAN

1.1 Latar Belakang

Dalam era digital saat ini, kebutuhan akan pencatatan dan pengelolaan informasi semakin meningkat. Banyak individu dan organisasi memerlukan sistem yang dapat menyimpan, mengelola, serta memperbarui catatan secara efisien. Namun, banyaknya data yang harus dikelola sering kali menjadi tantangan tersendiri, terutama jika masih menggunakan metode manual atau sistem yang tidak terstruktur. Hal ini dapat menyebabkan hilangnya data, duplikasi informasi, serta kesulitan dalam mengakses catatan yang dibutuhkan. Oleh karena itu, diperlukan sebuah solusi teknologi yang dapat mengatasi permasalahan tersebut dengan cara yang lebih modern, terstruktur, dan dapat diakses dengan lebih mudah.

Untuk mengatasi permasalahan tersebut, teknologi dalam bidang pengembangan perangkat lunak menawarkan solusi yang efektif, yaitu dengan membangun sistem web service berbasis CRUD (Create, Read, Update, Delete). Web service ini memungkinkan pengguna untuk melakukan pencatatan, pengelolaan, serta pemeliharaan catatan secara efisien dengan menggunakan RESTful API sebagai media komunikasi antara server dan klien. Dengan adanya sistem ini, proses penyimpanan dan pengelolaan informasi menjadi lebih terstruktur dan mudah diakses dari berbagai platform. Selain itu, integrasi dengan berbagai sistem lain juga dapat dilakukan untuk meningkatkan fungsionalitas dari sistem yang dikembangkan.

Teknologi yang dipilih dalam pengembangan sistem ini adalah web service berbasis REST API yang menggunakan bahasa pemrograman JavaScript dan database MySQL. REST API dipilih karena kemampuannya dalam menyediakan komunikasi yang ringan, fleksibel, dan kompatibel dengan berbagai teknologi lainnya. Selain itu, penggunaan MySQL sebagai sistem manajemen basis data memastikan data yang disimpan memiliki integritas yang baik serta dapat diakses dengan cepat. Dengan kombinasi teknologi ini, sistem web service yang dikembangkan dapat menjadi solusi yang tepat untuk permasalahan pencatatan dan pengelolaan informasi secara efisien dan efektif. Ditambah lagi, kemudahan implementasi serta skalabilitas dari teknologi yang

dipilih menjadikannya pilihan yang tepat untuk kebutuhan sistem yang akan berkembang di masa mendatang.

1.2 Rumusan Masalah

Berdasarkan latar belakang yang telah diuraikan, permasalahan yang muncul dalam pencatatan dan pengelolaan informasi dapat dirumuskan sebagai berikut:

1. Bagaimana membangun sebuah web service berbasis CRUD yang dapat mempermudah pengguna dalam pencatatan dan pengelolaan catatan?
2. Bagaimana penerapan teknologi REST API dalam membangun sistem pencatatan yang efisien?

1.3 Tujuan

Tujuan dari penelitian ini adalah:

1. Mengembangkan sistem web service berbasis CRUD yang memungkinkan pengguna untuk membuat, membaca, memperbarui, dan menghapus catatan secara efisien.
2. Menerapkan teknologi RESTful API untuk memastikan komunikasi antara server dan klien berjalan dengan optimal serta mempermudah integrasi dengan sistem lain.
3. Menghasilkan solusi berbasis web yang mudah digunakan, aman, dan dapat diakses dari berbagai platform guna meningkatkan efisiensi dalam pencatatan dan pengelolaan informasi.

1.4 Manfaat

Pembuatan web service Notes ini diharapkan dapat memberikan manfaat sebagai berikut:

1. Secara teknis, sistem ini memberikan solusi efektif dalam pengelolaan data dengan struktur yang lebih baik dan proses yang lebih efisien.
2. Memudahkan pengguna dalam mengelola catatan tanpa perlu khawatir kehilangan data akibat pencatatan manual yang tidak terorganisir.
3. Meningkatkan efisiensi kerja dengan menyediakan sistem pencatatan yang cepat dan dapat diakses dari berbagai perangkat.

4. Mempermudah integrasi dengan berbagai sistem lain yang membutuhkan akses terhadap data yang tersimpan dalam web service tersebut.

