LAPORAN UAS SISTEM GEOGRAFI

Disusun Oleh :
Naufal Danendra Amanullah (231240001425)

PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS ILMU KOMPUTER
UNIVERSITAS NAHDLATUL ULAMA JEPARA TAHUN 2025

Link github project qgis > https://github.com/NaufallAma/sistem-informasi-geografi.git

BAB I

PENDAHULUAN

1.1 Latar Belakang

Perkembangan teknologi informasi saat ini mendorong pemanfaatan Sistem Informasi Geografis (SIG) dalam berbagai bidang, seperti perencanaan wilayah, pemetaan fasilitas umum, serta pengelolaan data lokasi secara digital. SIG tidak hanya digunakan untuk menampilkan data spasial, tetapi juga untuk mengelola dan memperbarui informasi lokasi secara dinamis sesuai kebutuhan pengguna.

Pemanfaatan SIG berbasis web menjadi solusi yang efektif karena memungkinkan akses data peta secara luas melalui browser tanpa memerlukan perangkat lunak khusus. Integrasi antara perangkat lunak pengolahan data spasial, basis data, dan aplikasi web sangat penting untuk menghasilkan sistem yang interaktif dan mudah digunakan.

Pada project ini digunakan **QGIS** sebagai alat pengolahan dan pengelolaan data spasial, **PostgreSQL dengan PostGIS** sebagai basis data penyimpanan data geografis, serta **Node.js** sebagai backend untuk menghubungkan database dengan aplikasi web. Sistem yang dibangun mampu menampilkan peta pada web dan menyediakan fitur pengelolaan data lokasi berupa penambahan, pengubahan, dan penghapusan penanda lokasi (marker) beserta deskripsi singkatnya.

Dengan adanya sistem ini, pengguna dapat mengelola data lokasi secara lebih efisien dan interaktif, serta memperoleh informasi geografis yang selalu diperbarui melalui media web.

1.2 Tujuan

Tujuan dari pembuatan project Sistem Informasi Geografis berbasis web ini adalah sebagai berikut:

1. Menampilkan data peta geografis pada aplikasi web.
2. Mengintegrasikan QGIS dengan database PostgreSQL untuk pengelolaan data spasial.

3. Membangun sistem backend menggunakan Node.js untuk mengelola komunikasi antara web dan database.
4. Menyediakan fitur pengelolaan data lokasi berupa penambahan, pengubahan, dan penghapusan penanda lokasi pada peta.
5. Menghasilkan aplikasi SIG sederhana yang dapat diakses melalui browser.

1.3 Ruang Lingkup

Agar pembahasan dalam project ini lebih terarah, maka ruang lingkup yang dibatasi meliputi:

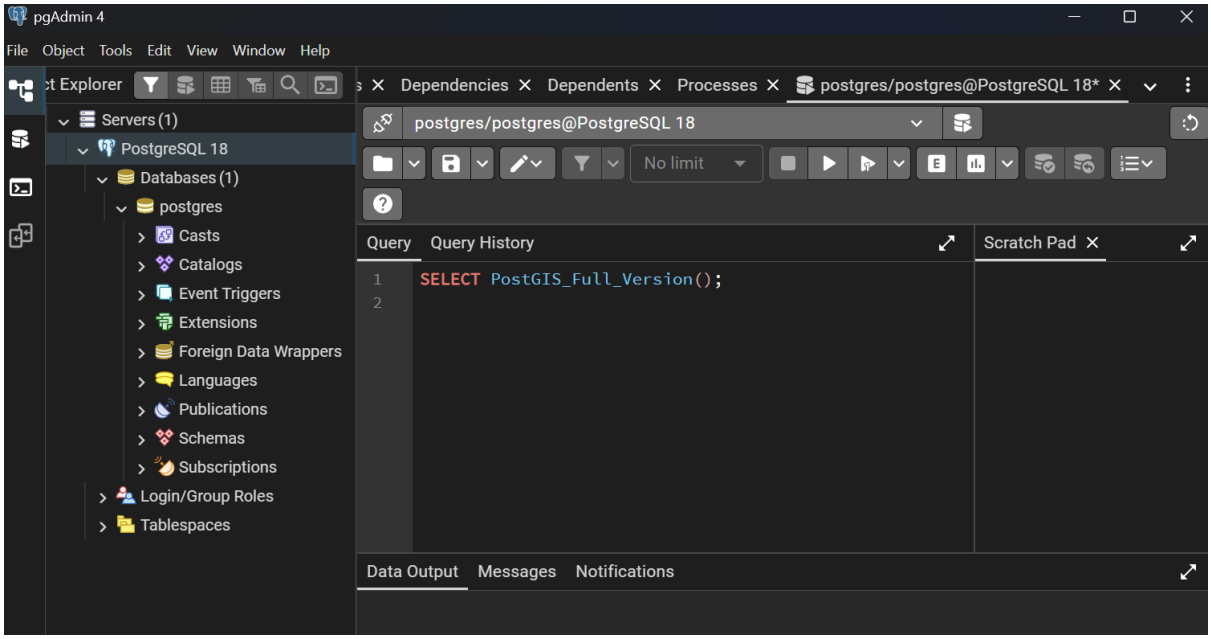
1. Pengolahan dan pengelolaan data spasial menggunakan QGIS.
2. Penyimpanan data lokasi dan atributnya menggunakan PostgreSQL dengan ekstensi PostGIS.
3. Pengembangan aplikasi web untuk menampilkan peta dan data lokasi.
4. Implementasi fitur CRUD (Create, Read, Update, Delete) untuk data penanda lokasi.
5. Sistem tidak membahas autentikasi pengguna dan pengamanan lanjutan.

