BAB 3
PELAKSANAAN KERJA MAGANG

3.1 Kedudukan dan Koordinasi

Selama kegiatan magang, pekerjaan dilakukan dalam divisi retail yang ada pada departemen IT, sebagai *intern web developer*. Divisi retail dipimpin oleh Bapak Shendy Harlim sebagai *product owner*, yang juga disertai oleh dua anggota lainnya. Secara berkala, kinerja divisi retail juga dievaluasi oleh CTO perusahaan.

Dalam pelaksanaan kerja magang, terutama pada divisi retail, koordinasi antar anggota dapat dilakukan secara langsung, melalui Telegram, atau Basecamp. Komunikasi secara langsung dan melalui Telegram dilakukan untuk membahas topik yang bersifat *urgent* dan memerlukan respons cepat. Namun untuk koordinasi yang berhubungan dengan *project management*, contohnya seperti pemberian *task*, *requirement*, jadwal, informasi penting, dan lainnya, dilakukan menggunakan Basecamp. Kemudian selama pekerjaan berlangsung, kontrol versi dikelola menggunakan sistem *branch* di GitHub dan aplikasi GUI Git, yaitu Fork.

Selain itu, divisi retail sendiri mengadakan rapat mingguan untuk berkoordinasi dengan semua anggota tim, dan dilakukan dua kali dalam seminggu, yaitu pada hari WFO dan WFH. Rapat pada hari WFO dilakukan secara *onsite* untuk membahas *progress*, kendala, dan *update* perusahaan secara keseluruhan. Sementara itu, rapat pada hari WFH, yang dilakukan melalui Google Meet, berfokus pada rencana kerja mingguan dan diskusi tugas-tugas terkait.

3.2 Tugas yang Dilakukan

Dari beberapa tugas yang telah dikerjakan selama masa magang, terdapat tiga tugas utama yang berkaitan dengan pengembangan layanan Whitelabel. Pada layanan ini, terdapat dua istilah pengguna, yaitu *tenant* dan agen. *Tenant*, dalam konteks ini adalah pemilik bisnis yang menggunakan Whitelabel untuk mengelola bisnis pembayaran digital-nya, dengan *branding* yang dapat disesuaikan sendiri. Sedangkan agen adalah pengguna aplikasi yang dimiliki *tenant*.

Dari ketiga tugas utama, tugas pertama adalah mengembangkan fitur promosi untuk produk prabayar PPOB yang memungkinkan *tenant* untuk mengatur *event* diskon pada Whitelabel-nya. Kemudian melakukan integrasi produk pascabayar Telkomsel Omni pada Whitelabel dengan memanfaatkan

beberapa *endpoint* yang disediakan oleh pihak Telkomsel. Tugas terakhir adalah mengembangkan fitur automasi migrasi antar *supplier tenant* pada situs web CMS, yang adalah situs web *supermaster* Whitelabel, untuk digunakan internal, dengan tujuan untuk meningkatkan kinerja *developer*.

Semua pekerjaan yang dilakukan disusun dalam sistem *cycle* yang digunakan oleh perusahaan, terutama pada departemen IT, sebagai sistem pengelolaan pekerjaan. Setiap *cycle* terdiri 6 minggu proses *development*, ditambah 2 minggu *buffer* untuk melakukan *planning* dan evaluasi tugas-tugas yang telah dilakukan. Setiap tugas yang dilakukan mengikuti alur sebagai berikut.

1. Tugas diberikan oleh PO melalui Basecamp, disertai dengan latar belakang, *requirement*, serta *due date* yang telah ditetapkan.
2. Setelah tugas selesai, *branch* yang digunakan selama proses pengembangan di Github dapat di-*open Pull Request* (PR) untuk proses *code review* oleh PO.
3. Pada tugas tertentu, akan dilakukan testing terlebih dahulu yang dilakukan oleh *developer* dan divisi *Security and Compliance*.
4. Ketika proses sebelumnya telah selesai, maka tugas yang telah dilakukan akan di-*release* ke *production*.

3.3 Uraian Pelaksanaan Magang

Rincian kerja magang di PT Indobest Artha Kreasi yang dilakukan selama 20 minggu diuraikan pada Tabel 3.1.

Tabel 3.1. Pekerjaan yang dilakukan tiap minggu selama pelaksanaan magang

Minggu Ke -	**Pekerjaan yang dilakukan**
1	Melakukan pengembangan dan testing *layout* agen baru, mengembangkan menu dinamis yang lebih fleksibel dan memperbaiki *bug* yang ada
2	Menyesuaikan *layout* halaman menu dinamis dan memperbaiki *bug* saat mengganti nama bagian, serta menerapkan menu dinamis untuk *layout* baru
Dilanjutkan pada halaman berikutnya	

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

Minggu Ke -	**Pekerjaan yang dilakukan**
3	Membuat *section* pada menu dinamis *draggable* dan menyesuaikan ukuran tulisan pada tampilan agen
4	Finalisasi tampilan dan fitur UI baru, dan memulai pengerjaan tugas pengaturan struk pascabayar lebih fleksibel
5	Menguji fitur struk pascabayar dan merancang komponen dan desain antarmuka (*wireframe*) untuk fitur promosi, serta penentuan alur transaksi promosi
6	Merancang desain antarmuka (*mockup*) untuk fitur promosi, implementasi *frontend* untuk halaman dan modal promosi, serta setup rute menuju halaman promosi
7	Menambahkan *countdown* promosi, membuat komponen transaksi promosi untuk pulsa, games, listrik, pulsa internasional, pertagas, e-money, dan TV prabayar, serta mengimplementasikan *endpoint* untuk mendapatkan produk promosi
8	Mengimplementasikan *endpoint* pengaturan promosi, menangani promosi tidak aktif, dan menambahkan *popup* promosi
9	Menambah validasi *form* promosi, memperbaiki beberapa bug pada fitur promosi
10	Menangani validasi *file popup* dan tampilan pada kartu promosi, serta menyesuaikan ukuran *font* untuk tampilan *mobile*
11	Menyinkronkan status produk promosi dengan status produk dari *supplier* dan membuka fitur pembatalan transaksi untuk CS
12	Memperbaiki bug transaksi produk promosi dan membuat generator *sitemap* untuk situs web *tenant*
13	Melakukan setup proyek Mobilepulsa untuk memperbaiki masalah *login*, dan menambahkan validasi untuk nilai deposit minimum dinamis serta pengaturan *topup* khusus
14	Mengerjakan integrasi produk Telkomsel Omni dengan Whitelabel, menambahkan *endpoint* untuk memilih produk dan menghasilkan kode pembayaran, serta menguji beberapa *test-case*
Dilanjutkan pada halaman berikutnya	

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

Minggu Ke -	Pekerjaan yang dilakukan
15	Mengerjakan fitur migrasi antar *supplier*, memperbaiki bug Telkomsel Omni
16	Melakukan migrasi klien otomatis, memperbaiki tombol *flash* rusak, menambah detail produk Telkomsel Omni
17	Memperbaiki masalah notifikasi, serta mengubah nilai *default* pembaruan harga otomatis untuk *tenant* baru
18	Menangani masalah *database* pada lingkungan *staging*, menyesuaikan pesan *error* aplikasi, dan menambah indikator visual untuk proses migrasi antar *supplier*
19	Menambah *alert* untuk *session timeout*, menambah pilihan *sort* produk prabayar, memperbaiki masalah pengingat deposit dan notifikasi lainnya
20	Memperbaiki masalah notifikasi massal yang tidak masuk dan notifikasi grup yang tidak dapat terkirim ke beberapa penerima

Dari rangkuman singkat mingguan yang tertera pada tabel diatas, berikut ini adalah uraian lengkap untuk setiap tugas yang dikerjakan, mulai dari *requirement*, perancangan sistem, hingga hasil implementasinya.

3.3.1 Fitur Promosi Prabayar PPOB

Fitur promosi yang dibuat ditujukan untuk meningkatkan minat dan tingkat penjualan produk dengan adanya diskon. Fitur ini dibatasi untuk hanya tersedia pada produk prabayar PPOB. Promosi ini akan menjadi salah satu fitur yang bisa digunakan dan diatur sesuai kebutuhan, dimana setiap *tenant* dapat mengatur judul, periode, gambar *popup*, status promosi, produk, dan harga produk promosi pada admin panel. Diluar dari pengaturan promosi oleh *tenant*, diperlukan juga tampilan yang dapat diakses agen untuk dapat mengetahui dan melakukan transaksi produk-produk promosi. Oleh karena itu, fokus tugas ini ada pada pembuatan tampilan produk promosi untuk agen yang mengakses aplikasi Whitelabel *tenant*. Fitur ini dikerjakan pada *project* Whitelabel yang menggunakan Laravel 5.8, Tailwind, dan Vue 2, dimana proses perancangan *design wireframe* dan *high-fidelity* dilakukan menggunakan Figma.

A. *Requirement*

Pada fitur promosi yang dirancang, ada beberapa *requirement* yang diperlukan agar fitur promosi ini dapat berjalan dengan baik dan sesuai tujuan.

A.1 Bagian Promosi

Untuk menarik minat agen atau pelanggan, produk promosi perlu ditampilkan pada halaman utama situs web dan aplikasi Whitelabel agar dapat dilihat langsung ketika Whitelabel baru dibuka. Namun, pada bagian promosi yang ditampilkan pada halaman utama, tidak semua produk akan ditampilkan. Dari banyaknya produk promosi yang ada, hanya ada 5 produk yang akan ditampilkan, yaitu 4 produk terlaris dan 1 produk *out of stock*, yang diurutkan dari produk paling laris. Selain itu, bagian promosi juga akan menampilkan informasi-informasi mengenai promosi tersebut, meliputi judul, periode, *countdown*, dan tombol untuk melihat produk promosi lainnya. Adanya bagian promosi ini dibuat agar transaksi bisa berjalan lebih cepat, sekaligus memberikan kesan urgensi agar segera membeli sebelum produk habis atau sebelum periode promosi berakhir.