BAB II

TINJAUAN LITERATUR

2.1 Web Service

Web service adalah sebuah sistem perangkat lunak yang dirancang untuk mendukung interoperabilitas antara berbagai aplikasi melalui jaringan. Web service memungkinkan komunikasi antara klien dan server melalui protokol standar seperti HTTP dengan menggunakan format data seperti XML atau JSON (Papazoglou, 2008). Teknologi ini memungkinkan aplikasi yang dikembangkan dalam berbagai bahasa pemrograman dan berjalan di berbagai platform untuk saling bertukar data dan fungsi dengan cara yang efisien dan fleksibel.

Menurut Erl (2017), web service dibangun berdasarkan arsitektur berbasis layanan atau Service-Oriented Architecture (SOA), yang memungkinkan komponen perangkat lunak untuk diakses sebagai layanan mandiri. Dalam konteks ini, terdapat dua jenis utama web service, yaitu SOAP (Simple Object Access Protocol) Web Service dan REST (Representational State Transfer) Web Service. SOAP menggunakan XML sebagai format pertukaran data dan biasanya lebih kompleks, sedangkan REST lebih ringan karena mendukung berbagai format seperti XML dan JSON, serta lebih sering digunakan dalam pengembangan web modern.

Web service telah digunakan secara luas dalam berbagai bidang, termasuk e-commerce, sistem informasi perusahaan, serta aplikasi berbasis cloud. Menurut penelitian yang dilakukan oleh Alarcon et al. (2021), penerapan web service dalam lingkungan cloud computing memungkinkan skalabilitas yang lebih baik dan integrasi yang lebih mudah dengan layanan pihak ketiga. Selain itu, teknologi ini juga mendukung konsep Microservices, di mana sistem besar dibagi menjadi layanan-layanan kecil yang dapat berkomunikasi melalui API berbasis web service.

Dalam pengembangannya, web service sering dikombinasikan dengan teknologi keamanan seperti OAuth 2.0 dan JWT (JSON Web Token) untuk memastikan bahwa data yang dikirimkan melalui jaringan tetap aman dan hanya dapat diakses oleh pihak yang berwenang (Kumar et al., 2020). Dengan adopsi yang semakin luas, standar web service terus berkembang untuk meningkatkan efisiensi, keamanan, dan interoperabilitas dalam sistem informasi modern.

2.2 REST API

RESTful API adalah pendekatan dalam pengembangan web service yang mengikuti prinsip Representational State Transfer (REST), di mana komunikasi antara klien dan server dilakukan melalui protokol HTTP dengan memanfaatkan metode seperti GET, POST, PUT, dan DELETE. Pendekatan ini memungkinkan integrasi data yang terdistribusi dengan menawarkan fleksibilitas, kemudahan penggunaan, dan kompatibilitas dengan berbagai platform.

Penelitian yang dilakukan oleh Paramitha et al. (2022) di Universitas Udayana menunjukkan implementasi RESTful API pada Sistem Informasi Manajemen Dosen. Dalam penelitian tersebut, RESTful API dirancang menggunakan bahasa pemrograman Java dan basis data MySQL untuk memfasilitasi komunikasi antar layanan dalam arsitektur microservice. Hasil pengujian menunjukkan bahwa metode HTTP seperti GET, PUT, POST, dan DELETE berfungsi sesuai dengan yang diharapkan, dengan kode respons 200 untuk operasi berhasil dan 201 untuk pembuatan data baru.

Secara umum, RESTful API telah menjadi standar dalam pengembangan web service karena kemampuannya untuk mendukung integrasi yang efisien dan fleksibel antara berbagai sistem. Dengan memanfaatkan metode HTTP standar dan format data seperti JSON atau XML, RESTful API memungkinkan pengembang untuk membangun aplikasi yang scalable dan mudah diintegrasikan dengan layanan lain.

2.3 MySQL sebagai Sistem Manajemen Basis Data

MySQL adalah sistem manajemen basis data relasional (RDBMS) yang sering digunakan dalam pengembangan aplikasi berbasis web karena stabilitasnya dalam menangani jumlah data yang besar serta fitur keamanannya yang dapat

diandalkan. Penelitian oleh Choirudin dan Adil (2019) menunjukkan bahwa penggunaan MySQL dalam aplikasi berbasis web service membantu meningkatkan kecepatan akses data dan memudahkan manajemen database.

