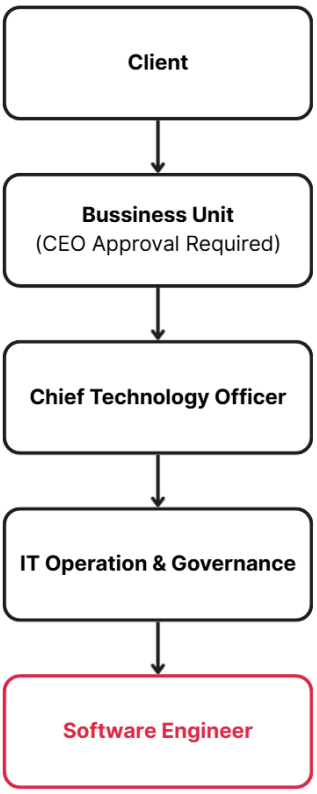
BAB III

PELAKSANAAN KERJA MAGANG

3.1 Kedudukan dan Koordinasi



Gambar 3.1 Kedudukan dan Organisasi

Struktur organisasi pada bagian ini menggambarkan alur kedudukan posisi penulis sebagai *Software Engineer (Full Stack Developer)* di dalam lingkungan perusahaan WIR Group selama masa pelaksanaan magang. Alur ini dimulai dari klien, sebagai pihak yang mengajukan kebutuhan atau permasalahan yang perlu diselesaikan melalui pengembangan solusi digital. Permintaan ini kemudian disalurkan ke *Business Unit* terkait di bawah naungan WIR Group, yang bertanggung jawab untuk mengelola kebutuhan klien sesuai dengan fokus bisnisnya. Setiap permintaan atau proyek dari *Business Unit* ini harus melalui proses persetujuan dari CEO *Business Unit*, sebagai bentuk validasi strategis dan kelayakan proyek sebelum dilanjutkan ke tim teknologi.

Setelah mendapatkan persetujuan, proyek diserahkan kepada *Chief Technology Officer (CTO)*, yang memimpin seluruh aspek teknologi dalam perusahaan. CTO berperan penting dalam mengarahkan implementasi teknologi secara menyeluruh, mulai dari strategi pengembangan sistem hingga tata kelola teknis dan operasional yang mendukung baik kebutuhan internal perusahaan maupun eksternal (klien/mitra). Di bawah CTO, terdapat unit IT *Operations & Governance*, yang berperan dalam menjalankan pengawasan terhadap operasional sistem informasi, pengelolaan infrastruktur, serta memastikan seluruh aktivitas pengembangan dan operasional IT berjalan sesuai dengan kebijakan dan standar perusahaan.

Penulis menjalankan peran sebagai *Full Stack Developer* yang berada langsung di bawah unit IT *Operations & Governance*. Dalam posisi ini, penulis terlibat secara langsung dalam proses pengembangan aplikasi, baik dari sisi *frontend* maupun *backend*.

Dalam pelaksanaan kegiatan magang ini, Bapak Budi Kurniawan berperan sebagai supervisor sekaligus mentor yang memberikan arahan dan bimbingan selama proyek berlangsung. Beliau bertanggung jawab dalam mendelegasikan tugas, mengkoordinasikan alur kerja pengembangan, serta memberikan panduan teknis yang sesuai dengan standar perusahaan. Selain itu, beliau juga turut membimbing dalam pengambilan keputusan teknis dan pendekatan pemecahan masalah selama proses pengembangan *software* berlangsung.

3.2 Tugas dan Uraian Kerja Magang

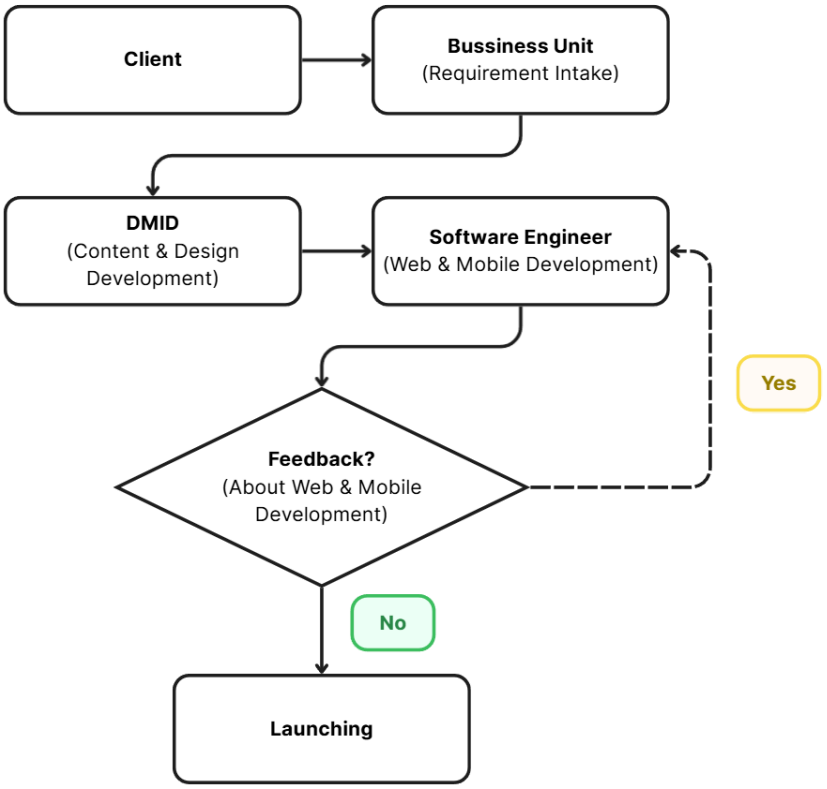
Selama menjalani program kerja magang di PT WIR Asia Tbk., penulis terlibat secara langsung dalam proses pengembangan yang terdiri dari *website* profil, *Content Management System* (CMS), serta *backend* aplikasi *mobile* Rumah Donor. Setiap minggu memberikan pengalaman dan pembelajaran yang berbeda, baik dari sisi teknis maupun kolaboratif. Berikut ini adalah rangkuman tugas dan kegiatan yang dilakukan penulis selama masa pelaksanaan magang, disusun berdasarkan minggu pelaksanaan untuk memberikan gambaran yang lebih terstruktur dan menyeluruh terhadap proses pengembangan yang dijalankan.

Tabel 3.1 Ringkasan Pekerjaan Setiap Minggu

Minggu Ke-	Ringkasan Pekerjaan
1	● Pengenalan lingkungan kerja dan *briefing* proyek. ● Analisis kebutuhan sistem Rumah Donor bersama *business unit*. ● Kolaborasi awal dengan tim DMID dalam pengembangan website menggunakan WordPress dan *custom* CSS.
2	● Lanjutan pengembangan UI *website* Rumah Donor
3	● Pengembangan responsif seluruh halaman *website*. ● Adaptasi *backend* NestJS dan diskusi teknis pengembangan *user module*.
4	● Revisi visual berdasarkan *feedback* dan finalisasi tampilan *website*. ● Pendalaman konsep *Guard* di NestJS.
5	● Lanjutan integrasi WP Mail SMTP dan Roundcube untuk *question form* dan Newsletter plugin untuk *subscription*.
6	● Finalisasi newsletter *subscription feature*.
7	● *Review* Rumah Donor UI/UX *Mobile App*. ● Pembuatan ERD untuk aplikasi *mobile* Rumah Donor.
8	● Riset integrasi fitur dari calon *partner* Rumah Donor untuk kebutuhan menu Rumah Gizi. ● Integrasi API statistik mitra Rumah Donor. ● *Refactor mailer module*.
9	● Lanjutan pengembangan *user module* (implementasi OTP)
10	● Pengembangan *news module* dan lanjutan pengembangan *user module*.
11	● Pengembangan awal *Content Management System*.
12	● Integrasi *news module* dengan *local bucket* dan mengembangkan *news picture module*.
13	● Lanjutan pengembangan CMS (*file-based routing*,

	custom hooks, integrasi API)
14	• Lanjutan pengembangan *backend* dan CMS
15	• Lanjutan pengembangan *backend* dan CMS

3.2.1 Alur Pembuatan *Website* Profil, *Content Management System*, dan *Backend* Aplikasi Rumah Donor



Gambar 3.2 Alur Pembuatan Proyek Rumah Donor

Alur pembuatan *website* dan *mobile app* dapat dilihat pada diagram di atas. Proses dimulai dari klien, yang memiliki kebutuhan tertentu terkait pengembangan produk digital. Kebutuhan tersebut kemudian disampaikan kepada *Business Unit* untuk dikaji melalui proses pengumpulan dan analisis kebutuhan (*requirement intake*) sebelum dilanjutkan ke tahap berikutnya.

Setelah kebutuhan dikaji, *Business Unit* akan meneruskannya kepada tim DMID, yaitu unit bisnis yang berperan dalam mengembangkan konten serta desain visual dari produk yang akan dibuat. Di tahap ini, DMID akan menyiapkan desain antarmuka dan materi konten yang mengacu pada kebutuhan awal dari klien, dan biasanya dilakukan melalui komunikasi dua arah antara tim internal dan pihak klien untuk memastikan semuanya selaras.

Setelah desain diselesaikan, tahap berikutnya adalah pengembangan teknis oleh *Software Engineer*. Pada fase ini, pengembangan dilakukan baik untuk *website* maupun *mobile app* sesuai dengan rancangan yang sudah disiapkan sebelumnya. Proses ini mencakup implementasi *frontend* dan *backend*, serta pengujian fungsionalitas awal dari sistem.

Hasil pengembangan yang sudah berjalan akan masuk ke tahap *Feedback*, di mana *output* dari tim *developer* akan di-*review* terlebih dahulu. Proses *feedback* ini melibatkan pengecekan dari sisi desain, fungsi, maupun kenyamanan penggunaan. Jika hasil sudah sesuai, maka pengembangan dinyatakan selesai dan produk bisa masuk ke tahap *Launching*. Namun apabila masih ada revisi, maka *feedback* yang berkaitan dengan hal teknis akan dikembalikan ke tim *developer* untuk diperbaiki hingga hasil akhirnya sudah memenuhi standar dan ekspektasi yang ditentukan.