A.2 Halaman Promosi

Bagian promosi yang ada pada halaman utama hanya menampilkan beberapa produk terlaris, oleh karena itu dibutuhkan halaman tersendiri yang dapat memperlihatkan semua produk promosi yang tersedia. Konten pada halaman promosi sama seperti yang ada pada bagian promosi, dan hanya berbeda pada jumlah produk yang ditampilkan dan bentuk *card* setiap produk promosi.

A.3 Popup Promosi

Sama seperti *e-commerce* lainnya, ketika ada *event* promosi yang sedang berlangsung, diperlukan adanya *popup* yang dapat menyambut pelanggan ketika Whitelabel baru dibuka. *Popup* promosi ditampilkan untuk memberi tahu pelanggan bahwa ada promosi yang sedang berlangsung, sehingga pelanggan tidak melewatkan penawaran meskipun tidak melihat bagian promosi di halaman utama. Gambar pada *popup* dapat diatur oleh *tenant* dan bersifat opsional. *Popup* promosi hanya akan muncul satu kali sehari pada setiap *device*.

A.4 *Flow* Transaksi Produk Promosi

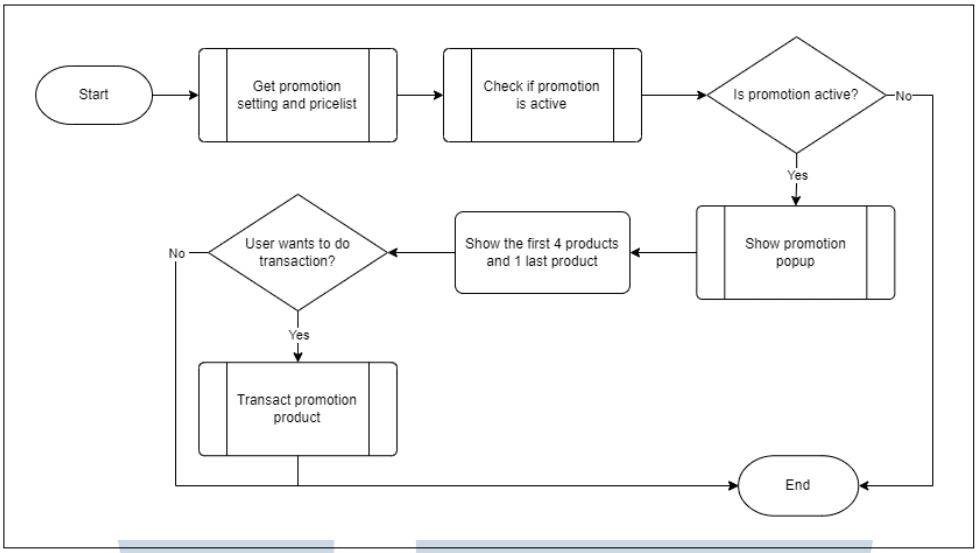
Flow transaksi produk promosi dibuat sedikit berbeda dari produk reguler. Pada transaksi produk reguler, pelanggan memasukkan nomor telepon terlebih dahulu, kemudian sistem menampilkan daftar produk yang sesuai dengan operator nomor tersebut. Setelah memilih produk, pelanggan diminta untuk memasukkan PIN atau kata sandi untuk menyelesaikan transaksi. Namun pada produk promosi, pelanggan langsung memilih produk terlebih dahulu. Setelah produk dipilih, pelanggan baru diminta untuk memasukkan nomor telepon, yang kemudian divalidasi oleh sistem untuk memastikan kesesuaian dengan operator produk promosi tersebut. Jika nomor valid, pelanggan diminta untuk memasukkan PIN atau kata sandi sebagai langkah terakhir untuk menyelesaikan transaksi.

B. Perancangan Sistem

Perancangan sistem untuk fitur promosi produk prabayar terbagi dalam beberapa bagian sesuai dengan *requirement* yang ada. *Flowchart* digunakan untuk menggambarkan perancangan sistem, sementara rancangan antarmuka hanya dibuat untuk beberapa *requirement* tertentu.

B.1 Bagian Promosi

Pada bagian promosi yang ada pada halaman utama Whitelabel, terdapat alur proses dari pengambilan data promosi hingga penampilan informasi dan produk yang ada.



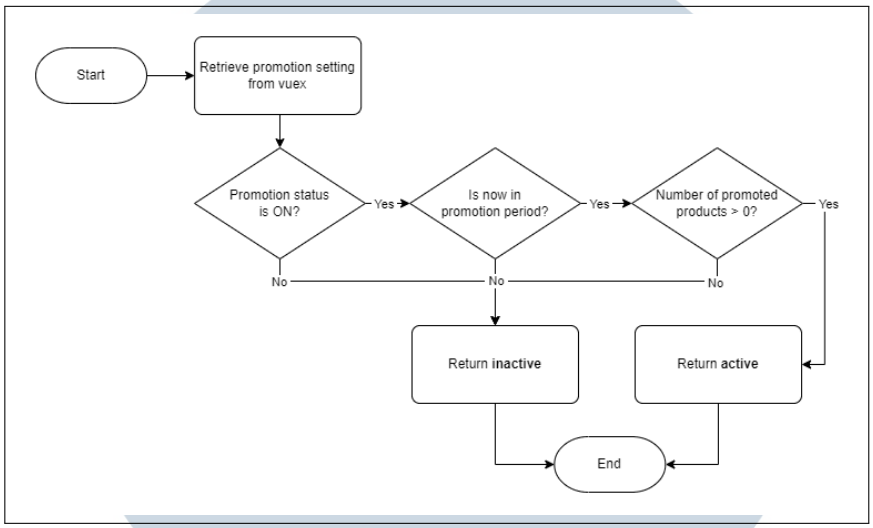
Gambar 3.1. *Flowchart Promo Section*

Gambar 3.1 menunjukkan *flowchart* utama untuk menampilkan bagian promosi. Proses dimulai dengan pengambilan pengaturan dan daftar produk promosi. Setelah pengaturan promosi berhasil diambil, sistem perlu memeriksa apakah promosi tersebut sedang aktif. Jika promosi aktif, sistem akan mencoba menampilkan *popup*. Namun, tampilan *popup* ini tergantung pada beberapa kondisi lain yang akan dijelaskan lebih lanjut dalam modul terkait. Selanjutnya, dari daftar harga yang telah diambil, sistem akan menampilkan empat produk terlaris serta satu produk yang sudah habis pada bagian promosi. Jika pelanggan memutuskan untuk melakukan transaksi pada produk promosi yang tersedia, proses akan dilanjutkan dengan modul transaksi yang akan dijelaskan lebih lanjut di bagian lain.

Gambar 3.2. *Flowchart* modul *Get promotion setting and pricelist*

Proses yang terjadi pada modul untuk mengambil pengaturan dan daftar produk promosi tertera pada Gambar 3.2 diatas. Pertama, sistem akan memanggil *endpoint* untuk mengambil daftar produk promosi sekaligus pengaturan promosi. Daftar produk promosi yang berhasil diambil kemudian akan diurutkan, dimulai dari produk terlaris hingga produk yang sudah habis. Peringkat produk terlaris ditentukan berdasarkan rasio atribut *sold_qty / stock* dari setiap produk yang ada. Daftar produk dan pengaturan ini kemudian disimpan pada Vuex agar nantinya

dapat diakses melalui *frontend*. Vuex sendiri merupakan manajemen *state* yang disediakan untuk *project* Vue, yang berfungsi sebagai penyimpanan terpusat untuk semua komponen di dalamnya [10].



Gambar 3.3. *Flowchart* modul *Check if promotion is active*

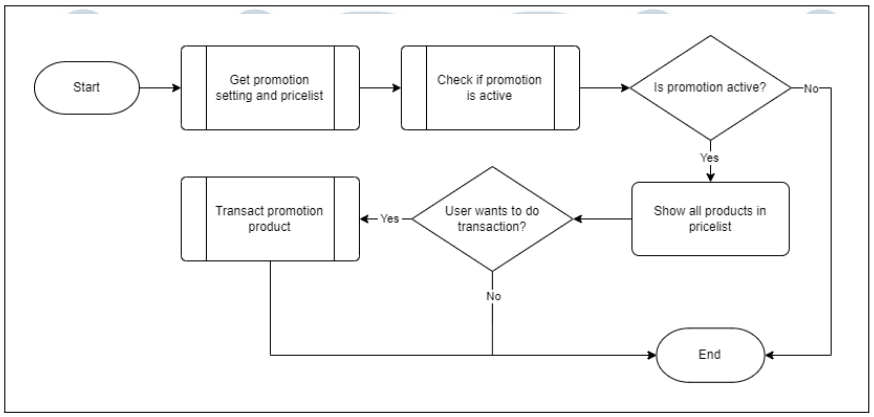
Meskipun daftar produk dan pengaturan promosi sudah disimpan, tetap perlu dilakukan pengecekan apakah promosi tersebut sedang aktif atau tidak. Proses ini digambarkan pada Gambar 3.3. Proses dimulai dengan mengambil pengaturan promosi yang tersimpan di Vuex, di mana status promosi akan diperiksa apakah bernilai on atau tidak. Jika tidak, modul akan mengembalikan nilai *inactive*, yang berarti promosi sedang tidak aktif. Namun, jika nilai status adalah on, akan dilakukan pengecekan lebih lanjut terhadap periode promosi. Apabila tanggal dan waktu akses halaman berada dalam rentang periode promosi (berdasarkan tanggal mulai dan selesai), maka selanjutnya perlu dipastikan bahwa jumlah produk promosi setidaknya ada satu agar modul mengembalikan nilai *active*. Sebaliknya, jika tidak ada produk dalam daftar promosi, atau periode promosi belum dimulai atau sudah berakhir, sistem akan mengembalikan nilai *inactive*.

Gambar 3.4. Rancangan antarmuka Bagian promosi

Terlepas dari *flowchart* yang ada untuk bagian promosi, Gambar 3.4 merupakan rancangan antarmuka. Pada rancangan yang ada, terdapat beberapa informasi yang hendak ditampilkan pada bagian promosi, yaitu judul, periode, daftar produk, serta tombol untuk mengakses halaman promosi.

