BAB 3
PELAKSANAAN KERJA MAGANG

3.1 Kedudukan dan Koordinasi

Pelaksanaan kerja magang di Dinas Perhubungan Kota Tangerang Selatan dilakukan dengan menempati posisi sebagai *backend developer intern* di bawah supervisi Bapak Heris Cahya Kusuma, S.T., yang bertindak sebagai *supervisor* selama periode magang. Kegiatan magang dimulai pada tanggal 10 Februari 2025 dan berakhir pada tanggal 13 Juni 2025.

Selama durasi magang, kegiatan dilaksanakan bersama tim pengembangan *website* yang terdiri dari empat orang. Tugas yang dikerjakan difokuskan pada pengembangan panel admin untuk pengelolaan konten. Selain itu, fitur-fitur yang bertujuan meningkatkan visibilitas *website* melalui integrasi kecerdasan buatan juga dikembangkan, dengan kemampuan memberikan rekomendasi SEO yang disesuaikan berdasarkan isi konten. Pengembangan turut dilanjutkan untuk meningkatkan aspek keamanan panel admin melalui implementasi *authentication logging* dan manajemen pengguna.

3.2 Tugas yang Dilakukan

Selama pelaksanaan kerja magang di Dinas Perhubungan Kota Tangerang Selatan, proyek akan dikembangkan menggunakan pendekatan *Minimum Viable Product*. Pendekatan *Minimum Viable Product*, atau disingkat MVP, dapat didefinisikan sebagai metode pengembangan dengan fitur terpenting sebagai titik minimum sebuah produk. Pengembangan berfokus pada skalabilitas demi kemudahan pengembangan dan penambahan fitur-fitur untuk waktu mendatang. Berdasarkan analisis terhadap kebutuhan *website*, pengembangan difokuskan pada fitur manajemen berita atau artikel, disertai dengan penerapan *Search Engine Optimization* (SEO).

Pengembangan *website* akan menggunakan bahasa pemrograman PHP, dengan *framework* Laravel. Untuk memudahkan pengembangan *panel admin*, digunakan Filament beserta *plugin* yang disediakan oleh Filament. Untuk keperluan penyimpanan data, digunakan PostgreSQL guna memanfaatkan fitur penyimpanan data berupa *JSON*, sehingga memungkinkan *website* menjadi lebih fleksibel dalam implementasi *Content Management System*.

Berikut rincian tugas yang dikerjakan selama magang:

1. Pembelajaran *framework* Laravel versi 12, Filament versi 3 untuk perancangan panel admin.
2. Pengembangan panel untuk pengelolaan konten *website*, publikasi berita, dan pengelolaan pengguna.
3. Melakukan *testing*, *debugging* serta *bug fixing* untuk memastikan aplikasi berjalan sesuai dengan yang fungsi yang dikembangkan.

3.3 Uraian Pelaksanaan Magang

Pelaksanaan kerja magang diuraikan seperti pada Tabel 3.1.

Tabel 3.1. Pekerjaan yang dilakukan setiap minggu selama pelaksanaan kerja magang

Minggu Ke-	Pekerjaan yang Dilakukan
1	Pembahasan dan persiapan proyek serta pembelajaran sistem, *framework*, dan DBMS. Mendesain ERD untuk Artikel Berita.
2	Riset *best practices* rancangan SEO untuk publikasi artikel pada website dan pengembangan flowchart untuk pembuatan artikel.
3	Mengubah *database default* dari MySQL menjadi PostgreSQL, dan mulai proses pengembangan sistem untuk CRUD artikel.
4	Pengembangan fitur penjadwalan, pengembangan API untuk merekam pengunjung yang membaca artikel, dan pembuatan sistem CRUD untuk mengelola video YouTube yang ditampilkan pada website.
5	Pengujian fitur penjadwalan, kemudian merancang fitur rekomendasi SEO menggunakan AI.
6	Pengembangan fitur rekomendasi SEO menggunakan integrasi AI Prism PHP, serta melakukan penyesuaian dan debugging.

Lanjut pada halaman berikutnya

Tabel 3.1 Pekerjaan yang dilakukan tiap minggu selama pelaksanaan kerja magang (lanjutan)

Minggu Ke-	Pekerjaan yang Dilakukan
7	Perbaikan *bug* dan fitur-fitur yang bermasalah, melakukan testing secara keseluruhan pada website untuk persiapan presentasi kepada perusahaan.
8	Perancangan dan pengembangan fitur berita terkait menggunakan *Relation Manager* dari Filament.
9	Pengembangan API untuk mengirim data berita terbaru dan berita terkait ke *frontend*. Pengembangan keamanan autentikasi.
10	Pengembangan pengelolaan pengguna dalam panel admin serta penerapan *role* untuk pembatasan izin pengelolaan konten.
11	Pengembangan *logging authentication* untuk merekam aktivitas login dari pengguna atau ancaman penyelundupan.
12	Pengembangan panel admin utama untuk menampilkan data-data yang perlu disorot.
13	Pengembangan *Media Library Manager* untuk mengelola konten-konten berupa file seperti gambar dan pdf yang dapat diakses dalam website.
14	Penyelesaian pengembangan dan inplementasi `Media Library Manager` pada *website*. Memulai Proses Pengembangan Fitur *Backup dan Restore database* pada *website*.
15	Pengembangan, pengujian, dan perbaikan fitur *backup dan restore*.
16	Pemeriksaan seluruh fitur dalam *website*, melakukan perbaikan dan pembersihan proyek untuk presentasi.

3.3.1 Riset mengenai SEO dan CMS

Selama masa pelaksanaan magang, dilakukan riset mendalam mengenai prinsip-prinsip dasar dan praktik terbaik dalam *Search Engine Optimization* (SEO) serta penggunaan *Content Management System* (CMS).Tujuan dari riset ini adalah

untuk memastikan bahwa sistem manajemen konten yang dikembangkan tidak hanya fungsional secara teknis, tetapi juga mampu mengoptimalkan visibilitas artikel di mesin pencari seperti Google. Riset ini dilakukan dengan mengacu pada literatur daring, dokumentasi teknis, serta studi kasus dari situs web berita profesional.

A *Search Engine Optimization*

SEO merupakan serangkaian teknik dan strategi yang digunakan untuk meningkatkan keterlihatan suatu situs web di halaman hasil pencarian. Dalam riset ini, didalami dua aspek utama dari *Search Engine Optimization* (SEO), yaitu SEO konten dan SEO teknis. SEO konten mencakup pengoptimalan elemen-elemen seperti judul artikel, meta deskripsi, struktur *heading*, penggunaan kata kunci, serta tautan internal. Elemen-elemen ini sangat berpengaruh terhadap bagaimana artikel dikenali dan ditampilkan oleh mesin pencari [7].

Sementara itu, SEO teknis lebih menekankan pada aspek struktur dan performa sistem, seperti kecepatan pemuatan halaman, struktur URL yang rapi, pemanfaatan schema markup, dan penggunaan sitemap XML [8]. Penerapan SEO teknis membantu mesin pencari dalam mengindeks konten dengan lebih baik, serta memastikan bahwa halaman-halaman dalam sistem dapat dirayapi dengan optimal. Riset terhadap SEO ini memberikan landasan penting dalam merancang sistem backend yang tidak hanya berfungsi sebagai alat pengelolaan konten, tetapi juga mendukung strategi pemasaran digital jangka panjang [9].

Tabel 3.2. Elemen-Elemen Utama SEO Konten dan SEO Teknis

SEO Konten	**SEO Teknis**
Judul SEO	Sitemap
Meta Description	Struktur URL
Heading	Mobile Friendly
Kata Kunci	Page Speed
Internal Link	Schema Markup

B *Content Management System*

Dalam upaya membangun sistem manajemen konten yang efektif, dilakukan studi literatur dan evaluasi mendalam terhadap berbagai pendekatan *Content Management System* (CMS), termasuk sistem berbasis *framework* seperti Laravel.

Salah satu penelitian yang relevan adalah pengembangan CMS berbasis Laravel untuk *website* jurusan di Politeknik Negeri Bali, yang menunjukkan bahwa CMS *custom* berbasis Laravel mampu mencapai kualitas UX dan fungsionalitas yang tinggi, dengan tingkat penilaian mencapai 86% "baik" dan 14% "sangat baik" dibandingkan implementasi sebelumnya [10]. Penelitian ini menegaskan keunggulan pendekatan *framework* dalam hal keamanan, performa, serta kemudahan penyesuaian struktur basis data dan fitur.

Selain itu, studi lain yang dilakukan di Universitas Djuanda memperlihatkan bahwa CMS berbasis Laravel mampu meningkatkan interaktivitas dan efisiensi pengelolaan konten pada institusi pendidikan, dengan menghadirkan antarmuka yang ramah pengguna serta arsitektur modular yang mudah diadaptasi sesuai kebutuhan [11]. Berdasarkan hasil riset yang dilakukan, Laravel dan Filament dipilih untuk membangun *Content Management System* (CMS) berita, karena antarmuka administrasi yang disediakan cukup intuitif bagi staf non-teknis, sekaligus memberikan fleksibilitas penuh untuk pengembangan lanjutan.

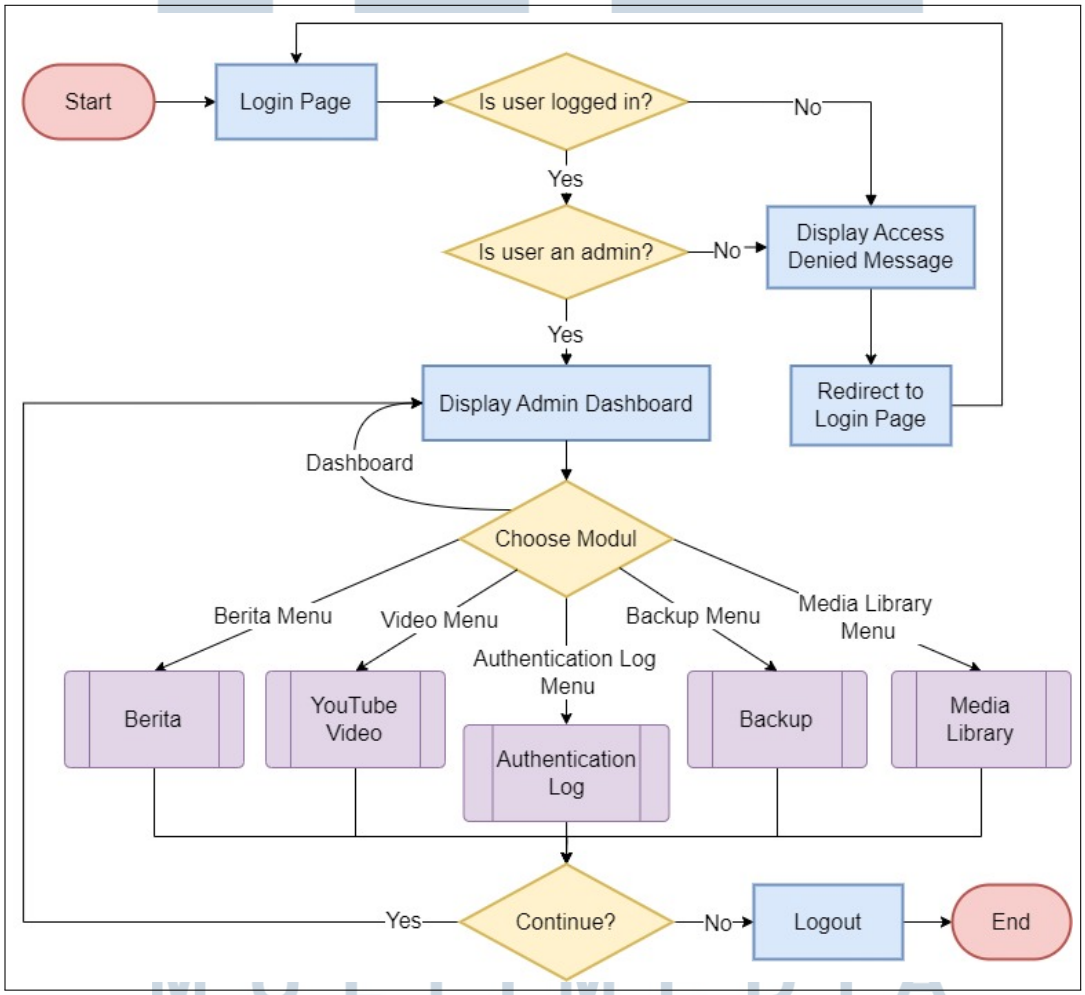
Melalui riset tersebut, sistem CMS berhasil dirancang dengan fitur-fitur penting seperti formulir pengisian artikel, halaman manajemen berita terkait, sistem *tagging*, dan metadata SEO dibangun menggunakan komponen inti dari Filament. Pendekatan ini tidak hanya mempermudah operasional staf, tetapi juga menghadirkan struktur kode dan basis data yang modular dan dapat dijaga skalabilitasnya seiring pertumbuhan konten.

C Proses Perancangan

Tahap perancangan merupakan fondasi kritis dalam siklus pengembangan sistem *Content Management System* (CMS), terutama untuk memastikan bahwa sistem yang dibangun mampu menjawab kebutuhan instansi secara tepat guna, terukur, dan dapat dikembangkan lebih lanjut. Dalam konteks pengembangan CMS untuk Dinas Perhubungan Kota Tangerang Selatan, proses perancangan diarahkan untuk mendeskripsikan alur kerja sistem dari perspektif pengguna internal, yaitu administrator dan editor, yang memiliki otoritas dalam mengelola konten digital yang akan ditampilkan kepada publik. Oleh karena itu, desain sistem tidak hanya mempertimbangkan aspek fungsionalitas, tetapi juga kemudahan penggunaan, efisiensi alur kerja, serta skalabilitas pengembangan.

Proses perancangan dimulai dengan identifikasi kebutuhan dan aktivitas utama yang dilakukan oleh pengguna sistem, mulai dari proses autentikasi

(login), validasi peran sebagai admin, akses ke panel admin utama, hingga navigasi ke berbagai modul pengelolaan konten. Modul-modul yang diidentifikasi pada tahap awal meliputi pengelolaan berita, video YouTube, manajemen log autentikasi, pengelolaan media dokumentasi (*media library*), serta sistem pencadangan data (*backup*). Alur-alur tersebut kemudian diformalkan dalam bentuk diagram *flowchart* utama yang menggambarkan proses interaksi pengguna secara menyeluruh. *Flowchart* ini disusun untuk memberikan representasi visual mengenai bagaimana pengguna mengakses sistem, modul apa saja yang tersedia, serta jalur keluarnya dari sistem setelah menyelesaikan tugasnya.



Gambar 3.1. Flowchart Admin Panel

Dalam *flowchart* yang dirancang pada Gambar 3.1, pengguna pertama-tama diarahkan ke halaman *login*, kemudian sistem akan memverifikasi apakah pengguna telah berhasil masuk dan memiliki hak akses sebagai admin. Jika lolos verifikasi, pengguna akan diarahkan ke panel admin utama yang menampilkan berbagai modul

yang dapat dipilih sesuai tugas dan tanggung jawabnya. Setelah memilih modul, pengguna dapat melakukan proses pengelolaan data, seperti membuat konten berita, mengunggah video, melihat log aktivitas pengguna, atau mengelola dokumen publikasi. Setelah tugas selesai, pengguna dapat memilih untuk melanjutkan ke tugas lainnya, atau keluar dari sistem melalui proses *logout*. Alur ini dirancang sedemikian rupa agar sistem dapat digunakan secara berulang dan fleksibel tanpa mengharuskan pengguna kembali ke halaman awal setiap kali menyelesaikan tugasnya.

Flowchart utama ini bukan hanya berfungsi sebagai alat dokumentasi awal, tetapi juga sebagai dasar pengambilan keputusan dalam pengembangan sistem secara modular. Dengan adanya kerangka alur kerja yang telah ditetapkan secara sistematis, proses pengembangan fitur-fitur lanjutan, seperti integrasi kecerdasan buatan untuk optimasi metadata SEO, penerapan sistem publikasi terjadwal berdasarkan zona waktu, dan kontrol akses berbasis peran (RBAC), dapat dilakukan dengan lebih mudah dan terarah. Selain itu, *flowchart* ini juga menjadi rujukan utama dalam proses pengujian fungsional dan validasi sistem, karena seluruh jalur interaksi telah dirancang dan didefinisikan dengan jelas.

Secara keseluruhan, proses perancangan pada tahap awal ini berperan sebagai kerangka konseptual untuk seluruh siklus hidup pengembangan CMS. Penjabaran lebih rinci terkait masing-masing modul, termasuk struktur basis data, skenario penggunaan, dan pengujian, akan disampaikan pada bagian-bagian selanjutnya. Dengan perancangan yang sistematis dan terdokumentasi dengan baik sejak tahap awal, diharapkan sistem CMS yang dikembangkan dapat memenuhi kebutuhan instansi secara optimal, sekaligus memberikan fondasi yang kuat untuk pengembangan berkelanjutan di masa depan.

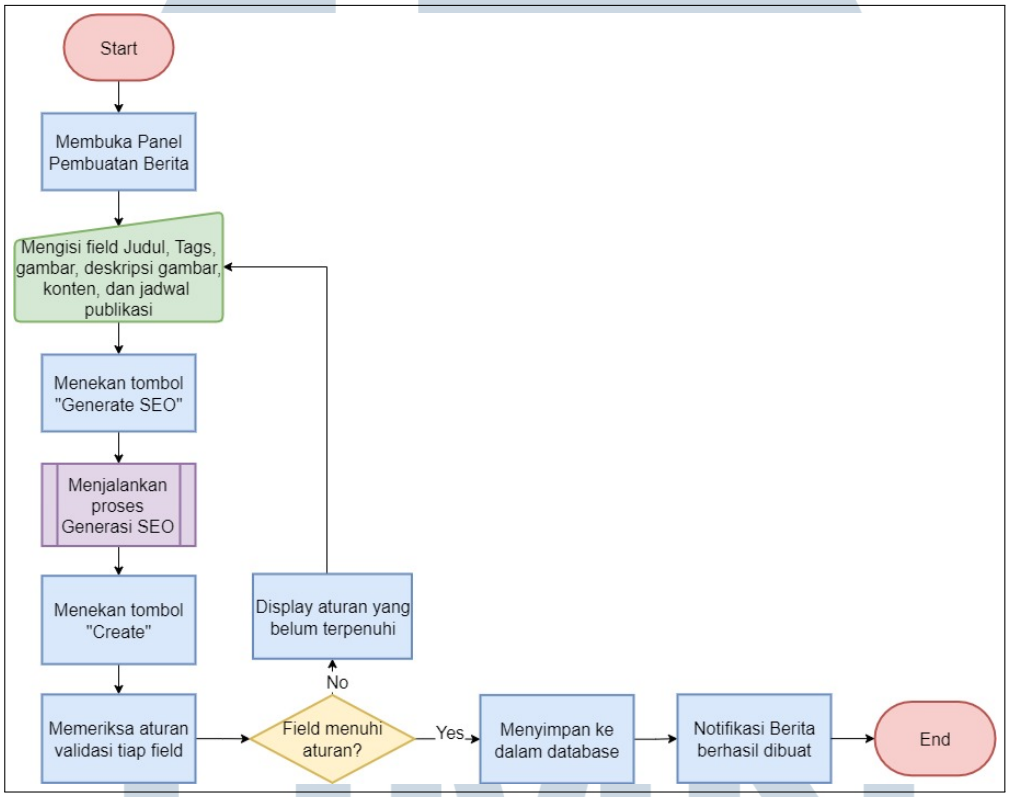
3.3.2 Perancangan Publikasi Berita

Perancangan publikasi berita pada situs pemerintah, dalam konteks ini pada Dinas Perhubungan Kota Tangerang Selatan, bertujuan untuk meningkatkan visibilitas dan penyebaran luas informasi. Oleh karena itu, diperlukan perancangan yang matang untuk menghasilkan fitur berita yang efektif pada *website*. Sebelum memulai pengembangan sistem CRUD berita, dilakukan analisis terhadap fitur-fitur yang umum diperlukan dalam publikasi berita. Berdasarkan hasil analisis tersebut, telah ditentukan fitur-fitur yang dibutuhkan untuk mendukung proses publikasi berita, sebagaimana ditampilkan pada Tabel 3.3.

