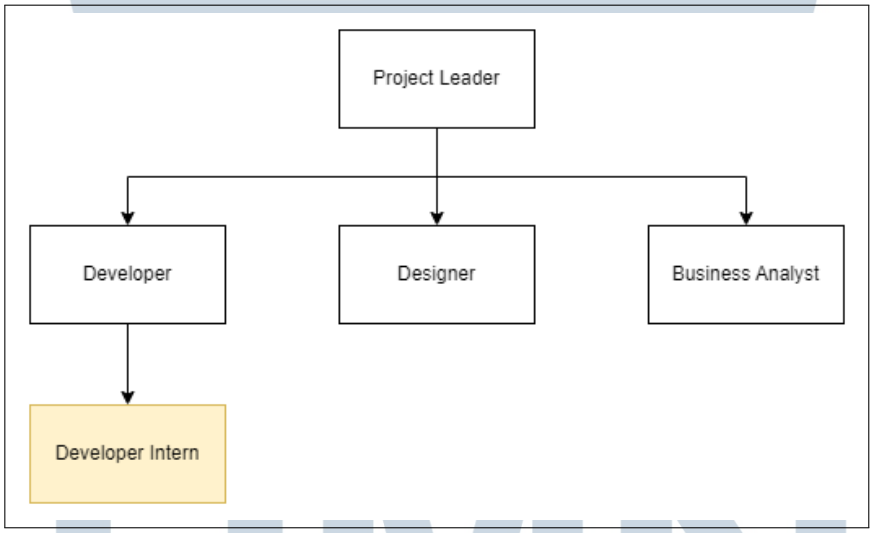
BAB 3
PELAKSANAAN KERJA MAGANG

3.1 Kedudukan dan Organisasi

Selama pelaksanaan kegiatan magang di PT Cranium Royal Aditama, posisi yang diisi adalah *fullstack developer intern*. Pada periode magang kali ini pengerjaan proyek ERP difokuskan ke bagian *front end*. Bapak Sugito, yang merupakan *Vice President Engineering* di PT Cranium Royal Aditama, memimpin proyek pembuatan sistem ERP dan juga pembimbing bagi *fullstack developer intern*.



Gambar 3.1. Struktur Proyek ERP

Gambar 3.1 merupakan struktur proyek ERP. Proyek ERP dikepalai oleh Bapak Sugito sebagai *project leader* dan di bawahnya terdapat tiga divisi, yaitu *developer, designer* dan juga *business analyst*. Di bawah *developer* terdapat *developer intern* yang merupakan kedudukan yang diduduki pada pelaksanaan kerja magang ini.

3.2 Tugas yang Dilakukan

Pada pelaksanaan magang di PT Cranium Royal Aditama di periode ini, tugas utama yang diberikan adalah membuat *interface* sistem ERP berdasarkan desain yang sudah dibuat oleh desainer dari PT Cranium Royal Aditama.

Pembuatan *interface* sistem ERP ini menggunakan React dan juga library Material UI (MUI). Tugas pertama adalah mempelejari *tech stack* yang digunakan, yaitu React, Next.Js, dan juga Material UI. Setelah pembelajaran secara mandiri tersebut, *supervisor* memberikan sebuah *template* proyek untuk dilakukan modifikasi dan percobaan secara mandiri. Percobaan tersebut meliputi *clonning*, yaitu membuat menu baru berdasarkan contoh yang sudah ada di *template*.

Setelah proses pembelajaran tersebut, *supervisor* membagi tugas berdasarkan tim. Dalam kasus ini, penulis mendapatkan bagian untuk membuat *template* bagian *view*. Setelah itu, *template* untuk *create, view, update* dan *list* dijadikan satu. Maka, pekerjaan untuk ke menu selanjutnya pun dapat dikerjakakan.

Penulis pun melanjutkan ke menu-menu selanjutnya. Menu-menu tersebut meliputi menu *supplier* dan *account type* dan *supplier type* di modul *master*, menu *purchase request* di modul *purchasing* dan menu *production order item receipt* di modul *production*.

A. Pembelajaran Tech Stack

Pembelajaran *tech stack* yang dimaksud adalah pembelajaran teknologi yang digunakan dalam pengerjaan proyek ERP secara mandiri dengan mencari referensi di internet. Teknologi yang dipelajari selama pembelajaran ini adalah React, Next.Js, dan juga library Material UI (MUI). Pembelajaran ini berlangsung selama satu minggu.

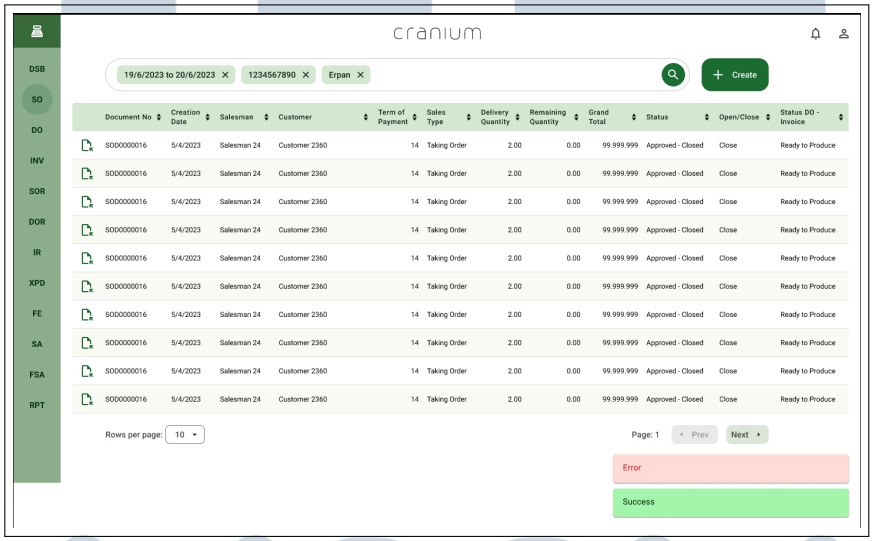
B. Pengenalan Struktur Proyek

Setelah pembalajaran mandiri, *supervisor* memberikan sebuah *template* untuk dipelajari. Dari *template* tersebut, pengembang dapat mempelajari struktur proyek ERP dan juga jalannya aplikasi. Pada pengenalan struktur proyek ini, pengembang juga diminta untuk membuat halaman untuk *method* CRUD untuk satu menu sebagai salah satu cara mempelajari proyek. Dalam struktur proyek ERP ada yang dinamakan *global component* yang merupakan component yang dipakai oleh banyak modul. Dalam kasus ini, penulis membuat global component pagination, yaitu component di bawah list untuk mengarahkan dan mengatur list tersebut. Pada bagian ini juga, *supervisor* membagi *requirements* kepada pengembang-pengambang dalam proyek ini. *Requirements* ataupun *task* yang didapatkan adalah membuat halaman untuk melakukan proses *create, read, update* dan *delete* untuk

menu *purchasing request, inventory item withdraw reservation, production order item receipt, master supplier, acccount type* dan juga *supplier type*.

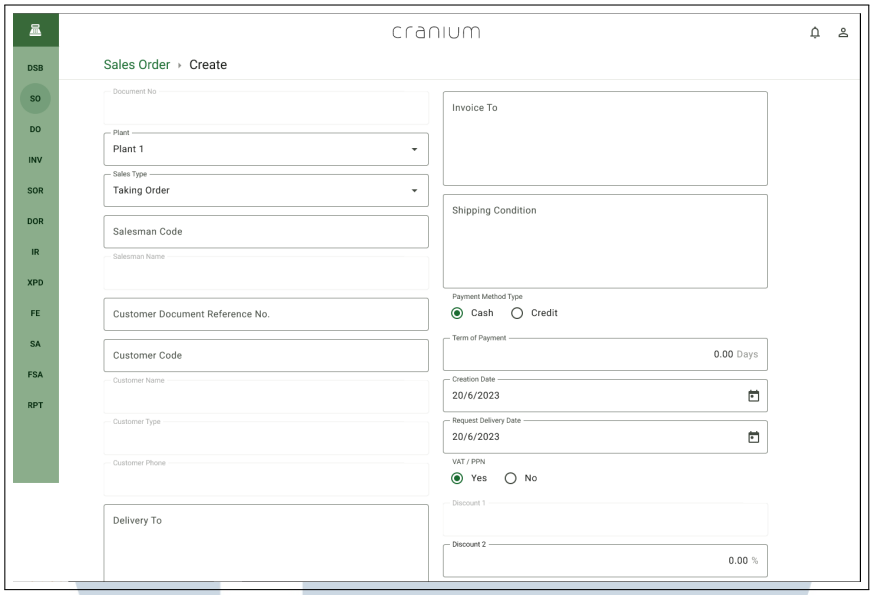
C. Pengenalan Desain

Dalam pengembangan sistem ERP di perusahaan Cranium, desain tampilan dari sistem tersebut dibuat oleh desainer dari perusahaan Cranium itu sendiri. Nantinya, pengembang diminta untuk mengimplementasikan desain tersebut pada setiap halaman yang dibuat.