B.2 Halaman Promosi

Selain dari bagian promosi yang ada pada halaman utama Whitelabel, daftar produk yang tersedia dan informasi mengenai promosi tersebut juga diletakkan di dalam halaman tersendiri. Halaman ini dapat diakses melalui tombol yang ada pada bagian promosi, atau dari *popup* yang muncul jika *tenant* mengatur *popup* promosi.



Gambar 3.5. *Flowchart Promo Page*

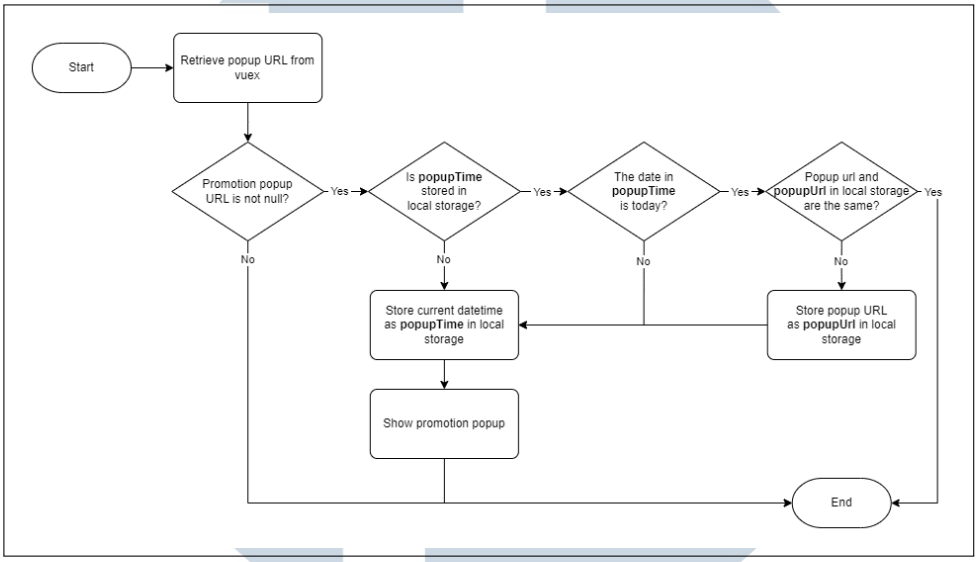
Flowchart yang ada pada Gambar 3.5 menunjukkan alur yang ada pada halaman promosi. Secara keseluruhan, alur pada halaman promosi mirip dengan yang ada pada bagian promosi (Gambar 3.1). Pada halaman promosi, *popup* tidak akan muncul, dan berbeda dari bagian promosi, halaman ini akan menampilkan semua produk yang sedang diskon diurutkan dari produk terlaris.

Gambar 3.6. Rancangan antarmuka Halaman promosi

Perbedaan lainnya dengan bagian promosi terlihat dari rancangan antarmuka halaman promosi yang dapat dilihat pada Gambar 3.6. Halaman promosi dirancang untuk menampilkan informasi lebih seperti sisa saldo deposit agen, judul promosi, periode, *countdown*, *search bar*, dan filter jenis produk.

B.3 Popup Promosi

Popup promosi merupakan salah satu sarana penarik minat pelanggan yang bersifat opsional bagi tenant untuk mengaturnya.

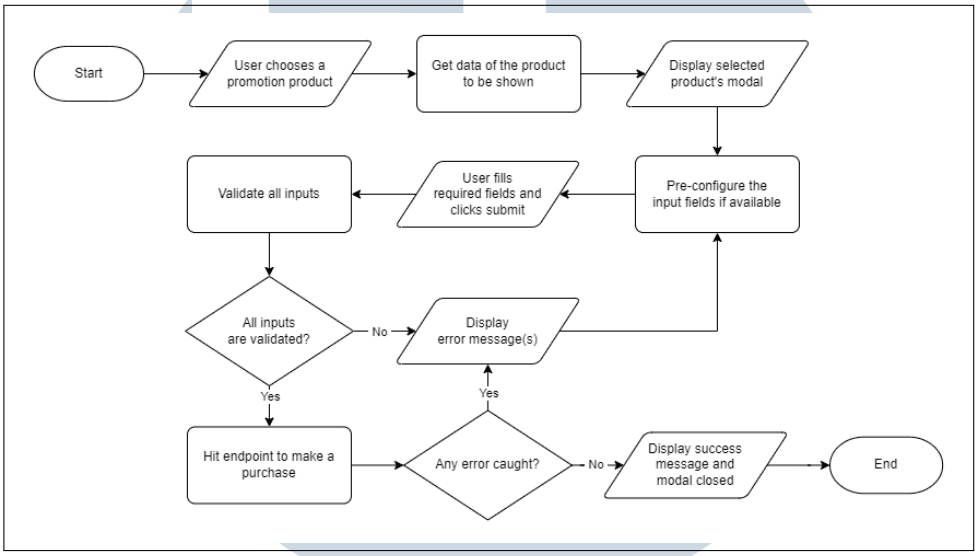


Gambar 3.7. *Flowchart* modul *Show promotion popup*

Gambar 3.7 menunjukkan alur modul untuk menunjukkan *popup* yang akan muncul pada halaman utama, yang dipanggil pada flowchart dalam Gambar 3.1. Pertama, dari data pengaturan promosi yang tersimpan di Vuex, sistem mengambil URL gambar *popup* yang telah diatur oleh *tenant* di admin panel. Jika URL tidak tersedia, proses dihentikan dan *popup* tidak akan muncul. Tetapi jika URL ada, sistem memeriksa keberadaan data *popupTime* di *local storage* perangkat. Ketika data belum ada, waktu saat ini disimpan ke *popupTime*, dan *popup* ditampilkan. Namun jika *popupTime* sudah ada, akan dicek apakah hari pada *popupTime* sama dengan tanggal saat ini untuk memastikan *popup* hanya muncul sekali sehari. Jika berbeda, tanggal baru akan disimpan, dan *popup* dimunculkan. Namun, jika tanggalnya masih sama, perlu dilakukan pengecekan perubahan URL *popup* dengan membandingkannya dengan data *popupUrl* di *local storage*. Jika URL masih sama, *popup* tidak akan ditampilkan lagi. Sebaliknya, jika URL berbeda, URL baru akan disimpan ke *local storage*, dan *popup* akan ditampilkan dengan URL terbaru.

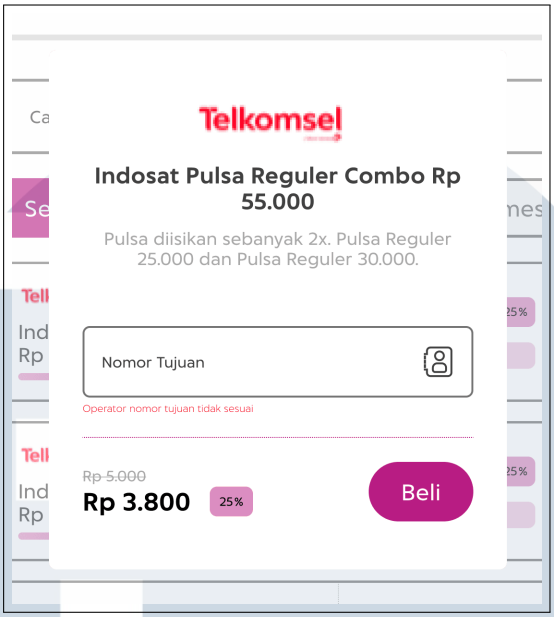
B.4 *Flow* Transaksi Produk Promosi

Produk-produk promosi yang tampil di bagian promosi dan halaman promosi dapat ditransaksikan sesuai dengan yang ada pada *flowchart* di Gambar 3.1 dan 3.5.



Gambar 3.8. *Flowchart* modul *Transact promotion product*

Rincian alur transaksi produk promosi ada pada Gambar 3.8. Awalnya pelanggan akan memilih *card* produk promosi yang diminati. Setelah sistem mengambil data yang berkaitan dengan produk yang dipilih, maka sebuah modal berisi informasi produk akan ditampilkan. Selain informasi produk, terdapat *input field* yang perlu diisi oleh pengguna. *Input field* ini dapat bervariasi tergantung pada jenis produk dan akan otomatis terisi jika terdapat data pelanggan yang tersimpan. Setelah pelanggan mengisi semua *input field* yang diperlukan, sistem akan melakukan validasi terhadap data yang dimasukkan. Jika validasi gagal, pesan *error* akan ditampilkan, dan pelanggan perlu mengoreksi datanya. Namun, jika validasi berhasil, sistem akan memanggil *endpoint* untuk memproses transaksi berdasarkan informasi produk dan data yang diberikan pelanggan. Apabila transaksi berhasil, pesan konfirmasi akan ditampilkan dan modal akan ditutup. Jika transaksi gagal, pesan *error* dari respons akan muncul untuk memberi tahu pelanggan apa yang perlu dilakukan dan memintanya mencoba kembali.



Gambar 3.9. Rancangan antarmuka modal produk promosi

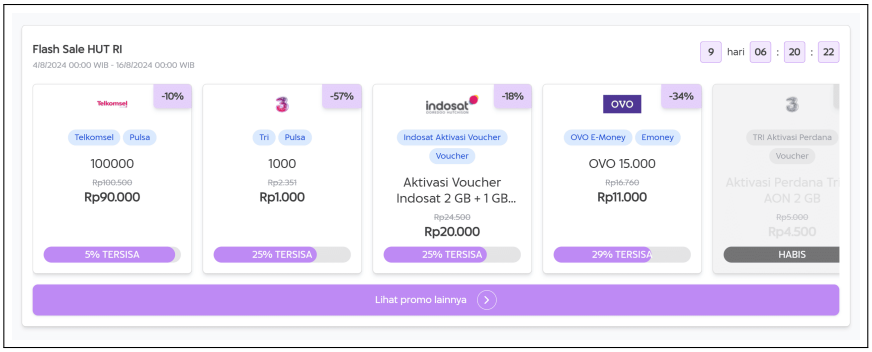
Rancangan antarmuka modal produk promosi yang memungkinkan pelanggan memasukkan data yang diperlukan untuk kepentingan bertransaksi digambarkan pada Gambar 3.9. Pada modal yang ditampilkan, data produk promosi yang ditampilkan meliputi logo operator, nama produk, deskripsi, harga sebelum dan sesudah diskon, serta besar diskon dalam bentuk persentase.