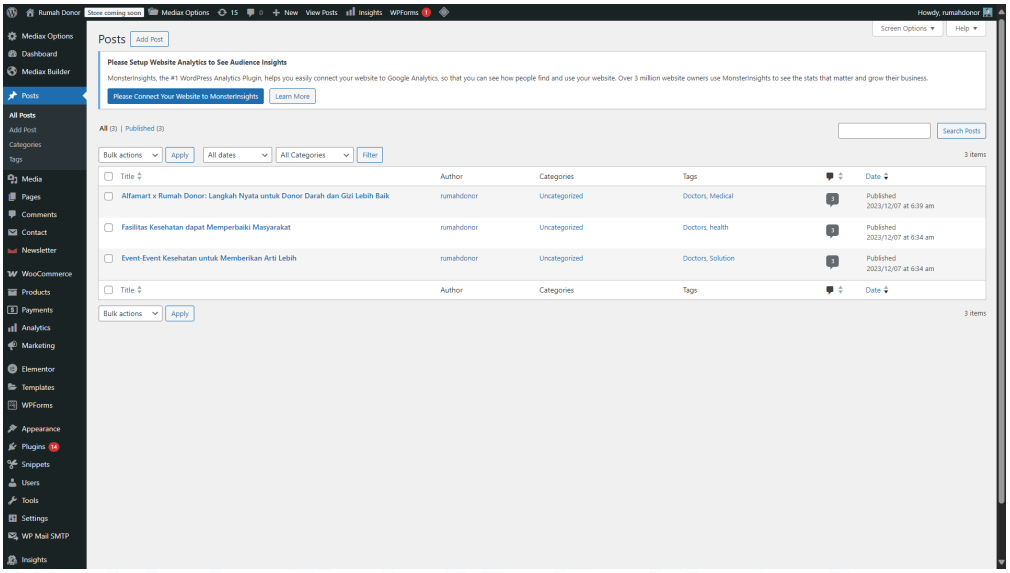
3.2.2 Pengembangan *Website* Profil Rumah Donor

3.2.2.1 Mempelajari Pengembangan *Website* dengan WordPress

Pemilihan WordPress sebagai platform pengembangan *website* profil Rumah Donor merupakan hasil kesepakatan antara pihak klien dan divisi *Information and Technology* WIR Group sebelum penulis bergabung dalam proyek ini. WordPress dipilih bukan tanpa alasan, selain karena tersedianya template khusus

yang telah disetujui oleh klien, penggunaan WordPress juga dinilai dapat mempercepat proses pengembangan serta mempermudah proses pemeliharaan dan pembaruan konten ke depannya.

WordPress sendiri merupakan sebuah sistem manajemen konten (*Content Management System*/CMS) berbasis web yang awalnya dikembangkan untuk kebutuhan publikasi *blog*. Seiring waktu, WordPress berevolusi menjadi platform yang mampu mengakomodasi berbagai jenis konten digital seperti *website* perusahaan, *mailing list*, forum internet, galeri media, situs keanggotaan, sistem pembelajaran daring, hingga toko online [2]. Dalam konteks proyek Rumah Donor, WordPress Admin digunakan sebagai portal *backend* yang memungkinkan pihak klien atau administrator mengelola konten *website* secara mandiri, tanpa perlu mengakses sisi pengembangan secara langsung. Fleksibilitas inilah yang menjadikan WordPress sebagai pilihan tepat untuk memenuhi kebutuhan pengelolaan konten dengan efisien dan praktis.

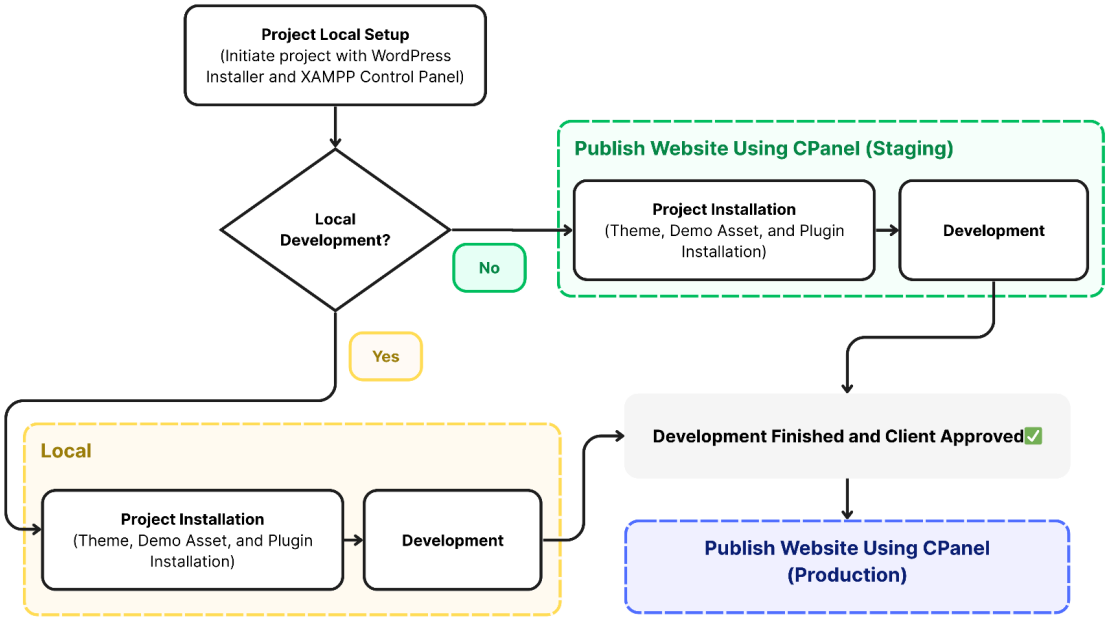


Gambar 3.3 Tampilan WordPress Admin

Pengembangan *website* menggunakan WordPress yang cukup kompleks merupakan pengalaman pertama bagi penulis.

Oleh karena itu, penulis perlu melakukan adaptasi, terutama dalam hal proses *setup project*, pengembangan, hingga tahap *deployment*. Proses adaptasi ini tidak memakan waktu terlalu lama, yakni sekitar dua hingga empat hari, karena sebagian teknologi yang digunakan dalam pengembangan WordPress sudah pernah dipelajari oleh penulis selama masa perkuliahan, seperti PHP, HTML5, dan CSS.

Selama masa adaptasi tersebut, penulis juga berdiskusi dengan Bapak Budi Kurniawan untuk memahami alur teknis pengembangan *website* berbasis WordPress berdasarkan beberapa proyek yang pernah dikerjakan oleh WIR Group. Berdasarkan hasil diskusi tersebut, penulis menyusun kembali alur teknis pengembangan WordPress yang umumnya dilakukan di perusahaan.



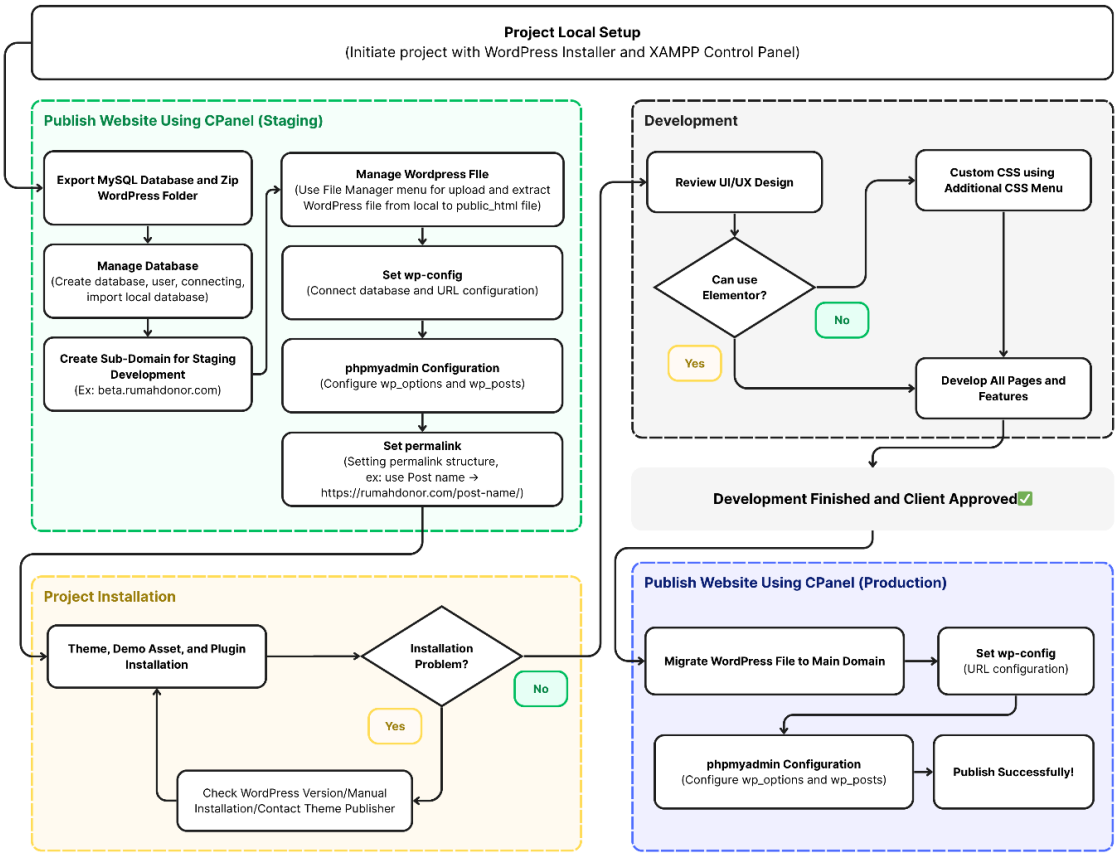
Gambar 3.4 Alur Pengembangan Website WordPress

Alur pengembangan website WordPress terbagi menjadi dua lokasi pengembangan, yaitu pengembangan di *local*

environment dan pengembangan setelah *website* di-*publish* (*staging*). Secara teknis, tidak terdapat perbedaan signifikan antara keduanya, kecuali pada lokasi pengembangannya. Namun demikian, setelah melakukan diskusi lebih lanjut bersama Bapak Budi Kurniawan, penulis merekomendasikan untuk langsung melakukan proses pengembangan setelah website di-*publish*, dan rekomendasi tersebut disetujui. Beberapa pertimbangan dari penulis terkait keputusan tersebut antara lain:

1. Menghindari risiko kehilangan halaman yang telah dibuat, karena jika pengembangan dilakukan secara lokal, terdapat kemungkinan file menjadi *corrupt* atau perangkat hilang dan rusak.
2. Progres pengembangan website dapat dipantau secara *real-time*.
3. Mempermudah proses pengujian langsung di berbagai perangkat dengan ukuran layar yang berbeda, seperti *smartphone* dan *tablet*.
4. Menghindari terjadinya *timeout* saat melakukan proses *import database* ke CPanel.