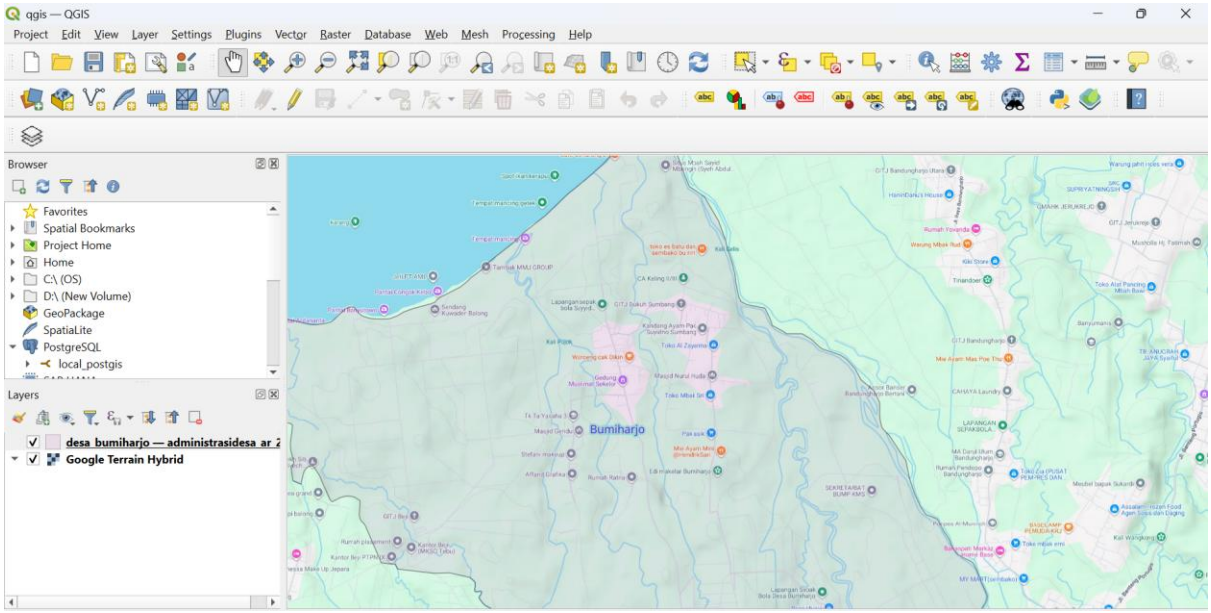
BAB II

2.1 Tools dan Teknologi yang Digunakan

Pada pembuatan Sistem Informasi Geografis berbasis web ini digunakan beberapa perangkat lunak dan teknologi pendukung, yaitu:

No	Komponen	Teknologi
1	Pengolahan Data Spasial	QGIS
2	Basis Data	PostgreSQL dengan PostGIS
3	Backend	Node.js
4	Frontend	HTML, CSS, JavaScript
5	Peta Web	Leaflet
6	Server	Localhost