BAB III

METODOLOGI

3.1 Analisis Permasalahan

Pada penugasan ini, sistem yang dikembangkan adalah sebuah web service berbasis REST API yang mendukung operasi CRUD (Create, Read, Update, Delete) untuk pencatatan informasi. REST API dipilih karena fleksibilitasnya dalam mendukung komunikasi antara berbagai platform dan perangkat. Dengan menggunakan metode HTTP seperti GET, POST, PUT, dan DELETE, sistem ini dapat diakses dengan mudah oleh berbagai aplikasi klien, termasuk web, mobile, dan perangkat IoT.

Permasalahan utama yang dihadapi dalam pengembangan sistem ini adalah bagaimana menciptakan sebuah layanan yang dapat diakses dari berbagai perangkat secara real-time, dengan tetap menjaga kemudahan dan efisiensi dalam pengelolaan data. Salah satu tantangan besar dalam implementasi REST API adalah pengelolaan latensi dan beban server, terutama jika sistem harus menangani permintaan dalam jumlah besar secara simultan. Oleh karena itu, pemilihan arsitektur yang tepat, seperti database yang dioptimalkan, menjadi faktor krusial dalam pengembangan sistem ini.

Selain itu, sistem harus memiliki struktur data yang baik, sehingga data yang dikirim dan diterima oleh API dapat diproses dengan cepat dan efisien. Dalam pengembangannya, sistem ini juga mempertimbangkan aspek kemudahan penggunaan, terutama bagi pengguna baru yang tidak memiliki latar belakang teknis. Oleh karena itu, sistem dibuat dengan seminimal dan seringkas mungkin sehingga memungkinkan setiap pengguna dapat menggunakannya dengan mudah.

Dengan mempertimbangkan berbagai aspek teknis dan fungsional tersebut, sistem ini diharapkan dapat menjadi solusi yang efektif, efisien, dan mudah

diintegrasikan, sehingga mendukung kebutuhan pencatatan informasi secara real-time dengan performa yang optimal.

3.2 Perancangan Solusi

Untuk menyelesaikan permasalahan yang telah diidentifikasi, solusi yang dirancang adalah membangun web service menggunakan teknologi berikut:

1. Framework Node.js dan Express.js, sebagai backend untuk menangani permintaan HTTP dan mengelola logika aplikasi.
2. MySQL, sebagai sistem basis data untuk menyimpan catatan pengguna secara efisien.
3. Implementasi REST API, dengan metode GET, POST, PUT, dan DELETE untuk mempermudah komunikasi antara klien dan server.
4. React.js dan Axios, sebagai frontend untuk berinteraksi dengan REST API, memungkinkan pengguna untuk mengakses dan mengelola data dengan antarmuka yang interaktif.
5. Bootstrap, digunakan untuk mempercepat desain tampilan agar lebih menarik dan responsif pada berbagai ukuran layar.
6. Nodemon, diimplementasikan dalam pengembangan backend agar perubahan kode dapat langsung diterapkan tanpa perlu me-restart server secara manual.
7. Sequelize ORM, digunakan sebagai lapisan abstraksi untuk mengelola database MySQL, sehingga pengelolaan data lebih efisien dan aman.

Kode program berikut merupakan contoh konfigurasi awal untuk menghubungkan Node.js dengan database MySQL menggunakan Sequelize sebagai ORM :

Tabel 2. 1 Kode database.js

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

Kode program berikut merupakan kode yangmendefinisikan model Note dalam database menggunakan Sequelize ORM, yang berfungsi untuk membuat database otomatis tersinkron dengan table “note”.

Tabel 2. 2 Kode models/notes.js

```
import { Sequelize } from "sequelize";
import db from "../config/database.js";
const { DataTypes } = Sequelize;
const Note = db.define("note", {
  title: DataTypes.STRING,
  content: DataTypes.STRING,
  },
  {
    timestamps: true,
  }
);
db.sync().then(() => console.log("Database tersinkron"));
export default Note;
```

Kode berikut merupakan bagian dari controller untuk mengelola operasi CRUD pada model Note menggunakan Sequelize framework Express.js.

