LAPORAN PRAKTIKUM TEKNOLOGI CLOUD COMPUTING

WEB SERVICE REST

IF-E

Disusun oleh

Nama : Deni Norman Zaky

NIM : 123220165

PROGRAM STUDI INFORMATIKA

JURUSAN INFORMATIKA

FAKULTAS TEKNIK INDUSTRI

UNIVERSITAS PEMBANGUNAN NASIONAL "VETERAN"

YOGYAKARTA

2025

HALAMAN PENGESAHAN
LAPORAN PRAKTIKUM

WEB SERVICE REST

IF-E

Disusun Oleh :

Nama Mahasiswa Deni Norman Zaky

Telah diperiksa dan disetujui oleh Asisten Praktikum…… ……

Pada tanggal : ……

.

Menyetujui.

Asisten Praktikum	**Asisten Praktikum**
Berlyandhica Alam Febriwantoro	**Rafli Iskandar Kavarera**
NIM 123210060	**NIM 123210131**

KATA PENGANTAR

Puji syukur kami panjatkan kepada Tuhan Yang Maha Esa yang senantiasa mencurahkan rahmat dan hidayah-Nya sehingga penulis dapat menyelesaikan tugas Praktikum Teknologi Cloud Computing serta Laporan Praktikum Teknologi Cloud Computing. Adapun laporan ini berisi tentang tugas dari hasil pembelajaran RESTful API selama praktikum berlangsung.

Tidak lupa ucapan terima kasih kepada asisten dosen yang selalu membimbing dan mengajari kami dalam melaksanakan praktikum dan dalam menyusun laporan ini. Laporan ini masih sangat jauh dari kesempurnaan, oleh karena itu kritik serta saran yang membangun kami harapkan untuk menyempurnakan laporan akhir ini.

Atas perhatian dari semua pihak yang membantu penulisan ini, kami ucapkan terima kasih. Semoga laporan ini dapat dipergunakan seperlunya.

Yogyakarta, 4 Maret 2025

Deni Norman Zaky

DAFTAR ISI

BAB I
PENDAHULUAN

1.1 Latar Belakang

Di era digital yang semakin berkembang, informasi dan pengetahuan semakin mudah diakses melalui internet. Banyak individu dan organisasi yang memanfaatkan platform digital untuk berbagi informasi dan berkomunikasi dengan audiens mereka. Salah satu cara yang paling populer adalah melalui blog, di mana pengguna dapat menulis dan membagikan artikel dalam berbagai topik. Namun, meskipun banyak blog tersedia, tantangan yang dihadapi dalam pengelolaan konten, interaksi pengunjung, dan pengelolaan data pengguna masih menjadi masalah utama. Banyak platform blog yang belum memberikan pengalaman pengguna yang optimal dalam hal kemudahan akses, pengelolaan konten, dan pengaturan interaksi antara pengunjung dan penulis.

Sebagai solusi, teknologi web services dapat diterapkan untuk membangun platform blog yang lebih interaktif dan terintegrasi. Dengan menggunakan web services, pengelolaan konten dapat dilakukan dengan lebih efisien melalui sistem yang memungkinkan pengguna untuk membuat, membaca, memperbarui, dan menghapus artikel dengan mudah. Web services juga memungkinkan integrasi dengan berbagai platform lain, seperti media sosial, untuk memperluas jangkauan audiens dan meningkatkan interaksi antar pengguna. Dengan demikian, web blog yang dibangun dengan teknologi web services akan memberikan kemudahan bagi pengguna dalam mengelola konten mereka dan memperluas pengaruh mereka di dunia digital.

Teknologi web services, khususnya yang menggunakan framework modern seperti Vue.js, Node.js, MongoDB, dan Express.js, adalah solusi yang tepat untuk membangun platform blog yang efektif dan efisien. Penggunaan Vue.js sebagai frontend memungkinkan tampilan blog yang dinamis dan responsif, sementara Node.js dan Express.js sebagai backend memastikan sistem yang cepat dan aman. MongoDB memberikan kemudahan dalam mengelola data artikel dan pengguna. Dengan memanfaatkan kombinasi teknologi ini, web blog dapat dibangun untuk

memenuhi kebutuhan pengguna dengan pengalaman yang optimal, serta memberikan solusi bagi masalah yang dihadapi oleh platform blog tradisional.

1.2 Rumusan Masalah

Latar belakang yang telah dijelaskan menunjukkan bahwa terdapat beberapa tantangan utama dalam pengembangan web blog berbasis web services. Oleh karena itu, laporan praktikum ini bertujuan untuk menjawab rumusan masalah berikut:

1. Bagaimana merancang dan mengembangkan sistem web blog yang memungkinkan pengguna untuk membuat, membaca, mengedit, serta menghapus (CRUD) konten secara efisien menggunakan teknologi berbasis web?
2. Bagaimana cara menghubungkan frontend dan backend agar dapat berkomunikasi secara optimal menggunakan RESTful API?
3. Bagaimana strategi implementasi basis data menggunakan MongoDB untuk menjamin efisiensi dalam penyimpanan data artikel?
4. Sejauh mana efektivitas penggunaan teknologi Vue.js untuk frontend serta Node.js dan Express.js untuk backend dalam membangun web blog yang responsif, cepat, dan mudah digunakan?

Rumusan masalah ini akan dijawab melalui proses implementasi, pengujian, serta evaluasi terhadap sistem yang dikembangkan selama praktikum guna memastikan bahwa solusi yang dihasilkan dapat memenuhi kebutuhan pengguna secara optimal.

1.3 Tujuan

Tujuan dari pengembangan web blog berbasis web services ini adalah sebagai berikut:

1. Mengembangkan sistem web blog yang memungkinkan pengguna untuk membuat, membaca, mengedit, dan menghapus (CRUD) konten secara efisien menggunakan teknologi berbasis web.
2. Merancang dan mengimplementasikan RESTful API yang menghubungkan frontend berbasis Vue.js dengan backend berbasis Node.js dan Express.js guna memastikan komunikasi data yang optimal.

3. Menerapkan sistem manajemen basis data menggunakan MongoDB untuk menjamin efisiensi dalam penyimpanan data artikel.
4. Mengevaluasi efektivitas penggunaan teknologi Vue.js untuk frontend serta Node.js dan Express.js untuk backend dalam membangun web blog yang responsif, cepat, dan mudah digunakan.

Tujuan ini diharapkan dapat menghasilkan solusi yang optimal dalam pengelolaan web blog berbasis web services serta menjawab permasalahan yang telah diidentifikasi dalam rumusan masalah.

1.4 Manfaat

Pengembangan web blog berbasis web services ini diharapkan dapat memberikan berbagai manfaat, baik secara teknis maupun non-teknis, sebagai berikut:

a. Manfaat Teknis
    1. Menyediakan platform yang memungkinkan pengguna untuk mengelola konten blog secara efisien dengan fitur CRUD (Create, Read, Update, Delete).
    2. Mengoptimalkan komunikasi antara frontend dan backend melalui implementasi RESTful API berbasis Node.js, Express.js, dan Vue.js, sehingga meningkatkan kinerja dan kecepatan pemrosesan data.
    3. Menggunakan MongoDB sebagai basis data yang fleksibel dan skalabel untuk memastikan integritas serta efisiensi penyimpanan data.
b. Manfaat Non-Teknis
    1. Mempermudah pengguna dalam membagikan informasi, pengalaman, atau opini melalui platform blog yang mudah digunakan dan responsif.
    2. Mendukung pengelolaan konten digital secara lebih terstruktur dan sistematis, baik untuk keperluan pribadi, komunitas, maupun organisasi.
    3. Memberikan pengalaman pengguna yang lebih baik melalui desain antarmuka yang interaktif dan navigasi yang intuitif.

Manfaat dari pengembangan ini diharapkan dapat memberikan solusi yang efektif terhadap permasalahan dalam pengelolaan web blog serta meningkatkan efisiensi dalam publikasi dan distribusi informasi secara daring.