Tabel 3.3. Fitur-fitur yang diperlukan untuk publikasi berita

Fitur	**Deskripsi**	**Alasan**
Judul	Judul utama dari artikel berita untuk ditampilkan di halaman *website*.	Menjadi informasi utama dari isi artikel untuk pembava.
Slug	Judul yang dijadikan tautan utama untuk mengakses artikel, menggunakan huruf kecil dan dipisahkan menggunakan tanda penghubung (-).	Digunakan sebagai tanda pengenal dalam *routing*.
Thumbnail	Gambar utama yang berhubungan dengan artikel.	Menjadi penarik mata bagi pengunjung untuk membaca artikel.
Konten	Konten utama artikel menggunakan format HTML.	Inti dari berita, terdiri dari paragraf, *heading* dan/atau gambar.
Tag	Label tambahan berupa kata kunci artikel.	Meningkatkan kemudahan pencarian dan pengelompokkan artikel berdasarkan kata kunci.
Status	Status artikel: `Draft`, `Published` atau `Archived`.	Memfasilitasi workflow editorial, Artikel dapat direncanakan dan tidak langsung dipublikasikan.
Tanggal Publikasi	Tanggal dan waktu penerbitan artikel.	Memungkinkan penjadwalan otomatis.
SEO Metadata	Metadata seperti `og_title`, `og_description`, `og_image`, dan `canonical_url`.	Penting untuk meningkatkan visibilitas dan mencegah duplikasi konten di mesin pencarian dan media sosial.
Views	Menghitung jumlah pembaca.	Digunakan untuk analitik popularitas.
Article Access Log	Menyimpan data pengunjung.	Digunakan untuk analitik lanjutan serta keamanan dan pelacakan aktivitas pengguna dalam situs.

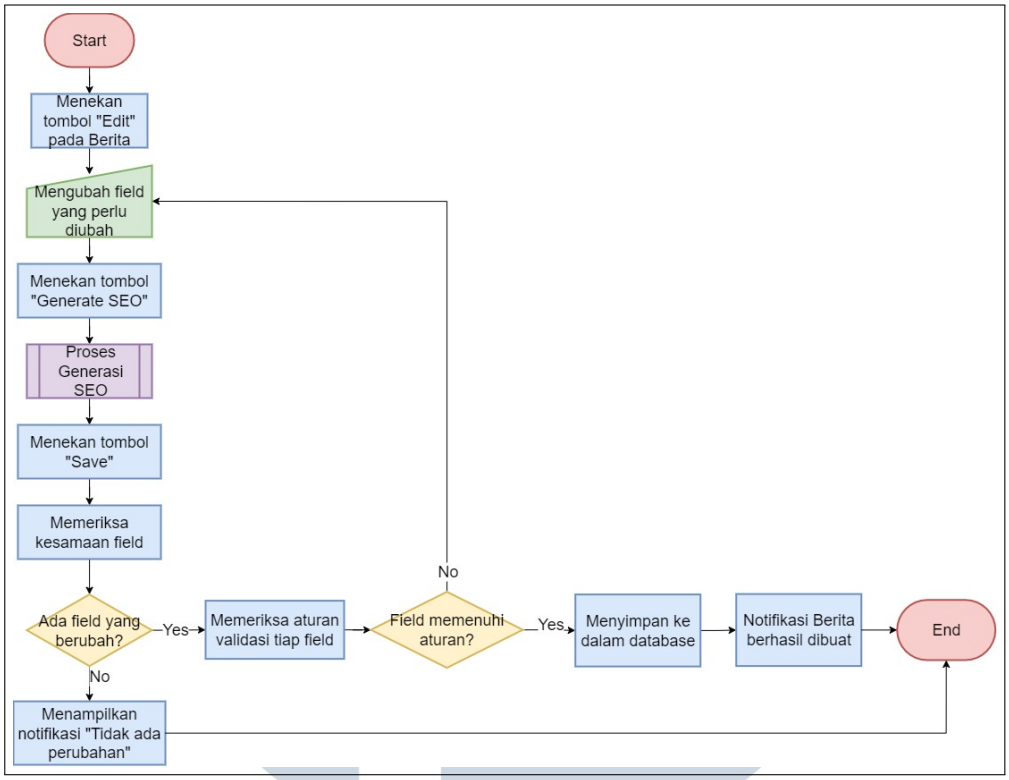
Untuk menggambarkan alur proses pembuatan dan publikasi berita dari sisi pengguna, telah dibuat sebuah diagram alur yang menunjukkan langkah-langkah yang dilakukan mulai dari pembukaan panel pembuatan berita, pengisian konten dan metadata, hingga penyimpanan data ke dalam sistem. Diagram ini juga menunjukkan tahapan validasi dan proses otomatisasi SEO sebelum berita dipublikasikan.



Gambar 3.2. Flowchart Pembuatan Berita

Gambar 3.2 memaparkan rangkaian proses yang terlibat dalam publikasi berita, yang mencakup pengisian data konten, pembangkitan otomatis metadata SEO, serta validasi terhadap setiap *field* sebelum data disimpan secara permanen. Proses validasi ini bertujuan untuk memastikan bahwa setiap isian telah memenuhi ketentuan sistem, seperti panjang karakter judul, ketersediaan gambar, dan keunikan *slug*. Apabila ditemukan ketidaksesuaian, sistem akan memberikan umpan balik agar pengguna dapat melakukan koreksi sebelum melanjutkan proses penyimpanan.

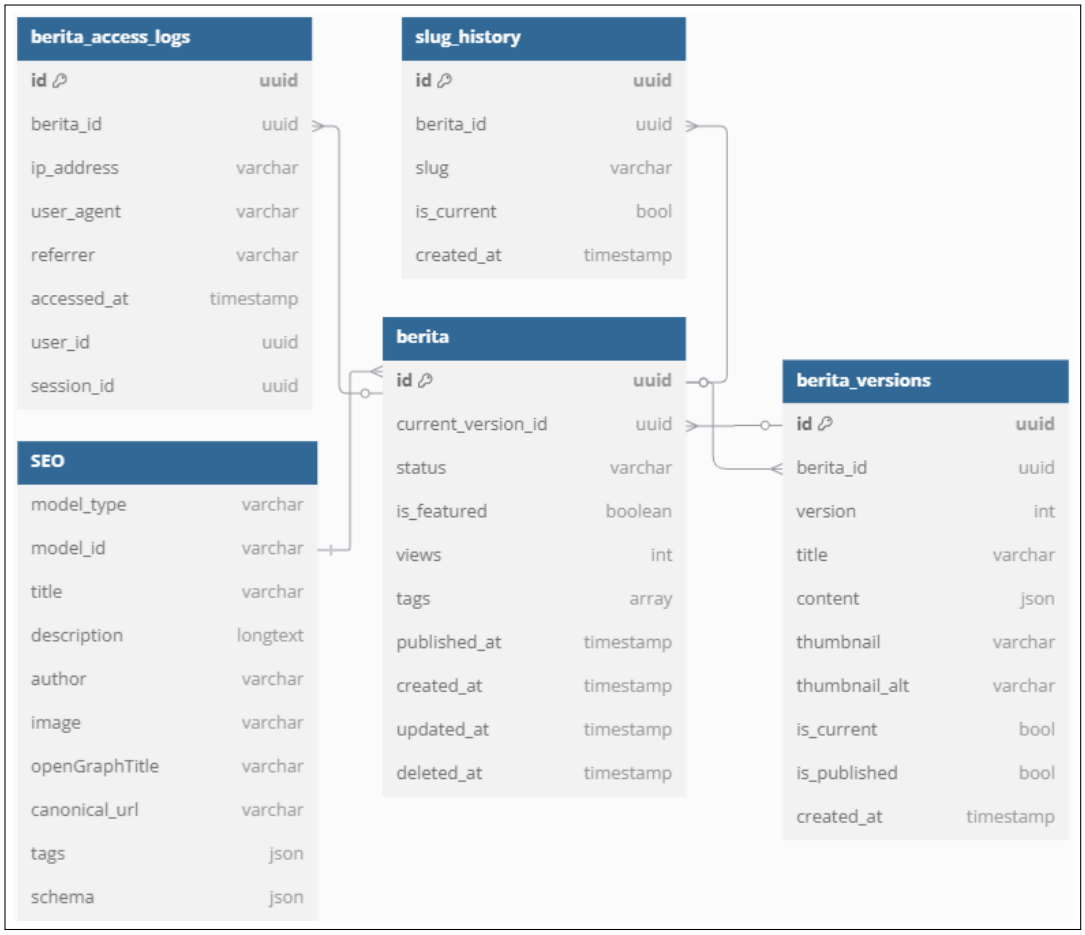
Selain proses pembuatan berita, sistem juga mendukung pembaruan atau pengeditan berita yang telah dibuat sebelumnya. Proses ini dirancang agar setiap perubahan yang dilakukan tetap mengikuti standar validasi dan dapat menghasilkan metadata SEO yang baru sesuai isi yang diperbarui.



Gambar 3.3. Diagram Alur Proses Pengeditan Berita

Gambar 3.3 menyajikan tahapan proses yang dilakukan pengguna saat melakukan pengeditan berita. Melalui alur ini, sistem memastikan bahwa hanya perubahan yang signifikan yang akan disimpan ke dalam basis data. Selain itu, setiap perubahan tetap melewati proses validasi untuk menjamin kelengkapan dan kualitas data yang diperbarui.

Berdasarkan fitur-fitur yang telah ditentukan untuk publikasi berita, dirancang *Entity Relation Diagram* (ERD) sebagaimana ditunjukkan pada Gambar 3.4.

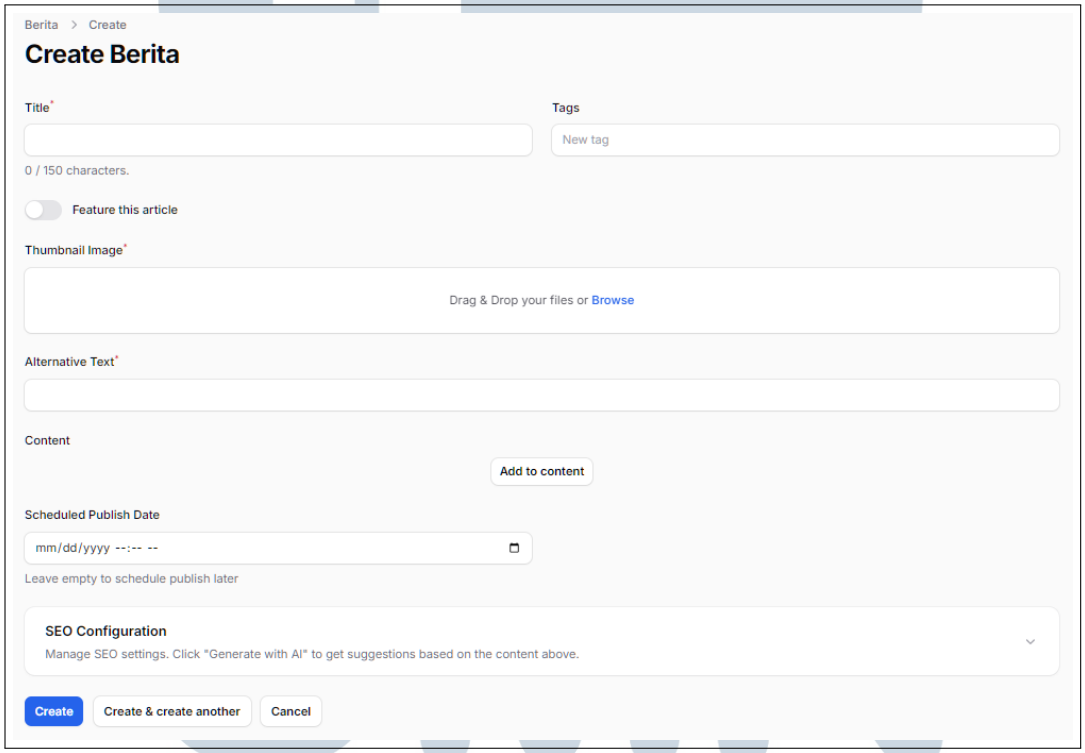


Gambar 3.4. *Entity Relation Diagram* untuk Berita

Gambar 3.4 menjadi acuan dari pengembangan sistem CRUD (*Create, Read, Update, Delete* sebuah artikel. Entitas berita, atau bisa disebut sebagai tabel berita dalam *database*, menjadi pusat dari tabel-tabel lainnya. Pada tabel ini, dimasukkan data penting seperti id, status, *views* dan waktu publikasi. Seluruh isi konten dari berita ditampung dalam tabel `berita_versions`. Menggunakan metode ini, *website* dapat dikembangkan lebih lanjut menggunakan *versioning*, memudahkan *editor* untuk menggunakan versi artikel sebelumnya atau melakukan pembaruan pada artikel. Terdapat juga tabel `slug_history` yang menampung seluruh rekaman *slug* dari berita, sehingga memungkinkan pengalihan dari *slug* lama menuju yang baru jika terjadi perubahan pada judul. SEO metadata dipisahkan dari tabel berita dikarenakan SEO metadata ingin dihubungkan juga pada Model lain dari Laravel. Kemudian, tabel `berita_access_logs` akan menyimpan data-data pengunjung yang mengakses berita, menjadi acuan ketika menambahkan *views*. Selain menjadi acuan, tabel ini juga direncanakan untuk analitik pengunjung serta keamanan *website*.

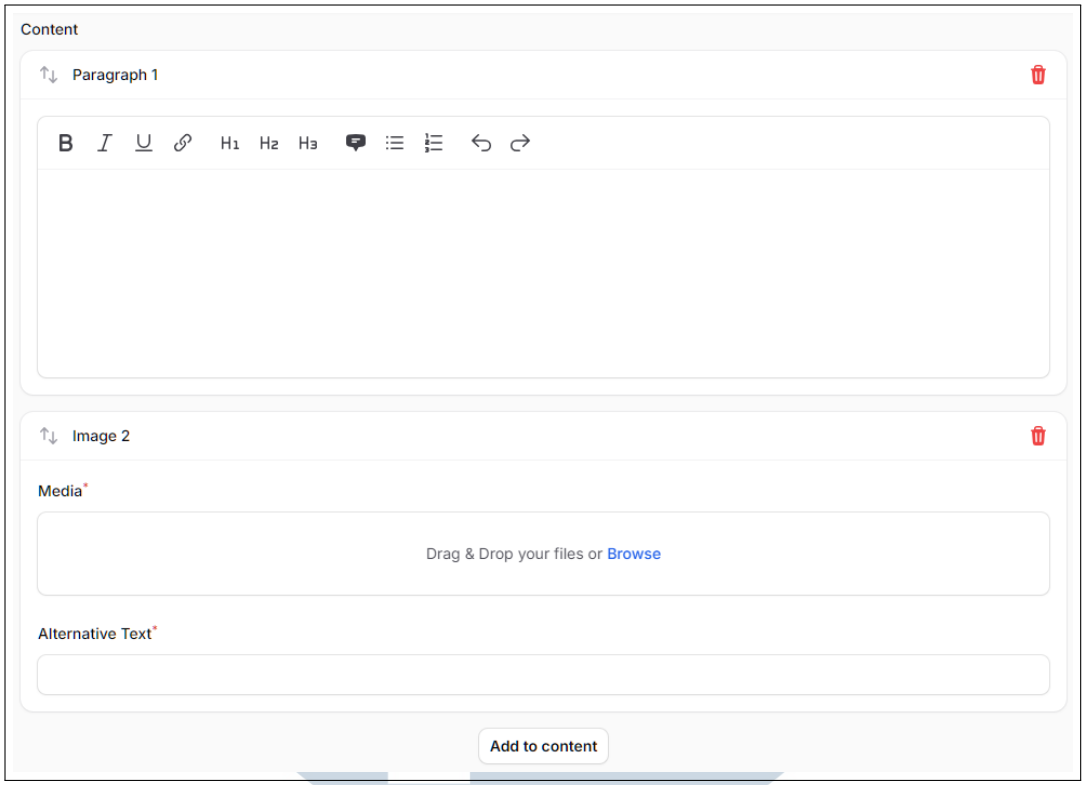
3.3.3 Pengembangan Modul Berita

Tugas selanjutnya adalah pengembangan modul untuk proses pembuatan, pembaruan, pembacaan, dan penghapusan data berita. Menggunakan Filament, proses ini telah dimudahkan dengan adanya komponen-komponen bawaan. Filament memiliki *Resources* yang merupakan kelas-kelas PHP yang digunakan untuk mengembangkan antarmuka CRUD untuk model `Eloquent` yang terdapat dalam proyek. Melalui *resource* tersebut, tersedia *form* dan *table* yang memungkinkan pengguna untuk berinteraksi dengan model `Berita`. Pengembangan modul menggunakan Filament ini mengacu pada ERD Gambar 3.4.



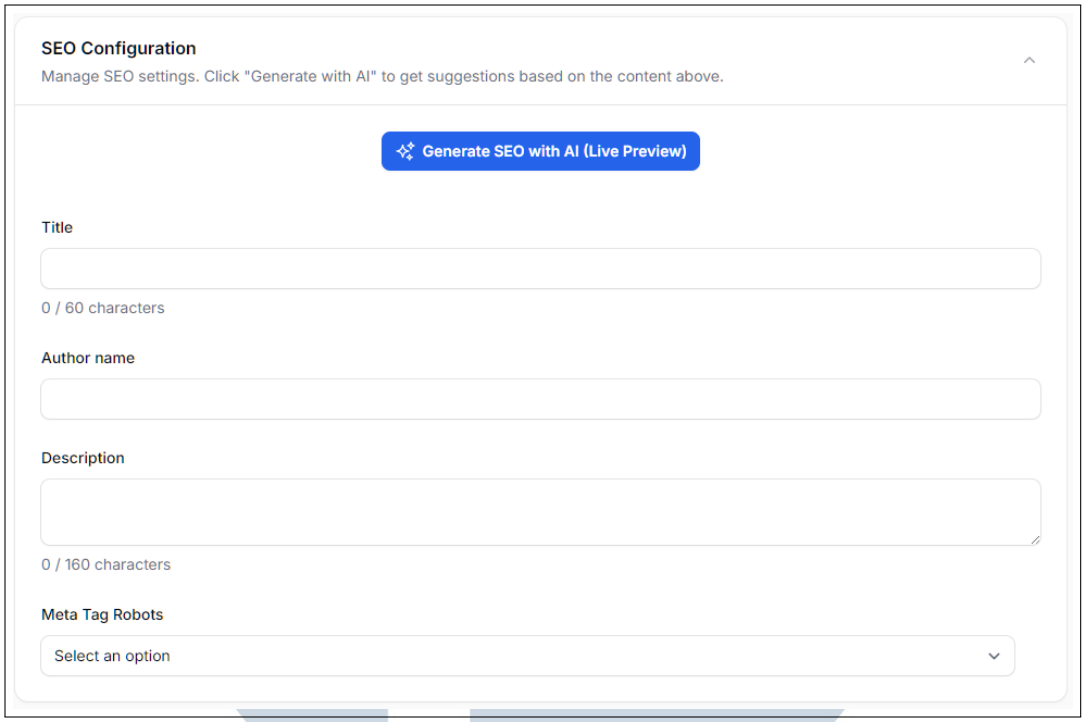
Gambar 3.5. Panel Pembuatan Berita

Modul Berita dikembangkan sebagai sarana utama bagi staf Dishub untuk mempublikasikan informasi penting kepada masyarakat. Untuk membuat berita, staf akan diarahkan menuju halaman tampilan pada Gambar 3.5. Pada halaman ini, staf akan diminta untuk mengisi judul, *tags*, gambar, penjelasan gambar, konten, dan tanggal publikasi.



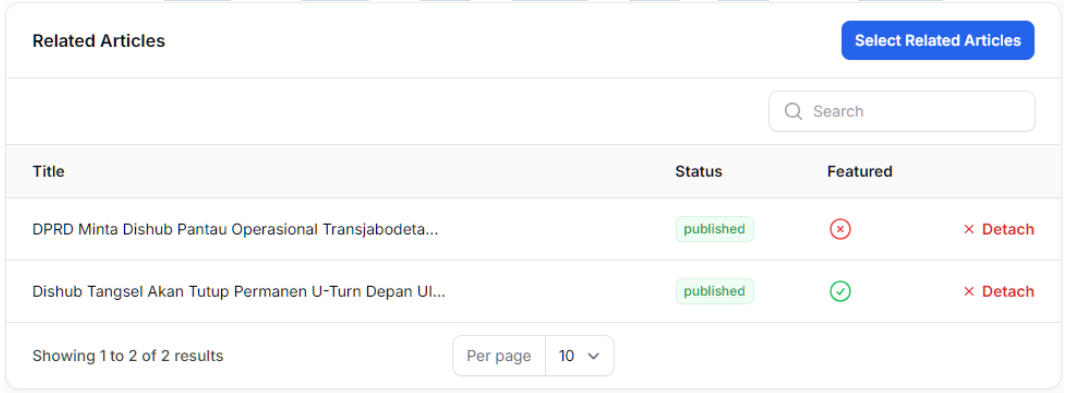
Gambar 3.6. *Form* content Berita

Konten yang terdapat pada *form* modul Berita terdiri dari dua jenis komponen, yaitu paragraf dan gambar. Agar staf Dishub dapat menuliskan berbagai bagian paragraf dengan lebih leluasa, bagian paragraf dari konten menggunakan *Rich Editor* dibanding *Textbox*. Pemilihan ini didasarkan pada alasan untuk membuat konten yang disajikan dalam berita, dapat dikembangkan menjadi informasi yang detail dan mudah dimengerti. *Rich Editor* memungkinkan penyusunan teks dengan berbagai opsi format seperti *heading*, garis bawah, cetak miring, dan berbagai format lain, serupa dengan menggunakan aplikasi pengolah kata pada umumnya. Pada Gambar 3.6, dapat dilihat susunan dari konten. Paragraf dan gambar yang diisikan dalam konten ini, akan disimpan dalam bentuk `JSONB` untuk mempertahankan fleksibilitas.