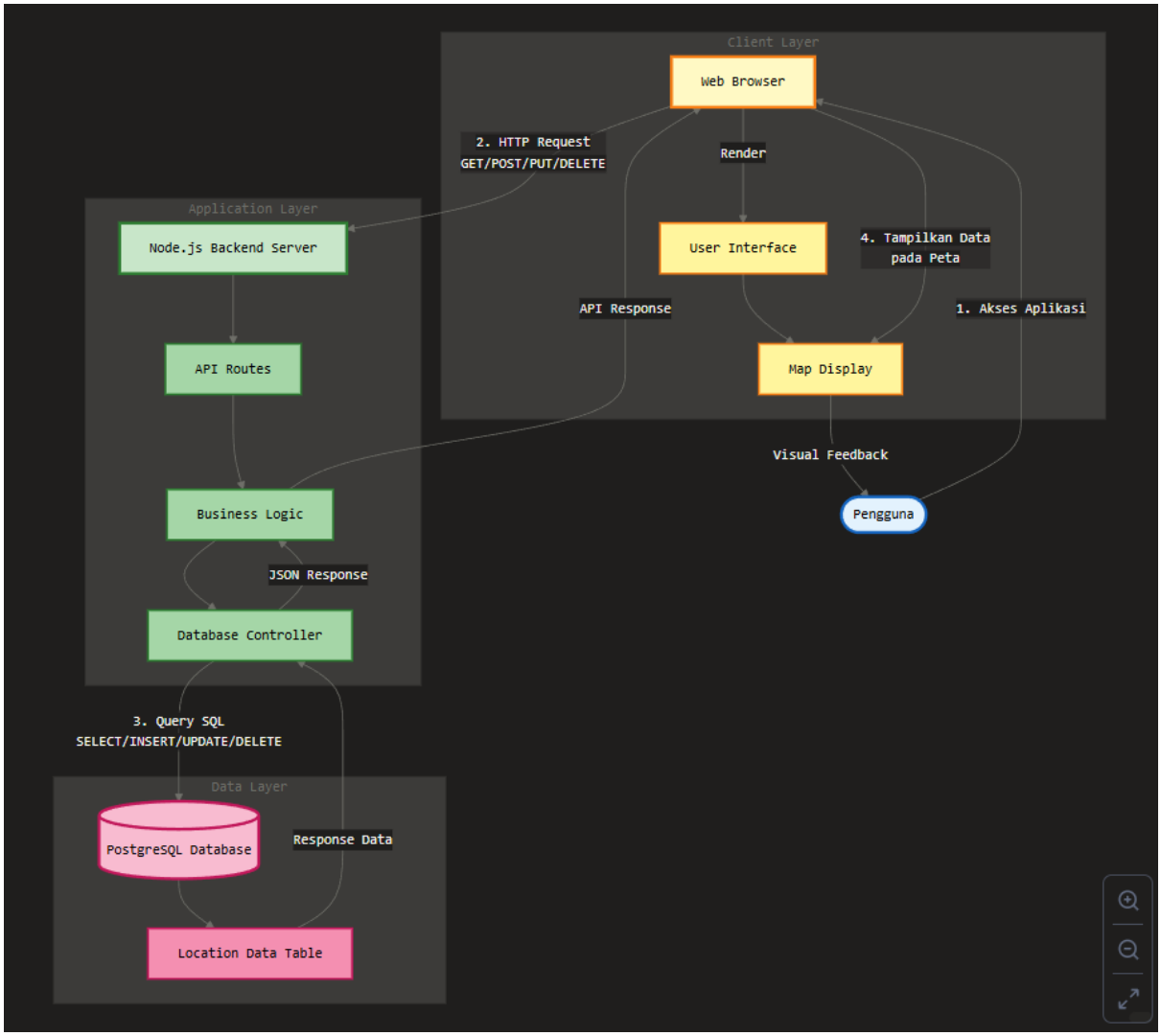


2.2 Arsitektur Sistem

Arsitektur sistem menggambarkan hubungan antara pengguna, aplikasi web, backend, dan basis data. Pengguna mengakses aplikasi melalui browser untuk melihat peta dan mengelola data lokasi. Permintaan dari pengguna akan diproses oleh backend Node.js yang kemudian berinteraksi dengan database PostgreSQL untuk menyimpan atau mengambil data lokasi.

Alur sistem secara umum adalah sebagai berikut:

1. Pengguna mengakses aplikasi melalui web browser.
2. Browser mengirim permintaan ke server Node.js.
3. Server Node.js mengakses database PostgreSQL.
4. Data lokasi ditampilkan kembali pada peta di web.



2.3 Perancangan Basis Data

Basis data digunakan untuk menyimpan informasi lokasi beserta atribut pendukungnya. Data lokasi disimpan dalam bentuk data spasial sehingga dapat ditampilkan dalam peta web.

Contoh atribut yang digunakan:

- id
- nama_lokasi
- deskripsi
- koordinat (geometry)

2.4 Perancangan Antarmuka Aplikasi

Antarmuka aplikasi dirancang agar pengguna dapat dengan mudah melihat peta serta mengelola data lokasi. Aplikasi menyediakan peta utama sebagai tampilan awal serta form untuk menambahkan, mengubah, dan menghapus penanda lokasi.

2.5 Perancangan Alur CRUD Lokasi

Sistem dirancang untuk mendukung pengelolaan data lokasi dengan metode CRUD (Create, Read, Update, Delete). Proses CRUD dilakukan melalui antarmuka web yang terhubung langsung dengan database PostgreSQL melalui backend Node.js.

Alur CRUD:

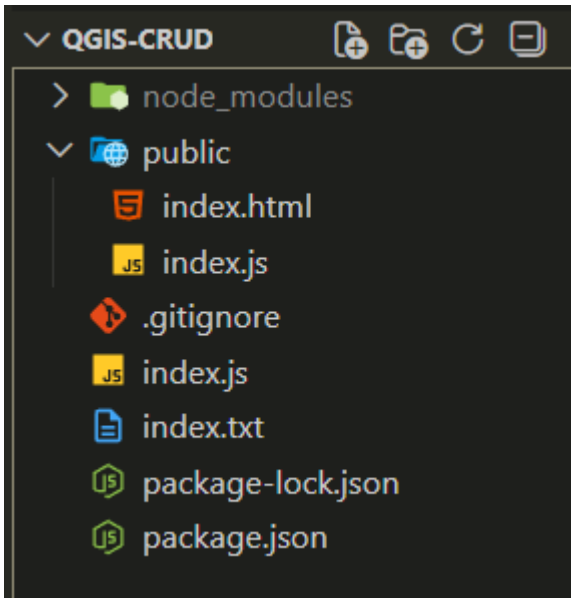
- **Create**: Menambahkan lokasi baru ke peta
- **Read**: Menampilkan seluruh lokasi pada peta
- **Update**: Mengubah data lokasi
- **Delete**: Menghapus lokasi dari database

BAB III

IMPLEMENTASI SISTEM

3.1 Struktur Folder Project

Struktur folder project mencerminkan pengelolaan kode dan asset secara teratur agar mudah di-maintain:



3.2 Backend Node.js dan Koneksi Database

Backend bertanggung jawab untuk:

- Menghubungkan web dengan database PostgreSQL
- Menangani operasi CRUD (Create, Read, Update, Delete)