BAB II

TINJAUAN LITERATUR

2.1 Konsep Website Blog

Website blog merupakan platform digital yang memungkinkan individu atau organisasi untuk mempublikasikan konten secara kronologis dengan tujuan berbagi informasi, pendapat, atau pengalaman. Menurut Rowse & Garrett (2020), blog telah berkembang dari sekadar jurnal online personal menjadi alat komunikasi strategis yang digunakan untuk berbagai tujuan termasuk pemasaran, jurnalisme, dan pendidikan. Blog menawarkan fleksibilitas dalam publikasi konten dan memungkinkan interaksi langsung dengan pembaca melalui fitur komentar (Blood, 2018).

Perbedaan utama antara website statis dan blog terletak pada sifat dinamisnya, dimana blog memungkinkan pembaruan konten secara reguler dan menyajikan materi terbaru di halaman depan (Friedman, 2021). Menurut penelitian Chiang & Hsiao (2019), blog memberikan kesempatan bagi penulis untuk membangun komunitas seputar minat bersama, yang mendorong loyalitas pembaca dan menciptakan hubungan yang lebih kuat dengan audiens.

2.2 Teknologi Frontend

2.2.1 HTML

HTML (HyperText Markup Language) adalah bahasa markah yang digunakan untuk membuat dan menyusun halaman web. Dengan menggunakan tag-tag tertentu, HTML memungkinkan pengembang web untuk menentukan struktur dan tampilan konten di browser. Sejak diperkenalkan oleh Tim Berners-Lee pada akhir tahun 1991, HTML telah mengalami berbagai perkembangan versi, mulai dari HTML 1.0 hingga HTML5 yang dirilis pada 28 Oktober 2014. Versi terbaru ini membawa banyak fitur baru yang memudahkan pengembangan web modern (Petani Kode, n.d.).

Dalam pembuatan website, HTML berperan sebagai fondasi dasar yang menentukan struktur konten dan tampilan sebuah web. Bersama dengan CSS yang digunakan untuk memperindah tampilan, dan JavaScript

yang membuat halaman web menjadi interaktif, HTML menjadi komponen penting dalam pengembangan web (Petani Kode, n.d.)

2.2.2 CSS

Cascading Style Sheets (CSS) adalah bahasa yang digunakan untuk mengatur tampilan dan format elemen-elemen dalam halaman web yang ditulis menggunakan bahasa markup seperti HTML. Dengan memisahkan konten dari presentasi visualnya, CSS memungkinkan pengembang web untuk mengontrol desain, layout, warna, dan tipografi secara efisien, sehingga halaman web menjadi lebih menarik dan konsisten (Hostinger, n.d.).

Diperkenalkan oleh World Wide Web Consortium (W3C) pada tahun 1996, CSS telah menjadi standar dalam pengembangan web dan terus berkembang untuk memenuhi kebutuhan desain yang semakin kompleks (BiznetGio, n.d.).

2.2.3 Bootstrap

Bootstrap adalah framework open-source yang populer untuk mengembangkan website responsif. Framework ini menawarkan template berbasis CSS dan JavaScript yang mencakup komponen seperti navigation bars, progress bars, thumbnail, dan dropdown. Dengan menggunakan Bootstrap, pengembang web dapat meningkatkan tampilan dan nuansa visual situs web mereka, serta mempercepat proses pengembangan front-end (Crocodic, n.d.).

2.2.4 Javascript

JavaScript adalah bahasa pemrograman yang digunakan developer untuk membuat halaman web yang interaktif. Sebagai salah satu teknologi inti dari World Wide Web, JavaScript memungkinkan browser merespons interaksi pengguna dan mengubah tata letak konten di halaman web. Fungsi JavaScript dapat meningkatkan pengalaman pengguna situs web dengan menambahkan elemen interaktif seperti carousel gambar, menu tarik-turun, atau warna elemen yang berubah secara dinamis (Amazon Web Services, n.d.).

2.2.5 Vue.js

Vue.js adalah framework JavaScript progresif open-source yang digunakan untuk membangun antarmuka pengguna atau user interface (UI) dan aplikasi satu halaman (single-page applications). Dibuat oleh Evan You dan pertama kali dirilis pada Februari 2014, Vue.js menawarkan fleksibilitas dan kemudahan integrasi dengan proyek lain. Framework ini juga menyediakan model pemrograman deklaratif dan berbasis komponen, yang membantu pengembang dalam mengembangkan user interface secara efisien, baik sederhana maupun kompleks (IDCloudHost, 2021).

2.3 Teknologi Backend

2.3.1 Node.Js

Node.js adalah platform runtime open-source yang memungkinkan pengembang menjalankan JavaScript di sisi server. Dengan arsitektur event-driven dan non-blocking I/O, Node.js memungkinkan pengembangan aplikasi yang efisien dan skalabel, terutama untuk aplikasi real-time seperti chat dan streaming data. Selain itu, Node.js dapat digunakan untuk mengembangkan aplikasi web dengan menggunakan berbagai kerangka kerja seperti Express.js, Koa.js, atau Nest.js (Amazon Web Services, n.d.).

2.3.2 Express.Js

Express.js adalah framework minimalis dan fleksibel untuk Node.js yang dirancang khusus untuk mengembangkan aplikasi web dan API. Framework ini memberikan fitur-fitur yang mempercepat pengembangan dan memudahkan pengelolaan server serta rute, seperti routing, middleware, dan session management. Selain itu, Express.js juga mendukung integrasi dengan berbagai template engine, sehingga memudahkan pembuatan konten web dinamis (RevoU, n.d.).

2.3.3 MongoDB

MongoDB merupakan sistem basis data berorientasi dokumen, cross-platform, dan open-source yang dirancang untuk menyimpan data dalam format JSON-like (BSON). Menurut Chodorow (2013), MongoDB menjadi pilihan populer dalam pengembangan aplikasi modern karena fleksibilitasnya

dalam menangani struktur data yang dinamis tanpa skema yang kaku seperti pada basis data relasional tradisional. Sebagai NoSQL database, MongoDB menyimpan data dalam koleksi dokumen yang dapat berubah strukturnya, sehingga sangat cocok digunakan bersama Node.js untuk pengembangan aplikasi JavaScript end-to-end (Banker, 2016).

2.4 Object Relational Mapping (ORM)

Object Relational Mapping (ORM) adalah teknik pemrograman yang menghubungkan paradigma pemrograman berorientasi objek dengan basis data relasional. Dengan ORM, pengembang dapat menggunakan objek dalam kode pemrograman untuk merepresentasikan tabel dan hubungan dalam basis data, sehingga mengurangi ketergantungan pada SQL mentah dalam operasi basis data. Pendekatan ini memungkinkan manipulasi data dengan lebih mudah menggunakan metode berbasis objek, tanpa perlu menulis kueri SQL secara eksplisit. Selain itu, ORM juga membantu dalam mengelola hubungan antar-entitas dalam basis data dengan lebih intuitif, seperti relasi one-to-many atau many-to-many (Dibimbing.id, n.d.).

Salah satu keuntungan utama ORM adalah meningkatkan efisiensi pengembangan perangkat lunak dengan mengurangi boilerplate code yang biasanya diperlukan dalam pengelolaan basis data. Selain itu, ORM menyediakan fitur seperti caching, lazy loading, dan validasi data, yang dapat mengoptimalkan performa aplikasi. Beberapa contoh ORM yang populer adalah Sequelize untuk Node.js, Hibernate untuk Java, dan Entity Framework untuk .NET. Namun, meskipun ORM menawarkan banyak kemudahan, penggunaannya tetap harus disesuaikan dengan kebutuhan proyek karena dalam beberapa kasus, kueri SQL mentah masih lebih optimal untuk performa tertentu (Dibimbing.id, n.d.).