3.2.2.2 Proses Pengembangan Seluruh Halaman *Website* Profil

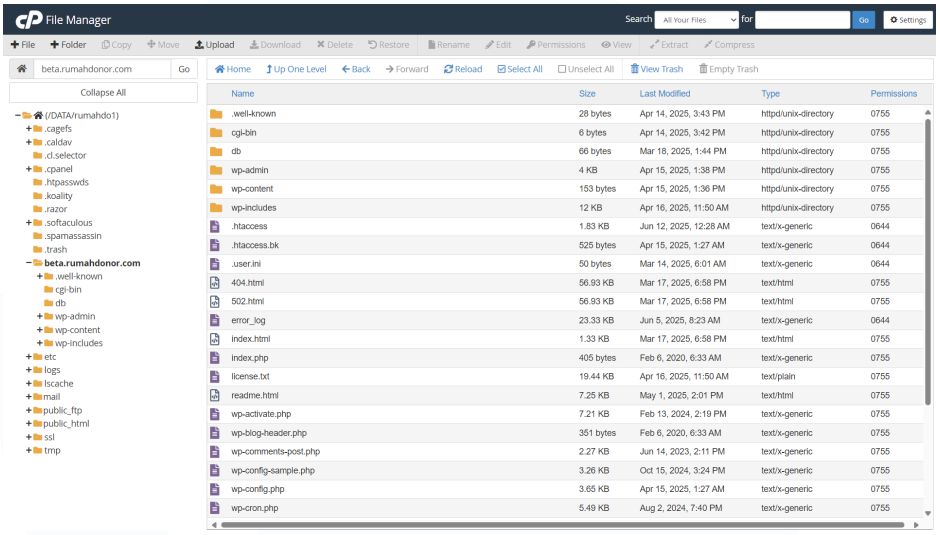


Gambar 3.5 Alur Pengembangan *Website* Profil Rumah Donor

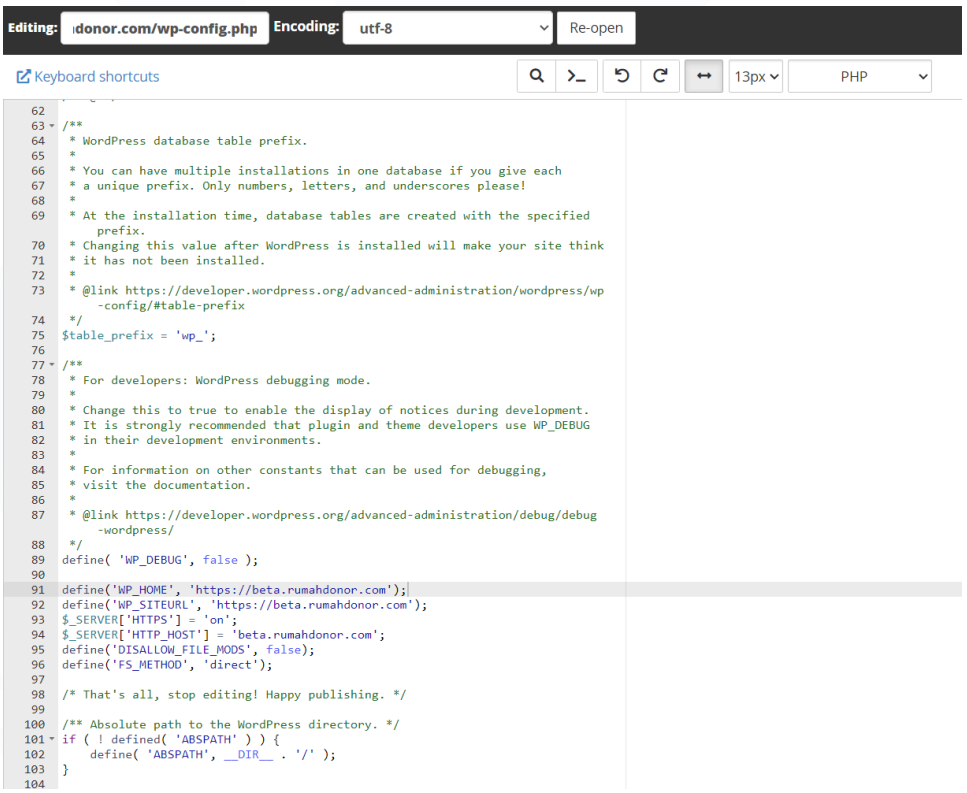
Penulis memulai pengembangan *website* profil Rumah Donor dengan melakukan proses *setup* proyek di *local environment*. Tahap ini diawali dengan menginstal WordPress Installer dan memindahkan file WordPress ke dalam folder htdocs pada XAMPP. Selanjutnya, penulis membuat database baru menggunakan MySQL dan menghubungkannya dengan file WordPress agar dapat dijalankan secara lokal.

Setelah proses *setup* di lokal selesai, penulis melakukan *export database* dan mengompres seluruh folder proyek WordPress ke dalam format .zip untuk selanjutnya di-*upload* ke CPanel. Di dalam CPanel, penulis melakukan beberapa langkah konfigurasi teknis, seperti manajemen *database*, pembuatan *subdomain* sebagai *staging*

environment, serta pengaturan agar halaman *website* dapat ditampilkan melalui *subdomain* tersebut.



Gambar 3.6 CPanel *File Manager*



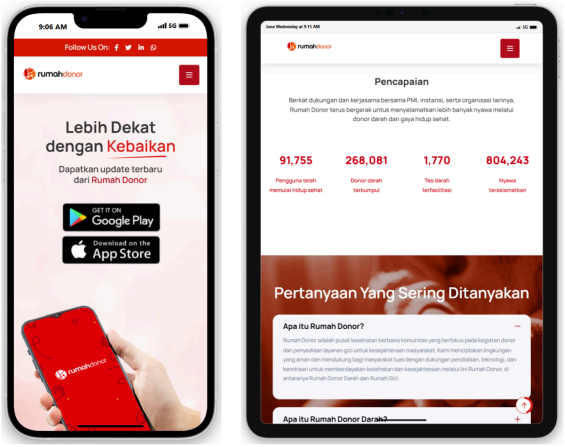
Gambar 3.7 Konfigurasi di wp-config.php

Setelah *website* berhasil ditampilkan pada *staging*, penulis melanjutkan dengan instalasi tema (*theme*), *demo asset*, serta *plugin* yang diperlukan sesuai kebutuhan proyek. Tahap selanjutnya masuk ke proses *development*, yang dimulai dengan melakukan *review* terhadap hasil desain UI/UX dari tim DMID. Pada tahap ini, penulis menganalisis setiap *section* pada desain untuk menentukan apakah cukup dibangun menggunakan Elementor atau memerlukan kustomisasi tambahan melalui *custom CSS*.

Dalam proses pengembangan, penulis secara aktif berkolaborasi dengan UI/UX *Designer* untuk memastikan setiap hasil implementasi sesuai dengan desain dan ekspektasi yang telah ditetapkan. Setelah seluruh halaman dan fungsionalitas selesai dikembangkan dan mendapatkan persetujuan dari pihak klien, penulis melanjutkan ke tahap publikasi *website*. Tahap ini meliputi proses migrasi *file* dari direktori *subdomain* ke direktori *main domain*, serta konfigurasi tambahan agar tampilan pada *main domain* sama dengan versi *staging* yang telah dikembangkan sebelumnya.

- Proses *Slicing* Halaman Beranda, Tentang, Kegiatan, Berita, Mitra, dan Kontak Kami

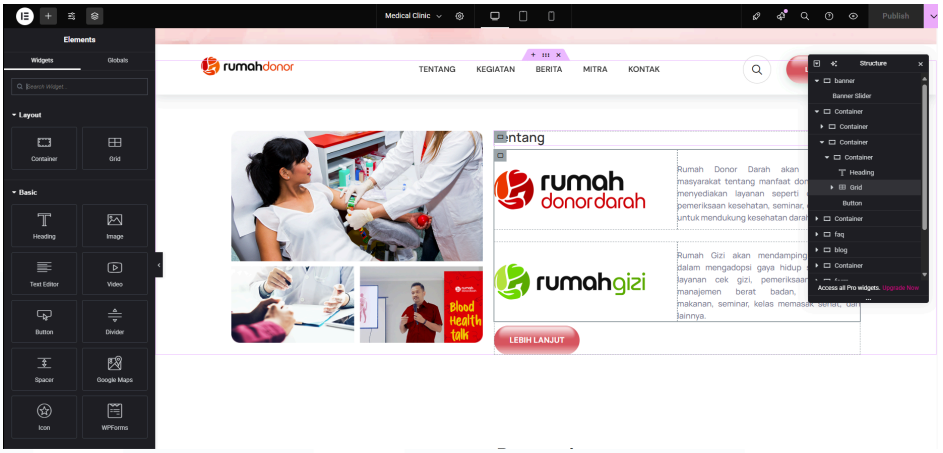
  Dalam proses *development*, penulis mengonversi desain antarmuka ke dalam tampilan *website* dengan pendekatan *responsive design*, dimulai dari tampilan untuk layar *desktop*, kemudian disesuaikan ke berbagai ukuran layar seperti *tablet* dan *mobile* demi memastikan aksesibilitas konten yang optimal, dapat dilihat pada gambar 3.8 *Responsive* di Layar yang berbeda. Selama proses pengerjaan, penulis juga memberikan masukan terkait peletakan beberapa komponen guna menciptakan tampilan yang lebih terstruktur dan nyaman dilihat oleh pengguna.



Gambar 3.8 *Responsive* di Layar yang berbeda

Proses *slicing* yang dilakukan terdiri atas dua pendekatan utama, yaitu menggunakan Elementor dan *custom* CSS. Elementor digunakan untuk menyusun struktur layout halaman, baik dengan *container* maupun *grid*, yang disesuaikan dengan isi kontennya. Namun, mengingat adanya keterbatasan dari Elementor maupun *theme* yang digunakan oleh klien, terutama dalam hal pengaturan *breakpoint* untuk kebutuhan *responsiveness* penulis melanjutkan proses *styling* menggunakan Additional CSS.

Selain untuk penyesuaian *breakpoint*, *custom CSS* juga digunakan untuk memodifikasi sejumlah komponen yang tidak dapat dikustomisasi secara langsung melalui Elementor, seperti *navigation bar*, *form*, serta beberapa *asset* ilustrasi yang ditampilkan pada halaman *website*.

