# PENGEMBANGAN INVENTORI EDC BERBASIS WEB MENGGUNAKAN TEKNOLOGI SPRING BOOT DAN THYMELEAF


Disusun Oleh:

N a m a : Nadira Adiva Wibowo

NIM : 20523202

**PROGRAM STUDI INFORMATIKA – PROGRAM SARJANA**

**FAKULTAS TEKNOLOGI INDUSTRI**

**UNIVERSITAS ISLAM INDONESIA**

**2024**

**HALAMAN PENGESAHAN DOSEN PEMBIMBING**

# PENGEMBANGAN INVENTORI EDC BERBASIS WEB MENGGUNAKAN TEKNOLOGI SPRING BOOT DAN THYMELEAF

## TUGAS AKHIR JALUR MAGANG

Disusun Oleh:

| | |
|---|---|
| N a m a | : Nadira Adiva Wibowo |
| NIM | : 20523202 |

Yogyakarta, 23 Juli 2024

Pembimbing,

( Moh. Idris, S.Kom., M.Kom. )

**HALAMAN PENGESAHAN DOSEN PENGUJI**

# PENGEMBANGAN INVENTORI EDC BERBASIS WEB MENGGUNAKAN TEKNOLOGI SPRING BOOT DAN THYMELEAF

## TUGAS AKHIR JALUR MAGANG

Telah dipertahankan di depan sidang penguji sebagai salah satu syarat untuk memperoleh gelar Sarjana Komputer dari Program Studi Informatika – Program Sarjana di Fakultas Teknologi Industri Universitas Islam Indonesia

Yogyakarta, 23 Juli 2024

Tim Penguji

Moh. Idris, S.Kom., M.Kom. _____

**Anggota 1**

Dr. Raden Teduh Dirgahayu, S.T., M.Sc. _____

**Anggota 2**

Feri Wijayanto, S.T., M.T. _____

Mengetahui,

Ketua Program Studi Informatika – Program Sarjana

Fakultas Teknologi Industri

Universitas Islam Indonesia

( Dhomas Hatta Fudholi, S.T., M.Eng., Ph.D. )

## HALAMAN PERNYATAAN KEASLIAN TUGAS AKHIR

Yang bertanda tangan di bawah ini:

Nama : Nadira Adiva Wibowo

NIM : 20523202

Tugas akhir dengan judul:

# PENGEMBANGAN INVENTORI EDC BERBASIS WEB MENGGUNAKAN TEKNOLOGI SPRING BOOT DAN THYMELEAF

Menyatakan bahwa seluruh komponen dan isi dalam tugas akhir ini adalah hasil karya saya sendiri. Apabila di kemudian hari terbukti ada beberapa bagian dari karya ini adalah bukan hasil karya sendiri, tugas akhir yang diajukan sebagai hasil karya sendiri ini siap ditarik kembali dan siap menanggung risiko dan konsekuensi apapun.

Demikian surat pernyataan ini dibuat, semoga dapat dipergunakan sebagaimana mestinya.

Yogyakarta, 4 Juli 2024


( Nadira Adiva Wibowo )

# HALAMAN PERSEMBAHAN

Penulis persembahkan laporan akhir ini untuk keluarga kecilnya, yaitu ayah, bunda, dan adik, yang senantiasa mendoakan dan memberikan perhatian serta dukungan yang sepenuh hati kepada penulis. Penulis juga persembahkan laporan akhir ini untuk diri penulis sendiri, yang selalu berusaha dengan keras dan maksimal untuk melewati semua kesulitannya. Maka dari itu, laporan akhir ini mampu penulis selesaikan sebagai tanggung jawab seorang mahasiswa.

## HALAMAN MOTO

“Maka, sesungguhnya bersama kesulitan ada kemudahan.”
- QS. Al-Insyirah: 5

“Ingatlah, sesungguhnya pertolongan Allah itu dekat.”
- QS. Al-Baqarah: 214

“Dirimu di masa depan, butuh kamu di masa ini untuk terus berjuang.”
- Sesakata

# KATA PENGANTAR

Assalamu'alaikum Warahmatullahi Wabarakatuh.

Alhamdulillah, penulis panjatkan segala puji dan rasa syukur kepada Allah Swt. yang telah melimpahkan rahmat dan karunia-Nya sehingga penulis dapat menyelesaikan tugas laporan akhir jalur magang yang berjudul "Pengembangan Inventori EDC Berbasis Web Menggunakan Teknologi Spring Boot dan Thymeleaf". Laporan akhir ini ditulis untuk memenuhi persyaratan kelulusan dari penjaluran magang di Program Studi Informatika - Program Sarjana, Fakultas Teknologi Industri, Universitas Islam Indonesia. Penulisan laporan akhir ini tentunya tidak terpisahkan dari kebaikan dan dukungan berbagai pihak. Maka dari itu, penulis menyampaikan rasa terima kasih yang sebesar-besarnya kepada:

1. Allah Swt. yang senantiasa memberikan rahmat, pertolongan, dan kekuatan kepada penulis semasa mengerjakan laporan akhir ini maupun kuliah.
2. Ayah dan Bunda penulis, Arif Wibowo dan Eni Yuni Astuti, yang senantiasa mendoakan dan memberikan perhatian, bantuan, serta dukungan yang sepenuh hati secara emosional dan materiil semasa pengerjaan laporan akhir maupun kuliah.
3. Adik penulis, Mutia Aisha Wibowo, yang siap sedia membantu, saling berbagi kesenangan, dan menyemangati semasa pengerjaan laporan akhir maupun kuliah.
4. Kedua eyang penulis, Affandi dan Siswanti, serta keluarga besar penulis, Bani Irsyad dan Bani Notowihardjo, yang memberikan doa, dukungan, dan nasihat.
5. Prof. Fathul Wahid, S.T., M.Sc., Ph.D., selaku Rektor Universitas Islam Indonesia.
6. Bapak Dr. Raden Teduh Dirgahayu, S.T., M.Sc., selaku Ketua Jurusan Informatika dan Bapak Dhomas Hatta Fudholi, S.T., M.Eng., Ph.D., selaku Ketua Program Studi Informatika – Program Sarjana.
7. Bapak Moh. Idris, S.Kom., M.Kom., selaku dosen pembimbing, yang meluangkan waktu untuk membimbing dan meninjau pengerjaan laporan akhir penulis dengan sabar dan perhatian.
8. Bapak dan Ibu dosen Program Studi Informatika, yang berjasa dalam memberikan berbagai ilmu yang bermanfaat kepada penulis semasa kuliah.
9. Bapak Agus Suhardi selaku *supervisor* dan rekan kerja di PT Tata Sarana Makmur, yang membimbing penulis dengan baik semasa pelaksanaan magang.
10. Bapak Marchi dan Bapak Oemar selaku *leader* di bidang *software application development* dan *information technology* serta semua anggota bidang di PT Tata Sarana Makmur, yang mendukung penulis semasa pelaksanaan magang.

11. Teman-teman Uwuwuww penulis, Fatimah Azzahra Kusuma Dewi, Intan Nabila Azmi, Kartika Salma, Mayla Ayyuni Sonya, Sonya Ainurohmah, dan Syuhda Fakhrunnisa, yang senantiasa terus meyakinkan, mendukung, menyemangati, dan mendengar keluh kesah semasa pengerjaan laporan akhir ini serta menemani suka duka perjalanan semasa kuliah hingga nanti.
12. Teman-teman Ghaniun YK penulis, Amanda Irbah, Rosidah Tamara, dan Salsabila Sampurnani, yang senantiasa berbagi momen dan cerita, dapat diandalkan, serta mendukung dan menyemangati satu sama lain sejak SMA hingga saat ini.
13. Teman-teman SMP penulis, Auliya Latifah, Bintang Permata Hati, Bunga Nafisa Mujahida, Saida Rahma Afifi, Sarah Nur Fadhila, yang senantiasa mendukung dan menyemangati satu sama lain sejak SMP hingga saat ini.
14. Teman SD penulis, Alvina Kaniarachma, yang senantiasa berbagi suka duka serta mendukung dan menyemangati satu sama lain sejak SD hingga saat ini.
15. Adik-adik tingkat penulis, angkatan 2021 dan 2022, yang memberikan warna indah tersendiri semasa kuliah dengan senantiasa mendukung dan menyemangati.
16. Semua pihak yang tidak dapat penulis sebutkan satu per satu, terima kasih atas semua dukungannya secara langsung maupun tidak langsung dalam mengerjakan laporan akhir ini.

Semoga segala kebaikan berupa doa, ilmu, bimbingan, perhatian, bantuan, dukungan, dan, nasihat yang telah diberikan kepada penulis mendapatkan balasan yang lebih baik dari Allah Swt. Penulis juga menyadari dan memahami bahwa laporan akhir ini belum sempurna sehingga penulis terbuka untuk menerima saran dan kritik yang membangun.

Wassalamu'alaikum Warahmatullahi Wabarakatuh.

Yogyakarta, 4 Juli 2024

( Nadira Adiva Wibowo )

# SARI

PT TSMakmur merupakan perusahaan yang berfokus dalam industri *fintech* di Indonesia. Di samping menghasilkan produk seperti QREDC, TSMakmur juga menyediakan jasa untuk mengembangkan suatu produk berupa aplikasi atau sistem yang masih berkaitan dengan *fintech* kepada pihak lain (klien perusahaan). Salah satunya adalah inventori EDC berbasis web yang bertujuan mengelola dan memantau perkembangan EDC mulai dari pengadaan, pemrosesan, penyimpanan sebagai persediaan, hingga pendistribusian atau pengembalian, dengan data yang akurat dan termutakhir. Dengan begitu, kebingungan yang dialami oleh suatu klien perusahaan mengenai bertambahnya jumlah EDC yang harus dikelola dengan kebutuhan data yang akurat dan termutakhir dapat ditangani. Pengembangannya melalui implementasi metode *waterfall* dengan teknologi *back-end* berupa bahasa pemrograman Java, *framework* Spring Boot, dan *database* PostgreSQL, serta teknologi *front-end* berupa *template* Thymeleaf, *library* jQuery, dan AJAX.

Hasil pengembangannya juga berjalan dengan baik karena sudah sesuai dengan kebutuhan yang ditentukan dan mencapai tujuan yang diharapkan melalui berbagai fitur yang telah diimplementasikan. Tiga fitur diantaranya adalah sebagai berikut. Pertama, fitur pengembalian EDC yang rusak untuk mengelola dan memantau pengembalian tersebut dari klien perusahaan ke vendor *arrival*. Fungsinya antara lain menambahkan EDC yang rusak sekaligus menentukan pembuatan pengembalian, mengajukan permintaan pengembalian berdasarkan *id* untuk disetujui atau ditolak, dan menerima pengembalian berdasarkan *id*. Kedua, fitur *merchant* untuk mengelola entitas *merchant* yang menjadi tujuan distribusi EDC dari SP. Fungsinya antara lain membuat, memperbarui, atau menghapus *merchant* dan mengajukannya sebagai permintaan berdasarkan *id* untuk disetujui atau ditolak. Ketiga, fitur SP untuk mengelola entitas SP yang menjadi tujuan distribusi EDC dari vendor *arrival* atau vendor *service management* dan menjadi syarat untuk membuat akun *user* SP *checker maker*. Fungsinya seperti penjelasan pada fitur SP, tetapi operasinya untuk entitas SP.

Kata kunci: EDC, inventori, Spring Boot, Thymeleaf, *Waterfall*, pengembalian, *merchant*, SP.

# GLOSARIUM

| | |
|---|---|
| *Dependency injection* | otomatisasi pembuatan objek yang bergantung pada objek lain. |
| DTO | *data transfer object* sebagai penampung data dari *server side* untuk dikirimkan ke *client side*. |
| EDC | *electronic data capture* sebagai pembayaran digital dengan kartu debit, kartu kredit, *e-wallet*, atau aplikasi *mobile banking*. |
| *Endpoint* | alamat spesifik untuk mengakses sumber daya sesuai permintaan API. |
| Inventori | persediaan barang pada perusahaan atau toko untuk didistribusikan. |
| Klien perusahaan | klien TSMakmur yang membutuhkan inventori EDC berbasis web dan perusahaan yang memproses EDC hingga siap didistribusikan kepada vendor *arrival* atau vendor *service management*. |
| *Member* | bank yang memiliki hak atas EDC. |
| *Merchant* | konsumen akhir yang membutuhkan EDC. |
| SP | lokasi layanan sebagai penerima distribusi EDC dari vendor *arrival* atau vendor *service management* dan sebagai distributor EDC kepada *merchant*. |
| Spring Boot | *framework* aplikasi Spring yang mempermudah proses pengembangan. |
| Spring Initializr | alat *generator* aplikasi Spring Boot berbasis web yang menyediakan struktur aplikasi dengan dependensi yang dibutuhkan. |
| Thymeleaf | *template engine* Java yang terintegrasi dengan *framework* Spring untuk menghasilkan tampilan web yang lebih dinamis. |
| VO | *value object* sebagai penampung data dari *client side* untuk dikirimkan ke *server side*. |
| Vendor *arrival* | pemasok EDC kepada klien perusahaan, penerima distribusi EDC dari klien perusahaan, pemilik sejumlah SP, distributor EDC kepada SP, serta bertanggung jawab atas pengembalian dan perbaikan EDC yang rusak. |
| Vendor *service* | penerima distribusi EDC dari klien perusahaan dan distributor EDC kepada SP. |
| Vendor SP | pemilik sejumlah SP. |
| *Waterfall* | metode pengembangan sistem yang memiliki lima tahapan berurutan dari awal hingga akhir tanpa pengulangan. |

# DAFTAR ISI

# DAFTAR TABEL

# DAFTAR GAMBAR

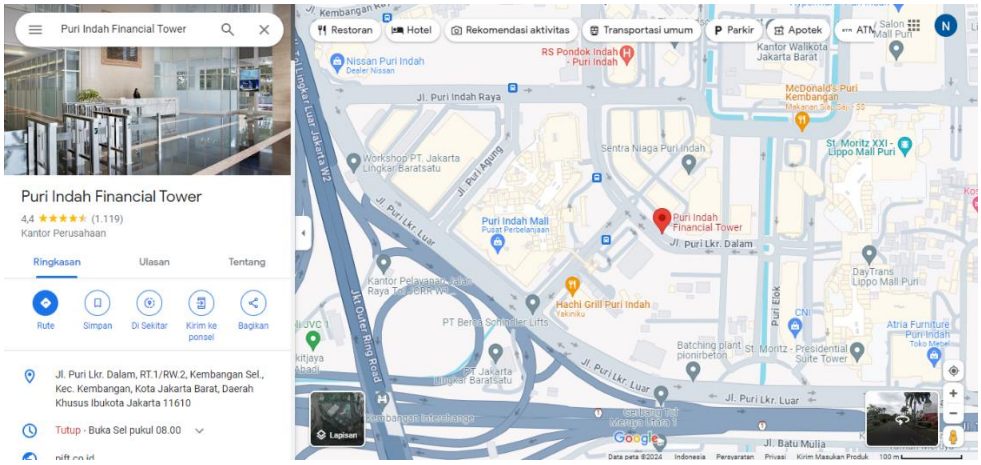
# BAB I
# PENDAHULUAN

## 1.1 Latar Belakang

Perkembangan teknologi yang semakin pesat telah memengaruhi gaya hidup manusia dan mengubah cara atau proses dalam melakukan kegiatan. Kini manusia cenderung menyukai kecepatan dan kepraktisan dalam melakukan kegiatan yang didukung dengan teknologi yang memberikan kemudahan, seperti *financial technology* (*fintech*). *Fintech* merupakan teknologi yang berkaitan dengan kegiatan finansial atau keuangan. Indonesia sendiri memiliki beberapa perusahaan yang beroperasi dalam industri *fintech*, salah satunya PT Tata Sarana Makmur.