index.js > ...

```
const express = require("express");
const cors = require("cors");
const { Pool } = require("pg");

const app = express();
app.use(cors());
app.use(express.json());
app.use(express.static("public"));

// ======= CONFIG POSTGRESQL =======
const pool = new Pool({
  host: "localhost",
  user: "postgres",
  password: "654321",   // ganti sesuai password kamu
  database: "postgres", // database yang berisi PostGIS
  port: 5432,
});

// ======= GET DATA DESA (GeoJSON) =======
app.get("/desa", async (req, res) => {
  try {
    const q = `
      SELECT jsonb_build_object(
        'type','FeatureCollection',
        'features', jsonb_agg(
          jsonb_build_object(
            'type','Feature',
            'geometry', ST_AsGeoJSON(geom)::jsonb,
            'properties', to_jsonb(d) - 'geom'
          )
        )
      ) AS geojson
      FROM desa_bumiharjo d;
    `;
    const { rows } = await pool.query(q);
    res.json(rows[0].geojson);
  } catch (err) {
```

```
    res.status(500).json({ error: "Database error" });
  }
});

// ======= GET SEMUA TITIK LOKASI (GeoJSON) =======
app.get("/titik", async (req, res) => {
  try {
    const q = `
      SELECT jsonb_build_object(
        'type','FeatureCollection',
        'features', COALESCE(jsonb_agg(
          jsonb_build_object(
            'type','Feature',
            'geometry', ST_AsGeoJSON(geom)::jsonb,
            'properties', jsonb_build_object(
              'id', id,
              'nama', nama,
              'keterangan', keterangan
            )
          )
        ), '[]'::jsonb)
      ) AS geojson
      FROM public.titik_lokasi;
    `;
    const { rows } = await pool.query(q);
    res.json(rows[0].geojson);
  } catch (err) {
    console.error(err);
    res.status(500).json({ error: "Query gagal" });
  }
});

// ======= CREATE TITIK LOKASI =======
app.post("/titik", async (req, res) => {
  try {
    const { nama, keterangan, latitude, longitude } = req.body;
```

```
70
71  // ======= CREATE TITIK LOKASI =======
72  app.post("/titik", async (req, res) => {
73    try {
74      const { nama, keterangan, latitude, longitude } = req.body;
75
76      if (!nama || !latitude || !longitude) {
77        return res.status(400).json({ error: "Data tidak lengkap" });
78      }
79
80      const q = `
81        INSERT INTO public.titik_lokasi (nama, keterangan, geom)
82        VALUES ($1, $2, ST_SetSRID(ST_Point($3, $4), 4326))
83        RETURNING *;
84      `;
85
86      const { rows } = await pool.query(q, [nama, keterangan, longitude, latitude]);
87      res.json(rows[0]);
88    } catch (err) {
89      console.error(err);
90      res.status(500).json({ error: "Insert gagal" });
91    }
92  });
93
94  // ======= UPDATE TITIK =======
95  app.put("/titik/:id", async (req, res) => {
96    try {
97      const { id } = req.params;
98      const { nama, keterangan, latitude, longitude } = req.body;
99
100     const q = `
101       UPDATE public.titik_lokasi
102       SET nama = $1,
103           keterangan = $2,
104           geom = ST_SetSRID(ST_Point($3, $4), 4326)
105       WHERE id = $5
106       RETURNING *;
```

```
105       WHERE id = $5
106       RETURNING *;
107     `;
108
109     const { rows } = await pool.query(q, [nama, keterangan, longitude, latitude, id]);
110     res.json(rows[0]);
111   } catch (err) {
112     console.error(err);
113     res.status(500).json({ error: "Update gagal" });
114   }
115 });
116
117 // ======= DELETE TITIK =======
118 app.delete("/titik/:id", async (req, res) => {
119   try {
120     const { id } = req.params;
121     await pool.query("DELETE FROM public.titik_lokasi WHERE id = $1", [id]);
122     res.json({ message: "Titik berhasil dihapus" });
123   } catch (err) {
124     console.error(err);
125     res.status(500).json({ error: "Delete gagal" });
126   }
127 });
128
129
130 // ======= START SERVER =======
131 app.listen(3000, () =>
132   console.log("✅ API jalan di http://localhost:3000")
133 );
134
```

3.3 Frontend dan Peta Web

Frontend menampilkan:

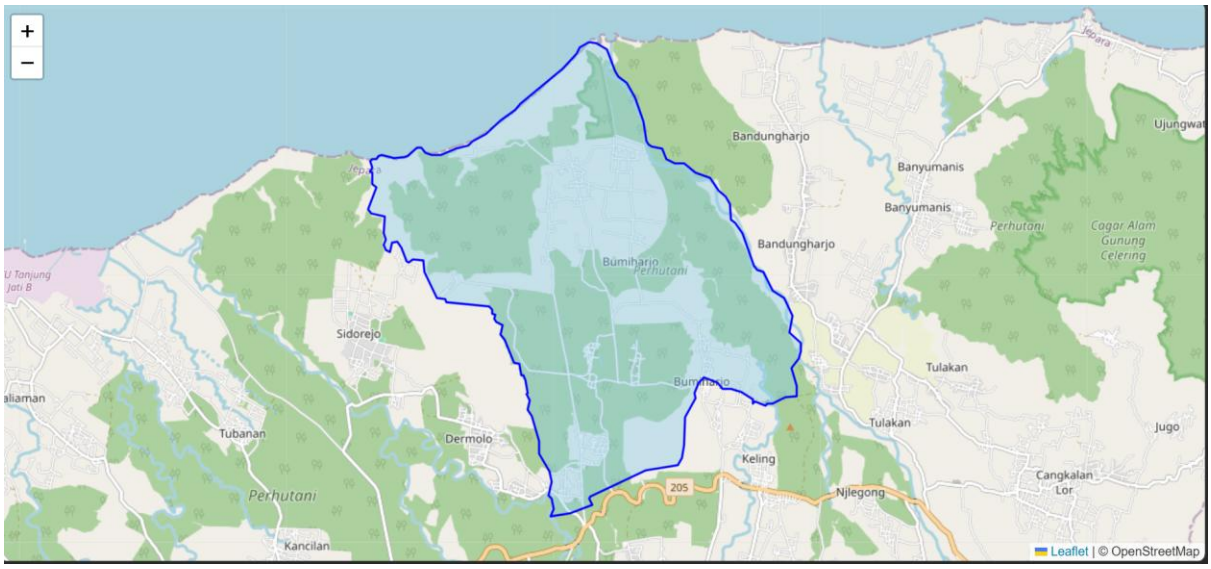
- Peta menggunakan Leaflet
- Marker lokasi berdasarkan data dari backend
- Form untuk menambah, mengubah, dan menghapus lokasi

```
public > index.js > ...
1   // 1 Inisialisasi peta