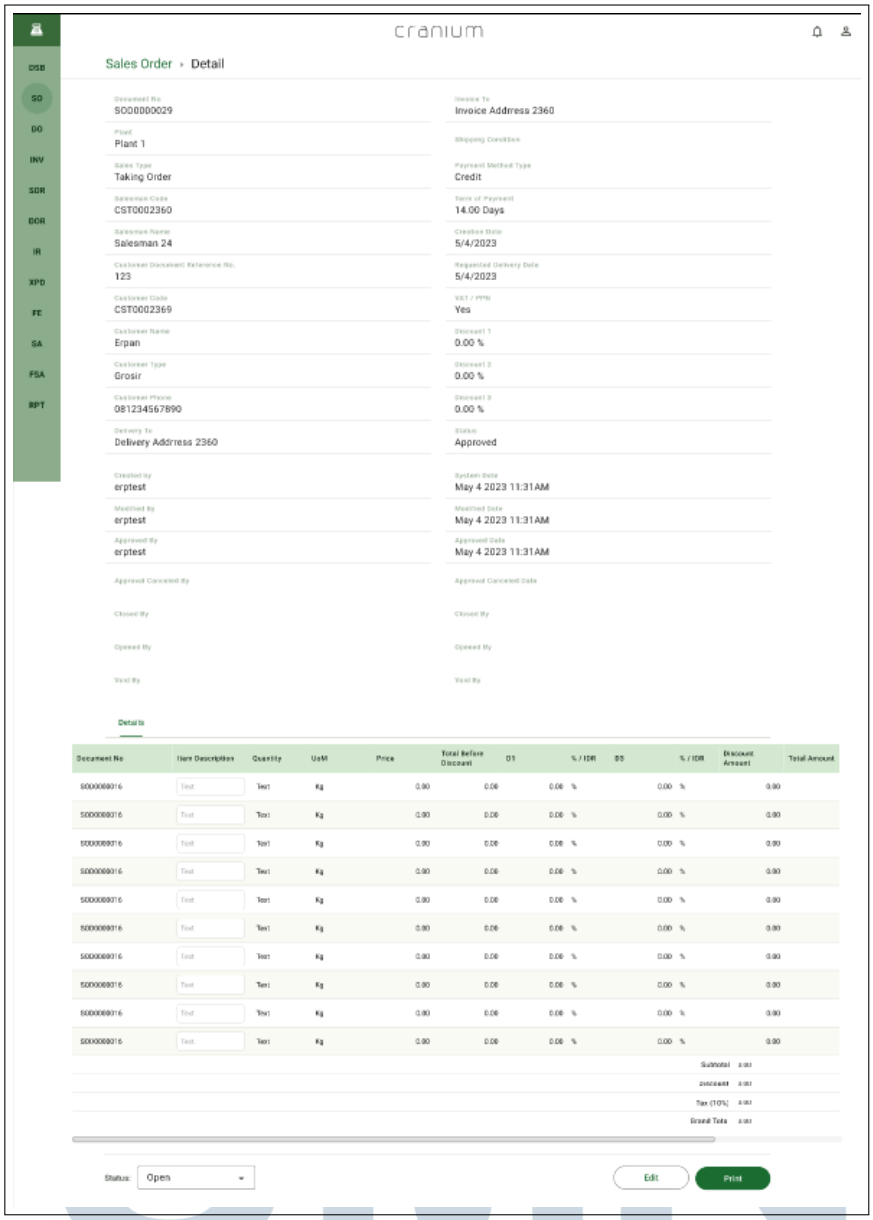
Gambar 3.2. Mockup halaman list

Gambar 3.2 merupakan *mockup* halaman *list* atau *find all*. Di halaman ini terdapat tabel dengan kolom pertama merupakan kolom untuk tombol *view* berdasarkan id dan kolom selanjutnya merupakan *field-field* dari menu yang diakses. Di bawah tabel juga terdapat *pagination* yang dapat mengatur berapa banyak data yang ingin ditampilkan di satu *page* dan terdapat juga tombol untuk mengakses *page* selanjutnya ataupun *page* sebelumnya. Di atas tabel terdapat kolom untuk melakukan pencarian atau *searching* dan di sebelah kanan kolom tersebut terdapat *button* untuk mengakses halaman *create*.



Gambar 3.3. *Mockup* halaman *create* dan *update*

Gambar 3.3 merupakan gambaran atau *mockup* untuk halaman *create* dan juga *update*. Karena halaman *create* dan *update* dapat dikatakan mirip atau serupa, maka hanya ada satu *mockup* yang disediakan. Dalam *mockup* terdapat *form* dengan *field-field* yang ada dapat diisi pengguna saat hendak melakukan *create* ataupun *update*. Perbedaan dari halaman *create* dan *update* adalah saat mengakses halaman *update*, *field-field* dalam *form* sudah terisi dari data yang hendak diubah. Sedangkan, dalam proses *create field-field* tersebut masih kosong dan harus diisi pengguna saat ingin melakukan *create.*



Gambar 3.4. *Mockup* halaman *view*

Gambar 3.4 merupakan *mockup* dari halaman *view* yang dapat melihat isi dari *field-field* dari suatu menu. Di bagian atas merupakan bagian yang berisi data dari *header* sedangkan di bawahnya merupakan data dari *detail* yang disajikan dalam bentuk tabel. Di bawah tabel tersebut terdapat sebuah *dropdown* untuk mengubah status dan juga tombol untuk mengakses halaman *update* dan tombol untuk *print*.

3.3 Uraian Pelaksanaan Magang

Pelaksanaan kerja magang diuraikan seperti pada Tabel 3.1.

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

Minggu ke-	Aktivitas yang dikerjakan
1	Mempelajari tentang sistem javascript dan react serta *training* dengan Pak Gito. Memasang hal yang dibutuhkan dalam proyek ERP dan mempelajari konsep *routing* di Next.Js
2	Mencoba melakukan clonning dari *template* yang diberikan dan mencohba membuat CRUD dan membagi tugas
3	Membuat *template* tampilan view dan *button-button* di dalamnya
4	Memperbaiki *styling* di bagian tampilan *view* dan menggabungkannya dengan kelompok lain
5	Membuat *global component* pagination serta mempelajari *unit test* dan membuat *unit test*
6	Melakukan *clonning* untuk membuat master supplier, namun terkendala di bagian *autocomplete*
7	Memperbaiki *autocomplete* supplier type di *update* dan *create* dan membuat *autocomplete* chart of account di bagian *create* supplier
8	Sudah melengkapi semua *autocomplete* untuk *update* supplier. Melanjutkan ke *unit test* supplier, namun ada beberapa kendala
9	Melakukan revisi *component* pagination
10	Membuat menu account type
11	Membuat *unit test* account type. Melakukan revisi *create* account type sesuai arahan Pak Gito dan membuat *validator*
12	Melanjutkan ke modul purchasing, menu purchase request. Membuat tampilan untuk *list* dan juga *view*
13	Membuat tampilan untuk *create* dan *update* purchase request
14	Membuat *unit test create, delete, list* dan *view* purchase request
15	Memindahkan modul inventory ke modul warehouse, menambahkan efek hover di bagian tabel dalam *list*, lalu melanjutkan ke modul production, menu production order item receipt