PT Tata Sarana Makmur atau TSMakmur merupakan anak dari perusahaan teknologi *software house* di Jakarta, yaitu PT Tata Sarana Mandiri atau TSMandiri, yang lebih berfokus pada *fintech*. Lebih detailnya, TSMakmur merupakan sebuah perusahaan yang menciptakan dan menyediakan teknologi serta solusi pada bidang pembayaran digital secara efektif dan aman. TSMakmur didirikan pada tahun 2015 dan beroperasi di Gedung Puri Indah Financial Tower (PIFT), tepatnya di lantai 20 unit 2007-2008, dengan alamat lengkapnya berada di Jalan Puri Lingkar Dalam Blok T8, Kembangan Selatan, Kembangan, Jakarta Barat. Penampakan gedungnya jika dilihat dari depan seperti pada Gambar 1.1 dan lokasinya jika dilihat melalui Google Maps seperti pada Gambar 1.2.

Gambar 1.1 Tampak Depan Gedung Puri Indah Financial Tower (PIFT)



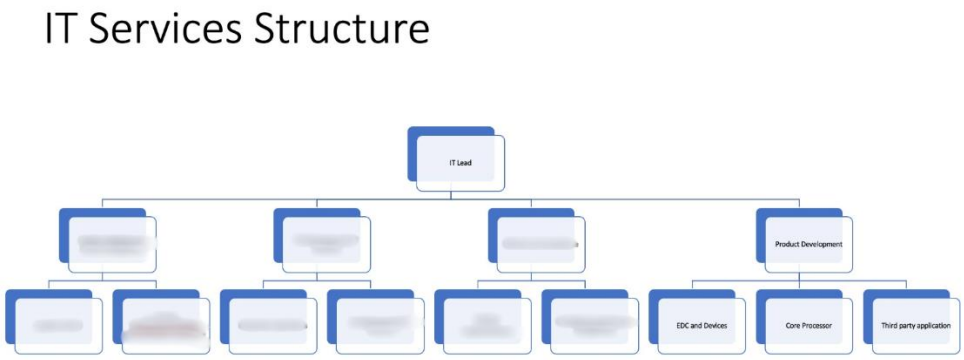

Gambar 1.2 Google Maps Lokasi Gedung Puri Indah Financial Tower (PIFT)

TSMakmur telah menghasilkan beberapa produk unggul selama masa beroperasinya, antara lain yaitu QREDC V1, UROVO, dan WIZARPOS. Ketiga produk tersebut merupakan perangkat *electronic data capture* (EDC) seluler yang dirancang dan digunakan sebagai solusi pembayaran digital yang ekonomis, aman, dan ringan. EDC dilengkapi dengan *quick response code Indonesian standard* (QRIS) yang dapat dilakukan *scanning* menggunakan *e-wallet* atau aplikasi *mobile banking* sebagai proses pembayaran oleh konsumen atau pembeli.

Untuk menghasilkan beberapa produk teknologi tersebut, diperlukan banyak tenaga keahlian lokal di bidang *information technology* (IT) yang mendukung dan terlibat dalam pengembangan serta pengelolaannya. Tenaga keahlian tersebut ditempatkan di bidang *software application development* dan *information technology* yang dipimpin oleh Bapak Oemar selaku *vice president*. Bidang *software application development* dan *information technology* memiliki berbagai divisi dengan beragam peran di dalamnya. Berbagai divisi tersebut dapat dilihat pada Gambar 1.3, sedangkan beragam peran tersebut meliputi UI/UX *designer*, *android developer*, *web developer* (*front-end, back-end, full-stack*), *DevOps engineer*, *quality engineer*, dan lain sebagainya.

Penulis melaksanakan magang di TSMakmur secara *work from home* (WFH) selama 6 bulan terhitung mulai tanggal 4 September 2023 hingga tanggal 4 Maret 2024 dan ditempatkan di divisi *third party application*, yang dapat dilihat pada Gambar 1.3, dengan peran awalnya sebagai *back-end web developer*. Divisi *third party application* bertanggung jawab mengelola pengembangan suatu produk berupa aplikasi atau sistem yang masih berkaitan dengan *fintech* kepada pihak lain (klien perusahaan). Dengan begitu, dapat disimpulkan bahwa TSMakmur tidak hanya menghasilkan produk, tetapi juga menawarkan jasa. Terdapat salah satu proyek

yang dikerjakan di divisi tersebut dan menjadi tugas penulis selama pelaksanaan magang, yaitu pengembangan inventori EDC berbasis web. Dikarenakan proyek tersebut termasuk proyek berskala sedang, sementara proyek hanya dikerjakan oleh dua anggota, maka penulis dialihkan perannya dari *back-end web developer* menjadi *full-stack web developer*.




Gambar 1.3 Struktur Organisasi Bidang *Software Application Development* dan *Information Technology*

Proyek pengembangan inventori EDC berbasis web tersebut merupakan permintaan dari salah satu klien perusahaan. Penulis dilibatkan dalam proyek ini selama 6 bulan terhitung seperti waktu pelaksanaan magang. Proyek ini secara keseluruhan memiliki tujuan inti, yaitu mengelola dan memantau perkembangan EDC yang meliputi pengadaan, pemrosesan hingga masuk ke penyimpanan sebagai persediaan, pendistribusian, maupun pengembalian, dengan data yang akurat dan termutakhir. Tujuan inti tersebut akan menangani dua permasalahan yang dialami oleh klien perusahaan. Pertama, bertambahnya jumlah EDC yang harus dikelola. Hal tersebut mengakibatkan kebingungan bagi klien perusahaan untuk mengetahui antara EDC yang belum dikelola dan sudah dikelola. Apalagi jika ternyata terdapat EDC yang mengalami kerusakan ketika diproses sehingga perlu dikembalikan untuk diperbaiki. Kedua, kebutuhan data yang akurat dan termutakhir.

Proyek tersebut juga melibatkan lima entitas penting, yaitu klien perusahaan, vendor, *service point* (SP), *merchant*, dan *member*. Klien perusahaan adalah pihak yang menerima EDC dari vendor *arrival* untuk diproses hingga siap digunakan dan masuk ke penyimpanan sebagai persediaan untuk kemudian didistribusikan kepada vendor *arrival* kembali atau vendor *service management*. Vendor terbagi menjadi tiga, yaitu vendor *arrival*, vendor *service management*, dan vendor *service point* (SP). Vendor *arrival* adalah pemasok EDC kepada klien perusahaan dan dapat memiliki sejumlah SP yang beroperasi di bawahnya sehingga dapat menerima EDC dari klien perusahaan untuk didistribusikan kepada SP yang dimilikinya. Vendor *arrival* juga

menerima pengembalian berupa EDC yang mengalami kerusakan ketika diproses oleh klien perusahaan untuk ditangani perbaikannya. Vendor *service management* adalah pihak yang menerima EDC dari klien perusahaan untuk didistribusikan kepada SP manapun. Vendor SP adalah pihak yang juga memiliki sejumlah SP yang beroperasi di bawahnya. SP adalah titik lokasi layanan yang menerima EDC dari vendor *arrival* atau vendor *service management* untuk didistribusikan atau disediakan kepada *merchant*. *Merchant* adalah konsumen akhir yang membutuhkan EDC. *Member* adalah bank yang memiliki hak atas EDC tersebut.

Selain itu, penulis juga terlibat ke dalam dua proyek lainnya yang waktu pengerjaannya paralel dengan proyek pengembangan inventori EDC berbasis web, yaitu *e-parking* pada rentang tanggal 5 – 11 Oktober 2023 serta 7 – 9 November 2023 dan *third party handle* (TPH) pada rentang tanggal 16 – 20 Oktober 2023. Proyek *e-parking* merupakan pembayaran parkir elektronik di Semarang dan Surabaya menggunakan produk QREDC V1 yang menghasilkan QRIS untuk dilakukan *scanning* menggunakan *e-wallet* atau aplikasi *mobile banking*. Penulis ditugaskan untuk membuat filter *user* dan *transaction* dengan tujuan mengoptimalkan fungsi sistem manajemen *e-parking* dalam melakukan pencarian yang memenuhi kondisi tertentu. Penulis juga ditugaskan untuk mengambil semua *app errors* dan *app params* dari *database* untuk ditampilkan di *dashboard* admin dengan tujuan mengoptimalkan fungsi sistem dalam melakukan pemantauan *error* dan penggunaan *third party* pada sistem. Sementara itu, proyek TPH merupakan proyek yang menjadi penghubung klien-klien perusahaan dengan *application programming interface* (API) pada sistem *multi channel payment* (MCP). Penulis ditugaskan untuk mengambil semua *partner request* (klien-klien perusahaan) yang ingin menggunakan API pada MCP dan membuatnya menjadi *endpoint* dengan tujuan dikonsumsi oleh bagian *front-end*.

Berdasarkan keterlibatan penulis dalam tiga proyek yang telah dijelaskan sebelumnya, proyek pengembangan inventori EDC berbasis web diangkat menjadi topik laporan akhir karena menurut penulis proyek tersebut memiliki lingkup yang cukup besar dan kompleksitas yang tinggi. Hal tersebut ditunjukkan dari adanya lima entitas penting yang terlibat dalam proyek dan memiliki peran masing-masing yang saling terhubung satu sama lain. Selain itu, penulis memiliki kontribusi yang paling banyak dalam proyek tersebut dibandingkan dengan dua proyek lainnya. Kehadiran laporan akhir dengan topik tersebut juga dapat menjadi referensi atau contoh implementasi nyata oleh pembaca yang tertarik dalam mengembangkan inventori suatu barang berbasis web.

## 1.2 Ruang Lingkup

Laporan akhir ini akan berfokus membahas mengenai proyek pengembangan inventori EDC berbasis web sesuai dengan topik yang diangkat. Proyek ini telah dijelaskan dengan cukup detail pada subbab sebelumnya. Proyek telah dikerjakan selama 2 minggu oleh rekan kerja sebelum akhirnya penulis diajak terlibat ke dalam pengerjaannya selama 6 bulan terhitung seperti waktu pelaksanaan magang. Proyek hanya dikerjakan oleh dua anggota, yaitu rekan kerja dan penulis, dengan peran penulis sebagai *full-stack web developer*. Penulis berkontribusi dalam tugas-tugas berskala besar, sedang, ataupun kecil dalam proyek ini. Beberapa kontribusi penulis yang termasuk berskala besar dan akan dibahas pada laporan akhir ini antara lain mengembangkan fitur pengembalian EDC yang rusak, fitur *merchant*, dan fitur SP.

Proyek ini dikembangkan dengan metode pengembangan *waterfall* dan teknologi pada sisi *back-end* serta *front-end*. Pada sisi *back-end*, teknologi yang digunakan adalah kombinasi dari bahasa pemrograman Java, *framework* Spring Boot, dan *database* PostgreSQL, sebagai pemrosesan data yang berhubungan dengan *database* untuk kemudian diatur logika bisnis atau fungsinya. Pada sisi *front-end*, teknologi yang digunakan adalah kombinasi dari *template* web desain Metronic (HTML, CSS, JavaScript, *framework* Bootstrap), *template* Thymeleaf, *library* jQuery, dan AJAX, sebagai pengaturan interaksi pengguna pada tampilan halaman web.

## 1.3 Tujuan

Tujuan dari proyek pengembangan inventori EDC berbasis web yang dikerjakan penulis adalah sebagai berikut:

a. Mengelola dan memantau pengembalian EDC yang mengalami kerusakan ketika diproses oleh klien perusahaan kepada vendor *arrival*.
b. Mengelola entitas *merchant* sebagai tujuan distribusi EDC dari SP.
c. Mengelola entitas SP sebagai tujuan distribusi EDC dari vendor *arrival* atau vendor *service management* dan untuk membuat akun *user* SP *checker maker*.

## 1.4 Manfaat

Manfaat dari proyek pengembangan inventori EDC berbasis web yang dikerjakan penulis adalah sebagai berikut:

a. Meningkatkan efisiensi alur proses inventori EDC karena vendor *arrival* dapat segera memperbaiki EDC yang mengalami kerusakan dan mengirimkan ke klien perusahaan untuk diproses kembali.

b. Memiliki riwayat kerusakan EDC yang jelas dan akurat sehingga dapat mencegah potensi kerusakan kembali dan mengontrol kualitasnya di masa mendatang.
c. Memiliki data entitas *merchant* dan SP yang terstruktur dan akurat sehingga dapat mengidentifikasi entitas dengan tepat saat dibutuhkan atau digunakan.

## 1.5 Sistematika Penulisan

Sistematika penulisan membahas inti dari setiap bab yang terdapat dalam laporan akhir ini. Adapun susunan sistematika penulisan yang digunakan dalam laporan akhir adalah sebagai berikut:

a. BAB I: Pendahuluan

   Bab ini membahas mengenai gambaran umum perusahaan sebagai tempat magang, rangkuman proyek-proyek yang dikerjakan selama magang, dan alasan penulis dalam memilih suatu proyek sebagai topik laporan akhir. Selain itu, membahas mengenai ruang lingkup magang, tujuan dan manfaat dari topik laporan akhir yang dikerjakan penulis, serta sistematika penulisan.
b. BAB II: Landasan Teori dan Tinjauan Pustaka

   Bab ini membahas mengenai teori-teori yang mendukung topik laporan akhir, berupa *framework* Spring Boot, *database* PostgreSQL, *template* Thymeleaf, *library* jQuery, AJAX, dan metode pengembangan *waterfall*. Selain itu, membahas persamaan dan perbedaan antara topik laporan akhir dan hasil penelitian-penelitian sebelumnya yang serupa atau mendekati.
c. BAB III: Pelaksanaan Magang

   Bab ini membahas mengenai pelaksanaan magang pada proyek yang menjadi topik laporan akhir. Mulai dari metode manajemen proyek yang digunakan, detail dari proses dan hasil pelaksanaan proyek, serta status proyek.
d. BAB IV: Refleksi Pelaksanaan Magang

   Bab ini membahas mengenai relevansi antara teori-teori yang telah didapatkan pada bab ke-2 dan pelaksanaan magang pada bab ke-3. Selain itu, membahas mengenai pembelajaran magang meliputi manfaat, kendala, hambatan, dan tantangan yang didapatkan selama pelaksanaan magang.
e. BAB V: Penutup

Bab ini membahas mengenai kesimpulan yang didapatkan dari pembahasan yang telah dilakukan pada bab-bab sebelumnya dan saran yang dapat diberikan untuk perbaikan ke depannya.

# BAB II
# LANDASAN TEORI DAN TINJAUAN PUSTAKA

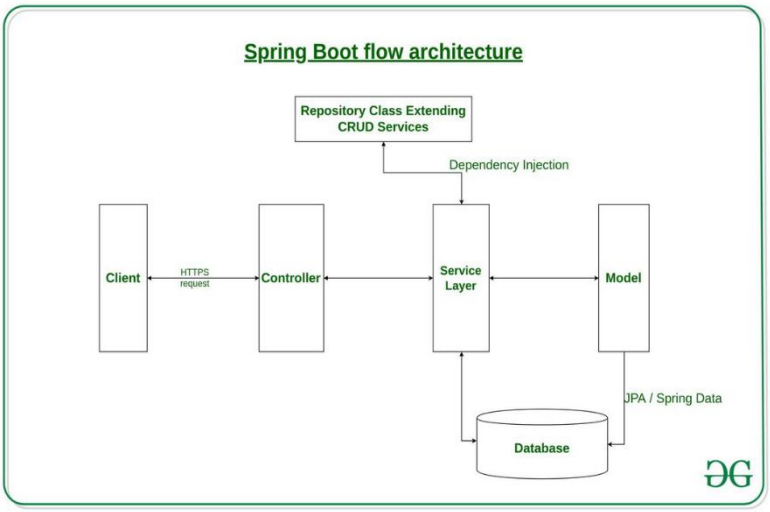
## 2.1 *Framework* Spring Boot

Dalam pengembangan suatu aplikasi, biasanya metode dengan memanfaatkan bantuan berupa *framework* lebih sering digunakan daripada metode secara *native* (pengodean langsung tanpa menggunakan bantuan apapun) karena akan lebih mempermudah pengembangannya. *Framework* merupakan kerangka kerja yang menyediakan sekumpulan kode berbentuk *class* dan *function* yang bertujuan membantu *programmer* menyelesaikan pengembangan dengan lebih mudah dan lebih efisien (Yudhanto & Prasetyo, 2019). Tujuan tersebut juga didukung dengan kemudahan yang dirasakan *programmer* berupa tidak perlu bersusah payah untuk membuat *class* dan *function* tersebut dari awal, hanya perlu memanggilnya ketika dibutuhkan dan dapat digunakan berulang kali (Yudhanto & Prasetyo, 2019).