C. Implementasi Sistem

Hasil implementasi sistem yang telah dirancang dijelaskan berdasarkan alur dari sudut pandang sebagai pelanggan Whitelabel. Terdapat empat tahapan dan tampilan utama yang akan ditemui pelanggan, mulai dari mendapat informasi mengenai adanya promosi yang sedang berlangsung, hingga proses transaksi berhasil dilakukan.

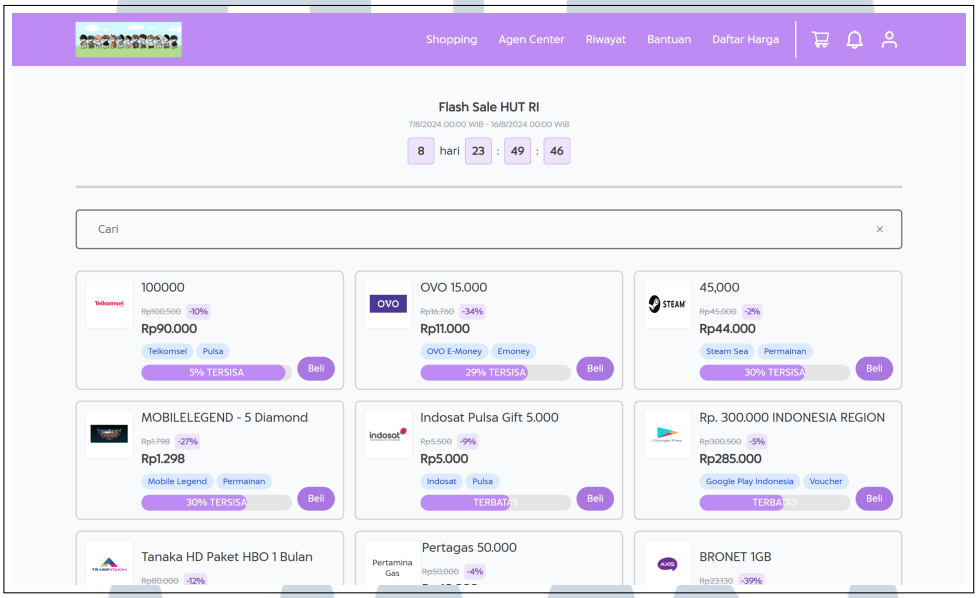
Gambar 3.10. Implementasi tampilan popup promosi

Saat pelanggan mengakses Whitelabel, tahap pertama yang akan dijumpai adalah *popup* promosi yang digambarkan pada Gambar 3.10. *Popup* promosi, seperti yang sudah dijelaskan dalam bagian perancangan sistem, mungkin saja tidak muncul karena tidak diatur oleh *tenant*, atau karena sudah muncul hari itu.



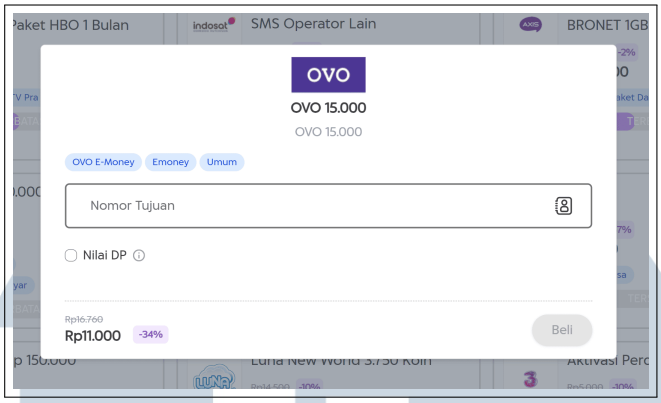
Gambar 3.11. Implementasi tampilan bagian promosi

Tampilan yang ada pada Gambar 3.11 merupakan bagian promosi yang ada pada halaman utama jika pelanggan menutup atau tidak menjumpai *popup* promosi. Terdapat beberapa perubahan tampilan dari rancangan antarmuka yang telah dibuat. Pada implementasinya, ditambahkan komponen *countdown* di bagian kanan atas, dan desain kartu setiap produk juga mengalami penyesuaian, termasuk penambahan *badge* nama operator serta jenis produk. Selain itu, ada perbedaan warna dengan rancangan awal karena warna yang digunakan akan mengikuti warna *primary* milik masing-masing *tenant*.



Gambar 3.12. Implementasi tampilan halaman promosi

Kemudian, jika pelanggan masuk ke dalam halaman promosi, pelanggan akan dihadapkan dengan tampilan seperti yang ada pada Gambar 3.12. Beberapa perubahan yang diterapkan mencakup penghilangan komponen sisa saldo deposit agen dan filter tipe produk, serta penyesuaian desain kartu produk. Perubahan ini diterapkan berdasarkan saran dan masukan dari pengembang lainnya.



Gambar 3.13. Implementasi tampilan modal transaksi produk promosi

Gambar 3.13 merupakan implementasi tampilan modal yang akan muncul saat pelanggan ingin melakukan transaksi pada produk promosi. Perubahan yang ada dari rancangan antarmuka-nya adalah penambahan *badge* nama operator, tipe produk, dan sub-tipe produk.

3.3.2 Integrasi Telkomsel Omni

Telkomsel Omni memiliki alur transaksi yang berbeda dari produk pascabayar lainnya, memungkinkan pelanggan memilih dan membayar paket secara terpisah melalui situs web Telkomsel dan mitra. Setelah pelanggan memilih paket di situs Telkomsel, sistem akan menghasilkan kode unik yang dapat digunakan untuk pembayaran di mitra. Awalnya, integrasi Telkomsel Omni di Whitelabel hanya mencakup fitur pembayaran dari kode yang telah dibuat. Pelanggan perlu memilih paket dan mendapatkan kode dari situs Telkomsel terlebih dahulu, lalu memasukkan kode tersebut di Whitelabel untuk menyelesaikan transaksi. Namun untuk meningkatkan pengalaman pelanggan, maka Telkomsel Omni ingin diintegrasikan secara keseluruhan ke dalam Whitelabel agar pelanggan tidak perlu berpindah-pindah situs web untuk melakukan sebuah transaksi.

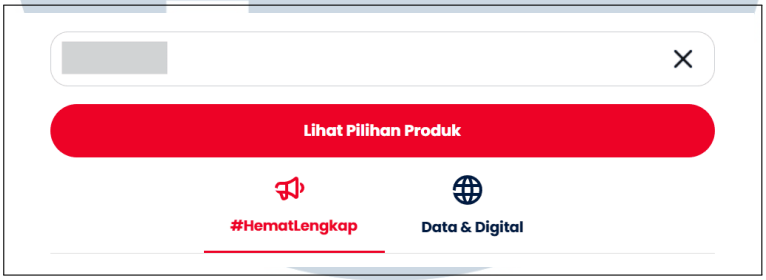
Integrasi ini dibatasi hanya untuk penyediaan *endpoint* dari *backend*, dan untuk mencapai tujuan ini, beberapa *endpoint* dari Telkomsel yang memang terbuka untuk digunakan secara umum akan digunakan untuk mengambil data yang diperlukan, guna memberikan pengalaman transaksi yang serupa bagi pengguna. Integrasi Telkomsel Omni sendiri dikerjakan pada *project* Whitelabel yang menggunakan Laravel 5.8 dan PHP 7.4. Diluar itu, testing *endpoint* selama proses pengembangan dilakukan menggunakan aplikasi Postman.

A. *Requirement*

Dalam proses integrasi Telkomsel Omni ke Whitelabel, terdapat beberapa *requirement* yang perlu diperhatikan untuk memungkinkan pengguna bertransaksi tanpa berpindah platform.

A.1 *Endpoint* Cek Menu Tersedia

Pengecekan menu yang tersedia merupakan tahap pertama dari alur transaksi Telkomsel Omni yang ingin diterapkan pada Whitelabel melalui penggunaan *endpoint* yang tersedia. Pada tahap ini, Telkomsel melakukan validasi nomor yang dimasukkan dan menampilkan menu yang tersedia untuk nomor tersebut.



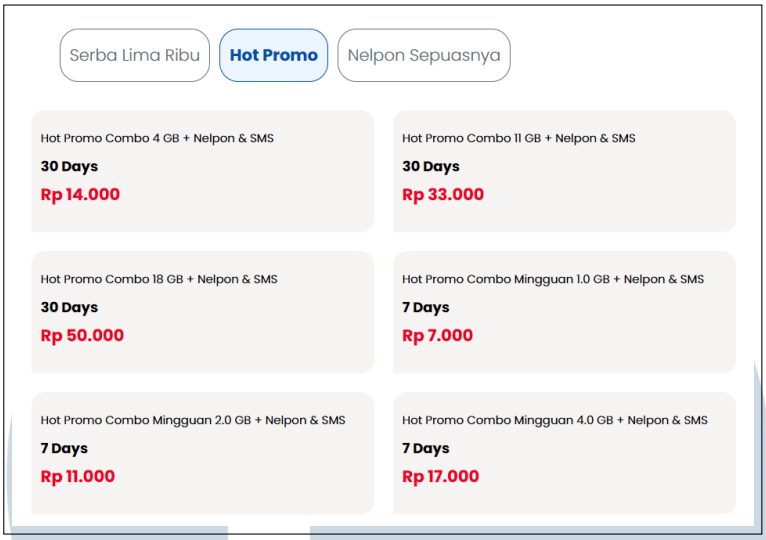
Gambar 3.14. Tampilan menu yang tersedia

Sumber: Situs web Telkomsel

Gambar 3.14 merupakan tampilan yang menampilkan hasil respons dari *endpoint* tersebut, dimana terdapat dua opsi menu yang tersedia untuk nomor yang diisi.

A.2 *Endpoint* Mengambil Paket yang Tersedia

Tahap selanjutnya yang ingin diterapkan pada Whitelabel adalah pengambilan paket yang tersedia untuk setiap menu yang dipilih. Melalui *endpoint* milik Telkomsel, sistem akan memperoleh daftar paket yang dapat diakses oleh nomor telepon pelanggan.