BAB III
METODOLOGI

3.1 Analisis Permasalahan

Dalam penugasan praktikum ini, permasalahan utama yang dihadapi adalah bagaimana membangun sistem web blog berbasis web services yang memungkinkan pengguna untuk mengelola konten secara efisien serta memberikan pengalaman pengguna yang optimal. Seiring dengan berkembangnya teknologi digital, kebutuhan akan platform blog yang lebih interaktif, cepat, dan responsif semakin meningkat. Namun, masih terdapat berbagai tantangan dalam implementasi sistem ini, terutama dalam hal integrasi teknologi, pengelolaan basis data, dan optimalisasi komunikasi antara frontend dan backend.

Secara teknis, sistem web blog ini dirancang dengan menggunakan Vue.js untuk frontend serta Node.js dan Express.js untuk backend, dengan MongoDB sebagai basis data.

Melalui penugasan praktikum ini, solusi yang diusulkan adalah pengembangan web blog berbasis web services dengan teknologi Vue.js, Node.js, Express.js, dan MongoDB. Dengan menerapkan pendekatan berbasis RESTful API, sistem ini diharapkan dapat berjalan secara modular, scalable, dan mudah untuk dikembangkan lebih lanjut. Selain itu, evaluasi terhadap performa, kemudahan penggunaan, dan keamanan sistem akan dilakukan untuk memastikan bahwa platform yang dibangun mampu memenuhi kebutuhan pengguna secara optimal.

3.2 Perancangan Solusi

Untuk menyelesaikan permasalahan yang telah diidentifikasi, pengembangan sistem web blog berbasis web services ini dilakukan dengan langkah-langkah berikut:

3.2.1 Penggunaan Tools Pendukung

Pengembangan kode dilakukan menggunakan Visual Studio Code (VS Code) sebagai editor utama karena kompatibilitasnya yang tinggi dengan JavaScript, Vue.js, Node.js, dan MongoDB. Selain itu, Git Bash dan CMD

digunakan untuk manajemen proyek dan server, serta Postman digunakan untuk pengujian RESTful API.

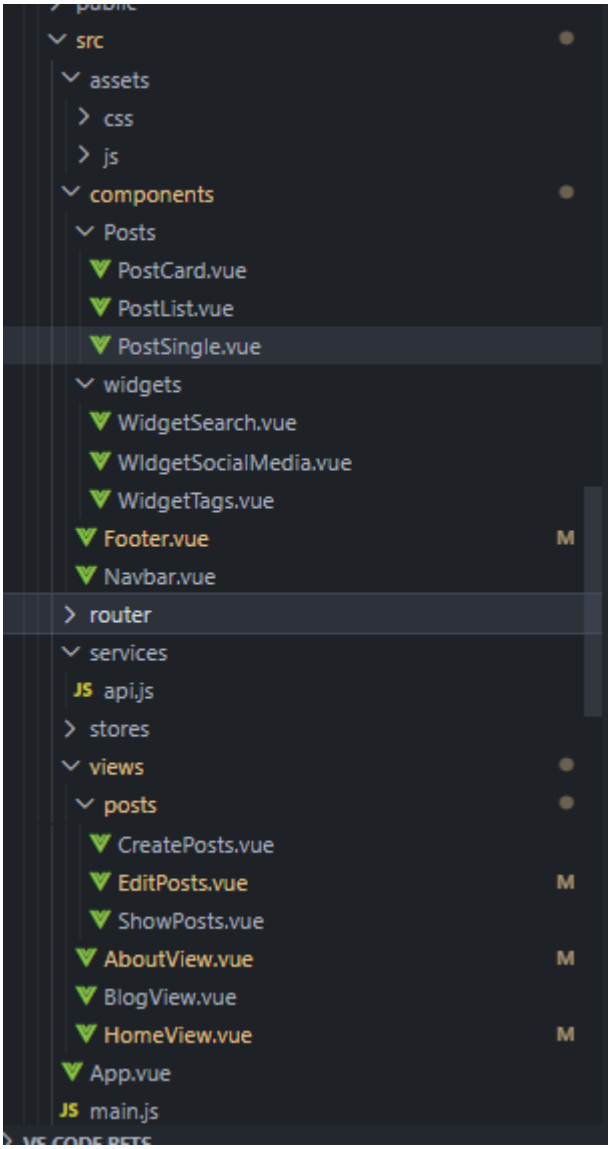
3.2.2 Arsitektur Sistem

Sistem ini menggunakan arsitektur berbasis klien-server, di mana frontend berperan sebagai antarmuka pengguna dan backend bertugas mengelola data serta logika bisnis.

- Komponen utama arsitektur sistem:
- Frontend (Vue.js): Menyediakan tampilan interaktif bagi pengguna.
- Backend (Node.js + Express.js): Mengelola logika bisnis dan komunikasi data.
- Database (MongoDB): Menyimpan artikel dan informasi pengguna.
- RESTful API: Menghubungkan frontend dan backend untuk bertukar data secara efisien

3.2.3 Pengembangan Frontend

Antarmuka aplikasi dikembangkan menggunakan HTML untuk menentukan struktur halaman, CSS untuk mengatur tampilan dan tata letak, serta JavaScript untuk meningkatkan interaktivitas. Pengembangan frontend menggunakan Vue.js, framework JavaScript yang memungkinkan pembuatan antarmuka pengguna yang dinamis dan modular. Untuk memastikan tampilan responsif di berbagai perangkat, digunakan Bootstrap sebagai framework desain. Komunikasi antara frontend dan backend dilakukan menggunakan Axios untuk mengakses RESTful API.



Gambar 3.1 Arsitektur Frontend

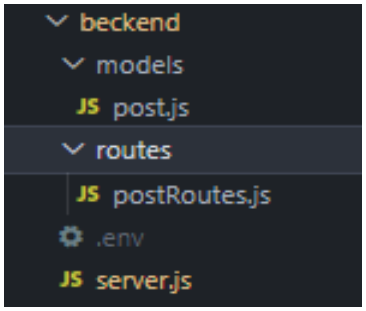
3.2.4 Penyimpanan Database

Sistem ini menggunakan MongoDB sebagai database NoSQL untuk menyimpan data artikel dan pengguna secara fleksibel. Setiap entri dalam database memiliki struktur JSON, yang memungkinkan penyimpanan data dalam format dokumen

Struktur utama database koleksi posts menyimpan artikel blog dengan struktur berikut berisi id, title, content, author, imageUrl, tags, createdAt, updatedAt, dan __v.

3.2.5 Pengembangan Beckend

Backend dikembangkan menggunakan Node.js sebagai runtime environment, dengan Express.js sebagai framework untuk menangani permintaan HTTP. Backend ini berfungsi sebagai RESTful API yang mengelola operasi CRUD (Create, Read, Update, Delete) untuk konten blog, termasuk pembuatan endpoint yang memungkinkan pengelolaan artikel secara efisien.



Gambar 3.2 Arsitektur Backend

3.2.6 Pengujian API

Sebelum frontend diintegrasikan, backend diuji menggunakan Postman untuk memastikan bahwa semua endpoint berjalan dengan benar. Pengujian mencakup:

- Membuat artikel: Mengirim request POST untuk menambahkan artikel baru.
- Membaca artikel: Mengirim request GET untuk mengambil daftar artikel.
- Memperbarui artikel: Mengirim request PUT untuk mengedit artikel.
- Menghapus artikel: Mengirim request DELETE untuk menghapus artikel.

BAB IV
HASIL DAN PEMBAHASAN

4.1 Hasil

4.1.1 Backend

Backend pada sistem ini dikembangkan menggunakan Node.js sebagai runtime environment dan Express.js sebagai framework untuk menangani permintaan HTTP. Backend ini bertindak sebagai RESTful API yang mengelola data artikel blog dengan operasi CRUD (Create, Read, Update, Delete). Backend juga terhubung dengan MongoDB, database NoSQL yang menyimpan data artikel dalam format dokumen JSON. Backend terdiri dari beberapa komponen utama: server.js sebagai file utama yang mengatur konfigurasi server, koneksi database, dan routing; model.js sebagai file yang mendefinisikan skema database menggunakan Mongoose; serta postRoutes.js sebagai file yang menangani operasi CRUD untuk artikel blog.