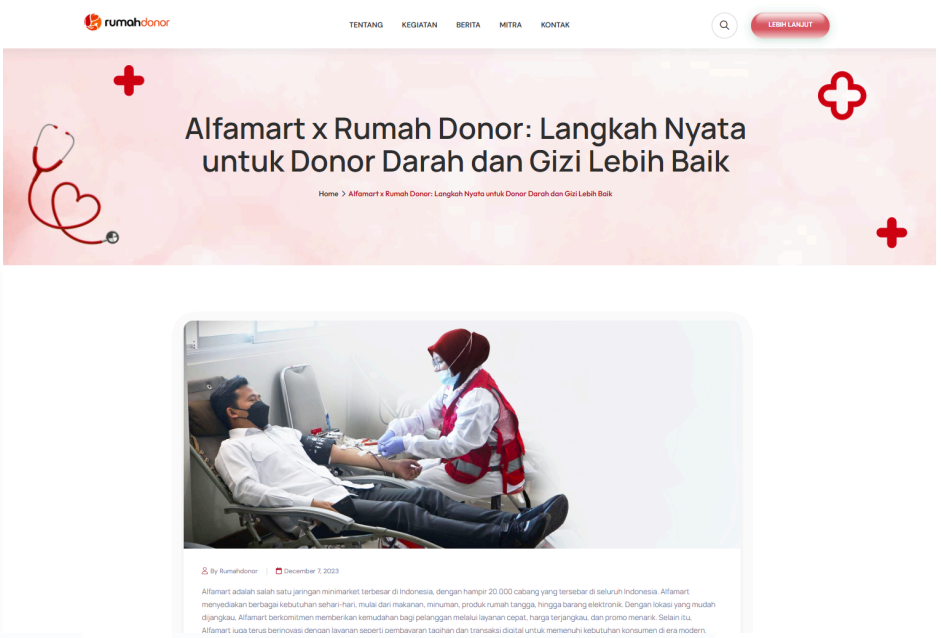


Gambar 3.9 Proses *Slicing* dengan Elementor

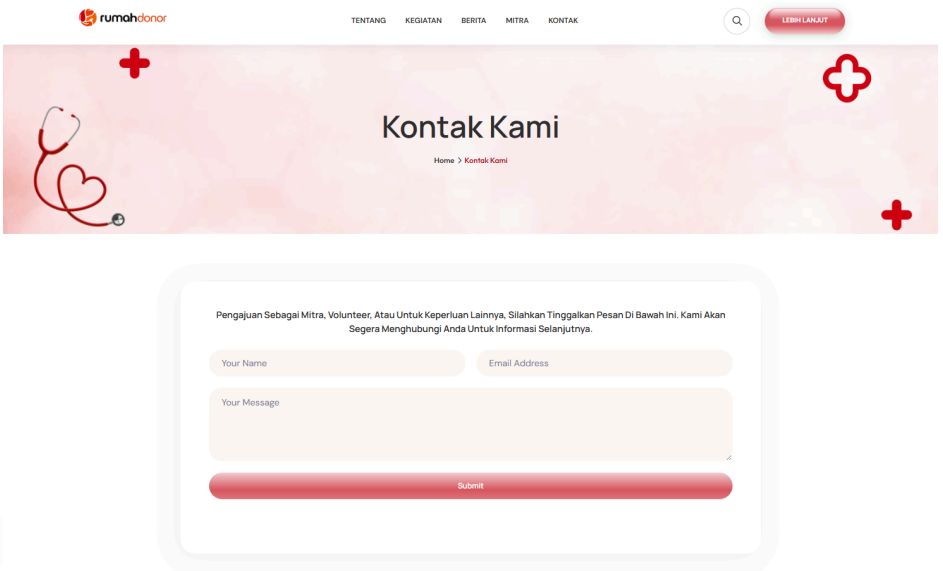
Gambar 3.10 *Custom* CSS dengan Additional CSS Menu

Gambar 3.11 Halaman Beranda

Gambar 3.12 Halaman Kegiatan



Gambar 3.13 Halaman Detail Berita



Gambar 3.14 Halaman Kontak Kami

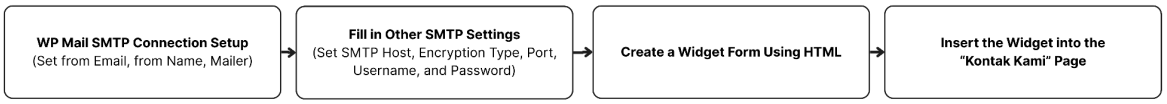
- Implementasi *Form* dengan WP Mail SMTP dan Roundcube

Pada website profil Rumah Donor terdapat satu section berupa *question form* yang memungkinkan pengguna untuk mengirimkan pesan langsung ke email resmi milik Rumah Donor. Untuk mengimplementasikan fitur ini, penulis menggunakan WP

Mail SMTP sebagai plugin utama untuk mengatur sistem pengiriman email agar *reliable* dan dapat masuk ke dalam *inbox* dengan baik. WP Mail SMTP memperbaiki masalah keterkiriman email (*email deliverability*) pada WordPress dengan cara mengkonfigurasi ulang sistem WordPress agar menggunakan penyedia SMTP yang sesuai saat mengirim email [3].

SMTP (*Simple Mail Transfer Protocol*) sendiri adalah protokol standar yang digunakan untuk pengiriman email antar server. Dalam konteks ini, SMTP digunakan untuk memastikan pesan dari form pada website dapat dikirim dengan aman dan sampai ke alamat email tujuan secara tepat waktu.

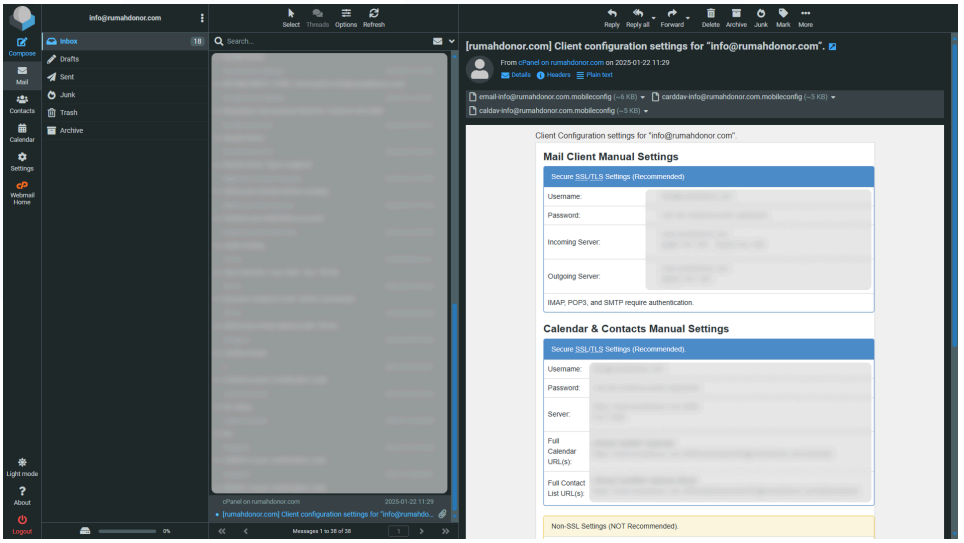
Selain WP Mail SMTP, penulis juga menggunakan Roundcube, yaitu *free and open-source webmail software*, untuk keperluan manajemen *email* yang masuk dari *website* Rumah Donor. Roundcube digunakan oleh klien sebagai antarmuka untuk membaca, mengelola, dan merespon *email* yang masuk dari pengguna *website*.



Gambar 3.15 Alur Implementasi *Form*

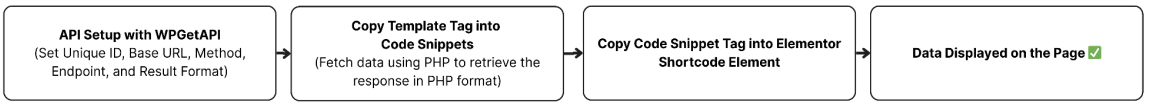
Langkah awal yang penulis lakukan adalah melakukan *setup* koneksi utama dengan mengatur beberapa konfigurasi seperti *From Email*, *From Name*, serta *Mailer* yang digunakan. Pada bagian *Mailer*, penulis memilih opsi Other SMTP, karena pihak klien telah menyediakan kredensial dari *mail client* SMTP yang akan digunakan. Setelah itu, penulis melanjutkan dengan mengisi pengaturan teknis seperti SMTP *Host*, *Encryption Type*, *Port*, *Username*, dan *Password*. Setelah proses konfigurasi selesai,

penulis membuat sebuah *widget* berbentuk *question form* menggunakan kode HTML yang terdiri atas beberapa *input field* seperti nama, alamat *email*, dan pesan. Data dari input tersebut kemudian dikirim menggunakan metode yang telah dikonfigurasi melalui WP Mail SMTP. *Form widget* yang telah penulis buat, dimasukkan ke dalam halaman Kontak Kami, yang dapat dilihat pada gambar 3.14 Halaman Kontak Kami.



Gambar 3.16 Tampilan Roundcube

- Implementasi API Statistik Mitra



Gambar 3.17 Alur Implementasi API hingga Tampil di Halaman

Dalam proses pengembangan *website*, klien menginginkan agar terdapat data statistik mitra yang bekerja sama dengan Rumah Donor, yang ditampilkan secara *real-time* dalam bentuk angka yang terus bertambah. Untuk memenuhi kebutuhan tersebut, penulis melakukan integrasi API dengan menggunakan metode

GET terhadap *endpoint* data statistik yang telah disediakan oleh pihak mitra.

Sebelum proses integrasi dilakukan, penulis terlebih dahulu melakukan pengecekan terhadap *response* yang diberikan oleh *endpoint* untuk memastikan kesesuaian struktur dan format data dengan kebutuhan tampilan yang diinginkan oleh klien.

Setelah memahami struktur *response* dari API tersebut, penulis menggunakan WPGetAPI, yaitu salah satu *plugin* WordPress yang memungkinkan integrasi langsung dengan *endpoint* API eksternal. Di dalam WPGetAPI, penulis melakukan konfigurasi awal seperti menetapkan *Unique* ID, *Base* URL, *Method*, *Endpoint*, serta *Result Format* yang disesuaikan menjadi PHP *Array Data* agar mudah diolah dalam WordPress.