Terdapat berbagai jenis *framework* yang tersedia di dunia pengembangan aplikasi, salah satunya adalah Spring Boot yang digunakan pada sisi *back-end* aplikasi berbasis web. Spring Boot merupakan bagian dari *framework* Spring yang mempermudah proses pengembangan aplikasi Spring dan sekarang wajib digunakan dalam pengembangan aplikasi Spring (Khaqiqi & Harani, 2023). Spring sendiri merupakan *framework* yang sangat terkenal dalam bahasa pemrograman Java karena dianggap mampu menyaingi Java *Enterprise* dalam hal kemudahan (Khaqiqi & Harani, 2023). Spring Boot memiliki beragam keunggulan yang didapatkan dari Spring, seperti Spring MVC yang memperkuat basis *model view controller* (MVC), Spring Rest yang menyediakan fitur RESTful *web service*, Spring Data yang menyediakan koneksi dengan *database*, *dependency injection* (DI) yang memudahkan pengaturan *dependency* dalam aplikasi, dan *aspect object programming* (AOP) yang memungkinkan pemisahan *core business* sistem dengan fungsi lain, seperti *logging* dan masalah keamanan (Jayanto, 2017). Selain itu, Spring Boot memiliki keunggulan tersendiri, yaitu *default server* Tomcat yang memudahkan *programmer* menjalankan aplikasi secara langsung dan *default* Maven yang menyediakan manajemen struktur aplikasi (hierarki folder dan *file*) serta struktur dependensi (pom.xml) secara otomatis (Jayanto, 2017).

Langkah pertama untuk dapat menggunakan Spring Boot adalah melakukan inisialisasi pembuatan proyek Spring Boot dengan cepat menggunakan Spring Initializr yang membantu menentukan manajemen struktur aplikasi (Maven atau Gradle), bahasa pemrograman (Java, Kotlin, atau Groovy), versi Spring Boot dan Java, penamaan proyek, dan pilihan dependensi

yang dibutuhkan (Spring Web, Spring Data JPA, Lombok, Thymeleaf, dan lainnya). Setelah inisialisasi berhasil dilakukan, secara otomatis dependensi disimpan pada file pom.xml dan terdapat folder src (*source*) untuk menyimpan berbagai *source code*. Kemudian, membuat *main class* dengan anotasi @SpringBootApplication sebagai kelas yang pertama kali dijalankan dan akan menjalankan semua kode yang terlibat dalam sistem proyek. Langkah selanjutnya adalah mengikuti alur arsitektur Spring Boot yang dapat dilihat pada Gambar 2.1.




Gambar 2.1 Alur Arsitektur Spring Boot
(Sumber: GeeksforGeeks, 2022)

Berdasarkan Gambar 2.1, dapat disimpulkan terdapat empat lapisan *back-end* pada arsitektur bagian *server side*. Pertama, lapisan *model* yang digunakan untuk mendefinisikan struktur data tabel di *database* dengan memberikan anotasi @Entity dan @Table supaya dapat terhubung dengan baik. Lapisan tersebut biasanya disimpan pada folder *model* atau *entity*. Kedua, lapisan *repository* yang digunakan untuk mengakses dan memanipulasi data tabel di *database* yang diwakili melalui lapisan *model*. Lapisan tersebut biasanya disimpan pada folder *repository* dan menggunakan Spring Data JPA *repository* untuk memudahkan operasi fungsi dasar pada data tanpa perlu menulis kueri SQL secara manual, seperti *cread, read, update,* dan *delete* (CRUD). Ketiga, lapisan *service* yang digunakan untuk membantu mengatur logika bisnis pada berbagai fungsi sistem dengan menggunakan bantuan lapisan *repository* yang diintegrasikan melalui *constructor injection*. Lapisan tersebut biasanya disimpan pada folder *service* dan memiliki anotasi @Service sebagai penanda menjadi bagian dari lapisan *service*. Keempat, lapisan *controller* yang digunakan untuk membuat *endpoint* dari fungsi pada lapisan *service* yang diintegrasikan melalui *constructor injection*. Lapisan tersebut biasanya disimpan

pada folder *controller* dan memiliki anotasi @RestController atau @Controller sebagai penanda menjadi bagian dari lapisan *controller*. Perbedaan kedua anotasi tersebut dapat dilihat dari bentuk *response* yang dikembalikan. Pada @RestController biasanya berbentuk data langsung dalam format JSON, sedangkan pada @Controller biasanya berbentuk tampilan HTML. Selain itu, *endpoint* pada lapisan *controller* berperan penting dalam menangani HTTP *request* dari *client side* dan memberikan HTTP *response* ke *client side*. Berikut contoh implementasi untuk lapisan *controller* yang dapat dilihat pada Gambar 2.2.

```
@RestController
@RequestMapping("/users")
@RequiredArgsConstructor
public class UserController {

    private final UserService userService;
    private final UserMapper mapper;

    @GetMapping
    public ResponseEntity<List<AppUserResponse>> getUsers() {
        List<AppUser> users = userService.find();
        var resp = users.stream().map(mapper::toResponse).toList();
        return ResponseEntity.ok(resp);
    }
```

Gambar 2.2 Contoh Implementasi Penggunaan Spring Boot
(Sumber: Albuquerque, 2023)

Berdasarkan Gambar 2.2, terdapat *class user controller* yang berada di lapisan *controller* dengan adanya anotasi @RestController. *Class* tersebut membuat *endpoint get users* yang diambil dari fungsi *find* pada *class user service* di lapisan *service*. Pengambilan fungsi dapat dilakukan karena anotasi @RequiredArgsConstructor dari dependensi Lombok secara otomatis akan membuat konstruktor dengan parameter yang mencakup semua atribut *final*, salah satunya adalah *user service*, sehingga Spring Boot dapat mengintegrasikan *class user service* dari lapisan *service* ke *class user controller* di lapisan *controller*. *Endpoint get users* yang dibuat menggunakan metode HTTP GET *request* dengan *path* berupa "/users" yang didapatkan dari anotasi @RequestMapping. *Response* yang dikembalikan akan berbentuk status 200 (ok) dan semua *user* dalam format JSON.

## 2.2 *Database* PostgreSQL

Penggunaan *database* telah menjadi keharusan dalam pengembangan suatu aplikasi. Hal tersebut dikarenakan *database* merupakan kumpulan informasi atau data terstruktur yang disimpan secara elektronik dalam sistem komputer dengan dibantu oleh *database management system* (DBMS) dan memiliki beberapa tipe, antara lain *relational*, *object oriented*, *noSQL*,

*distributed*, *graph*, *cloud*, dan *document* JSON (Oracle, 2024). DBMS merupakan antarmuka penghubung antara *database* dan pengguna sehingga memungkinkan pengguna untuk dapat mengakses, memodifikasi, dan mengelola informasi (Oracle, 2024).

Salah satu *database* yang sering digunakan dalam dunia pengembangan suatu aplikasi adalah PostgreSQL. PostgreSQL merupakan *object relational database management system* (ORDBMS) yang *open source* dan *free* dengan menggunakan dan mengembangkan bahasa SQL yang dikombinasikan dengan banyak fitur (PostgreSQL, 2024). ORDBMS merupakan penggabungan antara *relational database* dan *object oriented database* sehingga terdapat penambahan khusus berupa informasi yang sifatnya berorientasikan objek di dalam *relational database* dan informasinya disusun sebagai kumpulan tabel dengan kolom dan baris (Sofi et al., 2015). PostgreSQL dapat dijalankan di berbagai sistem operasi, seperti Windows, Linux, macOS, dan Solaris (PostgreSQL, 2024) serta didukung oleh banyak bahasa pemrograman, seperti C++, Java, PHP, dan Python (Munawaroh, 2005). PostgreSQL juga memiliki beberapa keunggulan lain, yaitu lebih fleksibilitas dalam melakukan *query* data berupa *select* yang sederhana hingga yang sangat kompleks, keamanan data yang lebih terlindungi karena admin data dapat mengatur akses *database* pengguna, dan dapat mengoptimalkan kecepatan kinerja aplikasi melalui pembuatan *function* dan *stored procedure* oleh pengguna karena prosesnya dilakukan di *server side* (Munawaroh, 2005).

## 2.3 *Template* Thymeleaf

Thymeleaf merupakan *template engine* Java pada *server side* yang sudah terintegrasi dengan *framework* Spring untuk mengembangkan suatu aplikasi berbasis web (Dagha & Susetyo, 2021). Thyemelaf juga dapat mendukung proses dengan teknologi HTML, XML, JavaScript, hingga CSS, dan memiliki konsep dasar untuk menerapkan pemisahan antara lapisan *model* dan lapisan *view* pada aplikasi Spring berbasis MVC sehingga dapat membantu menghasilkan tampilan halaman web yang lebih dinamis (Eriksson, 2022). Salah satu caranya dengan menyisipkan logika dan data dari *server side* ke dalam *file* HTML di *client side*, lebih tepatnya pada elemen HTML tertentu, dengan menggunakan berbagai atribut dan ekspresi variabel dari Thymeleaf yang disesuaikan dengan kebutuhan (Eriksson, 2022). Berikut contoh implementasinya yang dapat dilihat pada Gambar 2.3.

```
<p>Today is: <span th:text="${today}">13 february 2011</span>.</p>
```

Gambar 2.3 Contoh Implementasi Penggunaan Thymeleaf
(Sumber: Thymeleaf, 2023)

Berdasarkan Gambar 2.3, terdapat elemen *paragraph* pada suatu *file* HTML yang menggunakan *template* Thymeleaf. Di dalamnya terdapat elemen *span* yang memiliki atribut Thymeleaf berupa th:text dan ekspresi variabel Thymeleaf berupa ${today}. Atribut tersebut digunakan untuk menentukan teks yang akan ditampilkan sehingga akan menggantikan teks asli yang tertulis di dalam elemen *span*. Teks yang akan ditampilkan diambil dari ekspresi variabel tersebut, dimana ${today} adalah *key* yang memiliki suatu *value* yang telah diatur di *server side* (Thymeleaf, 2023).

Dari berbagai penjelasan sebelumnya mengenai Thymeleaf, dapat disimpulkan bahwa Thymeleaf bekerja di *server side* tetapi dapat memengaruhi *client side* dengan membantu menghasilkan tampilan web yang lebih dinamis. Selain itu, keuntungan penggunaan Thymeleaf yang lain adalah memiliki fleksibilitas dalam perubahan di lapisan *modal* atau *view* secara independen dan penggunaan kode dari *server side* berupa ekspresi variabel Thymeleaf dapat digunakan berulang kali pada berbagai elemen di berbagai *file* HTML yang menggunakan *template* Thymeleaf.

## 2.4 *Library* jQuery

jQuery merupakan *library* JavaScript yang memberikan kemudahan dan kecepatan bagi *programmer* dalam mengembangkan aplikasi web yang lebih interaktif karena menyediakan kumpulan kode berbentuk *function* yang disusun menggunakan JavaScript untuk kemudian hanya perlu dipanggil ketika dibutuhkan dan dapat digunakan berulang kali (Ganiardi et al., 2016). Selain itu, jQuery mudah dipahami oleh *programmer* karena *function* yang dihasilkan lebih sederhana dan singkat (Ganiardi et al., 2016). jQuery juga bersesuaian dengan semua versi CSS dan dengan berbagai web *browser*, seperti Chrome, Mozilla Firefox, Opera, dan Internet Explorer (Ganiardi et al., 2016).

Dari berbagai *library* yang dimiliki JavaScript, jQuery menjadi salah satu pilihan yang telah digunakan oleh perusahaan-perusahaan besar karena beberapa kemampuan unggulnya. Pertama, jQuery memiliki *selector* yang efisien untuk mempermudah mengakses suatu elemen tertentu pada *file* HTML dan dapat memanipulasinya sesuai kebutuhan dengan *function* yang

disediakan jQuery (Ganiardi et al., 2016). Kedua, dapat mengubah tampilan halaman web yang menggunakan CSS tanpa perlu khawatir akan berubah ketika digunakan pada web *browser* lain (Ganiardi et al., 2016). Ketiga, mempermudah penggunaan teknik AJAX karena kodenya telah disederhanakan (Ganiardi et al., 2016). Berikut contoh implementasi untuk kemampuan pertama yang dapat dilihat pada Gambar 2.4.

```
$(document).ready(function() {
    $('p').html('<b>JavaScript</b><br/> dapat mengubah isi simpul DOM');
});
```

Gambar 2.4 Contoh Implementasi Penggunaan jQuery
(Sumber: Sianipar, 2016)

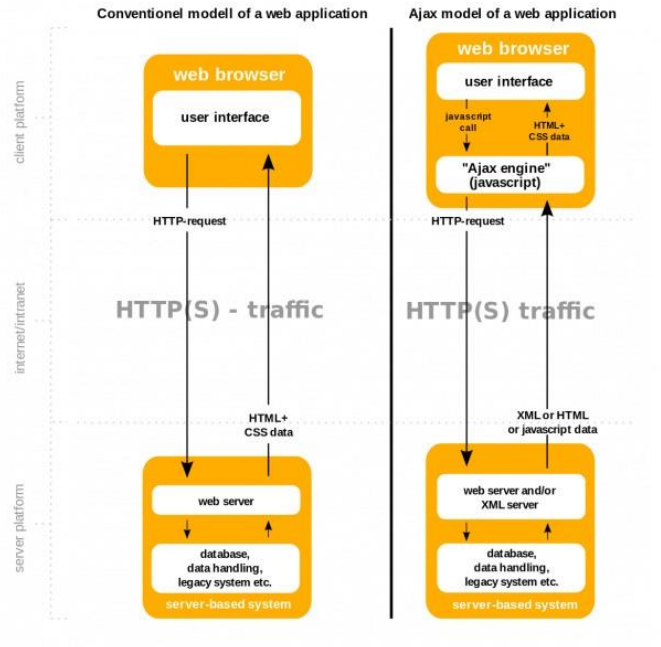
Berdasarkan Gambar 2.4, terdapat pengambilan elemen *paragraph* pada suatu HTML dengan *selector* jQuery berupa $('p'). *Selector* tersebut kemudian memanggil *function* jQuery berupa html untuk memanipulasi atau mengubah konten dari elemen. Selain itu, terdapat $(document).ready() untuk memastikan bahwa semua kode di dalamnya akan dijalankan secara langsung saat semua dokumen HTML selesai dimuat pertama kali. Dengan begitu, nantinya isi pada elemen *paragraph* akan berubah menjadi teks "JavaScript" yang ditebalkan, diikuti oleh pergantian baris, dan dilanjutkan dengan teks "dapat mengubah isi simpul DOM".

## 2.5 AJAX

AJAX (*asynchronous javascript and xml*) merupakan suatu teknik dalam pengembangan aplikasi berbasis web yang berperan sebagai penghubung antara *client side* dan *server side* untuk mengirim HTTP *request* dan menerima HTTP *response* yang dilakukan secara asinkron. Maksud dari asinkron tersebut adalah komunikasi antara *client* dan *server* dapat bekerja di belakang layar sehingga memungkinkan pengguna untuk tetap berinteraksi dengan halaman web tanpa adanya *reload* pada keseluruhan halaman web (Tahir, 2018).

Hadirnya AJAX menjadi solusi untuk permasalahan cara kerja web tradisional yang memiliki waktu tunggu *response* oleh *server* setelah dikirimkan *request* dari *client* dan waktu tunggu *reload* seluruh halaman web ketika *response* telah dikirim dari *server* dan diterima oleh *client* (Abdulloh, 2017). Di sisi lain, cara kerja web dengan AJAX adalah setiap kali terdapat *request*, *client* akan memanggil AJAX untuk membuat objek XMLHTTPRequest yang akan digunakan JavaScript untuk dikirimkan ke *server* (Abdulloh, 2017). *Server* akan mengirimkan *response* ke *client* dalam bentuk data XML ketika data selesai diproses oleh *server* (Abdulloh, 2017). Kemudian, AJAX akan mengubah data XML tersebut menjadi data HTML dan CSS

untuk memodifikasi bagian tertentu pada halaman web sehingga hanya bagian tersebut yang akan di-*reload* (Abdulloh, 2017). Ilustrasi perbandingan cara kerja antara web tradisional dan cara kerja web dengan AJAX dapat dilihat pada Gambar 2.5.