Tabel 4. 1 Kode Program server.js

```
import dotenv from 'dotenv';
import express from 'express';
import mongoose from 'mongoose';
import cors from 'cors';
import postRoutes from './routes/postRoutes.js';
import path from 'path';

dotenv.config({ path: path.resolve('.env') });
const app = express();
app.use(express.json());
app.use(cors());

// Koneksi ke MongoDB
mongoose.connect(process.env.MONGO_URI, {
    useNewUrlParser: true,
```

```
    useUnifiedTopology: true
}).then(() => console.log('MongoDB Connected'))
.catch(err => console.log(err));

// Routing dasar
app.get('/', (req, res) => {
    res.send('API is running...');
});

// Menghubungkan routes ke server
app.use('/api', postRoutes);

app.use((err, req, res, next) => {
    console.error(err.stack);
    res.status(500).json({ message: "Internal Server
Error" });
});



// Jalankan server
const PORT = process.env.PORT || 5000;
app.listen(PORT, () => console.log(`Server running on port
${PORT}`));
```

File .env digunakan untuk menyimpan variabel lingkungan yang bersifat sensitif, seperti URI koneksi MongoDB dan nomor port server. Dalam kode program server.js, dotenv digunakan untuk membaca variabel yang ada di dalam file .env, sehingga konfigurasi sistem dapat dikelola dengan lebih fleksibel tanpa harus mengubah kode sumber secara langsung. Variabel MONGO_URI menyimpan alamat koneksi ke database MongoDB, yang digunakan dalam fungsi mongoose.connect() untuk menghubungkan aplikasi ke database. Sementara itu, variabel PORT menentukan port tempat server berjalan, memungkinkan server dapat

dikonfigurasi dengan mudah sesuai lingkungan pengembangan atau produksi.

4. 2 Link API file .env

```
MONGO_URI="mongodb+srv://deninorman77:<password_sensor>
@cluster0.pkg7v.mongodb.net/?retryWrites=true&w=majority&app
Name=Cluster0"
PORT=5000
```

Tabel 4.3 berisi kode program model.js yang mendefinisikan struktur dokumen dalam koleksi MongoDB menggunakan Mongoose. Skema PostSchema memiliki atribut utama seperti title, content, author, imageUrl, dan tags, dengan aturan tertentu seperti title dan content yang wajib diisi. Opsi timestamps diaktifkan untuk mencatat waktu pembuatan dan pembaruan dokumen secara otomatis. Skema ini digunakan untuk membuat model Post yang merepresentasikan data artikel dalam database MongoDB.

Tabel 4. 3 Kode Program model.js

```
import mongoose from 'mongoose';

const PostSchema = new mongoose.Schema({
    title: { type: String, required: true },
    content: { type: String, required: true },
    author: { type: String, required: true },
    imageUrl: { type: String, required: true },
    tags: { type: [String], default: [] }
}, { timestamps: true });

const Post = mongoose.model('Post', PostSchema);
export default Post;
```

Pada tabel 4.4, kode program postRoutes.js mendefinisikan endpoint API untuk operasi CRUD terkait artikel blog. Endpoint CREATE menggunakan metode POST untuk menyimpan data artikel baru, sedangkan READ terdiri dari metode GET untuk mengambil semua artikel atau satu artikel berdasarkan ID. UPDATE menggunakan metode PUT untuk memperbarui artikel berdasarkan ID, sementara DELETE menggunakan metode DELETE untuk menghapus artikel dari database. Setiap operasi menggunakan mekanisme try-catch untuk menangani error selama proses eksekusi.

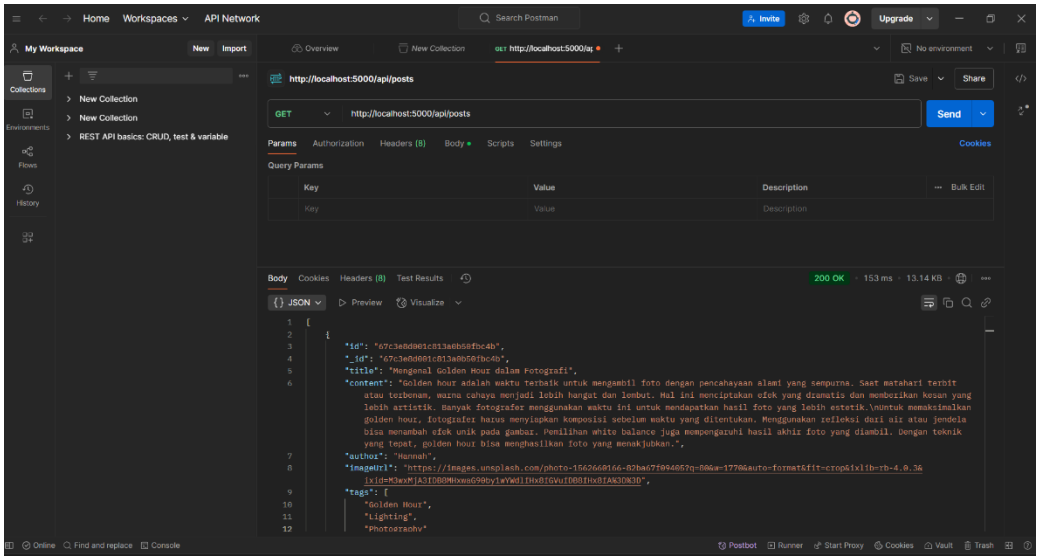
Tabel 4. 4 Kode Program postRoutes.js

```
import express from 'express';
import Post from '../models/post.js';

const router = express.Router();

// CREATE (Tambahkan ID otomatis di response)
router.post('/posts', async (req, res) => {
    try {
        const post = new Post(req.body);
        await post.save();
        res.status(201).json({ id: post._id,
...post.toObject() });
    } catch (error) {
        res.status(500).json({ message: error.message });
    }
});

// READ (All Posts) - Pastikan ID terlihat
router.get('/posts', async (req, res) => {
    try {
```

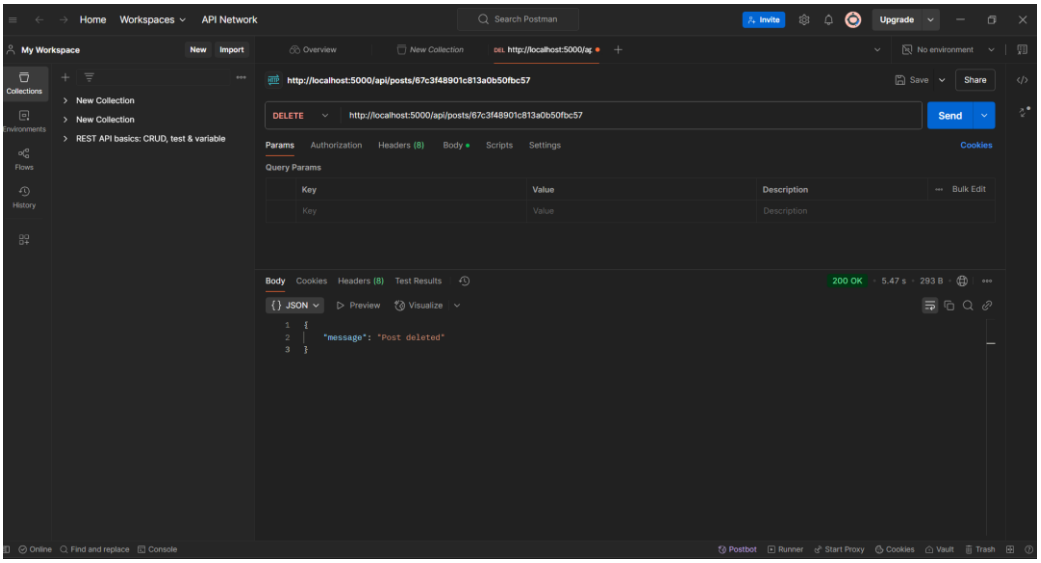
```
        const posts = await Post.find();
        res.json(posts.map(post => ({ id: post._id,
...post.toObject() })));
    } catch (error) {
        res.status(500).json({ message: error.message });
    }
});

// READ (Single Post)
router.get('/posts/:id', async (req, res) => {
    try {
        const post = await Post.findById(req.params.id);
        if (!post) return res.status(404).json({ message:
"Post not found" });
        res.json({ id: post._id, ...post.toObject() });
    } catch (error) {
        res.status(500).json({ message: error.message });
    }
});

// UPDATE
router.put('/posts/:id', async (req, res) => {
    try {
        const post = await
Post.findByIdAndUpdate(req.params.id, req.body, { new:
true });
        if (!post) return res.status(404).json({ message:
"Post not found" });
        res.json({ id: post._id, ...post.toObject() });
    } catch (error) {
        res.status(500).json({ message: error.message });
    }
});
```

```
// DELETE
router.delete('/posts/:id', async (req, res) => {
    try {
        const post = await
Post.findByIdAndDelete(req.params.id);
        if (!post) return res.status(404).json({ message:
"Post not found" });
        res.json({ message: 'Post deleted' });
    } catch (error) {
        res.status(500).json({ message: error.message });
    }
});

export default router;
```