Dilanjutkan di halaman selanjutnya

Tabel 3.1 – dilanjutkan dari tabel sebelumnya

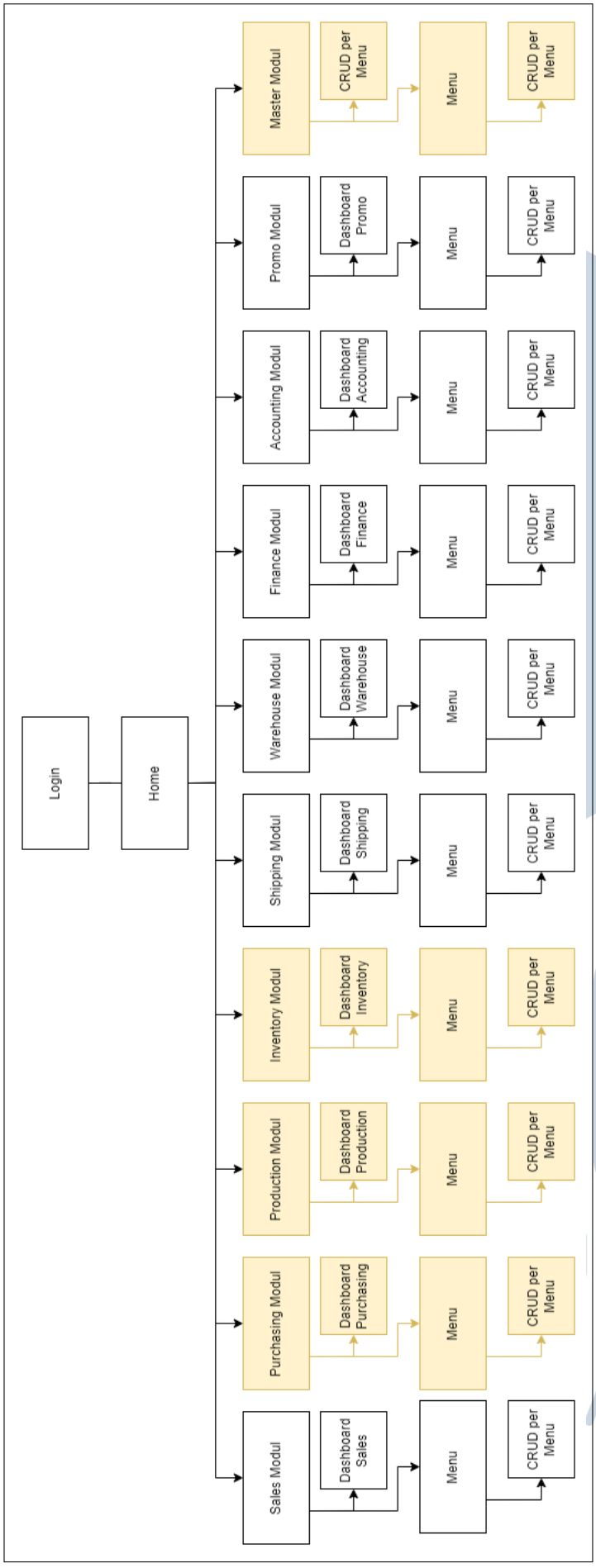
Minggu ke-	Aktivitas yang dikerjakan
16	Membuat update dan create untuk order item receipt
17	Membuat unit test untuk order item receipt
18	Menambahkan breadcrumb untuk modul-modul yang dikerjakan
19	Membuat halaman untuk inventory item withdraw reservation

3.4 Rancangan Sistem ERP

Pengerjaan sistem ERP dapat dijelaskan dalam *sitemap* dan *flowchart* dari setiap halaman yang dikerjakan.

3.4.1 Sitemap

Sitemap merupakan 'peta' yang menggambarkan halaman dari sebuah situs. Fungsi *sitemap* sendiri adalah memaksimalkan kerja SEO untuk situs tersebut. Namun, *sitemap* juga dapat menjadi gambaran umum dari sebuah situs.



Gambar 3.5. *Sitemap*

Gambar 3.5 merupakan gambaran umum dari sistem ERP yang dibuat jika digambarkan dengan *sitemap*. Diawali dengan *login* yang akan membawa pengguna ke halaman *home*. Di halaman *home* tersebut, pengguna dapat memilih modul mana yang hendak diakses. Di setiap modul terdapat *dashboard* dari modul tersebut yang akan terbuka jika pengguna mengklik modul tertentu. Namun, modul master tidak terdapat halaman *dashboard*. Lalu, pengguna dapat mengakses menu-menu yang ada di modul. Dan dari menu tersebut, proses CRUD dapat dilakukan.

Bagian yang diberi *highlight* merupakan bagian yang dikerjakan pada pelaksanaan kerja magang kali ini. Modul yang dikerjakan mencakup modul *purchasing, production, inventory* dan juga beberapa menu di modul *master*.

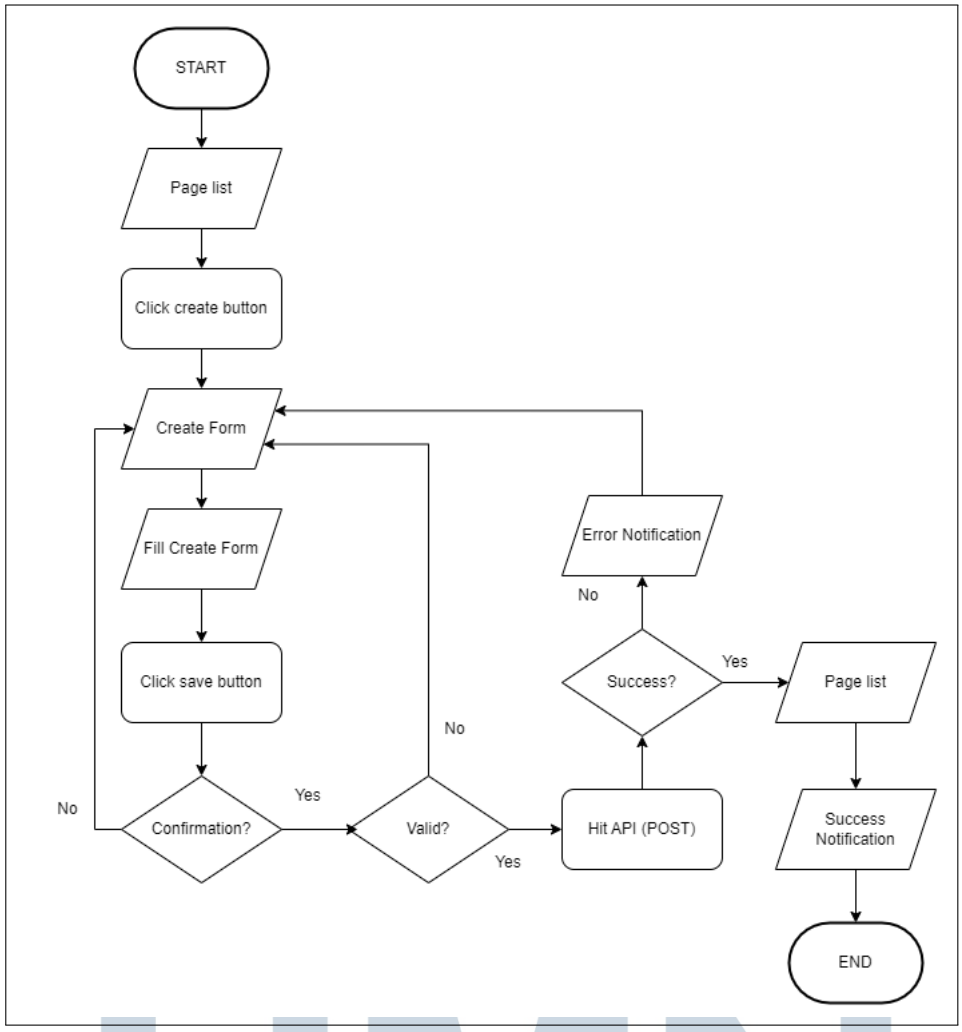
3.4.2 Flow Chart

Flow chart dapat menjadi acuan dan juga representasi pada saat pembuatan halaman. Halaman yang dibuat sendiri meliputi proses CRUD atau *create, read, update* dan *delete*. Proses pengguna mengakses halaman-halaman tersebut dapat direpresentasikan dengan *flowchart*. Semua proses dalam CRUD diawali dengan mengakses halaman *list* yang menunjukkan semua data yang belum terhapus dalam menu yang diakses. *List* yang ditampilkan berbentuk *Page* sehingga dapat dibatasi jumlah data yang ditampilkan di setiap halamannya dan dapat berpindah halaman untuk mengakses data di *Page* selanjutnya.