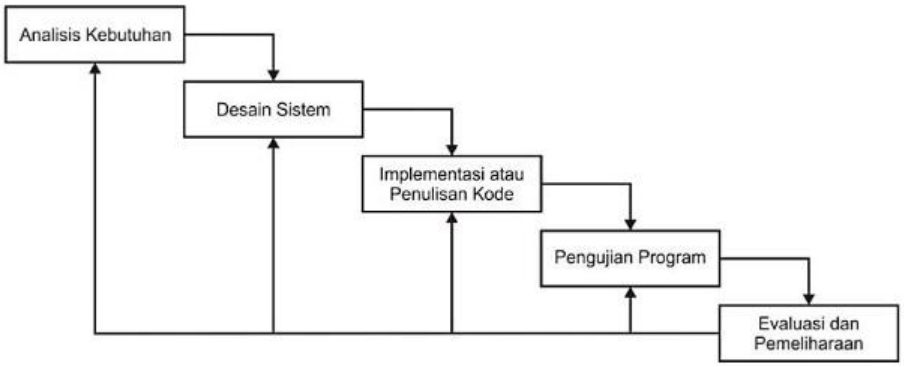

Gambar 2.5 Perbandingan Cara Kerja antara Web Tradisional dan Web dengan AJAX (Sumber: C, 2023)

Maka dari itu, penggunaan web dengan AJAX memiliki beberapa keuntungan tersendiri. Pertama, tentu saja mempercepat kinerja aplikasi karena halaman web hanya akan me-*reload* bagian tertentu yang diperlukan dan pengguna tetap dapat berinteraksi dengan halaman web (Abdulloh, 2017). Kedua, mengurangi beban pada *server side* karena sebagian pemrosesan dilakukan pada *client side* (Abdulloh, 2017). Ketiga, mengoptimalkan penggunaan *bandwidth* karena *client* hanya mengambil data yang dibutuhkan untuk memodifikasi bagian tertentu pada halaman web (Abdulloh, 2017). Keempat, tidak perlu khawatir web dengan AJAX tidak dapat berjalan pada web *browser* lain karena hampir semua web *browser* telah mendukung AJAX (Abdulloh, 2017).

## 2.6 Metode *Waterfall*

Metode *waterfall* merupakan salah satu model *system development life cycle* (SDLC) yang sering digunakan. Dalam pengembangan sistem informasi, SDLC merupakan metode umum yang sering digunakan. Metode *waterfall* sendiri menggambarkan pendekatan secara

terstruktur, berurutan, dan bersifat linear dari tahapan awal hingga akhir (Wahid, 2020). Setiap tahapan dapat dilewati dengan syarat bahwa tahapan tersebut sudah selesai dilakukan, tetapi setiap tahapan yang telah dilewati tidak dapat diulang (Wahid, 2020). Dengan begitu, metode ini cocok digunakan untuk kebutuhan yang sudah dipahami secara jelas dan memiliki peluang perubahan yang minimum selama pengembangannya (Solehudin et al., 2023). Terdapat lima tahapan dalam metode *waterfall* yang dijelaskan dan dapat dilihat pada Gambar 2.6.




Gambar 2.6 Tahapan Metode *Waterfall*
(Sumber: Hermansyah et al., 2023)

a. Analisis Kebutuhan

   Pada tahap ini, hal yang dilakukan adalah mengetahui dan mengumpulkan semua informasi terkait kebutuhan sistem yang diinginkan oleh *user* atau klien, seperti cakupan, batasan, dan fitur (Vicky & Syaripudin, 2022). Informasi tersebut biasanya didapatkan dari diskusi, wawancara, observasi, atau dokumen kebutuhan yang kemudian dianalisis sehingga mendapatkan semua data sistem yang lengkap dan sesuai kebutuhan *user* atau klien (Vicky & Syaripudin, 2022).

b. Desain Sistem

   Pada tahap ini, hal yang dilakukan adalah merancang desain sistem yang meliputi *use case diagram*, *activity diagram*, *entity relationship diagram*, hingga *prototype user interface* (Hermansyah et al., 2023). Semua perancangan desain sistem tersebut akan meningkatkan pemahaman kebutuhan sistem yang lebih jelas dan detail sehingga akan memberikan kemudahan dan kecepatan untuk implementasi atau penulisan kode program (Hermansyah et al., 2023).

c. Implementasi atau Penulisan Kode

   Pada tahap ini, hal yang dilakukan adalah implementasi atau menulis kode program supaya pengembangan sistem dapat segera terealisasikan. Implementasinya akan

mengintegrasikan berbagai fitur, dimana setiap fitur memiliki detail-detail tersendiri yang digambarkan dengan fungsi-fungsi (Vicky & Syaripudin, 2022). Hal tersebut disesuaikan dengan kebutuhan yang telah dijelaskan pada dua tahapan sebelumnya.

d. Pengujian Program

Pada tahap ini, hal yang dilakukan adalah menguji semua implementasi sistem yang telah direalisasikan. Pengujian tersebut untuk mengamati dan memastikan bahwa berbagai fitur dalam sistem dapat berjalan dengan baik sesuai kebutuhan dan harapan (Hermansyah et al., 2023).

e. Evaluasi dan Pemeliharaan

Pada tahap ini, hal yang dilakukan adalah memelihara sistem yang sudah dijalankan secara langsung oleh *user* atau klien supaya tetap berjalan sebagaimana mestinya (Hermansyah et al., 2023). Selain itu, memperbaiki sistem berdasarkan hasil evaluasi pemeliharaan tersebut jika ditemukan kesalahan atau ketidaksesuaian supaya sistem dapat kembali berjalan sebagaimana mestinya.

## 2.7 Tinjauan Pustaka

Pada subbab ini, penulis membahas mengenai beberapa hasil penelitian sebelumnya yang serupa atau mendekati dengan inti proyek yang menjadi topik laporan akhir ini supaya dapat dilihat persamaan dan perbedaannya. Berikut hasil penelitian tersebut yang dapat dilihat pada Tabel 2.1.

Tabel 2.1 Tinjauan Pustaka Penelitian Terdahulu

| No | Judul | Tujuan | Tekno-logi | Metode | Hasil |
|---|---|---|---|---|---|
| 1. | Pembangunan Sistem Manajemen Stok pada Bunyu Maju Bersama Berbasis *Website* | Mengelola manajemen stok, transaksi, dan membuat laporan dengan baik dan efisien | Java, Spring Boot, MySQL, VueJs | Waterfall | Sistem manajemen stok toko berbasis *website* berhasil dibangun. Dari hasil pengujian didapatkan bahwa pengguna merasa puas dan setuju bahwa tampilannya mudah digunakan dan dapat |

| No | Judul | Tujuan | Tekno -logi | Metode | Hasil |
|---|---|---|---|---|---|
| | | | | | membantu penjualan (Buak, 2023) |
| 2. | *Stock Management System for Small and Medium Sized Companies* | Membantu manajemen stok pada perusahaan kecil dan menengah dengan sistem *back-end* yang *open source* | Java, Spring Boot, MariaDB | Scrum | Sistem *back-end* yang *open source* berhasil dibangun. Setiap *endpoint* API telah diuji berkali-kali dengan *unit test* dan Postman (Brühwiler, 2023) |
| 3. | Pengembangan Sistem *Point of Sale* Berbasis Web (Studi Kasus: Bos Duren Malang) | Memperbaiki masalah *human error* dalam pencatatan transaksi | Java, Spring Boot, Type Script, ReactJs | Waterfall | Sistem POS berbasis web berhasil dikembangkan dan telah diuji dengan tiga macam pengujian, yaitu validasi (*black-box*), integrasi (*white-box*), dan unit (*white-box*). Ketiganya mendapatkan hasil 100% valid (Dirosa & Kurniawan, 2021) |
| 4. | Pembuatan REST API Manajemen Data Karyawan Berbasis *Website* | Mengelola data karyawan pada suatu perusahaan berskala besar atau kecil dengan mudah | Java, Spring Boot, Postgre SQL, MyBatis | Analisis kebutuhan, rancangan sistem, penerapan desain, | Sistem manajemen data karyawan berbasis *website* berhasil dibuat. Sistem juga telah diuji dengan *black box* dan mendapatkan hasil yang |

| No | Judul | Tujuan | Tekno-logi | Metode | Hasil |
|---|---|---|---|---|---|
| | Menggunakan Spring Boot | | | pengujian sistem | sukses sesuai harapan (Adji & Mailoa, 2024) |
| 5. | *Website* Rumah Sakit Pelayanan Kesehatan Umum Muhammadiyah Sragen | Memudahkan masyarakat untuk mengetahui pelayanan rumah sakit | Java, Spring Boot, ReactJS | Waterfall | Sistem pelayanan rumah sakit berbasis *website* berhasil dibuat. Sistem juga telah diuji dan mendapatkan hasil yang valid sesuai harapan (Ma'ruf et al., 2023) |
| 6. | Pengembangan Sistem *Inventory* Barang Perusahaan Dagang Berbasis *Website* (Studi Kasus: CV. Agung Nugraha) | Menyesuaikan data antara informasi persediaan barang dan stok fisik pada gudang. Juga memudahkan dalam kelola dan pantau barang | PHP, Code Igniter, MySQL, Java Script | Waterfall | Sistem *inventory* barang berbasis *website* berhasil dikembangkan. Sistem juga telah diuji dengan *black-box* dan mendapatkan hasil yang valid sesuai harapan (Azizah & Nurgiyatna, 2021) |
| 7. | Sistem Informasi *Inventory* pada PT Djaya Buah Bersinar Denpasar Berbasis Web | Mengatasi berbagai masalah dalam pengelolaan data persediaan barang yang sebelumnya | PHP, MySQL, HTML | Waterfall | Sistem *inventory* buah berbasis web berhasil dibangun. Sistem juga telah diuji dengan *black-box* dan mendapatkan hasil yang valid sesuai harapan (Desmayani et al., 2022) |

| No | Judul | Tujuan | Tekno -logi | Metode | Hasil |
|---|---|---|---|---|---|
| | | masih direkam manual | | | |
| 8. | Rancangan Sistem Informasi Persediaan Barang Berbasis Web pada Perusahaan Perdagangan | Mengelola persediaan barang yang dapat meng-integrasikan antara gudang persediaan dan toko sehingga informasi persediaan selalu terkini | PHP, Code Igniter, MySQLi | Waterfall | Sistem persediaan barang berbasis web berhasil dibangun. Sistem juga telah diuji dengan *black-box* dan mendapatkan hasil yang valid sesuai harapan (Maulana, 2022) |
| 9. | Perancangan Aplikasi Manajemen Persediaan Gudang Berbasis *Website* pada UMKM Batik Sinuwun dengan Agile Scrum *Development Method* | Mengelola gudang persediaan untuk pencatatan bahan (baku & jadi) yang masuk dan keluar dengan akurat dan terkini | PHP, Code Igniter, MySQL | Scrum | Sistem persediaan batik berbasis *website* berhasil dibangun. Sistem juga telah diuji dengan *black-box* dan mendapatkan hasil yang valid sesuai harapan. Selain itu, sistem telah lolos ISO:9126 (Majdina et al., 2020) |
| 10. | Model *Waterfall* untuk Rancang Bangun Sistem Informasi Pengadaan | Menggantikan metode manual yang memakan waktu pengadaan | Tidak diketahui | Waterfall | Sistem pengadaan mesin EDC berbasis web berhasil dibangun sesuai tujuan. Akan tetapi, pengujiannya |

| No | Judul | Tujuan | Tekno -logi | Metode | Hasil |
|---|---|---|---|---|---|
| | Mesin EDC pada E-Channel Operations Perbankan | EDC dari suatu gudang ke perbankan dengan lebih efisien dan efektif | | | tidak ditampilkan (Amrin & Aldiansyah, 2021) |

Dari semua penelitian yang terdapat pada Tabel 2.1, fokusnya dapat dibagi menjadi tiga bagian. Pertama, pengembangan sistem inventori dengan teknologi bahasa pemrograman Java dan *framework* Spring Boot pada penelitian nomor 1 dan 2. Kedua, berbagai pengembangan sistem berbasis web dengan teknologi bahasa pemrograman Java dan *framework* Spring Boot pada penelitian nomor 3 – 5. Ketiga, pengembangan sistem inventori berbasis web dengan teknologi selain bahasa pemrograman Java dan *framework* Spring Boot pada penelitian nomor 6 – 10. Ketiga fokus tersebut memiliki persamaan dengan proyek pada topik laporan akhir ini, yaitu:

a. Pada fokus pertama, berhasil mengembangkan sistem inventori dengan teknologi bahasa pemrograman Java dan *framework* Spring Boot untuk membantu mengelola persediaan pada toko atau perusahaan.
b. Pada fokus kedua, berhasil mengembangkan suatu sistem berbasis web dengan teknologi bahasa pemrograman Java dan *framework* Spring Boot.
c. Pada fokus ketiga, berhasil mengembangkan sistem inventori berbasis web untuk membantu mengelola persediaan barang pada toko atau perusahaan dengan memiliki informasi yang akurat dan terkini.
d. Pada fokus ketiga juga, tepatnya di penelitian nomor 10, barang yang dikelola dalam sistem inventori (pengadaan) berbasis web adalah EDC.
e. Terdapat tujuh penelitian yang menggunakan metode pengembangan *waterfall* dan semua penelitian tersebut berhasil dikembangkan.

Terdapat juga beberapa perbedaan antara ketiga fokus tersebut dan proyek pada topik laporan akhir ini, yaitu:

a. Pada fokus pertama, tepatnya di penelitian nomor 2, hasil akhirnya berupa *back-end* yang *open source* sehingga dapat digunakan secara umum oleh berbagai perusahaan

kecil dan menengah dengan disesuaikan pada kebutuhan masing-masing. Sementara itu, hasil akhir dari proyek ini berupa web yang menggabungkan *back-end* dan *front-end* serta hanya dapat digunakan oleh *user* entitas yang terlibat di dalamnya.

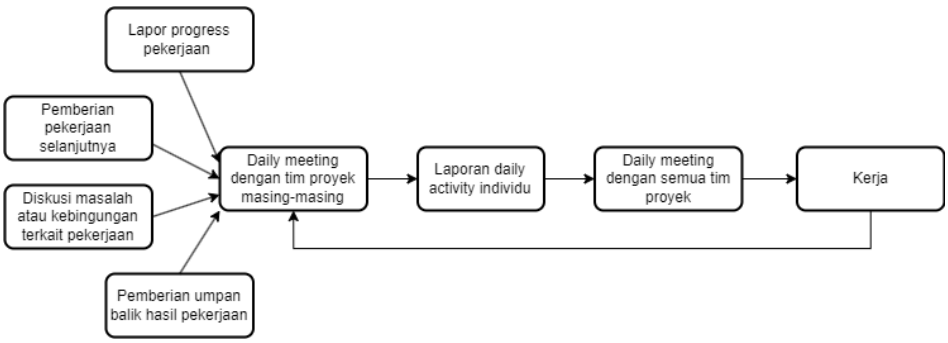
b. Pada fokus kedua, sistem berbasis web yang dikembangkan dengan teknologi bahasa pemrograman Java dan *framework* Spring Boot bukan tentang inventori barang, melainkan tentang *point of sale*, data karyawan, dan pelayanan rumah sakit.
c. Pada fokus ketiga, sistem inventori berbasis web yang dikembangkan bukan dengan teknologi bahasa pemrograman Java dan *framework* Spring Boot, melainkan dengan teknologi bahasa pemrograman PHP dan *framework* CodeIgniter secara umum.
d. Pada fokus ketiga juga, tepatnya di penelitian nomor 10, sistem inventori yang dimaksud hanya berfokus pada pengadaan EDC dari suatu gudang ke perbankan sesuai dengan permintaan pengadaan. Sementara itu, sistem inventori pada proyek ini fokusnya meliputi pengadaan, pemrosesan, persediaan, hingga pendistribusian atau pengembalian. Selain itu, penelitian tersebut tidak menyebutkan teknologi yang digunakan untuk membangun sistem.
e. Semua penelitian tersebut, kecuali penelitian nomor 10, melakukan pengujian pada hasil pengembangan sistem. Enam diantaranya menggunakan metode pengujian *black-box* yang telah mendapatkan hasil valid atau sukses sesuai harapan. Hal tersebut berbeda dengan pengujian pada hasil proyek ini yang belum dilakukan oleh *quality engineer* dengan menerapkan suatu metode, baik *black-box* atau lainnya. Pengujian masih sebatas lingkup *developer* saja dengan fokusnya untuk menguji dan memastikan bahwa setiap fungsi dapat berjalan dengan baik sesuai kebutuhan.

