PENGEMBANGAN MODUL *BACKEND* SISTEM PENILAIAN TINGKAT KEPARAHAN AREA PASCA KEBAKARAN HUTAN DAN LAHAN

ANDYANA LILMUTTAQINA MAFAZA
G6401211002

PROGRAM SARJANA ILMU KOMPUTER
SEKOLAH SAINS DATA MATEMATIKA DAN INFORMATIKA
INSTITUT PERTANIAN BOGOR
BOGOR
2025

PERNYATAAN MENGENAI SKRIPSI DAN SUMBER INFORMASI SERTA PELIMPAHAN HAK CIPTA

Dengan ini saya menyatakan bahwa laporan akhir dengan judul "Pengembangan Modul *Backend* Dengan Sistem Penilaian Tingkat Keparahan Area Pasca Kebakaran Hutan Dan Lahan" adalah karya saya dengan arahan dari dosen pembimbing dan belum diajukan dalam bentuk apa pun kepada perguruan tinggi mana pun. Sumber informasi yang berasal atau dikutip dari karya yang diterbitkan maupun tidak diterbitkan dari penulis lain telah disebutkan dalam teks dan dicantumkan dalam Daftar Pustaka di bagian akhir laporan akhir ini.

Dengan ini saya melimpahkan hak cipta dari karya tulis saya kepada Institut Pertanian Bogor.

Bogor, Maret 2024

Andyana Lilmuttaqina Mafaza

G6401211002

ABSTRAK

ANDYANA LILMUTTAQINA MAFAZA. Pengembangan Modul *Backend* Sistem Penilaian Tingkat Keparahan Area Pasca Kebakaran Hutan dan Lahan. Dibimbing oleh IMAS SUKAESIH SITANGGANG.

Penilaian tingkat keparahan pasca kebakaran hutan dan lahan diperlukan untuk mendukung pemulihan ekosistem, penegakan hukum, dan perlindungan kawasan terdampak. SIPARTAN dikembangkan untuk mengotomatisasi proses ini, namun modul *backend* sebelumnya memiliki keterbatasan, seperti tidak adanya verifikasi email, *login* menggunakan akun Google, dukungan untuk penilaian ulang lahan, serta fitur *pagination*, *search*, dan *filter*. Selain itu, sistem belum menerapkan *role-based access control* (RBAC) dan masih menggunakan titik koordinat untuk penyimpanan, yang kurang akurat dalam merepresentasikan area terdampak kebakaran. Penelitian ini bertujuan untuk mengatasi keterbatasan tersebut dengan mengembangkan modul *backend* SIPARTAN. Penelitian ini berhasil dalam menerapkan fitur verifikasi email, *login* menggunakan akun Google, dukungan penilaian ulang lahan, serta RBAC, yang meningkatkan keamanan dan fleksibilitas sistem. Selain itu, sistem kini telah mendukung *pagination*, *search*, dan *filter*, serta mengubah penyimpanan data lahan dari titik koordinat menjadi poligon, sehingga lebih akurat dalam merepresentasikan luas area terdampak. Seluruh API yang dikembangkan, terdiri dari kelompok *Auth*, *User*, Info, Lahan, dan Observasi, diuji menggunakan *blackbox testing* dengan 104 skenario pengujian yang seluruhnya berhasil. *Deployment* dilakukan menggunakan Docker dengan menggunakan *docker compose*, memastikan bahwa modul *backend* dapat diakses oleh *frontend* dan *mobile*. Hasil penelitian ini meningkatkan efisiensi, keamanan, dan akurasi penilaian pasca kebakaran hutan dan lahan.

Kata kunci : *API, kebakaran hutan dan lahan, penilaian tingkat keparahan, role-based access control (RBAC), SIPARTAN*

ABSTRACT

ANDYANA LILMUTTAQINA MAFAZA. *Development of the Backend Module for the Severity Assessment System in Post-Forest and Land Fire Areas.*. *Supervised by* IMAS SUKAESIH SITANGGANG.

The assessment of post-wildfire severity is essential to support ecosystem recovery, law enforcement, and the protection of affected areas. SIPARTAN was developed to automate this process; however, the previous backend module had several limitations, such as the absence of email verification, login via Google accounts, support for land reassessment, as well as pagination, search, and filter features. Additionally, the system had not yet implemented role-based access control (RBAC) and still used coordinate points for data storage, which was less accurate in representing burned areas. This study aims to address these limitations by developing the SIPARTAN backend module. The study successfully implemented email verification, Google login, land reassessment support, and RBAC, enhancing system security and flexibility. Moreover, the system now supports pagination,

search, and filter functions, and replaces coordinate point-based storage with polygons, improving accuracy in representing affected areas. All developed APIs, categorized into Auth, User, Info, Land, and Observation groups, were tested using black-box testing with 104 test scenarios, all of which were successful. Deployment was carried out using Docker and Docker Compose, ensuring backend accessibility for both frontend and mobile applications. This study improves the efficiency, security, and accuracy of post-wildfire severity assessments.

Keywords: API, forest and land fires, role-based access control (RBAC), severity assessment, SIPARTAN

PENGEMBANGAN MODUL *BACKEND* SISTEM PENILAIAN TINGKAT KEPARAHAN AREA PASCA KEBAKARAN HUTAN DAN LAHAN

ANDYANA LILMUTTAQINA MAFAZA

Skripsi sebagai salah satu syarat untuk memperoleh gelar
Sarjana pada Program Sarjana Ilmu Komputer

PROGRAM SARJANA ILMU KOMPUTER
SEKOLAH SAINS DATA MATEMATIKA DAN INFORMATIKA
INSTITUT PERTANIAN BOGOR
BOGOR
2025

Tim Penguji pada Ujian Skripsi:
Muhammad Asyhar Agmalaro, S.Si, M.Kom
Hari Agung Adrianto, S.Kom., M.Si., Ph.D

Judul skripsi : Pengembangan Modul *Backend* Sistem Penilaian Tingkat Keparahan Area Pasca Kebakaran Hutan dan Lahan
Nama : Andyana Lilmuttaqina Mafaza
NIM : G6401211002

Disetujui oleh

Pembimbing 1:
Prof. Dr. Imas Sukaesih Sitanggang, S.Si., M.Kom.
197501301998022001

Diketahui oleh

Ketua Program Sarjana Ilmu Komputer:
Dr. Sony Hartono Wijaya, S.Kom., M.Kom.
198108092008121002

Tanggal Ujian:
6 Maret 2025

Tanggal Lulus:

PRAKATA

Puji dan syukur penulis panjatkan kepada Allah subhanaahu wa ta'ala atas segala karunia-Nya sehingga karya ilmiah dengan judul "Pengembangan Modul *Backend* Sistem Penilaian Tingkat Keparahan Area Pasca Kebakaran Hutan dan Lahan" ini dapat diselesaikan.

Penulis juga ingin menyampaikan terima kasih yang sebesar-besarnya kepada semua pihak yang telah memberikan dukungan dan bimbingan selama proses penulisan ini. Terima kasih penulis ucapkan kepada:

1. Ayah, Bunda, beserta adik-adik penulis dan seluruh keluarga yang senantiasa memberikan dukungan kepada penulis,
2. Ibu Prof. Dr. Imas Sukaesih Sitanggang S.Si., M.Kom., selaku dosen pembimbing yang senantiasa memberikan bimbingan dan dorongan kepada penulis dalam penyelesaian skripsi,
3. Bapak Firman Ardiansyah S.Kom., M.Si., Bapak Muhammad Asyhar Agmalaro, S.Si., M.Kom., Bapak Hari Agung Adrianto S.Kom., M.Si., Ph.D., selaku moderator pada sesi kolokium, pra-seminar hasil, dan penguji pada seminar hasil yang telah memberikan saran untuk meningkatkan penelitian ini menjadi lebih baik,
4. Tim pengembang SIPARTAN, Rifqi Fauzan Azzam dan Khalid Zia Rabbani yang telah bekerja sama dan membantu penulis dalam pembuatan modul *backend*,
5. Hariz Krisha Muhammad, Irfan Alamsyah, Mochammad Kevin Ariobimo, Naufal Akbar Rahardjo, Rahmad Ilham Sani, Rifqi Fauzan Azzam, selaku warga alhur yang senantiasa memberikan bantuan kepada penulis.
6. Teman-teman warga *Kevallen Institute of Technology* yang senantiasa memberi masukan kepada penulis untuk menjadi lebih baik.
7. Teman-teman PPM Baitul Ilmaini, IT TODAY 2022, Sekben PPM BI, EMOJIFY 4.0, EMOJIFY 5.0, HPMB yang telah memberikan kesempatan kepada penulis untuk menjadi lebih baik lagi.

Semoga karya ilmiah ini bermanfaat bagi pihak yang membutuhkan dan bagi kemajuan ilmu pengetahuan.

Bogor, Maret 2024

Andyana Lilmuttaqina Mafaza

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

DAFTAR LAMPIRAN

I PENDAHULUAN

1.1 Latar Belakang

Kebakaran hutan dan lahan (karhutla) di Indonesia memperlihatkan dampak yang luas dan kompleks terhadap lingkungan, ekonomi, dan kesehatan masyarakat. Praktik pembukaan lahan melalui pembakaran yang tidak terkontrol, dipadukan dengan kondisi iklim seperti musim kemarau yang panjang, menjadi faktor utama kejadian karhutla (Yusuf *et al*. 2019). Berdasarkan data dari laman SiPongi+, yang dikelola oleh Kementerian Lingkungan Hidup dan Kehutanan (KLHK), menunjukkan fluktuasi luas karhutla dalam beberapa tahun terakhir. Pada tahun 2021, luas karhutla tercatat sebesar 358.867,00 hektare. Setelahnya, pada tahun 2022 terjadi penurunan luas karhutla menjadi 204.894,00 hektare. Kemudian terjadi peningkatan tajam pada tahun 2023 menjadi 1.161.192,90 hektare (MenLHK 2024).

Penilaian tingkat keparahan area pasca karhutla menjadi krusial dalam upaya pemulihan dan rehabilitasi ekosistem yang terdampak, sebab informasi yang diperoleh membantu mengidentifikasi daerah yang membutuhkan rehabilitasi (KLHK 2018). Selain itu, hasil penilaian ini berperan penting dalam proses penegakan hukum dan menjadi acuan dalam kegiatan pemulihan serta perlindungan kawasan lindung atau konservasi, memastikan langkah pemulihan yang efektif berdasarkan kondisi lahan pasca kebakaran (Syaufina 2017). Oleh karena itu, penilaian akurat tingkat keparahan karhutla menjadi langkah esensial untuk memastikan langkah pemulihan lingkungan yang tepat, dengan metode penilaian yang sudah ada seperti *Normalized Burn Ratio* (NBR) dan penilaian manual langsung di lapangan. Penilaian manual dengan metode *fire severity*, memungkinkan pengumpulan data yang lebih detail dan akurat mengenai kondisi lahan pasca karhutla. Akan tetapi prosesnya yang memakan waktu menunjukkan kebutuhan akan pengembangan teknologi untuk meningkatkan efisiensi proses penilaian.

Saat ini sudah dikembangkan sebuah sistem yang dapat mengefisiensikan proses penilaian lahan pasca karhutla. Sistem Penilaian Tingkat Keparahan Area Pasca Kebakaran Hutan dan Lahan (SIPARTAN) meningkatkan efisiensi proses penilaian karhutla yang sebelumnya dilakukan secara manual di lapangan oleh Afina (2022) dengan metode *fire severity* (Baihaqi 2024). SIPARTAN dirancang untuk menyimpan data penilaian tingkat keparahan karhutla dan melakukan perhitungan terhadap data tersebut, menghasilkan interpretasi keparahan tingkat keparahan karhutla (Maulana 2024).

Terdapat tiga modul utama yang menyusun SIPARTAN yaitu, modul *backend*, *frontend*, dan *mobile*. Modul *backend* telah dikembangkan pada penelitian sebelumnya oleh Maulana (2024) sebagai modul utama untuk menunjang modul *mobile* dan modul *frontend*. Fitur utama modul *backend* adalah penilaian area pasca karhutla. Modul ini akan menerima data dari modul *mobile* atau *frontend*, lalu memberikan hasil penilaian berdasarkan metode yang dikembangkan oleh Syaufina (2017). Modul *frontend* yang dikembangkan oleh Baihaqi (2024) menyediakan antarmuka berbasis web yang memungkinkan pengguna mengakses dan mengelola laporan penilaian. Sementara itu, modul *mobile* yang dikembangkan oleh Hutapea (2024) memungkinkan pengguna melakukan penilaian di lapangan melalui aplikasi

Android dengan fitur input data langsung. Setiap modul berperan dalam mendukung alur kerja secara terintegrasi, di mana data yang dikirimkan dari aplikasi *mobile* dan *frontend* diproses oleh *backend*, dan hasilnya dapat diakses secara *real-time* oleh pengguna melalui kedua platform tersebut.

Penelitian ini bertujuan untuk melanjutkan pengembangan modul *backend* SIPARTAN berdasarkan saran dan kekurangan dari penelitian Maulana (2024). Beberapa kekurangan utama yang perlu diperbaiki adalah belum tersedianya verifikasi email pada registrasi dan opsi *login* dengan akun Google, yang dapat meningkatkan keamanan dan kemudahan akses pengguna. Selain itu, sistem belum mendukung penilaian ulang pada lahan yang sama, sehingga berisiko terjadi duplikasi data jika kebakaran atau perubahan kondisi terjadi kembali. *Backend* SIPARTAN juga belum menerapkan *pagination*, *search*, dan *filter*, yang penting untuk mempercepat pemrosesan data dan mengurangi beban pada basis data.

Keterbatasan lain adalah tidak adanya perbedaan peran (*role*) antara admin dan pengguna biasa, sehingga pengelolaan hak akses belum optimal. Selain itu, dalam penilaian lahan masih terbatas pada satu titik koordinat, sehingga belum dapat menggambarkan luas area terdampak, karena modul *backend* belum memanfaatkan PostGIS. Oleh karena itu, penelitian ini akan berfokus pada pengembangan fitur verifikasi email, integrasi login pihak ketiga, penilaian ulang lahan, *role-based access control* (RBAC), dan *pagination*. Metode penyimpanan data plot juga akan diubah dari titik koordinat menjadi poligon untuk memperjelas luas area karhutla yang terdampak.

1.2 Rumusan Masalah

Pada penelitian sebelumnya Maulana (2024) telah mengembangkan modul *backend* SIPARTAN. Namun, masih terdapat beberapa kekurangan dari modul *backend* yang ada seperti:

1. Belum terdapat sistem verifikasi email.
2. Belum tersedia opsi *login* dengan akun pihak ketiga seperti Google.
3. Sistem belum mendukung penilaian ulang lahan yang sama.
4. Belum diterapkan *pagination*, *search*, dan *filter*.
5. Belum ada pembagian hak akses dengan *role-based access control* (RBAC).
6. Penyimpanan koordinat lahan masih terbatas pada satu titik, belum mendukung bentuk poligon.

Rumusan masalah pada penelitian ini merupakan bagaimana cara mengembangkan fitur pada modul *backend* SIPARTAN berdasarkan kekurangan dan saran dari penelitian Maulana (2024).

1.3 Tujuan

Berdasarkan rumusan masalah, tujuan dari penelitian ini adalah mengembangkan modul *backend* Sistem Penilaian Tingkat Keparahan Area Pasca Kebakaran Hutan dan Lahan (SIPARTAN) yang sebelumnya dikembangkan oleh Maulana (2024).

1.4 Manfaat

Pengembangan modul *backend* ini bertujuan untuk meningkatkan efisiensi digitalisasi dalam penilaian lahan pasca kebakaran hutan. Dengan sistem yang lebih terstruktur, tim lapangan dapat mengevaluasi area pasca karhutla lebih cepat dan akurat. Selain itu, integrasi dengan aplikasi web dan *mobile* akan mempermudah akses informasi serta mendukung pengambilan keputusan yang lebih tepat dalam pengelolaan lahan dan hutan pasca karhutla.

1.5 Ruang Lingkup

Ruang lingkup dari penelitian ini adalah Modul *backend* Sistem Penilaian Tingkat Keparahan Area Pasca Kebakaran Hutan dan Lahan (SIPARTAN) yang dikembangkan oleh Maulana (2024).

II TINJAUAN PUSTAKA

2.1 Kebakaran hutan dan lahan

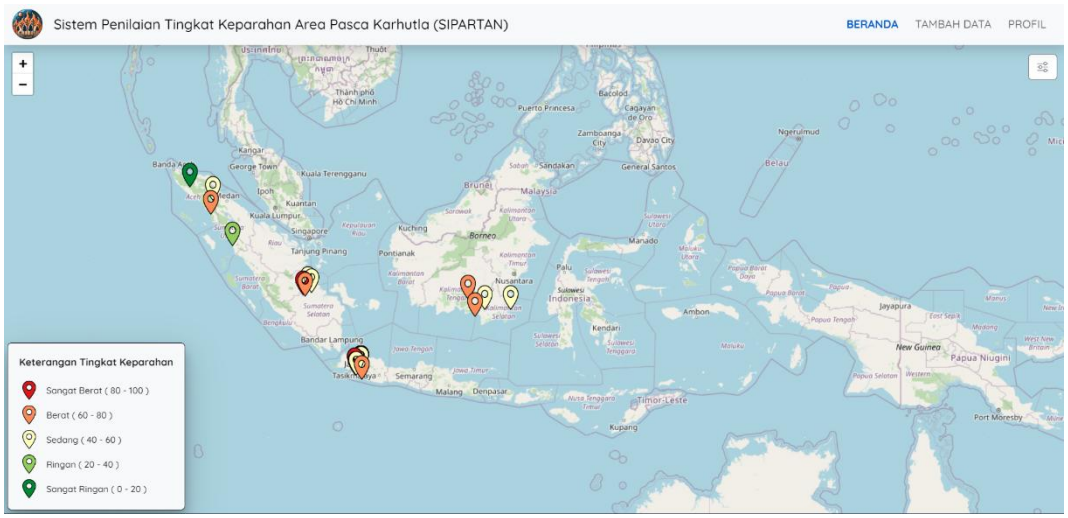
Kebakaran hutan dan lahan (karhutla) di Indonesia merupakan bencana alam nasional yang terjadi akibat faktor alam dan manusia, mempengaruhi ekologi, ekonomi, sosial-budaya, dan politik (KLHK 2018). Faktor alami seperti sambaran petir, suhu panas yang ekstrim, dan bahan bakar alam yang mudah terbakar dapat memicu karhutla (Sari 2023). Namun, kontribusi manusia dalam bentuk persiapan lahan dengan menggunakan api, praktik *illegal logging* dan aktivitas lainnya, menjadi penyebab utama terjadinya karhutla di Indonesia. Rachman *et al*. (2020) menekankan bahwa faktor utama penyebab karhutla adalah manusia, dengan kontribusi sekitar 99.9%, sementara faktor alami hanya berkontribusi sekitar 0.01%.

Karhutla tidak hanya merugikan dari segi kesehatan dan ekonomi masyarakat lokal tetapi juga memiliki implikasi politik internasional, mempengaruhi hubungan Indonesia dengan negara tetangga seperti Malaysia dan Singapura karena kabut asap lintas batas (Pasai 2020). Faktor seperti pertumbuhan ekonomi yang mendorong penggunaan api dalam pengelolaan lahan dan kurangnya kesadaran masyarakat akan bahaya kebakaran memperparah situasi, menunjukkan perlunya pengendalian dan pengawasan yang lebih efektif untuk mencegah kejadian karhutla di masa depan (Edwards *et al*. 2019).