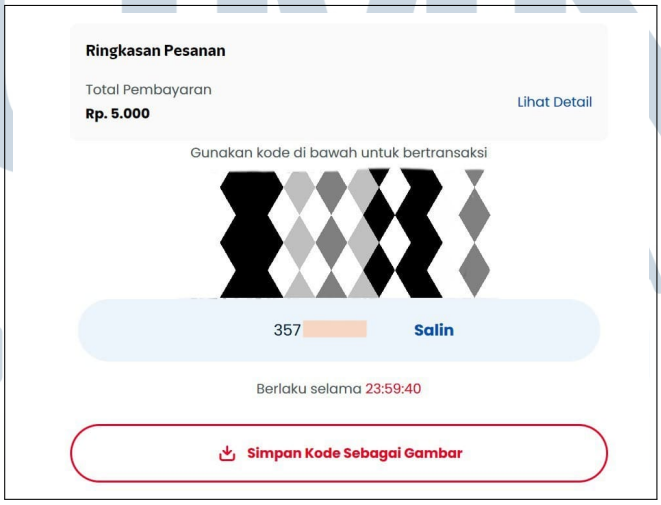
Gambar 3.15. Tampilan paket yang tersedia

Sumber: Situs web Telkomsel

Pada situs web Telkomsel, respons paket-paket yang didapat dari *endpoint* tersebut kemudian ditampilkan seperti yang ada pada Gambar 3.15 diatas.

A.3 *Endpoint* Order dan Transaksi

Terakhir, ketika pelanggan memilih salah satu paket yang tersedia, maka *barcode* dan kode unik pembayaran akan di-*generate* melalui *endpoint* tertentu dengan *payload* kode paket.



Gambar 3.16. Tampilan kode unik pembayaran

Sumber: Situs web Telkomsel

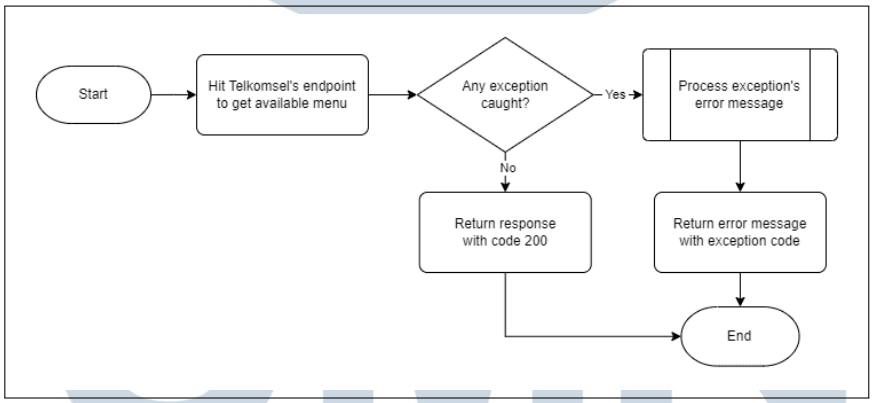
Kode unik yang perlu disalin ke Whitelabel untuk melakukan transaksi pada integrasi yang telah diterapkan sebelumnya ditunjukkan pada Gambar 3.16. Namun, pada integrasi kali ini, kode tersebut akan langsung ditampilkan dan dimasukkan secara otomatis sebagai kode untuk menghasilkan *inquiry* di Whitelabel. Dari hasil *inquiry*, baru pelanggan memasukkan pin / kata sandi untuk menyelesaikan transaksi.

B. Perancangan Sistem

Perancangan sistem untuk integrasi Telkomsel Omni pada Whitelabel memanfaatkan beberapa *endpoint* yang tersedia, yang kemudian di-*handle* lagi dalam sistem Whitelabel.

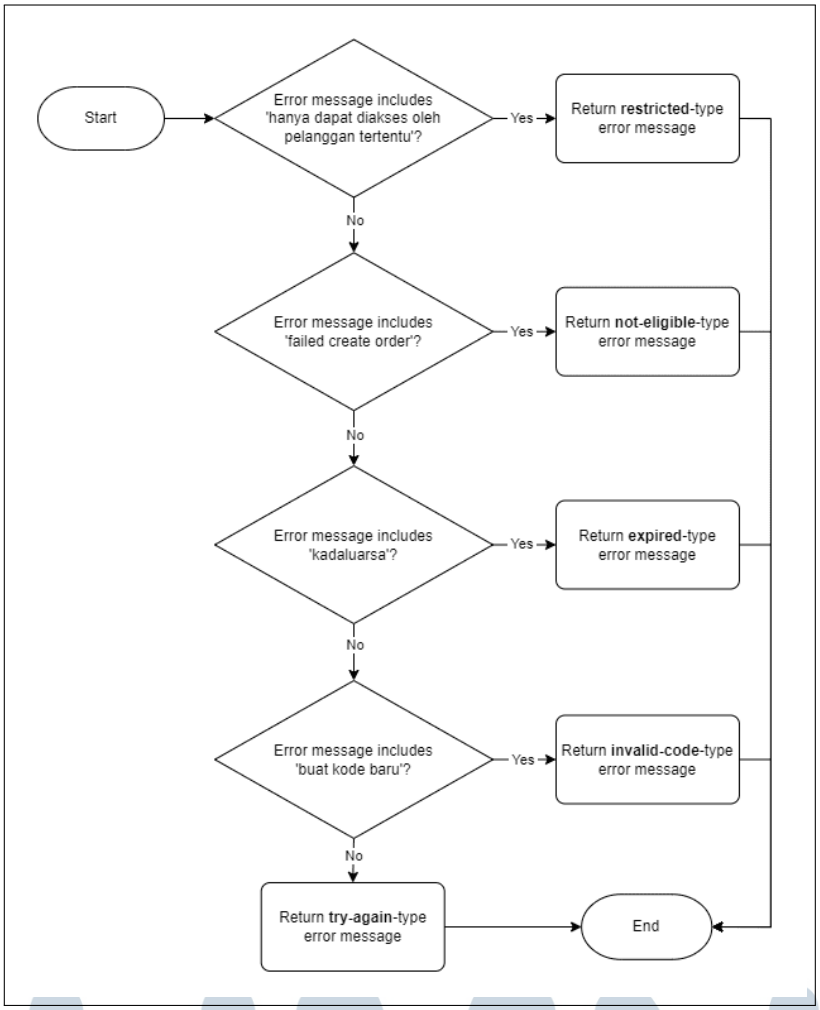
B.1 *Endpoint* Cek Menu Tersedia

Proses pengecekan menu yang tersedia pada dasarnya hanya mengantarkan respons sukses atau *error* yang didapat dari Telkomsel.



Gambar 3.17. *Flowchart* pengecekan menu yang tersedia

Pada Gambar 3.17, proses diawali dengan menembak *endpoint* Telkomsel untuk mendapatkan menu yang tersedia untuk nomor pelanggan yang ada dalam *payload*. Jika tidak ada *exception* yang muncul, maka sistem akan mengembalikan data-data menu yang tersedia. Namun jika ada *exception*, maka pesan *error* yang diterima akan diolah terlebih dahulu sebelum kemudian dikembalikan.

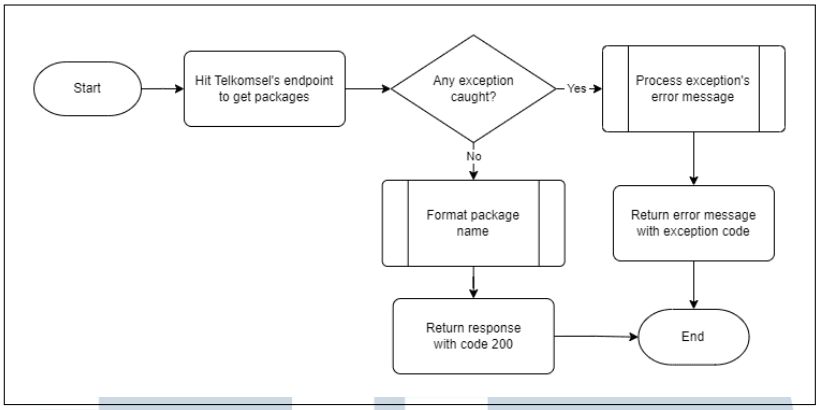


Gambar 3.18. *Flowchart* modul *Process exception error message*

Proses pengolahan pesan *error* yang akan digunakan untuk proses yang bersinggungan dengan sisi Telkomsel digambarkan pada Gambar 3.18. Terdapat beberapa jenis pesan *error* yang dapat dikembalikan oleh Telkomsel. Oleh karena itu, sesuai dengan *flowchart* yang ada, pesan yang dikembalikan akan disesuaikan dengan jenis *error* yang diterima dari Telkomsel.

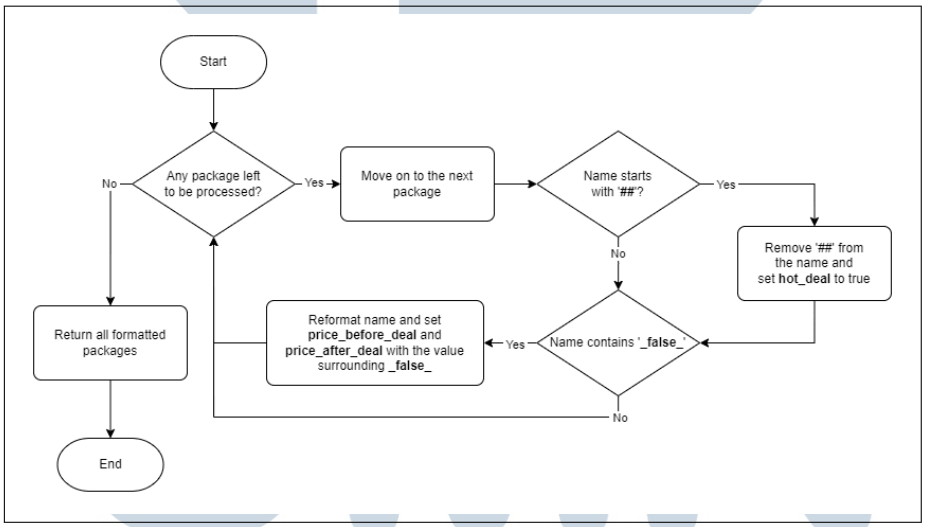
B.2 *Endpoint* Mengambil Paket yang Tersedia

Endpoint yang kemudian perlu dibuat kemudian digunakan untuk mengambil semua paket yang tersedia berdasarkan menu yang dipilih.