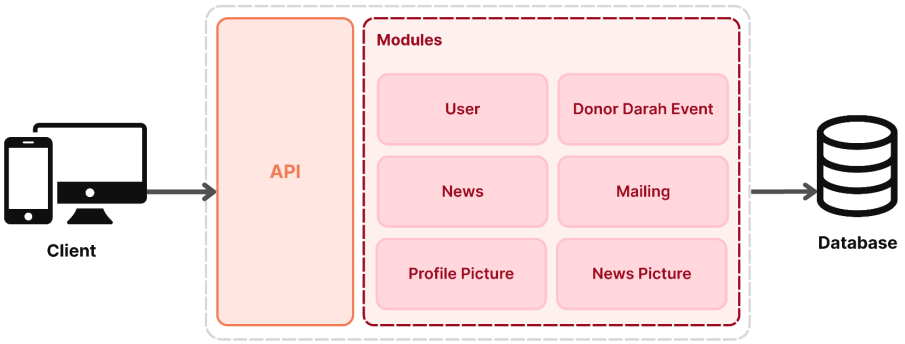
Setelah selesai melakukan *setup*, tahap berikutnya adalah menyalin *template tag* dari WPGetAPI (wpgetapi_endpoint('unique_id', 'endpoint', array('debug' => false));) ke Code Snippet *plugin* untuk melakukan *fetching* data. Kemudian, *tag* dari Code Snippet disisipkan ke dalam elemen Shortcode dari Elementor agar data dapat ditampilkan di halaman website. Tahap terakhir, penulis melakukan *styling* tampilan menyesuaikan dengan desain untuk menampilkan data statistik yang dapat dilihat pada gambar 3.8 *Responsive* di Layar yang berbeda (layar tablet).

3.2.3 Pengembangan *Backend* Rumah Donor

Dalam proyek pengembangan *backend* Rumah Donor, penulis berkolaborasi dengan satu orang *developer backend* lainnya untuk membangun dan mengelola sistem *backend* yang akan digunakan oleh aplikasi *mobile* Rumah Donor serta *Content Management System* (CMS) Rumah Donor. Proyek ini sebelumnya telah diinisialisasi oleh developer lain dengan menggunakan NestJS sebagai Node.js *framework* dan PostgreSQL sebagai sistem *database*. Dalam proses pengembangan lebih

lanjut, penulis menambahkan beberapa *library* utama yang dibutuhkan, antara lain Prisma sebagai ORM untuk mempermudah interaksi dengan *database*, Nodemailer sebagai pengiriman *email* melalui SMTP, serta bcrypt untuk melakukan *hashing* pada *password user*.

Struktur arsitektur *backend* Rumah Donor menggunakan pendekatan *Modular Monolith Architecture*, yaitu pendekatan di mana seluruh sistem dibangun dan dijalankan dalam satu *codebase* yang sama, namun dipecah ke dalam modul-modul terpisah berdasarkan *domain* bisnis tertentu. Dengan kata lain, *backend* tetap berbentuk satu kesatuan aplikasi (*single deployable unit*), namun struktur kodenya dibagi menjadi bagian-bagian fungsional yang terorganisir, seperti modul *user*, *mailing*, *news*, dan lain-lain. Masing-masing modul bertanggung jawab atas fungsi spesifik yang sesuai dengan kebutuhan aplikasi.

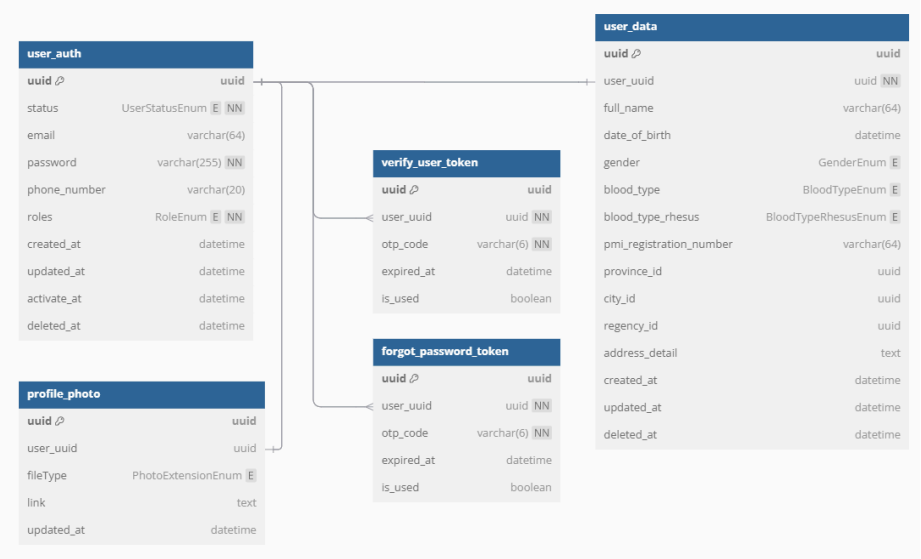


Gambar 3.18 *Modular Monolith Architecture*

- ***Entity Relationship Diagram***

Berikut ini adalah beberapa tabel yang sudah dibuat dan beberapa diantaranya masih dalam pengembangan (ditandai dengan kotak berwarna jingga).

a. Tabel *user_auth* dan relasinya



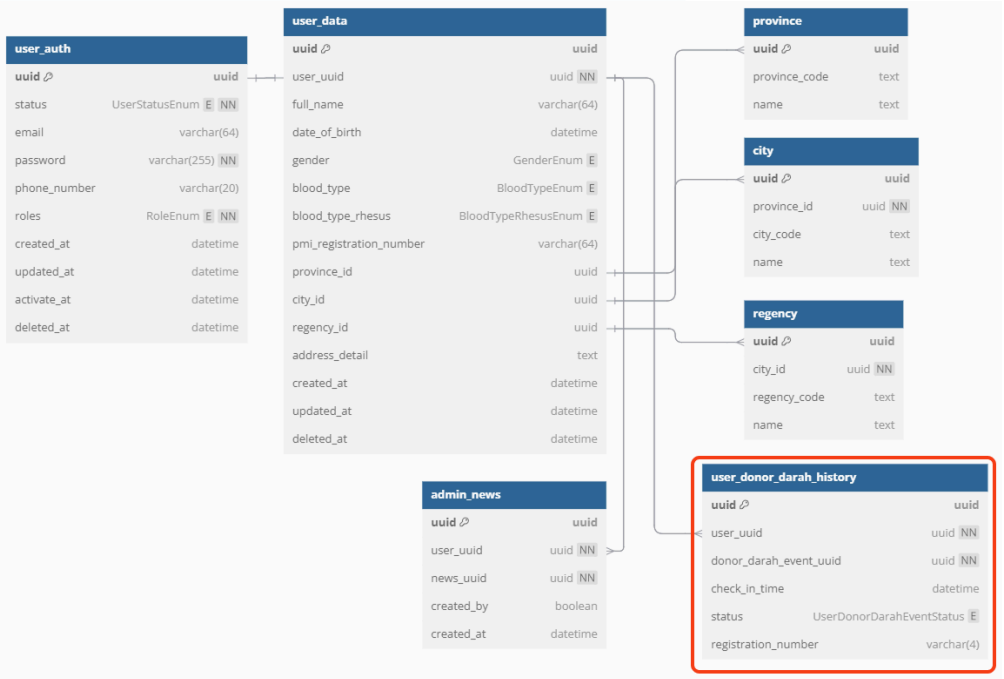
Gambar 3.19 Tabel *user_auth* dan relasinya

Tabel *user_auth* digunakan untuk autentikasi *user* seperti proses *login* dan registrasi *user*. Beberapa atribut di dalamnya dapat dilihat di gambar 3.19 Tabel *user_auth* dan relasinya. Beberapa relasi yang terjadi antara tabel *user_auth* dengan tabel lainnya sebagai berikut:

1. *One to One relation* dengan tabel *user_data* dan *profile_photo*. Tabel *user_data* digunakan untuk menyimpan berbagai atribut yang berkaitan dengan profil pengguna. Sedangkan, tabel *profile_photo* digunakan untuk menyimpan foto dari pengguna dalam bentuk *link*.

2. *One to Many relation* dengan tabel *verify_user_token* dan *forgot_password_token*. Tabel *verify_user_token* digunakan untuk menyimpan OTP (*One-Time Password*) saat pengguna melakukan registrasi. OTP digunakan untuk proses verifikasi akun pengguna yang telah teregistrasi dan mengubah status pengguna dari *REQUEST* menjadi *ACTIVE*. Sedangkan, tabel *forgot_password_token* digunakan untuk menyimpan OTP saat pengguna memilih menu *forgot password*.

b. Tabel *user_data* dan relasinya



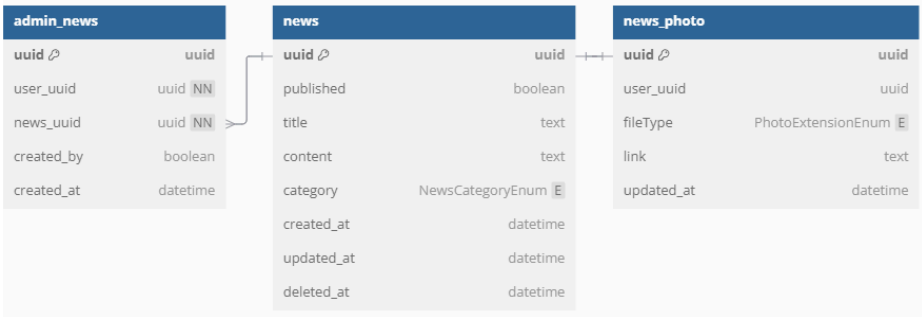
Gambar 3.20 Tabel *user_data* dan relasinya

Tabel *user_data* digunakan untuk menyimpan data personal pengguna. Beberapa atribut di dalamnya dapat dilihat di gambar 3.20 Tabel *user_data* dan relasinya.