Gambar 3.7. *Form* konfigurasi SEO

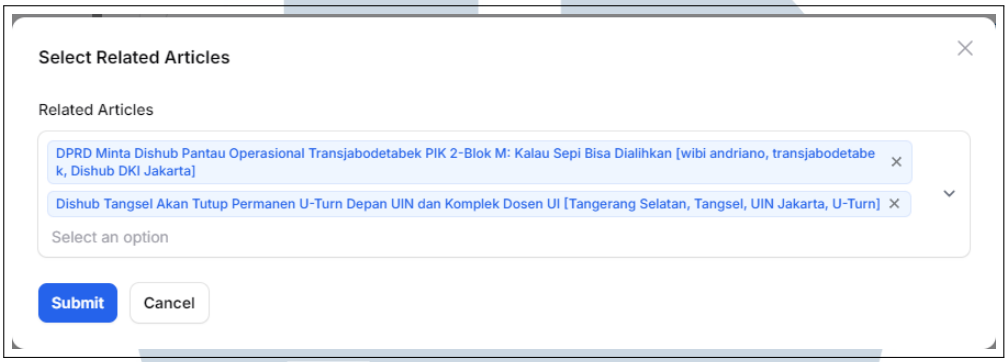
Bagian terakhir dari *form* modul Berita terlihat pada Gambar 3.7, terdapat konfigurasi SEO. Konfigurasi ini bertujuan untuk mengisi metadata SEO yang diperlukan dalam berita. Dalam konfigurasi SEO, staf tidak diperlukan untuk membuat sendiri SEO untuk berita, melainkan cukup menekan tombol "*Generate SEO with AI*" yang akan mengisi judul dan deskripsi. Judul dan deskripsi yang dihasilkan akan mengikuti aturan-aturan yang disarankan untuk SEO, seperti gaya bahasa dan panjang dari teks. AI akan mengambil isi dari judul dan konten sebagai parameter untuk menghasilkan metadata SEO yang sesuai.



Gambar 3.8. *Form* tabel pengelolaan berita terkait

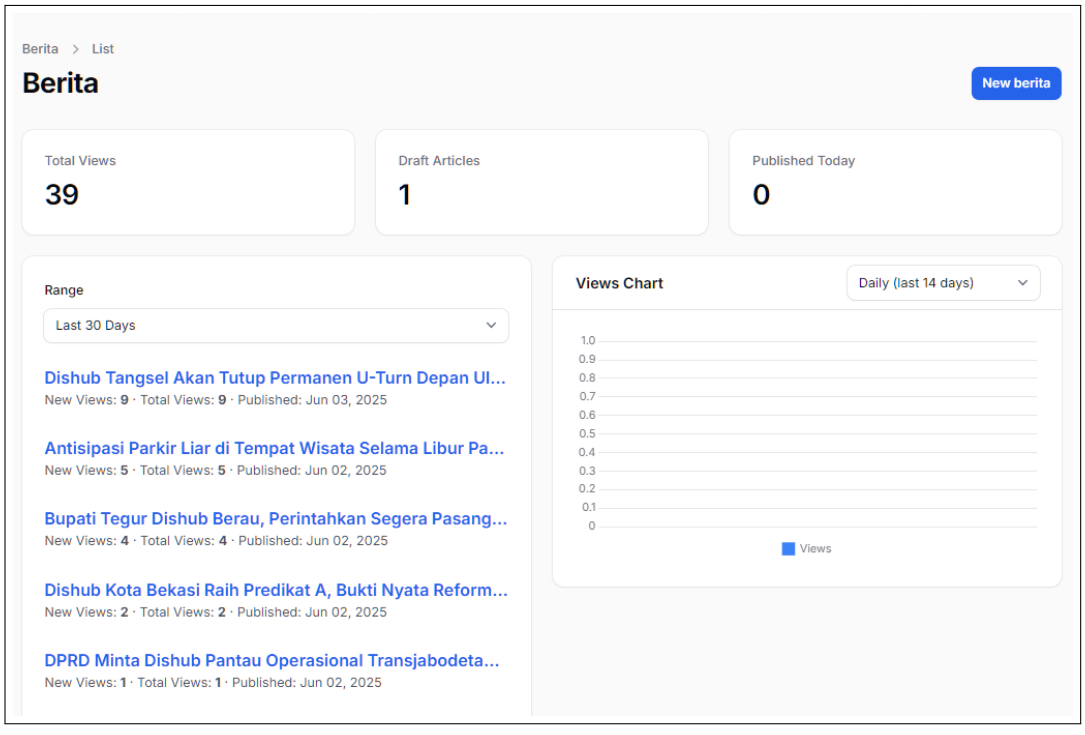
Setelah menyimpan berita yang telah dibuat, staf dapat mengatur berita lain

yang memiliki keterkaitan dengan berita yang telah dibuat, atau sebaliknya. Bagian ini, terlihat pada Gambar 3.8, ditampilkan dalam bentuk tabel berisikan judul, status dari berita, dan ”apakah difiturkan?”. Bagian ini menggunakan fitur yang dimiliki oleh Filament untuk mengatur relasi, yaitu *Relation Manager*.



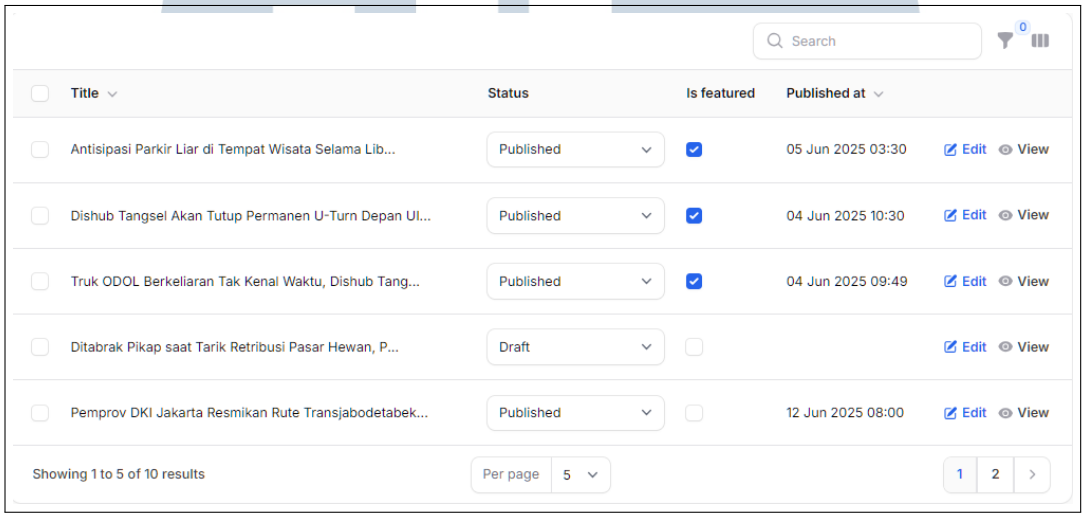
Gambar 3.9. *Modal* penambahan berita terkait

Dengan mendefinisikan hubungan dari Model yang ingin dikelola, Filament dapat mempermudah pengaturan dari hubungan tersebut. Dalam konteks ini, berita terkait memiliki hubungan *Many-to-Many*. Setiap berita terkait yang ditambahkan melalui *form* pada Gambar 3.9 akan secara otomatis menciptakan hubungan dua arah. Hal ini menghilangkan kebutuhan staf untuk menautkan kembali berita dari arah sebaliknya.



Gambar 3.10. *Widget* Halaman Berita

Gambar 3.10 menampilkan *widget* dari panel modul Berita. Panel admin secara dasar, tersusun dari *widget* yang menjadi penampil informasi utama. Pada bagian atas, diberikan data-data secara keseluruhan seperti jumlah *views*, jumlah berita yang belum dipublikasikan dan yang dipublikasikan pada hari ini. Di bawahnya, terdapat *widget* yang menampilkan berita-berita yang sering diakses berdasarkan jangkauan durasi seperti 30 hari yang lalu, 7 hari yang lalu, atau jangkauan hari yang diinginkan oleh staf. Selanjutnya adalah *widget* untuk statistik jumlah pembaca dengan jangkauan harian, mingguan, bulanan, hingga tahunan.



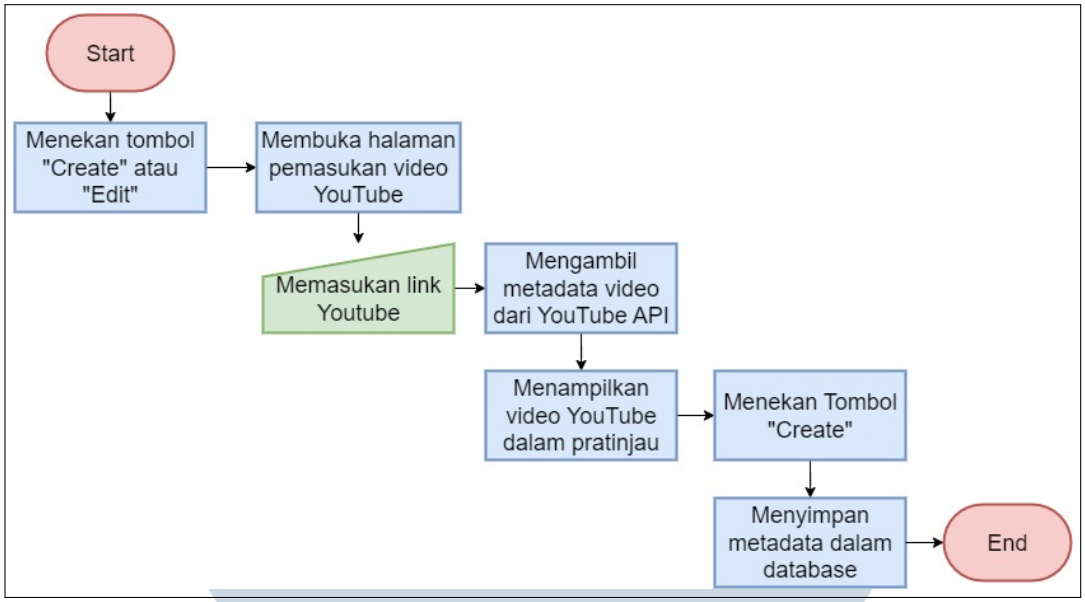
Gambar 3.11. Tabel Halaman Berita

Bagian terakhir adalah tabel untuk mengatur berita-berita yang telah dibuat. Tabel dapat terlihat pada Gambar 3.11. Tabel terdiri dari kolom judul, status, ”Apakah difiturkan?”, dan tanggal publikasi. Terdapat juga tombol untuk mengedit berita dan untuk melihat berita. Tabel juga memiliki fitur untuk mencari berita berdasarkan judul, sehingga mempermudah pencarian berita. Staf bisa langsung mengubah status berita melalui tabel. Jika statusnya diubah menjadi ”*Published*”, sistem akan otomatis menetapkan tanggal publikasi ke waktu sekarang, asalkan tanggalnya belum diatur atau jadwal publikasinya belum lewat.

3.3.4 Pengembangan Modul YouTube

Untuk memperkuat aspek visual dan memperluas jangkauan informasi kepada masyarakat, direncanakan penayangan video YouTube pada *website* Dinas Perhubungan Kota Tangerang Selatan. Video tersebut akan ditampilkan baik pada halaman beranda sebagai video profil instansi, maupun pada halaman berita yang

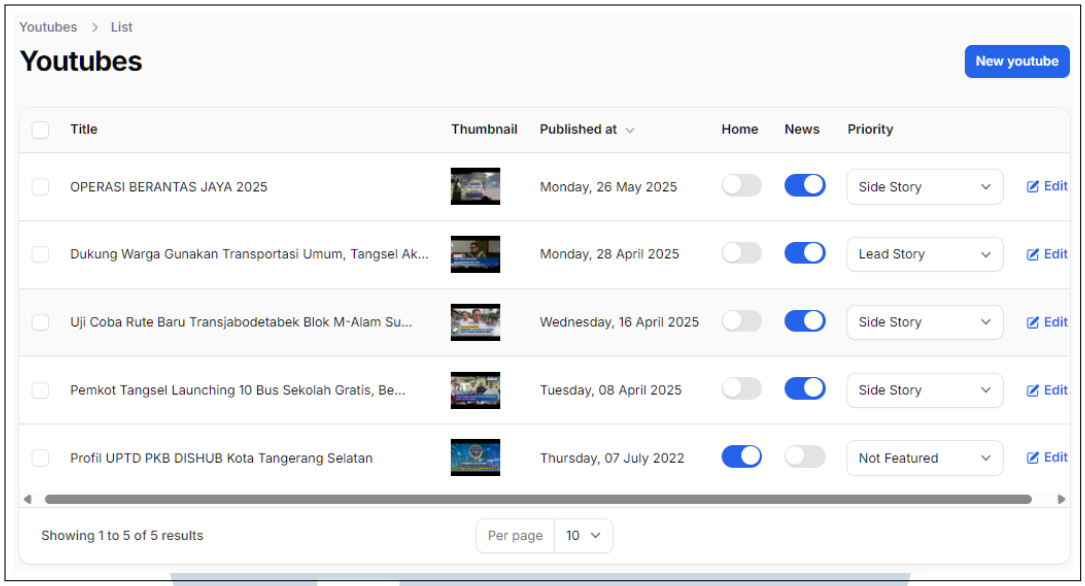
berkaitan dengan kegiatan dan informasi publik Dishub.



Gambar 3.12. Flowchart Penambahan Video YouTube

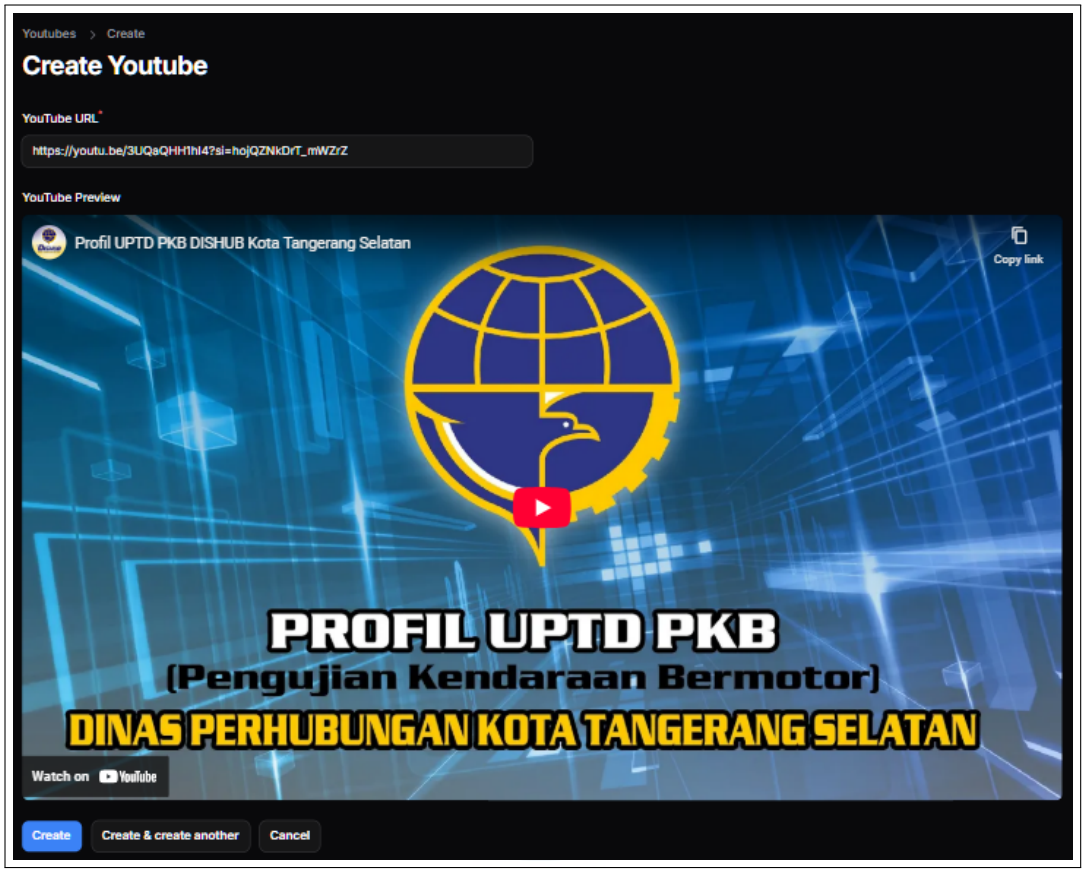
Gambar 3.12 menyajikan alur sistem dalam proses penambahan video YouTube ke dalam sistem manajemen konten situs. Proses dimulai ketika pengguna memasukkan tautan video ke dalam formulir input. Sistem kemudian mengambil metadata video secara otomatis dari API YouTube, seperti judul, gambar *thumbnail*, dan tanggal publikasi. Data ini ditampilkan dalam bentuk pratinjau untuk memastikan kesesuaian dengan konten yang dimaksud. Setelah diverifikasi oleh pengguna, data akan disimpan ke dalam basis data, dan sistem akan menampilkan video tersebut sesuai dengan pengaturan visibilitas (beranda atau halaman berita) serta prioritas tampilan. Alur ini dirancang agar proses pengelolaan video dapat dilakukan dengan efisien dan akurat tanpa perlu intervensi manual yang kompleks.

Selain berita yang ditampilkan pada *website*, direncanakan juga penampilan video YouTube sebagai profil dari Dinas Perhubungan Kota Tangerang Selatan. Video YouTube direncanakan untuk ditampilkan pada halaman beranda dan berita. Pada halaman beranda akan ditampilkan video profil dari Dishub, dan pada halaman berita akan ditampilkan video-video berita yang berhubungan dengan Dinas Perhubungan Kota Tangerang Selatan.



Gambar 3.13. Halaman Utama Video YouTube

Pengelolaan video YouTube dapat dilakukan pada modul Video YouTube dalam halaman utama modul. Terlihat pada Gambar 3.13, diberikan tabel berisi judul video, gambar *thumbnail*, tanggal video dipublikasikan, tombol untuk mengatur apakah video ditampilkan di halaman beranda atau di halaman berita, dan prioritas untuk mengatur posisi video pada halaman berita.



Gambar 3.14. *Form* Video YouTube

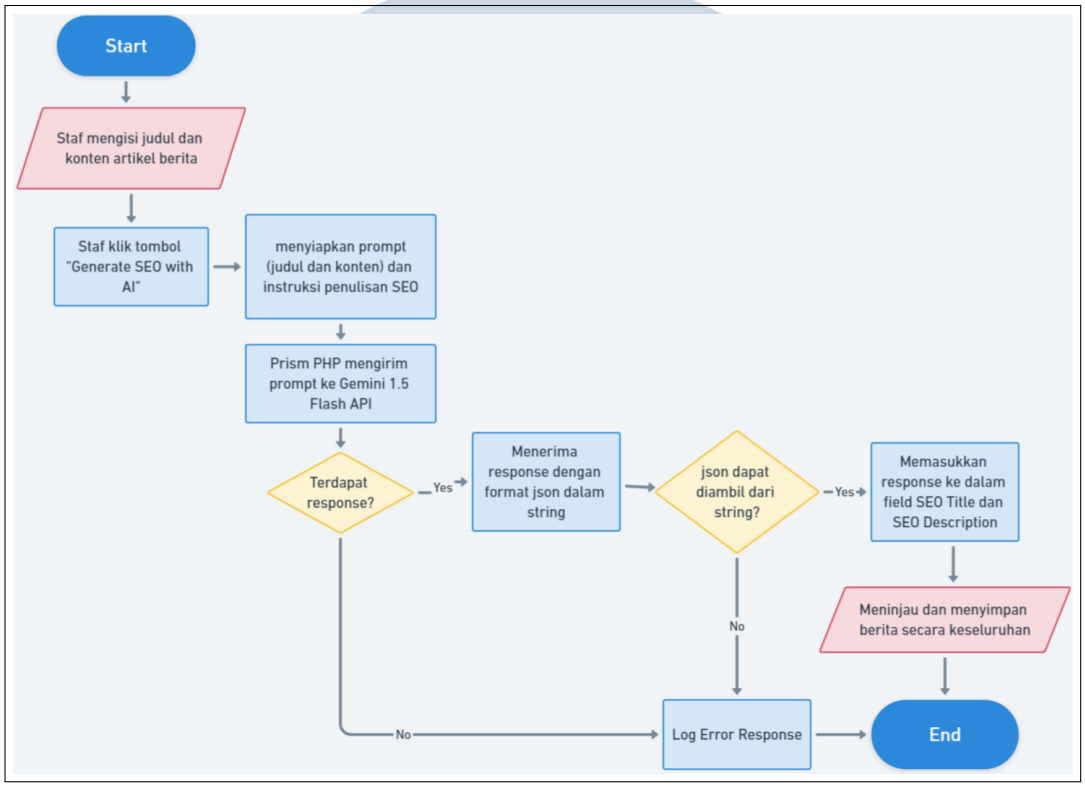
Untuk menambahkan atau mengedit video YouTube, cukup dengan memasukkan tautan URL dari video yang diinginkan. Sebuah pratinjau akan muncul untuk memastikan bahwa URL yang dimasukkan sesuai dengan konten yang diharapkan.