Contohnya adalah proses CRUD *purchasing request*. Saat pertama mengakses menu *purchasing request*, pengguna akan membuka halaman *list* yang menunjukan data-data *purchasing request* yang belum terhapus dalam bentuk tabel. Dalam kolom pertama tabel tersebut terdapat tombol *view* yang jika diklik akan membuka halaman *view* atau proses *read*. Di halaman *view* tersebut juga terdapat tombol *update* yang berguna untuk membuka halaman *update* untuk melakukan proses *update* pada data *purchasing request* yang sedang dibuka. Selain tombol *view* yang ada di dalam tabel, di halaman *list* juga terdapat tombol *create* yang berguna untuk membuat data *purchasing request* yang baru atau proses *create*.

A. Create

Halaman *create* merupakan halaman untuk membuat data dari suatu menu atau *method create*.



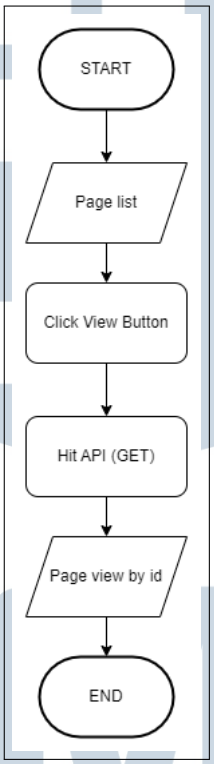
Gambar 3.6. Flowchart Create

Gambar 3.6 merupakan representasi alur halaman *create*. Diawali dengan membuka halaman *list*, lalu dilanjutkan dengan mengklik tombol *create*. Halaman *create* pun akan terbuka dan menampilkan *form* berisi *field-field* yang sesuai dengan data dari menu yang hendak dibuat. Pengguna perlu mengisi *form* tersebut dan mengklik tombol simpan atau *save*. Selanjutnya, akan ada modal konfirmasi, yang jika tidak dikonfirmasi akan kembali ke *form create* yang sudah diisi sebelumnya. Jika dikonfirmasi, akan berlanjut ke pengecekan validasi *front end* dan jika tidak valid, maka akan kembali ke *create form* sebelumnya dengan tanda di *field* yang tidak valid. Namun, jika valid sistem akan melakukan *hit* ke API dengan tipe POST dengan parameter berdasarkan *form* yang diisi oleh pengguna. Lalu, akan dicek respon dari API, jika tidak sukses, sistem akan menampilkan notifikasi error dan kembali ke *form*. Namun jika respon sukses, halaman akan diarahkan ke halaman

list dan menampilkan notifikasi sukses.

B. Read

Selain halaman untuk melihat semua data, ada juga halaman untuk melihat satu data dari sebuah menu secara detail. Halaman tersebut adalah tersebut adalah halaman *read*. Halaman *read* ini juga diakses melalui tombol *view* yang ada pada kolom pertama di tabel yang ada di halaman *list*.

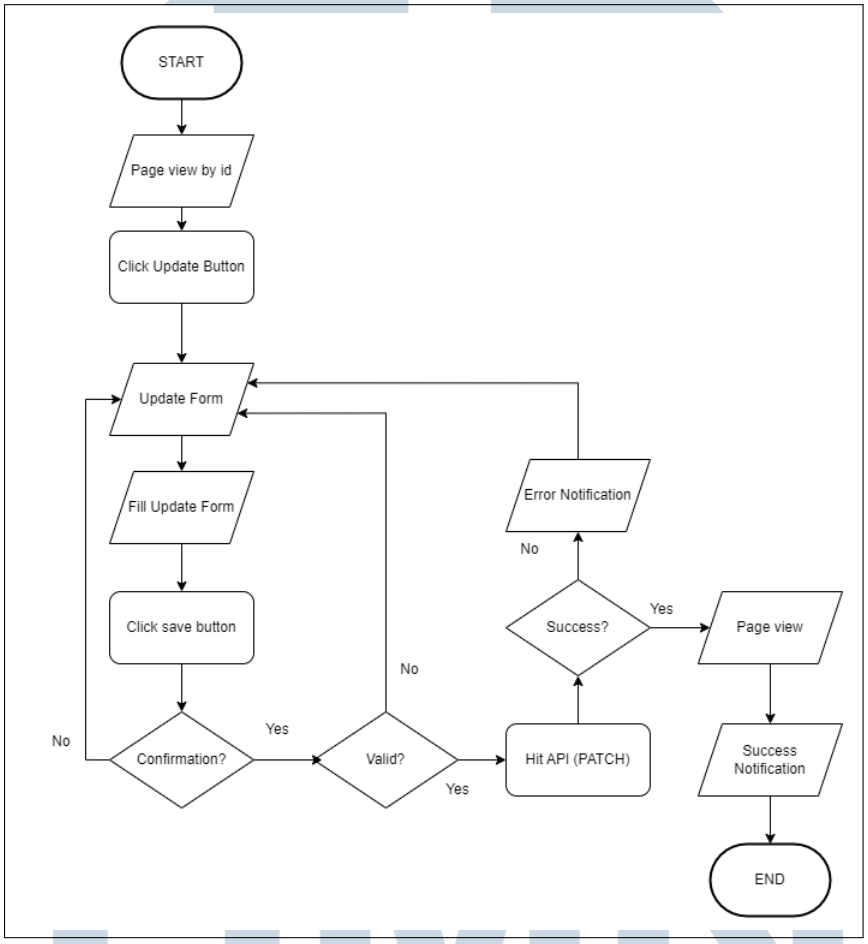


Gambar 3.7. Flowchart Read

Gambar 3.7 adalah alur halaman *read* yang direpresentasikan dengan *flowchart*. Sama seperti halaman lain, halaman *read* ini juga diawali dengan membuka halaman *list*. Di tabel data dalam halaman *list*, terdapat tombol *view* di setiap baris datanya. Tombol *view* tersebut diklik untuk membuka halaman *read* ini. Setelah diklik, sistem akan melakukan hit ke API dengan tipe GET untuk mendapatkan data dari sebuah menu berdasarkan id dari baris data yang diklik pengguna. Setelah *hit* ke API tersebut, halaman *read* akan terbuka yang berisi *field-field* yang sesuai dengan data dari suatu menu yang diakses.

C. Update

Sistem ERP yang dibuat juga dapat melakukan *method update* yang dilakukan dengan membuka halaman *update*.



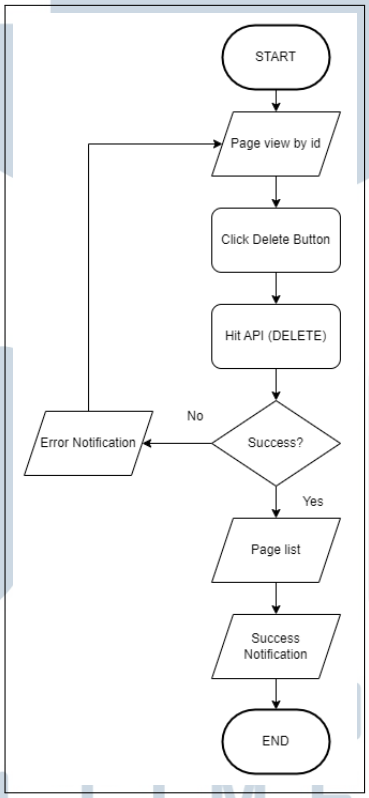
Gambar 3.8. Flowchart Update

Gambar 3.8 merupakan *flowchart* dari halaman *update*. Halaman *update* ini dapat diakses melalui halaman *read* berdasarkan id. Di halaman *read* tersebut tombol *update* atau ubah. Jika tombol tersebut diklik, maka halaman akan berpindah ke halaman *update* yang berisi *form update*. *form update* meliputi *field-field* yang dapat diubah data dari suatu data di sebuah menu. Dalam *form update* tersebut data dari *field-field* dalam *form* sudah berisi *default value* yang berasal dari data sebelum diubah. *User* perlu mengubah isi *field-field* yang dibutuhkan dan dapat mengklik tombol simpan untuk menyimpan perubahan. Lalu, sama seperti proses *create*, modal konfirmasi juga akan ditampilkan. Jika tidak dikonfirmasi oleh pengguna, maka akan kembali ke *form update* dan jika dikonfirmasi, maka

akan lanjut ke pengecekan validasi di *front end*. Jika *input* tidak valid, maka akan kembali ke *form update* dengan tanda di field yang tidak valid. Jika *input* sudah valid, maka sistem akan melakukan *hit* ke API dengan tipe PATCH. Jika respon tidak sukses, maka akan menampilkan notifikasi *error* dan kembali ke *form update*. Jika respon sukses, maka akan halaman diarahkan ke halaman *read* dan juga menampilkan notifikasi sukses.