Beberapa relasi yang terjadi antara tabel *user_auth* dengan tabel lainnya sebagai berikut:

1. *One to One relation* dengan tabel *user_auth*. Tabel *user_auth* digunakan untuk menyimpan berbagai atribut yang berkaitan dengan autentikasi pengguna.
2. *One to Many relation* dengan tabel *user_donor_darah_history*, *admin_news*, *province*, *city*, dan *regency*. Tabel *user_donor_darah_history* digunakan untuk menyimpan data riwayat pengguna registrasi *event* dari donor darah. Tabel *admin_news* digunakan untuk menyimpan riwayat pengguna dengan *role admin* yang membuat dan mengedit *news*. Tabel *province*, *city*, dan *regency* menyimpan nama-nama wilayah Indonesia yang digunakan untuk melengkapi data personal pengguna.

c. Tabel news dan relasinya

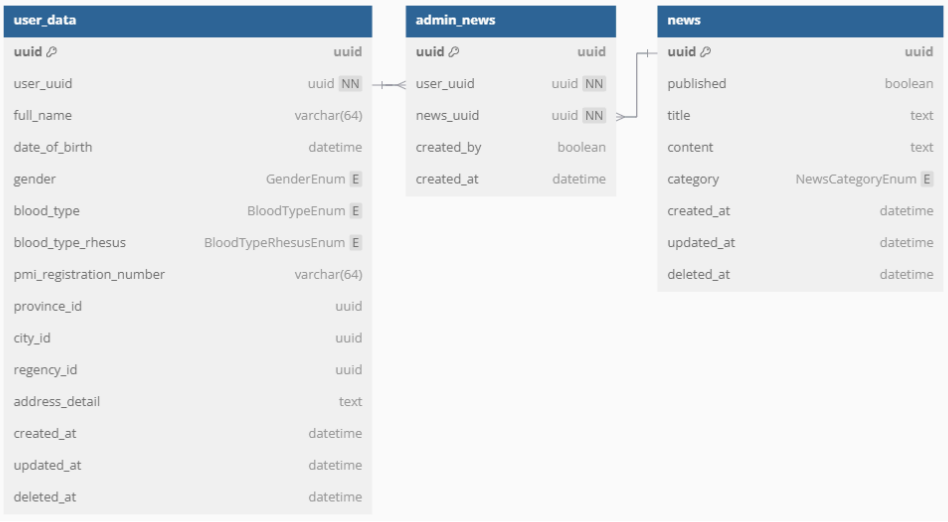


Gambar 3.21 Tabel *news* dan relasinya

Tabel *news* digunakan untuk menyimpan data *news* yang dibuat oleh pengguna dengan *role admin*. Beberapa atribut di dalamnya dapat dilihat di gambar 3.21 Tabel *news* dan relasinya. Beberapa relasi yang terjadi antara tabel *news* dengan tabel lainnya sebagai berikut:

1. *One to One relation* dengan tabel *news_photo*. Tabel *news_photo* digunakan untuk menyimpan gambar dari *news* dalam bentuk *link*.
2. *One to Many relation* dengan tabel *admin_news*. Tabel *admin_news* digunakan untuk menyimpan riwayat *news* yang dibuat atau diedit oleh pengguna dengan *role admin*.

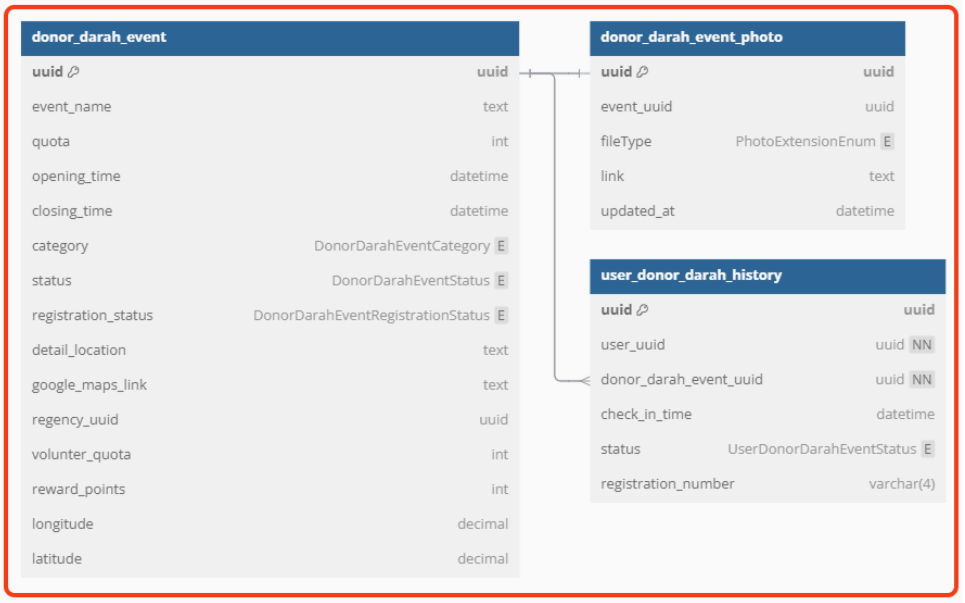
d. Tabel *admin_news* dan relasinya



Gambar 3.22 Tabel *admin_news* dan relasinya

Tabel *admin_news* merupakan *junction table* digunakan untuk menyimpan riwayat pembuatan dan pengeditan dari *news*. Beberapa atribut di dalamnya dapat dilihat di gambar 3.22 Tabel *admin_news* dan relasinya. Table *admin_news* mempunyai atribut tambahan salah satunya *created_by* dengan tipe data *boolean*, apabila nilainya true, pengguna tersebut diberikan tanda sebagai pembuat dari *news*, sedangkan apabila nilainya *false*, pengguna tersebut sebagai penyunting *news*.

e. Tabel *donor_darah_event* dan relasinya

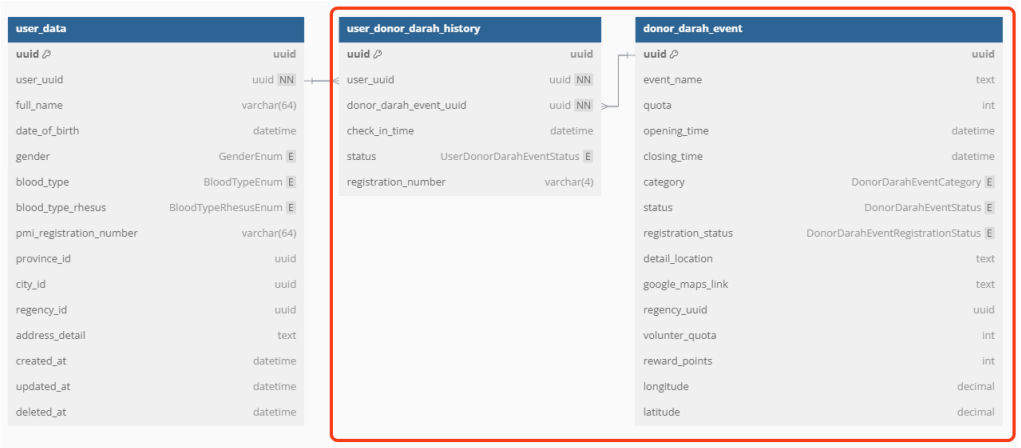


Gambar 3.23 Tabel *donor_darah_event* dan relasinya

Tabel *donor_darah_event* digunakan untuk menyimpan data dari *event* Rumah Donor Darah. Beberapa atribut di dalamnya dapat dilihat di gambar 3.23 Tabel *donor_darah_event* dan relasinya. Beberapa relasi yang terjadi antara tabel *donor_darah_event* dengan tabel lainnya sebagai berikut:

1. *One to One relation* dengan tabel *donor_darah_event_photo*. Tabel *donor_darah_event_photo* digunakan untuk menyimpan gambar dari *news* dalam bentuk *link*.
2. *One to Many relation* dengan tabel *user_donor_darah_history*. Tabel *user_donor_darah_history* digunakan untuk menyimpan riwayat registrasi *event* donor darah oleh pengguna.

f. Tabel *user_donor_darah_history* dan relasinya



Gambar 3.24 Tabel *user_donor_darah_history* dan relasinya

Tabel *user_donor_darah_history* merupakan *junction table* digunakan untuk menyimpan riwayat registrasi *event* donor darah dari pengguna. Beberapa atribut di dalamnya dapat dilihat di gambar 3.24 Tabel *user_donor_darah_history* dan relasinya.

- ***Module***

Beberapa *module* yang telah dan berada dalam tahap pengembangan, diantaranya sebagai berikut:

a. *User Module*

User module merupakan *module* pertama yang penulis buat dan kembangkan. *Module* tersebut secara keseluruhan menangani berbagai proses yang berkaitan erat dengan pengguna. Beberapa proses atau *service* yang terdapat di dalam modul tersebut diantaranya, yaitu *register* akun pengguna dengan *full_name*, *email*, dan *password*. Kemudian OTP akan dibuat dengan *expired date* dan juga status dengan nilai *false* (OTP belum digunakan). OTP dikirimkan melalui *mailer service* yang sudah dikonfigurasi

host, *port*, dan autentikasi dengan *username* serta *password*.

Proses aktivasi akun pengguna dilakukan dengan beberapa mekanisme pengecekan diantaranya, yaitu:

1. Status akun pengguna, jika bernilai *true* maka akun tersebut bisa langsung digunakan dan melakukan *login*.
2. Pengecekan apakah OTP tersebut tersedia atau tidak tersedia, karena sudah digunakan ataupun sudah kedaluwarsa.
3. Pengecekan *input* OTP pengguna apakah sesuai dengan OTP yang didapatkan melalui *email*.

OTP yang sudah digunakan atau kedaluwarsa akan dihapus dari database dalam rentang waktu tertentu dengan menggunakan menggunakan *cron job* dan pengguna dapat melakukan membuat OTP yang baru.

Terdapat juga *service* untuk pengguna dapat melakukan *reset password*. Mekanisme yang terjadi sama seperti pengguna melakukan aktivasi akun dengan OTP, namun ditujukan untuk *reset password* dan membutuhkan *password* yang digunakan oleh pengguna saat ini.