# BAB III
# PELAKSANAAN MAGANG

## 3.1 Manajemen Proyek

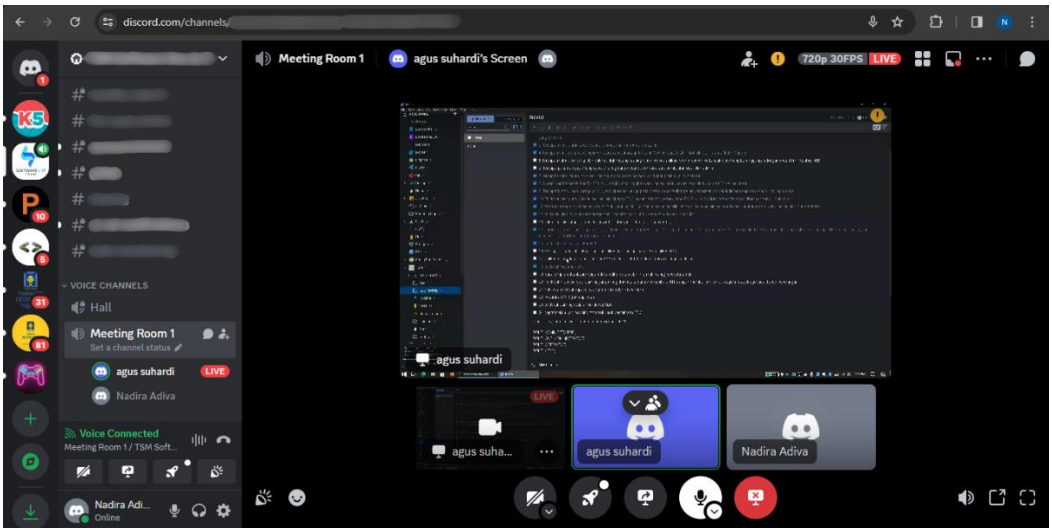
Penulis menjalani kegiatan magang sebagai *full-stack web developer* selama 6 bulan secara WFH pada setiap hari Senin – Jumat dengan waktu kerja pada pukul 09.00 – 18.00 WIB. Selama 6 bulan tersebut, penulis ikut mengerjakan proyek pengembangan inventori EDC berbasis web bersama satu rekan kerja. Proyek tersebut merupakan permintaan dari salah satu klien perusahaan. Permintaan tersebut menyertakan dokumen dengan informasi yang lengkap untuk keberlangsungan pengerjaan proyek kepada *developer*, diantaranya cakupan sistem, fitur-fitur yang dibutuhkan, alur kerja sistem, dan rancangan *database*. Berbekal dokumen tersebut, penulis dapat ikut lanjut mengerjakan proyek yang sebelumnya telah dikerjakan selama 2 minggu oleh rekan kerja. Selain itu, penulis dan rekan kerja memanfaatkan *platform* GitLab untuk mendukung pengerjaan secara kolaboratif dan efisien karena dapat menyimpan, mengelola, menggabungkan, dan melacak perubahan kode-kode pemrograman. Rekan kerja tersebut bernama Bapak Agus Suhardi, yang sekaligus menjadi *supervisor* bagi penulis selama pelaksanaan magang. Bapak Agus Suhardi juga bekerja secara WFH dan berperan sebagai *full-stack web developer*.

Selanjutnya, khusus proyek pengembangan inventori EDC berbasis web ini memiliki metode manajemen yang berbeda dengan proyek lain. Proyek lain menggunakan metode manajemen yang sudah tersedia dan kini lazim digunakan, seperti Scrum, beserta alat yang mendukung manajemennya, seperti Trello. Sementara itu, proyek ini menggunakan metode manajemen tersendiri karena kurangnya anggota yang terlibat pada proyek ini, dimana hanya dikerjakan oleh dua anggota, yaitu penulis dan rekan kerja. Alur pada metode manajemen proyek ini dapat dilihat pada Gambar 3.1.




Gambar 3.1 Alur Metode Manajemen Proyek

Berdasarkan Gambar 3.1, terlihat bahwa penulis dan rekan kerja biasanya melakukan *daily meeting* setiap pagi dan umumnya meliputi satu atau beberapa pembahasan dari empat pembahasan, yaitu melaporkan *progress* pekerjaan masing-masing, *supervisor* memberikan pekerjaan selanjutnya, diskusi masalah atau kebingungan terkait pekerjaan, dan *supervisor* memberikan umpan balik terhadap hasil pekerjaan sebelumnya. Umpan balik tersebut dapat berupa ulasan, *review* untuk perbaikan, atau masukan untuk ke depannya. *Daily meeting* dilakukan secara *online* atau daring melalui Discord, Google Meet, atau Whatsapp karena penulis dan *supervisor* bekerja secara WFH. Terkadang penulis dan rekan kerja melakukan *meeting* tambahan ketika masih ada suatu hal yang harus dibahas di luar *daily meeting*. Berikut salah satu bukti *daily meeting* melalui Discord yang dapat dilihat pada Gambar 3.2.




Gambar 3.2 Bukti *Daily Meeting* Proyek

Setelah melakukan *daily meeting* dengan tim proyek masing-masing, setiap anggota pada bidang *software application development* dan *information technology* melaporkan *daily activity*. Pelaporan melalui Google Form dengan menuliskan semua pekerjaan yang telah dilakukan dalam 24 jam terakhir dan semua pekerjaan yang akan dilakukan dalam 24 jam berikutnya. Kemudian, pada pukul 09.45 WIB biasanya dilakukan *daily meeting* atau *huddle* dengan semua tim proyek pada bidang *software application development* dan *information technology* melalui Discord untuk menyampaikan laporan *daily activity* setiap anggota yang sebelumnya sudah dilaporkan melalui Google Form. Akan tetapi, *huddle* tersebut sudah ditiadakan mulai bulan Oktober 2023 karena dianggap tidak efisien yang memakan jam kerja. Sebagai gantinya, mulai Januari 2024, *daily meeting* diganti menjadi *weekly meeting* melalui Discord untuk menyampaikan *progress* proyek oleh penanggung jawab masing-masing dan

dapat saling memberikan umpan balik. *Weekly meeting* tersebut biasanya dilakukan setiap hari Rabu dengan waktu yang menyesuaikan situasi dan kondisi di kantor.

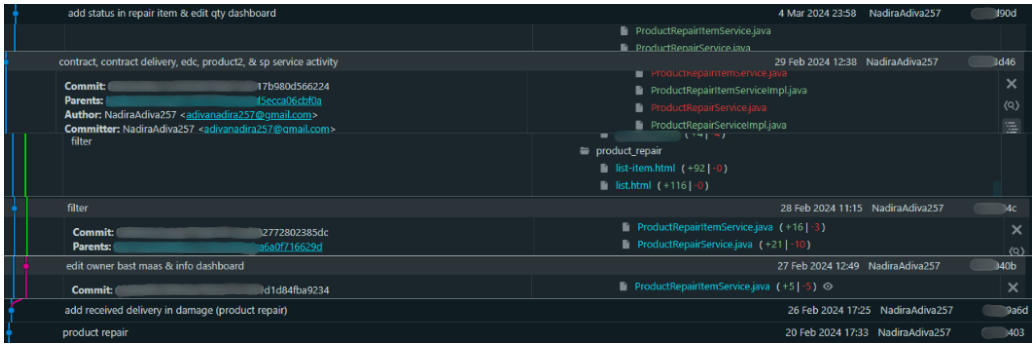
Selain itu, khusus proyek pengembangan inventori EDC berbasis web ini juga memiliki metode pengembangan yang berbeda dengan proyek lain. Proyek lain menggunakan metode pengembangan berupa Scrum. Sementara itu, proyek ini menggunakan metode pengembangan berupa *waterfall* karena dua alasan. Pertama, proyek ini hanya dikerjakan oleh dua anggota, yaitu penulis dan rekan kerja. Kedua, proyek ini merupakan permintaan dari salah satu klien perusahaan yang ikut menyertakan berbagai dokumen dengan informasi yang lengkap kepada *developer* sehingga semua kebutuhan sistem diharapkan sudah jelas dan detail serta berpeluang kecil mengalami perubahan.

## 3.2 Pelaksanaan Proyek

Proyek ini menggunakan MVC dan alur arsitektur Spring Boot seperti yang dijelaskan pada bab ke-2 (landasan teori dan tinjauan pustaka), tepatnya pada subbab 2.1 (*framework* Spring Boot), sehingga teknologi yang digunakan pada sisi *back-end* (*server side*) adalah bahasa pemrograman Java, *framework* Spring Boot, dan *database* PostgreSQL. Untuk metode HTTP *request* yang digunakan dalam proyek ini, yaitu POST untuk mengirim data ke *server* dan dibuatkan menjadi data baru, PUT untuk memperbarui data di *server* secara keseluruhan, PATCH untuk memperbarui data tertentu di *server*, GET untuk mendapatkan data dari *server*, dan DELETE untuk menghapus data dari *server*. Sementara itu, teknologi yang digunakan pada sisi *front-end* (*client side*) adalah gabungan dari *template* web desain Metronic, *template* Thymeleaf, *library* jQuery, dan AJAX. Sebagai tambahan, setiap *entity* memiliki *class* atau *record* untuk *value object* (VO) dan *data transfer object* (DTO). VO dan DTO memiliki berbagai atribut sebagai penampung data. Akan tetapi, VO didapatkan dari *client side* untuk dikirimkan ke *server side*, sedangkan DTO adalah kebalikannya.

### 3.2.1 Pengembangan Fitur Pengembalian EDC yang Rusak

Sebelum membahas pengembangan dari tahapan awal hingga akhir, berikut gabungan dari beberapa bukti pengerjaan yang terkait dengan fitur ini dan dapat dilihat pada Gambar 3.3.




Gambar 3.3 Bukti Pengerjaan pada Pengembalian EDC yang Rusak

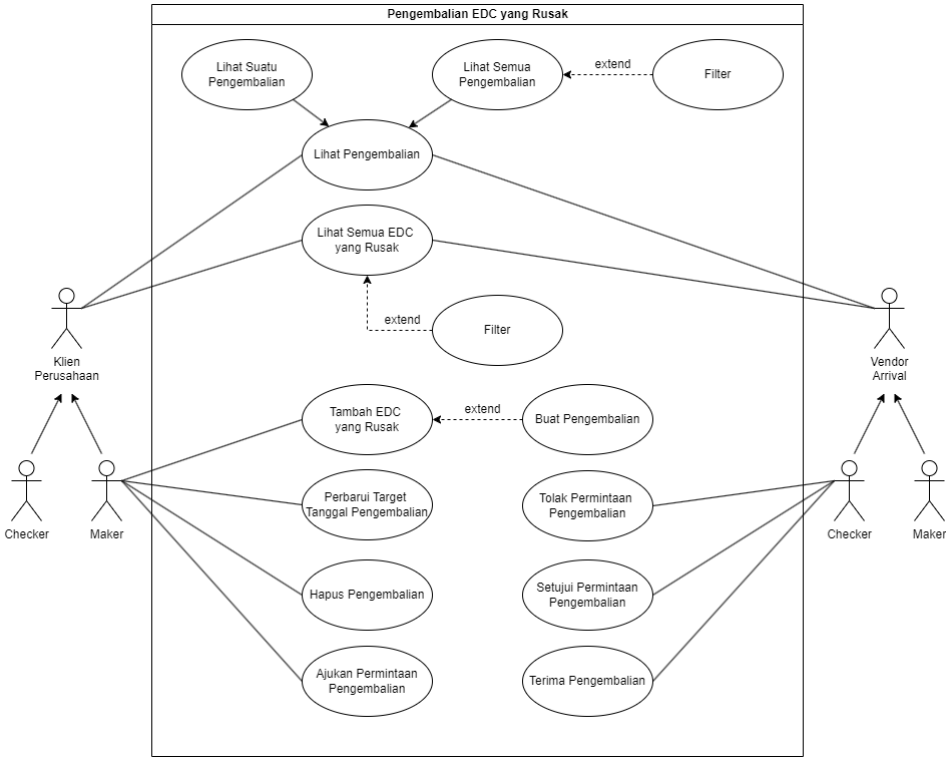
**Analisis Kebutuhan**

Fitur pengembalian EDC yang rusak merupakan salah satu kebutuhan yang didapatkan melalui dokumen kedua (pembaruan) yang disertakan kembali oleh klien perusahaan kepada *developer* di tengah proses pengembangan fitur lain yang masih berjalan. Setelah dokumen tersebut dianalisis, dibutuhkan dua sub fitur lagi. Pertama, fitur *product repair* yang merupakan pengelolaan suatu pengembalian terhadap beberapa EDC yang mengalami kerusakan (*product repair item*) ketika diproses oleh klien perusahaan kepada vendor *arrival*. Kedua, fitur *product repair item* yang merupakan EDC yang mengalami kerusakan dan dikumpulkan dalam jumlah tertentu untuk dikelola menjadi suatu pengembalian (*product repair*) dan disimpan sebagai riwayat.

Terdapat lima fungsi dasar dan empat fungsi tambahan pada *product repair*, yaitu CRUD, *request delivery*, *reject request delivery*, *approve request delivery*, dan *received delivery*. CRUD disebut sebagai lima fungsi dasar karena fungsi *read* terbagi menjadi dua, yaitu *list* dan *view*. Terdapat juga dua fungsi dasar pada *product repair item*, yaitu *create* dan *read-list*. Proses dasar *back-end* pada fungsi CRUD tersebut, kecuali *read-list*, dibuatkan oleh rekan kerja untuk selanjutnya dikerjakan oleh penulis. Fungsi *read-list* pada penjelasan sebelumnya dikecualikan karena prosesnya dibuat oleh penulis, begitu juga dengan proses keempat fungsi tambahan pada *product repair*.

**Desain Sistem**

Terdapat empat rancangan dalam tahap desain sistem, yaitu *use case diagram*, *activity diagram*, *wireframe*, dan relasi antar tabel. Berikut rancangan *use case diagram* pada fitur ini yang dapat dilihat pada Gambar 3.4.