Karhutla di Indonesia telah menghanguskan jutaan hektar hutan, menyebabkan degradasi ekosistem gambut serta ancaman terhadap keanekaragaman hayati, merusak ekosistem gambut yang luas, dan menimbulkan dampak mendalam (Prayoga dan Koestoer 2021). Upaya penanggulangan karhutla membutuhkan pendekatan terintegrasi yang melibatkan pemerintah, masyarakat, dan pemangku kepentingan lainnya untuk meminimalkan dampak bencana ini dan merestorasi area yang terkena dampak. Pentingnya pengawasan dan kebijakan pengelolaan hutan yang efektif menjadi kunci untuk mengatasi tantangan karhutla dan menjaga keberlanjutan lingkungan hidup di Indonesia (Yulianti 2018).

2.2 SIPARTAN

Sistem Penilaian Tingkat Keparahan Area Pasca Kebakaran Hutan dan Lahan (SIPARTAN) merupakan sistem yang dirancang untuk menyimpan dan menganalisis data penilaian dampak karhutla, sehingga dapat menghasilkan interpretasi keparahan secara akurat dan efisien (Maulana 2024). Sistem ini terdiri dari tiga modul utama: *frontend*, *backend*, dan *mobile*, yang memungkinkan penerapannya dalam bentuk *website* dan aplikasi *mobile*. SIPARTAN dikembangkan untuk memberikan informasi cepat dan akurat mengenai keparahan area pasca karhutla, dengan modul aplikasi mobile yang memfasilitasi penginputan dan pengumpulan data oleh petugas lapangan. Data yang dikumpulkan akan disimpan dalam basis data di modul *backend*, kemudian diakses dan divisualisasikan oleh modul frontend melalui REST-API dalam bentuk Sistem Informasi Geografis menggunakan komponen peta library *LeafletJs* (Baihaqi 2024). Dengan kemampuan ini, pihak berwenang dapat menggunakan hasil penilaian SIPARTAN untuk mengambil keputusan yang lebih efektif dan efisien dalam menangani situasi pasca kebakaran (Hutapea 2024). Adapun tampilan dari modul *frontend* dan *mobile* dari SIPARTAN dapat dilihat pada Gambar 1 dan Gambar 2.



Gambar 1 Tampilan modul *frontend* SIPARTAN

Gambar 2 Tampilan modul *mobile* SIPARTAN

SIPARTAN memiliki berbagai macam fitur yang mendukung penilaian area pasca karhutla. Fitur utama dari sistem ini adalah penilaian area pasca karhutla, di mana pengguna dapat melakukan penilaian lahan dengan mengisi formulir yang berisi data umum tentang lahan dan memilih indikator-indikator kerusakan yang terjadi. Fitur pendukung SIPARTAN mencakup mekanisme *authentication* dan *authorization* untuk memastikan keamanan akses. Sistem ini juga menampilkan hasil penilaian di peta, memungkinkan pengguna melihat lokasi-lokasi penilaian di Indonesia. Pengguna dapat melihat detail suatu penilaian dengan jelas dan ringkas, serta melakukan edit atau perubahan data penilaian setelah penilaian selesai. Hasil penilaian dapat diekspor menjadi format PDF yang dapat diunduh, dan pengguna juga memiliki opsi untuk menghapus penilaian yang telah dilakukan. Selain itu, pengguna dapat mengunggah foto untuk setiap indikator yang dipilih selama proses penilaian lahan pasca karhutla, dan *file* foto tersebut akan tersimpan di *server*. Sistem ini juga memastikan bahwa hanya pengguna yang berwenang yang dapat mengakses dan mengelola data penilaian.

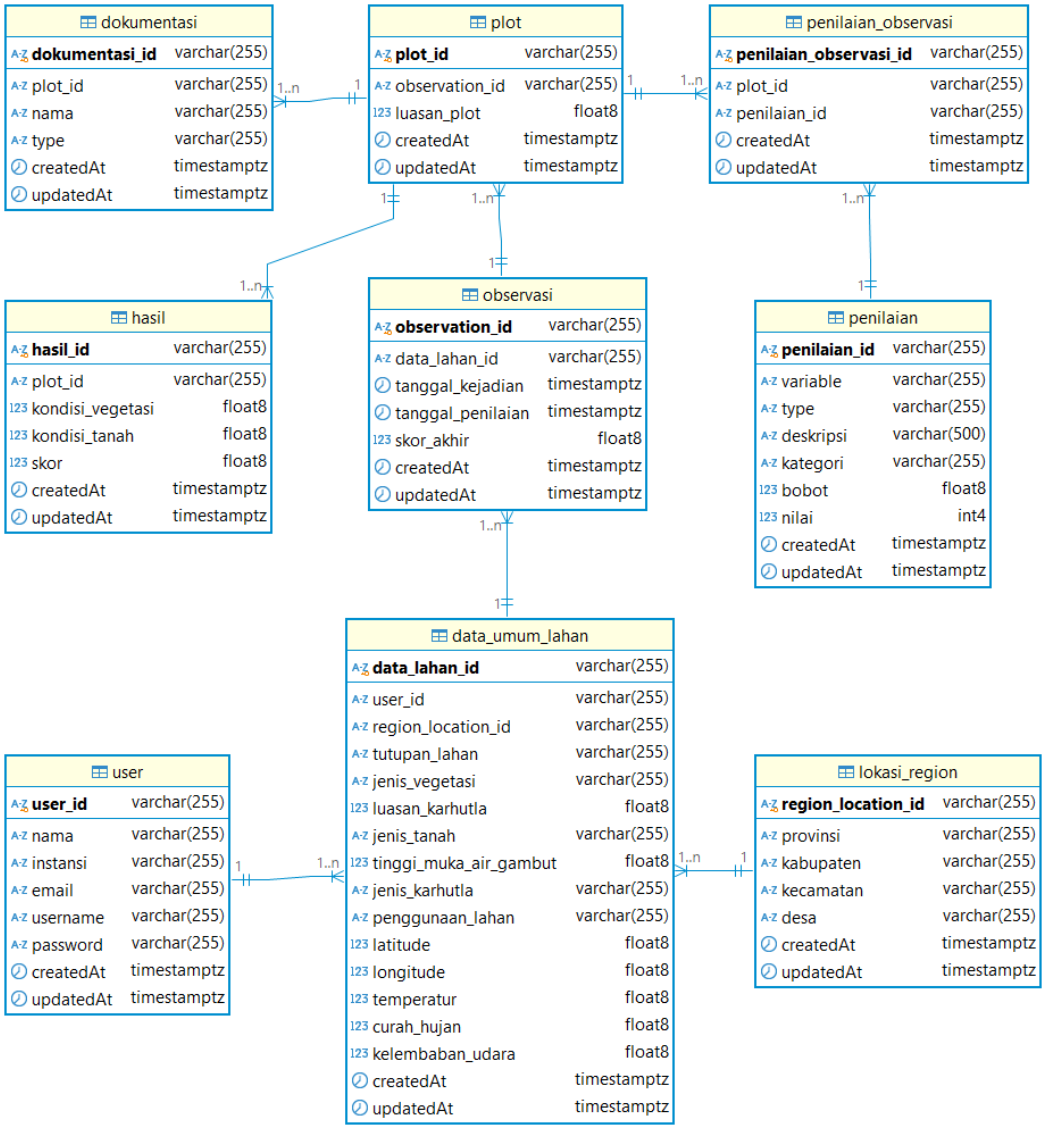
2.3 Modul *Backend* SIPARTAN pada Penelitian Maulana (2024)

Pada pengembangan sebelumnya, Maulana (2024) mengembangkan modul *backend* SIPARTAN dengan menerapkan arsitektur *REST* API serta merancang *Entity-Relationship Diagram* (ERD) untuk mengelola data dalam sistem. ERD SIPARTAN, sebagaimana ditampilkan pada Gambar 3 terdiri dari sembilan entitas utama, yaitu user, lokasi_region, data_umum_lahan, plot, dokumentasi, observasi, penilaian_observasi, penilaian, dan hasil.

Entitas *user* berfungsi untuk menyimpan informasi pengguna dan memiliki hubungan *one-to-many* dengan data_umum_lahan, yang menyimpan informasi karakteristik lahan seperti jenis tanah, curah hujan, dan tutupan lahan. Entitas lokasi_region digunakan untuk menyimpan informasi wilayah administratif, seperti provinsi, kabupaten, kecamatan, dan desa, serta memiliki hubungan *one-to-many* dengan data_umum_lahan, memungkinkan satu wilayah memiliki banyak data lahan. Entitas plot digunakan untuk memecah lahan menjadi unit-unit kecil yang lebih spesifik untuk penilaian dan memiliki hubungan *one-to-many* dengan data_umum_lahan. Observasi dan penilaian dilakukan di tingkat plot, yang disimpan dalam entitas observasi, sedangkan indikator penilaian disimpan dalam entitas penilaian. Relasi *many-to-many* antara observasi dan penilaian dikelola melalui entitas penilaian_observasi, yang mencatat setiap indikator yang digunakan dalam sebuah observasi.

Selain itu, entitas hasil berfungsi untuk menyimpan evaluasi akhir pada setiap plot, termasuk skor tingkat keparahan pasca karhutla. Entitas dokumentasi menyimpan *file* foto yang terkait dengan observasi, dengan hubungan *one-to-many* dengan plot, memungkinkan setiap plot memiliki dokumentasi yang relevan. Dengan desain ini, sistem SIPARTAN mampu menyimpan dan mengelola data penilaian lahan secara lebih terstruktur, memungkinkan pemisahan informasi lahan, observasi, dan indikator penilaian, sehingga proses pengolahan data menjadi lebih optimal.

Dalam implementasinya, REST API yang dikembangkan pada penelitian sebelumnya dibagi menjadi tiga kelompok utama, yaitu kelompok API lahan, kelompok API observasi, dan kelompok API *user*. Setiap kelompok API memiliki peran spesifik dalam mendukung pengelolaan data dalam sistem SIPARTAN.



Gambar 3 ERD SIPARTAN (Maulana 2024)

Kelompok API lahan menangani fitur-fitur terkait pengelolaan data lahan, termasuk pembuatan, pembaruan, penghapusan, dan pengambilan data lahan. Kelompok API observasi menangani fitur-fitur yang terkait dengan proses penilaian lahan, yang mencakup penginputan indikator penilaian, dokumentasi hasil observasi, serta pengolahan data hasil observasi lahan. Kelompok API *user* menangani proses autentikasi dan otorisasi pengguna, mencakup registrasi, *login*, serta pengelolaan informasi akun pengguna. Adapun detail masing-masing kelompok API dapat dilihat pada Tabel 1, Tabel 2, dan Tabel 3.

Rancangan ERD SIPARTAN dan REST API yang dikembangkan dalam penelitian Maulana (2024) memungkinkan sistem untuk mengelola data penilaian lahan. Dengan adanya struktur *database* yang jelas serta pembagian API berdasarkan fungsinya, pengolahan data penilaian karhutla menjadi lebih terstruktur, mendukung proses penyimpanan informasi lahan, hasil observasi, serta autentikasi pengguna dalam satu sistem yang terintegrasi.

Tabel 1 Daftar API kelompok API lahan (Maulana 2024)

No	*Endpoint*	Metode	Deskripsi
1	/lahan-karhutla	*POST*	Membuat data lahan baru yang akan dinilai tingkat keparahannya
2	/lahan-karhutla/{id}/{obsId}	*PUT*	Melakukan perubahan data pada data lahan yang sudah dibuat sebelumnya
3	/lahan-karhutla/{id}	*DELETE*	Menghapus data lahan yang sudah pernah dibuat
4	/lahan-karhutla/{id}/{obsId}	*GET*	Mengambil data spesifikasi satu lahan secara detail
5	/lahan-karhutla	*GET*	Mengambil data seluruh lahan yang ada pada program Sipartan
6	/lahan-karhutla/pdf/{id}/{obsId}	*GET*	Melakukan konversi data detail suatu lahan menjadi format PDF lalu mendownloadnya

Tabel 2 Daftar API kelompok API observasi (Maulana 2024)

No	*Endpoint*	Metode	Deskripsi
1	/observasi	*POST*	Melakukan penilaian pada lahan yang sudah dibuat sebelumnya dengan mengirim data indikator-indikator yang dipilih
2	/observasi/penilaian	*POST*	Melakukan input data indikator yang nantinya dapat dipilih oleh *user* sebagai penilaian
3	/observasi/penilaian	*GET*	Mengembalikan data indikator yang sudah di-input sebelumnya untuk menampilkannya kepada *user*
4	/observasi/dokumentasi	*POST*	Melakukan input data dokumentasi untuk setiap indikator pada penilaian
5	/observasi/dokumentasi/:fileName	*GET*	Mengembalikan *file* gambar yang telah diinput untuk ditampilkan ke *user*

Tabel 3 Daftar API kelompok API *user* (Maulana 2024)

No	*Endpoint*	Metode	Deskripsi
1	/user	*POST*	Melakukan registrasi user atau membuat data user yang akan disimpan dalam *database*
2	/user/login	*POST*	Melakukan *login* untuk masuk ke dalam website atau *mobile app* agar dapat mengakses fitur-fitur sistem
3	/user	*GET*	Mengembalikan data *user* yang sedang *login* ke sistem pada perangkat *user*

Meskipun modul *backend* yang dikembangkan oleh Maulana (2024) telah menyediakan REST API yang mendukung proses penilaian tingkat keparahan karhutla, masih terdapat beberapa keterbatasan yang perlu diperbaiki dan dikembangkan lebih lanjut. Salah satu kekurangan utama adalah belum adanya sistem verifikasi email pada saat registrasi, yang menyebabkan akun pengguna dapat dibuat tanpa validasi identitas, sehingga meningkatkan risiko penyalahgunaan sistem. Selain itu, sistem juga belum menyediakan opsi *login* dengan akun pihak ketiga, seperti Google, yang dapat meningkatkan kemudahan akses bagi pengguna serta memperkuat keamanan autentikasi.

Selain aspek keamanan, keterbatasan lain dalam sistem adalah belum didukungnya penilaian ulang lahan yang sama, yang menyebabkan potensi duplikasi data ketika kebakaran terjadi kembali atau kondisi lahan berubah. Selain itu, API yang menangani pengambilan data dalam jumlah besar belum menerapkan mekanisme *pagination*, *search*, dan *filter*, sehingga pemrosesan data menjadi kurang efisien dan dapat membebani performa *database* ketika jumlah data meningkat. Dari sisi manajemen akses, sistem belum menerapkan *role-based access contro*l (RBAC), sehingga tidak ada perbedaan hak akses antara admin dan pengguna biasa, yang berpotensi membatasi kontrol terhadap data yang lebih sensitif. Kekurangan lainnya adalah sistem masih menggunakan titik koordinat tunggal untuk menyimpan lokasi lahan, sehingga belum dapat merepresentasikan luas area terdampak secara akurat. Dengan belum diterapkannya PostGIS, sistem belum mampu menyimpan dan mengelola data dalam bentuk poligon untuk menggambarkan area yang lebih luas dan kompleks.

2.4 Penilaian Tingkat Keparahan Pasca Karhutla

Menurut Syaufina (2017), dalam metode penilaian areal pasca kebakaran hutan, tingkat keparahan dampak karhutla dibagi menjadi lima kategori, yaitu sangat ringan, ringan, sedang, berat, dan sangat berat. Penilaian dilakukan dengan mempertimbangkan dampak kebakaran terhadap kondisi vegetasi dan tanah. Aspek kondisi vegetasi memiliki bobot 70%, dengan indikator kerusakan individu pohon (bobot 46%) dan tingkat keparahan vegetasi terbakar (bobot 24%). Sementara itu, aspek kondisi tanah memiliki bobot 30%, dengan indikator kondisi tanah fisik, kimia, dan biologi serta tingkat keparahan kerusakan tanah. Penetapan tingkat keparahan karhutla didasarkan pada total nilai dari seluruh indikator kondisi vegetasi dan tanah sebagaimana dapat dilihat pada Tabel 4, Tabel 5, dan Tabel 6.

Tabel 4 Data penilaian dampak karhutla

No.	Parameter	Kondisi	Nilai	Bobot (46%)
1	Kematian pohon	Pohon mati	1	8
		Pohon hidup	0	
2	Kerusakan batang			
a	Bagian terbakar	Batang bagian bawah dan bagian atas terbakar	2	1
		Batang bagian bawah terbakar	1	
		Batang tidak terbakar	0	
b	Jenis kerusakan	Hangus dan luka	3	2
		Hangus terbakar	2	
		Luka	1	
		Tidak hangus dan tidak luka	0	
3	Kerusakan tajuk	75% - 100% tajuk terbakar	3	2
		50% ≤75% tajuk terbakar	2	
		25% ≤50% tajuk terbakar	1	
		<25% tajuk terbakar	0	
4	Kerusakan cabang	Patah dan terbakar	3	2
		Terbakar	2	
		Patah	1	
		Tidak patah dan tidak terbakar	0	
5	Kerusakan dedaunan	75% - 100% dedaunan terbakar	3	3
		50% - 75% dedaunan terbakar	2	
		25% - 50% dedaunan terbakar	1	
		<25% dedaunan terbakar	0	
6	Kerusakan akar	Mengalami luka dan terbakar	3	3
		Terbakar	2	
		Luka	1	
		Tidak luka dan terbakar	0	

Dimodifikasi dari Syaufina (2017)

Tabel 5 Indikator penilaian vegetasi berdasarkan tingkat keparahan

No.	Tingkat Keparahan	Kondisi	Nilai	Bobot (24%)
1.	Rendah	Sekurang-kurangnya 50% pohon (diganti tumbuhan) tidak terlihat rusak, sisa tajuk hangus, pucuk terbakar tapi bertunas, dan akar mati. Lebih dari 80% pohon yang terbakar dapat bertahan hidup	1	8
2.	Sedang	20-50% pohon tidak terlihat rusak, 40-80% pohon yang terbakar dapat bertahan hidup	2	
3.	Tinggi	Kurang dari 20% pohon tidak terlihat rusak dan akar mati. Kurang dari 40% pohon yang terbakar dapat bertahan	3	

Dimodifikasi dari Syaufina (2017)

Tabel 6 Indikator penilaian kondisi tanah berdasarkan parameter

No.	Parameter	Kondisi	Nilai	Bobot (30%)
1	Tanah Mineral			
	Rendah	Serasah terbakar habis atau mengarang, tetapi lapisan duff tidak rusak, walaupun permukaannya hangus. Sebagian terakumulasi sisa/sampah berkayu terbakar atau hangus. Tanah mineral tidak berubah, permukaan tanah hitam, abu terjadi untuk waktu yang singkat	1	10
	Sedang	Pengarangan bagian bawah sedang, serasah terbakar habis atau mengarang, dan lapisan duff mengarang atau terbakar habis, lapisan di bawahnya tidak terlihat berubah. Abu berwarna terang. Sampah berkayu terbakar, kecuali log yang mengarang. Abu berwarna putih dan kelabu, arang terjadi pada 1 cm lapisan atas dari tanah mineral, tetapi soil tidak berubah	2	
	Tinggi	Pengarangan bagian bawah dalam, lapisan duff terbakar habis, bagian atas mineral terlihat kemerahan atau oranye. Warna tanah di bawah 1 cm lebih gelap atau mengarang dari bahan organik. Lapisan arang dapat meluas hingga kedalaman 10 cm atau lebih. Log terbakar atau mengarang dalam yang juga terjadi pada tumpukan potongan limbah kayu. Tekstur tanah di lapisan permukan berubah. Semua batang semak terbakar dan hanya batang yang besar mengarang yang terlihat	3	
2	Tanah Gambut			
	Rendah	Bila lapisan gambut yang terbakar sampai kedalaman kurang dari 25 cm	1	10
	Sedang	Bila lapisan gambut yang terbakar sampai kedalaman 25–50 cm	2	
	Tinggi	Bila lapisan gambut yang terbakar sampai kedalaman lebih dari 50 cm	3	

Dimodifikasi dari Syaufina (2017)

Penilaian area pasca karhutla dilakukan dengan mengalikan nilai parameter dengan bobotnya, dan tingkat keparahan ditentukan dari total nilai parameter. Evaluasi kondisi pasca karhutla dilakukan dengan mempertimbangkan indikator-

indikator terpilih, dan hasilnya direpresentasikan dalam tingkat keparahan dari sangat ringan hingga sangat berat, seperti yang tercantum pada Tabel 7.