Selain proses aktivasi, modul ini juga menyediakan *service* untuk *forgot password*, yang menerapkan mekanisme verifikasi serupa dengan aktivasi akun. Fitur *login* dan *logout* juga diimplementasikan di modul ini, dengan menggunakan JWT (JSON Web Token) sebagai metode autentikasi yang disimpan di dalam *cookie* untuk menjaga keamanan sesi pengguna. Modul ini menyediakan *service* untuk melakukan *update* data pengguna yang dapat dilihat pada gambar 3.20 Tabel *user_data* dan relasinya.

b. Mailing Module

Mailing module merupakan modul pendukung yang saat ini digunakan untuk mengirimkan OTP aktivasi pengguna dan melakukan *reset password*. *Mailing module* yang dikembangkan menggunakan Nodemailer.

c. ***News Module***

News module merupakan modul yang digunakan untuk menangani fitur *news* dengan beberapa *service* yang diantaranya yaitu membuat *news*, *update news* dengan riwayat pengeditan dari *news*, *delete news* yang dapat dilakukan oleh pengguna dengan *role admin* dan *service* lainnya untuk menampilkan *news* di aplikasi *mobile* Rumah Donor.

d. Donor Darah Event Module

Donor Darah Event Module merupakan modul yang digunakan untuk menangani beberapa *service* berkaitan dengan *event* donor darah diantaranya, yaitu registrasi *event* Rumah Donor Darah dan pengguna mendapatkan kode registrasi yang digunakan untuk presensi kehadiran, melakukan pembatalan registrasi, menampilkan riwayat donor darah, menampilkan *event* donor darah berdasarkan lokasi terdekat dari pengguna, kategori, status registrasi dari *event*, dan *event status*, serta juga beberapa *service* yang hanya dapat dilakukan oleh Rumah Donor Darah *admin* yaitu membuat *event*, mengedit *event*, dan melakukan presensi terhadap kehadiran peserta. Untuk saat ini, *Donor Darah Event Module* sedang dalam tahap pengembangan.

e. Picture Module

Picture Module merupakan modul yang digunakan untuk menangani beberapa *service* berkaitan dengan gambar. *Picture Module* dibagi menjadi beberapa modul untuk setiap modul yang membutuhkan gambar, seperti *User Module* untuk gambar profil pengguna, *News Module* untuk gambar dari suatu *news*, dan *Rumah Donor Event Module* untuk gambar dari suatu *event* donor darah. Service yang terjadi diantaranya adalah *upload* gambar, *delete* gambar, mendapatkan gambar dalam bentuk *link* yang akan digunakan dari sisi *client*, dan juga menampilkan gambar dari *link* tersebut.

3.2.4 Pengembangan *Content Management System* Rumah Donor

Untuk mendukung serta melengkapi fungsi dari aplikasi *mobile* Rumah Donor, diperlukan sebuah situs *admin* atau yang lebih dikenal sebagai *Content Management System* (CMS). CMS ini ditujukan untuk digunakan oleh berbagai peran pengguna yang memiliki tugas dan tanggung jawab yang berbeda. Beberapa peran tersebut antara lain adalah *superadmin*, *admin*, *Rumah Donor Darah admin*, Rumah Gizi *admin*, *volunteer*, serta beberapa peran lainnya yang relevan dengan operasional Rumah Donor.

Dalam proses pengembangan CMS ini, penulis bertanggung jawab secara individu, mulai dari tahap awal hingga tahap pengembangan yang sedang berjalan saat ini. Proses diawali dengan sesi *briefing* dan diskusi bersama tim Rumah Donor, yang membahas cakupan halaman yang akan dikembangkan beserta fitur-fitur yang dibutuhkan untuk mendukung masing-masing peran pengguna. Salah satu tantangan yang dihadapi adalah tidak tersedianya desain antarmuka (*UI design*) dari tim desainer, sehingga penulis perlu menyusun struktur tampilan dan *user experience*

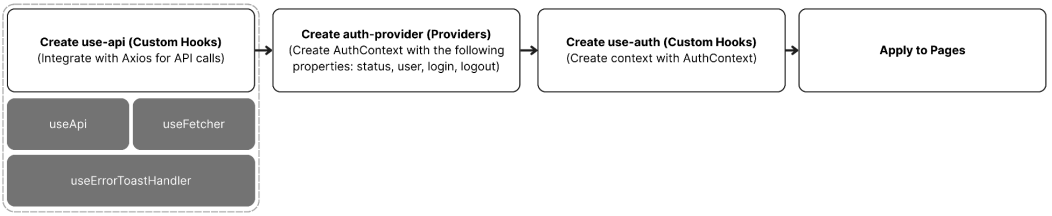
secara mandiri dengan tetap mengacu pada kebutuhan fungsional yang telah disampaikan.

- ***Technology Stack* dan *Library* CMS Rumah Donor**

  Penulis diberikan kebebasan oleh Bapak Budi Kurniawan di dalam menentukan *tech stack* dan berbagai *library* yang dibutuhkan untuk pengembangan CMS Rumah Donor. Namun, kebebasan yang diberikan tetap mengacu kepada tujuan dan kebutuhan dari pengembangan CMS tersebut. CMS yang dikembangkan berbasis *website*, maka dari itu penulis menggunakan React sebagai *frontend library* dengan Typescript sebagai bahasa pemrogramannya. React dipilih karena sebagai salah satu *modern component library* yang berbasis komponen dan mendapat *support* dari berbagai *library* sehingga membantu dalam proses pengembangan. Penggunaan Typescript ditujukan untuk memberikan *type safety* dan kepraktisan di dalam penulisan berbagai properti yang digunakan.

  Beberapa *library* yang saat ini penulis gunakan dibagi menjadi beberapa jenis diantaranya yaitu *library* untuk tampilan, *file project structure*, dan proses integrasi API. *Library* untuk tampilan menggunakan Tailwind CSS sebagai *CSS framework*, Shadcn sebagai *component library*, dan Lucide sebagai *library icon*. *Library* untuk *file project* menggunakan generouted/react-router dengan tujuan menyederhanakan proses *routing* dan memanfaatkan *file-based routing*. *Library* untuk proses integrasi API menggunakan axios untuk HTTP *request handle*, SWR untuk *data fetching*, dan Zod untuk validasi data antara CMS dan API.

- **Integrasi API**



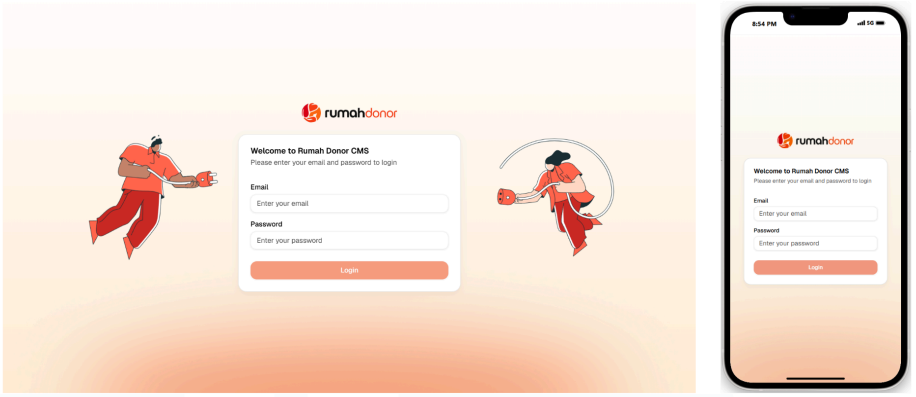
Gambar 3.25 Alur Proses Integrasi API

Sebelum melakukan pengembangan halaman-halaman yang dibutuhkan, penulis melakukan integrasi API dari *backend* yang telah penulis buat sebelumnya. Penulis membuat *custom hooks* use-api dengan menggunakan axios. use-api memberikan beberapa *export custom hook* diantaranya useApi digunakan untuk HTTP *method* seperti *GET*, *POST*, *PATCH*, dan *DELETE*, kemudian useFetcher digunakan untuk *fetching data* dengan SWR, dan useErrorToastHandler digunakan untuk memunculkan *error message* dengan *toast component* dari *library* sonner.

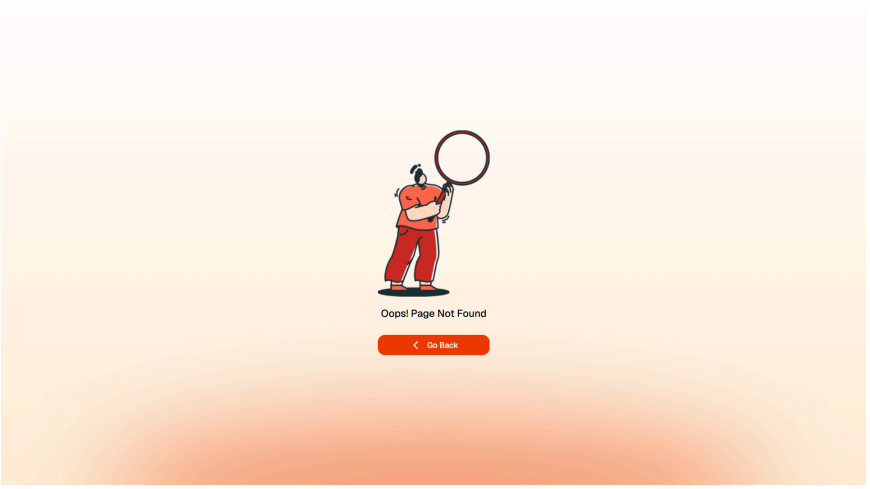
Kemudian, penulis membuat auth-provider untuk menangani proses autentikasi pengguna dan membuat AuthContext dengan beberapa properti diantaranya status, *user*, *login*, dan *logout*. AuthContext yang telah dibuat, kemudian dijadikan *custom hooks* yaitu use-auth, untuk dapat digunakan di berbagai halaman.

- ***Login* & 404 *Page***

Setelah API dari *backend* sudah berhasil terintegrasi ke dalam *codebase project* CMS, selanjutnya penulis memasuki tahap awal pengembanggan dengan mengembangkan halaman *Login* dan 404. Di dalam mengembangkan halaman *Login*, *Email* dan *Password* digunakan untuk proses autentikasi dan penulis menggunakan react-hook-form untuk membuat *Login form* tersebut dikombinasikan dengan Zod untuk proses validasi data.