3.3.5 Pengembangan Fitur Rekomendasi SEO Menggunakan AI

Fitur rekomendasi SEO menjadi salah satu fitur utama yang dikembangkan selama masa magang. Melalui fitur ini, dapat dipelajari bagaimana kecerdasan buatan (AI) dapat diintegrasikan ke dalam produk untuk mempermudah kinerja pengguna. Dengan menggunakan AI, proses pembuatan SEO dapat dilakukan secara otomatis.

Penggunaan AI dapat menghemat biaya dan waktu, serta meningkatkan akurasi pembuatan SEO [12]. Automasi AI memperlancar proses pembuatan konten dengan mengurus pekerjaan yang memerlukan waktu. Dalam konteks ini, AI dapat digunakan untuk mengambil dan menganalisis data atau isi berita yang diberikan,

kemudian menghasilkan kesimpulan yang akurat. Menggunakan *prompt* yang terstruktur, AI dapat dikonfigurasikan untuk memproses data dengan lebih optimal.



Gambar 3.15. Flowchart pembuatan SEO

Prism PHP merupakan sebuah *open-source library* berbasis bahasa pemrograman PHP yang digunakan untuk mengintegrasi *model* AI ke dalam *web application*. Dalam konteks ini, Prism PHP menghubungkan aplikasi Laravel dengan *model Gemini 1.5 Flash* oleh Google untuk menghasilkan judul dan deskripsi yang teroptimasi untuk SEO. Alur proses dapat dilihat pada Gambar 3.15.

Integrasi AI untuk membuat judul dan deskripsi SEO berjalan dengan baik dan sesuai dengan tujuan yang direncanakan. Setelah fitur otomasi metadata diintegrasikan, AI mampu menghasilkan judul dan deskripsi yang menarik dan sesuai dengan isi artikel. Hasil juga memenuhi standar panjang dan struktur yang digunakan dalam SEO, yaitu maksimal 60 karakter untuk judul dan 160 karakter untuk deskripsi.

Selama proses pengembangan, penyesuaian *prompt* menjadi tantangan yang perlu ditelusuri. *Prompt* adalah kunci dari konfigurasi hasil yang diberikan oleh AI. Seperti yang telah disebutkan sebelumnya, *prompt* yang terstruktur dapat menghasilkan hasil yang optimal.

Proses perancangan ini disebut *Prompt Engineering*, yaitu pendekatan teknis dalam menyusun instruksi yang ditujukan untuk *Large Language Model* seperti Gemini, ChatGPT, atau Claude, untuk menghasilkan respons yang sesuai dengan kebutuhan spesifik pengguna. *Prompt Engineering* secara umum menggunakan tiga prinsip utama, yaitu penokohan (*Role Assignment*) untuk menetapkan peran AI sebagai ahli pada bidang tertentu, pemberian instruksi yang jelas (*Instructional Prompting*) untuk membatasi jenis dan cakupan respons yang dihasilkan, dan pembatasan hasil (*Output Constraining*) melalui penetapan format dan panjang karakter yang ditentukan [13].

A *Role Assignment*

Langkah pertama dalam *Prompt Engineering* adalah penokohan atau *Role Assignment* yang akan digunakan oleh *model* AI. Dengan prompt "*You are an expert SEO analyst*", ditetapkan tokoh yang digunakan oleh *model* sebagai seorang *expert SEO analyst*. Penokohan ini bertujuan untuk memberikan gaya berpikir yang profesional, menghasilkan jawaban yang selaras dengan praktik dan standar kerja seorang analis SEO. Dengan menetapkan peran ini sejak awal, *model* diinstruksikan untuk memberikan respons yang spesifik dan berfokus pada *Search Engine Optimization*. Penokohan juga memastikan gaya bahasa dan struktur jawaban tetap konsisten dengan peran yang ditetapkan.

B *Instructional Prompting*

Prompt dilanjutkan dengan menetapkan instruksi yang jelas mengenai tugas yang harus dilakukan oleh *model* AI. Instruksi yang digunakan adalah "*Your primary task is to generate SEO metadata for the provided article content*," dan "*You MUST output your response *only* as a single, valid JSON object*." Instruksi tersebut ditulis secara langsung dan tegas untuk menghindari interpretasi bebas oleh *model*. Dengan menggunakan batasan seperti *MUST*, *model* diberi batasan keras untuk tidak menambahkan teks, penjelasan, atau format yang tidak diperlukan. Batasan diperkuat dengan tambahan instruksi "*Do not include any text, explanations, or markdown formatting like "'JSON "' before or after the JSON object*." Pendekatan ini memungkinkan hasil dari *model* dapat digunakan langsung oleh sistem backend tanpa perlu diproses ulang. Instruksi yang jelas dan terarah merupakan kunci dalam prompt engineering untuk menghasilkan keluaran yang

sesuai kebutuhan.

C *Output Constraining*

Prinsip terakhir adalah pembatasan terhadap bentuk dan isi dari *output* yang dihasilkan. Pada *prompt* ini, struktur `JSON` yang dihasilkan telah ditentukan secara detail seperti berikut:

```
1
2    {
3        "seo_title": "string (concise, under 60 characters)",
4        "meta_description": "string (engaging, under 160
     characters)"
5    }
```

Kode 3.1: *Output* dengan format `JSON`

Data yang dikembalikan oleh *model* AI telah ditetapkan gaya penulisan dan maksimal panjang karakternya. Pada Kode 3.1, judul SEO telah ditetapkan untuk memiliki batas panjang karakter sebanyak 60 dengan gaya penulisan yang ringkas, dan deskripsi yang menarik dengan panjang maksimal 160 karakter. Selain gaya penulisan dan batasan jumlah karakter, *prompt* ditambahkan pembatas dengan menghasilkan respons dalam bahasa Indonesia. Pembatasan format ini disebut sebagai *structured output prompting*, yang bertujuan agar *model* menghasilkan respons dengan struktur tetap, konsisten, dan mudah diproses secara otomatis oleh sistem. Dengan cara ini, integrasi AI dalam sistem metadata menjadi efisien dan dapat diandalkan.

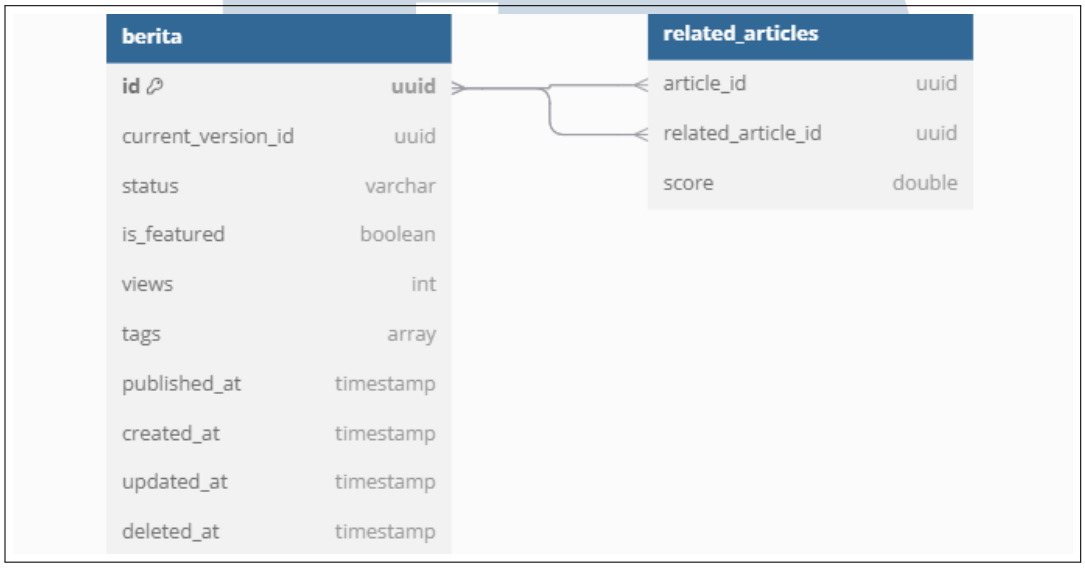
3.3.6 Pengembangan Fitur Berita Terkait

Dalam berita, akan selalu terdapat berita-berita yang dapat dikaitkan. Tujuan dari fitur ini adalah untuk menampilkan beberapa berita yang memiliki keterkaitan topik dengan berita yang sedang dibaca. Melalui fitur ini, pembaca atau pengunjung dapat diarahkan menuju berita-berita lain, sehingga meningkatkan *retention rate* situs. Fitur ini dapat membantu membangun struktur tautan yang baik, yang bermanfaat untuk SEO.

Pengelolaan berita terkait menggunakan komponen bawaan *Relation Manager*. Komponen ini memungkinkan staf Dinas Perhubungan Kota Tangerang Selatan untuk menambahkan atau mengeluarkan berita dari daftar berita terkait dengan mudah. Gambar 3.8, Staf dapat melihat berita-berita yang memiliki kaitan

dan menghapus keterkaitan tersebut dengan menekan tombol ”*Detach*”. Untuk menambahkan berita pada daftar berita terkait, staf dapat menekan tombol biru dengan tulisan ”*Select Related Articles*” untuk memunculkan modal pada Gambar 3.9. Staf dapat mencari berita berdasarkan judul atau berdasarkan *tags*.

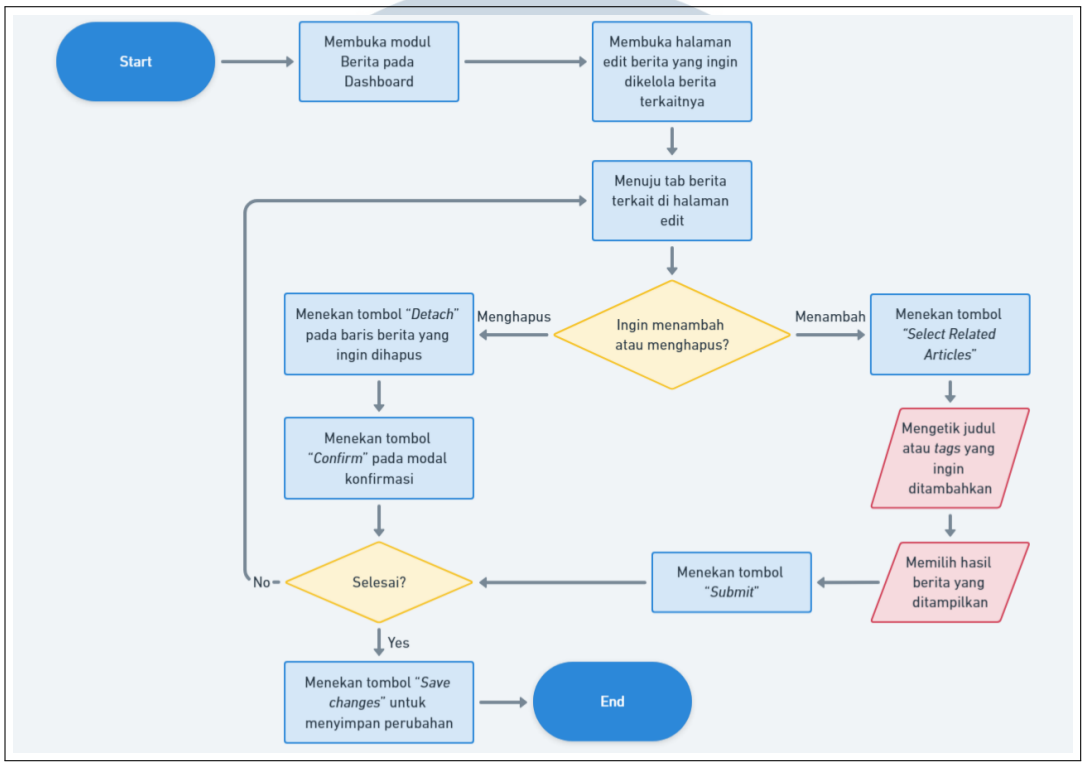
A Diagram Struktur Data



Gambar 3.16. *Entity Relation Diagram* untuk berita terkait

Untuk mendukung fitur berita terkait, diperlukan desain struktur data yang fleksibel agar setiap berita dapat dikaitkan dengan berita lainnya. Dalam implementasinya, digunakan pendekatan relasi *Many-to-Many* pada entitas berita, yang mana hubungan antar artikel disimpan pada *pivot table* bernama `related_articles`. Dengan desain ini, semua berita dapat memiliki banyak berita terkait, dan sebaliknya. Struktur ini juga memungkinkan pengelolaan relasi yang dinamis tanpa perlu melakukan modifikasi pada tabel utama. Gambar 3.16 menampilkan diagram relasi antar entitas dalam sistem yang digunakan untuk mendukung fitur ini.

B Diagram Alur Pengelolaan



Gambar 3.17. *Flowchart* untuk pengelolaan berita terkait

Pengelolaan berita terkait dilakukan langsung oleh staf melalui halaman edit berita. Pada halaman edit berita, staf dapat melihat berita-berita terkait pada bagian *Related Articles* yang berada di bawah *form* edit berita. Terlihat sebuah tabel yang menunjukkan data seperti judul dan status dari berita tersebut. Staf dapat menghapus keterkaitan berita dengan menekan tombol "*Detach*" dan menekan tombol "*Confirm*" pada modal konfirmasi yang muncul. Untuk menambahkan artikel, staf perlu menekan tombol biru "*Select Related Articles*", kemudian mencari artikel berita berdasarkan judul atau *tags* yang akan memunculkan hasil pencarian yang dapat dipilih. Untuk menyimpan perubahan, staf perlu menekan tombol "*Submit*" kemudian tombol "*Save changes*". Gambar 3.17 menunjukkan alur proses ini.

C Pengembangan Lebih Lanjut

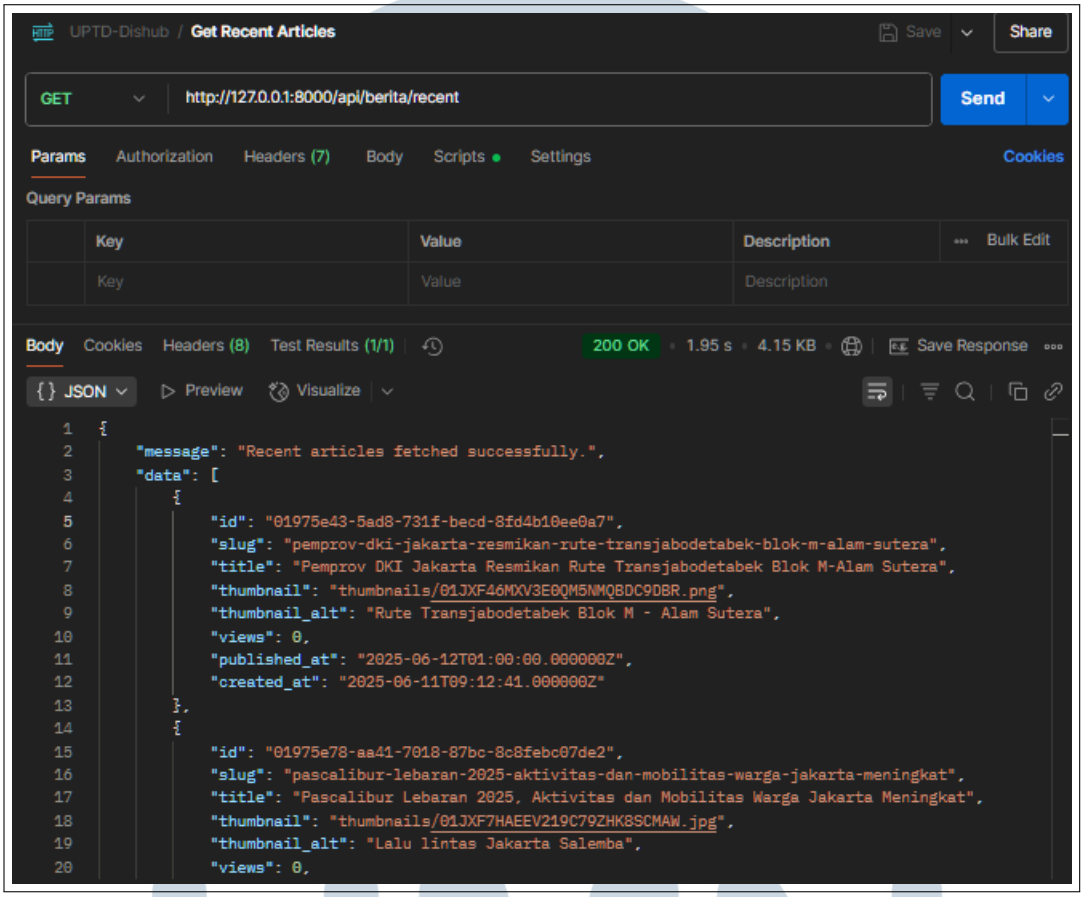
Secara keseluruhan, fitur ini berhasil diterapkan sesuai dengan rencana. Namun, fitur ini dapat dikembangkan lebih lanjut dengan penerapan *automation*

pemilihan berita menggunakan pendekatan berdasarkan kemiripan konten. Salah satu metode yang dapat diterapkan adalah penggunaan algoritma seperti *cosine similarity* atau *Term Frequency and Inverse Document Frequency*, dapat menghitung seberapa mirip suatu artikel dengan artikel lain. Menggunakan algoritma tersebut, sistem dapat memberikan rekomendasi berita-berita berdasarkan kemiripan yang dapat ditetapkan sebagai berita terkait.

3.3.7 Pengembangan API Berita

Dalam pengembangan sistem informasi berita, salah satu komponen penting adalah penyediaan API (*Application Programming Interface*) yang dapat diakses oleh *frontend* untuk menampilkan data secara dinamis. API ini dikembangkan menggunakan Laravel sebagai *backend framework*, dengan pendekatan RESTful API. Terdapat tiga jenis API utama yang dikembangkan dalam proyek ini, yaitu *recent news*, *featured news*, dan *related news*.