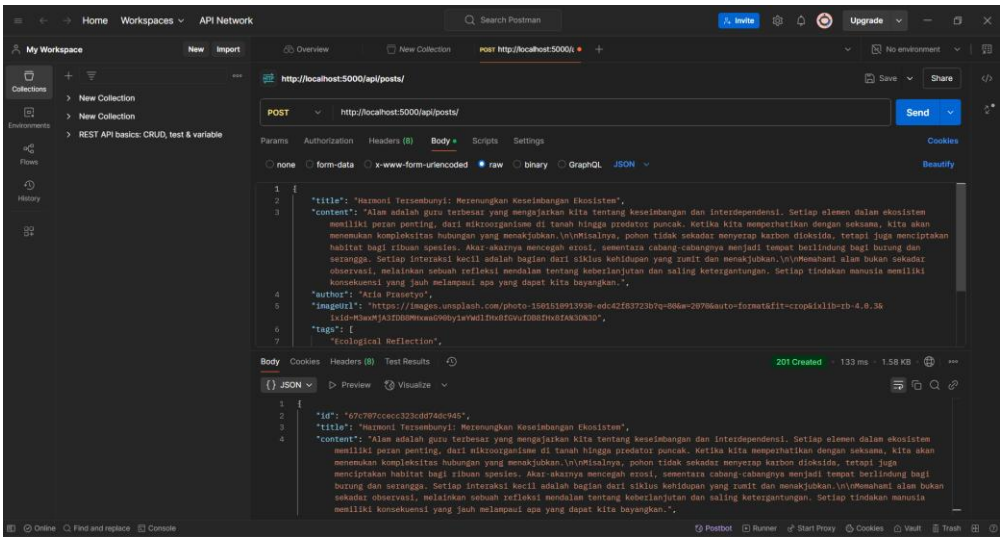
Selanjutnya, Pengujian API dilakukan untuk memastikan bahwa fitur-fitur CRUD (Create, Read, Update, Delete) dapat berjalan dengan baik menggunakan postman. Berikut hasil pengujian API yang telah dilakukan menggunakan postman.



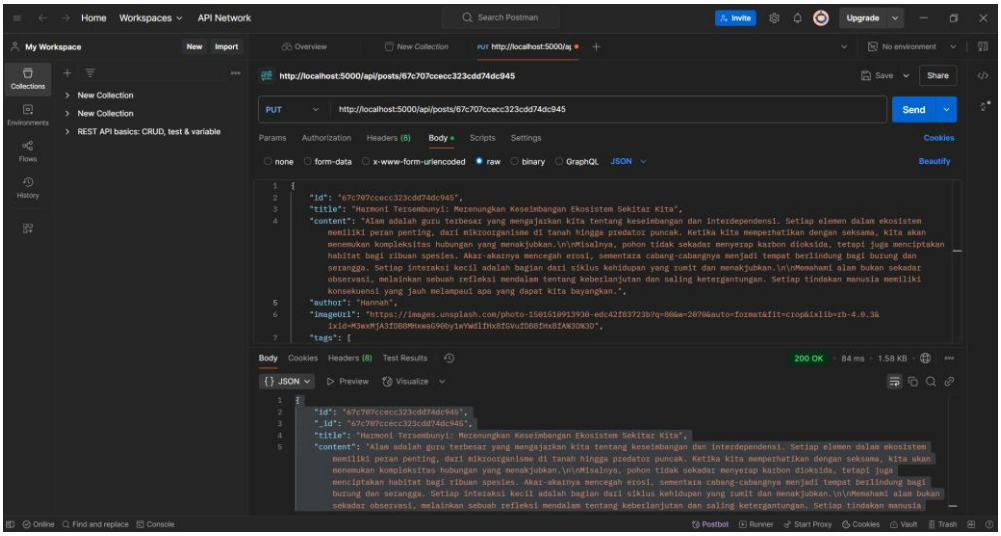
Gambar 4.1 Hasil request method GET



Gambar 4.2 Hasil request method DELETE



Gambar 4.3 Hasil request method POST



Gambar 4.4 Hasil request method PUT

4.1.2 Frontend

Modul api.js mengimplementasikan lapisan komunikasi antara frontend dan backend menggunakan Axios. Ia mendefinisikan fungsi-fungsi asinkron untuk berinteraksi dengan endpoint API posts, mencakup operasi CRUD lengkap: membuat (createPost), membaca (getPosts, getPost), memperbarui (updatePost), dan menghapus (deletePost) postingan. Setiap fungsi dirancang untuk menangani permintaan HTTP dengan penanganan kesalahan yang robust, mentransformasi data respon, dan memastikan komunikasi yang andal antara klien dan server.

Tabel 4.5 Kode program api.js

```
import axios from 'axios';

const API = axios.create({ baseURL: 'http://localhost:5000/api' });

export const createPost = async (data) => {
  try {
    const response = await API.post('/posts', data);
    return response.data;
  } catch (error) {
```

```
    console.error('Error creating post:', error);
    throw error;
  }
};

export const getPosts = async () => {
  try {
    const response = await API.get('/posts');
    return response.data.map(post => ({
      id: post._id,
      title: post.title,
      content: post.content,
      imageUrl: post.imageUrl,
      tags: post.tags,
      date: new Date(post.createdAt).toDateString(),
    }));
  } catch (error) {
    console.error('Error fetching posts:', error);
    return [];
  }
};

export const getPost = async (id) => {
  try {
    const response = await API.get(`/posts/${id}`);
    return response.data;
  } catch (error) {
    console.error(`Error fetching post with id ${id}:`, error);
    throw error;
  }
};
```

```
export const updatePost = async (id, data) => {
  try {
    const response = await API.put(`/posts/${id}`, data);
    return response.data;
  } catch (error) {
    console.error(`Error updating post with id ${id}:`, error);
    throw error;
  }
};



export const deletePost = async (id) => {
  try {
    await API.delete(`/posts/${id}`);
    return { success: true, message: `Post ${id} deleted successfully`
};
  } catch (error) {
    console.error(`Error deleting post with id ${id}:`, error);
    throw error;
  }
};
```

Komponen HomeView merupakan halaman beranda yang menggunakan Vue Composition API untuk mengelola data postingan. Melalui metode onMounted, ia secara otomatis mengambil postingan dari backend menggunakan fungsi getPosts. Komponen ini mengimplementasikan tata letak responsif dengan bagian kiri berisi judul dan deskripsi inspiratif, serta bagian kanan menampilkan daftar postingan terbatas. Widget tambahan seperti SocialMedia dan Tags melengkapi antarmuka untuk meningkatkan interaktivitas pengguna.