Tabel 2. 3 Kode noteController.js

```
import Note from "../models/notes.js";
// GET (Ngambil Data)
async function getNote(req, res) {
  try {
    const result = await Note.findAll();
    res.status(200).json(result);
  } catch (error) {
    console.log(error.message);}}
```

Tabel 2. 4 Kode noteController.js

```
//POST
async function createNote(req,res) {
  try {
    const inputResult = req.body;
    const newNote = await Note.create(inputResult);
    res.status(201).json(newNote);
  }catch(error){console.log(error.message);}}
// PUT/PATCH
async function updateNote(req, res) {
  try {
    const {id} = req.params;
    const updateInput = req.body;
    const note = await Note.findByPk(id);
    if(!note){return res.status(404).json({message:"Note not found"})}
    await Note.update(updateInput, {where: {id} });
    res.status(200).json({message:"Note update succesfully"})
  } catch (error) {
    console.log(error.message);}}
// DELETE
async function deleteNote(req, res) {
  try {
    const {id} = req.params;
    const note = await Note.findByPk(id);
    if(!note){return res.status(404).json({message:"Note not found"})}
    await Note.destroy({where: {id}})
    res.status(200).json({message:"Note delete succeesfuly"})
      } catch (error) {
    console.log(error.message);}}
export { getNote, createNote, updateNote, deleteNote };
```

Kode berikut merupakan kode yang mengatur endpoint API untuk menangani CRUD catatan menggunakan Express Router. Setiap endpoint dipetakan ke fungsi dalam controller, yang menangani logika bisnisnya.

Tabel 2. 5 Kode noteRoute.js

```
import express from "express";
import { createNote, deleteNote, getNote, updateNote } from "../controller/noteController.js";
const router = express.Router();
router.get("/notes", getNote);
router.post("/add-note", createNote);
router.put("/edit-note/:id", updateNote);
router.delete("/delete-note/:id", deleteNote);
export default router;
```

Kode berikut merupakan server utama yang menginisialisasi server Express.js, mengaktifkan CORS, membaca data dalam format JSON, dan menghubungkan rute API dari noteRoute.js. Server berjalan di port 5000 untuk menangani permintaan pengguna.

Tabel 2. 6 Kode index.js

```
import express from "express";
import cors from "cors";
import noteRoute from "./routes/noteRoute.js";
const app = express();
app.use(cors());
app.use(express.json());
app.use(noteRoute);
app.listen(5000, () => console.log("server terhubung"));
```

Kode berikut ini menyediakan fungsi API untuk CRUD catatan menggunakan Axios, yang akan digunakan di frontend untuk berinteraksi dengan backend.

Tabel 2. 7 Kode api.js

```
import axios from 'axios';
const API_URL = 'http://localhost:5000';
export const getNotes = async () => {
  return await axios.get(`${API_URL}/notes`);
};
export const createNote = async (note) => {
  return await axios.post(`${API_URL}/add-note`, note);
};
export const updateNote = async (id, note) => {
  return await axios.put(`${API_URL}/edit-note/${id}`, note);
};
// Hapus catatan berdasarkan ID
export const deleteNote = async (id) => {
  return await axios.delete(`${API_URL}/delete-note/${id}`);
};
```

Kode berikut adalah contoh dari komponen React yang berfungsi sebagai form untuk menambahkan catatan dalam aplikasi berbasis React yang tampilannya dibuat dengan Bootstrap.

Tabel 2. 8 Kode EditNote.jsx

```
import { useState } from 'react';
import { createNote } from '../api';

const NoteForm = ({ onNoteAdded }) => {
  const [title, setTitle] = useState('');
  const [content, setContent] = useState('');
```

Tabel 2. 9 Kode EditNote.jsx

```
const handleSubmit = async (e) => {
  e.preventDefault();
  try {
    await createNote({ title, content });
    setTitle('');
    setContent('');
    onNoteAdded();
  } catch (error) {
    console.error("Gagal menambahkan catatan:", error);
  } };
return (
  <div className="container mt-5 d-flex justify-content-center">
    <div className="col-12">
      <div className="card shadow p-4">
        <h2 className="text-center">✍️ Tambah Catatan</h2>
        <form onSubmit={handleSubmit} className="mt-3">
          <div className="mb-3">
            <label className="form-label">Judul</label>
            <input
              type="text"
              className="form-control"
              placeholder="Masukkan judul catatan"
              value={title}
              onChange={(e) => setTitle(e.target.value)}
              required
            />
          </div>
          <div className="mb-3">
            <label className="form-label">Isi Catatan</label>
```