2   const map = L.map("map").setView([-6.43, 110.85], 12);
3
4   // 2 Basemap
5   L.tileLayer("https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png", {
6     attribution: "© OpenStreetMap"
7   }).addTo(map);
8
9   // 3 Layer Desa
10  fetch("http://localhost:3000/desa")
11    .then(res => res.json())
12    .then(data => {
13      const desaLayer = L.geoJSON(data, {
14    style: {
15      color: "blue",
16      weight: 2,
17      fillColor: "skyblue",
18      fillOpacity: 0.4
19    },
20    onEachFeature: (feature, layer) => {
21      layer.bindPopup("Wilayah Desa");
22
23      // Tangkap klik di polygon agar bisa menambahkan titik
24      layer.on("click", (e) => {
25        const lat = e.latlng.lat;
26        const lng = e.latlng.lng;
27
28        const nama = prompt("Nama lokasi:");
29        if (!nama) return alert("Nama wajib diisi");
30
31        const keterangan = prompt("Keterangan (opsional):") || "";
32
33        fetch("http://localhost:3000/titik", {
34          method: "POST",
35          headers: { "Content-Type": "application/json" },
36          body: JSON.stringify({
37            nama,
38            keterangan
```

```
      fetch("http://localhost:3000/titik", {
        method: "POST",
        headers: { "Content-Type": "application/json" },
        body: JSON.stringify({
          nama,
          keterangan,
          latitude: lat,
          longitude: lng
        })
      })
      .then(() => loadTitik())
      .then(() => alert("Titik berhasil disimpan"))
      .catch(console.error);
    });
  }
}).addTo(map);

    map.fitBounds(desaLayer.getBounds());
  });

// 4️⃣ Layer Titik Lokasi
let titikLayer = L.geoJSON(null).addTo(map);

// Fungsi load titik dari backend
function loadTitik() {
  fetch("http://localhost:3000/titik")
    .then(res => res.json())
    .then(data => {
      titikLayer.clearLayers();
      L.geoJSON(data, {
        onEachFeature: (feature, layer) => {
          layer.bindPopup(`<b>${feature.properties.nama}</b><br>${feature.properties.keterangan}`);
```

```
            layer.on("dblclick", () => {
              if (pilihan === "U") {
                const nama = prompt("Nama baru:", feature.properties.nama);
                const keterangan = prompt("Keterangan baru:", feature.properties.keterangan);
                if (!nama) return alert("Nama wajib diisi");

                fetch(`http://localhost:3000/titik/${feature.properties.id}`, {
                  method: "PUT",
                  headers: { "Content-Type": "application/json" },
                  body: JSON.stringify({
                    nama,
                    keterangan,
                    latitude: layer.getLatLng().lat,
                    longitude: layer.getLatLng().lng
                  })
                })
                .then(() => {
                  alert("Titik berhasil diupdate");
                  loadTitik();
                })
                .catch(console.error);

              } else if (pilihan === "D") {
                if (!confirm("Yakin ingin menghapus titik ini?")) return;
                fetch(`http://localhost:3000/titik/${feature.properties.id}`, { method: "DELETE" })
                  .then(() => {
                    alert("Titik berhasil dihapus");
                    loadTitik();
                  })
                  .catch(console.error);
              }
            });
          }
        }).addTo(titikLayer);
      });
```

```
105
106  // Load titik awal
107  loadTitik();
108
109  // 5 Tambah titik dengan klik peta
110  map.on("click", e => {
111    const nama = prompt("Nama lokasi:");
112    if (!nama) return alert("Nama wajib diisi");
113
114    const keterangan = prompt("Keterangan (opsional):") || "";
115
116    fetch("http://localhost:3000/titik", {
117      method: "POST",
118      headers: { "Content-Type": "application/json" },
119      body: JSON.stringify({
120        nama,
121        keterangan,
122        latitude: e.latlng.lat,
123        longitude: e.latlng.lng
124      })
125    })
126    .then(res => res.json())
127    .then(() => {
128      alert("Titik berhasil disimpan!");
129      loadTitik();
130    })
131    .catch(err => {
132      console.error(err);
133      alert("Gagal menyimpan titik");
134    });
135  });
136
```