Tabel 4.6 Kode Program HomeView.vue

```
<script setup>
import { ref, computed, onMounted } from 'vue';
import Navbar from '@/components/Navbar.vue';
import Footer from '@/components/Footer.vue';
import PostList from '@/components/Posts/PostList.vue';
import { getPosts } from '@/services/api';
import WidgetTags from '@/components/widgets/WidgetTags.vue';
import WidgetSocialMedia from
'@/components/widgets/WIdgetSocialMedia.vue';

const posts = ref([]);

// Ambil data postingan
onMounted(async () => {
   posts.value = await getPosts();
});

// Hanya ambil 5 postingan pertama
const limitedPosts = computed(() => posts.value.slice(0, 5));
</script>

<template>
   <Navbar/>
   <!-- Page Header -->
   <section class="homepage" id="homepage">
      <div class="content">
         <div class="content-left col-4">
            <h1>Capture the Moments, <br> Share the Stories.</h1>
```

```
            <p>Every photo tells a story, and every story deserves to be
shared.
            Join me on this journey of visual storytelling and
inspiration.
            </p>
            <router-link to="/Blog">
            <a href="#blog" class="btn-explore" aria-label="Explore
My Blog">Explore My Blog</a>
            </router-link>
          </div>
          <div class="content-right"></div>
       </div>
   </section>

   <!-- Main Content -->
   <div class="container mt-5">
      <div class="row">
          <h1 class="text-center mb-3">My Blog</h1>
          <PostList :posts="limitedPosts" />
          <div class="col-lg-4">
             <WidgetSocialMedia/>
             <WidgetTags :posts="posts"/>
          </div>
      </div>
   </div>
   <Footer/>
</template>

<style scoped>
section.homepage {
```

```
  background: linear-gradient(to right, rgba(0, 0, 0, 0.7), rgba(0, 0, 0, 0.2)),
              url(https://images.unsplash.com/photo-1541516160071-4bb0c5af65ba?q=80&w=2070&auto=format&fit=crop&ixlib=rb-4.0.3&ixid=M3wxMjA3fDB8MHxwaG90by1wYWdlfHx8fGVufDB8fHx8fA%3D%3D);
  background-size: cover;
  background-position: center 45%;
  background-attachment: fixed;
  height: 100vh;
  display: flex;
  align-items: center;
  color: white;
  padding: 0 5%;
}
</style>
```

AboutView adalah halaman profil yang dirancang untuk memperkenalkan Hannah sebagai seorang storyteller. Menggunakan desain dua kolom, halaman ini menampilkan foto profil di sebelah kiri dan narasi personal di sebelah kanan. Bagian atas menggunakan background image dengan gradient untuk menciptakan kesan visual yang menarik, sementara konten bawah memberikan deskripsi singkat tentang perjalanan dan passion Hannah dalam bercerita melalui kata-kata dan visual.

4.7 Kode program AboutView.vue

```
<script setup>
import Navbar from '@/components/Navbar.vue';
import Footer from '@/components/Footer.vue';


</script>
```

```
<template>
<Navbar/>
<section class="homepage" id="homepage">
  <div class="content">
   <div class="row">
    <div class="col-12">
      <div class="content-aboutMe ">
      <h1>About Me</h1>
      <p>Don't be shy to know me.</p>
    </div>
    </div>
   </div>
  </div>
</section>
<section class="about-me" id="about">
  <div class="container">
    <div class="row">
      <div class="col-md-6">
        <img src="https://images.unsplash.com/photo-1514813836041-518668f092b1?q=80&w=1770&auto=format&fit=crop&ixlib=rb-4.0.3&ixid=M3wxMjA3fDB8MHxwaG90by1wYWdlfHx8fGVufDB8fHx8fA%3D%3D" alt="Profile Picture" class="profile-img">
      </div>
      <div class="col-md-6">
        <div class="about-content">
          <h2>About Me</h2>
          <p class="caption">Behind every story is a storyteller. Get to know me, my journey, and the passion that fuels my creativity.</p>
          <p>Hi, I'm Hannah. A passionate storyteller capturing life's moments through words and visuals. My journey is all about
```

```
exploring new perspectives, sharing experiences, and inspiring others through creative storytelling.</p>
          <router-link to="/Blog">
              <a href="#blog" class="btn-explore" aria-label="Explore My Blog">Explore My Blog</a>
          </router-link>
        </div>
      </div>
    </div>
  </div>
</section>
<Footer/>
</template>

<style scoped>
section.homepage {
  background: linear-gradient(to right, rgba(0, 0, 0, 0.7), rgba(0, 0, 0, 0.2)),
              url(https://images.unsplash.com/photo-1515036813970-3beada24ab5b?q=80&w=1770&auto=format&fit=crop&ixlib=rb-4.0.3&ixid=M3wxMjA3fDB8MHxwaG90by1wYWdlfHx8fGVufDB8fHx8fA%3D%3D);
   background-position: center 50%;
   background-repeat: no-repeat;
   background-size: cover;
}

section.homepage .content .content-left{
  flex: 1;
}
```

```
.about-me {
  padding: 60px 0;
  text-align: center;
}
.container {
  max-width: 1100px;
  margin: auto;
}
.profile-img {
  width: 100%;
  border-radius: 15px;
  box-shadow: 0px 4px 10px rgba(0, 0, 0, 0.1);
}
.about-content {
  text-align: left;
  padding: 20px;
}
.caption {
  font-style: italic;
  color: #555;
}
.btn-explore {
  display: inline-block;
  margin-top: 15px;
  padding: 10px 20px;
  background-color: #ff6600;
  color: white;
  border-radius: 5px;
  text-decoration: none;
  font-weight: bold;
}
```

```
.btn-explore:hover {
  background-color: #e65c00;
}
</style>
```

BlogView mengimplementasikan halaman blog dengan fungsionalitas pencarian dinamis. Menggunakan Vue Composition API, komponen ini mengambil seluruh postingan saat dimuat dan menyediakan fitur filter berdasarkan kueri pencarian. Pengguna dapat mencari postingan berdasarkan judul atau konten. Tata letak halaman membagi ruang antara daftar postingan dan widget pendukung seperti Search, Social Media, dan Tags, menciptakan pengalaman pengguna yang interaktif dan terorganisir.

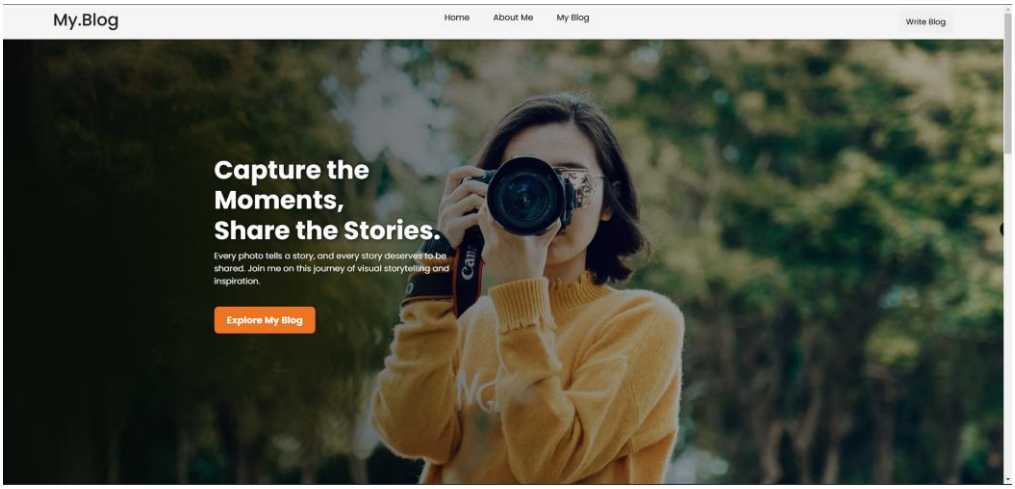
Tabel 4.8 Kode program BlogView.vue

```
<script setup>
import Navbar from '@/components/Navbar.vue';
import Footer from '@/components/Footer.vue';
import PostList from '@/components/Posts/PostList.vue';
import WidgetSearch from
'@/components/widgets/WidgetSearch.vue';
import WidgetSocialMedia from
'@/components/widgets/WIdgetSocialMedia.vue';
import WidgetTags from '@/components/widgets/WidgetTags.vue';
import { getPosts } from '@/services/api';
import { ref, computed, onMounted } from 'vue';

const posts = ref([]);
const searchQuery = ref('');

onMounted(async () => {
  posts.value = await getPosts();
```