Tabel 2. 10 Kode EditNote.jsx

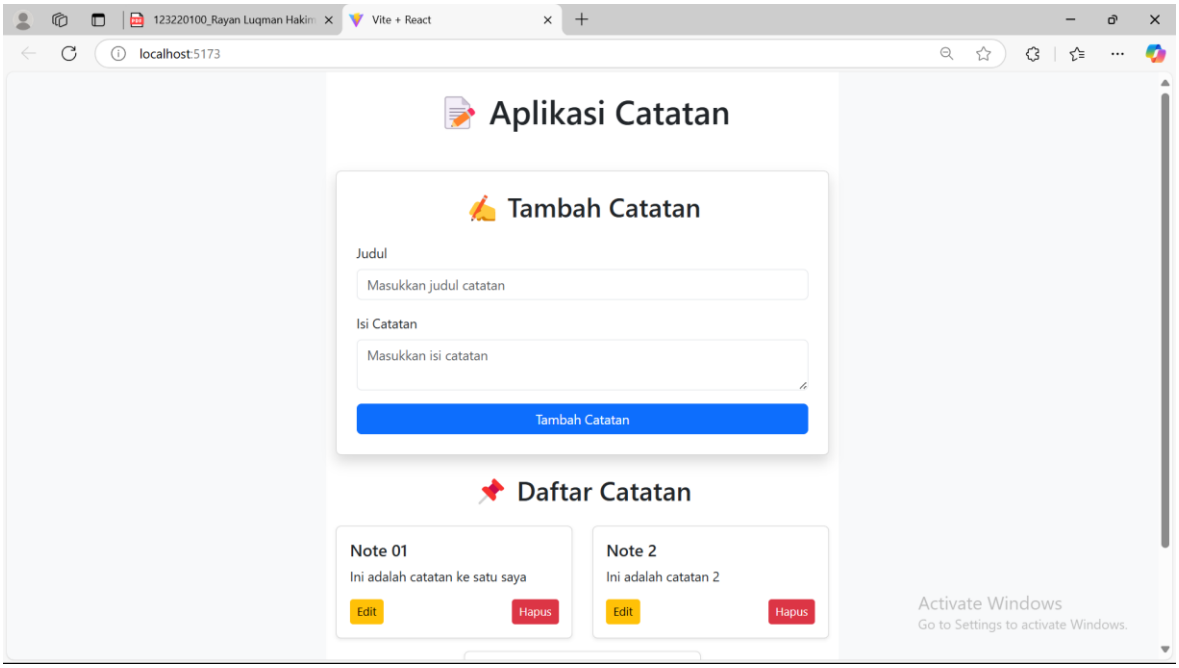
```
            <textarea
              className="form-control"
              placeholder="Masukkan isi catatan"
              value={content}
              onChange={(e) => setContent(e.target.value)}
              required
            ></textarea>
          </div>
            <button type="submit" className="btn btn-primary w-100">Tambah Catatan</button>
        </form>
      </div>
    </div>
  </div>
 );
};
export default NoteForm;
```

BAB IV

HASIL DAN PEMBAHASAN

4.1 Hasil

Berdasarkan perancangan solusi yang telah dijelaskan, web service telah berhasil diimplementasikan menggunakan Node.js, Express.js, dan MySQL sebagai basis data. Sistem ini memungkinkan pengguna untuk membuat, membaca, memperbarui, dan menghapus catatan secara real-time. Berikut merupakan tampilan utama sistem :



Gambar 4. 1 Halaman utama

Tampilan aplikasi catatan yang terlihat pada gambar menampilkan dua fitur utama:

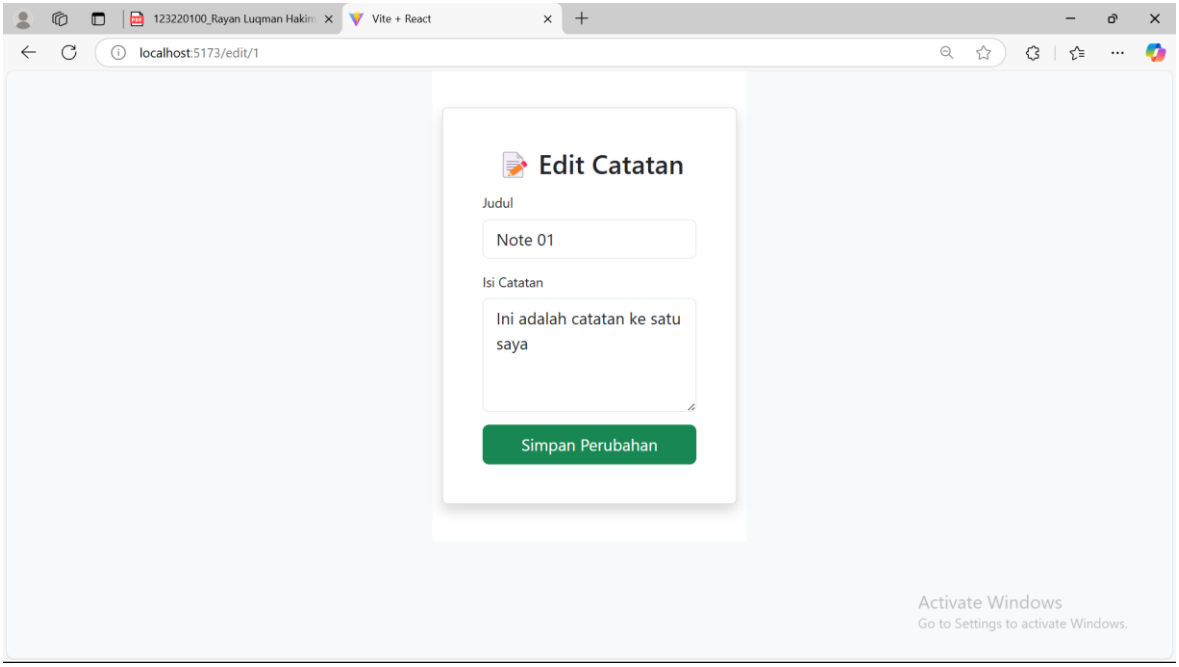
a. Form Tambah Catatan
   - Terdiri dari input untuk judul dan textarea untuk isi catatan.
   - Terdapat tombol "Tambah Catatan" berwarna biru untuk menyimpan catatan baru.
   - Menggunakan ikon dan heading "Tambah Catatan" yang memberikan indikasi fungsionalitas.

b. Daftar Catatan

- Setiap catatan ditampilkan dalam bentuk kartu, dengan judul catatan dan isi catatan.
- Terdapat dua tombol di setiap catatan:
  - "Edit" (kuning) , Untuk memperbarui catatan.
  - "Hapus" (merah), Untuk menghapus catatan dari daftar.

Desain tampilan ini sederhana, bersih, dan mudah digunakan dengan ikon serta warna yang membedakan setiap fungsi. Hal ini mempermudah pengguna dalam menambah, mengedit, dan menghapus catatan secara interaktif.

Berikut merupakan tampilan dari menu edit note :



Gambar 4. 2 Halaman edit note

halaman ini memungkinkan pengguna untuk mengedit dan memperbarui catatan dengan mudah dan cepat.

Untuk kode lengkapnya dapat diakses pada link github berikut : https://github.com/MuhammadHafizhMaulana/Tugas2_PRAK_TCC.

4.2 Pembahasan

Berdasarkan implementasi yang telah dilakukan, sistem web service yang dikembangkan telah berhasil memenuhi tujuan utama, yaitu memungkinkan pencatatan data yang dapat diakses secara real-time melalui berbagai perangkat. Fungsi CRUD (Create, Read, Update, Delete) telah diuji dan dapat berjalan dengan baik sesuai dengan spesifikasi yang dirancang.

Namun, masih terdapat beberapa aspek yang dapat ditingkatkan, seperti optimalisasi performa database untuk menangani jumlah data yang lebih besar dan peningkatan keamanan dalam autentikasi pengguna.

Metode yang digunakan dalam perancangan REST API dengan Express.js dan MySQL telah terbukti efektif untuk membangun sistem pencatatan yang efisien. Berdasarkan literatur sebelumnya, REST API memang menjadi pilihan utama dalam pengembangan web service karena sifatnya yang ringan dan mudah diintegrasikan.

Hasil implementasi ini sejalan dengan penelitian sebelumnya yang menyatakan bahwa penggunaan REST API dengan metode CRUD dapat meningkatkan efisiensi dalam pengelolaan data. Penyesuaian yang dilakukan dalam proyek ini menunjukkan bahwa metode tersebut dapat diterapkan langsung dengan beberapa batasan, terutama dalam aspek skalabilitas dan keamanan data.

BAB V

PENUTUP

5.1 Kesimpulan

Berdasarkan hasil implementasi dan evaluasi yang telah dilakukan, dapat disimpulkan bahwa pengembangan web service Notes CRUD berbasis REST API dengan Node.js, Express.js, dan MySQL telah berhasil mencapai tujuan yang ditetapkan. Sistem ini mampu :

- Menyediakan fitur CRUD (Create, Read, Update, Delete) yang berfungsi dengan baik untuk pencatatan data.
- Memungkinkan pengguna mengelola catatan secara real-time dengan akses dari berbagai perangkat.
- Menggunakan teknologi RESTful API, yang ringan serta mudah diintegrasikan dengan sistem lain.