Tabel 7 Skor tingkat keparahan pasca karhutla

Tingkat keparahan dampak kebakaran	Nilai total (nilai kondisi vegetasi × 70%) + (nilai kondisi tanah × 30%)
Sangat ringan	0-20
Ringan	>20-40
Sedang	>40-60
Berat	>60-80
Sangat berat	>80-100

2.5 REST API

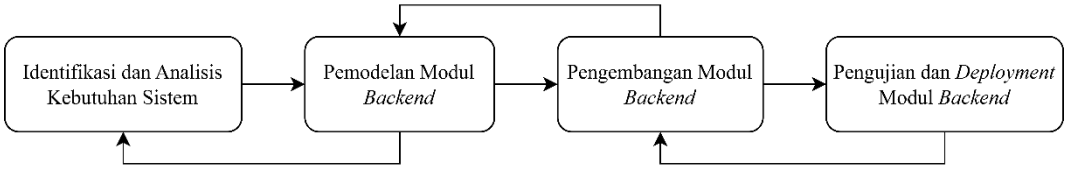
Representational State Transfer (*REST*) adalah sebuah gaya arsitektural yang diperkenalkan oleh Fielding (2000). Arsitektur *REST* digunakan untuk mengembangkan layanan *web* yang dapat mendukung berbagai jenis aplikasi dengan menggunakan serangkaian operasi terbatas (Webber *et al*. 2010). *REST* memfasilitasi komunikasi antara *client* dan *server* melalui protokol HTTP, dengan client mengirimkan *request* dan *server* merespons dengan sumber daya yang diperlukan (Fielding 2000). Arsitektur *REST* tidak hanya mengatur komunikasi antara *client* dan *server*, tetapi juga memperjelas konsep bahwa setiap komponen dalam sistem dianggap sebagai sumber daya yang dapat diakses melalui metode HTTP yang sesuai, seperti *GET*, *POST*, *PUT*, dan *DELETE* (Mumbaikar dan Padiya 2013). Metode *GET* digunakan untuk meminta representasi sumber daya tertentu tanpa mengubahnya, seperti mengunduh data dari server. *POST* digunakan untuk mengirim data ke server untuk membuat sumber daya baru atau memperbarui sumber daya yang ada. *PUT* digunakan untuk memperbarui atau menggantikan sumber daya yang ada di server dengan data yang dikirimkan, sementara *DELETE* digunakan untuk menghapus sumber daya yang ditentukan.

API, atau *Application Programming Interface*, merupakan antarmuka yang memfasilitasi komunikasi antara program perangkat lunak, memungkinkan pertukaran data antara client dan server (Massé 2011). Salah satu implementasi terkenal dari API adalah REST API, yang mengikuti standar arsitektur *REST* dalam membangun layanan web. REST API menggunakan protokol HTTP untuk mengatur interaksi antara *client* dan *server*, memungkinkan pengembangan layanan pengambilan data yang efisien dan mudah dipetakan ke *URL* sumber daya (Soni dan Ranga 2019).

III METODE

3.1 Tahapan Penelitian

Pengembangan modul *backend* SIPARTAN dilaksanakan dalam 4 tahapan utama yaitu identifikasi dan analisis kebutuhan sistem, pemodelan modul *backend*, pengembangan modul *backend*, dan pengujian modul *backend* serta *deployment*. Jika ditemukan kesalahan dalam salah satu tahapan, proses dapat kembali ke tahap sebelumnya untuk memastikan hasil pengembangan modul yang lebih akurat. Diagram tahapan penelitian ini dapat dilihat pada Gambar 4.



Gambar 4 Tahapan penelitian

3.1.1 Identifikasi dan Analisis Kebutuhan Sistem

Pada tahap ini, komunikasi dilakukan dengan stakeholders terkait dan peneliti sebelumnya untuk membahas kebutuhan sistem yang akan dikembangkan. Setelah komunikasi selesai, langkah selanjutnya adalah melakukan studi literatur terhadap penelitian terkait, yakni penelitian yang dilakukan oleh Maulana (2024). Studi literatur ini bertujuan untuk mengidentifikasi dan menganalisis modul *backend* yang telah dikembangkan oleh Maulana (2024) sebagai komponen inti dalam sistem SIPARTAN dan memahami struktur API yang mendukung fungsionalitas sistem secara menyeluruh.

3.1.2 Pemodelan Modul *Backend*

Pada tahap ini, dilakukan penyesuaian desain ERD dengan merujuk pada desain basis data yang telah dikembangkan pada penelitian sebelumnya. Penyesuaian ini mencakup penambahan atau pengurangan kolom pada tabel-tabel tertentu sesuai dengan kebutuhan fitur yang telah ditentukan, guna memastikan bahwa struktur data dapat mengakomodasi perubahan dan peningkatan fungsionalitas sistem. Selain itu, dilakukan perubahan dalam struktur *folder* pada modul *backend* untuk menyesuaikan berbagai komponen tambahan yang diperlukan, sehingga sistem lebih terorganisir dan mudah dikelola dalam proses pengembangan.

3.1.3 Pengembangan Modul *Backend*

Pada tahap ini, pengembangan modul dilakukan dengan melakukan implementasi fitur berdasarkan hasil analisis sistem dan rancangan pemodelan yang telah dibuat sebelumnya. Implementasi fitur akan dilakukan pada modul backend yang sebelumnya dikembangkan oleh Maulana (2024). Adapun *tools* yang akan digunakan meliputi *Nodemailer* untuk mengirim email verifikasi kepada pengguna baru, memastikan identitas akun sebelum diberikan akses penilaian pada aplikasi

SIPARTAN. *Sequelize* akan digunakan untuk memodifikasi struktur database lahan, memungkinkan penilaian dilakukan lebih dari satu kali tanpa redundansi, serta mengimplementasikan query tambahan untuk *pagination*, *search*, dan *filter* data.

Selain itu, MinIO akan di *deploy* untuk mengelola penyimpanan gambar secara terstruktur dan aman, karena MinIO menawarkan solusi penyimpanan berbasis objek serta menyediakan tampilan UI untuk mempermudah manajemen gambar. Passport.js akan diterapkan untuk mengintegrasikan fitur *login* menggunakan akun Google. Fitur *reverse geocoding* akan menggunakan Google Maps API untuk mengkonversi koordinat menjadi alamat lengkap. Terakhir, PostGIS akan digunakan untuk mengubah penyimpanan data dari koordinat menjadi poligon, mendukung representasi area lahan secara lebih akurat dan komprehensif. Tools tersebut dipilih untuk memastikan pengembangan modul berjalan efisien dan sistem dapat berfungsi sesuai kebutuhan fitur yang diidentifikasi.

3.1.4 Pengujian dan *Deployment* Modul *Backend*

Pengujian dilakukan menggunakan metode *blackbox testing* dengan Postman untuk menguji semua *endpoint* secara manual, menggunakan beberapa skenario tes untuk memastikan *endpoint* berjalan. Jika ditemukan ketidaksesuaian, perbaikan akan dilakukan sebelum pengujian ulang. Untuk fitur *login* dengan akun Google, pengujian dilakukan langsung melalui *browser* dengan mengakses *endpoint* autentikasi guna memastikan *login* Google berjalan dengan benar. Setelah semua pengujian berhasil tanpa *error*, modul *backend* akan dideploy ke *server*, memungkinkan akses bagi pengembang *frontend* dan *mobile* untuk integrasi lebih lanjut serta memastikan stabilitas sistem dalam lingkungan produksi.

3.2 Lingkungan Pengembangan

Spesifikasi *hardware* dan *software* yang digunakan dalam penelitian ini sebagai berikut:

a) Spesifikasi *hardware* yang digunakan:
   1. *Processor AMD Ryzen 5 4600H with Radeon Graphics, 3000 Mhz, 6 Core(s), 12 Logical Processor(s)*
   2. Penyimpanan berupa SSD dengan kapasitas 256 GB dan HDD dengan kapasistas 1 TB.
   3. RAM sebesar 16 GB

b) Spesifikasi *software* yang digunakan:
   1. Bahasa pemrograman Javascript
   2. *Runtime environtment* Node.js
   3. *Framework* Express.js
   4. *Object Relational Mapping Sequelize*
   5. Sistem Manajemen Basis Data PostgreSQL
   6. Postman sebagai alat pengujian dan dokumentasi API
   7. MinIO sebagai *object storage*

IV HASIL DAN PEMBAHASAN

Pengembangan fitur pada modul *backend* SIPARTAN pada penelitian ini berfokus pada pada perbaikan beberapa aspek dari modul *backend* yang telah dikembangkan pada penelitian Maulana (2024). Tahapan yang dilakukan pada penelitian ini memiliki empat bagian yaitu identifikasi dan analisis kebutuhan sistem, pemodelan modul *backend*, pengembangan modul *backend*, pengujian dan *deployment* modul *backend*.

4.1 Identifikasi dan Analisis Kebutuhan Sistem

Tahap ini diawali dengan diskusi daring bersama tim pengembang SIPARTAN yang terdiri dari penulis, Khalid Zia Rabbani sebagai pengembang *frontend*, Rifqi Fauzan Azzam sebagai pengembang *mobile*, serta dibimbing oleh Prof. Dr. Imas S. Sitanggang, S.Si, M.Kom. Selain itu, turut hadir peneliti sebelumnya, yaitu Maulana (2024) sebagai pengembang *backend*, Hutapea (2024) sebagai pengembang *mobile*, dan Baihaqi (2024) sebagai pengembang *frontend*.

Pada pertemuan ini, dijelaskan secara umum mengenai alur kerja sistem SIPARTAN serta kekurangan yang terdapat pada sistem saat ini. Setiap pengembang dari penelitian sebelumnya memberikan masukan terkait modul yang telah mereka kembangkan, mencakup kendala yang dihadapi serta potensi perbaikan. Dari diskusi ini, diperoleh beberapa kebutuhan fitur baru yang harus dikembangkan agar sistem SIPARTAN lebih optimal dalam penggunaannya.

Selain diskusi, studi literatur terhadap penelitian sebelumnya, yaitu penelitian Maulana (2024), juga dilakukan untuk memahami lebih dalam struktur *backend* yang telah dikembangkan. Dari hasil analisis tersebut, disusun daftar fitur yang perlu ditambahkan atau diperbaiki pada modul *backend*, sebagaimana ditampilkan pada Tabel 8.

Tabel 8 Rancangan fitur sistem

No	Fitur	Deskripsi
1	Manajemen User	Penambahan fitur CRUD pengguna, termasuk perubahan dan manajemen *role* pengguna.
2	Verifikasi Email	Implementasi sistem verifikasi email untuk memastikan identitas pengguna.
3	Penilaian Berulang pada Lahan yang Sama	Memungkinkan lahan dinilai lebih dari satu kali tanpa menimpa data sebelumnya.
4	Perubahan Penyimpanan Dokumentasi Menggunakan MinIO	Perubahan penyimpanan file dokumentasi dari penyimpanan lokal ke MinIO.
5	Login Menggunakan Google	Implementasi *login* dengan akun Google guna mempermudah akses.

No	Fitur	Deskripsi
6	*Reverse Geocode* API Menggunakan Google Maps API	Menambahkan *endpoint* untuk mengkonversi koordinat menjadi alamat lengkap.
6	Pembuatan API data wilayah Indonesia	Membuat API baru untuk mendapatkan data wilayah Indonesia
8	Penyimpanan bentuk poligon	Mengubah penyimpanan koordinat plot dan lahan dari titik menjadi bentuk poligon untuk representasi area yang lebih akurat.

4.2 Pemodelan Modul *Backend*

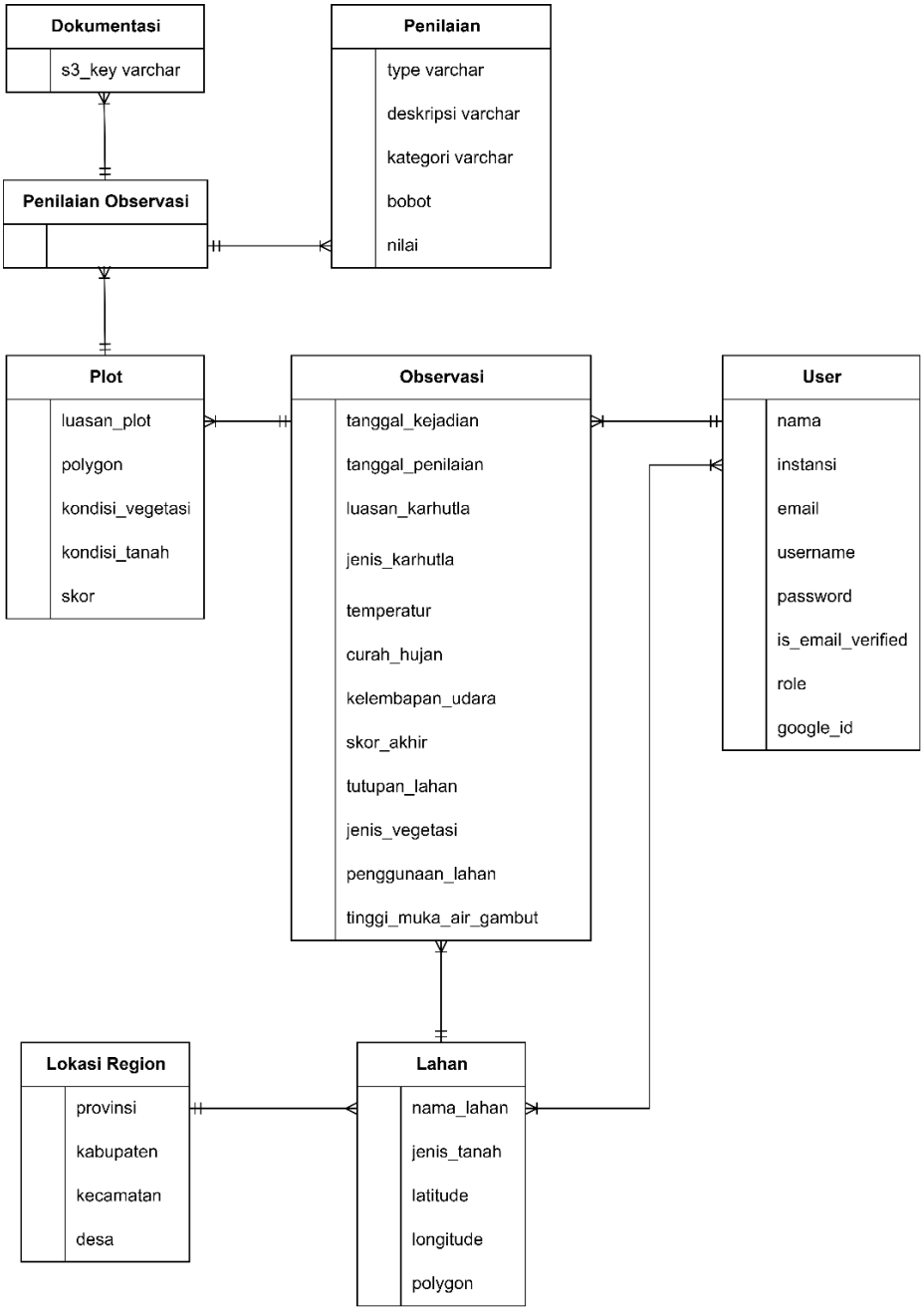
4.2.1 Penyusunan Rancangan Basis Data

Pemodelan modul *backend* diawali dengan perancangan desain konseptual yang bertujuan untuk memastikan struktur basis data dapat mendukung kebutuhan sistem secara optimal. Desain konseptual ini mengacu pada penelitian sebelumnya oleh Maulana (2024) dengan beberapa perubahan signifikan yang disesuaikan dengan fitur baru yang dikembangkan dalam penelitian ini. Beberapa perubahan utama mencakup penyesuaian relasi antar entitas, penambahan dan penghapusan kolom, serta optimalisasi penyimpanan data yang lebih fleksibel. Salah satu perubahan yang diterapkan adalah pada hubungan antara *User*, Lahan, dan Observasi. Dalam penelitian sebelumnya, hubungan antara *User* dan Lahan dibuat secara langsung, namun pada pengembangan kali ini, Observasi menjadi entitas perantara antara User dan Lahan. Dengan perubahan ini, setiap *User* kini dapat melakukan observasi pada banyak Lahan, dan setiap Lahan dapat memiliki banyak Observasi dari berbagai *User*, sementara Observasi hanya dimiliki oleh satu *User* yang melakukan penilaian. Model ini memungkinkan penyimpanan histori observasi tanpa menimpa data lama.

Selain perubahan dalam relasi entitas, dilakukan juga optimalisasi penyimpanan data spasial dengan menambahkan dukungan untuk format poligon pada tabel Lahan dan Plot. Sebelumnya, data lahan hanya direpresentasikan dengan titik koordinat, namun kini sistem mendukung penyimpanan geometri poligon, yang memungkinkan representasi area lahan dan plot secara lebih akurat. Dengan perubahan ini, setiap plot dalam suatu observasi kini memiliki poligon sendiri, sehingga sistem dapat menghitung luasan karhutla dengan lebih presisi. Hasil dari rancangan desain konseptual yang telah diperbarui dapat dilihat pada Gambar 5.

Setelah desain konseptual selesai, langkah berikutnya adalah mengonversi desain tersebut ke dalam skema relasional yang kemudian dikembangkan menjadi *Entity-Relationship Diagram* (ERD). ERD yang diperbarui dalam penelitian ini mengadopsi perubahan dari desain konseptual, dengan beberapa optimasi pada struktur basis data. Salah satu perubahan utama dalam ERD adalah penyederhanaan hubungan antara *User* dan Lahan. Sebelumnya, hubungan ini dibuat secara langsung, namun dalam rancangan baru, hubungan antara *User* dan Lahan dikelola melalui Observasi, di mana *User* memiliki banyak Observasi, dan setiap Lahan juga

memiliki banyak Observasi. Dengan model ini, hubungan tidak perlu lagi direpresentasikan secara langsung di dalam ERD, melainkan cukup dengan melihat hubungan antara User dan Observasi, serta Observasi dan Lahan.



Gambar 5 Desain konseptual

Perubahan lain terjadi pada tabel Lahan, di mana hubungan yang sebelumnya langsung dengan *User* kini dialihkan ke Observasi. Setiap Lahan kini dapat memiliki banyak Observasi, sehingga sistem dapat menyimpan histori penilaian tanpa menimpa data lama. Untuk meningkatkan representasi data, ditambahkan kolom nama_lahan sebagai identitas unik bagi setiap lahan yang dinilai, serta kolom polygon untuk menyimpan bentuk lahan dalam format geospasial yang lebih akurat. Selain itu, beberapa atribut yang sebelumnya berada di tabel Lahan, seperti

tutupan_lahan, jenis_vegetasi, luasan_karhutla, jenis_tanah, TMA_gambut, jenis_karhutla, temperatur, curah_hujan, dan kelembapan udara, kini dipindahkan ke tabel Observasi karena sifatnya yang dinamis dan berubah setiap kali observasi baru dilakukan. Dengan perubahan ini, sistem kini dapat menyimpan informasi kondisi lahan yang lebih fleksibel, tanpa mengubah data historis yang telah ada.