Gambar 3.19. *Flowchart* pengambilan paket yang tersedia

Flowchart pada Gambar 3.19 dimulai dengan pemanggilan *endpoint* Telkomsel untuk mengambil paket. Jika tidak ada *exception*, respons paket akan diformat terlebih dahulu baru dikembalikan. Tetapi, jika terjadi *exception*, pesan *error* akan disesuaikan dan dikembalikan bersama kodenya.

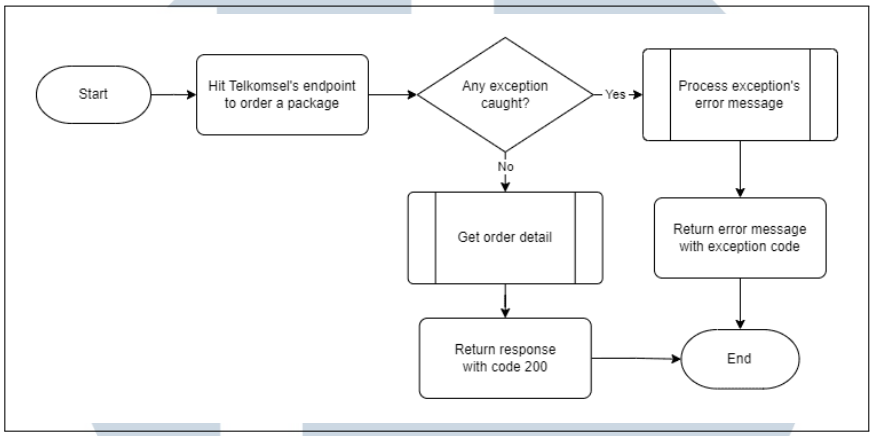


Gambar 3.20. *Flowchart* modul *Format package name*

Gambar 3.20 menunjukkan alur format nama paket. Jika nama paket mengandung '##', karakter tersebut dihapus dan atribut *hot_deal* ditambahkan dengan nilai *true*. Kemudian jika nama mengandung '_false_', itu menandakan adanya promo, seperti pada contoh 'Paket Nonton 2GB_false_3GB', yang berarti paket 2GB sedang promo menjadi 3GB. Nama akan disesuaikan dengan menghapus bagian tersebut, sementara atribut *price_before_deal* dan *price_after_deal* diisi dengan '2GB' dan '3GB' masing-masing. Ketika semua paket sudah disesuaikan, maka hasilnya akan dikembalikan dan proses selesai.

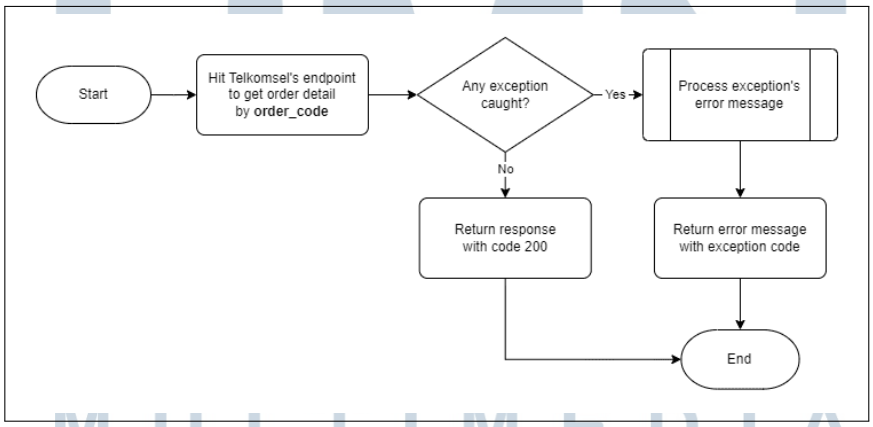
B.3 *Endpoint* Order dan Transaksi

Hal terakhir yang dibuat adalah *endpoint* untuk mendapatkan kode pembayaran dan mengambil detail pesanan.



Gambar 3.21. *Flowchart* untuk *generate* kode bayar

Flowchart pada Gambar 3.16 dimulai dengan pemanggilan *endpoint* Telkomsel untuk memproses pemesanan paket yang dipilih agar kode bayar dapat diperoleh. Jika tidak ada *exception*, langkah berikutnya adalah mendapatkan detail order menggunakan kode bayar tersebut. Namun jika terjadi *exception*, maka pesan *error* akan diproses dan dikembalikan sebagai respons *error*.



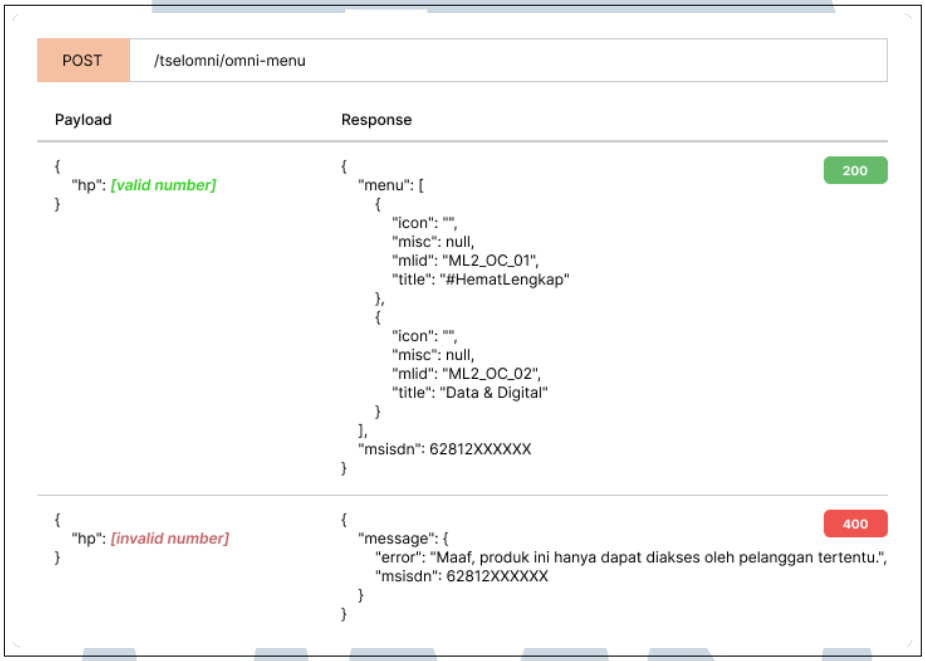
Gambar 3.22. *Flowchart* modul *Get order detail*

Modul untuk mengambil detail order digambarkan pada Gambar 3.22. Untuk mengambil detail order, sistem akan memanggil *endpoint* Telkomsel lain dengan *payload* kode bayar yang didapat sebelumnya. Jika tidak ada *exception*, maka detail yang diterima akan dikembalikan sebagai respons. Sebaliknya, jika ada

exception yang muncul, maka proses akan berhenti dengan mengembalikan pesan *error* yang telah disesuaikan.

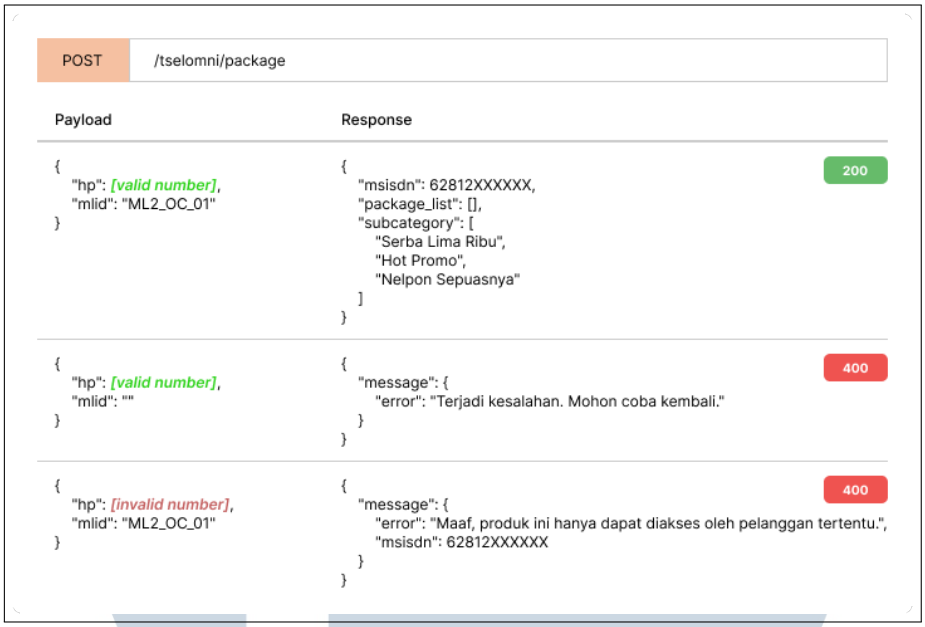
C. Implementasi Sistem

Implementasi sistem untuk integrasi dengan Telkomsel Omni berupa beberapa *endpoint* yang nantinya dapat dipanggil dari *frontend*.