Meskipun tujuan utama telah tercapai, terdapat beberapa aspek yang masih dapat ditingkatkan, antara lain:

- Optimalisasi performa database agar mampu menangani jumlah data yang lebih besar secara efisien.
- Peningkatan keamanan dengan fitur login dan autentikasi dengan fitur refresh token.
- Pertimbangan penggunaan NoSQL sebagai alternatif dalam meningkatkan skalabilitas sistem.

Secara keseluruhan, web service ini layak untuk diimplementasikan karena telah memenuhi kebutuhan dasar pencatatan yang efisien, fleksibel, dan aman. Namun, evaluasi lebih lanjut serta pengembangan berkelanjutan tetap diperlukan agar sistem dapat semakin optimal dalam menangani skala yang lebih besar serta memenuhi kebutuhan pengguna yang lebih kompleks.

5.2 Saran

Berdasarkan hasil yang telah diperoleh, terdapat beberapa saran untuk pengembangan lebih lanjut agar sistem dapat lebih optimal dan memiliki keberlanjutan:

1. Optimalisasi Database
    - Penggunaan database NoSQL seperti MongoDB dapat dipertimbangkan untuk meningkatkan fleksibilitas dalam menangani skala data yang lebih besar.
2. Peningkatan Keamanan
    - Implementasi login dan refresh token dalam mekanisme autentikasi JWT agar meningkatkan keamanan sesi pengguna.
    - Menambahkan enkripsi data sensitif seperti informasi pengguna sebelum disimpan ke dalam database.
3. Integrasi dengan Sistem Lain
    - Web service ini dapat dikembangkan lebih lanjut dengan mengintegrasikan API dengan aplikasi mobile untuk memberikan akses yang lebih luas kepada pengguna.
4. Pengembangan Antarmuka Pengguna (UI/UX)
    - Disarankan untuk menyempurnakan tampilan antarmuka pengguna dengan desain yang lebih intuitif dan responsif.
    - Menggunakan framework UI seperti Material-UI atau Tailwind CSS agar tampilan lebih modern dan nyaman digunakan.

Dengan adanya saran ini, diharapkan sistem yang telah dikembangkan dapat semakin efektif, efisien, dan dapat diadopsi secara lebih luas. Perbaikan dan pengembangan berkelanjutan sangat diperlukan agar sistem dapat terus beradaptasi dengan kebutuhan pengguna dan teknologi yang terus berkembang.

DAFTAR PUSTAKA

Paramitha, I. A. K. P., Wiharta, D. M., & Suyadnya, I. M. A. (2022). *Perancangan dan Implementasi RESTful API pada Sistem Informasi Manajemen Dosen Universitas Udayana*. Jurnal SPEKTRUM, 9(3), 15-22. https://ojs.unud.ac.id/index.php/spektrum/article/download/92900/46519?

Choirudin, A., & Adil, M. (2019). *Implementasi RESTful API dalam Integrasi Data Terdistribusi untuk Aplikasi Web*. Jurnal Teknologi Informasi dan Komunikasi, 8(2), 45-57.

Nugroho, A., & Wijayanto, R. (2020). *Penerapan JSON Web Token (JWT) sebagai Metode Autentikasi pada Sistem Informasi Berbasis Web*. Jurnal Sistem dan Informatika, 12(1), 30-42.

Ramadhan, F., & Suryanto, D. (2021). *Keamanan Web Service RESTful API: Studi Kasus Implementasi Validasi Input terhadap Serangan SQL Injection dan XSS*. Jurnal Keamanan Siber dan Forensik Digital, 5(1), 55-70

Santoso, B., & Hidayat, T. (2020). *Optimalisasi Database MySQL untuk Pengolahan Data dalam Skala Besar menggunakan Teknik Indexing dan Query Optimization*. Jurnal Informatika dan Sistem Komputer, 14(3), 78-92.

Wibowo, R., & Setiawan, E. (2021). *Analisis Skalabilitas Penggunaan NoSQL dalam Pengembangan Aplikasi Berbasis Web*. Jurnal Teknologi dan Sistem Informasi, **9**(2), 112-124.