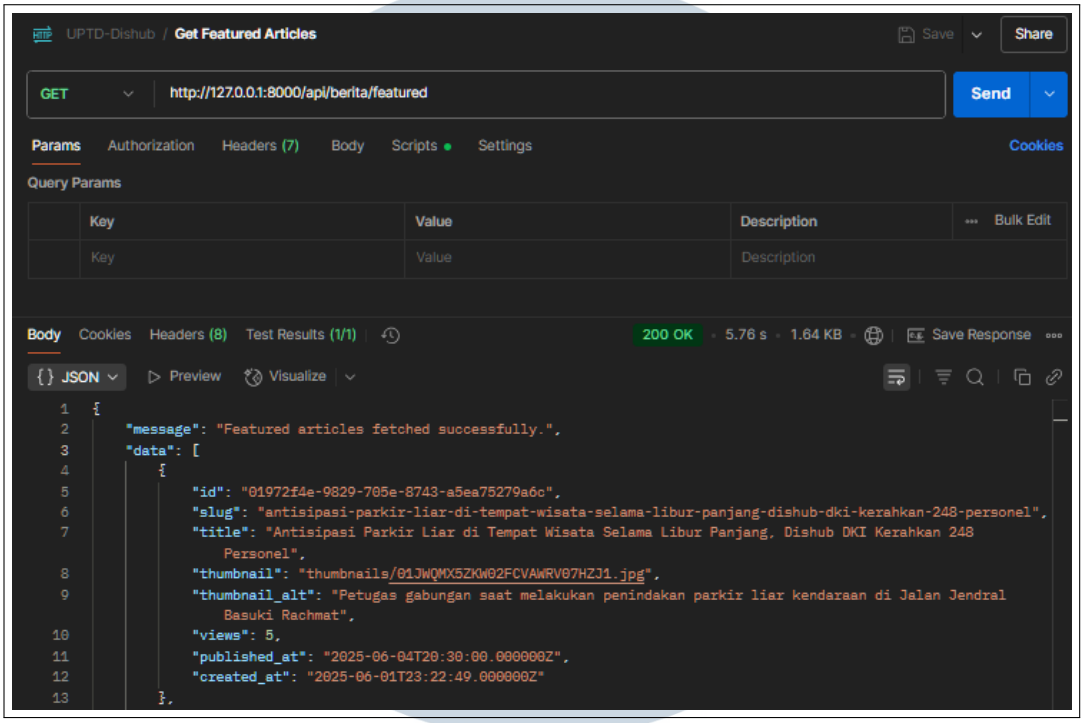
A Dokumentasi API *Recent News*



Gambar 3.18. Dokumentasi *endpoint* API berita terbaru

Endpoint ini bertugas menyediakan daftar berita terbaru yang telah dipublikasikan, diurutkan berdasarkan waktu publikasi dari yang paling baru. Secara teknis, API ini menggunakan metode GET dan tidak memerlukan parameter tambahan. Hasil yang dikembalikan berupa data dalam format JSON yang telah diserialisasi, mencakup beberapa field penting seperti `id`, `title`, `slug`, `thumbnail_url`, dan `published_at`. API ini umumnya digunakan pada halaman utama (*homepage*) situs web untuk menampilkan berita terkini. Dengan adanya endpoint ini, pengguna dapat dengan mudah mengakses berita-berita terkini tanpa harus menelusuri keseluruhan arsip artikel.

B Dokumentasi API *Featured News*

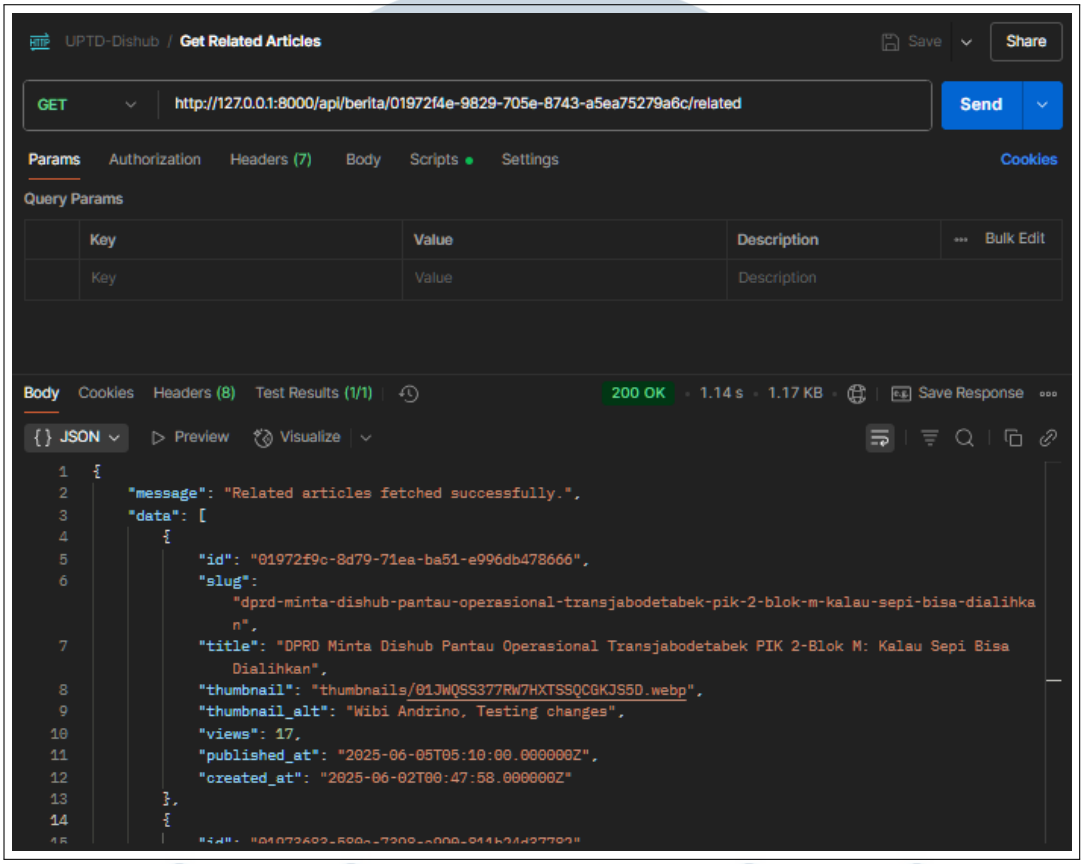


Gambar 3.19. Dokumentasi *endpoint* API berita unggulan

Endpoint featured news digunakan untuk mengambil daftar artikel yang telah ditandai sebagai berita unggulan oleh administrator. Berita unggulan diseleksi secara manual oleh administrator dan ditandai dalam sistem menggunakan atribut khusus. *Endpoint featured news* mengembalikan artikel-artikel yang memiliki atribut `is_featured = true`, dan hanya menampilkan artikel yang telah dipublikasikan.

Sama seperti sebelumnya, API ini menggunakan metode GET dan menghasilkan respons JSON. Data *featured* ini dalam prakteknya ditempatkan pada bagian *hero* atau *banner* halaman utama, untuk menyorot informasi penting atau berita besar yang ingin ditampilkan lebih tersorot.

C Dokumentasi API *Related News*



Gambar 3.20. Dokumentasi *endpoint* API berita terkait

API ini digunakan untuk mengambil daftar berita yang terkait dengan sebuah artikel tertentu. *Endpoint* ini menggunakan parameter berupa `id` dari artikel utama, kemudian *backend* melakukan query terhadap tabel *pivot* yang menyimpan relasi *many-to-many* antar berita. Proses pencarian dilakukan secara efisien melalui *eager loading* dan relasi *database* yang telah dioptimalkan. Hasil yang dikembalikan akan berisi daftar berita terkait dengan struktur yang sama seperti *endpoint* lainnya. Contoh respons dapat dilihat pada Gambar 3.20. Data ini ditampilkan pada bagian bawah atau kanan halaman artikel, untuk mendorong pengunjung melanjutkan membaca konten lain yang relevan.

D Pengembangan Lebih Lanjut

Seiring dengan kebutuhan sistem yang terus berkembang, endpoint API untuk berita unggulan, berita terbaru, dan berita terkait masih memiliki ruang

untuk dikembangkan lebih lanjut. Salah satu pengembangan yang dapat diterapkan adalah penambahan fitur *pagination* dan *filtering* dinamis. Dengan menambahkan parameter seperti `page`, `limit`, `category`, atau `tags`, API akan menjadi lebih fleksibel dan efisien, terutama ketika diakses oleh aplikasi *frontend* atau pihak ketiga yang membutuhkan kontrol lebih atas data yang ditampilkan.

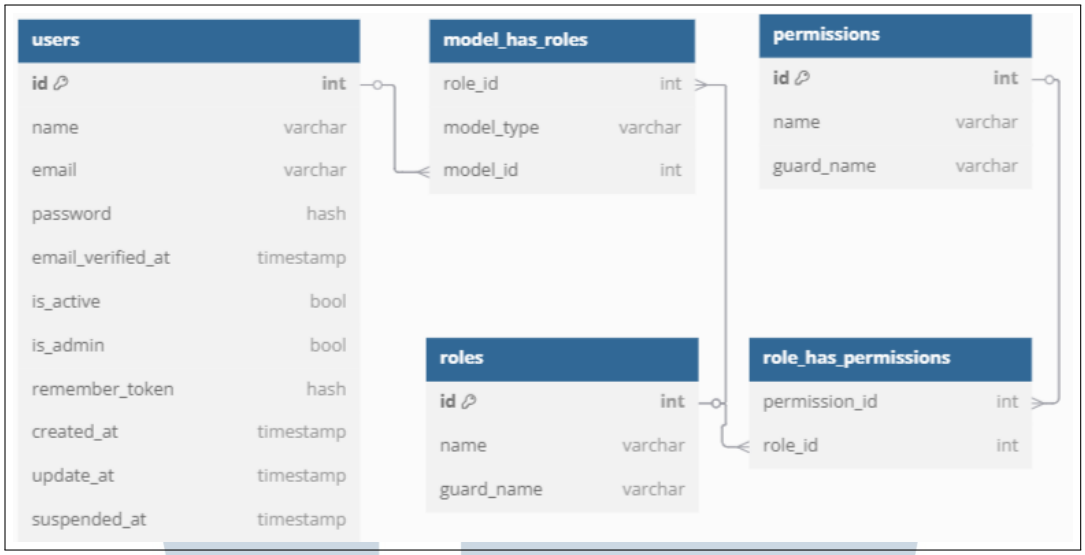
Selain itu, aspek keamanan dan manajemen akses juga penting untuk dikembangkan. Penerapan autentikasi menggunakan `API token` atau `JWT token` dapat membatasi akses terhadap *endpoint* tertentu yang bersifat internal atau sensitif. Pengembangan lain yang tak kalah penting adalah penerapan sistem *logging* untuk mencatat aktivitas penggunaan API. Dengan begitu, tim pengelola dapat melakukan pemantauan, evaluasi performa, serta analisis terhadap pola akses dan popularitas artikel.

Terakhir, sistem dapat menyediakan alternatif format output seperti RSS (*Really Simple Syndication*) atau `JSON Feed` untuk keperluan integrasi dengan platform lain, seperti aplikasi mobile, newsletter, atau mitra media. Dengan berbagai pengembangan ini, API yang telah dibangun tidak hanya mampu memenuhi kebutuhan saat ini, tetapi juga siap digunakan dalam skenario sistem yang lebih kompleks dan dinamis di masa mendatang.

3.3.8 Pengembangan Modul *User* dengan Manajemen *Role* dan *Permission*

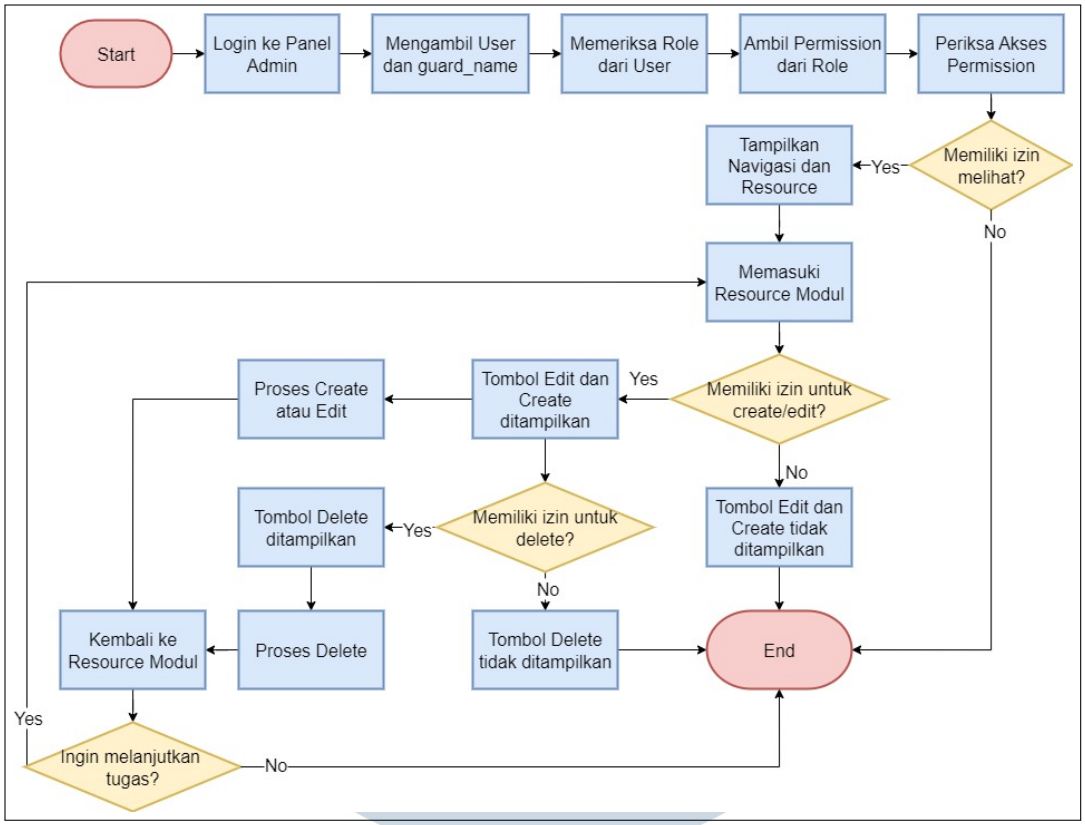
Pengembangan modul manajemen pengguna merupakan bagian krusial dari sistem CMS yang sedang dikembangkan. Modul ini tidak hanya mencakup fitur CRUD (*Create, Read, Update, Delete*) untuk pengguna, tetapi juga pengaturan akses melalui sistem *Role-Based Access Control* (RBAC). Sistem RBAC yang diterapkan menggunakan kombinasi Laravel, package Spatie Laravel Permission, serta integrasi visual antarmuka panel admin menggunakan Filament.

Untuk menggambarkan hubungan antar entitas utama yang digunakan dalam pengelolaan akses ini, Gambar 3.21 disajikan *Entity Relationship Diagram* (ERD) yang menunjukkan relasi antara `users`, `roles`, dan `permissions` dalam konteks RBAC.



Gambar 3.21. Diagram relasi entitas Sistem RBAC

Setelah memahami struktur data melalui ERD, penting untuk menjelaskan bagaimana sistem RBAC bekerja secara proses. Flowchart Gambar 3.22 menggambarkan alur keputusan sistem dalam mengatur hak akses pengguna berdasarkan peran (*role*) dan izin akses (*permission*) yang dimilikinya. Proses ini terjadi ketika pengguna mencoba mengakses suatu fitur atau melakukan suatu aksi di dalam sistem. Sistem akan memeriksa apakah pengguna memiliki peran, lalu mengecek apakah peran tersebut memiliki izin akses yang dibutuhkan untuk aksi tersebut.

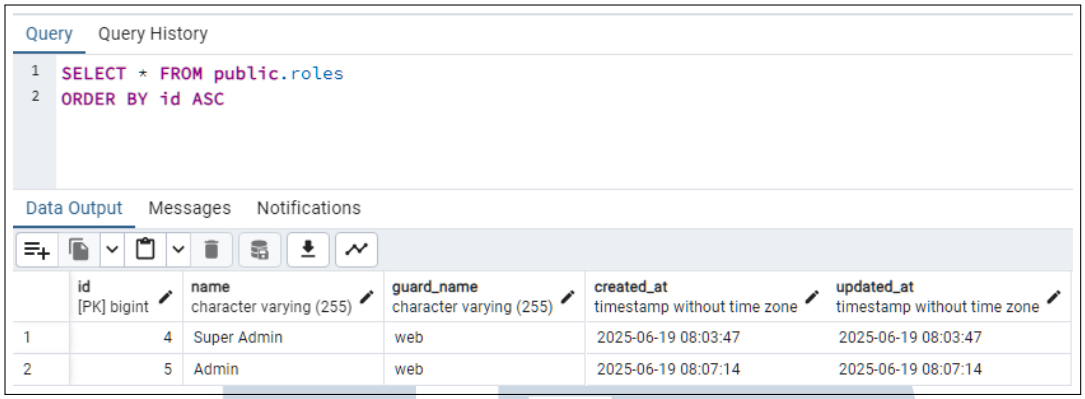


Gambar 3.22. Flowchart pengecekan *role* dan *permission*

A Proses Pengembangan

A.1 *Install* dan konfigurasi *library* Spatie Laravel Permission

Pengembangan dimulai dengan pemasangan *library* dari spatie/laravel-permission yang menyediakan fitur *Role* dan *Permission*. Dari *library* ini, terdapat migrasi yang perlu dipublikasikan ke dalam proyek untuk pendukung implementasi *library*. Melalui *library* ini, dibuatkan tabel `roles`, `permissions`, `model_has_permissions`, `model_has_roles`, dan `role_has_permissions`.



Query | Query History

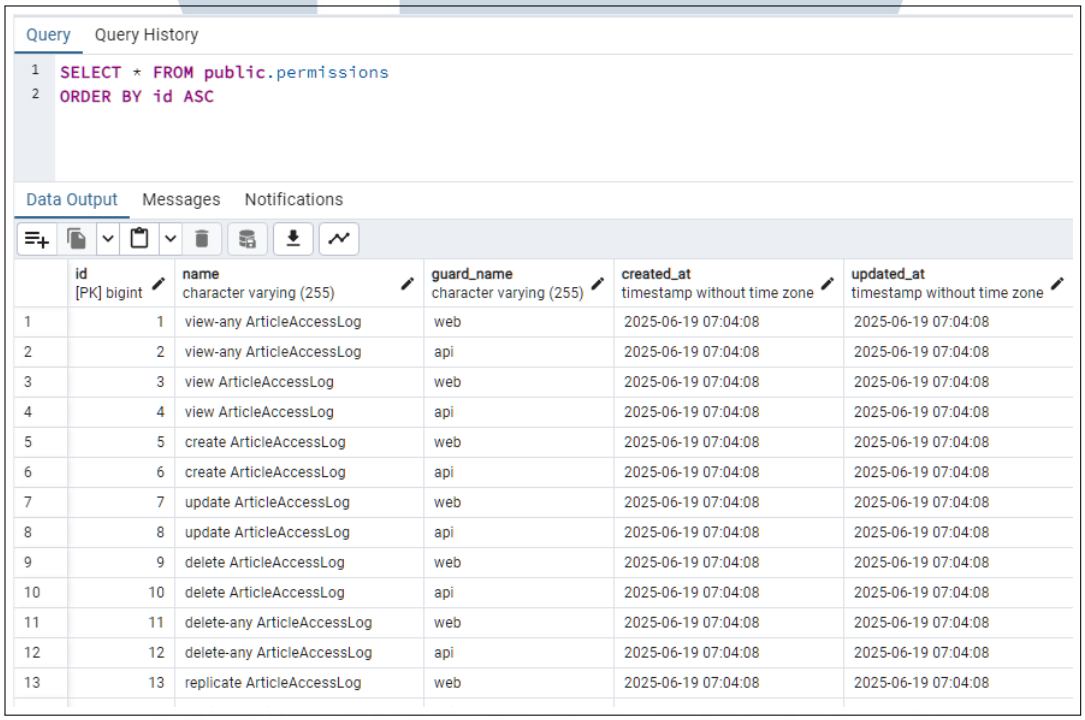
```
SELECT * FROM public.roles
ORDER BY id ASC
```

Data Output | Messages | Notifications

	id [PK] bigint	name character varying (255)	guard_name character varying (255)	created_at timestamp without time zone	updated_at timestamp without time zone
1	4	Super Admin	web	2025-06-19 08:03:47	2025-06-19 08:03:47
2	5	Admin	web	2025-06-19 08:07:14	2025-06-19 08:07:14

Gambar 3.23. Tabel Roles pada database

Tabel `roles` pada Gambar 3.23 berfungsi untuk menyimpan daftar peran yang dapat diberikan kepada pengguna.