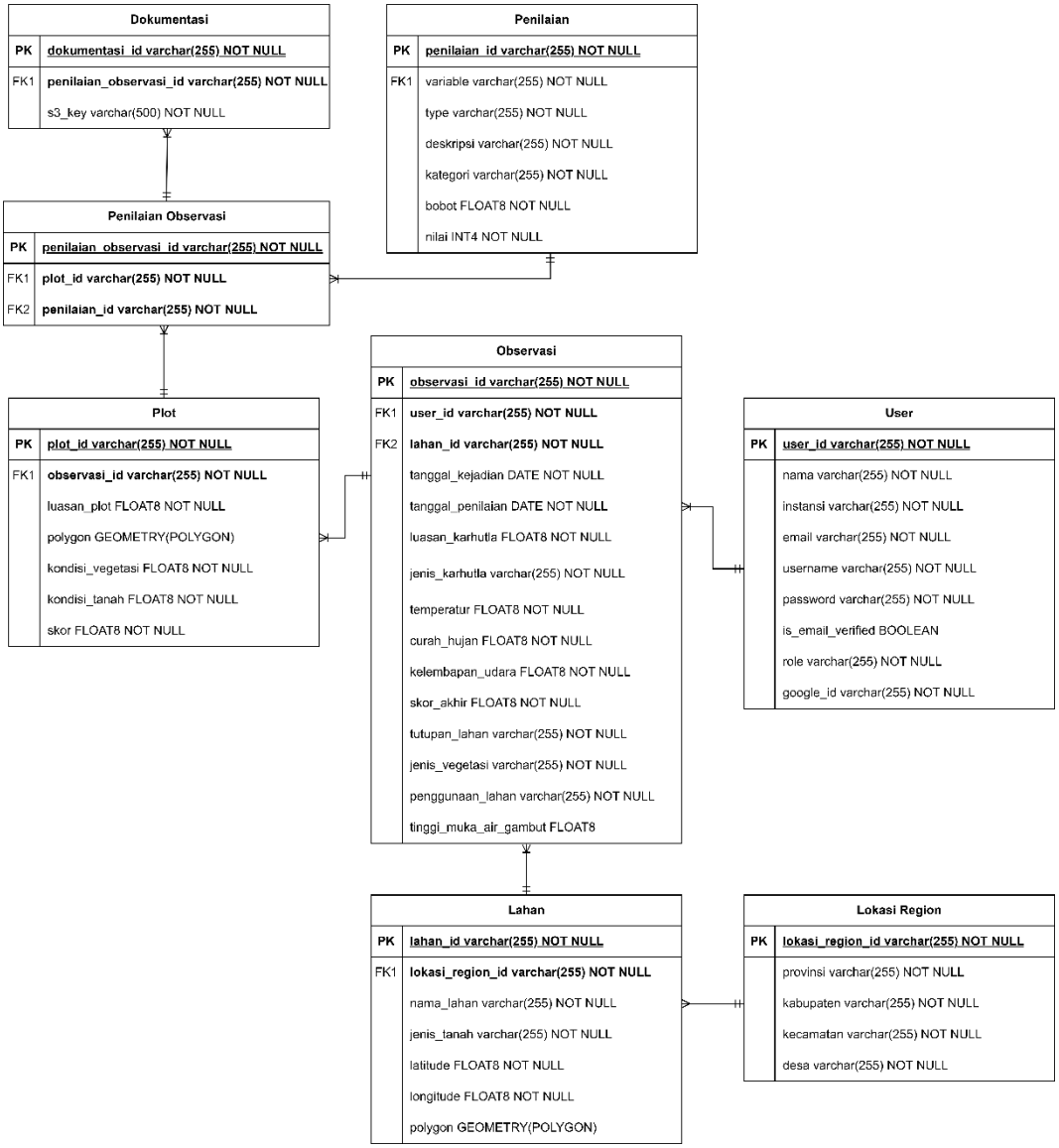
Selain perubahan dalam hubungan entitas, dilakukan juga beberapa perubahan pada struktur tabel untuk meningkatkan efisiensi pengolahan data dan mendukung fitur baru dalam SIPARTAN. Pada tabel *User*, ditambahkan kolom *is_email_verified* untuk mendukung verifikasi email saat registrasi, meningkatkan keamanan sistem. Selain itu, kolom *role* ditambahkan untuk menerapkan *Role-Based Access Control* (RBAC), sehingga hak akses pengguna dapat diatur berdasarkan perannya, seperti admin, penilai, atau *guest*. Ditambahkan pula kolom *google_id* untuk mendukung fitur *login* menggunakan akun Google, yang mempercepat proses autentikasi.

Pada tabel Observasi, perubahan besar dilakukan dengan menghubungkannya secara langsung ke *User* dalam relasi *one-to-many*, memungkinkan sistem untuk mencatat siapa yang melakukan observasi terhadap suatu lahan. Beberapa kolom yang sebelumnya berada di tabel Lahan kini dipindahkan ke tabel Observasi agar sistem dapat menyimpan informasi kondisi lahan yang lebih dinamis. Dengan perubahan ini, setiap observasi akan memiliki data lingkungan terkini tanpa mengubah rekaman observasi sebelumnya, memastikan bahwa informasi tetap akurat dan tidak tertimpa oleh data baru.

Perubahan juga dilakukan pada tabel Plot, di mana ditambahkan kolom *polygon* untuk menyimpan bentuk plot dalam format geospasial, memungkinkan representasi yang lebih akurat dari setiap plot observasi. Selain itu, tabel Hasil yang sebelumnya terpisah kini dihapus, dan data terkait kondisi_vegetasi, kondisi_tanah, serta skor penilaian dipindahkan ke dalam tabel Plot. Perubahan ini bertujuan untuk mengurangi redundansi data dan menyederhanakan pemrosesan hasil penilaian, memastikan bahwa setiap plot memiliki skor yang dihitung secara langsung tanpa perlu tabel tambahan.

Sementara itu, tabel Dokumentasi mengalami perubahan signifikan, di mana hubungan yang sebelumnya terhubung ke tabel Plot, kini dipindahkan ke tabel Observasi. Dengan perubahan ini, setiap penilaian observasi memiliki dokumentasi yang lebih spesifik, dibandingkan sebelumnya yang hanya terkait dengan satu plot. Selain itu, kolom nama dan *type* dihapus, digantikan oleh kolom *s3_key* untuk mendukung sistem penyimpanan berbasis MinIO, memastikan penyimpanan file menjadi lebih efisien dan *scalable* dibandingkan metode penyimpanan lokal sebelumnya. Hasil dari rancangan ERD yang telah diperbarui dapat dilihat pada Gambar 6.

Secara keseluruhan, perubahan dalam pemodelan basis data ini menjadikan SIPARTAN lebih efisien, fleksibel, dan sesuai dengan fitur-fitur baru yang dikembangkan. Relasi antar tabel telah diperbaiki untuk mendukung penilaian ulang lahan, penyimpanan histori observasi, serta autentikasi dan manajemen file yang lebih baik.



Gambar 6 Rancangan ERD

4.2.2 Penyusunan Struktur Sistem

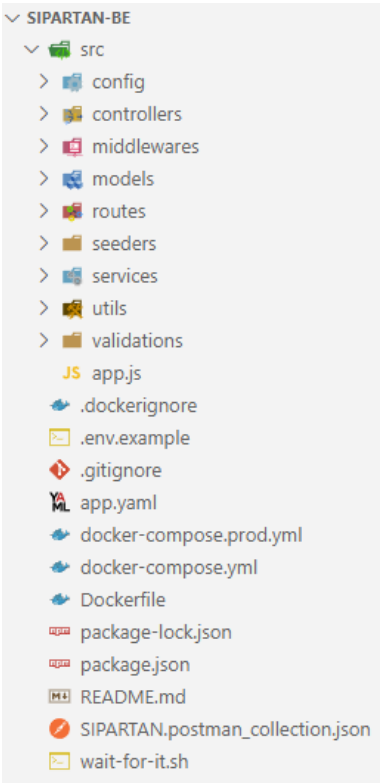
Selain melakukan pemodelan basis data, penyusunan struktur sistem juga dilakukan untuk meningkatkan keteraturan, keterbacaan kode, dan skalabilitas aplikasi. Dengan struktur yang lebih sistematis, pengembangan sistem SIPARTAN di masa depan dapat lebih mudah dilakukan, baik dalam hal pemeliharaan, *debugging*, maupun pengembangan fitur baru tanpa mengganggu bagian lain dari sistem. Struktur kode program yang diterapkan dalam pengembangan *backend* SIPARTAN dapat dilihat pada Gambar 7.

Dalam struktur terbaru ini, *folder* utama yang menyimpan kode *backend* berada di dalam *folder src*, yang terdiri dari beberapa *subfolder* dengan fungsi spesifik. Salah satu perubahan utama adalah penghapusan *folder image*, karena sistem penyimpanan dokumentasi kini telah dialihkan ke MinIO, sehingga tidak lagi memerlukan penyimpanan gambar secara lokal. Selain itu, ditambahkan *folder seeders* yang berisi data awal untuk variabel penilaian, sehingga ketika sistem

pertama kali dibuat atau dideploy, data penilaian sudah tersedia tanpa perlu input manual. *Folder* ini juga mencakup *seeders* akun pengguna yang menyediakan contoh akun admin, penilai, dan guest, yang masing-masing dibuat untuk mempermudah pengujian serta memastikan bahwa sistem dapat berjalan dengan baik dengan berbagai peran pengguna.

Perubahan lain yang signifikan adalah penambahan folder *validations*, yang kini berfungsi untuk menyimpan seluruh kode validasi yang sebelumnya tersebar di dalam *controller*. Dengan adanya *folder* ini, kode validasi lebih terorganisir, modular, dan mudah digunakan kembali dalam berbagai bagian sistem tanpa perlu duplikasi. Di luar *folder src*, terdapat beberapa *file* baru yang ditambahkan untuk mendukung pengelolaan sistem dalam berbagai lingkungan. Salah satu yang paling penting adalah docker-compose.prod.yml, yang berfungsi untuk men-*deploy* aplikasi dalam lingkungan *server* menggunakan Docker. Sebelumnya, hanya tersedia *file* docker-compose.yml yang digunakan untuk pengembangan lokal. Dengan adanya pemisahan antara konfigurasi *development* dan *production*, sistem dapat lebih fleksibel dan siap untuk *deployment* ke lingkungan yang berbeda.

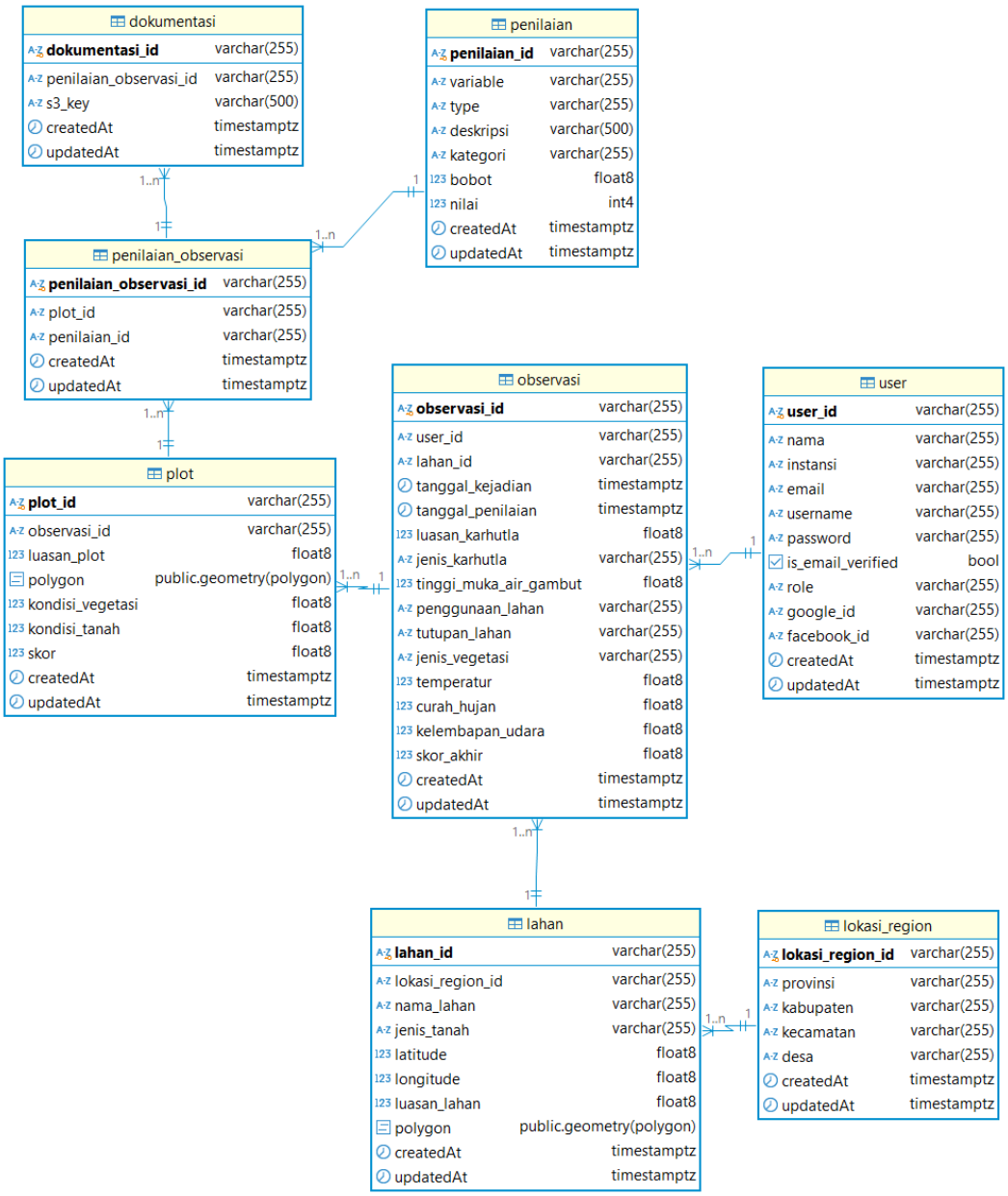
Selain itu, *file* README.md kini diperbarui dengan informasi singkat mengenai cara menjalankan proyek baik di lingkungan lokal maupun *server*, sehingga mempermudah pengembang dalam memahami langkah-langkah setup sistem. *File* SIPARTAN.postman_collection.json juga disertakan sebagai dokumentasi API yang dapat langsung digunakan untuk menguji *endpoint* melalui Postman. Untuk mendukung pengelolaan *container* secara lebih baik dalam lingkungan produksi, ditambahkan juga *file* wait-for-it.sh, yaitu *script* pembantu yang memastikan bahwa layanan Node.js hanya akan berjalan setelah *database* PostgreSQL berhasil diinisialisasi.



Gambar 7 Struktur kode SIPARTAN

4.3 Pengembangan Modul *Backend*

Setelah rancangan basis data dan struktur sistem selesai, tahap selanjutnya adalah pengembangan modul *backend*, yang dilakukan dengan menyesuaikan kode model menggunakan ORM Sequelize. Perubahan ini mengacu pada desain *Entity-Relationship Diagram* (ERD) yang telah diperbarui untuk mendukung fitur baru, seperti penilaian ulang lahan, penyimpanan data berbasis poligon, serta optimalisasi sistem autentikasi dan otorisasi pengguna. Adapun diagram basis data yang telah diubah sesuai dengan rancangan ERD dapat dilihat pada Gambar 8.



Gambar 8 Diagram struktur basis data SIPARTAN

Selain pembaruan pada struktur data, sistem API juga dioptimalkan untuk meningkatkan modularitas dan kemudahan pengelolaan data. Saat ini, API SIPARTAN dikelompokkan menjadi lima bagian utama, yaitu *auth*, *user*, info,

lahan, dan observasi, untuk memastikan pengelolaan fitur dalam setiap kelompok API lebih terstruktur.

Dalam pengembangan ini, total *endpoint* dalam sistem mencapai 38, yang mencakup beberapa *endpoint* baru, perubahan signifikan pada beberapa *endpoint* lama, serta modifikasi minor dari pengembangan sebelumnya. Beberapa perubahan mencakup optimasi pengelolaan data dan restrukturisasi otorisasi akses, di mana setiap akses dikontrol dengan *minimum role access*, yang terdiri dari *guest*, penilai, dan admin, dengan admin memiliki hak akses penuh terhadap sistem. Beberapa endpoint dapat diakses tanpa autentikasi, sementara sebagian besar lainnya memerlukan peran tertentu untuk menjaga keamanan data, memastikan hanya pengguna yang berwenang yang dapat mengakses atau memodifikasi informasi tertentu.

Pengembangan modul *backend* SIPARTAN menggunakan Sequelize untuk modifikasi *database*, Nodemailer untuk pengiriman email, dan MinIO untuk penyimpanan gambar dokumentasi. Ketiga tools ini meningkatkan efisiensi, skalabilitas, dan keamanan sistem, memastikan pengolahan data, autentikasi pengguna, serta manajemen *file* dapat berjalan lebih optimal.

Sequelize adalah *Object-Relational Mapping* (ORM) untuk Node.js yang memungkinkan pengelolaan database tanpa perlu menulis *query* SQL secara langsung. Dengan Sequelize, penulis dapat mengubah model data dalam database dengan memodifikasi kode model. Dalam penelitian ini, Sequelize juga digunakan untuk menangani berbagai operasi *database*, termasuk penerapan *pagination*, *search*, dan *filter* menggunakan fungsi *findAndCountAll*, yang mempermudah pengambilan data secara terstruktur dan efisien, terutama saat menangani jumlah data yang besar. Nodemailer adalah *library* untuk pengiriman email dalam sistem SIPARTAN, memungkinkan sistem untuk mengirimkan email verifikasi akun dan *forgot password* kepada pengguna. Dalam implementasinya, Nodemailer dikonfigurasi menggunakan SMTP Google, memastikan keamanan dalam proses autentikasi email.

MinIO digunakan sebagai *object storage* untuk menyimpan gambar dokumentasi lahan pasca kebakaran hutan, menggantikan metode penyimpanan sebelumnya yang hanya mengandalkan penyimpanan langsung dalam sistem. Dengan MinIO, penyimpanan gambar menjadi lebih *scalable*, terstruktur, dan terorganisir, karena setiap *file* disimpan dalam direktori berbasis *s3_key*, yang memungkinkan pengelolaan lebih baik berdasarkan tahun, wilayah, dan kategori dokumentasi. Selain itu, MinIO juga menyediakan UI manajemen *file*, memungkinkan admin untuk mengakses, menghapus, dan mengelola dokumentasi dengan lebih mudah.

4.3.1 Kelompok API *Auth*

Kelompok API *Auth* bertanggung jawab atas proses autentikasi dan pendaftaran pengguna dalam sistem SIPARTAN. Kelompok API Auth berfungsi untuk memungkinkan pengguna membuat akun, melakukan *login*, serta memverifikasi email sebelum mendapatkan akses ke fitur-fitur lainnya dalam sistem. Kelompok API ini mendukung berbagai mekanisme autentikasi, termasuk registrasi akun baru, *login* dengan kredensial email dan *password*, pengaturan ulang kata sandi, serta *login* menggunakan akun Google. Selain itu, kelompok API ini juga menangani proses verifikasi email, yang menjadi syarat utama sebelum

pengguna dapat diverifikasi perannya oleh admin dan mendapatkan akses ke fitur lebih lanjut. Adapun rincian API yang terdapat pada kelompok API *Auth* dapat dilihat pada Tabel 9.