D. Delete

Selain mengubah atau *update* sistem ERP yang dibuat juga dapat melakukan *delete* atau penghapusan yang dilakukan secara *soft delete* atau hanya mengubah status *field* 'deleted' menjadi *true* sehingga tidak dapat diakses. Proses *delete* ini tidak menggunakan halaman tersendiri, namun ada di halaman *read*.



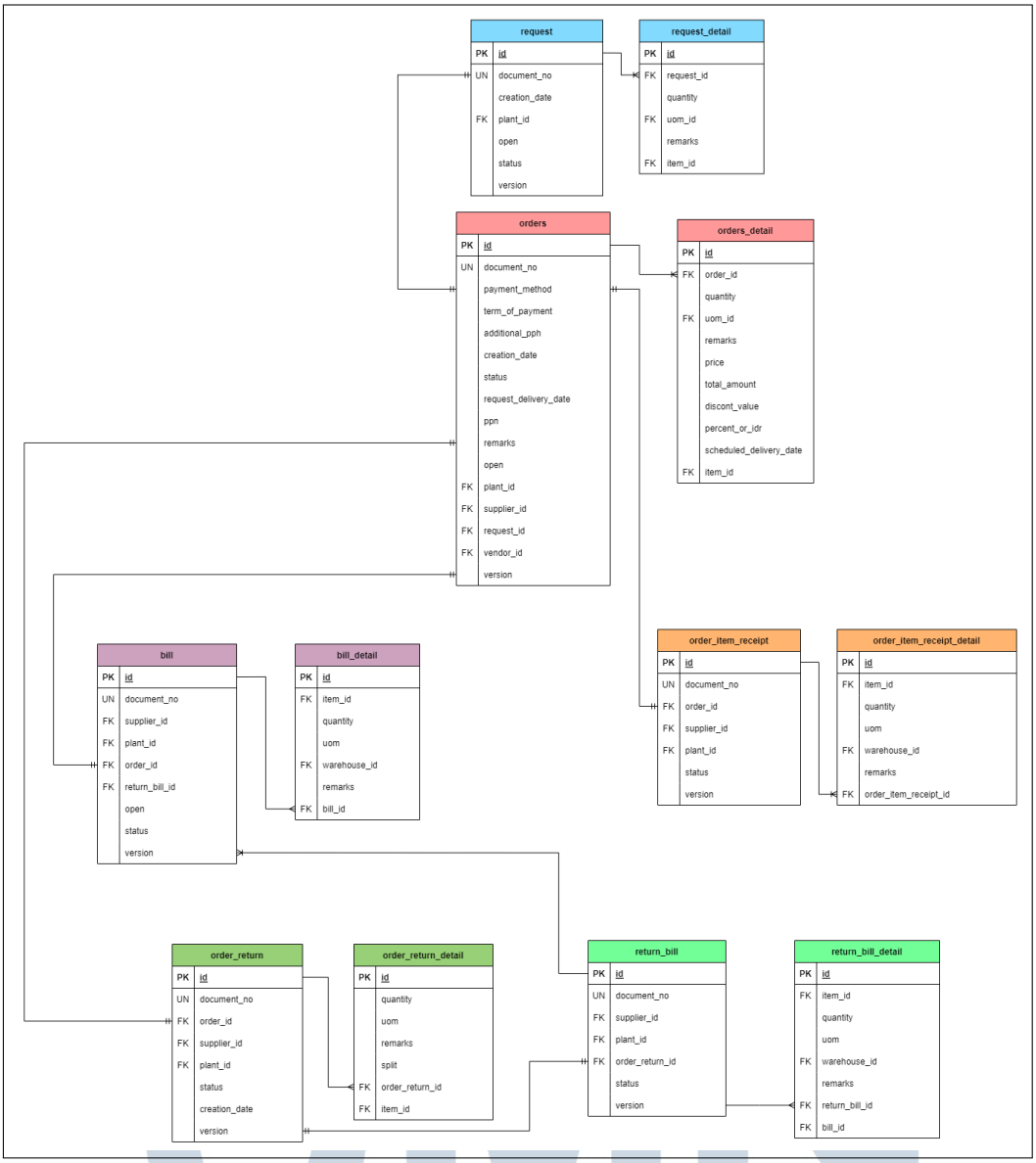
Gambar 3.9. Flowchart Delete

Gambar 3.18 merupakan alur melakukan *delete* untuk satu data di sistem ERP yang dibuat. Alurnya diawali dengan membuka halaman *read* karena *delete* tidak memiliki halaman tersendiri. Di dalam halaman *read* tersebut, terdapat tombol hapus atau *delete*. Jika diklik, sistem akan melakukan hit ke API dengan tipe

DELETE untuk melakukan *soft delete*. Jika respon tidak sukses, maka sistem akan menampilkan notifikasi *error* dan kembali ke halaman *read*. Jika respon sukses, maka halaman akan diarahkan ke halaman list dengan menampilkan notifikasi sukses.

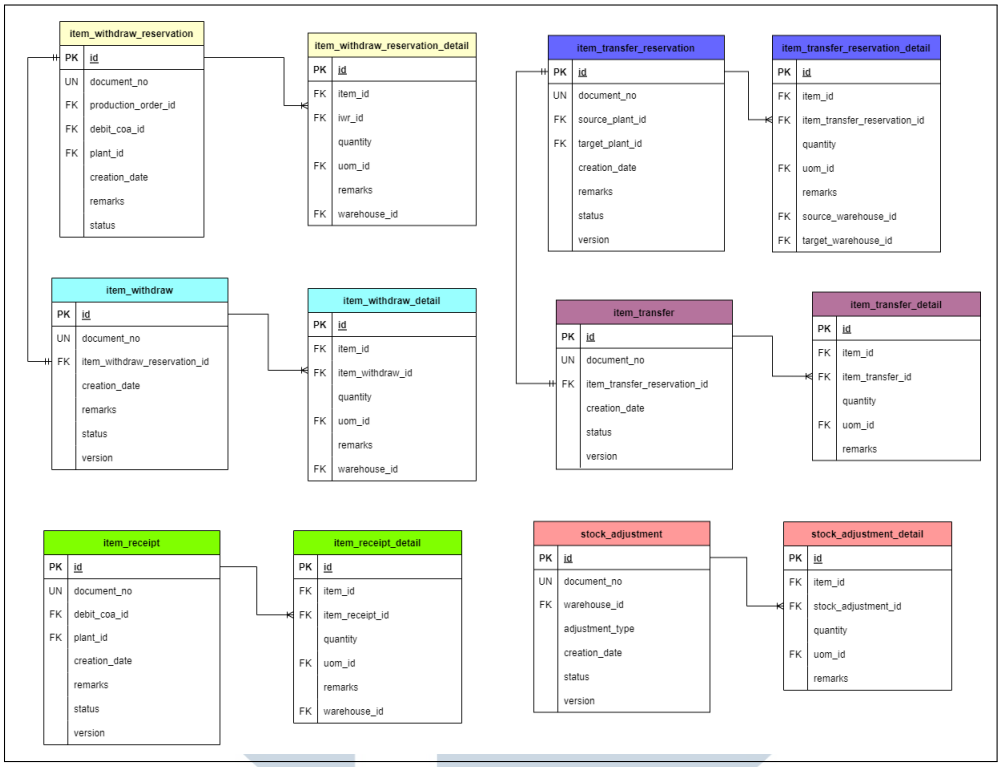
3.4.3 Entity Relation Diagram

Dalam perancangan sistem ERP dibuat juga *Entity Relation Diagram* atau ERP sebagai acuan *database* dalam proyek ini. ERD yang dibuat adalah ERD dari modul-modul yang dikerjakan bersama tim.