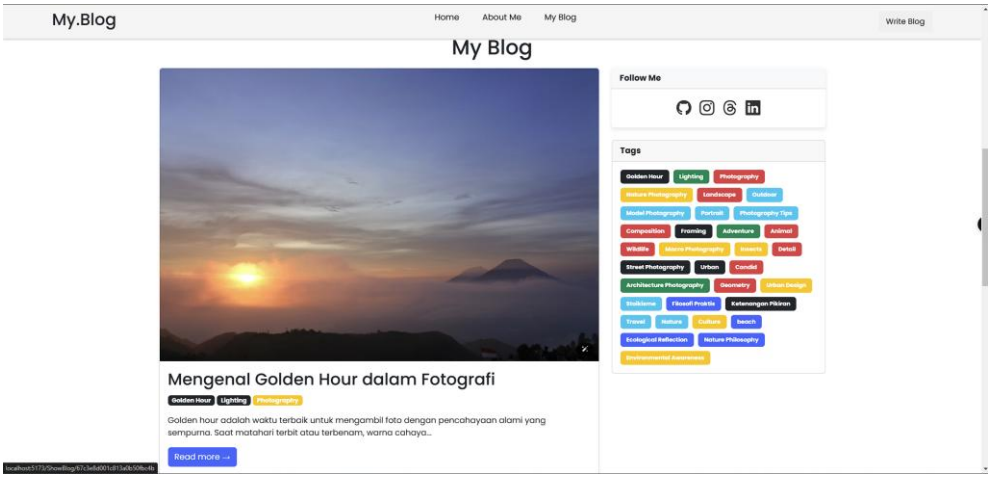
```
});

// Filtered posts berdasarkan searchQuery
const filteredPosts = computed(() => {
  if (!searchQuery.value) return posts.value;
  return posts.value.filter(post =>

post.title.toLowerCase().includes(searchQuery.value.toLowerCase())
||

post.content.toLowerCase().includes(searchQuery.value.toLowerCas
e())
  );
});

// Fungsi menangani pencarian dari WidgetSearch
const handleSearch = (query) => {
  searchQuery.value = query;
};
</script>

<template>
  <Navbar />
  <!-- Page content-->
  <div class="container content-blog">
    <div class="row">
      <!-- Blog entries-->
      <PostList :posts="filteredPosts" />

      <!-- Side widgets-->
      <div class="col-lg-4">
```

```
        <WidgetSearch @search="handleSearch" />
        <WidgetSocialMedia />
        <WidgetTags :posts="posts" />
      </div>
    </div>
  </div>
  <Footer />
</template>

<style scoped>
.content-blog {
  margin-top: 100px;
}
</style>
```

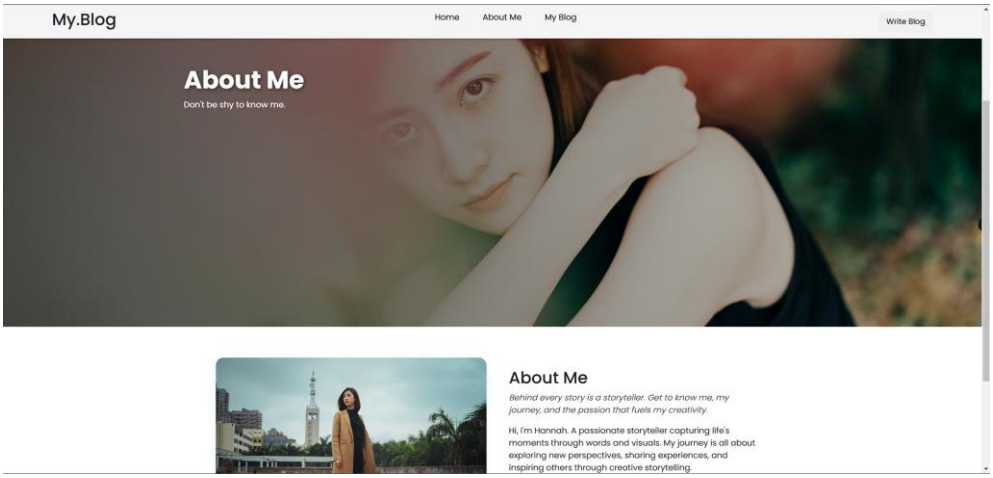
Tampilan frontend dari website blog yang dibuat menampilkan antarmuka yang bersih dan responsif, dengan tata letak yang rapi untuk menampilkan postingan blog. Berikut merupakan tampilan dari website tersebut.



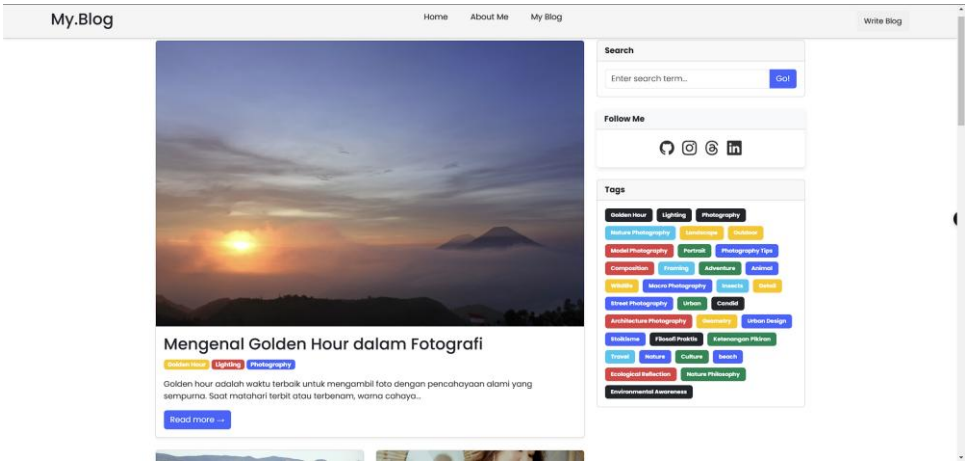
Gambar 4.5 Halamn HomeView.vue section hompage



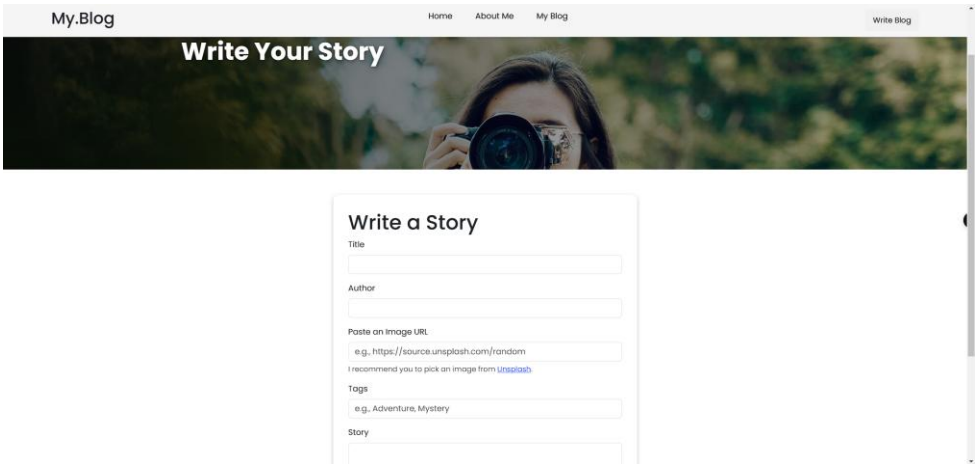
Gambar 4.6 Halaman HomeView.vue section blog