Tabel 9 Daftar API pada kelompok API *auth*

Endpoint	**Metode**	***Minimum Role***	**Deskripsi**
/auth/register	*POST*	Tidak ada	Mendaftarkan pengguna baru ke dalam sistem.
/auth/login	*POST*	Tidak ada	Melakukan autentikasi pengguna dan mendapatkan *token* akses.
/auth/forgot-password	*POST*	Tidak ada	Mengirim email untuk permintaan *reset* password.
/auth/reset-password	*POST*	Tidak ada	Mengatur ulang *password* menggunakan *token* yang dikirimkan.
/auth/send-verification-email	*POST*	*Guest*	Mengirim email untuk verifikasi akun pengguna.
/auth/verify-email	*POST*	Tidak ada	Memverifikasi email pengguna melalui *token* yang dikirimkan.
/auth/google	*GET*	Tidak ada	Menginisiasi proses *login* menggunakan akun Google.
/auth/google/callback	*GET*	Tidak ada	Menangani *callback* dari Google dan memberikan *token*.

Endpoint /auth/register digunakan untuk mendaftarkan pengguna baru ke dalam sistem. Proses registrasi mencakup validasi data input serta hashing *password* menggunakan algoritma *bcrypt* untuk memastikan keamanan akun pengguna. Kata sandi yang dimasukkan harus memenuhi standar keamanan yang ketat, yaitu memiliki minimal 8 karakter, mengandung huruf besar, huruf kecil, angka, serta *special* karakter, dan tidak boleh mengandung spasi. Pada tahap awal registrasi, *role* pengguna secara default akan diatur sebagai *guest*. Jika pengguna ingin mengubah peran menjadi penilai, mereka harus memverifikasi email terlebih dahulu, kemudian menunggu admin untuk mengubah *role* pengguna menjadi penilai. Setelah registrasi berhasil, pengguna tetap harus melakukan *login* untuk mendapatkan *token* autentikasi, yang diperlukan untuk mengakses fitur sistem yang memerlukan otorisasi. Adapun data yang harus diisi oleh pengguna dalam *request body* dapat dilihat pada Gambar 9.

Parameter	Type	Required
nama	string	Yes
instansi	string	Yes
email	string	Yes
username	string	Yes
password	string	Yes

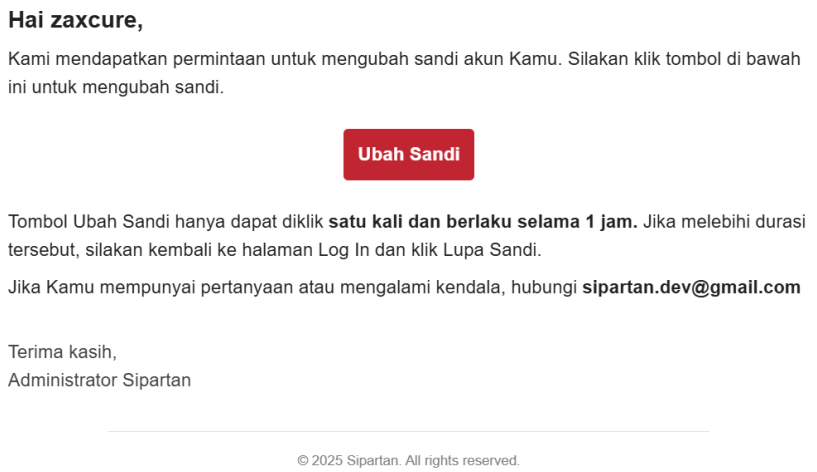
Gambar 9 *Request body* API /auth/register

Endpoint /auth/login digunakan untuk autentikasi pengguna yang telah terdaftar. Saat pengguna memasukkan kredensial mereka, sistem akan mencocokkan *password* yang dikirimkan dengan *password* yang telah di-*hash* di dalam basis data menggunakan bcrypt. Jika *login* berhasil, pengguna akan diberikan *token* JWT, yang memiliki masa berlaku satu hari. *Token* ini digunakan untuk mengakses *endpoint* lain yang memerlukan autentikasi. Adapun *response body* yang dikembalikan oleh *endpoint* ini dapat dilihat pada Gambar 10.


```
{
    "status": 200,
    "message": "Login successful",
    "data": {
        "user_id": "72ed3c33-8c6b-41aa-86f9-3f9bc0f39377",
        "nama": "akuadalahadmin",
        "instansi": null,
        "email": "akuadalahadmin@gmail.com",
        "username": "akuadalahadmin",
        "role": "admin",
        "is_email_verified": false
    },
    "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9..."
}
```


Gambar 10 *Response* dari *endpoint POST* /auth/login

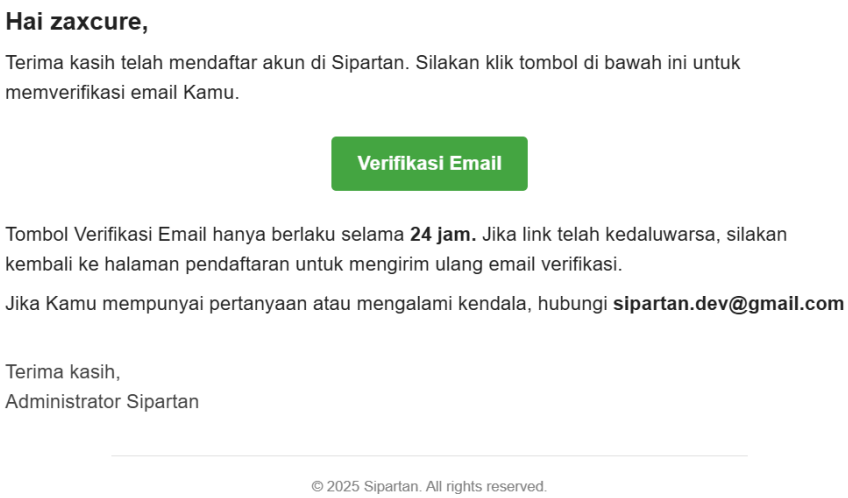
Endpoint /auth/forgot-password digunakan untuk mengirimkan email kepada pengguna yang ingin mereset kata sandinya. Sistem ini menggunakan SMTP Google sebagai layanan pengiriman email, dengan bantuan Nodemailer sebagai *library* yang menangani proses pengiriman. Saat pengguna mengajukan permintaan *reset password*, sistem akan menghasilkan token yang hanya berlaku selama satu jam. *Token* ini kemudian dikirimkan melalui email ke alamat pengguna yang terdaftar. Adapun contoh email yang dikirimkan oleh sistem dapat dilihat pada Gambar 11.



Gambar 11 Email *reset password*

Endpoint /auth/reset-password digunakan untuk mengatur ulang *password* pengguna yang telah mengajukan permintaan reset melalui /auth/forgot-password. Proses ini memerlukan *token reset password* yang dikirimkan melalui email sebelumnya. Sistem akan memvalidasi *token* tersebut sebelum mengizinkan pengguna mengatur ulang kata sandinya. Jika *token* valid, sistem akan memperbarui *password* pengguna dengan *password* baru yang telah di-hash menggunakan algoritma *bcrypt*.

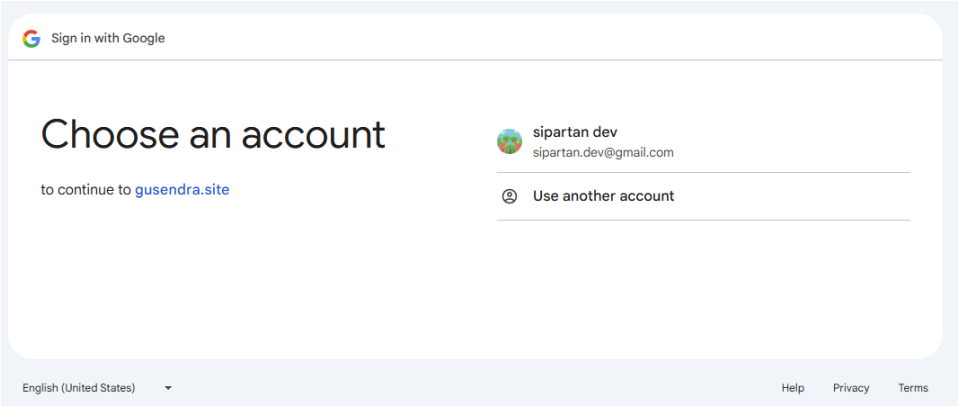
Endpoint /auth/send-verification-email digunakan untuk mengirimkan email verifikasi akun kepada pengguna yang telah melakukan registrasi. *Endpoint* ini memerlukan autentikasi dengan *token*, sehingga hanya pengguna yang telah *login* yang dapat mengaksesnya. Pengiriman email verifikasi menggunakan metode yang sama dengan mekanisme *forgot password*, di mana sistem akan mengirimkan token unik ke alamat email pengguna. Setelah menerima email verifikasi, pengguna harus mengirimkan kembali *token* tersebut ke *endpoint* /auth/verify-email dengan cara menekan tombol pada email untuk memverifikasi email. Proses verifikasi ini diperlukan agar akun pengguna dapat ditingkatkan perannya menjadi penilai oleh admin, sehingga pengguna dapat mengakses fitur penilaian. Adapun contoh email yang dikirimkan untuk verifikasi dapat dilihat pada Gambar 12.



Gambar 12 Email untuk verifikasi email.

Endpoint /auth/google menginisiasi proses login menggunakan akun Google. Jika pengguna yang mencoba *login* menggunakan Google belum memiliki akun di sistem SIPARTAN, maka sistem akan secara otomatis mendaftarkan akun baru berdasarkan informasi yang diberikan oleh Google. Adapun tampilan halaman saat pengguna diarahkan untuk *login* melalui Google dapat dilihat pada Gambar 13.

Endpoint /auth/google/callback menangani *callback* yang dikirimkan oleh Google setelah proses autentikasi berhasil. Setelah pengguna berhasil *login* melalui Google, sistem akan mengembalikan *token* JWT yang dapat digunakan untuk mengakses *endpoint* lainnya. *Response body* yang diterima dari *endpoint* ini memiliki struktur yang serupa dengan *response body* yang diterima dari *endpoint* /auth/login.



Gambar 13 Tampilan *login* menggunakan akun Google

4.3.2 Kelompok API *User*

Kelompok API *User* digunakan untuk mengelola data pengguna dalam sistem. Semua *endpoint* dalam kelompok ini memerlukan *token* autentikasi, sehingga hanya pengguna yang telah *login* yang dapat mengakses fitur-fitur yang tersedia. *Endpoint* dalam kelompok ini mencakup manajemen pengguna, pembaruan informasi akun, penghapusan akun, serta pengelolaan peran pengguna oleh admin. Adapun rincian API yang terdapat pada kelompok API *User* dapat dilihat pada Tabel 10.

Tabel 10 Daftar API pada kelompok API *user*

Endpoint	**Metode**	***Minimum Role***	**Deskripsi**
/user	*GET*	Admin	Mendapatkan daftar seluruh pengguna yang terdaftar di sistem.
/user/:user_id	*GET*	*Guest*	Mengambil informasi detail dari seorang pengguna tertentu.
/user/:user_id	*PATCH*	*Guest*	Melakukan perubahan data pada pengguna tertentu.
/user/:user_id	*DELETE*	*Guest*	Menghapus data pengguna tertentu.

Endpoint	Metode	*Minimum Role*	Deskripsi
/user/:user_id/verify-role	*PATCH*	Admin	Melakukan perubahan *role* pada pengguna tertentu
/user/profile	*GET*	*Guest*	Mengambil data pengguna sendiri yang sedang *login*

Endpoint GET /user digunakan untuk mengambil daftar seluruh pengguna yang terdaftar dalam sistem. Akses ke *endpoint* ini hanya diperbolehkan untuk admin, memastikan bahwa hanya pengguna dengan hak akses tertinggi yang dapat melihat seluruh data pengguna dalam sistem. Selain mengambil daftar pengguna, *endpoint* ini juga mendukung berbagai *query parameter* yang memungkinkan pencarian dan penyaringan data. Selain itu, tersedia parameter *pagination* yang memungkinkan admin melihat daftar pengguna dalam jumlah tertentu per halaman, serta parameter *sortBy* dan *order* yang memungkinkan hasil diurutkan berdasarkan kolom tertentu dalam urutan *ascending* (ASC) atau *descending* (DESC). Adapun *query parameter* yang dapat digunakan dalam *endpoint* ini dapat dilihat pada Gambar 14.

Parameter	Type	Required
nama	string	No
role	string	No
email	string	No
page	number	No
limit	number	No
sortBy	string	No
order	string	No

Gambar 14 *Query parameter endpoint GET* /user

Sistem sistem *pagination*, *search*, dan *filter* pada *endpoint* ini diterapkan dengan menggunakan bantuan ORM Sequelize. *Pagination* dalam sistem ini bekerja dengan cara menghitung jumlah total pengguna yang sesuai dengan *query parameter* yang diberikan, lalu membagi hasil pencarian ke dalam beberapa halaman dengan jumlah data tertentu per halaman. Fungsi *pagination* menggunakan *offset* dan *limit*, di mana *offset* digunakan untuk melewati sejumlah data tertentu berdasarkan halaman yang dipilih, sedangkan *limit* menentukan jumlah maksimum data yang akan dikembalikan per halaman. Dengan adanya fitur ini, sistem dapat menangani daftar pengguna dalam jumlah besar tanpa mengurangi performa *query* ke database. Adapun fungsi *pagination* yang diterapkan dalam sistem dapat dilihat pada Gambar 15.

```
async function paginate(model, options) {
    const {
        where = {},
        attributes = null,
        include = null,
        order = [['createdAt', 'DESC']],
        page = 1,
        limit,
    } = options;

    const offset = limit ? (page - 1) * limit : 0;

    const { count, rows } = await model.findAndCountAll({
        where,
        attributes,
        include,
        order,
        limit: limit || null,
        offset,
    });

    const totalPages = Math.ceil(count / limit) || 1;

    return {
        results: rows,
        page,
        limit,
        totalPages,
        totalResults: count,
    };
}
```

Gambar 15 Fungsi *paginate*

Setelah *query* dijalankan, sistem akan mengembalikan daftar pengguna sesuai dengan *parameter* yang diberikan. *Response* dari *endpoint GET* /user mencakup data pengguna yang sesuai dengan *filter*, total jumlah pengguna, jumlah halaman, serta informasi *pagination* lainnya. Adapun contoh *response* yang dikembalikan oleh *endpoint* ini dapat dilihat pada Gambar 16.


```
{
    "status": 200,
    "message": "Users retrieved successfully",
    "data": {
        "results": [
            {
                "user_id": "abbdaefe-f1e4-4534-ad8b-ceb709db2e6c",
                "nama": "admin",
                "instansi": null,
                "email": "admin@gmail.com",
                "username": "admin",
                "role": "admin",
                "createdAt": "2024-11-16T11:12:09.980Z",
                "is_email_verified": false
            }
        ],
        "page": 1,
        "limit": 10,
        "totalPages": 1,
        "totalResults": 2
    }
}
```


Gambar 16 *Response body endpoint GET* /user

Endpoint GET /user/:user_id digunakan untuk mengambil informasi detail dari seorang pengguna tertentu. *Endpoint* ini dapat diakses oleh *guest* dan peran di atasnya, namun pengguna hanya dapat mengambil data akun mereka sendiri, kecuali jika mereka adalah admin. *Response* yang dikembalikan oleh *endpoint* ini mencakup informasi lengkap pengguna, seperti nama, email, *role*, serta status verifikasi email. Contoh *response* dari *endpoint* ini dapat dilihat pada Gambar 17.


```
{
    "status": 200,
    "message": "User retrieved successfully",
    "data": {
        "user_id": "6ddc80b3-2a88-4dd2-82f9-a677b0bb3cd8",
        "nama": "Testing",
        "instansi": "IPB",
        "email": "testing@gmail.com",
        "username": "testing",
        "role": "guest",
        "is_email_verified": false,
        "createdAt": "2025-02-21T07:44:47.905Z",
        "updatedAt": "2025-02-21T07:44:47.905Z"
    }
}
```


Gambar 17 *Response body endpoint GET* /user/:user_id

Endpoint PATCH /user/:user_id digunakan untuk memperbarui informasi akun pengguna. Pengguna hanya dapat memperbarui data diri mereka sendiri, sedangkan admin memiliki akses untuk mengubah data pengguna lain. Dalam penerapannya, metode *PATCH* dipilih karena perubahan yang dilakukan hanya mencakup sebagian data dari tabel *user*, bukan melakukan pembaruan keseluruhan seperti yang terjadi pada metode *PUT*. Pengguna dapat mengubah informasi akun mereka, namun jika mereka mengganti email, status verifikasi email akan diatur ulang menjadi tidak terverifikasi, sehingga admin tidak dapat mengubah *role* pengguna tersebut hingga email berhasil diverifikasi kembali. Hal ini diterapkan untuk memastikan bahwa hanya akun dengan email yang telah diverifikasi yang dapat memperoleh hak akses lebih tinggi dalam sistem. Adapun *request body* yang diperbolehkan dalam *endpoint* ini dapat dilihat pada Gambar 18.

Parameter	Type	Required
nama	string	No
instansi	string	No
email	string	No
username	string	No

Gambar 18 *Request body endpoint PATCH* /user/:user_id

Endpoint DELETE /user/:user_id digunakan untuk menghapus akun pengguna tertentu. Pengguna dengan *role guest* dapat menghapus akun mereka

sendiri, sedangkan admin memiliki hak untuk menghapus akun pengguna lain jika diperlukan. Sistem akan melakukan validasi kepemilikan akun, memastikan bahwa pengguna tidak dapat menghapus akun lain kecuali jika mereka memiliki hak admin. *Endpoint PATCH* /user/:user_id/verify-role digunakan untuk mengubah peran pengguna dalam sistem dari *guest* ke penilai atau sebaliknya. Hanya admin yang memiliki izin untuk mengakses *endpoint* ini. Pengguna tidak dapat ditingkatkan menjadi penilai kecuali jika email mereka telah diverifikasi.

Endpoint GET /user/profile digunakan untuk mendapatkan informasi akun pengguna yang sedang *login*. *Endpoint* ini memanfaatkan *token* autentikasi yang dikirimkan dalam *header* permintaan untuk mengidentifikasi pengguna yang sedang masuk. *Response* yang dikembalikan oleh *endpoint* ini serupa dengan *response* dari *GET* /user/:user_id, tetapi hanya mengembalikan informasi akun pengguna yang sedang *login*, bukan pengguna lain. Dengan adanya *endpoint* ini, pengguna dapat melihat data mereka sendiri tanpa perlu mengetahui user_id pengguna, sehingga mempermudah akses ke informasi akun pengguna.

4.3.3 Kelompok API Lahan

Kelompok API Lahan digunakan untuk mengelola data lahan dalam sistem SIPARTAN. Pada penelitian sebelumnya, kelompok ini dinamakan lahan-karhutla, namun dalam pengembangan ini namanya diubah menjadi lahan saja karena kelompok API ini sekarang hanya berfokus pada manajemen lahan, tanpa menyangkut observasi dari suatu lahan tersebut. Kelompok API ini memungkinkan pengguna untuk menambahkan, mengambil, memperbarui, dan menghapus data lahan. Terdapat 5 *endpoint* dalam kelompok ini, di mana dua *endpoint* (*GET* /lahan dan *GET* /lahan/:lahan_id) dapat diakses tanpa autentikasi, sementara tiga *endpoint* lainnya hanya dapat diakses oleh pengguna dengan *minimum role* penilai. Adapun rincian API yang terdapat pada kelompok API Lahan dapat dilihat pada Tabel 11.