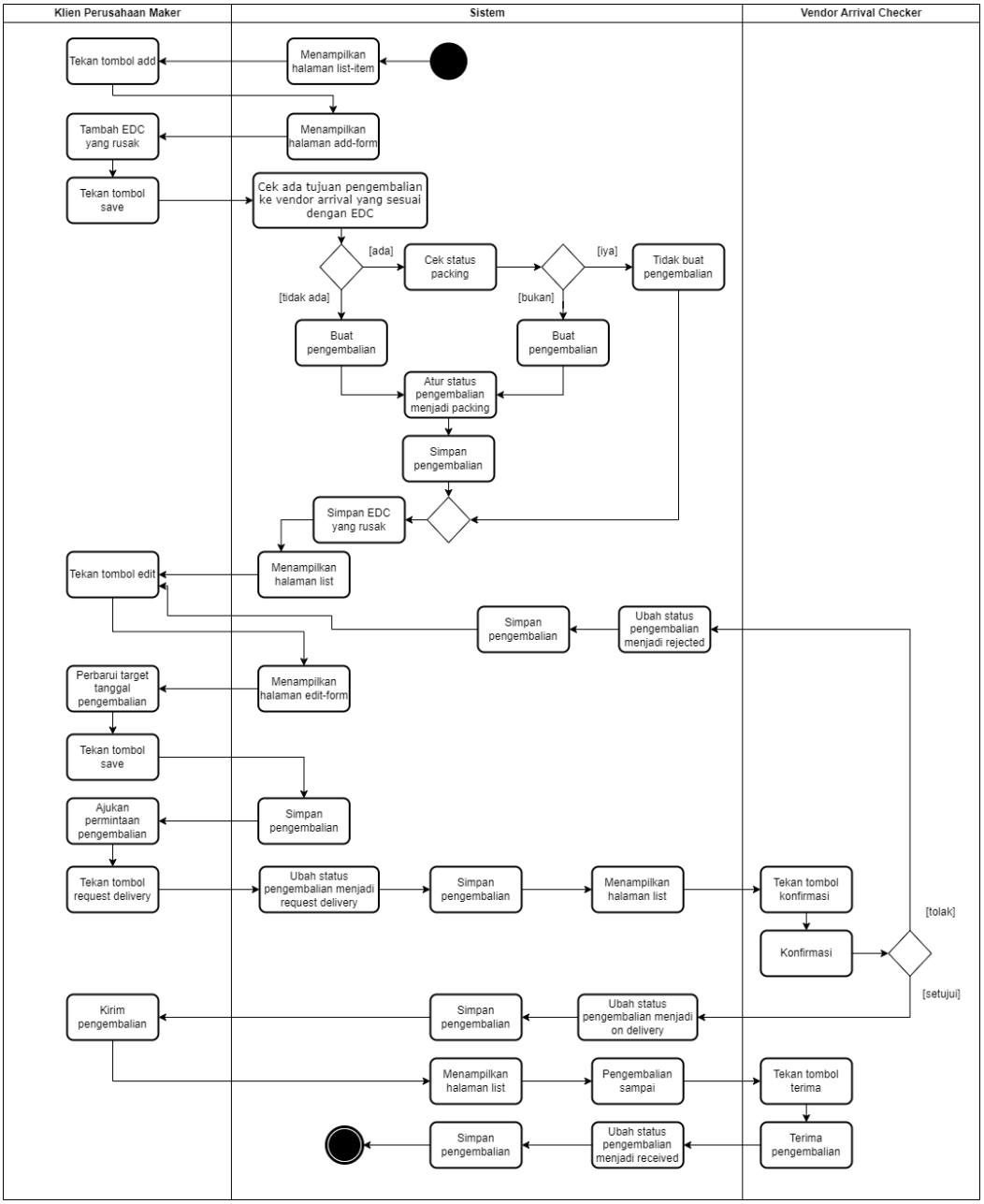



Gambar 3.4 *Use Case Diagram* pada Pengembalian EDC yang Rusak

Berdasarkan Gambar 3.4, terdapat dua *user* yang dapat mengakses berbagai fungsi pada fitur ini, yaitu klien perusahaan dan vendor *arrival* dengan *privilege checker maker* masing-masing. Semua *user* tersebut, baik *checker* maupun *maker*, dapat melihat semua pengembalian dengan dilengkapi filter dari fungsi *read-list* milik *product repair*, melihat semua EDC yang rusak dengan dilengkapi filter dari fungsi *read-list* milik *product repair item*, dan melihat suatu pengembalian berdasarkan *id* dari fungsi *read-view*.

Khusus untuk *user* klien perusahaan *maker*, dapat menambahkan EDC yang rusak dari fungsi *create* milik *product repair item* yang juga dapat melibatkan pembuatan pengembalian dari fungsi *create* milik *product repair*, memperbarui pengembalian berdasarkan *id* dari fungsi *update*, menghapus pengembalian berdasarkan *id* dengan cara *soft delete* dari fungsi *delete*, dan mengajukan permintaan pengembalian berdasarkan *id* dari fungsi *request delivery*. Khusus untuk *user* vendor *arrival checker*, dapat menolak permintaan pengembalian berdasarkan *id* dari fungsi *reject request delivery*, menyetujui permintaan pengembalian berdasarkan *id* dari fungsi *approve request delivery*, dan menerima pengembalian berdasarkan *id* juga dari fungsi *received delivery*.

Kemudian, berikut rancangan *activity diagram* pada fitur ini yang dapat dilihat pada Gambar 3.5.



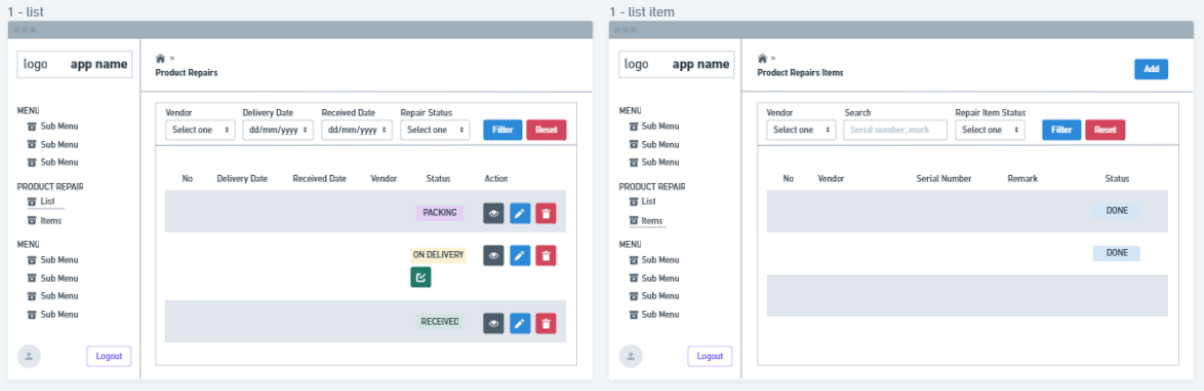

Gambar 3.5 *Activity Diagram* pada Pengembalian EDC yang Rusak

Berdasarkan Gambar 3.5, alur proses pada fitur ini dapat dijelaskan secara singkat. EDC yang ternyata mengalami kerusakan ketika diproses oleh klien perusahaan akan ditambahkan ke *product repair item* oleh klien perusahaan *maker*. Penambahan tersebut akan memengaruhi tiga kondisi tentang *product repair*. Pertama, jika ternyata ada pengembalian yang ditujukan ke vendor *arrival* yang sama dengan vendor *arrival* yang dimiliki EDC dan statusnya masih berupa *packing*, maka tidak membuat pengembalian yang baru sehingga EDC yang rusak dikumpulkan pada pengembalian tersebut. Kedua, jika tidak ada pengembalian yang ditujukan

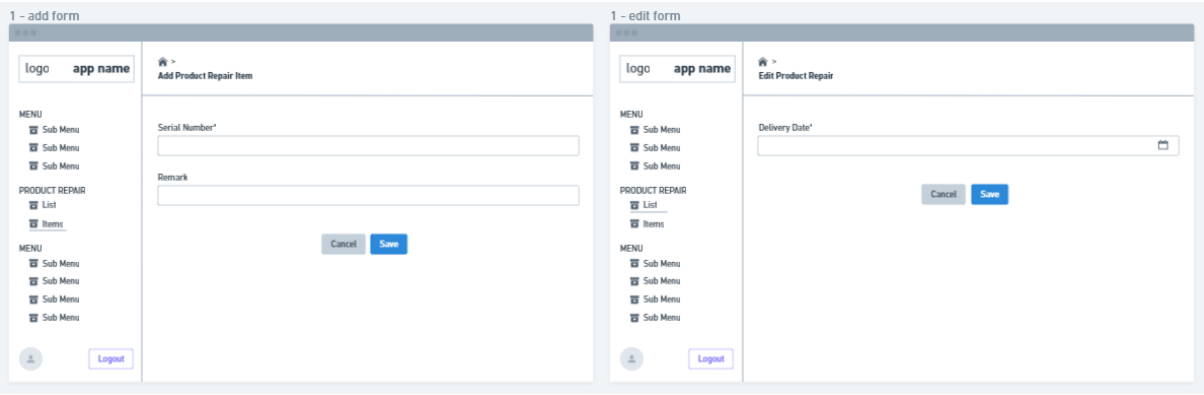
ke vendor *arrival* yang dimiliki EDC, maka membuat pengembalian yang baru dan EDC yang rusak dikumpulkan pada pengembalian baru tersebut. Ketiga, jika ternyata ada pengembalian yang kondisinya hampir sama dengan kondisi pertama tetapi statusnya bukan berupa *packing*, maka membuat pengembalian yang baru dan EDC yang rusak dikumpulkan pada pengembalian baru tersebut. Kedua pengembalian yang baru tersebut akan diatur statusnya menjadi *packing*.

Selanjutnya, klien perusahaan *maker* harus memperbarui target tanggal pengembalian untuk dapat mengajukan permintaan pengembalian sehingga status berubah menjadi *request delivery*. Hal tersebut dilakukan ketika kumpulan EDC yang rusak pada pengembalian tersebut telah mencapai jumlah tertentu. Vendor *arrival checker* lalu akan mengonfirmasi apakah permintaan pengembalian ditolak atau disetujui. Jika ditolak, maka status berubah menjadi *rejected* dan klien perusahaan *maker* harus memperbarui ulang target tanggal pengembalian untuk dapat mengajukan permintaan pengembalian lagi. Jika disetujui, maka status berubah menjadi *on delivery*. Sebagai tambahan informasi, persetujuan harus dilakukan sebelum target tanggal pengembalian yang telah diperbarui atau diatur supaya klien perusahaan dapat segera mengirimkan pengembalian melalui ekspedisi dan dapat diterima oleh vendor *arrival checker* sesuai target atau bahkan sebelum target. Jika pengembalian sudah sampai, maka vendor *arrival checker* menerima pengembalian dan status berubah menjadi *received*. Kemudian, EDC yang rusak akan segera diperbaiki oleh vendor *arrival*. EDC yang sudah diperbaiki nantinya akan dikirimkan kembali oleh vendor *arrival* ke klien perusahaan untuk diproses kembali sehingga EDC yang rusak akan berstatus *done* dan menjadi riwayat.

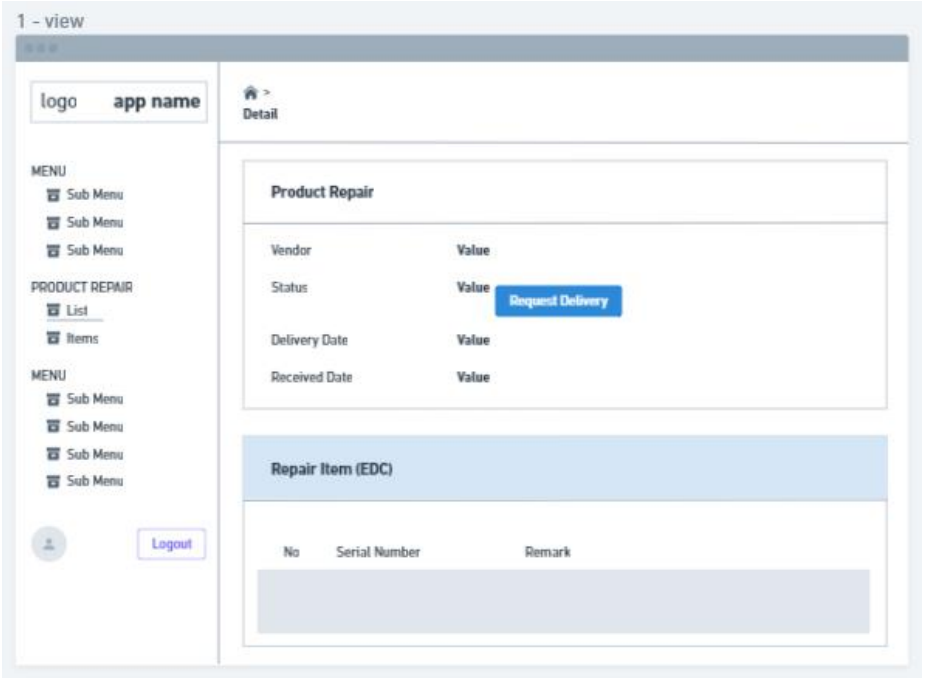
Kemudian, berikut rancangan *wireframe* pada fitur ini yang dapat dilihat pada Gambar 3.6, Gambar 3.7, dan Gambar 3.8.




Gambar 3.6 *Wireframe* pada Pengembalian EDC yang Rusak (1)




Gambar 3.7 *Wireframe* pada Pengembalian EDC yang Rusak (2)

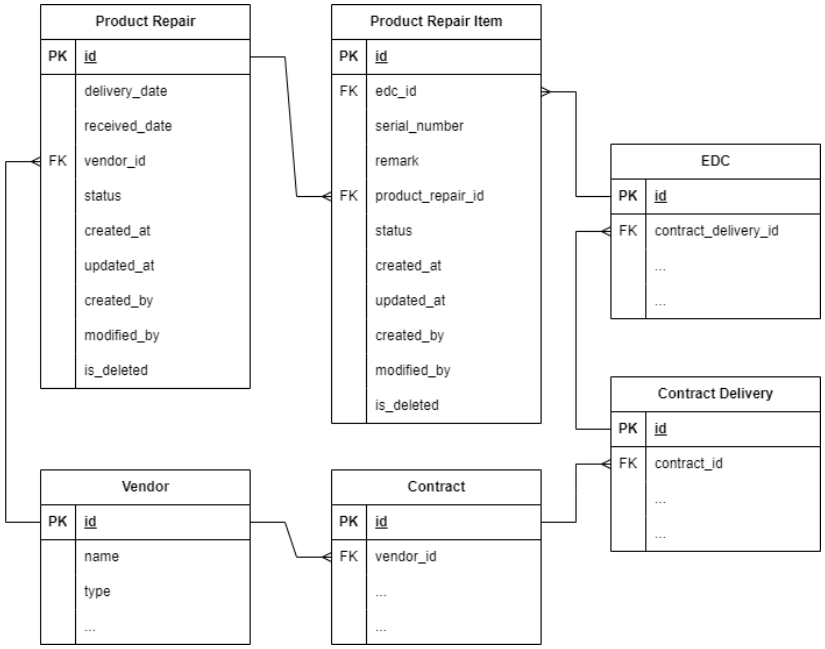



Gambar 3.8 *Wireframe* pada Pengembalian EDC yang Rusak (3)

Berdasarkan Gambar 3.6, Gambar 3.7, dan Gambar 3.8, desain antarmuka pada fitur ini dapat dijelaskan secara singkat. Terdapat lima halaman yang dibagi untuk kedua sub fitur, yaitu *list*, *view*, dan *edit-form* untuk *product repair* serta *list-item* dan *add-form* untuk *product repair item*. Pada halaman *list*, *list-item*, dan *view* terdapat tabel untuk menampilkan semua data. Tabel akan dibuat dengan DataTables (*plugin* jQuery) yang mendukung manipulasi data dalam tabel, proses paginasi, dan proses pengurutan menggunakan bantuan AJAX. Khusus halaman *list*, terdapat kolom *action* yang memiliki tiga tombol ke halaman *view*, ke halaman *edit-form*, dan untuk fungsi *delete*, serta kolom status yang memiliki tombol centang dan silang jika status berupa *request delivery* atau *on delivery* dan *user* yang *login* adalah vendor *arrival checker*. Khusus halaman *list-item*, terdapat tombol *add* ke halaman *add-form*. Selain itu, pada halaman *list* dan *list-item* terdapat *dropdown* untuk menampilkan pilihan. *Dropdown* akan dibuat dengan select2 (*plugin* jQuery) yang dapat mendukung pencarian dalam pilihan dan pengisian pilihan secara dinamis.

Dapat disimpulkan juga bahwa setiap halaman mengerjakan fungsi tertentu melalui *endpoint* yang dimiliki. Pertama, halaman *list* memiliki *endpoint read-list* milik *product repair* dengan HTTP GET, *endpoint delete* dengan HTTP DELETE, serta *endpoint reject request delivery*, *approve request delivery*, dan *received delivery* dengan HTTP PATCH. Kedua, halaman *view* memiliki *endpoint read-view* dengan HTTP GET, *endpoint request delivery* dengan HTTP PUT, dan *endpoint read-list* milik *product repair item* dengan HTTP GET. Ketiga, halaman *edit-form* memiliki *endpoint read-view* juga dan *endpoint update* dengan HTTP PUT. Keempat, halaman *list-item* memiliki *endpoint read-list* milik *product repair item* juga. Kelima, halaman *add-form* memiliki *endpoint create* milik *product repair item* dengan HTTP POST.

Terakhir, berikut rancangan relasi antar tabel pada fitur ini yang dapat dilihat pada Gambar 3.9.