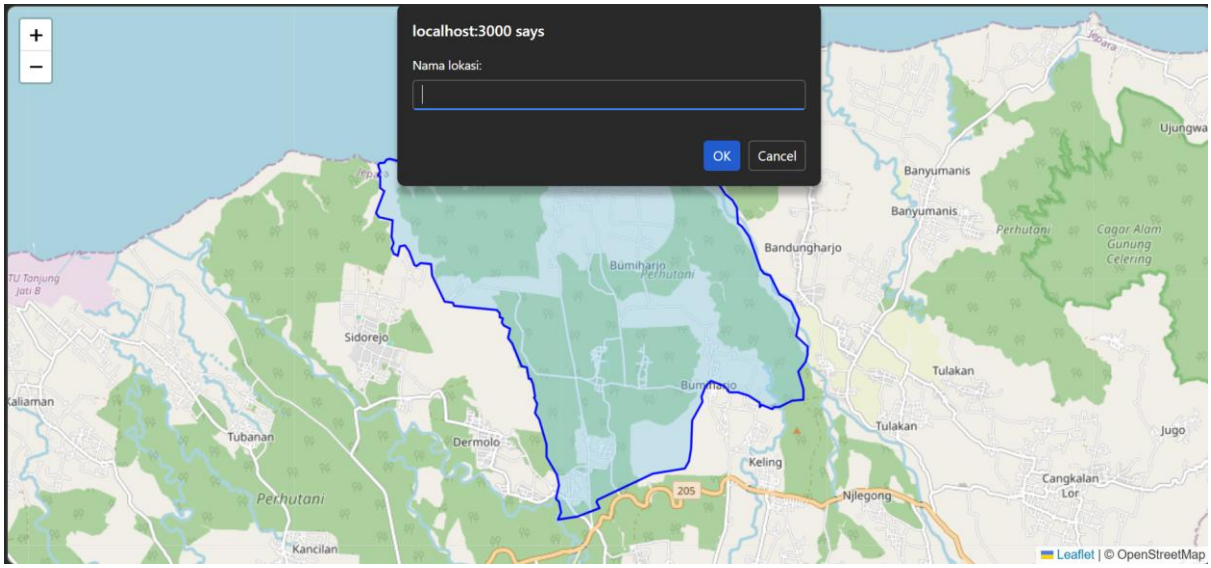
3.4 Implementasi CRUD Lokasi

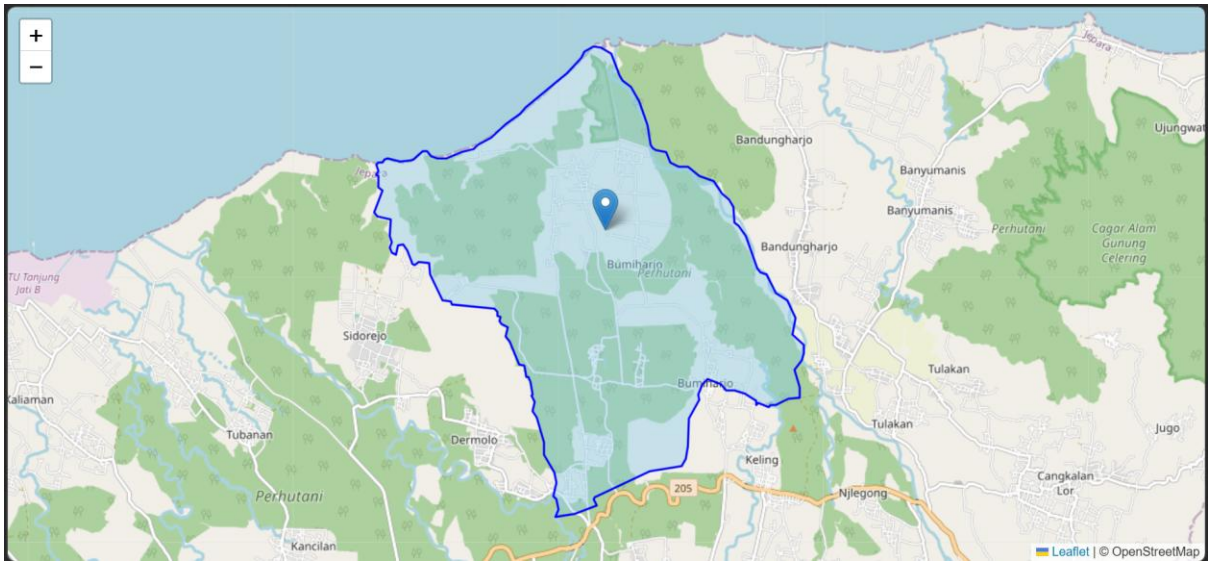
Pada sistem ini, CRUD (Create, Read, Update, Delete) diimplementasikan sebagai berikut:

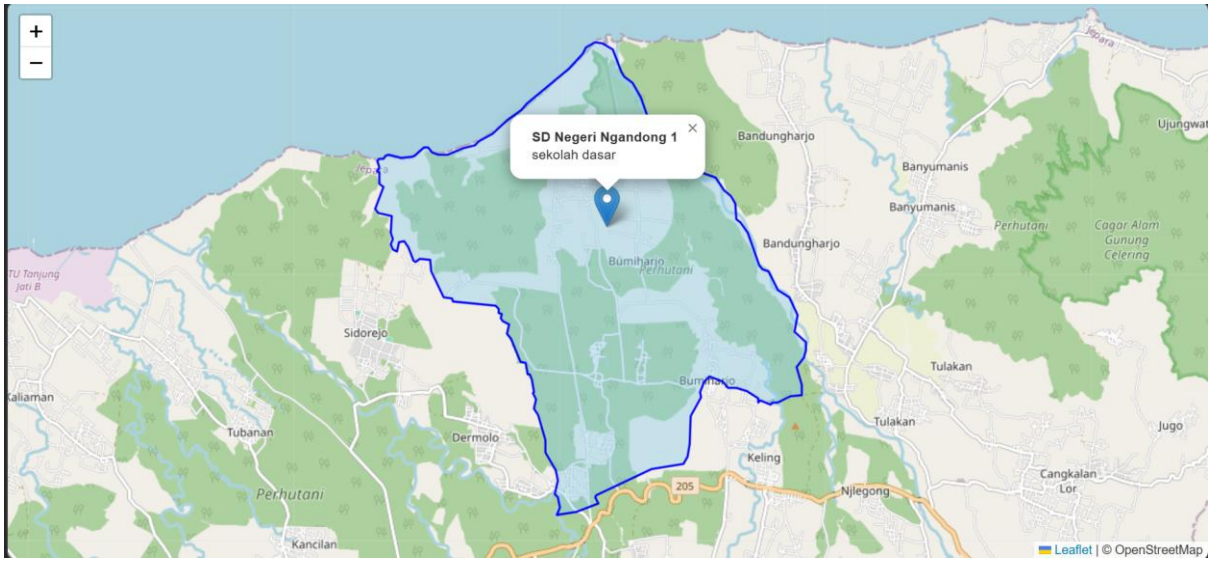
1. **Alur Tambah Data (Create)**
    - Pengguna mengisi form lokasi di web (nama lokasi, deskripsi, koordinat)

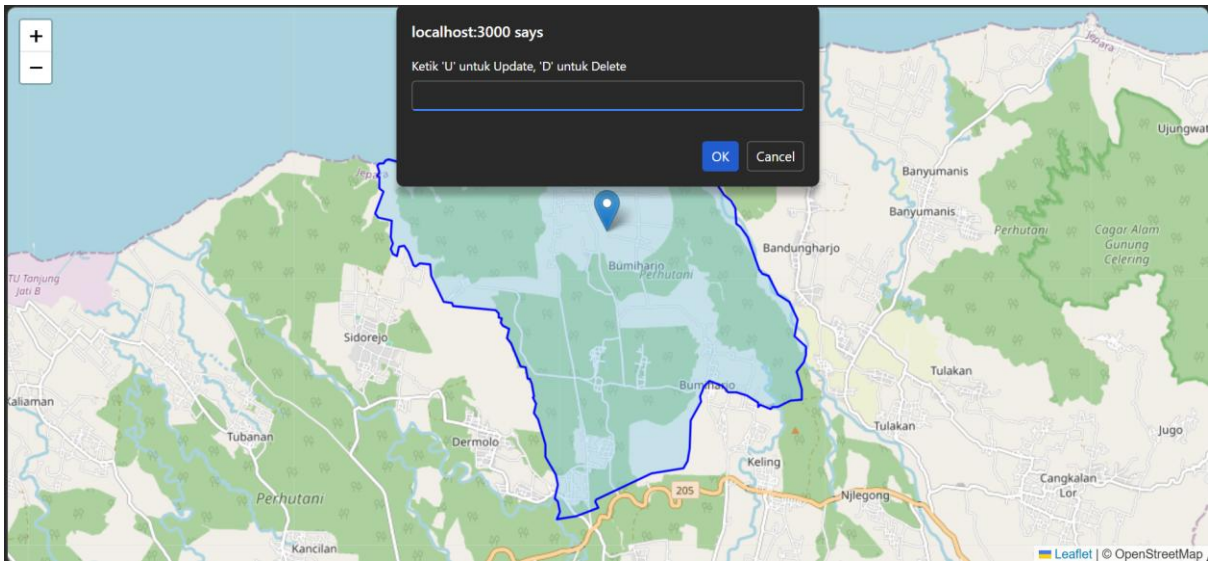
- Frontend mengirim request `POST` ke backend
- Backend menambahkan data ke tabel PostgreSQL
- Frontend menampilkan marker baru di peta secara otomatis