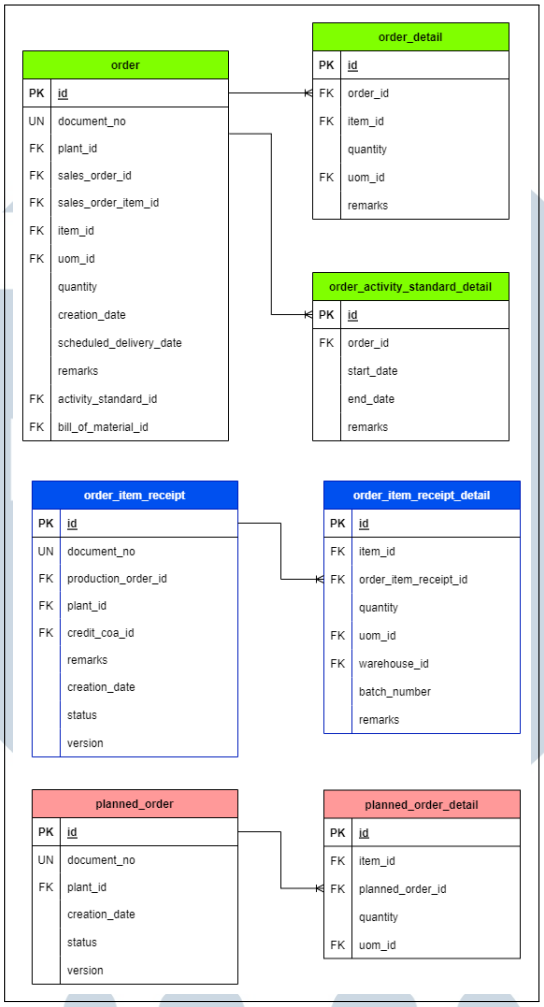
Gambar 3.10. ERD modul purchasing

Gambar 3.10 dari ERD modul purchasing. Setiap menu dari modul purchasing terdiri dua bagian, yaitu tabel utama atau *header* dan juga tabel detail. Setiap tabel utama dari sebuah menu memiliki relasi *one to many* dengan tabel detailnya. Yang berarti satu tabel *header* dapat memiliki banyak detail di dalamnya. Di dalam modul purchasing sendiri terdapat menu, yaitu purchase request, purchase order, purchase order item receipt, purchase bill, purchase order return, dan juga purchase return bill.



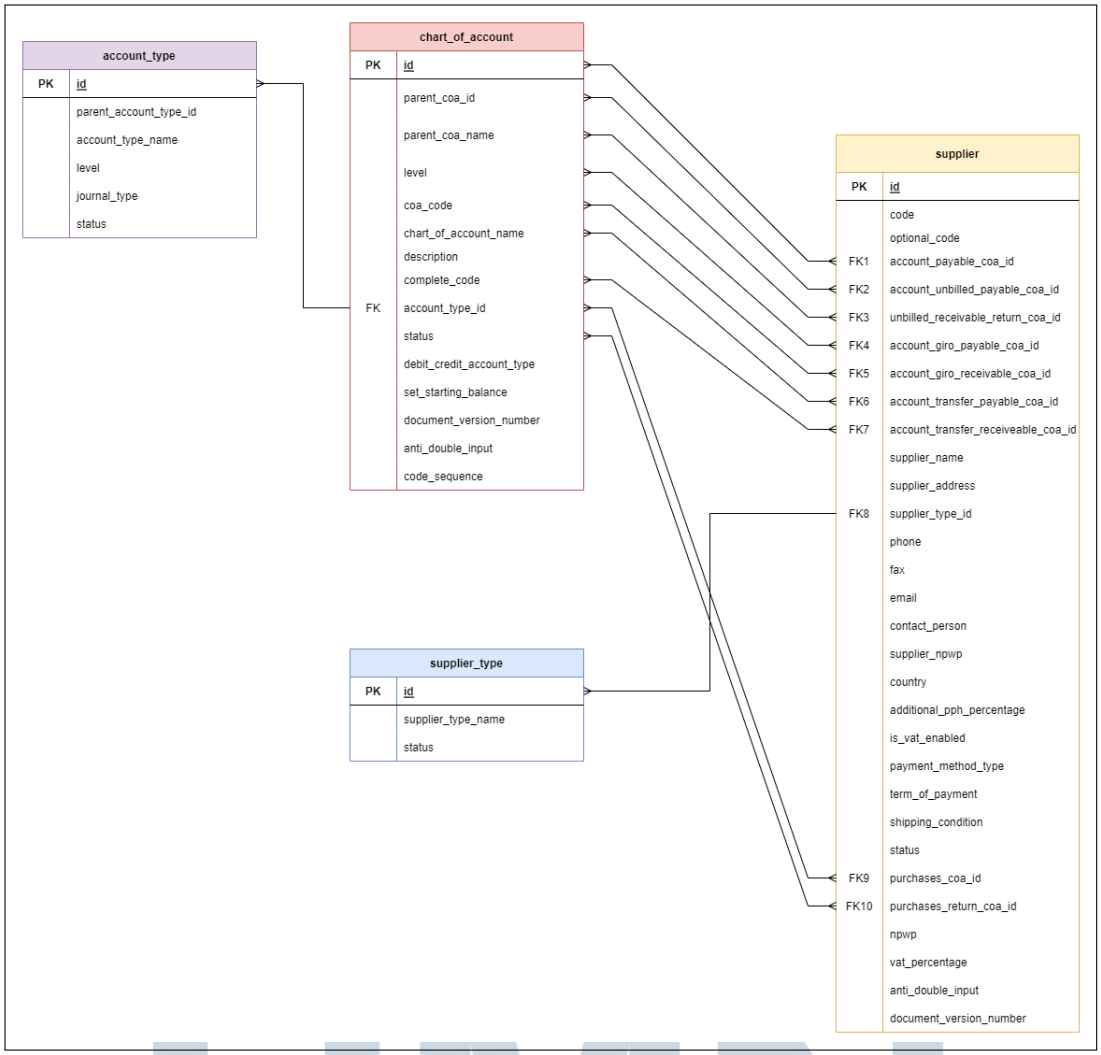
Gambar 3.11. ERD modul inventory

Gambar 3.11 adalah gambar dari ERD modul inventory yang dibuat oleh tim penulis. Sama seperti modul purchasing, modul inventory juga memiliki tabel utama atau *header* dan juga tabel detail untuk setiap menunya. Menu dari inventory sendiri terdiri dari item withdraw reservation, item withdraw, item transfer reservation, item transfer, item receipt, dan stock adjustment



Gambar 3.12. ERD modul production

Gambar 3.12 di atas merupakan gambar dari ERD modul production. Sama seperti dua modul sebelumnya, modul production ini juga memiliki tabel *header* dan juga tabel detail untuk setiap menunya. Menu atau submodul dari modul ini meliputi production order, production order item receipt, dan planned order.



Gambar 3.13. ERD modul production

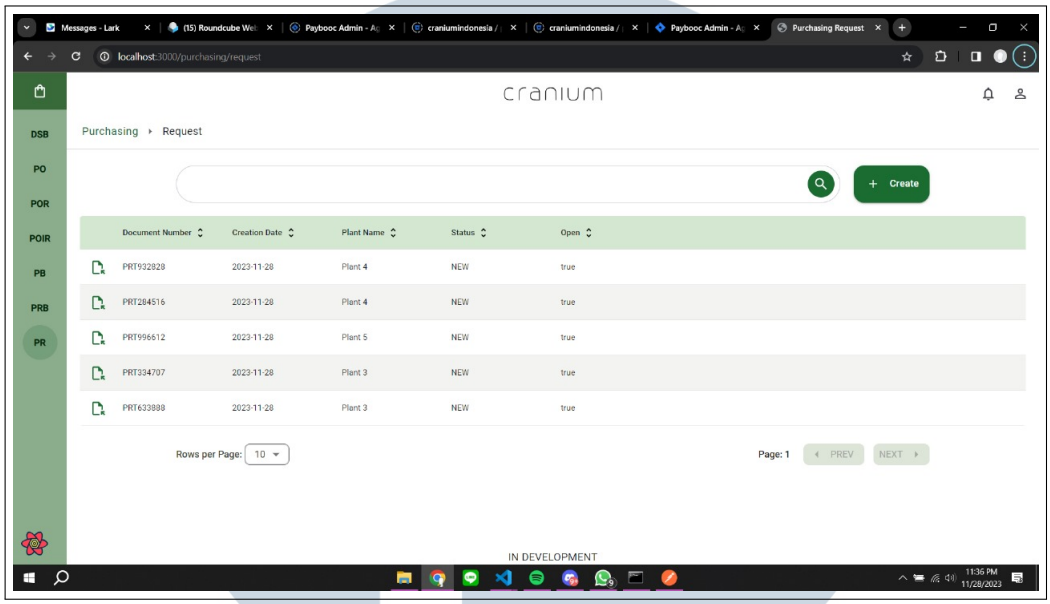
Gambar 3.13 di atas merupakan gambar dari ERD modul master yang menunya dibuat. Menu dari modul master tidak memiliki tabel *detail*, maka setiap menunya hanya ada satu tabel saja. Menu dalam ERD meliputi supplier, supplier type, account type dan juga tabel untuk chart of account yang memiliki hubungan dengan tabel-tabel yang dibuat.