Tabel 11 Daftar API pada kelompok API lahan

Endpoint	**Metode**	***Minimum Role***	**Deskripsi**
/lahan	*POST*	Penilai	Menambahkan data lahan baru ke dalam sistem.
/lahan	*GET*	Tidak ada	Mendapatkan seluruh data lahan yang terdaftar dalam sistem.
/lahan/:lahan_id	*GET*	Tidak ada	Mengambil data lahan tertentu.
/lahan/:lahan_id	*PATCH*	Penilai	Melakukan perubahan pada data lahan tertentu.
/lahan/:lahan_id	*DELETE*	Penilai	Menghapus data lahan tertentu.

Endpoint POST /lahan digunakan untuk membuat lahan baru yang nantinya akan dinilai tingkat keparahannya pasca kebakaran. *Endpoint* ini menerima beberapa *field*, di antaranya nama_lahan yang berfungsi sebagai nama lahan, serta lokasi_region yang berisi informasi mengenai lokasi lahan, termasuk provinsi, kabupaten, kecamatan, dan desa. Selain itu, terdapat *field coordinates*, yang

digunakan untuk membentuk poligon lahan sekaligus untuk menghitung luas lahan. Sebagian besar *field* pada *endpoint* ini bersifat wajib diinput, kecuali *coordinates*, karena pada modul *mobile* terdapat kemungkinan bahwa pengguna membuat data lahan terlebih dahulu tanpa langsung membuat poligonnya. Dalam kasus ini, poligon dapat bernilai *null*, namun dapat diperbarui di kemudian hari melalui *PATCH* /lahan/:lahan_id. Adapun *request body* yang harus di-input oleh pengguna dapat dilihat pada Gambar 19.

Parameter	Type	Required
lokasi_region.provinsi	string	Yes
lokasi_region.kabupaten	string	Yes
lokasi_region.kecamatan	string	Yes
lokasi_region.desa	string	Yes
lahan.nama_lahan	string	Yes
lahan.jenis_tanah	string	Yes
lahan.latitude	number	Yes
lahan.longitude	number	Yes
lahan.coordinates	array	No

Gambar 19 *Request body endpoint POST* /lahan

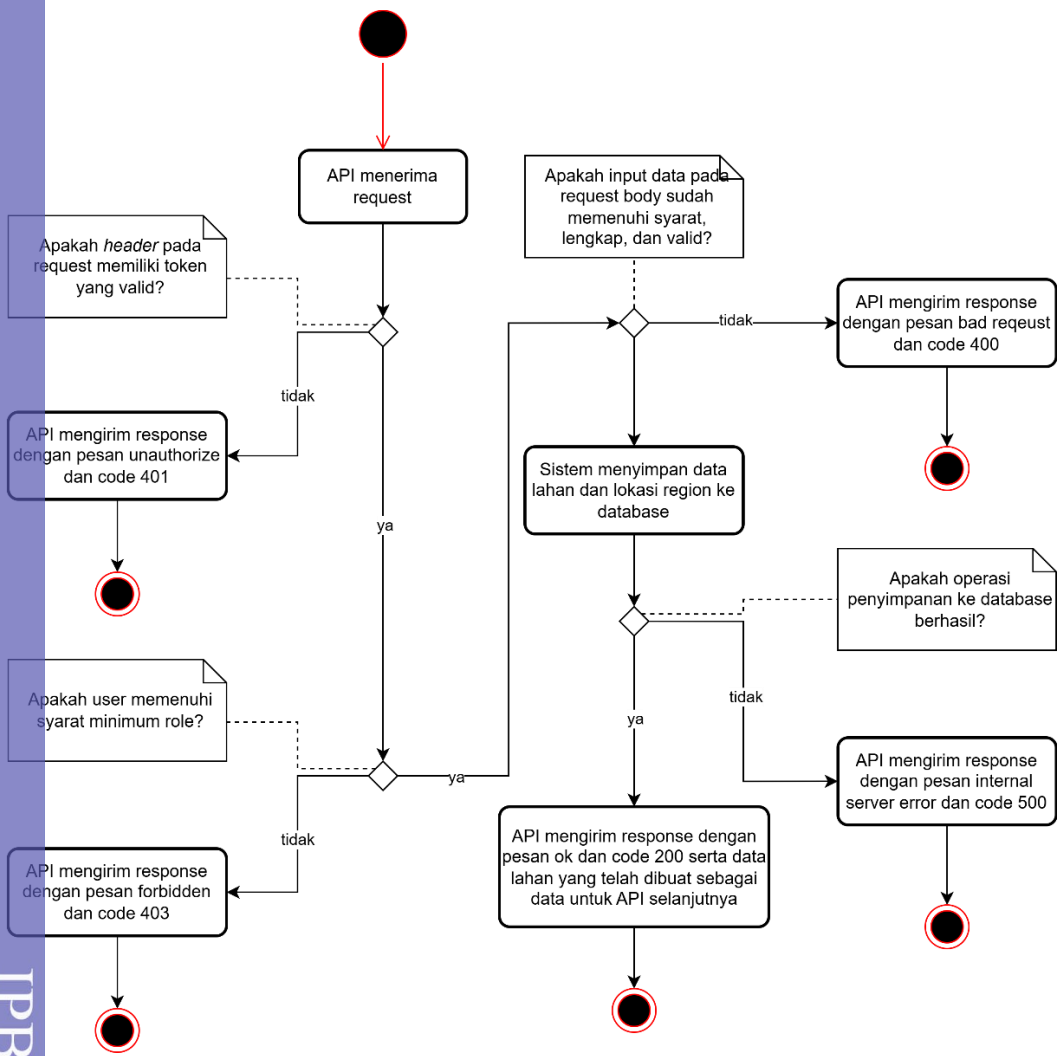
Ketika *endpoint* ini menerima permintaan untuk membuat lahan baru, sistem akan melalui beberapa tahap validasi dan pemrosesan data. Proses ini mencakup validasi *token* autentikasi, pengecekan *role* pengguna, serta validasi kelengkapan data yang dikirimkan. Jika data tidak valid, API akan mengembalikan *response error* 400 *Bad Request*. Jika valid, data lahan akan disimpan ke dalam *database*, dan sistem akan mengevaluasi keberhasilan operasi penyimpanan. Jika terjadi kegagalan, API akan mengembalikan *error* 500 *Internal Server Error*, namun jika berhasil, API akan mengembalikan *response* 200 OK beserta data lahan yang telah dibuat. Adapun alur pembuatan data lahan dapat dilihat pada *activity diagram* pada Gambar 20.

Pada proses penyimpanan data lahan, sistem melakukan pembentukan poligon serta perhitungan luas_lahan menggunakan fungsi dari PostGIS. Proses ini diawali dengan pembuatan GeoJSON Polygon, yang merupakan representasi geometri dari lahan berdasarkan koordinat yang diberikan. Setelah GeoJSON Polygon terbentuk, sistem menggunakan fungsi ST_GeomFromGeoJSON('${polygon}') dari PostGIS untuk mengonversi data GeoJSON menjadi tipe geometri yang dapat disimpan dalam DBMS PostgreSQL. ST_GeomFromGeoJSON adalah fungsi yang digunakan dalam PostGIS untuk mengubah string GeoJSON menjadi objek geometri yang dikenali oleh sistem basis data spasial.

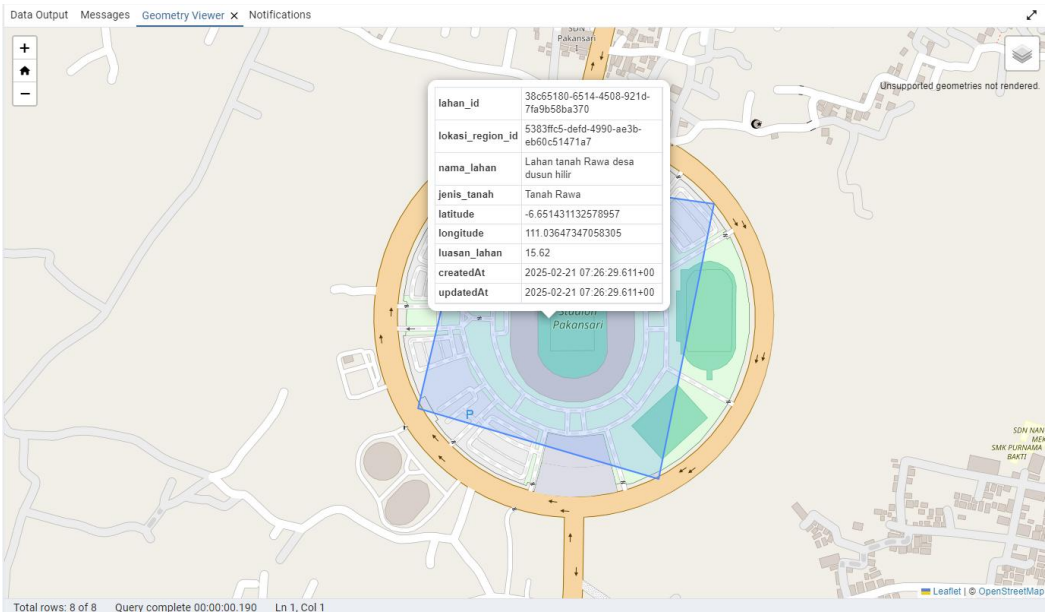
Selain pembuatan poligon, sistem juga menghitung luas area lahan secara otomatis menggunakan *query* “SELECT ST_Area(ST_GeomFromGeoJSON(:geoJson)::geography) / 10000 AS

area_in_hectares;" fungsi ST_Area digunakan untuk menghitung luas suatu geometri. Namun, karena satuan yang dihasilkan adalah meter persegi, maka hasilnya dibagi dengan 10.000 agar diperoleh nilai dalam satuan hektar. ST_Area bekerja dengan tipe data spasial yang memiliki sistem koordinat yang jelas. Meskipun ST_GeomFromGeoJSON sudah mengubah GeoJSON menjadi tipe geometri, hasilnya secara default masih dalam bentuk *geometry*, yang berarti satuannya masih dalam derajat, bukan dalam meter atau hektar. Oleh karena itu, diperlukan konversi ke tipe *geography* menggunakan casting ::geography agar perhitungan luas dapat dilakukan dalam satuan meter yang lebih akurat.

Secara default, tipe data *geometry* dalam PostGIS menggunakan sistem koordinat kartesian (planar system), yang tidak mempertimbangkan kelengkungan bumi. Hal ini menyebabkan perhitungan luas dalam ST_Area tidak akurat jika menggunakan unit derajat, terutama untuk wilayah yang luas. Dengan mengubah geometri menjadi geography menggunakan casting ::geography, PostGIS akan melakukan perhitungan luas berdasarkan model bumi elipsoidal, yang lebih sesuai untuk pengukuran luas area di dunia nyata. Contoh bentuk *geometry* poligon yang telah dibuat dapat dilihat pada Gambar 21.



Gambar 20 *Activity diagram endpoint POST* /lahan



Gambar 21 Tampilan poligon pada *software* pgAdmin

Endpoint GET /lahan digunakan untuk mengambil seluruh daftar lahan yang tersedia dalam sistem. *Endpoint* ini telah dilengkapi dengan fitur *pagination*, *search*, dan *filter*, yang memungkinkan pengguna untuk melakukan pencarian serta penyaringan data secara lebih efisien. Dalam proses pengambilan data, *endpoint* ini menggunakan mekanisme yang sama seperti pada *endpoint GET* /user. Dengan adanya fitur ini, pengguna dapat dengan mudah menemukan lahan berdasarkan berbagai parameter yang ada.

Parameter *skor_min* dan *skor_max* digunakan untuk menyaring lahan berdasarkan skor_akhir dari observasi terbaru pada lahan tersebut. Demikian pula, hasil_penilaian juga merujuk pada hasil observasi terbaru dari lahan yang bersangkutan. Selain itu, *endpoint* ini juga menyediakan *filter* berdasarkan tanggal kejadian dan tanggal penilaian. Parameter tanggal_kejadian_start dan tanggal_kejadian_end digunakan untuk menyaring lahan berdasarkan rentang waktu kejadian kebakaran yang tercatat dalam observasi terakhir. Sementara itu, tanggal_penilaian_start dan tanggal_penilaian_end memungkinkan penyaringan berdasarkan rentang waktu penilaian yang dilakukan dalam observasi terbaru. Jika sebuah lahan belum memiliki observasi, maka parameter ini tidak akan mempengaruhi hasil pencarian terhadap lahan tersebut. Adapun rincian *query parameter* yang dapat digunakan pada *endpoint* ini dapat dilihat pada Gambar 22.

Parameter	Type	Required
page	number	No
limit	number	No
sortBy	string	No

Gambar 22 *Query parameter endpoint GET* /lahan

order	string	No
lahan_id	string	No
nama_lahan	string	No
jenis_tanah	string	No
provinsi	string	No
kabupaten	string	No
kecamatan	string	No
desa	string	No
hasil_penilaian	string	No
skor_min	number	No
skor_max	number	No
tanggal_kejadian_start	date	No
tanggal_kejadian_end	date	No
tanggal_penilaian_start	date	No
tanggal_penilaian_end	date	No

Gambar 22 *Query parameter endpoint GET* /lahan (lanjutan)

Semua *query parameter* yang ada bersifat opsional, dan *response* dari *endpoint* ini mencakup daftar lahan sesuai dengan filter yang ada jika diberikan. Pada bagian poligon ataupun observasiTerakhir dapat *null*, karena ada kondisi dimana poligon lahan tersebut belum terbuat, atau pada lahan tersebut belum memiliki observasi. Contoh *response* dari *endpoint* ini dapat dilihat pada Lampiran 1.

Endpoint GET /lahan/:lahan_id digunakan untuk mengambil detail suatu lahan tertentu. *Endpoint* ini tidak memerlukan autentikasi, sehingga dapat diakses oleh semua pengguna. Data yang dikembalikan mencakup lokasi lahan, informasi poligon, serta observasi terakhir yang dilakukan pada lahan tersebut. Secara umum, struktur *response endpoint* ini mirip dengan *endpoint GET* /lahan, namun perbedaannya terletak pada hasil yang dikembalikan, di mana *endpoint* ini hanya mengembalikan data untuk satu lahan saja, bukan daftar lahan secara keseluruhan. Contoh *response* dari *endpoint* ini dapat dilihat pada Gambar 23.

```
{
    "status": 200,
    "message": "Berhasil get single result",
    "data": {
        "lokasi_region": {
            "provinsi": "Kalimantan Tengah",
            "kabupaten": "Kabupaten Barito Utara",
            "kecamatan": "Kecamatan Dusun Hilir",
            "desa": "Desa Dusun Halah"
        },
        "lahan": {
            "lahan_id": "20ed44de-7813-47e8-9c10-9417fbb23c9e",
            "nama_lahan": "Lahan tanah baru di desa dusun halah",
            "jenis_tanah": "Tanah Rawa",
            "latitude": -6.651431132578957,
            "longitude": 111.03647347058305,
            "luasan_lahan": 6823806.62,
            "polygon": {
                "crs": {
                    "type": "name",
                    "properties": {
                        "name": "EPSG:4326"
                    }
                },
                "type": "Polygon",
                "coordinates": [
                    [
                        [
                            114.87144055484606,
                            0.21631826025005188
                        ],
                        [
                            111.22630049471834,
                            0.3534992343076691
                        ],
                        [
                            113.37549597819869,
                            -2.7679608401120754
                        ],
                        [
                            114.87144055484606,
                            0.21631826025005188
                        ]
                    ]
                ]
            }
        }
    }
}
```


Gambar 23 *Response endpoint GET* /lahan/:lahan_id

Endpoint PATCH /lahan/:lahan_id digunakan untuk melakukan perubahan pada data lahan tertentu. Format *request body* yang digunakan sama persis dengan *request body POST* /lahan, namun bedanya adalah semua *field* bersifat opsional serta hanya mengubah sebagian data pada tabel lahan, karena metode yang digunakan adalah *PATCH*, bukan *PUT*. Dengan demikian, pengguna hanya perlu mengirimkan *field* yang ingin diperbarui tanpa perlu mengirimkan seluruh data lahan kembali. Sementara itu, *endpoint DELETE* /lahan/:lahan_id digunakan untuk menghapus data lahan tertentu dari sistem. Selain menghapus data lahan, *endpoint* ini juga akan menghapus semua data observasi yang terkait dengan lahan tersebut. Sistem juga memastikan bahwa hanya pengguna dengan *minimum role* penilai yang dapat mengakses *endpoint* ini.

4.3.4 Kelompok API Observasi

Kelompok API Observasi merupakan inti dari sistem SIPARTAN, yang menangani proses penilaian lahan, perhitungan skor tingkat keparahan kebakaran hutan, serta penyimpanan data indikator yang diinput oleh pengguna. Kelompok API ini memungkinkan pengguna dengan *minimum role* penilai untuk melakukan observasi pada lahan yang telah dibuat serta mengelola data hasil penilaian. Adapun rincian API yang terdapat pada kelompok API Observasi dapat dilihat pada Tabel 12.

Tabel 12 Daftar API pada kelompok API observasi

Endpoint	**Metode**	***Minimum Role***	**Deskripsi**
/observasi	*POST*	Penilai	Membuat data observasi baru pada lahan tertentu
/observasi	*GET*	Tidak ada	Mengambil seluruh data observasi pada sistem.
/observasi/:observasi_id	*GET*	Tidak ada	Mengambil data detail observasi tertentu.
/observasi/:observasi_id	*PATCH*	Penilai	Melakukan perubahan pada data observasi.
/observasi/:observasi_id	*DELETE*	Penilai	Menghapus observasi tertentu.
/observasi/:observasi_id/pdf	*GET*	Penilai	Melakukan konversi data detail observasi tertentu kedalam bentuk pdf dan mengembalikannya.
/observasi/dokumentasi	*POST*	Penilai	Menambahkan dokumentasi baru untuk indikator penilaian observasi tertentu.
/observasi/dokumentasi/:dokumentasi_id	*GET*	Penilai	Mengambil gambar dokumentasi tertentu.

Endpoint	Metode	*Minimum Role*	Deskripsi
/observasi/dokumentasi/:dokumentasi_id	*DELETE*	Penilai	Menghapus gambar dokumentasi tertentu.
/observasi/penilaian	*GET*	Penilai	Mengambil seluruh data indikator penilaian.
/observasi/penilaian	*POST*	Penilai	Menambahkan data indikator penilaian baru.
/observasi/plot/:plot_id	*PATCH*	Penilai	Melakukan perubahan pada data plot tertentu.
/observasi/plot/:plot_id	*DELETE*	Penilai	Menghapus data plot tertentu.

Endpoint POST /observasi digunakan untuk membuat data observasi baru pada lahan tertentu yang telah terdaftar dalam sistem. *Endpoint* ini merupakan kelanjutan dari alur *POST* /lahan, di mana pengguna dapat melakukan observasi pada lahan yang telah dibuat sebelumnya dengan menggunakan lahan_id sebagai referensi. Observasi ini bertujuan untuk menilai tingkat keparahan area pasca karhutla dengan menggunakan serangkaian indikator yang telah ditentukan.

Dalam proses pembuatan observasi, sistem akan menyimpan informasi observasi yang mencakup data umum seperti jenis kebakaran, temperatur, curah hujan, kelembapan udara, tanggal kejadian, dan tanggal penilaian. Selain itu, sistem juga akan membuat data plot, di mana setiap observasi terdiri dari beberapa plot yang mewakili bagian dari lahan yang sedang dinilai. Luas area karhutla dihitung dengan membentuk poligon berdasarkan koordinat plot yang diinput oleh pengguna, kemudian total luas lahan terdampak akan dihitung dari seluruh plot yang ada dalam observasi tersebut.