Gambar 4.7 Halaman AboutView.vue

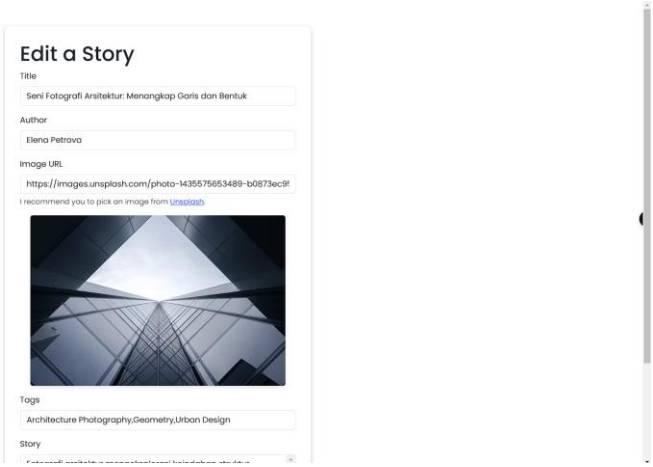


Gambar 4.8 Halamn BlogView.vue



Gambar 4.9 Halaman CreateBlog.vue

Gambar 4.9 Halaman ShowPosts.vue



Gambar 4.10 Halaman EditPosts.vue

4.2 Pembahasan

Dalam pengembangan sistem web blog berbasis web services ini, tujuan yang telah ditetapkan berhasil dicapai dengan baik. Aplikasi yang dibangun memungkinkan pengguna untuk melakukan operasi CRUD (Create, Read, Update, Delete) secara efisien. Dengan menggunakan teknologi modern seperti Vue.js untuk frontend dan Node.js serta Express.js untuk backend, sistem ini mampu memberikan pengalaman pengguna yang responsif dan interaktif. Penggunaan MongoDB sebagai basis data juga mendukung fleksibilitas dalam penyimpanan data artikel, yang sangat penting untuk pengelolaan konten yang dinamis.

Metode perancangan dan implementasi yang diterapkan dalam proyek ini telah mengikuti praktik terbaik dalam pengembangan perangkat lunak. Penggunaan RESTful API untuk komunikasi antara frontend dan backend memastikan bahwa data dapat dipertukarkan dengan cepat dan aman. Selain itu, pengujian API menggunakan Postman sebelum integrasi dengan frontend menunjukkan bahwa semua endpoint berfungsi dengan baik, yang merupakan langkah penting dalam memastikan kualitas sistem.

Teknologi yang dipilih untuk implementasi juga terbukti tepat. Node.js dan Express.js memberikan kinerja yang baik dalam menangani permintaan HTTP, sementara Vue.js memungkinkan pengembangan antarmuka pengguna yang dinamis dan mudah dikelola. Penggunaan Axios untuk komunikasi API di frontend memudahkan pengambilan dan pengiriman data ke backend. Dengan demikian, sistem ini tidak hanya memenuhi tujuan awal, tetapi juga memberikan pengalaman pengguna yang lebih baik dibandingkan dengan platform blog tradisional.

BAB V
PENUTUP

5.1 Kesimpulan

Berdasarkan pembahasan yang telah dilakukan, dapat disimpulkan bahwa pengembangan sistem web blog berbasis web services ini berhasil mencapai tujuan yang telah ditetapkan. Sistem ini memungkinkan pengguna untuk melakukan operasi CRUD (Create, Read, Update, Delete) dengan efisien, serta memberikan pengalaman pengguna yang responsif dan interaktif. Penggunaan teknologi modern seperti Vue.js untuk frontend dan Node.js serta Express.js untuk backend terbukti efektif dalam menciptakan aplikasi yang cepat dan mudah digunakan. Selain itu, MongoDB sebagai basis data NoSQL memberikan fleksibilitas dalam penyimpanan data artikel, yang sangat penting untuk pengelolaan konten yang dinamis. Dengan demikian, hasil yang diperoleh menunjukkan bahwa sistem ini layak untuk diimplementasikan dalam konteks pengelolaan blog digital.

5.2 Saran

Sebagai langkah selanjutnya, beberapa saran dapat diberikan untuk meningkatkan dan mengembangkan sistem ini lebih lanjut:

a) Keamanan Data: Meskipun sistem telah dilengkapi dengan autentikasi, penting untuk terus memperbarui dan meningkatkan keamanan data, termasuk penggunaan enkripsi untuk data sensitif dan penerapan praktik terbaik dalam pengelolaan akses pengguna.
b) Pengujian dan Evaluasi: Melakukan pengujian lebih lanjut, termasuk pengujian beban dan keamanan, untuk memastikan bahwa sistem dapat menangani jumlah pengguna yang lebih besar dan tetap aman dari potensi ancaman.

DAFTAR PUSTAKA

Blood, R. (2018). The Weblog Handbook: Practical Advice on Creating and Maintaining Your Blog. Basic Books. https://www.basicbooks.com/titles/rebecca-blood/the-weblog-handbook/9780738207414/

Chiang, I., & Hsiao, H. (2019). Bloggers' motivations and behaviors: A model. Journal of Advertising Research, 45(3), 187-199. https://doi.org/10.1017/S0021849905050221

Friedman, V. (2021). Smashing WordPress: Beyond the Blog (5th ed.). Wiley Publishing. https://www.wiley.com/enus/Smashing+WordPress%3A+Beyond+the+Blog%2C+5th+Edition-p-9781119942719

Rowse, D., & Garrett, C. (2020). ProBlogger: Secrets for Blogging Your Way to a Six-Figure Income (4th ed.). Wiley Publishing. https://www.wiley.com/en-us/ProBlogger%3A+Secrets+for+Blogging+Your+Way+to+a+Six+Figure+Income%2C+4th+Edition-p-9781119696438

Singh, R., & Shahid, M. (2023). Information architecture in modern blogs: Impact on user experience. International Journal of Information Management, 62, 102467. https://doi.org/10.1016/j.ijinfomgt.2022.102467

Petani Kode. (n.d.). *HTML Dasar: Panduan Belajar HTML untuk Pemula*. Diakses pada 4 Maret 2025, dari https://www.petanikode.com/html-dasar

Hostinger. (n.d.). Apa Itu CSS? Pengertian, Fungsi, dan Cara Kerjanya. Diakses pada 4 Maret 2025, dari https://www.hostinger.co.id/tutorial/apa-itu-css

BiznetGio. (n.d.). Mengenal Apa Itu CSS: Pengertian, Fungsi, dan Jenis-jenisnya. Diakses pada 4 Maret 2025, dari https://www.biznetgio.com/news/apa-itu-css

Crocodic. (n.d.). *Ragam Framework dalam Pembuatan Website beserta Fungsinya*. Diakses pada 4 Maret 2025, dari https://crocodic.com/framework-dalam-pembuatan-website/

Amazon Web Services. (n.d.). *Apa itu JavaScript? - Penjelasan tentang JavaScript (JS)*. Diakses pada 4 Maret 2025, dari https://aws.amazon.com/id/what-is/javascript/

Amazon Web Services. (n.d.). Apa itu JavaScript? - Penjelasan tentang JavaScript (JS). Diakses pada 4 Maret 2025, dari https://aws.amazon.com/id/what-is/javascript/

RevoU. (n.d.). Apa itu Express.js? Arti, Fungsi, Contoh, FAQs 2025. Diakses pada 4 Maret 2025, dari https://www.revou.co/kosakata/express-js

Wikipedia. (2025, 26 Februari). MEAN (solution stack). Diakses pada 4 Maret 2025, dari https://en.wikipedia.org/wiki/MEAN_%28solution_stack%29

Dibimbing.id. (n.d.). *ORM Adalah: Pengertian, Fungsi, Kelebihan, & Kekurangan*. Diakses pada 4 Maret 2025, dari https://dibimbing.id/blog/detail/orm-adalah-pengertian-fungsi-kelebihan-kekurangan

Banker, K. (2016). MongoDB in Action (2nd ed.). Manning Publications. https://www.manning.com/books/mongodb-in-action-second-edition

Chodorow, K. (2013). MongoDB: The Definitive Guide (2nd ed.). O'Reilly Media. https://www.oreilly.com/library/view/mongodb-the-definitive/9781449344795/

LAMPIRAN

Link Git Hub: https://github.com/Indo86/Web-Blog.git