3.5 Implementasi Halaman

Setelah perancangan, sistem ERP dapat dibuat. Tampilan dari sistem ini dibuat berdasarkan *design* yang diberikan oleh *designer*.

1. Halaman *List*

Halaman *list* merupakan halaman yang pertama terbuka jika membuka suatu menu dalam modul di sistem ERP.

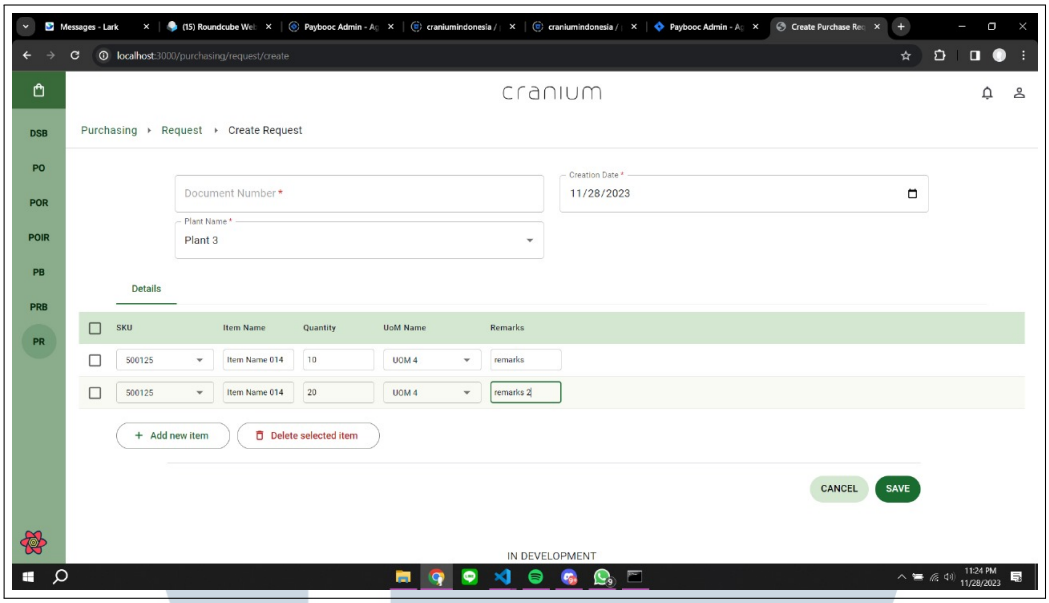


Gambar 3.14. Tampilan Halaman *List*

Halaman *list* sendiri berisikan tabel yang menunjukan *list* semua data yang belum terhapus yang ada di menu. Kolom-kolom pada tabel tersebut meliputi *field-field* yang ada di menu tersebut. Di dalam tabel ini juga terdapat tombol untuk membuka halaman *view* di kolom pertama setiap baris data. Selain tabel tersebut, terdapat juga tombol *create* yang menuju ke halaman *create*.

2. Halaman *Create*

   Halaman *create* akan terbuka setelah tombol *create* di halaman *list* diklik.

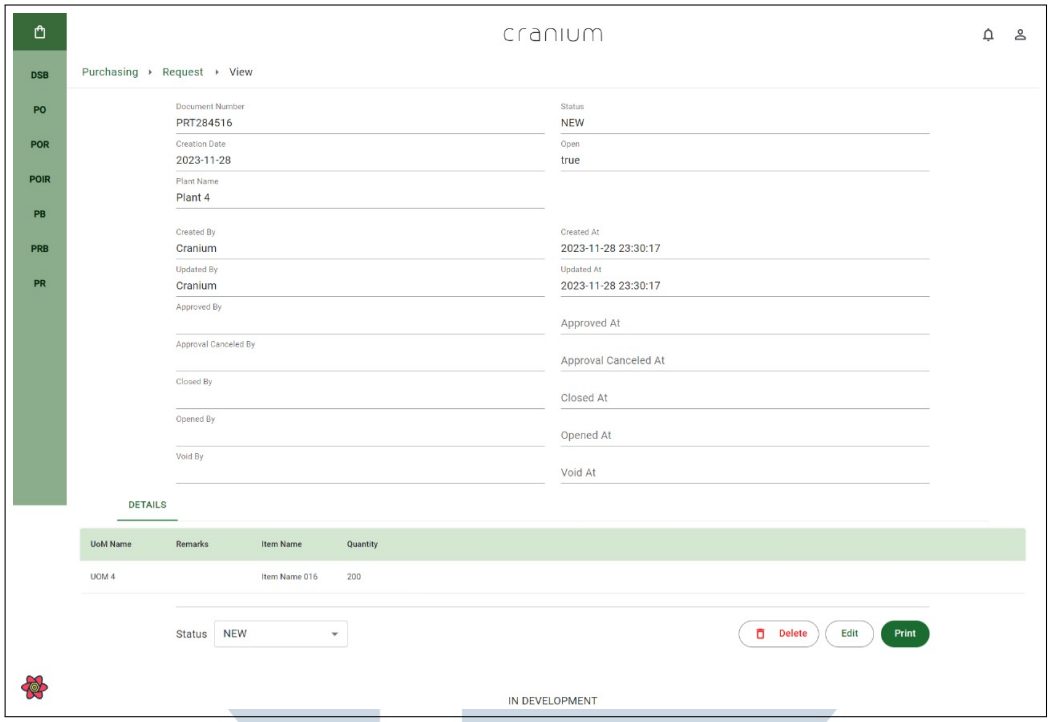


Gambar 3.15. Tampilan Halaman *Create*

Dalam halaman *create* ini terdapat *form* yang merupakan *field-field* yang ada dalam menu yang diakses. Di bawah *form* tersebut juga terdapat tabel *form* untuk data detail dari menu tersebut. Jika, semua *field* dalam *form* sudah terisi, pengguna dapat mengklik tombol *save* di bawah tabel *form* detail untuk menyimpan data yang sudah diisi.

3. Halaman *View*

   Halaman *view* dapat dibuka dengan mengklik tombol *view* pada tabel di halaman *list*.

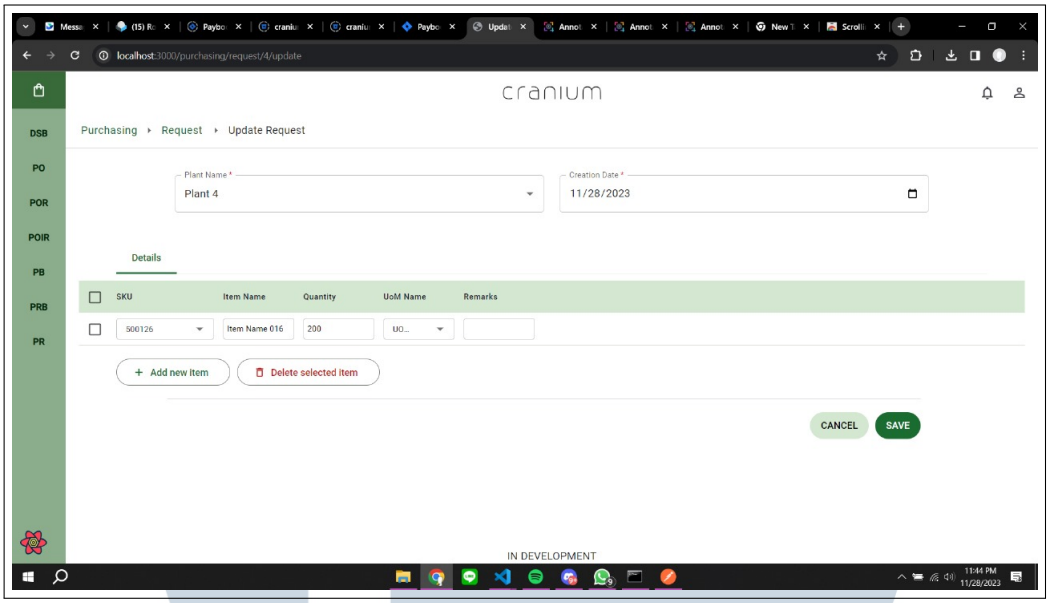


Gambar 3.16. Tampilan Halaman *View*

Di halaman *view* terdapat *field-field* dari menu yang diakses yang mana *value* datanya berasal dari database. Di bawahnya *field-field* tersebut juga terdapat tabel yang berisi data detail dari menu yang diakses. Di paling bawah halaman *view*, terdapat *button delete* yang akan memunculkan modal konfirmasi pada saat melakukan penghapusan dan juga tombol *update* yang mengarah ke halaman *update*.