Setiap plot dalam observasi memiliki sembilan indikator penilaian yang harus dipilih oleh pengguna. Indikator-indikator tersebut meliputi kematian pohon, kerusakan batang bagian terbakar, kerusakan batang jenis kerusakan, kerusakan tajuk, kerusakan akar, kerusakan dedaunan, kerusakan cabang, tingkat keparahan vegetasi terbakar, dan tingkat keparahan kondisi tanah. Masing-masing indikator memiliki bobot tersendiri, yang akan digunakan dalam perhitungan sistem untuk menghasilkan skor setiap plot. Setelah skor untuk setiap plot dihitung, sistem akan melakukan perataan nilai skor dari seluruh plot dalam observasi untuk mendapatkan skor akhir dari lahan yang dinilai. Skor akhir ini kemudian digunakan untuk menentukan tingkat keparahan area pasca karhutla, yang dikategorikan menjadi sangat ringan, ringan, sedang, berat, dan sangat berat.

Semua data yang terlibat dalam proses ini akan disimpan dalam masing-masing tabel pada *database*. Dalam struktur data *request*, dataPlot merupakan *array of objects*, di mana setiap *object* merepresentasikan satu plot dalam lahan tersebut. Setiap plot memiliki atribut penilaian_id, yang merupakan array berisi sembilan ID indikator penilaian yang telah dipilih oleh pengguna. Sistem kemudian akan mengaitkan setiap penilaian_id dengan plot yang sesuai sebelum melakukan perhitungan skor. Adapun *request body* yang harus di-input oleh pengguna dapat dilihat pada Gambar 24.

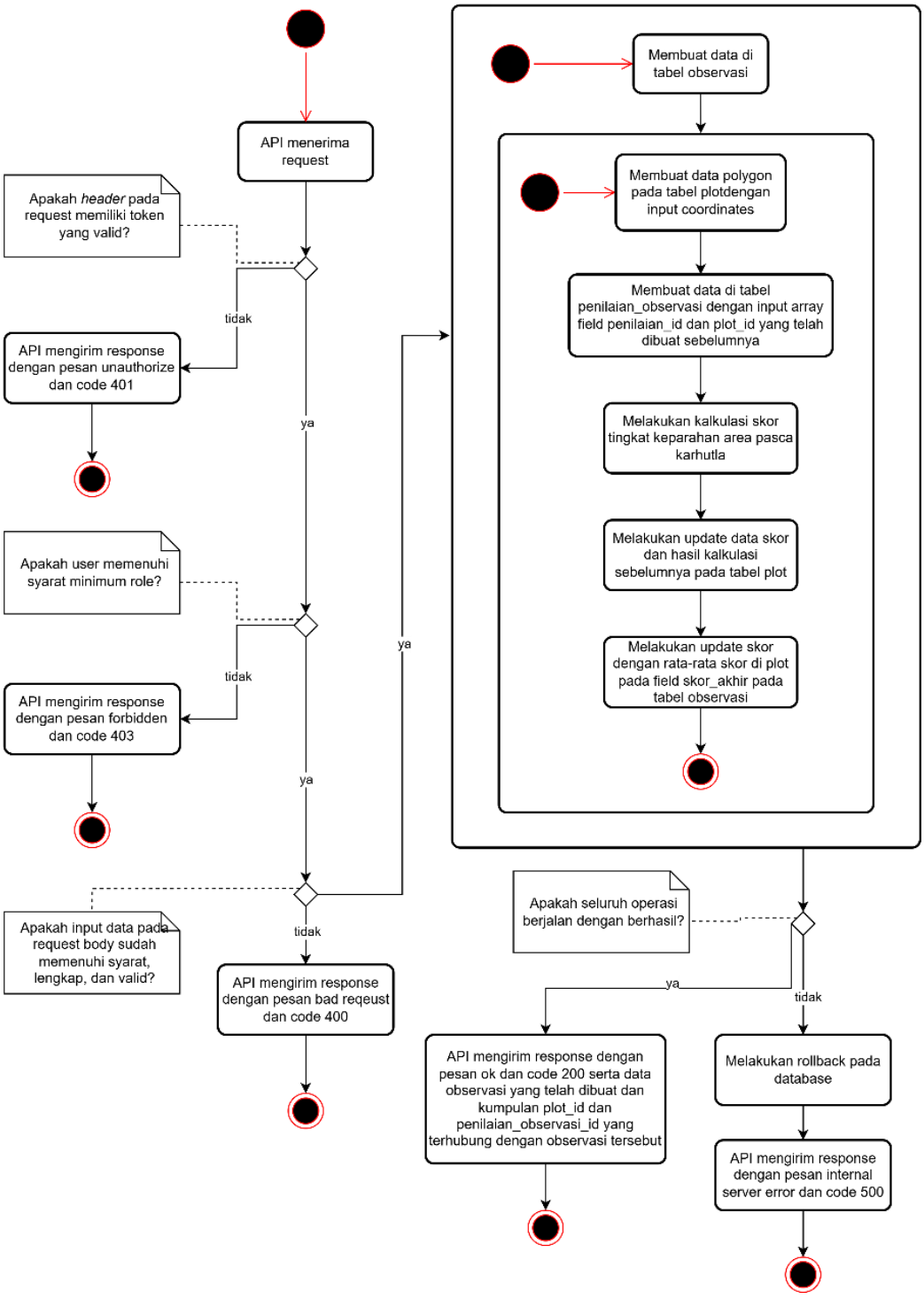
Parameter	Type	Required
lahan_id	string	Yes
jenis_karhutla	string	Yes
temperatur	number	Yes
curah_hujan	number	Yes
kelembapan_udara	number	Yes
tanggal_kejadian	string	Yes
tanggal_penilaian	string	Yes
tutupan_lahan	string	Yes
jenis_vegetasi	string	Yes
tinggi_muka_air_gambut	number	No
penggunaan_lahan	string	Yes
dataPlot	array	Yes
dataPlot[].coordinates	array	Yes
dataPlot[].penilaian_id	array	Yes

Gambar 24 *Request body endpoint POST* /observasi

Ketika API menerima *request* untuk membuat observasi baru, sistem akan melalui beberapa tahap validasi dan pemrosesan data. Proses ini mencakup validasi token autentikasi, pengecekan *role* pengguna, serta validasi kelengkapan data yang dikirimkan. Jika data yang diberikan tidak valid, API akan mengembalikan *response error* yang sesuai. Namun, jika semua data valid, sistem akan menyimpan informasi observasi, kemudian mengonversi koordinat plot menjadi poligon serta menghitung luas setiap plot menggunakan metode yang sama seperti pada endpoint *POST* /lahan. Setelah itu, sistem akan menghitung luasan karhutla sebagai total dari semua luas plot yang ada dalam observasi tersebut. Selain menghitung luas lahan terdampak, sistem juga akan melakukan perhitungan skor untuk setiap plot berdasarkan data penilaian_id yang telah dipilih. Setelah skor individu untuk setiap plot dihitung, sistem akan menghitung skor_akhir sebagai rata-rata dari skor seluruh plot dalam observasi tersebut, yang kemudian akan disimpan dalam tabel observasi. Alur pembuatan data observasi dapat dilihat pada *activity diagram* pada Gambar 25.

Endpoint GET /observasi digunakan untuk mengambil daftar seluruh observasi yang telah dilakukan dalam sistem. *Endpoint* ini dapat diakses tanpa autentikasi, sehingga pengguna yang belum *login* tetap dapat melihat daftar observasi yang telah dilakukan oleh pengguna lain. *Endpoint* ini telah dilengkapi

dengan *pagination*, *search*, dan *filter*, sehingga pengguna dapat melakukan pencarian berdasarkan berbagai kriteria, seperti tanggal kejadian, provinsi, kabupaten, jenis karhutla, hasil penilaian, dan skor keparahan kebakaran hutan. Sistem *pagination* pada *endpoint* ini memiliki mekanisme yang sama seperti pada *endpoint GET* /user. Adapun *query parameter* yang dapat digunakan pada *endpoint* ini dapat dilihat pada Gambar 26.



Gambar 25 *Activity diagram endpoint POST* /observasi

Parameter	Type	Required
page	number	No
limit	number	No
sortBy	string	No
order	string	No
lahan_id	string (UUIDv4)	No
user_id	string (UUIDv4)	No
hasil_penilaian	string	No
skor_min	number	No
skor_max	number	No
tanggal_kejadian_start	string (ISO8601)	No
tanggal_kejadian_end	string (ISO8601)	No
tanggal_penilaian_start	string (ISO8601)	No
tanggal_penilaian_end	string (ISO8601)	No
jenis_karhutla	string	No

Gambar 26 *Query parameter endpoint GET* /observasi

Response dari *endpoint* ini mencakup data observasi beserta informasi lahan dan pengguna yang melakukan observasi. Contoh *response* dapat dilihat pada Gambar 27.


```
{
    "status": 200,
    "message": "Berhasil get observasi",
    "data": {
        "results": [
            {
                "user": {
                    "user_id": "5b2d3ed7-78a4-4dc5-8207-978973c73a50",
                    "nama": "akunnyapenilai",
                    "email": "akunnyapenilai@gmail.com",
                    "instansi": null
                },
                "lokasi_region": {
                    "provinsi": "JAWA BARAT",
                    "kabupaten": "KOTA BOGOR",
                    "kecamatan": "BOGOR BARAT",
                    "desa": "CILENDEK TIMUR"
                },
```


Gambar 27 *Response endpoint GET* /observasi

```
                "lahan": {
                    "nama_lahan": "Lahan rifqi",
                    "jenis_tanah": "Tanah Rawa",
                    "latitude": -6.5759077,
                    "longitude": 106.7777646,
                    "luasan_lahan": 0,
                    "polygon": null
                },
                "observasi": {
                    "observasi_id": "89aa5071-9c64-4695-b0dd-1dc7240e8569",
                    "lahan_id": "fc4a78ff-7fa4-4cae-ad6d-33de6a4af366",
                    "jenis_karhutla": "Kebakaran permukaan",
                    "temperatur": 24.95,
                    "curah_hujan": 3.65,
                    "kelembapan_udara": 91,
                    "tanggal_kejadian": "2025-02-25T00:00:00.000Z",
                    "tanggal_penilaian": "2025-02-25T00:00:00.000Z",
                    "luasan_karhutla": 0,
                    "skor_plot": 56,
                    "hasil_penilaian": "Sedang",
                    "tutupan_lahan": "Perkebunan Monokultur",
                    "jenis_vegetasi": "Jati",
                    "tinggi_muka_air_gambut": 0,
                    "penggunaan_lahan": "Padang rumput",
                    "createdAt": "2025-02-25T13:49:30.845Z",
                    "updatedAt": "2025-02-25T13:49:30.888Z"
                }
            }
        ],
        "page": 1,
        "limit": 1,
        "totalPages": 21,
        "totalResults": 21
    }
}
```


Gambar 27 *Response endpoint GET* /observasi (lanjutan)

Endpoint GET /observasi/:observasi_id digunakan untuk mengambil detail dari observasi tertentu. *Endpoint* ini dapat diakses tanpa autentikasi, sehingga informasi observasi dapat dilihat oleh semua pengguna tanpa memerlukan *token* akses. *Response* dari *endpoint* ini mencakup informasi pengguna yang melakukan observasi, lokasi lahan, data observasi yang telah dilakukan, serta daftar plot yang terkait dengan observasi tersebut. Setiap plot memiliki informasi terkait luas plot, skor plot, serta indikator penilaian yang digunakan. Jika plot memiliki dokumentasi yang telah diunggah sebelumnya, maka data dokumentasi juga akan disertakan dalam *response*. Adapun contoh *response* dari *endpoint GET* /observasi/:observasi_id dapat dilihat pada Lampiran 2.

Endpoint PATCH /observasi/:observasi_id digunakan untuk memperbarui data observasi yang telah dibuat sebelumnya. Perubahan hanya dapat dilakukan pada data observasi utama, seperti jenis karhutla, temperatur, curah hujan, kelembapan udara, tutupan lahan, jenis vegetasi, dan penggunaan lahan. Namun, pengguna tidak dapat mengedit data plot melalui *endpoint* ini. Jika pengguna ingin

mengubah informasi pada plot, seperti poligon plot atau indikator penilaian, maka perubahan harus dilakukan melalui *PATCH* /observasi/plot/:plot_id. Metode *PATCH* dipilih karena logika yang diterapkan dalam sistem tidak mengubah keseluruhan data observasi, melainkan hanya memperbarui sebagian informasi tanpa menghapus atau mengganti data lainnya. Selain itu, pengguna juga tidak dapat mengubah tanggal kejadian dan tanggal penilaian, karena kedua informasi ini bersifat tetap dan hanya bisa ditetapkan saat observasi pertama kali dibuat. Adapun *request body* yang dapat dikirim pada *endpoint* ini dapat dilihat pada Gambar 28.

Parameter	Type	Required
jenis_karhutla	string	No
temperatur	number	No
curah_hujan	number	No
kelembapan_udara	number	No
tutupan_lahan	string	No
jenis_vegetasi	string	No
tinggi_muka_air_gambut	number	No
penggunaan_lahan	string	No

Gambar 28 *Request body endpoint PATCH* /observasi/:observasi_id

Endpoint DELETE /observasi/:observasi_id digunakan untuk menghapus observasi tertentu dari sistem. Jika sebuah observasi dihapus, maka seluruh data terkait, termasuk plot, penilaian observasi, dan dokumentasi dari plot tersebut juga akan ikut terhapus. Sistem akan melakukan validasi kepemilikan observasi, memastikan bahwa hanya pengguna dengan *minimum role* penilai atau admin yang dapat menghapus observasi. Jika permintaan penghapusan valid, maka sistem akan menghapus semua data terkait, termasuk poligon plot, penilaian observasi, serta dokumentasi yang telah diunggah ke MinIO.

Endpoint GET /observasi/:observasi_id/pdf digunakan untuk mengonversi data observasi ke dalam format PDF, yang dapat digunakan sebagai laporan dalam bentuk dokumen. Konversi data dilakukan menggunakan Puppeteer, yang memungkinkan perenderan data dalam format HTML, yang kemudian dikonversi menjadi PDF. Data yang digunakan dalam *endpoint* ini menggunakan data yang sama pada *endpoint GET* /observasi/:observasi_id, sehingga seluruh informasi yang ditampilkan dalam laporan PDF sesuai dengan data observasi ada. Dengan adanya fitur ini, pengguna dapat mengunduh dan menyimpan laporan observasi dalam bentuk dokumen PDF, yang dapat digunakan untuk keperluan dokumentasi, analisis lebih lanjut, serta pelaporan kepada pihak terkait.

Endpoint POST /observasi/dokumentasi digunakan untuk mengunggah dokumentasi dalam bentuk gambar yang berkaitan dengan indikator penilaian

observasi tertentu. Dokumentasi ini berperan dalam memberikan bukti visual mengenai kondisi lahan pasca kebakaran. Sistem membatasi jenis *file* yang dapat diunggah agar sesuai dengan standar dokumentasi, hanya memperbolehkan format *PNG*, *JPG*, dan *JPEG* dengan ukuran maksimum 10MB per *file*. Untuk setiap observasi, pengguna hanya dapat mengunggah maksimal 3 gambar per indikator penilaian, dan setiap indikator dalam satu plot hanya dapat memiliki hingga 3 gambar dokumentasi. *File* yang diunggah dikirimkan dalam format multipart/form-data, yang memungkinkan pengiriman file gambar bersamaan dengan metadata seperti penilaian_observasi_id, lokasi wilayah, tipe dokumentasi, kategori indikator, dan variabel yang dinilai. Metadata ini penting karena akan digunakan untuk mengelompokkan dokumentasi sesuai dengan wilayah dan kategori indikator penilaian yang telah ditentukan.

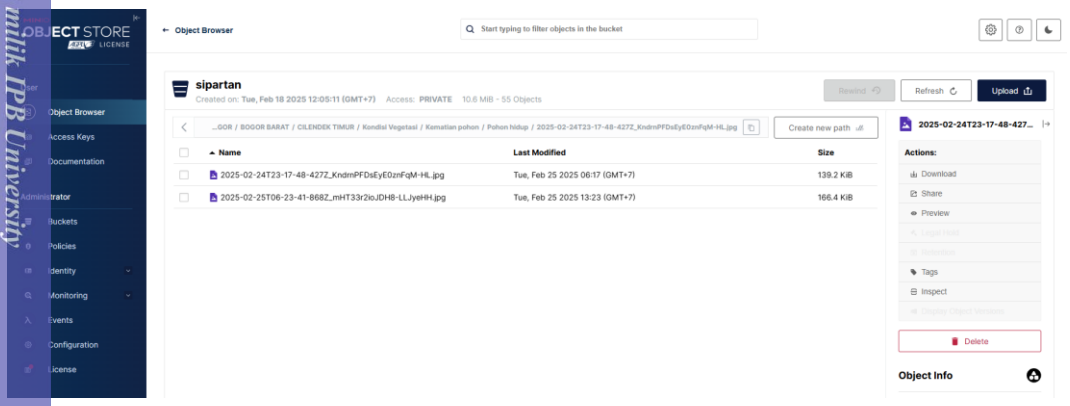
Setiap *file* yang diunggah akan disimpan dalam MinIO, dengan struktur direktori yang telah disusun secara sistematis untuk mempermudah pencarian dan pengelolaan data. Sistem akan membangun *s3_key* dengan format **"**tahun/bulan/provinsi/kabupaten/desa/tipe/kategori/variable/filename**".** Format *filename* menggunakan pola berikut untuk memastikan keunikan dan keteraturan *file* **"**dateISO_nanoID.extension**"** Di mana dateISO merepresentasikan waktu unggah dalam format ISO, nanoID adalah string unik yang dibuat menggunakan *library* nanoID, dan *extension* adalah format asli dari file yang diunggah. Setelah file berhasil diunggah, informasi dokumentasi akan disimpan dalam *database* beserta *s3_key*, yang akan digunakan dalam proses pengambilan dan penghapusan *file* di MinIO. Adapun *request body* yang dikirim dalam endpoint ini dapat dilihat pada Gambar 29.

Parameter	Type	Required
penilaian_observasi_id	string	Yes
provinsi	string	Yes
kabupaten	string	Yes
kecamatan	string	Yes
desa	string	Yes
kategori	string	Yes
tipe	string	Yes
variable	string	Yes
files	file	Yes

Gambar 29 *Request body endpoint POST* /observasi/dokumentasi

Setelah dokumentasi berhasil diunggah, *file* akan disimpan di MinIO di dalam *bucket* sipartan, dengan *s3_key* yang telah dibuat sebelumnya. *File* ini dapat

diakses dan dikelola dengan lebih mudah karena MinIO menyediakan UI manajemen *file*, yang memungkinkan admin untuk melihat, menghapus, atau mengelola file dokumentasi secara langsung melalui antarmuka berbasis web. Dengan adanya UI MinIO, pengguna dapat menelusuri folder penyimpanan, melihat detail dari setiap *file*, serta melakukan tindakan seperti mengunduh atau menghapus dokumentasi langsung dari dashboard. Hal ini mempermudah manajemen dokumentasi dalam skala besar. Tampilan UI dari MinIO untuk penyimpanan dokumentasi dalam *bucket* sipartan dapat dilihat pada Gambar 30.