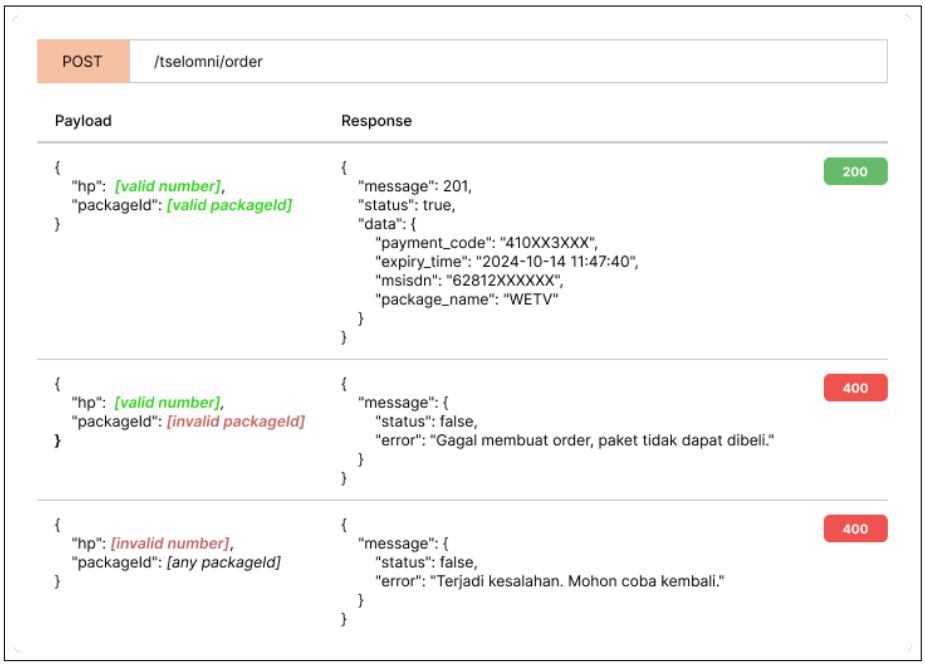
Gambar 3.23. Detail *endpoint* untuk cek menu

Gambar 3.23 merupakan detail *endpoint* yang dibuat untuk mengambil menu yang tersedia untuk nomor telepon pelanggan yang disertakan sebagai *payload*. *Endpoint* ini menggunakan metode HTTP POST dan akan mengembalikan respons berupa daftar menu yang tersedia beserta dengan nomor telepon pelanggan jika berhasil, dan pesan *error* jika gagal.



Gambar 3.24. Detail *endpoint* untuk mengambil paket berdasarkan menu

Gambar 3.24 menunjukkan detail *endpoint* untuk pengambilan daftar paket. Menggunakan metode HTTP POST, *endpoint* ini memerlukan nomor telepon dan nama menu yang ingin didapatkan paketnya. Respons berhasilnya berupa nomor telepon, daftar paket, beserta sub-kategori yang ada.



Gambar 3.25. Detail *endpoint* untuk melakukan order

Gambar 3.25 menunjukkan detail *endpoint* untuk melakukan order pada paket yang dipilih yang menggunakan metode POST. Dari respons daftar paket yang diterima sebelumnya, setiap paket memiliki atribut *packageId*, atribut inilah yang kemudian akan disertakan sebagai *payload* untuk mendapat kode bayar beserta detailnya. Respons berhasil akan mengembalikan *message*, *status*, dan data berisi kode bayar, waktu kedaluwarsa, nomor telepon, serta nama paket yang dibeli.

3.3.3 Fitur Automasi Migrasi Antar *Supplier*

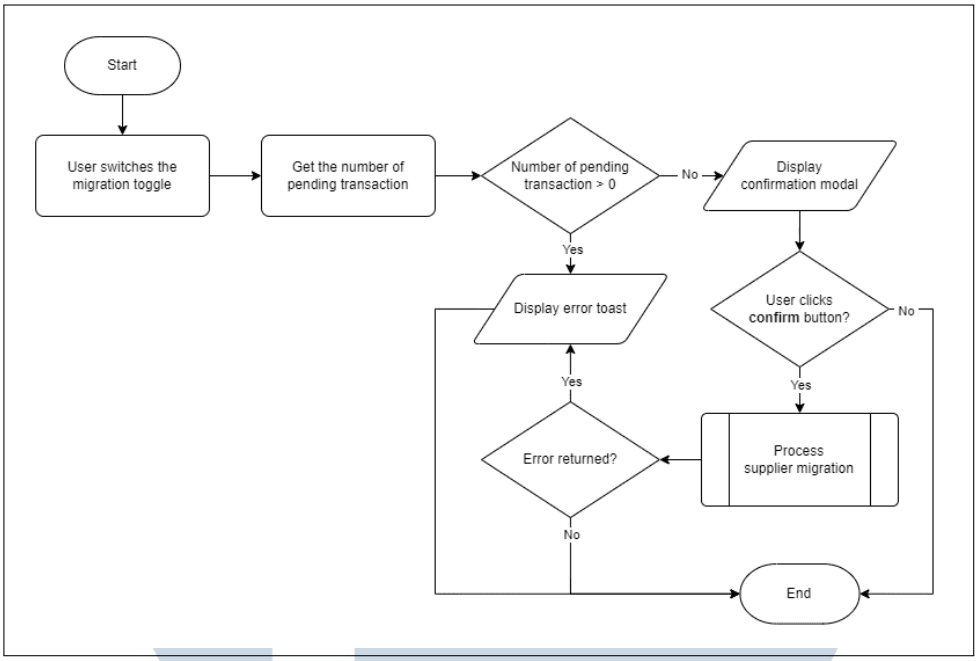
Awalnya, migrasi antar *supplier* A dan *supplier* B dilakukan secara manual dengan mengikuti tahapan-tahapan yang telah ditetapkan. Namun, karena proses manual ini rentan terhadap *human error*, dikembangkan sebuah fitur untuk mengotomatisasi proses tersebut pada CMS Whitelabel. CMS yang dimaksud adalah panel admin yang digunakan secara internal untuk mengelola *tenant* Whitelabel dengan lebih efisien. Fitur automasi migrasi antar *supplier* dikerjakan pada *project* CMS Whitelabel, dan menggunakan Laravel 6.0, PHP 7.4, dan Vue 2. Pada salah satu proses yang ada dalam proses migrasi, terdapat proses yang berhubungan dengan *cache*, sehingga digunakan Another Redis Desktop sebagai aplikasi pendukung selama pengembangan.

A. *Requirement*

Requirement pada pengembangan automasi migrasi antar *supplier* merangkup tampilan sederhana berupa *toggle supplier* dan API untuk melakukan migrasi. Pada salah satu menu di situs web CMS Whitelabel, terdapat daftar situs web *tenant* yang terdaftar. Di halaman ini, akan ditempatkan *toggle* yang memungkinkan tim internal Whitelabel untuk melakukan migrasi antar *supplier* dilakukan dengan mudah.

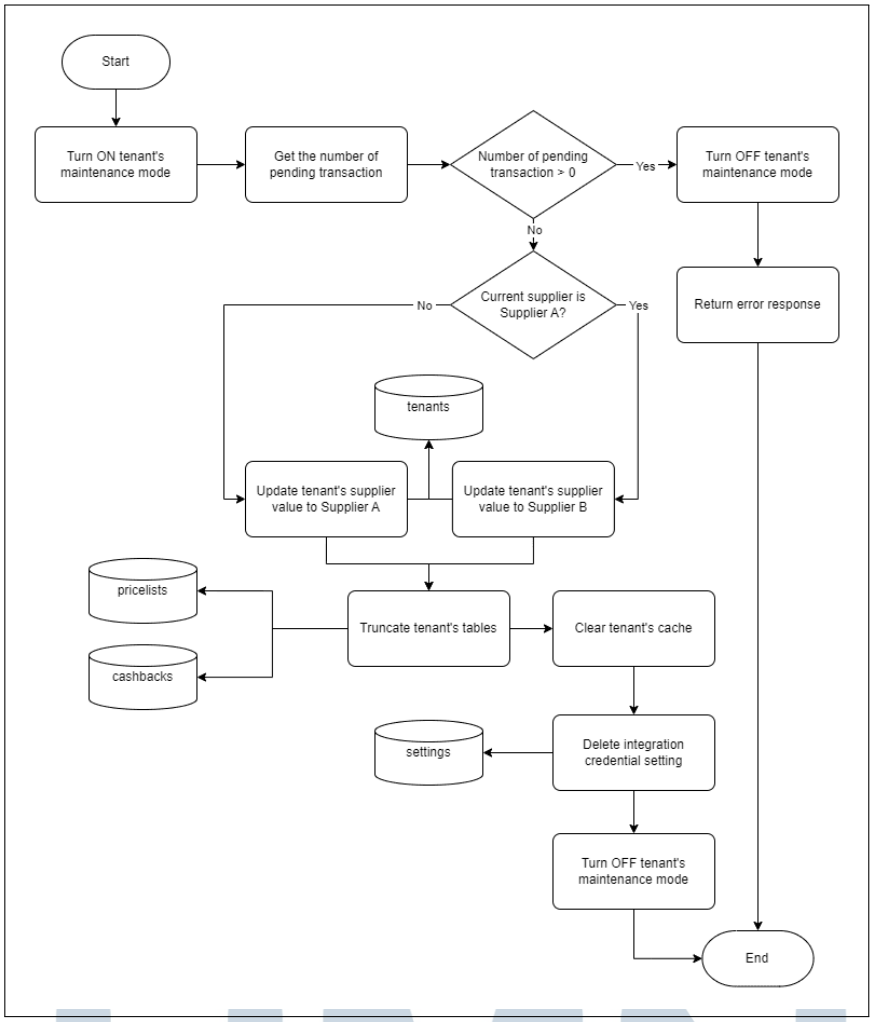
B. Perancangan Sistem

Automasi pada proses migrasi antar *supplier* ini dirancang menggunakan *flowchart* yang menggabungkan *frontend* sekaligus *backend*.



Gambar 3.26. *Flowchart* utama migrasi antar *supplier*

Secara keseluruhan, proses automasi migrasi antar *supplier* digambarkan pada *flowchart* yang ada di Gambar 3.26. Ketika seorang *tenant* memutuskan untuk mengganti *supplier* untuk produk PPOB-nya, maka tim internal Whitelabel akan pergi ke menu *website* di CMS, menuju ke situs web *tenant*. Pada halaman ini, *user*, atau yang dalam konteks ini adalah tim internal Whitelabel, akan memindahkan *toggle* untuk melakukan migrasi. Ketika *toggle* dipindahkan, maka sistem akan mengambil jumlah transaksi prabayar dan pascabayar yang memiliki status *pending*. Jika ternyata *tenant* masih memiliki transaksi dengan status *pending*, maka sistem akan menampilkan pesan *error* dan proses migrasi tidak dapat dilakukan. Tetapi jika tidak ada transaksi *pending*, sistem baru akan menampilkan modal konfirmasi untuk memastikan perpindahan *supplier*, baik dari *supplier* A ke B maupun sebaliknya. Proses migrasi hanya akan dilakukan jika migrasi ke *supplier* lain dikonfirmasi.