Query | Query History

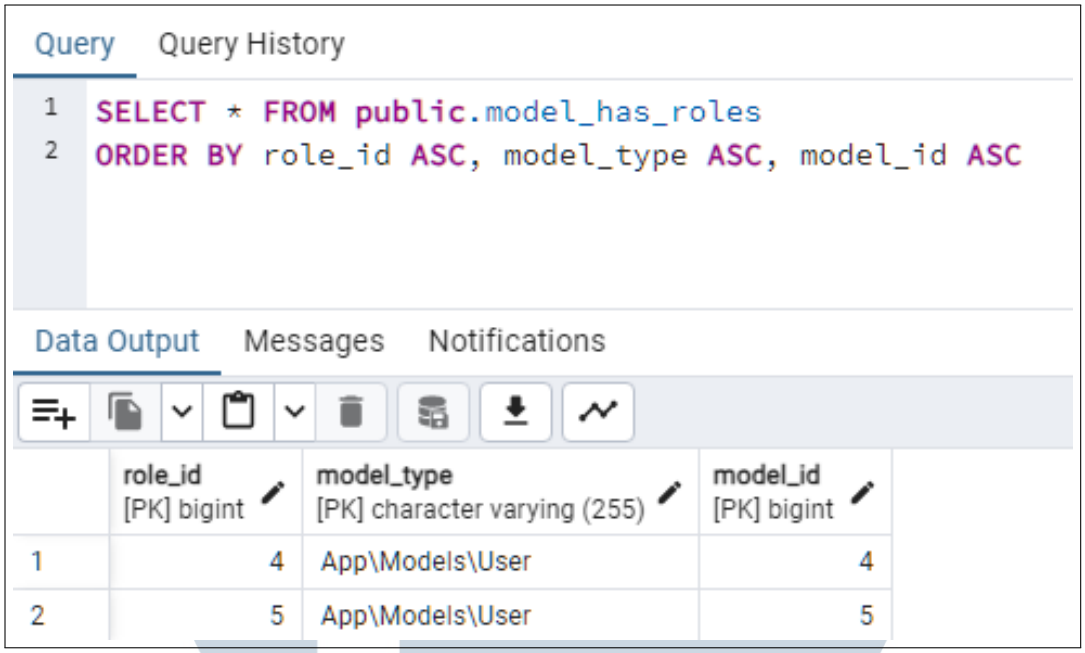
```
SELECT * FROM public.permissions
ORDER BY id ASC
```

Data Output | Messages | Notifications

	id [PK] bigint	name character varying (255)	guard_name character varying (255)	created_at timestamp without time zone	updated_at timestamp without time zone
1	1	view-any ArticleAccessLog	web	2025-06-19 07:04:08	2025-06-19 07:04:08
2	2	view-any ArticleAccessLog	api	2025-06-19 07:04:08	2025-06-19 07:04:08
3	3	view ArticleAccessLog	web	2025-06-19 07:04:08	2025-06-19 07:04:08
4	4	view ArticleAccessLog	api	2025-06-19 07:04:08	2025-06-19 07:04:08
5	5	create ArticleAccessLog	web	2025-06-19 07:04:08	2025-06-19 07:04:08
6	6	create ArticleAccessLog	api	2025-06-19 07:04:08	2025-06-19 07:04:08
7	7	update ArticleAccessLog	web	2025-06-19 07:04:08	2025-06-19 07:04:08
8	8	update ArticleAccessLog	api	2025-06-19 07:04:08	2025-06-19 07:04:08
9	9	delete ArticleAccessLog	web	2025-06-19 07:04:08	2025-06-19 07:04:08
10	10	delete ArticleAccessLog	api	2025-06-19 07:04:08	2025-06-19 07:04:08
11	11	delete-any ArticleAccessLog	web	2025-06-19 07:04:08	2025-06-19 07:04:08
12	12	delete-any ArticleAccessLog	api	2025-06-19 07:04:08	2025-06-19 07:04:08
13	13	replicate ArticleAccessLog	web	2025-06-19 07:04:08	2025-06-19 07:04:08

Gambar 3.24. Tabel Permissions pada database

Tabel `permissions` pada Gambar 3.24 digunakan untuk menjabarkan izin yang diperlukan untuk berinteraksi dalam panel admin. Ini menampung izin untuk membuat, menghapus, membaca atau mengubah data baik dalam bentuk web atau API.



Gambar 3.25. Tabel acuan `model_has_roles` pada database

Tabel `model_has_roles` pada Gambar 3.25 berfungsi sebagai *pivot* yang menjadi acuan sistem terhadap hubungan *Many-to-Many*, yang mana model `User` dapat memiliki lebih dari satu hak akses dan hak akses dapat diberikan kepada banyak pengguna.

Query Query History

```
1  SELECT * FROM public.role_has_permissions
2  ORDER BY permission_id ASC, role_id ASC
```

Data Output Messages Notifications

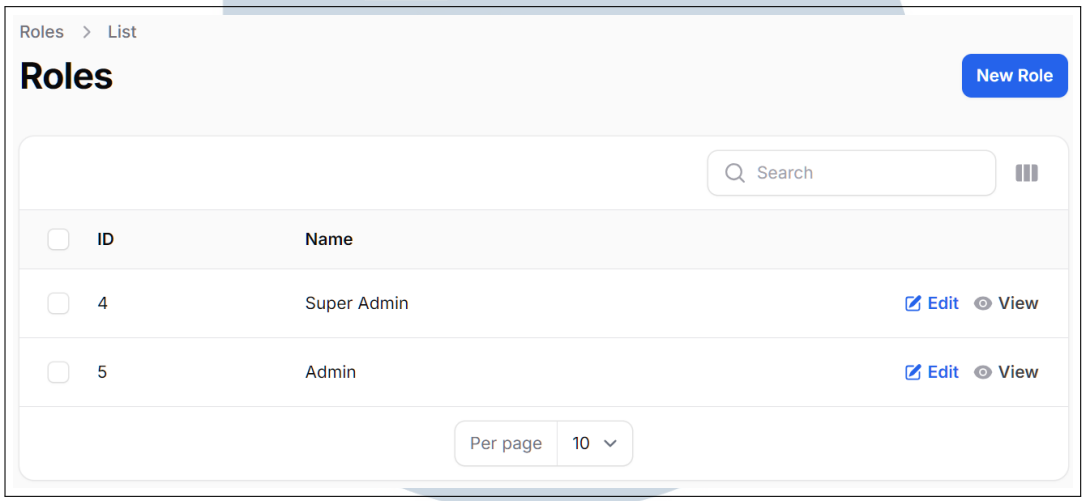
	permission_id [PK] bigint	role_id [PK] bigint
1	1	4
2	3	4
3	5	4
4	7	4
5	9	4
6	11	4
7	13	4
8	15	4
9	17	4
10	19	4
11	21	4
12	23	4
13	25	4

Gambar 3.26. Tabel acuan role_has_permissions pada database

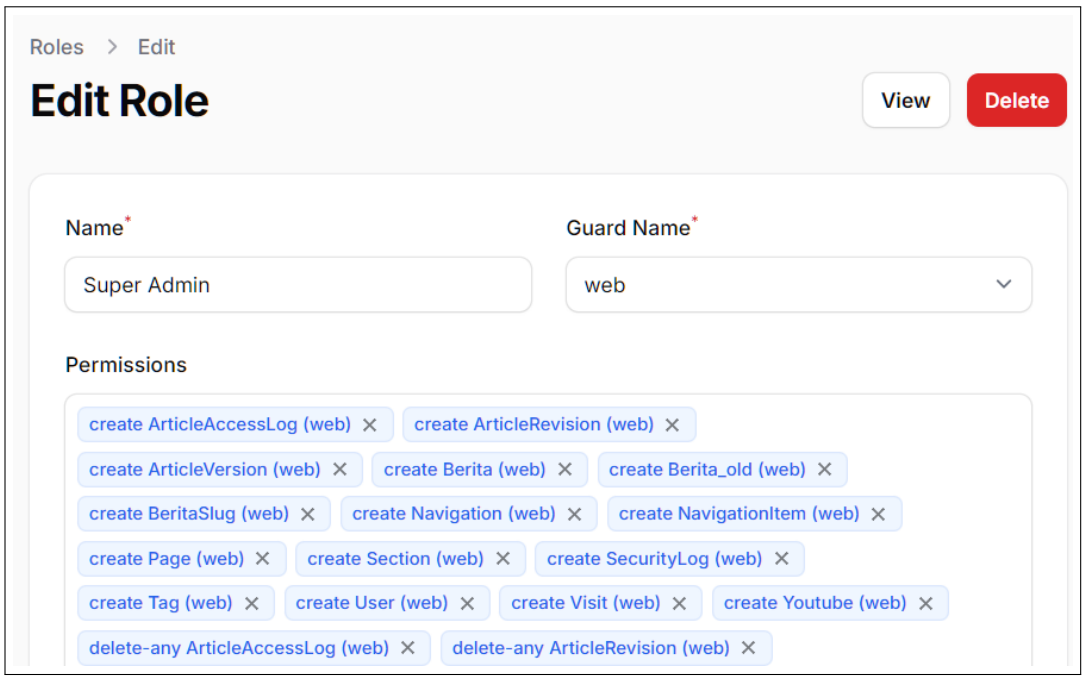
Tabel role_has_permissions pada Gambar 3.26 dibuat untuk menjadi acuan relasi *Many-to-Many* antara role dan permission.

A.2 Integrasi pada Filament

Untuk antarmuka pada panel admin, digunakan plugin `Filament Spatie Role Permissions`. Namun, karena sistem ini memerlukan fleksibilitas yang lebih tinggi, plugin tersebut hanya dikonfigurasi untuk fitur dasar. Sementara itu, `Resource` untuk `User`, `Role`, dan `Permission` dikembangkan secara manual dengan menonaktifkan *resource* bawaan dari plugin.



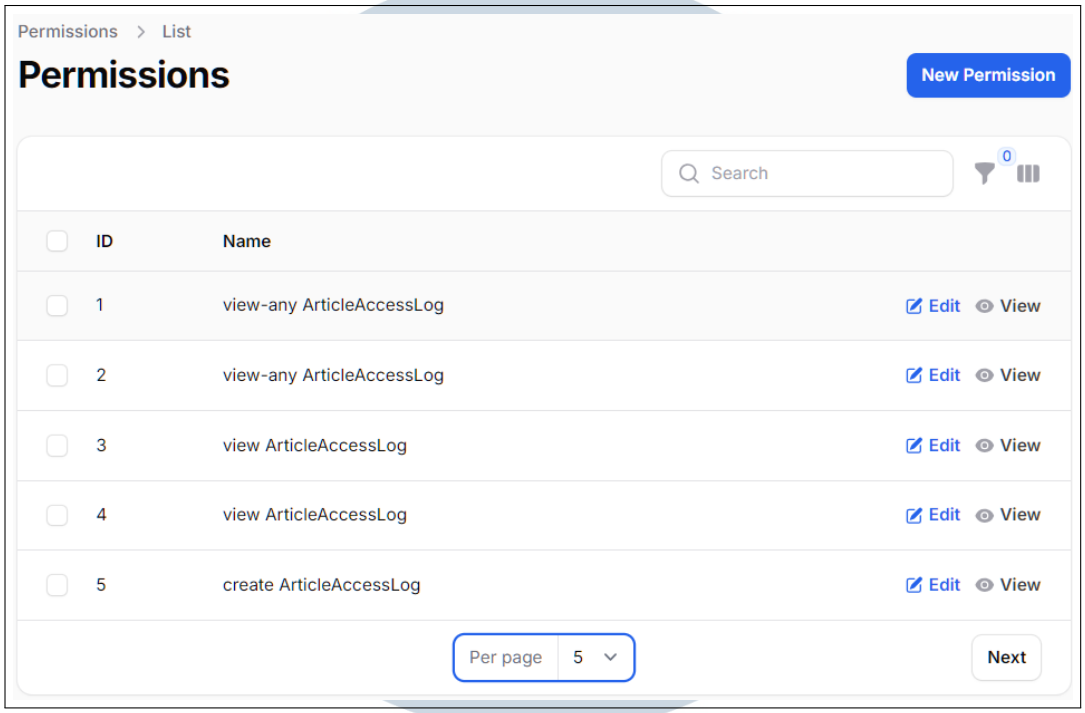
Gambar 3.27. Tabel roles pada panel admin



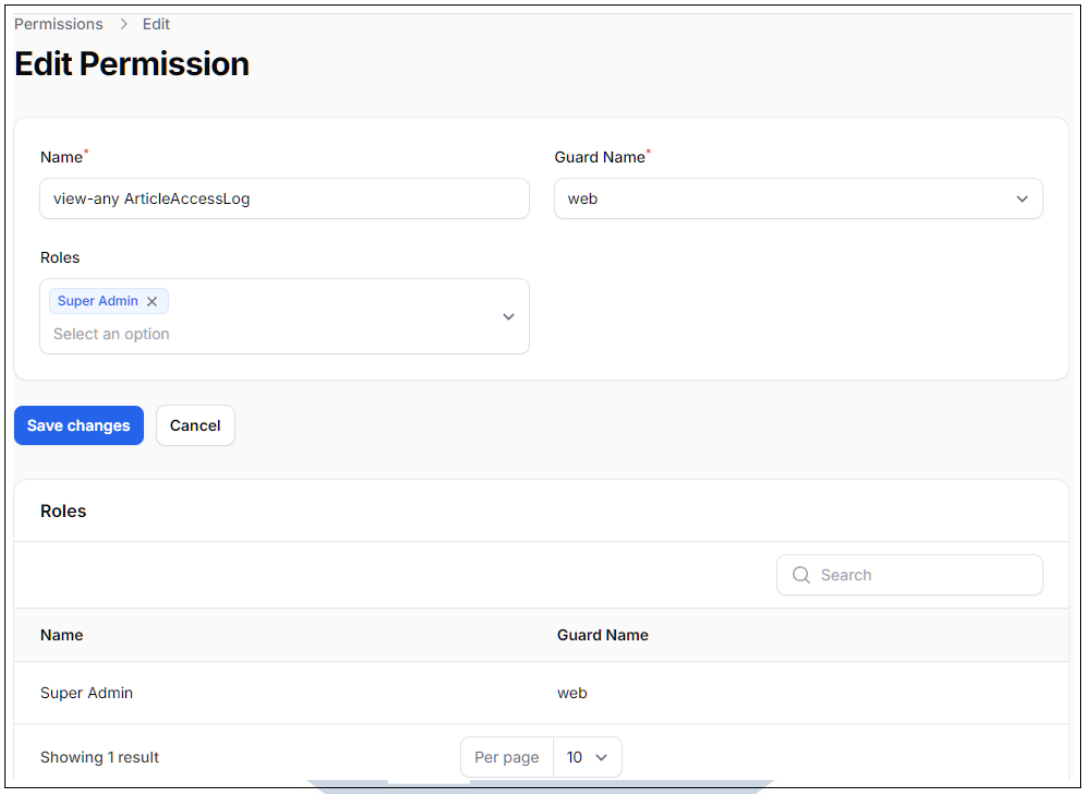
Gambar 3.28. Form roles pada panel admin

Tampilan untuk *resource role* dapat dilihat pada Gambar 3.27, serta *form*

untuk pembuatan dan perubahan data dapat dilihat pada Gambar 3.28. Staf dapat mengatur izin-izin yang diperbolehkan pada suatu *role*.



Gambar 3.29. Tabel permissions pada panel admin



Gambar 3.30. Form permissions pada panel admin

Untuk *resource permissions*, staf dapat mengelola izin-izin yang tersedia dalam panel admin. Tabel *permissions* panel admin dapat dilihat pada Gambar 3.29. Kemudian, *form* untuk pembuatan dan perubahan data dapat dilihat pada Gambar 3.30. Pada bagian ini, staf, khususnya yang memegang hak akses sebagai Super Admin, dapat melihat dan mengatur izin-izin, serta melihat pengguna-pengguna yang memiliki izin tersebut.

A.3 Kontrol Akses pada *Resource* Filament

Setiap *Resource* dalam proyek, secara bawaan akan diberikan metode otorisasi berbasis *permission*. Proses ini memastikan bahwa *Resource* hanya dapat diakses jika memiliki izin untuk melihat, menambahkan, mengubah, atau menghapus data. Ini merupakan fungsi utama dari *roles* dan *permissions*, untuk membatasi akses dan meminimalkan potensi penyalahgunaan akun.

Untuk menerapkan izin otoritas ini, digunakan `Gate` sebagai penjaga akses pada *resource*. Contoh penerapan dapat dilihat pada Kode 3.2.

```
1
2    class PermissionResource extends BasePermissionResource
```

```
    {
        public static function canViewAny(): bool
        {
            return Gate::allows('manage permissions');
        }
    }
```

Kode 3.2: Penerapan Gate pada *Resource* Filament

A.4 Kendala Pengembangan

Beberapa kendala yang terjadi selama pengembangan adalah masalah *guard* yang tidak sesuai. Saat awal pengembangan, muncul error `GuardDoesNotMatch`. Ini terjadi karena *guard* yang digunakan oleh role Super Admin adalah `web`, sedangkan beberapa permission memiliki *guard* `api`.

Guard merupakan sebuah mekanisme yang menentukan bagaimana pengguna diautentikasi dan dari mana asal mereka. *Guard* pada sistem autentikasi Laravel menentukan bagaimana pengguna diautentikasi, apakah melalui `session`, `token`, atau permintaan API. *Guard* juga menentukan sumber model dan driver autentikasi pengguna. Secara bawaan, Laravel telah dikonfigurasikan untuk menggunakan *guard* `web`.

Saat menggunakan `Spatie Laravel Permission`, setiap `role` dan `permission` harus dikaitkan dengan `guard` tertentu. Akibatnya, sebuah `role` yang memiliki `guard web` tidak dapat mengakses `permission` yang memiliki `guard api`.

Untuk mengatasi masalah tersebut, dilakukan penelaahan terhadap `guard` yang digunakan oleh `User`, `Role`, dan `Permission`. Penelaahan ini bertujuan untuk menetapkan `guard` yang konsisten pada panel admin. Berdasarkan hasil evaluasi, diputuskan untuk menggunakan `guard web`, disesuaikan dengan konfigurasi aplikasi. Setelah penyesuaian dilakukan secara konsisten, error `GuardDoesNotMatch` tidak lagi muncul, dan proses pengecekan *role* serta *permission* berjalan sesuai harapan dalam panel admin Filament.

B Pengembangan Lebih Lanjut

Meskipun sistem telah berjalan sesuai harapan, terdapat beberapa aspek yang masih dapat dikembangkan. Salah satu pengembangan yang dapat diimplementasikan adalah audit perubahan. Setiap perubahan pada `role`, `permission`, atau pengaturan hak akses pengguna perlu direkam untuk

meningkatkan keamanan dan memungkinkan penelusuran riwayat perubahan. Pengembangan lebih lanjut juga dapat mencakup integrasi *multi-tenant*, di mana `role` dan `permission` dihubungkan ke organisasi atau entitas yang berbeda. Pendekatan ini memungkinkan sistem digunakan oleh banyak organisasi dengan data dan hak akses yang terisolasi. Terakhir, jika skema *multi-tenant* diterapkan, dapat dipertimbangkan pengembangan *Per-Model Permissions*, yaitu hak akses spesifik pada entitas tertentu, misalnya pembatasan pengelolaan berita hanya untuk penulis yang bersangkutan.

3.3.9 Pengembangan Fitur Logging Authentication

Dalam sistem manajemen konten yang digunakan oleh instansi seperti Dinas Perhubungan Kota Tangerang Selatan, keamanan akses merupakan aspek yang sangat krusial. Sistem ini diakses oleh berbagai pengguna dengan peran dan tingkat otorisasi yang berbeda, seperti administrator, editor, dan staf publikasi. Tanpa pengawasan terhadap aktivitas *login* pengguna, sistem menjadi rentan terhadap berbagai bentuk penyalahgunaan akun.

Salah satu kerentanan utama adalah kemungkinan terjadinya akses tidak sah oleh pihak yang memperoleh kredensial pengguna, baik secara disengaja maupun tidak disengaja. Misalnya, jika akun staf digunakan oleh pihak yang tidak berwenang, maka seluruh aktivitas yang dilakukan atas nama akun tersebut tidak dapat dilacak secara akurat. Hal ini dapat menyebabkan manipulasi konten, penghapusan data penting, atau penyebaran informasi yang tidak semestinya, tanpa jejak digital yang memadai.

Untuk menjawab permasalahan ini, dikembangkan fitur *authentication logging* yang secara otomatis mencatat aktivitas *login* setiap pengguna ke dalam sistem. Pencatatan ini mencakup informasi *IP address*, *user agent*, waktu *login*, dan lokasi. Dengan adanya fitur ini, administrator dapat memantau setiap aktivitas *login* dan segera mengidentifikasi pola *login* yang mencurigakan, sehingga mengurangi potensi penyalahgunaan akun.