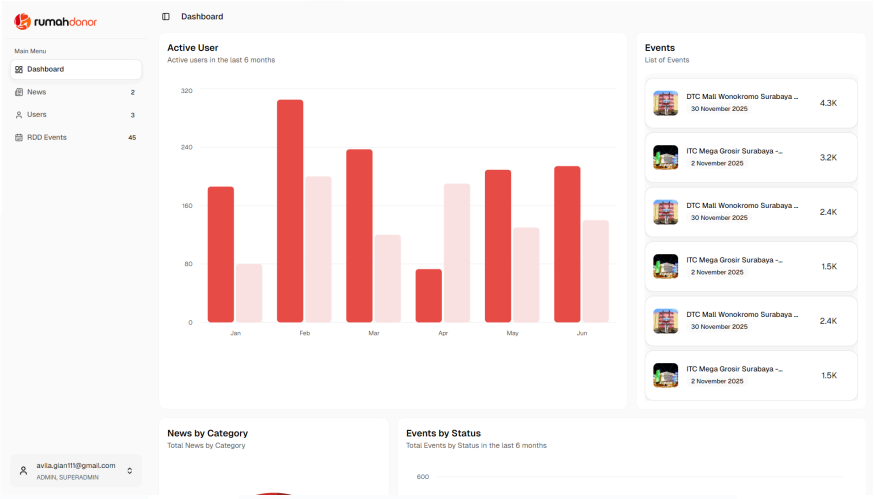


Gambar 3.26 Halaman Login



Gambar 3.27 Halaman 404

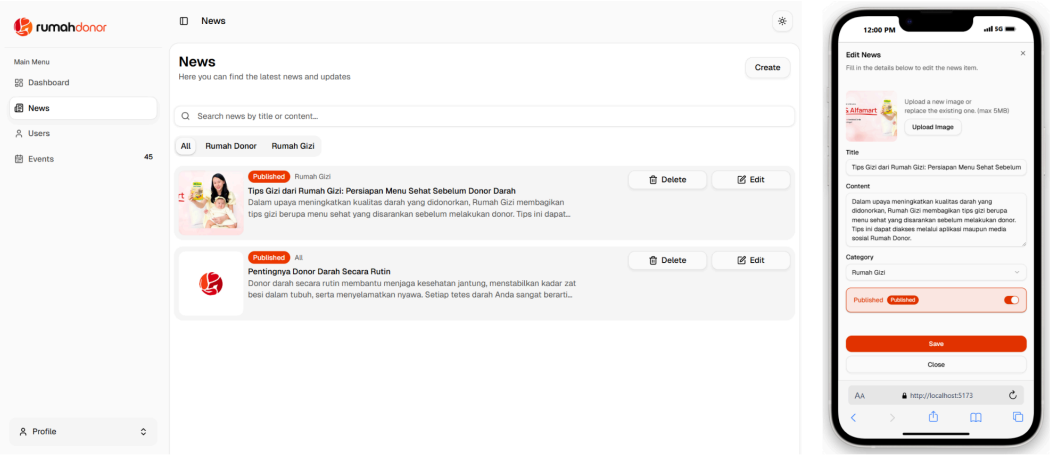
- ***Dashboard***



Gambar 3.28 Halaman *Dashboard*

Halaman *Dashboard* digunakan untuk menampilkan beberapa data yang dibutuhkan oleh beberapa *user role* seperti *superadmin*, *admin*, *Rumah Donor Darah admin*, dan *Rumah Gizi admin*. Beberapa data yang ditampilkan diantaranya, yaitu total pengguna berdasarkan status dari akun, total dan *list event* berdasarkan status, dan beberapa data lainnya yang kedepannya akan disesuaikan lebih lanjut berdasarkan kebutuhan masing-masing *user role*. Untuk saat ini halaman *Dashboard* dalam proses pengembangan.

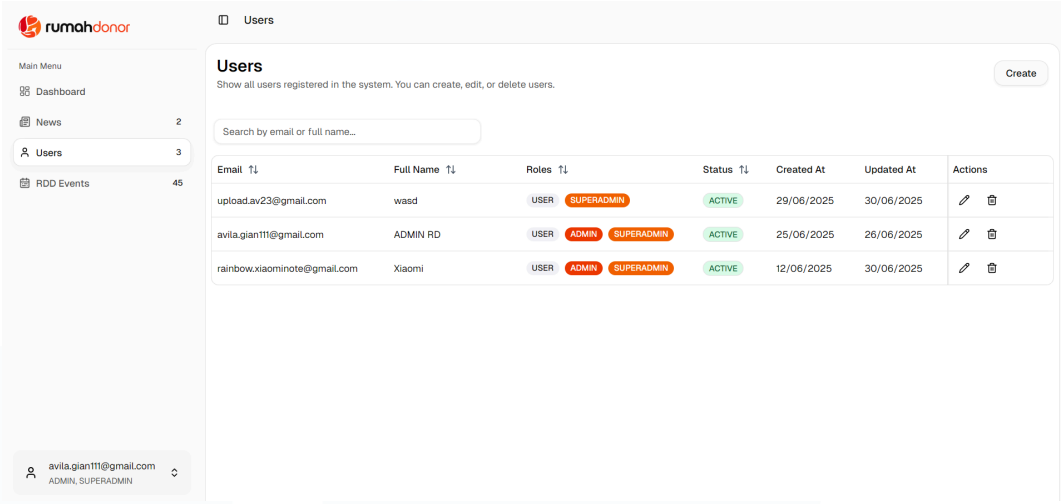
- ***News Management***



Gambar 3.29 Halaman *News Management*

Halaman *News Management* akan digunakan oleh pengguna dengan *role admin* untuk manajemen *news* yang akan dipublikasikan di aplikasi *mobile* Rumah Donor. Penulis mengembangkan fitur-fitur pendukung dengan melakukan implementasi dari API yang telah dibuat. Beberapa fitur diantaranya yaitu membuat, mengedit, menghapus, dan melihat riwayat *edit news*, serta beberapa fitur pendukung seperti *search* dan *sorting* berdasarkan kategori *news*.

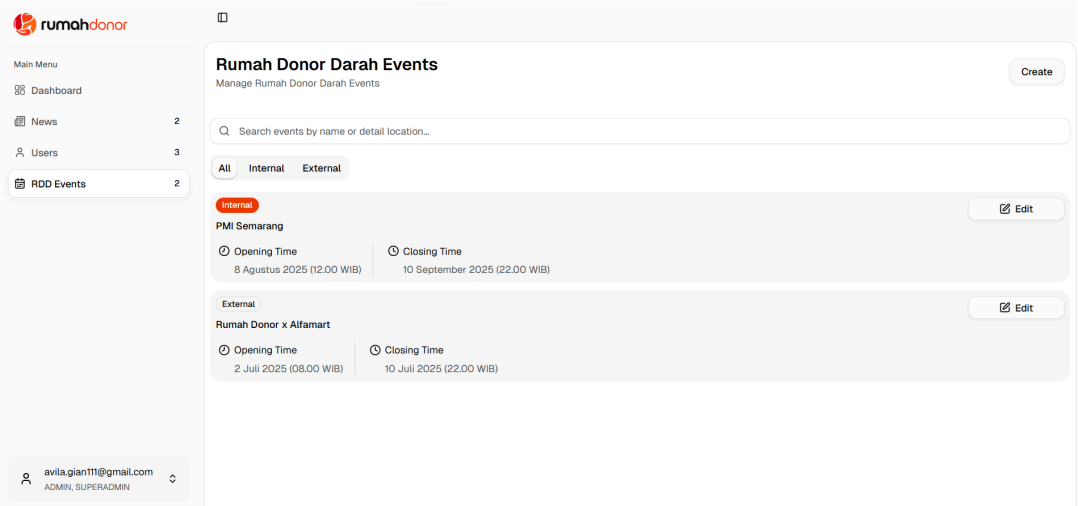
- ***User Management***



Gambar 3.30 Halaman *User Management*

Penulis menggunakan *library* TanStack untuk membuat tabel berisikan list dari pengguna, di mana pengguna yang ditampilkan disesuaikan berdasarkan role, sebagai contoh pengguna dengan *role superadmin* mempunyai akses untuk membuat dan mengedit *role* dari pengguna, sedangkan Rumah Donor Darah *admin* mempunyai akses untuk melihat dan melakukan presensi kehadiran dari pengguna yang terdaftar dalam *event* donor darah. Untuk saat ini halaman *User Management* dalam proses pengembangan.

- ***Event Rumah Donor Darah Management***



Gambar 3.31 Halaman *Event Rumah Donor Darah Management*

Halaman *Event Rumah Donor Darah Management* digunakan oleh pengguna terutama dengan *role* Rumah Donor Darah *admin* untuk melihat pengguna yang teregistrasi di dalam suatu *event* donor darah serta melakukan absensi kehadiran peserta. Kedepannya penulis akan melakukan implementasi fitur *scan QR Code* untuk kebutuhan presensi. Untuk saat ini halaman *Event Rumah Donor Darah Management* dalam proses pengembangan.

3.3 Kendala yang Ditemukan

- Menggunakan NestJS sebagai Node.js *framework* untuk *backend* merupakan pengalaman pertama penulis, sehingga membutuhkan waktu proses adaptasi terutama menyesuaikan struktur *file* yang telah ditentukan oleh NestJS.
- Pengembangan *website* menggunakan WordPress yang cukup kompleks juga merupakan pengalaman pertama penulis, sehingga membutuhkan waktu beberapa hari untuk mempelajari cara pengembangan *website* dengan WordPress sesuai standar yang telah ditentukan oleh perusahaan.
- Alur proses pengembangan yang tidak awam bagi penulis, di mana penulis sebagai *developer* membuat *user flow* untuk beberapa fitur. Hal ini

menimbulkan kebingungan, karena berdasarkan pengalaman penulis sebelumnya, penyusunan *user flow* umumnya merupakan tanggung jawab tim UI/UX *design*. Tim desain biasanya merancang alur dan struktur halaman terlebih dahulu untuk memastikan bahwa pengalaman pengguna sesuai dengan kebutuhan klien dan pengguna akhir, sebelum diserahkan kepada tim pengembang untuk dieksekusi. Ketidaksesuaian ini mempengaruhi kejelasan pembagian tugas dan memperlambat proses pengembangan di tahap awal.

3.4 Solusi atas Kendala yang Ditemukan

- Penulis memanfaatkan waktu disela-sela mengerjakan proyek lain untuk belajar dan *hands-on* membuat project sederhana menggunakan NestJS dengan membaca dokumentasi *official* serta *best practice tutorial* dari NestJS.
- Penulis berdiskusi dan berkolaborasi dengan Bapak Budi Kurniawan terkait proses pengembangan *website* dengan WordPress dari tahap *setup* sampai *deployment*.
- Penulis tetap bersikap terbuka dengan alur proses pengembangan tersebut dan juga turut memberikan masukan terkait alur kerja yang dirasa dapat disempurnakan. Salah satu saran yang disampaikan adalah mengenai pentingnya keterlibatan tim UI/UX *design* dalam proses awal pembuatan *user flow*. Penulis berpendapat bahwa apabila *user flow* ditentukan sepenuhnya oleh tim developer tanpa adanya referensi visual atau validasi dari desainer, terdapat risiko inkonsistensi terhadap prinsip desain yang *user-friendly*, serta kemungkinan pengulangan pengerjaan apabila *flow* yang dibuat ternyata tidak sesuai dengan ekspektasi klien atau kebutuhan pengguna akhir.