4. Halaman *Update*

Halaman *update* dapat dibuka dengan mengklik tombol *update* atau ubah pada bagian bawah halaman *view*.

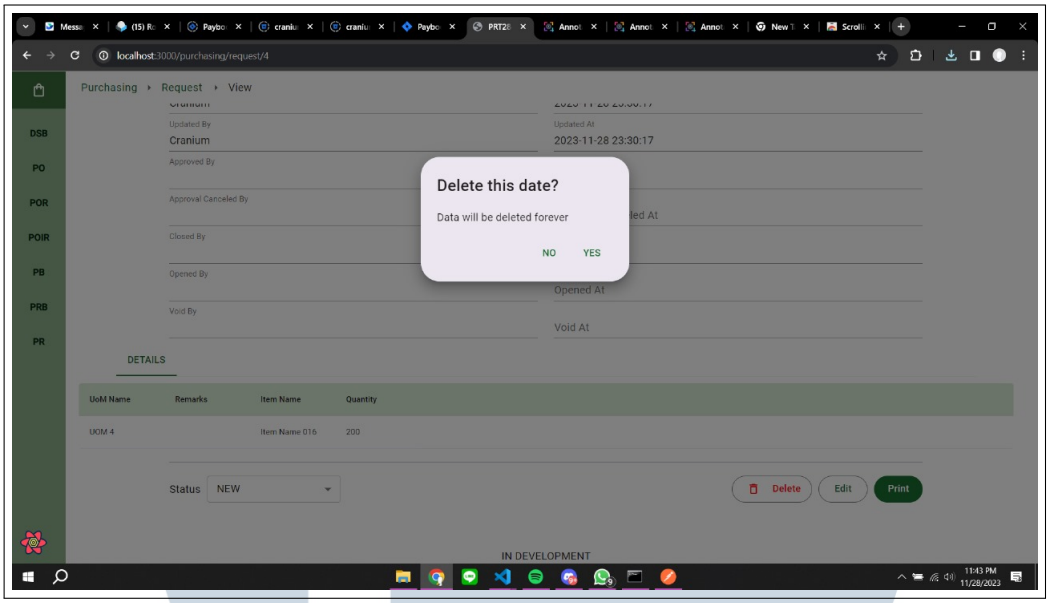


Gambar 3.17. Tampilan Halaman *Update*

Mirip seperti halaman *create*, halaman *update* juga terdapat *form* dari *field-field* dari menu yang diakses. Namun, *value* dari *field-field* tersebut sudah terisi dengan *default value* yang merupakan data yang diambil dari database. Di bawah *form* tersebut juga terdapat tabel *form* detail yang *value*nya juga sudah terisi oleh *default value*.

5. *Modal Delete*

Method delete tidak memiliki halaman tersendiri, melainkan hanya menggunakan modal konfirmasi. *Modal* tersebut dapat diakses melalui tombol *delete* di halaman *view*



Gambar 3.18. Tampilan *Modal Delete*

Modal delete merupakan sebuah modal konfirmasi, maka isi dari modal ini adalah konfirmasi kepada *user* terkait penghapusan sebuah data menu atau submodul.

3.6 Pembuatan Unit Test

Selain pembuatan halaman, dibuat juga *unit test*. Fungsi *unit test* sendiri adalah untuk menguji jalannya aplikasi sesuai dengan *requirement* yang ditentukan. Unit test yang dibuat penulis sendiri mencakup semua halaman yang dibuat. Yang berarti semua proses CRUD dibuatkan *unit test* masing-masing. Unit test dibuat menggunakan Jest yang merupakan *framework* testing untuk React. Dalam *unit test*, data yang digunakan bukan dari *database* asli, namun *mock* dari *database* tersebut. Proses *unit test* sendiri juga merupakan proses *mocking* atau meniru.

1. Unit Test *List*

   Dalam *unit test* halaman *list*, Jest akan memanggil halaman *list* dengan data dari hasil *mock* database. Di *unit test* ini, akan diperiksa tabel dalam halaman *list* berdasarkan data-test-id. Jika semua *field* dari menu tersedia, maka *unit test* berhasil.

2. Unit Test *View*

Halaman *view* akan dipanggil oleh Jest dengan data yang digunakan adalah salah satu data yang ada di dalam *mocked database*. Yang diperiksa oleh *unit test* halaman *view* ini adalah keberadaan dan kesesuaian data dari setiap *field* yang ditampilkan di halaman *view*. Jika semua *field* ada dan datanya sesuai dengan *mocked database*, maka *unit test* berhasil.

3. Unit Test *Create*

   Di *unit test* ini, halaman *create* yang berisikan *form* akan dipanggil. Pengetesan dilakukan dengan meniru pengguna saat memasukkan data di *field-field* yang ada di halaman *create*. Setelah semua data telah terisi, dilakukan juga peniruan saat pengguna melakukan *save*, yaitu menekan tombol *create*. Lalu, modal konfirmasi akan muncul dan *unit test* juga akan meniru untuk menekan tombol konfirmasi dalam modal tersebut. Setelah itu, dilakukan pengecekan keberadaan notifikasi *create* berhasil. Jika notifikasi tersebut ada, maka *unit test* berhasil.

4. Unit Test *Update*

   Sama seperti *unit test create*, *unit test update* juga akan memanggil halaman dengan *form* di dalamnya. Perbedaannya adalah *unit test update* akan memanggil halaman *update*, yang mana *field* dalam *form* sudah terisi dengan salah satu data dari *mocked database* yang dipanggil. Maka, pengetesan dilakukan dengan mengubah data dalam *field-field* tersebut. Unit test *update* juga akan melakukan peniruan saat pengguna melakukan *save*, yaitu saat mengklik tombol *save* dan tombol konfirmasi yang ada di dalam modal konfirmasi yang muncul setelah tombol *save* diklik. Sama seperti *unit test* create, dilakukan juga pengecekan keberadaan notifikasi *update* berhasil. Jika notifikasi tersebut ada, maka *unit test* berhasil.

5. Unit Test *Delete*

   Proses *delete* tidak menggunakan satu halaman terpisah, maka dalam *unit test delete*, halaman *view* dengan salah satu data dari *mocked database* yang dipanggil. Dari halaman tersebut, *unit test* akan memeriksa kesamaan dari salah satu *field* yang ada di halaman *view*. Tujuannya adalah untuk memastikan data yang dipanggil adalah data yang benar. Jika sudah benar, maka dilakukan peniruan saat pengguna melakukan delete, yaitu saat pengguna mengklik tombol *delete* dan juga konfirmasinya yang muncul saat setelah tombol *delete* dimunculkan. Sama seperti *unit test create* dan *update*,

unit test delete juga akan melakukan pengecekan keberadaan notifikasi *delete* berhasil. Jika notifikasi tersebut ditemukan, maka *unit test* berhasil.

3.7 Kendala dan Solusi yang Ditemukan

Berikut adalah beberapa kesulitan yang dihadapi selama pelaksanaan kerja magang.

- Penggunaan beberapa teknologi seperti react dan *component*nya yang belum pernah digunakan sebelumnya. Selain itu, Jest yang digunakan untuk pembuatan *unit test* pada proyek ERP ini juga belum pernah digunakan sebelumnya.
- Terjadi banyak perubahan saat pembuatan proyek. Perubahan yang dimaksud seperti perubahan *field*

Solusi yang ditemukan saat pelaksanaan magang yaitu sebagai berikut.

- Mencari referensi atau pun sumber belajar terkait react dan *component*nya, dan juga mencari referensi dan mencoba Jest untuk membuat *unit test* react.
- Menyesuaikan diri dengan adanya perubahan yang terjadi saat pengerjaan proyek.