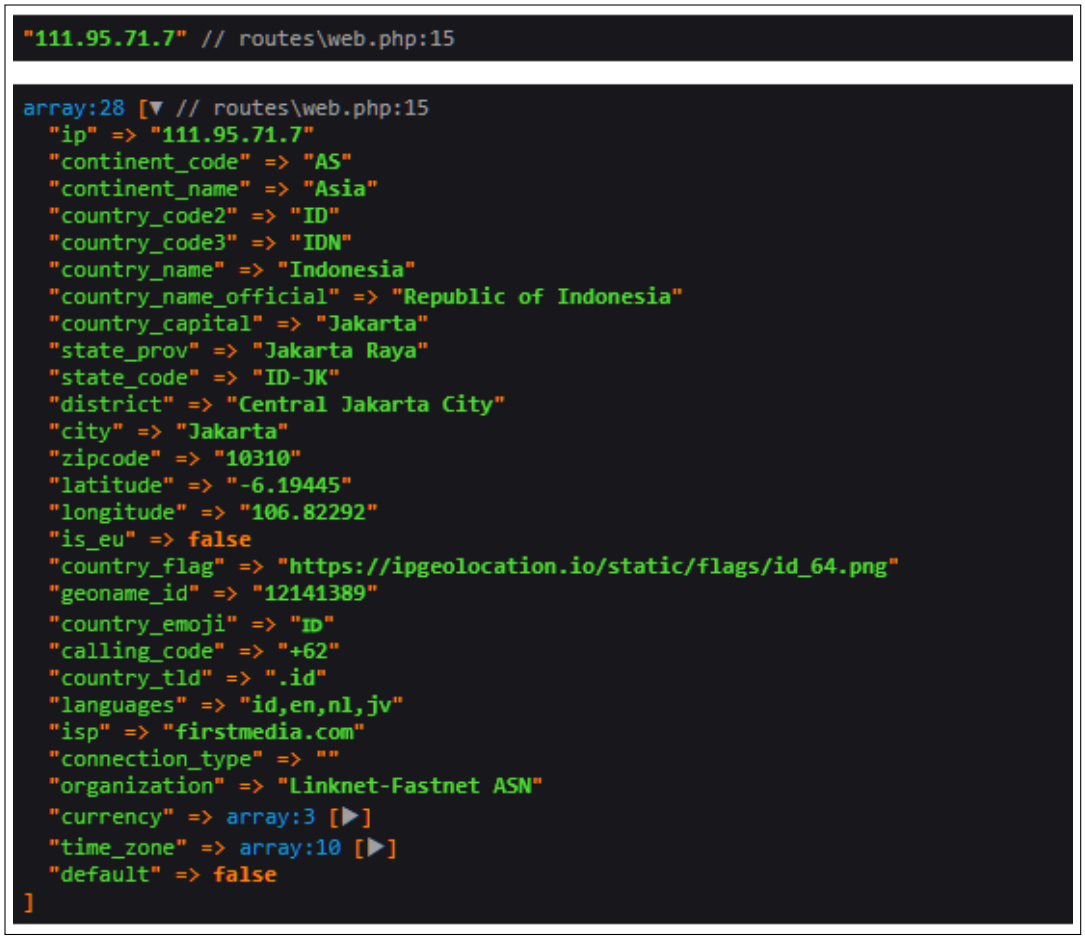
A Proses Pengembangan

Pengembangan fitur pencatatan autentikasi dilakukan secara terstruktur dengan memanfaatkan ekosistem Laravel yang telah digunakan dalam proyek. Pada tahap awal, dilakukan studi terhadap kebutuhan pencatatan aktivitas *login*

pengguna, serta peninjauan terhadap beberapa pendekatan dan *package* Laravel yang mendukung fitur tersebut. Berdasarkan hasil studi tersebut, diputuskan untuk menggunakan package pihak ketiga bernama Laravel Authentication Log oleh Rappasoft untuk melakukan pencatatan *login* secara otomatis, serta Filament Authentication Log oleh Tapp Network untuk mengintegrasikan tampilan log ke dalam panel admin Filament.

Untuk mendapatkan informasi lokasi dari alamat IP, dilakukan integrasi dengan paket GeoIP oleh Torann, yang merupakan *library* Laravel untuk melakukan geolokasi berdasarkan alamat IP. Layanan `ipgeolocation.io` digunakan sebagai penyedia data lokasi karena menyediakan informasi geografis yang cukup lengkap dan akurat, seperti kota, provinsi, negara, kode pos, ISP, hingga koordinat geografis.

Langkah konfigurasi dilakukan dengan mengganti *service* bawaan pada `config/geoip.php` menjadi `ipgeolocation`, serta menambahkan `API key` dari layanan tersebut ke dalam berkas `.env`. Selanjutnya, pengujian fungsi `geoip($ip)` dilakukan melalui Laravel Tinker untuk memastikan bahwa layanan berhasil memetakan alamat IP menjadi informasi lokasi. Contoh respons API yang diambil dari IPGeolocation dapat dilihat pada Gambar 3.31.



Gambar 3.31. Contoh hasil respon GeoIP untuk IP publik pengguna

Selanjutnya, ditambahkan kolom baru pada tampilan tabel Filament Authentication Log untuk menampilkan data lokasi berdasarkan isi kolom `location` (bertipe JSON) dari tabel `authentication_log`. Data tersebut diformat agar menampilkan kota, provinsi, dan negara pengguna dalam bentuk yang mudah dibaca.

B Tantangan

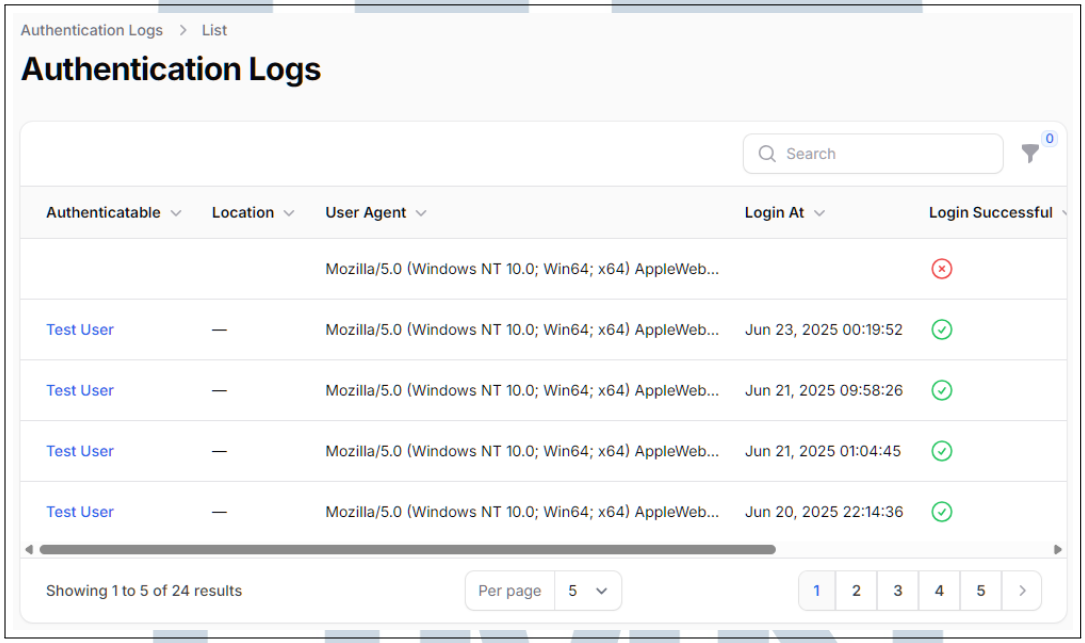
Selama proses pengembangan, terdapat beberapa tantangan teknis yang harus diatasi. Saat pengujian log autentikasi dilakukan menggunakan `localhost`, alamat IP yang tercatat adalah `127.0.0.1`, sehingga GeoIP mengembalikan lokasi default. Hal ini terjadi karena Laravel tidak menerima IP publik asli dari pengguna.

Namun, dari hasil pengujian menggunakan IP `111.95.71.7` pada Gambar 3.31, dapat disimpulkan bahwa GeoIP berfungsi dengan baik. Oleh karena itu, untuk mengatasi tantangan ini, diperlukan proses *hosting* situs web melalui layanan

web-hosting guna mendukung pengujian lebih lanjut.

C Hasil dan Pengembangan Lebih Lanjut

Seluruh konfigurasi dapat dinyatakan berhasil, mengecualikan tantangan untuk mendapatkan IP Publik. Sistem berhasil merekam log ketika ada pengguna yang berusaha untuk melakukan autentikasi melalui *login*. Data seperti user yang digunakan untuk autentikasi, perangkat atau *user agent* yang digunakan, dan waktu autentikasi terekam sesuai dengan harapan penggunaan terlihat pada Gambar 3.32.



Gambar 3.32. Data Hasil Log *Login* dan Lokasi Pengguna

Untuk pengembangan lebih lanjut, dapat ditambahkan sistem *labeling* untuk menentukan apakah terjadi pelanggaran keamanan (*security breach*), serta pengembangan aksi lanjutan untuk menindaklanjuti potensi ancaman pada sistem. Selain itu, notifikasi juga dapat dikembangkan jika teridentifikasi aktivitas mencurigakan, seperti lokasi yang tidak sesuai atau upaya masuk panel yang gagal secara berulang.

3.3.10 Pengembangan *Media Library Manager*

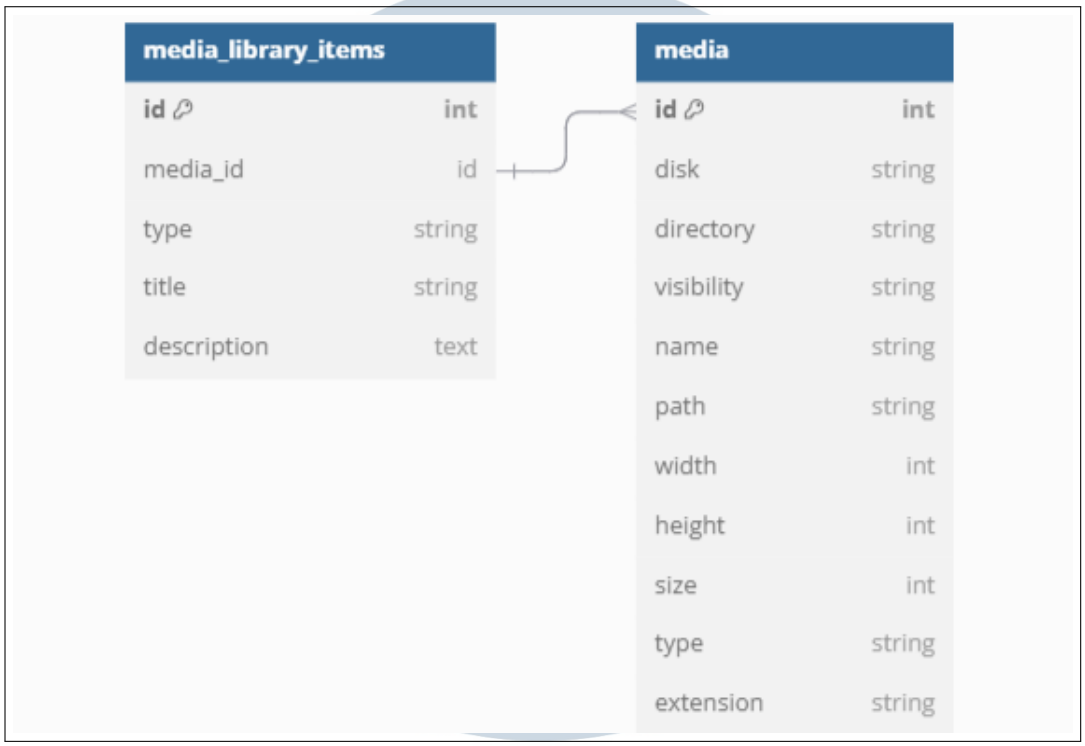
Dalam rangka meningkatkan kualitas layanan informasi publik dan mendukung keterbukaan data, Dinas Perhubungan Kota Tangerang Selatan memerlukan sistem informasi yang mampu menampilkan dokumentasi instansi,

dokumentasi publikasi serta dokumen prosedur layanan secara terstruktur dan mudah diakses. Beberapa halaman pada website resmi instansi ini membutuhkan penyajian dokumen seperti foto-foto dokumentasi kegiatan, serta file SOP (*Standard Operating Procedure*) dalam bentuk PDF untuk kebutuhan pelayanan kepada masyarakat.

Namun, dalam implementasinya, ditemukan bahwa sistem manajemen konten (*Content Management System* atau CMS) yang sedang dibangun belum memiliki fitur khusus untuk mengelola media seperti gambar dan dokumen. File-file media yang ada masih disimpan secara manual tanpa metadata yang jelas dan belum terintegrasi dengan sistem *backend*. Hal ini menyulitkan dalam proses pengelolaan, pencarian, dan penayangan file media secara efisien.

Sebagai solusi atas permasalahan tersebut, dikembangkan fitur *Media Library Manager* pada sistem CMS yang sedang dibangun selama masa magang. Tujuan dari pengembangan ini adalah untuk menyediakan panel administrasi yang memungkinkan admin mengelola dokumen dan gambar secara lebih terstruktur, serta menyediakan antarmuka publik berupa API untuk mendukung penayangan konten media secara dinamis di *website* layanan.

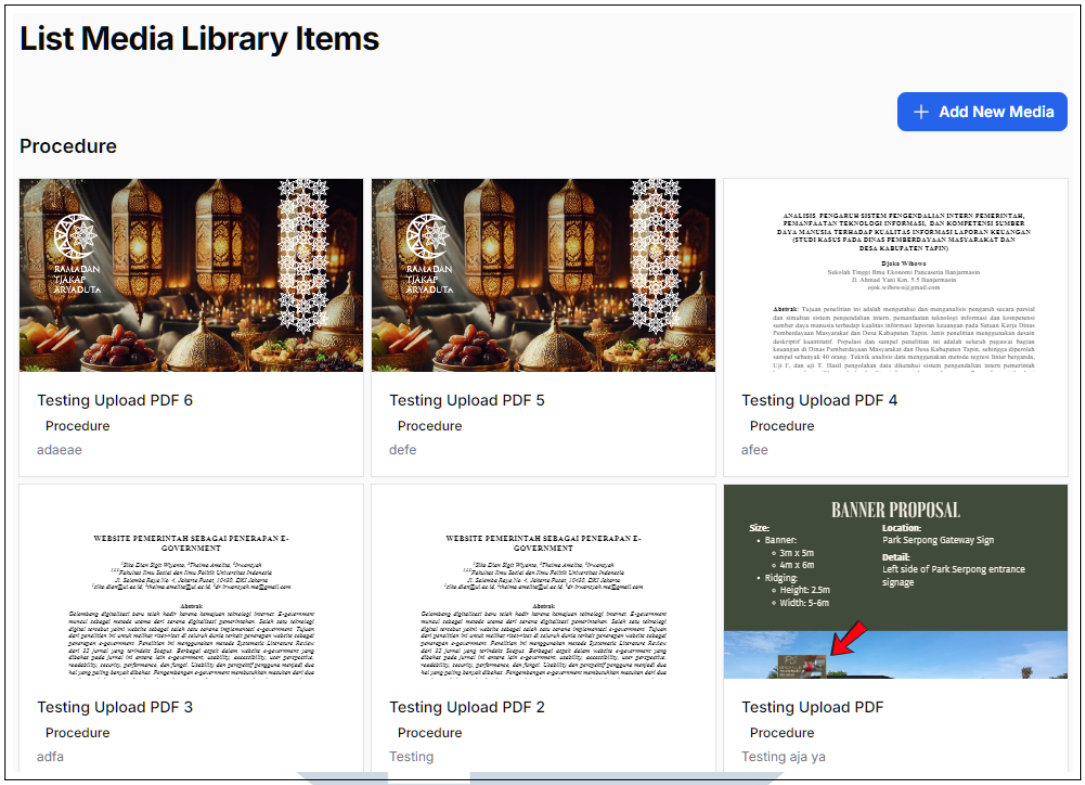
A Proses Pengembangan



Gambar 3.33. Diagram Relasi Entitas untuk Media Library Manager

Tahapan pengembangan fitur diawali dengan analisis kebutuhan sistem dan studi terhadap struktur data yang diperlukan. Tabel `media_library_items` didesain untuk merepresentasikan entitas media yang akan ditampilkan. Setiap item media memiliki atribut seperti judul, deskripsi, dan tipe (misalnya: *documentation* atau *procedure*).

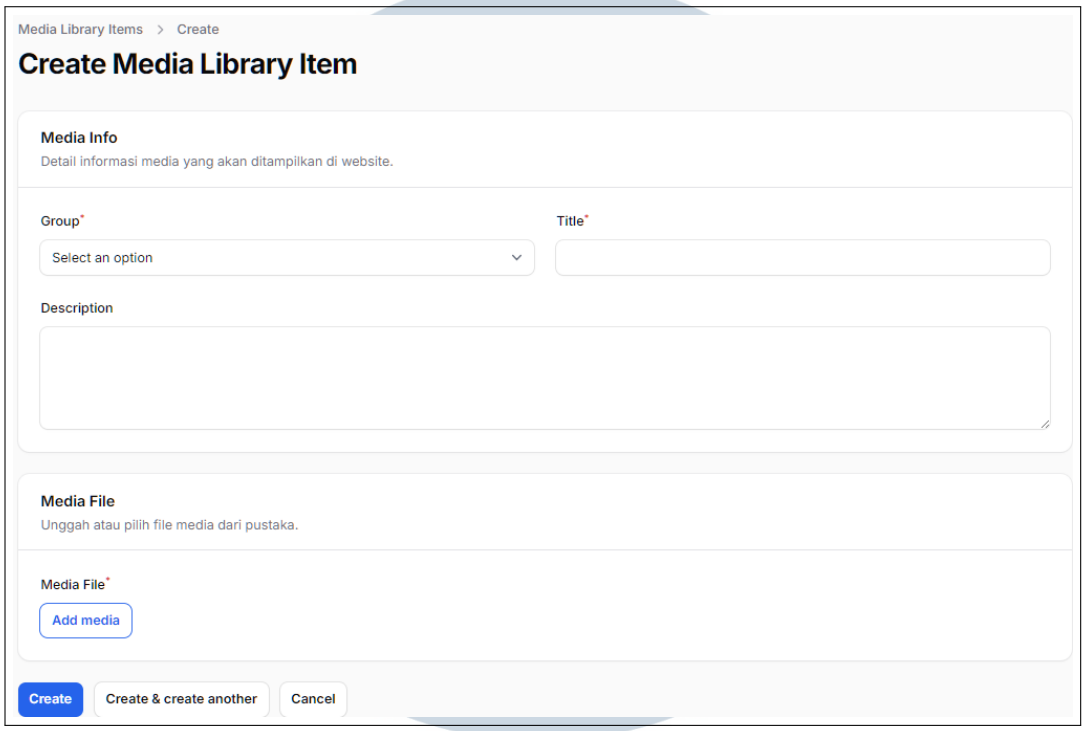
Untuk pengelolaan media, sistem menggunakan pustaka AWCodes Curator, yang memungkinkan lampiran media secara otomatis serta menyimpan informasi seperti `URL` dan `MIME type` dari media. Relasi antar entitas dapat dilihat pada Gambar 3.33.



Gambar 3.34. Tampilan panel admin *Media Library Manager*

Pada sisi admin, digunakan *plugin* FilamentPHP sebagai *framework* untuk *admin panel*. Antarmuka ini menyediakan fitur untuk menambah, mengedit, dan menghapus media. Untuk menampilkan media yang telah diunggah, dibuat tampilan galeri dengan *layout grid* yang dikelompokkan berdasarkan tipe, yaitu dokumentasi atau prosedur. Tampilan galeri ini ditunjukkan pada Gambar 3.34.