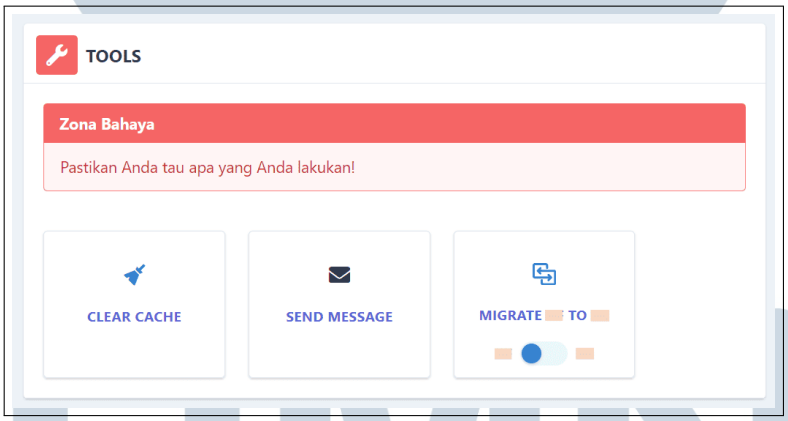
Gambar 3.27. *Flowchart* modul *Process supplier migration*

Pada proses migrasi dari *backend* yang ditunjukkan pada Gambar 3.27 dimulai dengan menyalakan mode pemeliharaan untuk situs web *tenant*. Hal ini dilakukan karena proses migrasi dapat mengganggu *flow* transaksi *tenant*. Kemudian, sistem akan mengecek dan memastikan lagi bahwa tenant tidak memiliki transaksi yang sedang *pending*. Pengecekan kedua ini bertujuan untuk mencegah terjadinya transaksi *pending* yang masuk dalam rentang waktu antara pengecekan pertama di frontend dan saat pengguna mengonfirmasi migrasi. Jika ditemukan transaksi *pending*, maka proses migrasi akan dihentikan dengan mengaktifkan kembali mode pemeliharaan situs web *tenant*, serta mengembalikan respons *error*. Namun, jika tidak ada transaksi *pending*, sistem akan memeriksa apakah *tenant* melakukan migrasi ke *supplier* A atau B. Penentuan *supplier* tujuan diperlukan untuk memperbarui atribut *supplier* pada tabel ”tenants” di *database*.

Setelah mengubah nilai yang menjadi indikator *supplier* yang digunakan *tenant*, maka sistem akan menghapus semua data pada tabel ”pricelists” dan ”cashbacks”, karena produk yang disediakan oleh setiap *supplier* berbeda. Proses selanjutnya adalah menghapus *cache tenant* untuk menghindari adanya data lama yang tersangkut di *cache*. Kemudian karena setiap integrasi dengan *supplier* memerlukan kredensial, maka kredensial yang terkait dengan *supplier* sebelumnya perlu dihapus, karena tidak lagi relevan dengan *supplier* yang baru. Setelah semua proses tersebut selesai, maka mode pemeliharaan situs web akan dinyalakan kembali, dan proses migrasi telah selesai dilakukan.

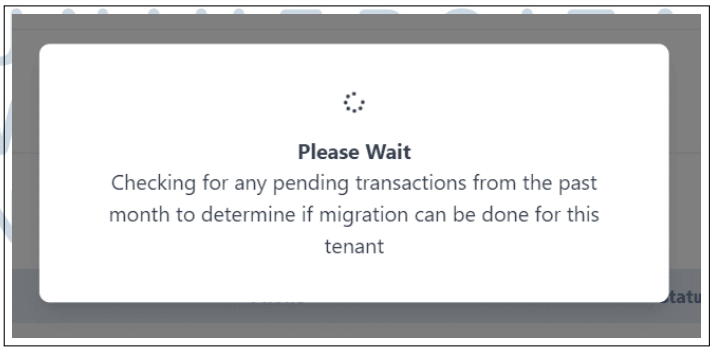
C. Implementasi Sistem

Proses automasi untuk migrasi antar *supplier* ini diimplementasikan pada situs web CMS pada bagian *tools* dari situs web yang ingin diubah *supplier*-nya.



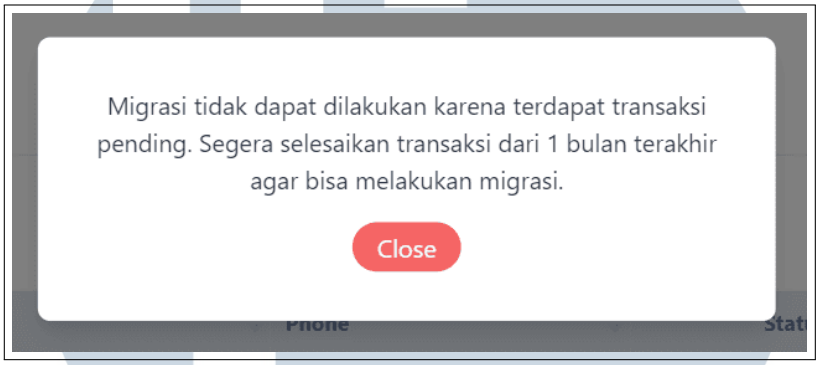
Gambar 3.28. Implementasi tampilan *toggle* migrasi *supplier*

Gambar 3.28 merupakan *toggle* yang dapat digunakan untuk mengubah *supplier* yang digunakan *tenant* tersebut.



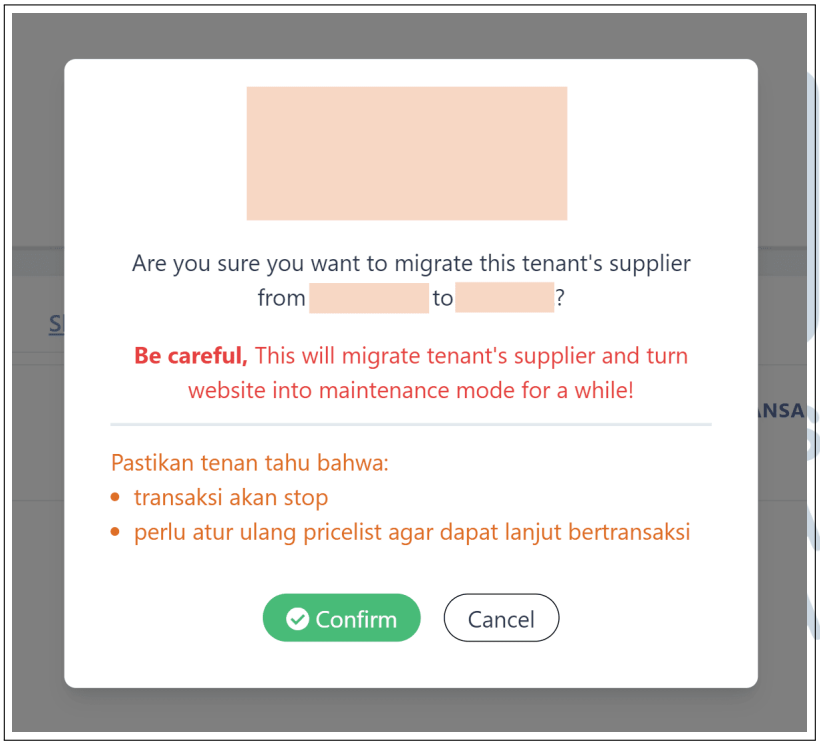
Gambar 3.29. Implementasi tampilan modal *loading* dalam pengecekan transaksi *pending*

Sedikit berbeda dari perancangan, karena proses pengecekan tidak secepat itu, maka ditambahkan modal *loading* untuk memberitahu bahwa sedang dilakukan proses pengecekan. Oleh karena itu, ketika *toggle* dipindahkan, maka modal *loading* yang ada pada Gambar 3.29 akan muncul selagi sistem melakukan pengecekan.



Gambar 3.30. Implementasi tampilan modal ketika ditemukan transaksi *pending*

Kemudian, jika selama pengecekan ditemukan adanya transaksi berstatus pending, maka akan ditampilkan modal seperti pada Gambar 3.30 yang menginformasikan bahwa masih terdapat transaksi pending.



Gambar 3.31. Implementasi tampilan modal konfirmasi

Tetapi jika tidak ada transaksi pending yang ditemukan, maka modal pada Gambar 3.31 akan dimunculkan untuk melakukan konfirmasi kembali. Setelah dikonfirmasi, dan jika pada prosesnya tidak terjadi *error*, maka proses migrasi berhasil dilakukan dan sistem akan menampilkan *toast* berhasil serta memuat ulang halaman. Namun jika terdapat *error* selama proses migrasi berlangsung, pesan *error* akan ditampilkan melalui *toast* dan status *supplier tenant* akan kembali ke awal.

3.4 Kendala dan Solusi yang Ditemukan

Selama kerja magang berlangsung, terdapat beberapa kendala yang ditemukan, yaitu sebagai berikut.

1. Kurang terbiasa dengan *framework* yang digunakan, terutama Vue untuk *frontend*, sehingga beberapa teknik yang dapat membantu tidak dimanfaatkan, dan menyebabkan waktu pengembangan yang lebih lama.
2. Penulisan kode masih kurang rapi, efisien, dan belum konsisten dengan standar penulisan yang telah ada sebelumnya.
3. Kurangnya pengetahuan mengenai proyek, sehingga terdapat beberapa hal yang terlewat selama proses pengembangan.

Dari kendala-kendala diatas, ditemukan solusi untuk mengatasi setiap kendala yang ada.

1. Membiasakan diri untuk menggunakan *framework* tersebut serta membaca dokumentasi dan contoh teknik-teknik yang sudah diterapkan.
2. Melihat referensi dari kode yang ada pada proyek dan memanfaatkan *feedback* dari *code review* untuk meningkatkan penulisan kode.
3. Lebih sering membaca dokumentasi proyek, mencoba aplikasi dari perspektif pengguna, dan berdiskusi dengan anggota tim untuk menambah pengetahuan proyek.