Gambar 3.9 Relasi Antar Tabel pada Pengembalian EDC yang Rusak

Berdasarkan Gambar 3.9, terdapat rancangan dua tabel pada *database* untuk kedua sub fitur. Pada *product repair* terdapat beberapa atribut, seperti *id* sebagai *primary key*, *delivery date*, *received date*, vendor *id*, dan status. Pada *product repair item* terdapat beberapa atribut, seperti *id* sebagai *primary key*, EDC *id*, *serial number*, *remark*, *product repair id*, dan status. Kedua tabel tersebut dikelompokkan ke dalam skema *product*. Tabel *product repair item* memiliki relasi *many to one* dengan tabel *product repair*. Hal tersebut ditunjukkan dengan *foreign key* pada *product repair item* (*product repair id*) yang merujuk ke *primary key* pada *product repair* (*id*) sehingga setiap EDC yang rusak pasti dimiliki oleh satu pengembalian dan setiap pengembalian dapat memiliki satu atau banyak EDC yang rusak.

Tabel *product repair* juga memiliki relasi *many to one* dengan tabel vendor. Hal tersebut ditunjukkan dengan *foreign key* pada *product repair* (vendor *id*) yang merujuk ke *primary key* pada *vendor* (*id*) sehingga setiap pengembalian pasti memiliki satu vendor *arrival* sebagai tujuan pengembalian dan setiap vendor *arrival* dapat memiliki satu atau banyak pengembalian. Tabel *product repair item* juga memiliki relasi *many to one* dengan tabel EDC. Hal tersebut ditunjukkan dengan *foreign key* pada *product repair item* (EDC *id*) yang merujuk ke *primary key* pada EDC (*id*) sehingga setiap EDC yang rusak pasti berkaitan dengan satu EDC dan setiap EDC dapat memiliki satu atau banyak riwayat kerusakan EDC.

**Implementasi**

Implementasi kode dilakukan setelah mengetahui dan memahami kebutuhan fungsi, hak akses *user* melalui *use case diagram*, alur proses melalui *activity diagram*, desain antarmuka melalui *wireframe*, dan relasi antar tabel. Dengan begitu, setiap fungsi pada fitur pengembalian EDC yang rusak dapat terealisasikan. Sebagai tambahan informasi, fungsi *create* dan *update* membutuhkan VO, fungsi *read-view* membutuhkan DTO, dan fungsi *read-list* membutuhkan keduanya.

Kebutuhan VO pada fungsi *read-list* digunakan untuk proses filter. Filter untuk mencari data yang memenuhi satu atau beberapa kondisi dari banyaknya kondisi yang disediakan. Pada *product repair*, filter untuk mencari pengembalian berdasarkan empat kondisi, yaitu *delivery date*, *received date*, vendor, dan status. Pada *product repair item*, filter untuk mencari EDC yang rusak berdasarkan lima kondisi, yaitu *serial number*, *remark*, vendor, *product repair*, dan status. Adanya filter berdasarkan vendor pada kedua sub fitur tersebut juga digunakan untuk otomatisasi penyesuaian pengambilan data berdasarkan *user* yang *login*. Jika *user* yang *login* adalah klien perusahaan, maka dapat mengambil semua pengembalian dan semua EDC yang rusak ke vendor *arrival* manapun. Jika *user* yang *login* adalah vendor *arrival*, maka hanya dapat mengambil pengembalian dan EDC yang rusak yang ditujukan kepada vendor mereka.

Selanjutnya, berikut potongan kode tentang pemanfaatan Spring Boot sebagai teknologi pada sisi *back-end* yang dapat dilihat pada Gambar 3.10 dan Gambar 3.11.

```
1. public interface ProductRepairRepository extends JpaRepository
<ProductRepair, String> {}
```

Gambar 3.10 Kode Program Lapisan *Repository* pada *Product Repair*

```
1. @Service
2. @RequiredArgsConstructor
3. public class ProductRepairServiceImpl implements ProductRepairService {
4.    private final ProductRepairRepository productRepairRepository;
      ...

5.    @Override
6.    public String add(ProductRepairVO vO) {
7.       ProductRepair bean = new ProductRepair();
8.       BeanUtils.copyProperties(vO, bean);
9.       ...
10.      bean = productRepairRepository.save(bean);
11.      return bean.getId();
12.   }
      ...
13.}
```

Gambar 3.11 Kode Program Lapisan *Service* pada *Product Repair*

Gambar 3.10 adalah lapisan *repository* yang menggunakan Spring Data JPA *repository* untuk memudahkan operasi fungsi dasar pada data *entity product repair* tanpa perlu menulis kueri SQL secara manual. Sementara itu, Gambar 3.11 adalah lapisan *service* yang mengatur logika bisnis pada fungsi *product repair* dengan menggunakan bantuan lapisan *repository* yang diintegrasikan melalui *constructor injection*. *Constructor injection* merupakan salah satu jenis *dependency injection* yang menjadi ciri khas dan keunggulan dari Spring Boot.

*Constuctor injection* tersebut dibuktikan pada Gambar 3.11 di baris ke-2 yang memiliki anotasi berupa @RequiredArgsConstructor dari dependensi Lombok dan secara otomatis akan membuat konstruktor dengan parameter yang mencakup semua atribut *final* pada *product repair service impl*. Salah satunya pada Gambar 3.11 di baris ke-4 yang memiliki atribut *final* bertipe *product repair repository* dari lapisan *repository*. Dengan begitu, Spring Boot dapat mengintegrasikan *product repair repository* dari lapisan *repository* ke *product repair service impl* di lapisan *service* yang dapat dibuktikan pada Gambar 3.11 di baris ke-10. Baris ke-10 tersebut adalah penggunaan fungsi *save* dari *product repair repository* untuk penyimpanan objek *bean* ke *database product repair*. *Bean* tersebut merupakan objek hasil inisialisasi *entity product repair* pada Gambar 3.11 di baris ke-7 dan telah berisi data *entity product repair* dari penyalinan atribut-atribut di VO pada Gambar 3.11 di baris ke-8.

Untuk potongan kode tentang pemanfaatan Thymeleaf sebagai teknologi pada sisi *front-end* dapat dilihat pada Gambar 3.12 dan Gambar 3.13.

```
1. public enum ProductRepairStatus {
2.    PACKING, REQUEST_DELIVERY, REJECTED, ON_DELIVERY, RECEIVED;
3. }
```

Gambar 3.12 Kode Program pada *Enum Product Repair Status*

```
1. <select ...>
2.    <option ...>Select one</option>
3.    <option th:each="item : ${T(...ProductRepairStatus).values()}"
th:value="${item.name()}">
4.       <span th:text="${item.name()}"></span>
5.    </option>
6. </select>
```

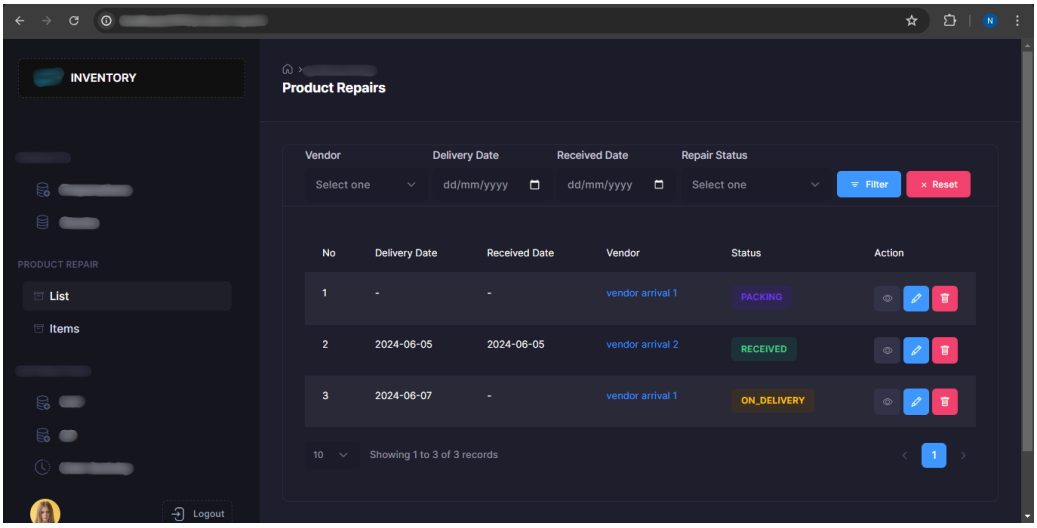
Gambar 3.13 Kode Program Halaman *List* pada *Dropdown Product Repair Status*

Gambar 3.12 adalah *enum* untuk status *product repair* yang terdiri dari kumpulan nilai konstan yang telah ditentukan dan tidak dapat diubah. Gambar 3.13 adalah *dropdown* untuk menampilkan pilihan status *product repair* di halaman *list* dengan Thymeleaf. Hal tersebut dibuktikan pada Gambar 3.13 di baris ke-3 yang melakukan pengambilan semua nilai dari

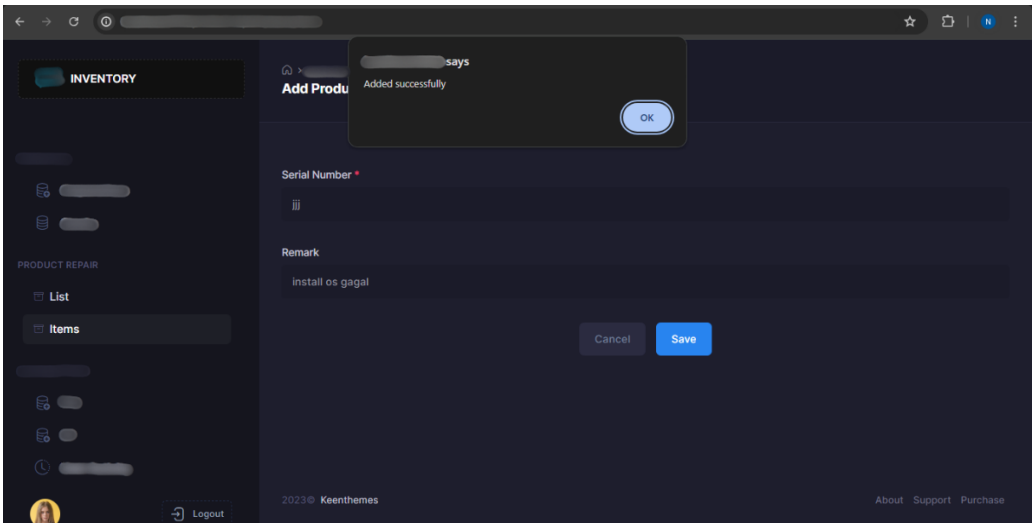
*enum product repair status* dengan kode khusus Thymeleaf, yaitu T, yang dapat mengakses kelas Java secara hierarki folder dan *file*. Selanjutnya, semua nilai *enum* tersebut diwakilkan sebagai *item* di setiap pengulangan pilihan *dropdown* melalui atribut Thymeleaf, yaitu th:each. Maka dari itu, setiap pilihan *dropdown* akan memiliki nilai berupa nama dari nilai *enum* melalui atribut Thymeleaf, yaitu th:value. Pada Gambar 3.13 di baris ke-4, setiap pilihan *dropdown* juga akan memiliki teks berupa nama dari nilai *enum* melalui atribut Thymeleaf, yaitu th:text.

Selanjutnya, jika implementasi setiap fungsi di sisi *back-end* sudah selesai, maka penulis membuat berbagai halaman di sisi *front-end*. Pembuatan halaman mengikuti rancangan desain antarmuka dengan melibatkan *endpoint* dari setiap fungsi. Penjelasan mengenai hal tersebut dapat dilihat pada Gambar 3.6, Gambar 3.7, dan Gambar 3.8. Hasil implementasi dari kedua sisi tersebut pada fitur ini dapat dilihat mulai dari Gambar 3.14.

Pada Gambar 3.14 adalah halaman *list* yang menjalankan fungsi *read-list* milik *product repair* sehingga menampilkan semua pengembalian ke vendor *arrival* manapun dalam bentuk tabel karena *user* yang *login* adalah klien perusahaan *maker*. Selanjutnya, pada Gambar 3.15 adalah halaman *add-form* yang menjalankan fungsi *create* milik *product repair item* sehingga menampilkan keberhasilan penambahan EDC yang rusak dengan *serial number* jjj oleh klien perusahaan *maker*. Hasil penambahan tersebut tidak membuat pengembalian yang baru karena vendor yang dimiliki oleh EDC yang rusak tersebut adalah vendor *arrival* 1 dan ternyata pada Gambar 3.14 ada pengembalian yang ditujukan ke vendor *arrival* 1 juga dengan statusnya masih berupa *packing*. Maka dari itu, tampilan pada halaman *list* setelah penambahan tersebut masih sama seperti pada Gambar 3.14.



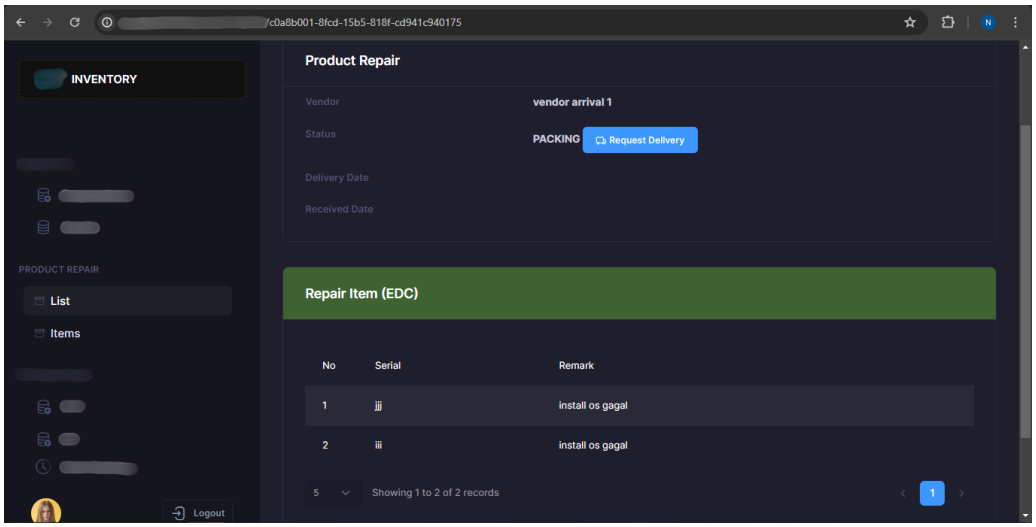

Gambar 3.14 Tampilan Halaman *List* pada *Product Repair* (1)




Gambar 3.15 Tampilan Halaman *Add-Form* pada *Product Repair Item*

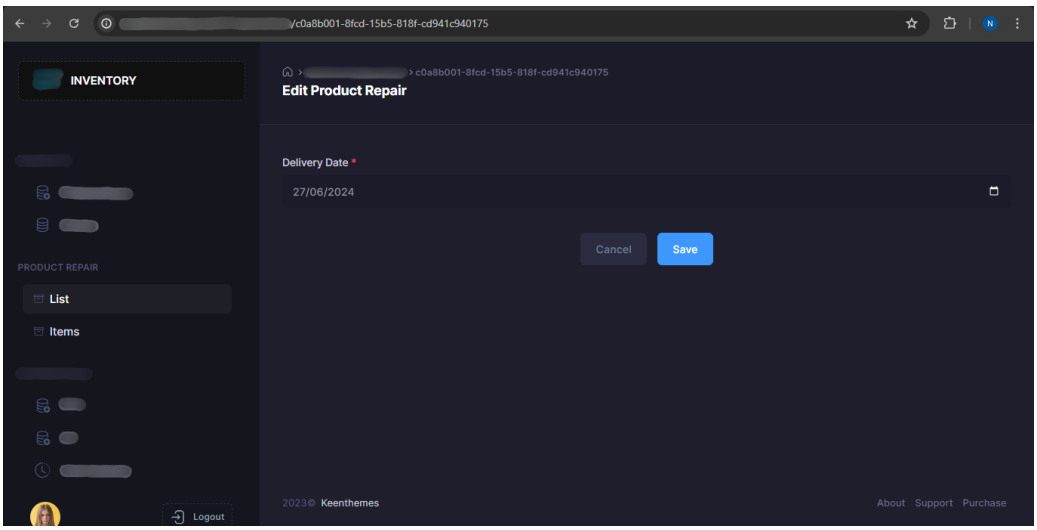
Pada Gambar 3.16 adalah halaman *view* yang menjalankan fungsi *read-view* sehingga menampilkan detail pengembalian berdasarkan *id* yang dimiliki oleh pengembalian dengan tujuan ke vendor *arrival* 1 dan statusnya berupa *packing*. Selain itu, dapat menjalankan fungsi *request delivery* yang ditunjukkan dengan tombol *request delivery* dan menjalankan fungsi *read-list* milik *product repair item* sehingga menampilkan semua kumpulan EDC yang rusak pada pengembalian tersebut menggunakan filter berdasarkan kondisi *product repair* ke dalam bentuk tabel. Terdapat dua EDC yang rusak, yaitu EDC dengan *serial number* iii yang sudah ditambahkan sebelumnya dan EDC dengan *serial number* jjj yang ditambahkan terakhir kali. Maka dari itu, dapat disimpulkan bahwa walaupun hasil penambahan EDC yang rusak pada

Gambar 3.15 tidak membuat pengembalian yang baru, tetapi menambah kumpulan EDC yang rusak pada pengembalian tersebut.