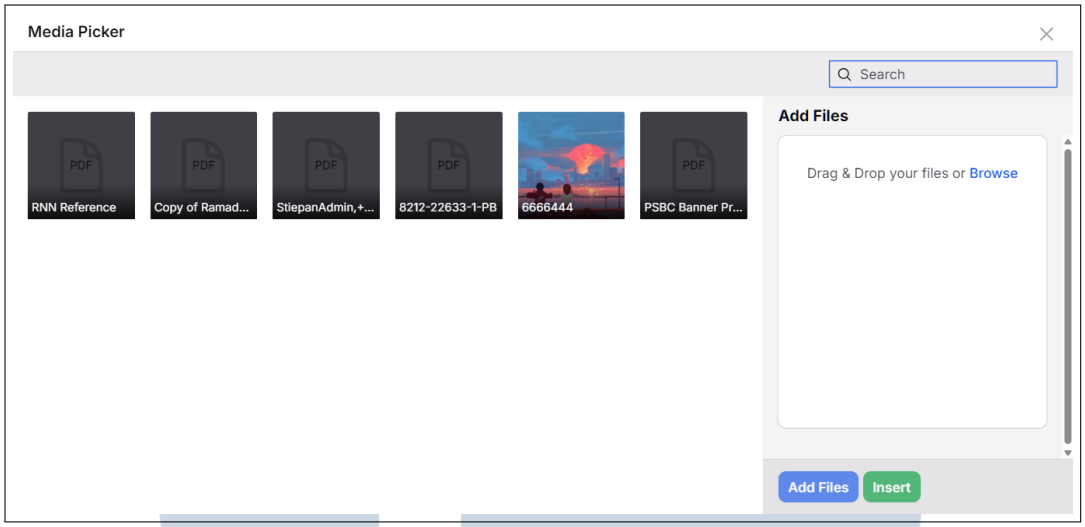
B Tampilan Form Media Library Manager



Gambar 3.35. Tampilan form *Media Library Manager*

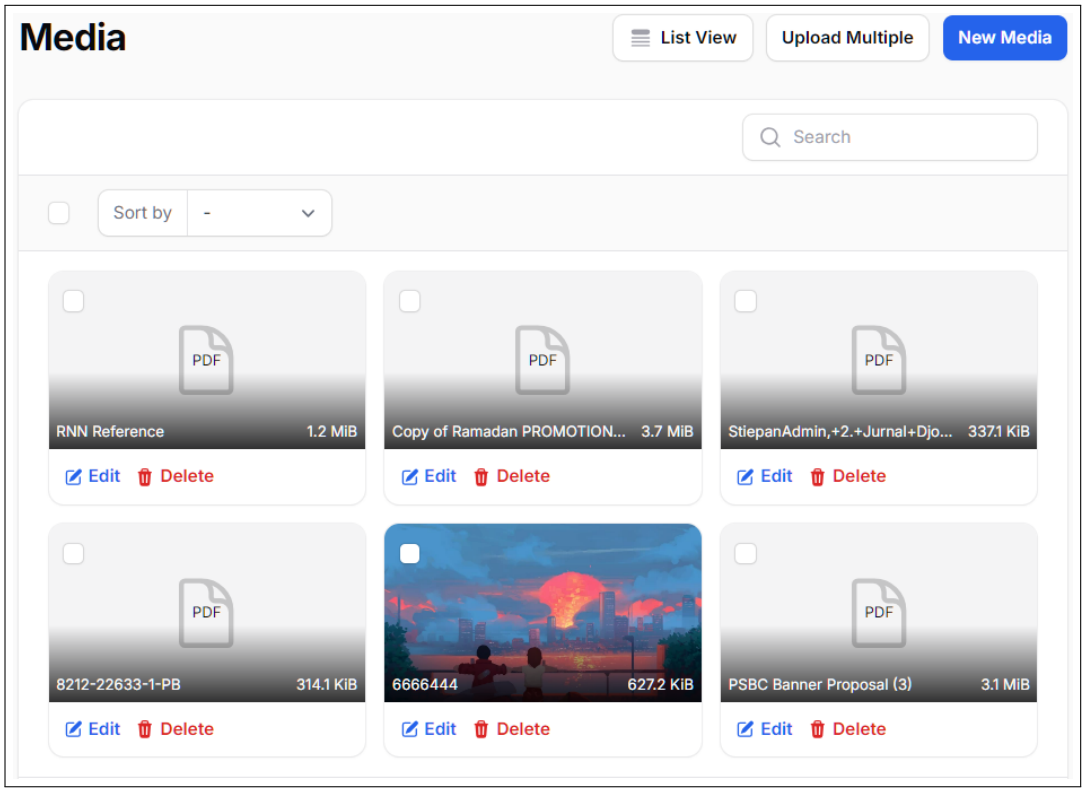
Bagian penting dari fitur ini adalah tampilan *form* input media pada Gambar 3.35 yang digunakan untuk menambah atau mengubah data media. *Form* ini dirancang menggunakan *schema* dari Filament, yang terdiri dari beberapa komponen yang saling terintegrasi.

Komponen pertama adalah `TextInput` untuk mengisi judul media, yang berfungsi sebagai nama identifikasi file. Selanjutnya terdapat `Textarea` untuk deskripsi yang menjelaskan isi atau konteks dari media tersebut. Komponen `Select` digunakan untuk memilih tipe media (dokumentasi atau prosedur), yang menjadi dasar pengelompokan dan penampilan media pada halaman galeri. Terakhir, ditambahkan komponen `CuratorPicker` untuk unggahan media yang mendukung format gambar (`.jpg`, `.png`, `.webp`) dan dokumen (`.pdf`).



Gambar 3.36. Tampilan *Media Picker* oleh Plugin Curator

Pengelolaan gambar dan file PDF menjadi lebih efisien dengan menggunakan plugin Curator dari AWCodes. urator mengelola media yang telah diunggah ke dalam situs web, sehingga memungkinkan staf untuk menggunakan kembali media tersebut tanpa perlu mengunggah duplikat. Curator juga menyediakan komponen `CuratorPicker`, yang menampilkan modal pada lihat Gambar 3.36 untuk memilih atau mengunggah media baru.

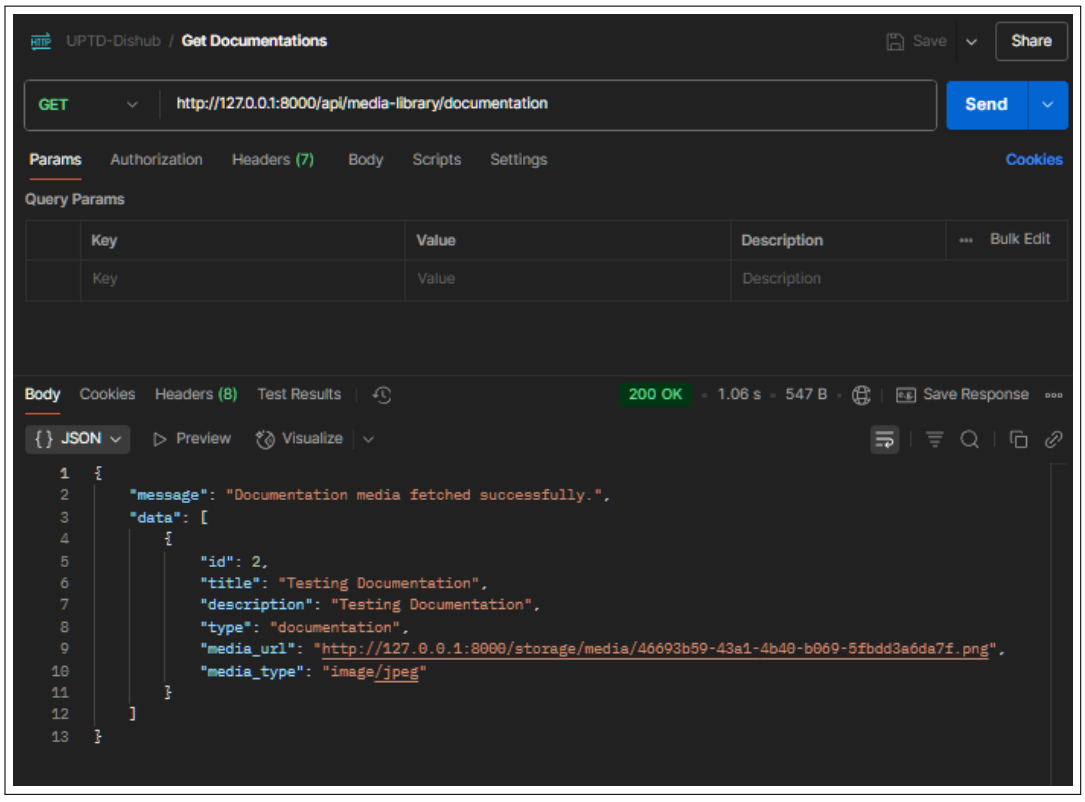


Gambar 3.37. Tampilan panel admin *Resource* Media oleh Curator

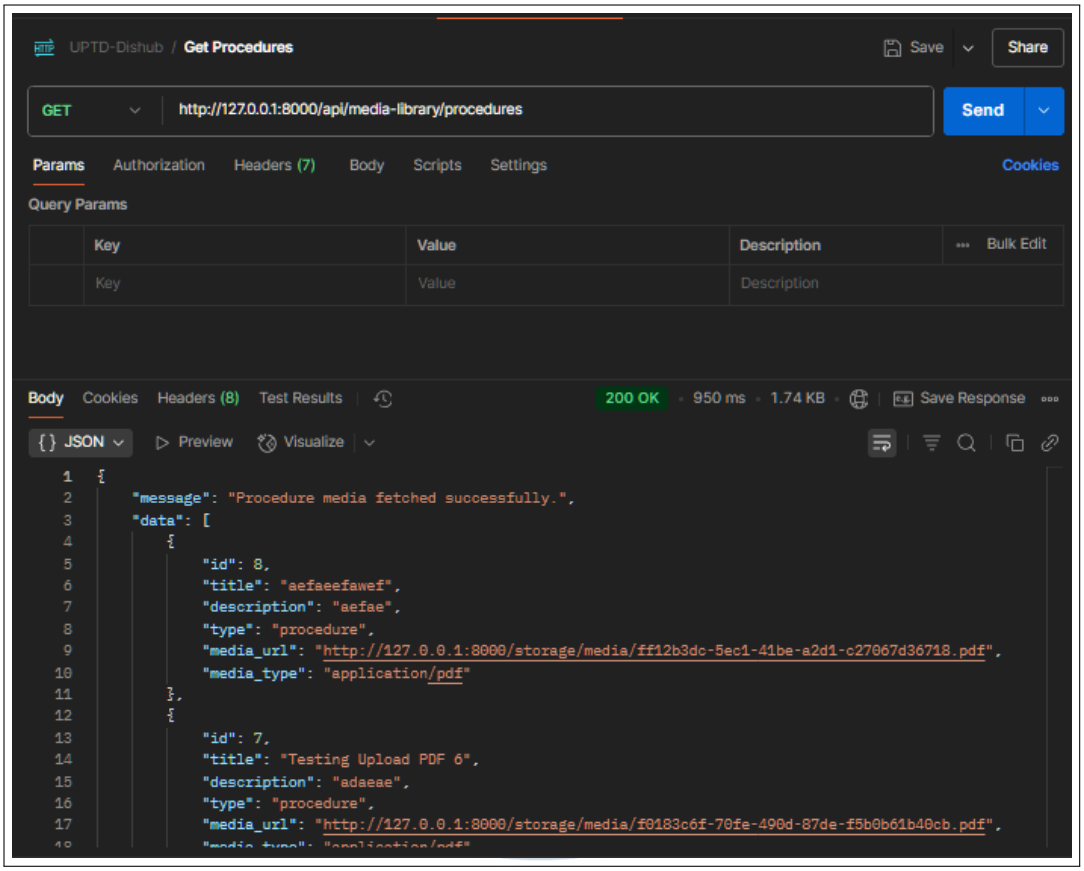
Curator juga menyediakan *Resource* Filament dengan label Media, seperti yang ditampilkan pada Gambar 3.37. Melalui panel *resource* ini, staf dapat melihat seluruh media yang telah diunggah ke dalam situs web. Fitur ini memudahkan pengelolaan media dan membantu meminimalkan penggunaan ruang penyimpanan agar lebih efisien.

C Pengembangan API dan Hasil

Setelah pengembangan selesai, ditambahkan fitur API publik yang memungkinkan *frontend* situs web menampilkan media yang telah dikelola oleh admin. API ini dibagi menjadi dua *endpoint*, yaitu `/api/media/documentation` dan `/api/media/procedures`, yang masing-masing mengembalikan data dalam format `JSON` sesuai dengan kategori media. Setiap item media pada respons API mencakup informasi penting seperti `id`, `title`, `description`, `type`, `media_url`, dan `media_type`. Fungsi `map()` digunakan untuk mengambil seluruh data secara efisien tanpa menerapkan *pagination*, agar lebih sesuai dengan kebutuhan tampilan statis di *frontend*. Hasil respons API dapat dilihat pada Gambar 3.38 dan Gambar 3.39.



Gambar 3.38. Hasil respons API Get Documentations



Gambar 3.39. Hasil respons API Get Procedures

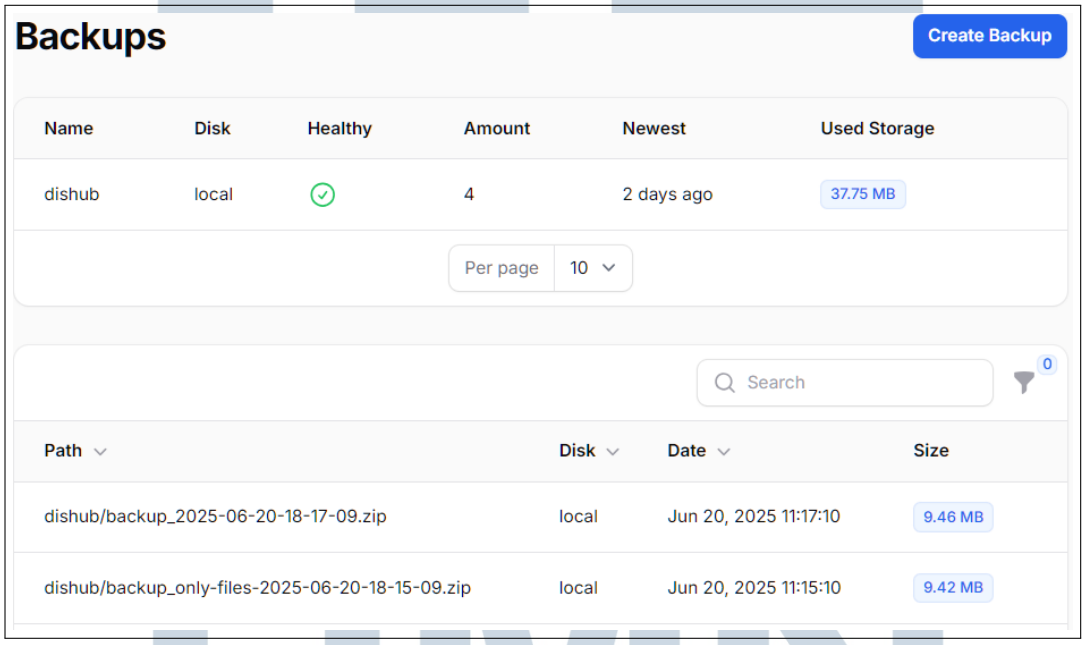
Fitur ini berhasil dirancang dan diimplementasikan dengan mempertimbangkan kemudahan penggunaan bagi admin serta kompatibilitas dengan kebutuhan publikasi informasi digital. Dengan hasil ini, diharapkan sistem CMS mampu menyajikan dokumentasi dan prosedur pelayanan yang lebih baik, informatif, dan dapat diakses oleh masyarakat luas.

3.3.11 Pengembangan Fitur Backup dan Restore

Dalam upaya menjaga integritas dan ketersediaan data pada sistem CMS Dinas Perhubungan, salah satu fitur krusial yang dikembangkan adalah modul untuk pencadangan (*backup*) dan pemulihan (*restore*) data. Mengingat pentingnya data seperti artikel berita, data pengguna, dan konten lainnya, fitur ini dirancang sebagai lapisan keamanan fundamental untuk melindungi sistem dari potensi kehilangan data. Tujuannya adalah untuk menyediakan mekanisme pemulihan bencana (*disaster recovery*) yang memungkinkan administrator mengembalikan sistem ke kondisi stabil sebelumnya, sehingga meminimalisir risiko kehilangan

informasi dan waktu henti (*downtime*) layanan.

Untuk mempercepat proses pengembangan, implementasi fitur ini memanfaatkan *plugin* dari Filament yang terintegrasi dengan *package* Spatie Laravel Backup. Pendekatan ini menambahkan antarmuka grafis langsung ke dalam panel admin, sehingga menyederhanakan proses yang biasanya bersifat teknis. Fungsionalitas utama yang berhasil diimplementasikan dapat dilihat pada Gambar 3.40. Melalui halaman ini, administrator dapat memicu proses pencadangan data secara manual (*on-demand*) yang mencakup basis data PostgreSQL dan file proyek, serta melihat dan mengunduh daftar *backup* yang telah dibuat.



Gambar 3.40. Tampilan panel Backup

Meskipun demikian, terdapat kendala dalam pengembangan fitur ini, yaitu keterbatasan waktu pelaksanaan magang. Fitur untuk melakukan pencadangan (*backup*) telah berhasil diimplementasikan dan berjalan sesuai dengan fungsionalitas yang diharapkan. Namun, untuk fitur pemulihan (*restore*) data langsung melalui panel admin, tahap implementasi, pengujian, dan perbaikan belum dapat diselesaikan secara menyeluruh karena keterbatasan waktu. Oleh karena itu, fitur *restore* menjadi salah satu saran utama untuk pengembangan sistem di masa mendatang.

3.4 Kendala dan Solusi yang Ditemukan

Selama kegiatan magang dalam pengembangan CMS, dihadapi sejumlah tantangan teknis yang signifikan, terutama dalam hal integrasi fitur kecerdasan buatan (AI) untuk optimasi SEO, serta pengelolaan penjadwalan publikasi berita yang melibatkan perbedaan zona waktu. Berikut merupakan uraian lengkap mengenai kendala yang dihadapi, analisis penyebabnya, dan solusi yang diterapkan secara sistematis.

3.4.1 Integrasi *Model* AI untuk Optimasi SEO

Salah satu fitur yang dirancang dalam pengembangan sistem adalah modul yang dapat menghasilkan metadata SEO `title` dan `meta description` secara otomatis. Gagasan ini bertujuan untuk meningkatkan visibilitas artikel di mesin pencari secara efisien dengan bantuan *model* AI generatif.

Namun, ditemukan kendala dalam integrasi *model* kecerdasan buatan (AI) yang diharapkan, khususnya *model* dengan kemampuan tingkat lanjut seperti OpenAI GPT-4 dan Claude dari Anthropic. Jenis-jenis *model* tersebut membutuhkan akses API berbayar yang berada di luar cakupan sumber daya magang. Selain itu, sebagian besar *model* gratis yang tersedia memiliki keterbatasan dalam memahami konteks berita lokal serta penguasaan terhadap praktik SEO terkini.

Sebagai solusi dari kendala integrasi *model* kecerdasan buatan (AI) berbayar dengan biaya tinggi, diputuskan untuk menggunakan Gemini 1.5 Flash sebagai alternatif yang cukup efisien dan dapat diakses secara praktis. Gemini 1.5 Flash merupakan salah satu *model* generatif terbaru dari Google DeepMind yang mengutamakan kecepatan dan efisiensi, sehingga sangat cocok untuk mendukung proses otomatisasi metadata SEO pada sistem manajemen konten (CMS) yang sedang dikembangkan. *Model* AI ini digunakan untuk menghasilkan meta title dan meta description berdasarkan konten artikel yang ditulis oleh pengguna.

Untuk memaksimalkan potensi dari *model* kecerdasan buatan (AI) yang digunakan, diterapkan prinsip *prompt engineering* sebagai pendekatan utama untuk mengarahkan *model* AI agar memberikan keluaran yang tepat sasaran. Prompt engineering adalah teknik merancang instruksi (*prompt*) secara spesifik dan terstruktur agar *model* AI memahami peran dan tugasnya dengan jelas. Teknik ini memastikan bahwa integrasi AI tidak hanya efisien, tetapi juga dapat diandalkan

dalam menghasilkan metadata SEO yang konsisten, terstruktur, dan sesuai dengan kebutuhan konten.

3.4.2 Ketidaksesuaian Zona Waktu pada Penjadwalan Publikasi

Dalam proses pengembangan fitur penjadwalan publikasi artikel pada CMS, dihadapi permasalahan ketidaksesuaian zona waktu yang berdampak langsung pada akurasi waktu penerbitan konten. Fitur ini sejatinya dirancang agar pengguna dapat menentukan waktu publikasi secara otomatis sesuai kebutuhan, misalnya menjadwalkan artikel untuk terbit pada pukul 08.00 WIB keesokan harinya. Namun, selama tahap pengujian, ditemukan bahwa waktu publikasi yang dijalankan oleh sistem tidak sesuai dengan waktu lokal pengguna, melainkan mengalami keterlambatan sekitar tujuh jam.

Setelah dilakukan penelusuran teknis, diidentifikasi bahwa akar permasalahan terletak pada perbedaan zona waktu antara aplikasi dan sistem basis data. Secara default, Laravel menggunakan zona waktu UTC dalam seluruh operasi waktu internalnya, termasuk saat menyimpan atau mengambil data dari basis data. PostgreSQL sebagai sistem basis data yang digunakan juga menyimpan nilai timestamp dalam format UTC, tanpa memperhitungkan zona waktu lokal. Akibatnya, artikel yang dijadwalkan untuk terbit pada pukul tertentu di Waktu Indonesia Barat (GMT+7) justru diterbitkan berdasarkan waktu UTC, sehingga muncul ketidaksesuaian dan membingungkan pengguna yang mengandalkan fitur ini.

Untuk mengatasi permasalahan tersebut, dilakukan konfigurasi ulang pada pengaturan zona waktu, baik di sisi Laravel maupun PostgreSQL. Di Laravel, file konfigurasi `config/app.php` disesuaikan dengan mengganti nilai timezone menjadi '`Asia/Jakarta`'. Selain itu, dipastikan bahwa koneksi basis data PostgreSQL juga mempertimbangkan zona waktu yang sesuai melalui konfigurasi tambahan pada *driver* basis data. Setelah perbaikan ini diterapkan, sistem berhasil menyesuaikan seluruh proses penjadwalan dengan waktu lokal pengguna, sehingga artikel dapat terbit tepat waktu sesuai ekspektasi.