Gambar 30 Tampilan UI manajamen *file* MinIO

Endpoint GET /observasi/dokumentasi/:dokumentasi_id digunakan untuk mengambil gambar dokumentasi tertentu berdasarkan id dokumentasi yang diberikan sebagai parameter dalam URL. Sistem akan mencari dokumentasi_id dalam *database*, lalu menggunakan *s3_key* pada tabel tersebut untuk mengambil *file* dari MinIO. Setelah *file* ditemukan, sistem akan mengembalikannya langsung dalam format gambar kepada pengguna, sehingga gambar dapat ditampilkan di *frontend* atau aplikasi *mobile* tanpa perlu pengunduhan manual. Proses ini memastikan bahwa pengguna hanya perlu mengetahui ID dokumentasi untuk dapat mengakses gambar yang mereka butuhkan tanpa perlu menelusuri direktori penyimpanan secara manual.

Endpoint DELETE /observasi/dokumentasi/:dokumentasi_id digunakan untuk menghapus dokumentasi observasi tertentu dari sistem. Ketika permintaan penghapusan dilakukan, sistem pertama-tama akan mencari *s3_key* dari dokumentasi_id tersebut di dalam *database*. Setelah ditemukan, sistem akan melakukan dua langkah utama: pertama, menghapus file gambar yang tersimpan di MinIO menggunakan *s3_key* yang terkait, dan kedua, menghapus metadata dokumentasi dalam *database*. Langkah ini memastikan bahwa tidak ada dokumentasi yang tertinggal dalam sistem setelah dihapus, serta menjaga konsistensi antara *database* dan penyimpanan *file*. Sistem juga melakukan validasi untuk memastikan bahwa hanya pengguna dengan *minimum role* penilai yang dapat menghapus dokumentasi.

Endpoint GET /observasi/penilaian digunakan untuk mengambil seluruh data indikator penilaian yang tersedia dalam sistem, sementara *POST* /observasi/penilaian digunakan untuk menambahkan indikator penilaian baru. Dalam versi terbaru sistem ini, terdapat beberapa perubahan utama dalam kedua *endpoint* ini. Salah satu perubahan utama adalah pada struktur *response message*, yang telah diperbarui agar lebih mudah digunakan oleh *frontend* dan aplikasi *mobile*.

Selain itu, kini terdapat validasi *role* pengguna, di mana hanya pengguna dengan *minimum role* penilai yang dapat mengakses *endpoint* ini. Hal ini bertujuan untuk memastikan bahwa hanya pengguna yang memiliki *minimum role* penilai yang dapat menambah atau melihat indikator penilaian dalam sistem.

Endpoint PATCH /observasi/plot/:plot_id digunakan untuk memperbarui informasi plot dalam suatu observasi. Melalui *endpoint* ini, pengguna dapat melakukan perubahan pada koordinat plot serta indikator penilaian yang terkait dengan plot tersebut. Jika koordinat plot diperbarui, sistem akan menghitung ulang luasan plot, untuk memastikan bahwa data tetap akurat dan konsisten. Jika pengguna melakukan perubahan pada indikator penilaian yang terkait dengan plot, maka sistem akan melakukan kalkulasi ulang skor plot, yang pada akhirnya akan berdampak pada skor akhir observasi. Metode *PATCH* dipilih karena sistem hanya melakukan perubahan pada sebagian data dalam plot, bukan mengganti keseluruhan informasi yang ada. Pendekatan ini memungkinkan pengguna untuk melakukan pembaruan data secara lebih fleksibel tanpa kehilangan informasi yang tidak diubah. Adapun *request* body yang dapat dikirimkan pada *endpoint* ini dapat dilihat pada Gambar 31.

Parameter	Type	Required
coordinates	array	No
penilaianList	array	No

Gambar 31 *Request body endpoint PATCH* /observasi/plot/:plot_id

Endpoint DELETE /observasi/plot/:plot_id digunakan untuk menghapus plot dari suatu observasi yang telah dibuat sebelumnya. Saat sebuah plot dihapus, sistem akan melakukan beberapa langkah untuk memastikan konsistensi data dalam sistem. Langkah pertama adalah menghapus data plot dari *database*, yang akan diikuti dengan penghapusan data penilaian observasi yang terkait dengan plot tersebut. Setelah itu, sistem akan melakukan perhitungan ulang skor observasi, di mana skor keseluruhan akan diperbarui berdasarkan rata-rata dari skor plot yang tersisa. Serta sistem akan menghitung kembali luasan_karhutla dari observasi plot tersebut.

4.3.5 Kelompok API Info

Kelompok API Info berfungsi untuk mengambil data API dari pihak ketiga, seperti informasi wilayah administratif, cuaca, dan lokasi berdasarkan koordinat. Semua *endpoint* dalam kelompok ini memerlukan *minimum role* penilai, memastikan bahwa hanya pengguna dengan akses yang sesuai dapat mengambil data dari sumber eksternal. Seluruh data dari API pihak ketiga diambil dengan menggunakan Axios yang memungkinkan sistem untuk melakukan permintaan ke sumber eksternal. Dalam implementasinya, API ini mengambil data wilayah administratif dari API Wilayah Indonesia (emsifa github), data cuaca dari OpenWeather API, dan data lokasi dari Google Maps API untuk *melakukan reverse*

geocoding. Berikut adalah daftar *endpoint* dalam kelompok API Info, sebagaimana ditampilkan pada Tabel 13.

Tabel 13 Daftar API pada kelompok API info

Endpoint	**Metode**	***Minimum Role***	**Deskripsi**
/info/provinces	*GET*	Penilai	Mengambil daftar seluruh provinsi.
/info/regencies/:province_id	*GET*	Penilai	Mengambil list kabupaten berdasarkan provinsi tertentu.
/info/districts/:regency_id	*GET*	Penilai	Mengambil list kecamatan berdasarkan kabupaten tertentu.
/info/villages/:district_id	*GET*	Penilai	Mengambil list desa berdasarkan kecamatan tertentu.
/info/weather/coordinates	*GET*	Penilai	Mengambil data cuaca berdasarkan koordinat.
/info/reverse-geocode/google	*GET*	Penilai	Mengambil data lokasi berdasarkan koordinat.

Endpoint GET /info/provinces digunakan untuk mengambil daftar seluruh provinsi di Indonesia. Data ini diperoleh dari API Wilayah Indonesia, yang menyediakan informasi administratif hingga tingkat desa. Adapun contoh *response body* dari *endpoint* ini dapat dilihat pada Gambar 32.


```
{
    "status": 200,
    "message": "Provinces retrieved successfully",
    "data": [
        {
            "id": "11",
            "name": "ACEH"
        },
        {""
        }
    ]
}
```


Gambar 32 *Response body endpoint GET* /provinces

Endpoint GET /info/regencies/:province_id digunakan untuk mendapatkan daftar kabupaten berdasarkan provinsi tertentu. Pengguna perlu memberikan id

provinsi sebagai parameter dalam URL untuk mendapatkan daftar kabupaten yang berada dalam provinsi tersebut. Adapun contoh *response body* dari *endpoint* ini dapat dilihat pada Gambar 33.


```
{
    "status": 200,
    "message": "Regencies retrieved successfully",
    "data": [
        {
            "id": "9401",
            "province_id": "94",
            "name": "KABUPATEN MERAUKE"
        },
        {""
        }
    ]
}
```


Gambar 33 *Response body endpoint GET* /regencies/:province_id

Endpoint GET /info/districts/:regency_id digunakan untuk mengambil daftar kecamatan berdasarkan kabupaten, sedangkan GET /info/villages/:district_id digunakan untuk mengambil daftar desa berdasarkan kecamatan tertentu. Struktur *response body* dari kedua *endpoint* ini serupa dengan *response* dari *endpoint GET* /info/regencies/:province_id.

Endpoint GET /info/weather/coordinates digunakan untuk mengambil data cuaca berdasarkan koordinat lokasi tertentu. Data cuaca ini diperoleh dari OpenWeather API dan digunakan untuk mendukung pengisian otomatis kondisi cuaca pada observasi lahan pasca kebakaran. Dari keseluruhan data yang dikembalikan oleh API ini, hanya data temperatur, curah hujan, dan kelembapan udara yang digunakan oleh modul *frontend* dan *mobile* untuk mendukung observasi lapangan. Adapun contoh *response body* dari *endpoint* ini dapat dilihat pada Lampiran 3.

Endpoint GET /info/reverse-geocode/google digunakan untuk mengambil informasi lokasi berdasarkan koordinat. Dengan memberikan latitude dan longitude, sistem akan mengembalikan data administratif lokasi, seperti provinsi, kabupaten, kecamatan, dan desa, berdasarkan informasi dari Google Maps API. Fitur ini memungkinkan pengguna untuk mengisi data wilayah administratif secara otomatis, sehingga tidak perlu memasukkan informasi wilayah satu per satu secara manual.

4.4 Pengujian dan *Deployment* Modul *Backend*

Pengujian terhadap modul *backend* SIPARTAN dilakukan menggunakan metode *blackbox testing*, yang bertujuan untuk memastikan bahwa seluruh API yang dikembangkan dapat berfungsi sesuai dengan skenario yang telah dirancang. Pengujian ini dilakukan dengan 104 skenario pengujian, yang mencakup berbagai kondisi skenario positif dan negatif. Setiap skenario diuji secara manual menggunakan Postman, di mana setiap *request* dikirimkan sesuai dengan spesifikasi API yang telah dikembangkan, lalu hasilnya diperiksa berdasarkan status *response* yang diharapkan. Seluruh skenario pengujian berhasil lolos tanpa ada kegagalan, menunjukkan bahwa API *backend* telah berjalan dengan baik sesuai dengan ekspektasi. Pengujian mencakup semua kelompok API yang tersedia dalam sistem, yaitu Auth, User, Lahan, Observasi, dan Info. Rincian skenario pengujian untuk setiap kelompok API dapat dilihat pada Lampiran 4 hingga Lampiran 8.

Selain pengujian yang dilakukan melalui Postman, dua *endpoint* terkait *login* menggunakan akun google, yaitu /auth/google dan /auth/google/callback, diuji langsung melalui browser. Hal ini dilakukan karena *endpoint* /auth/google membutuhkan interaksi UI, di mana pengguna harus memilih akun Google untuk *login* atau memasukkan informasi kredensial akun google. Setelah pengguna memilih akun, mereka diarahkan ke /auth/google/callback, yang menangani *callback* dari Google dan mengembalikan *token* autentikasi beserta data pengguna yang berhasil masuk ke sistem. Pengujian ini memastikan bahwa integrasi *login* menggunakan Google berjalan dengan lancar, tanpa adanya kendala dalam proses autentikasi maupun *callback* data pengguna.

Setelah modul *backend* berhasil diuji, tahap selanjutnya adalah *deployment* menggunakan sistem kontainerisasi dengan Docker. Dalam proses ini, *backend* dikelola menggunakan Dockerfile dan Docker Compose, yang memungkinkan setiap layanan berjalan dalam lingkungan yang terisolasi dan lebih mudah untuk dipindahkan ke server produksi. Infrastruktur yang digunakan untuk *deployment* adalah VPS dari Digital Ocean, dengan spesifikasi 4 GB RAM, 2 CPU, 80 GB SSD Disk, dan menggunakan sistem operasi Ubuntu 24.10 x64. Proses *deployment* dilakukan dengan menggunakan Docker Compose untuk menjalankan *container backend* serta layanan pendukungnya, seperti PostgreSQL untuk *database* dan MinIO untuk *object storage*. Dengan pendekatan ini, sistem dapat dikelola secara efisien, lebih mudah untuk dilakukan *scaling*, serta lebih fleksibel dalam proses *update* dan perbaikan di masa mendatang.

4.5 Kekurangan Modul *Backend*

Meskipun pengembangan modul *backend* SIPARTAN telah menghadirkan berbagai peningkatan signifikan dalam sistem penilaian tingkat keparahan area pasca karhutla, masih terdapat beberapa aspek yang perlu diperbaiki untuk meningkatkan akurasi, efisiensi, dan fleksibilitas sistem.

Salah satu kekurangan utama adalah bahwa sistem saat ini masih memungkinkan posisi poligon dari plot berada di luar poligon lahan yang bersangkutan. Hal ini berpotensi menimbulkan inkonsistensi dalam representasi spasial dan mengurangi akurasi dalam perhitungan luas area terdampak kebakaran. Idealnya, setiap plot yang dibuat dalam sistem harus selalu berada di dalam batas poligon lahan yang sesuai. Selain itu, sistem saat ini hanya memungkinkan setiap poligon lahan atau plot berada dalam satu wilayah administratif, baik itu kecamatan maupun desa. Padahal, dalam beberapa kasus nyata, suatu lahan dapat membentang melintasi lebih dari satu wilayah administratif. Hal ini menyebabkan data wilayah administratif yang tersimpan dalam sistem menjadi kurang akurat, terutama ketika pengguna ingin melakukan analisis berbasis wilayah. Terakhir, keterbatasan dalam penggunaan data wilayah administratif juga menjadi tantangan. Saat ini, data wilayah administratif yang digunakan dalam sistem masih mengacu pada API pihak ketiga yang memiliki keterbatasan dalam akurasi dan pembaruan data.

V SIMPULAN DAN SARAN

5.1 Simpulan

Penelitian ini berhasil mengembangkan modul *backend* Sistem Penilaian Tingkat Keparahan Area Pasca Kebakaran Hutan dan Lahan (SIPARTAN) dengan menambahkan fitur-fitur untuk meningkatkan efisiensi dan akurasi dalam penilaian tingkat keparahan area pasca karhutla. Fitur yang dikembangkan mencakup sistem verifikasi email, *login* menggunakan akun Google, dukungan untuk penilaian ulang lahan yang sama, serta optimasi pencarian data melalui implementasi *pagination*, *search*, dan *filter*. Selain itu, sistem telah diperbarui dengan *role-based access control* (RBAC) untuk mengelola hak akses pengguna, serta penyimpanan data koordinat dalam format poligon untuk representasi area terdampak yang lebih akurat. Pengujian dilakukan menggunakan metode *blackbox testing* dengan 104 skenario uji yang semuanya berhasil tanpa *error*. Sementara itu, pengujian *login* menggunakan akun google dilakukan melalui *browser* dan menunjukkan hasil yang sesuai dengan harapan. Hasil dari penelitian ini dapat meningkatkan efisiensi digitalisasi dalam penilaian lahan pasca kebakaran hutan, membantu tim lapangan melakukan evaluasi secara lebih cepat dan akurat.

5.2 Saran

Pada pengembangan lebih lanjut, disarankan agar menerapkan sistem *blacklist token* untuk meningkatkan keamanan dalam proses verifikasi email, *reset password*, serta *logout*, sehingga *token* yang telah direvokasi tidak bisa lagi digunakan. Penggunaan *access token* dan *refresh token* juga direkomendasikan untuk manajemen sesi yang lebih aman dan efisien, memungkinkan pengguna memperpanjang sesi tanpa harus *login* ulang secara terus-menerus. Selain itu, sistem belum memiliki mekanisme yang menangani perubahan atau penghapusan indikator dengan baik, sehingga dapat menyebabkan inkonsistensi data skor observasi. Selain itu, saat ini penambahan plot ke dalam observasi setelah observasi dibuat belum didukung, membatasi fleksibilitas dalam pencatatan observasi di lapangan. Sebaiknya, sistem memungkinkan penambahan plot tambahan dengan tetap menjaga keakuratan perhitungan skor dan luasan karhutla. Terakhir, untuk mengurangi ketergantungan pada API pihak ketiga, disarankan agar SIPARTAN menyimpan dan mengelola sendiri data wilayah administratif Indonesia dalam *database* sistem.

Untuk meningkatkan akurasi representasi spasial dalam sistem, disarankan agar dilakukan validasi geometri plot sehingga setiap plot yang dibuat harus berada di dalam poligon lahan yang bersangkutan. Hal ini dapat diterapkan dengan memanfaatkan fungsi spasial PostGIS, seperti *ST_Within*, guna memastikan bahwa semua plot berada dalam batas lahan yang sesuai. Selain itu, sistem juga perlu dikembangkan agar mampu menangani lahan yang membentang di lebih dari satu wilayah administratif, seperti kecamatan atau desa.

DAFTAR PUSTAKA

Afina FS. 2022. Estimasi tingkat keparahan kebakaran lahan gambut di Kabupaten Siak, Provinsi Riau [skripsi]. Bogor: Institut Pertanian Bogor.

Baihaqi R. 2024. Modul Front-End pada Sistem Penilaian Tingkat Keparahan Area Pasca Kebakaran Hutan dan Lahan [skripsi]. Bogor: Institut Pertanian Bogor.

Edwards RB, Naylor RL, Higgins MM, Falcon WP. 2019. Causes of Indonesia's forest fires. World Development. 127(2020):1-13. doi: https://doi.org/10.1016/j.worlddev.2019.104717.

Fielding RT. 2000. Software architectural styles for network-based applications [disertasi]. Irvine: University of California.

Hutapea OC. 2024. Aplikasi Mobile Penilaian Tingkat Keparahan Area Pasca Kebakaran Hutan dan Lahan [skripsi]. Bogor: Institut Pertanian Bogor.

Kementrian Lingkungan Hidup dan Kehutanan. 2018. *The State of Indonesia's Forest 2018*. Jakarta: Kementrian Lingkungan Hidup dan Kehutanan.

Massé M. 2012. REST API Design Rulebook – Designing Consistent RESTful Web Service Interfaces. California (US): O'Reilly Media.

Maulana R. 2024. Modul Backend dalam Sistem Penilaian Tingkat Keparahan Area Pasca Kebakaran Hutan dan Lahan [skripsi]. Bogor: Institut Pertanian Bogor.

[MenLHK] Kementerian Lingkungan Hidup dan Kehutanan. 2024. SiPongi+. Jakarta (ID): Menteri Lingkungan Hidup dan Kehutanan Republik Indonesia.

Mumbaikar S, Padiya P. 2013. Web services based on soap and rest principles. Int. J. Sci. Res. Publ. 3(5):1–4.

Pasai M. 2020. Dampak kebakaran hutan dan penegakan hukum. Jurnal Pahlawan. 3(1):36-45. doi: https://doi.org/10.31004/jp.v3i1.609.

Prayoga MBR, Koestoer RH. 2021. Improving Forest Fire Mitigation in Indonesia: A Lesson from Canada. Jurnal Wilayah Dan Lingkungan. 9(3) : 293–305.

Rachman A, Saharjo BH, Putri EIK. 2020. Strategi Pencegahan Kebakaran Hutan dan Lahan di Kesatuan Pengelolaan Hutan Kubu Raya, Ketapang Selatan, dan Ketapang Utara di Provinsi Kalimantan Barat. Jurnal Ilmu Pertanian Indonesia (JIPI). 25(2):213-223.

Sari F. 2023. Identifying anthropogenic and natural causes of wildfres by maximum entropy method-based ignition susceptibility distribution models. Journal of Forestry Research. 34(5) : 355–371.

Soni A, Ranga V. 2019. API features individualizing of web services: REST and SOAP. Int J Innov Technol Explor Eng. 8(9 Special Issue):664–671. doi:10.35940/ijitee.I1107.0789S19.

Syaufina L. 2017. Metode penilaian areal pasca kebakaran hutan. 1st ed. Gumelar AD, editor. Bogor: IPB Press.

Webber J, Parastatidis S, Robinson I. 2010. REST in Practice Hypermedia and Systems Architecture. CA(USA): O'Reilly Media.Inc

Yulianti N. 2018. Pengenalan Bencana Kebakaran dan Kabut Asap Lintas Batas [Studi Kasus Eks Proyek Lahan Gambut Sejuta Hektar]. PT Penerbit IPB Press.

Yusuf A, Hapsoh H, Siregar SH, Nurrochmat DR. 2019. Analisis Kebakaran Hutan Dan Lahan Di Provinsi Riau. *Dinamika Lingkungan Indonesia*. 6(2):67.doi:10.31258/dli.6.2.p.67-84.