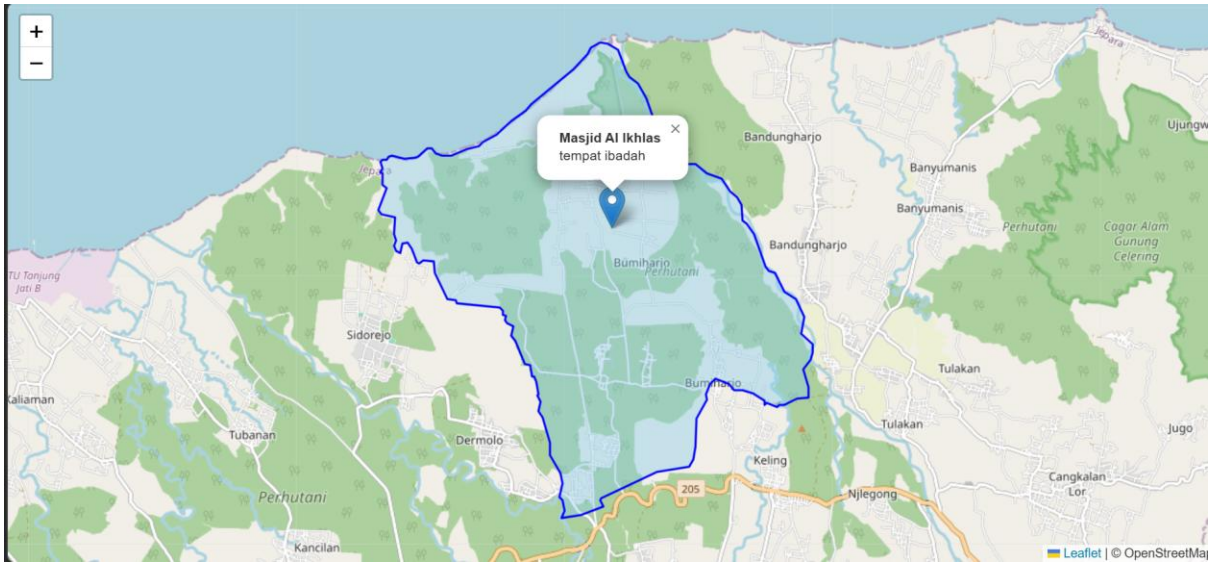
2. **Alur Tampilkan Data (Read)**
   - Saat web dibuka, frontend melakukan request `GET /lokasi`
   - Backend mengirim seluruh data lokasi dari database
   - Marker ditampilkan di peta sesuai koordinat
3. **Alur Ubah Data (Update)**
   - Pengguna memilih marker yang ingin diubah
   - Form edit muncul, diubah di web
   - Frontend mengirim `PUT` request ke backend
   - Backend memperbarui data di database
   - Marker diperbarui di peta
4. **Alur Hapus Data (Delete)**
   - Pengguna memilih marker yang ingin dihapus
   - Frontend mengirim `DELETE` request ke backend
   - Backend menghapus data dari database
   - Marker hilang dari peta

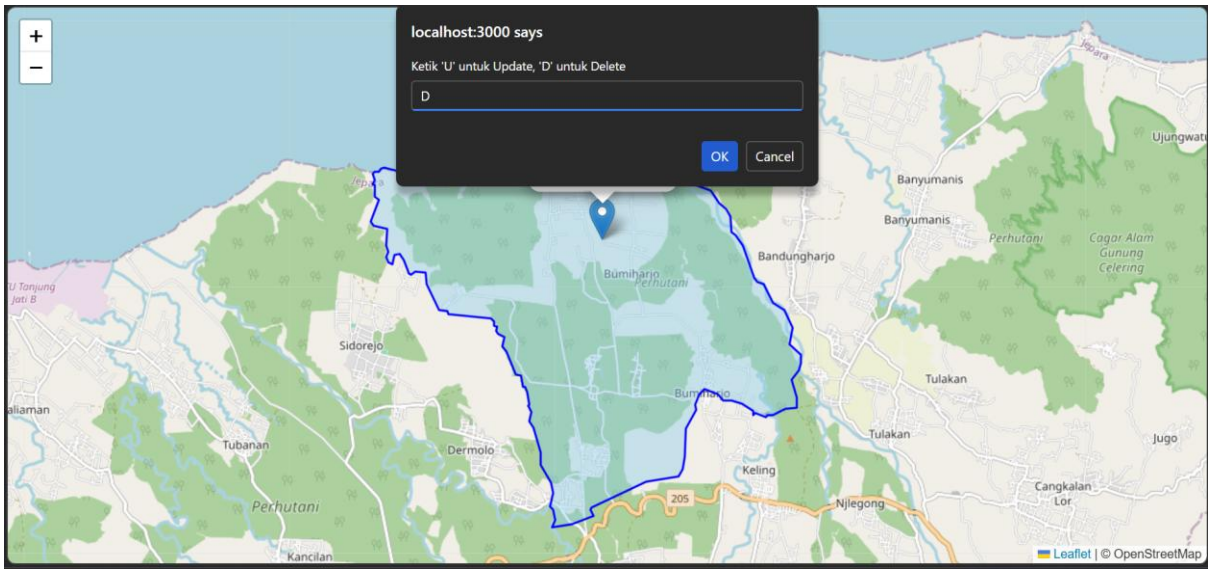


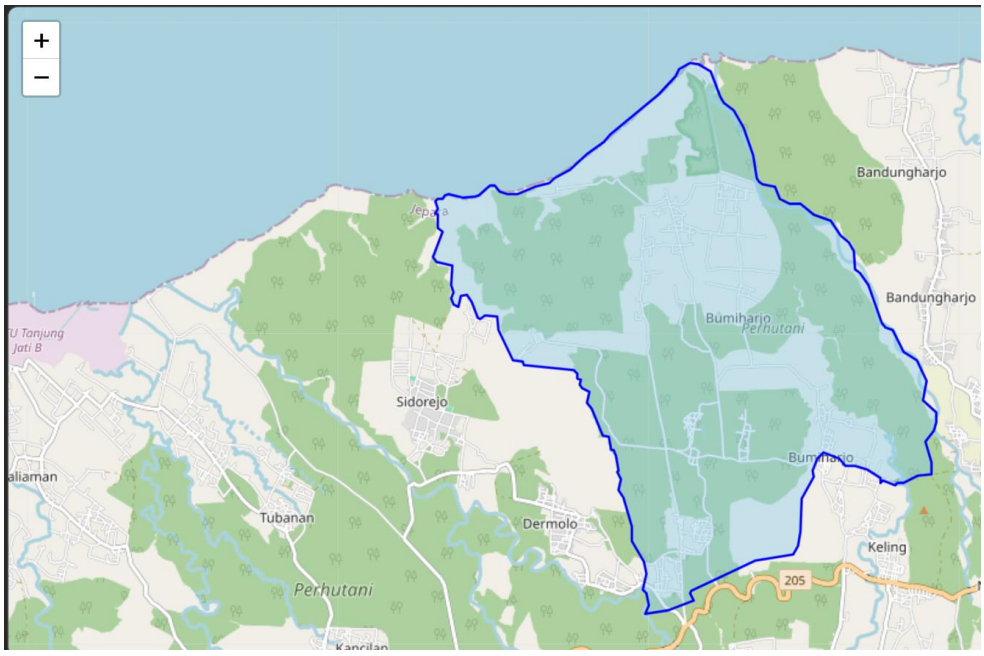

Bandungharjo
Banyumanis
Banyumanis
Bandungharjo
Perhutani
Cagar Alam Gunung Celering
Bumiharjo
Perhutani
Sidorejo
Tulakan
Tulakan
Jugo
Tubanan
Dermolo
Keling
Cangkalan Lor
Perhutani
205
Njlegong
Kancilan
Leaflet | © OpenStreetMap


SD Negeri Ngandong 1
sekolah dasar
Bandungharjo
Banyumanis
Banyumanis
Bandungharjo
Perhutani
Cagar Alam Gunung Celering
Bumiharjo
Perhutani
Sidorejo
Tulakan
Tulakan
Jugo
Tubanan
Dermolo
Keling
Cangkalan Lor
Perhutani
205
Njlegong
Kancilan
Leaflet | © OpenStreetMap


localhost:3000 says
Ketik 'U' untuk Update, 'D' untuk Delete
OK
Cancel
Bandungharjo
Banyumanis
Banyumanis
Bandungharjo
Perhutani
Cagar Alam Gunung Celering
Bumiharjo
Perhutani
Sidorejo
Tulakan
Tulakan
Jugo
Tubanan
Dermolo
Keling
Cangkalan Lor
Perhutani
205
Njlegong
Kancilan
Leaflet | © OpenStreetMap


Masjid Al Ikhlas
tempat ibadah
Bandungharjo
Banyumanis
Bumiharjo
Perhutani
Sidorejo
Tulakan
Tubanan
Dermolo
Keling
Njlegong
Cangkalan
Jugo
205
Leaflet | © OpenStreetMap


localhost:3000 says
Ketik 'U' untuk Update, 'D' untuk Delete
D
OK
Cancel
Bandungharjo
Banyumanis
Bumiharjo
Perhutani
Sidorejo
Tulakan
Tubanan
Dermolo
Keling
Njlegong
Cangkalan
Jugo
205
Leaflet | © OpenStreetMap


Bandungharjo
Bumiharjo
Perhutani
Sidorejo
Tubanan
Dermolo
Keling
205

BAB V

KESIMPULAN DAN PENUTUP

5.1 Kesimpulan

Berdasarkan hasil perancangan, implementasi, dan pengujian Sistem Informasi Geografis (SIG) berbasis web ini, dapat ditarik beberapa kesimpulan:

1. Sistem berhasil dibuat menggunakan **QGIS** untuk pengolahan data spasial, **PostgreSQL** sebagai basis data penyimpanan lokasi, dan **Node.js** sebagai backend untuk menghubungkan database dengan aplikasi web.
2. Aplikasi web mampu menampilkan **peta interaktif** dengan marker lokasi yang dapat dilihat secara langsung di browser.
3. Fitur **CRUD (Create, Read, Update, Delete)** berjalan dengan baik, sehingga pengguna dapat menambahkan, mengubah, dan menghapus data lokasi beserta deskripsinya dengan mudah.
4. Integrasi antara QGIS, PostgreSQL, dan Node.js berjalan secara efektif, menghasilkan sistem SIG sederhana yang dapat diakses melalui web secara interaktif.

5.2 Saran

Untuk pengembangan dan peningkatan sistem di masa depan, beberapa saran yang dapat diberikan adalah:

1. Menambahkan **autentikasi pengguna** agar data lokasi lebih aman dan dapat diatur hak aksesnya.
2. Menambahkan **fitur filter dan kategori lokasi** agar pengguna dapat mencari lokasi tertentu dengan lebih cepat.
3. Mengoptimalkan **tampilan peta dan performa aplikasi**, terutama jika jumlah marker lokasi banyak.
4. Menyediakan **backup data otomatis** untuk menjaga integritas database.