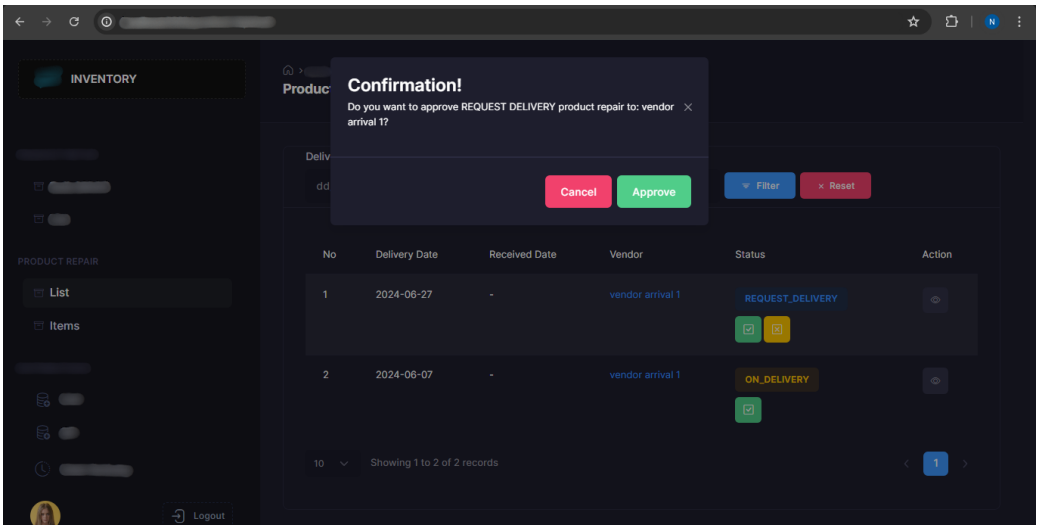

Gambar 3.16 Tampilan Halaman *View* pada *Product Repair*

Pada Gambar 3.17 adalah halaman *edit-form* yang menjalankan fungsi *read-view* sehingga menampilkan target tanggal pengembalian berdasarkan *id* dan menjalankan fungsi *update* sehingga target tanggal pengembalian tersebut dapat diperbarui. Pengembalian yang ditampilkan pada Gambar 3.17 adalah pengembalian dengan tujuan ke vendor *arrival* 1 dan statusnya berupa *packing*. Target tanggal pengembaliannya diperbarui atau diatur menjadi 27 Juni 2024. Dengan begitu, tombol *request delivery* pada Gambar 3.16 dapat ditekan.



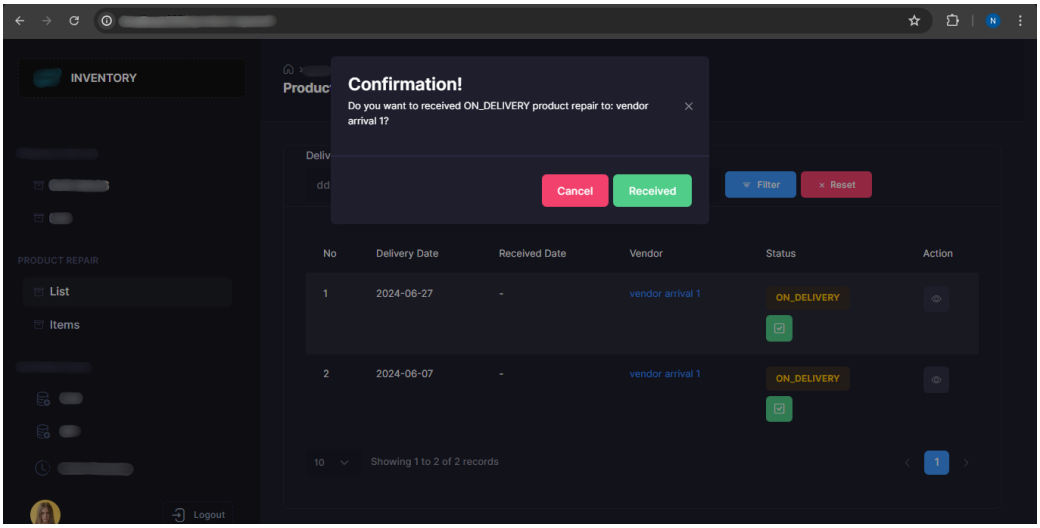

Gambar 3.17 Tampilan Halaman *Edit-Form* pada *Product Repair*

Pada Gambar 3.18 adalah halaman *list* yang sama seperti pada Gambar 3.14. Akan tetapi, *user* yang *login* adalah vendor *arrival* 1 *checker* sehingga hanya menampilkan pengembalian ke vendor *arrival* 1 saja. Terdapat pengembalian dengan tujuan ke vendor *arrival* 1, target tanggal pengembalian pada 27 Juni 2024, dan statusnya berupa *request delivery*. Pengembalian tersebut adalah keberhasilan pengajuan permintaan melalui tombol *request delivery* pada Gambar 3.16 yang sudah ditekan. Terdapat juga *pop up modal* yang muncul ketika tombol centang pada kolom status pengembalian tersebut ditekan oleh *user* vendor *arrival* 1 *checker*. Hal tersebut untuk menjalankan fungsi *approve request delivery* yang mengonfirmasi bahwa permintaan pengembalian akan disetujui.



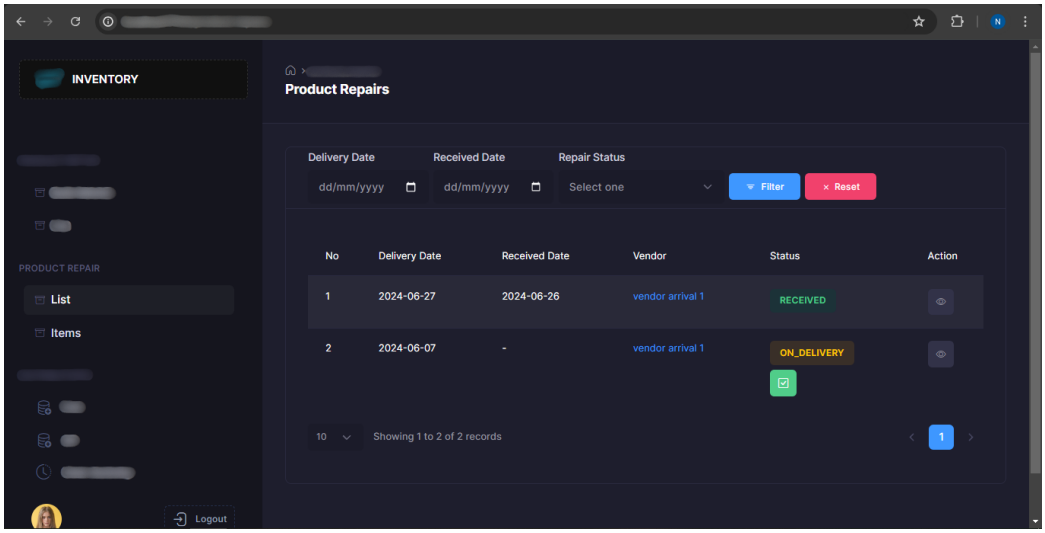

Gambar 3.18 Tampilan Halaman *List* pada *Product Repair* (2)

Pada Gambar 3.19 adalah lanjutan halaman pada Gambar 3.18 yang menampilkan bahwa pengembalian yang dimaksud pada Gambar 3.18 sudah berganti status menjadi *on delivery*. Pengembalian tersebut adalah keberhasilan persetujuan permintaan pada Gambar 3.18. Ada juga *pop up modal* yang muncul ketika tombol centang pada kolom status pengembalian tersebut ditekan oleh *user* vendor *arrival* 1 *checker*. Hal tersebut untuk menjalankan fungsi *received delivery* yang mengonfirmasi bahwa pengembalian akan diterima ketika sudah sampai. Tanggal penerimaan akan diatur secara otomatis sesuai zona waktu *default*.

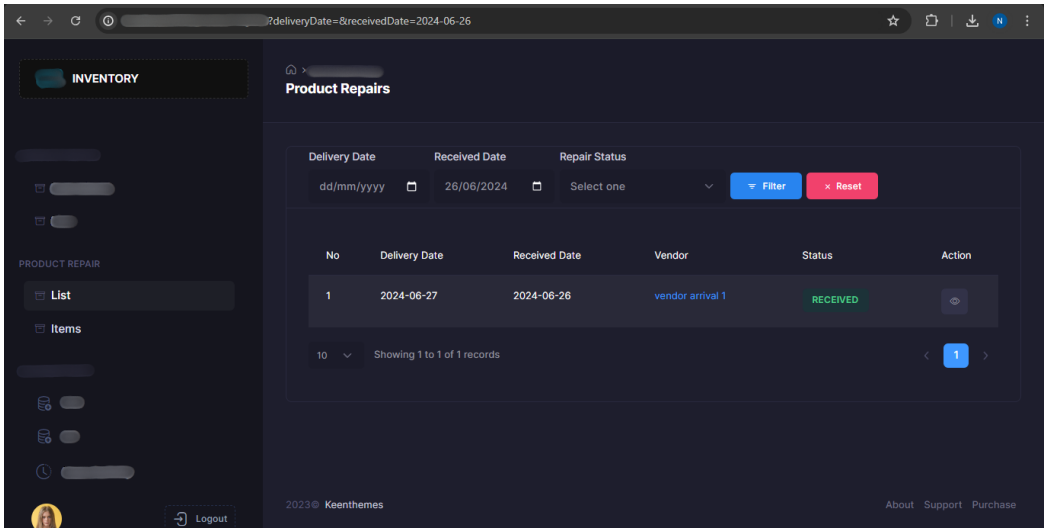



Gambar 3.19 Tampilan Halaman *List* pada *Product Repair* (3)

Pada Gambar 3.20 adalah lanjutan halaman pada Gambar 3.19 yang menampilkan bahwa pengembalian yang dimaksud pada Gambar 3.19 sudah berganti status menjadi *received.* Pengembalian tersebut adalah keberhasilan penerimaan pada Gambar 3.19 yang diterima sebelum target tanggal pengembalian. Kemudian, pada Gambar 3.21 adalah lanjutan halaman pada Gambar 3.20 yang menampilkan semua pengembalian dengan filter yang didasarkan pada kondisi tanggal penerimaan berupa 26 Juni 2024.

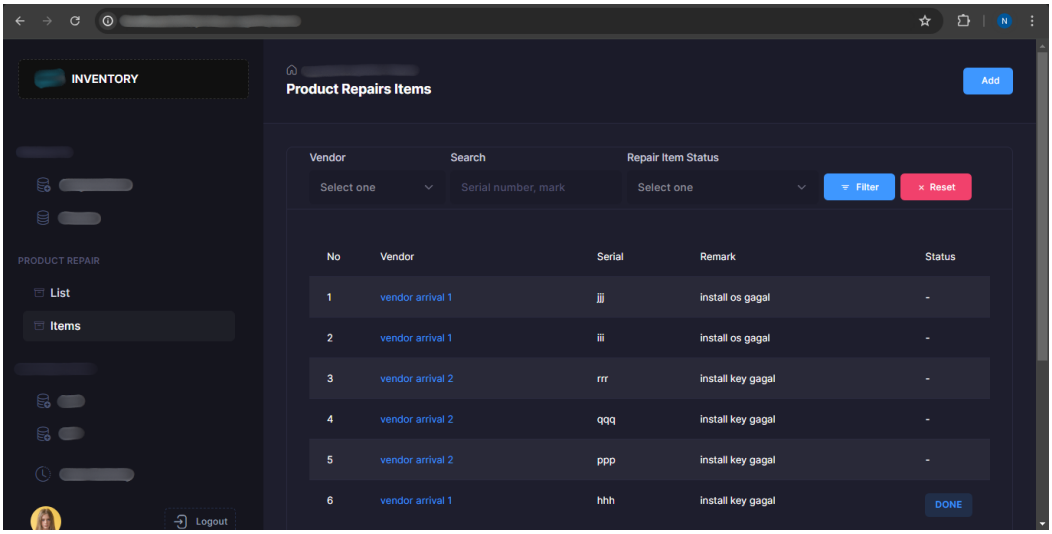



Gambar 3.20 Tampilan Halaman *List* pada *Product Repair* (4)

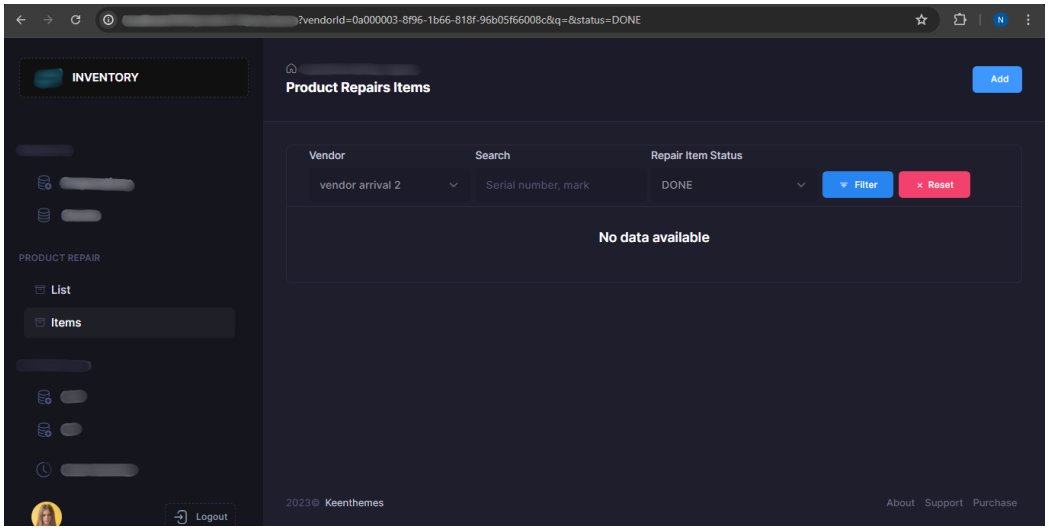



Gambar 3.21 Tampilan Halaman *List* pada *Product Repair* (5)

Pada Gambar 3.22 adalah halaman *list-item* yang menjalankan fungsi *read-list* milik *product repair item* sehingga menampilkan semua EDC yang rusak ke vendor *arrival* manapun dalam bentuk tabel karena *user* yang *login* adalah klien perusahaan *maker*. Pada kolom status dapat menampilkan "*done*" atau tidak menampilkan apapun. Status *done* tersebut menandakan bahwa EDC yang rusak sudah diperbaiki oleh vendor yang bersangkutan dan dikirimkan lagi ke klien perusahaan untuk diproses kembali. Pada Gambar 3.23 adalah lanjutan halaman pada Gambar 3.22 yang menampilkan semua EDC yang rusak dengan filter yang didasarkan pada kondisi vendor berupa vendor *arrival* 2 dan kondisi status berupa *done*.




Gambar 3.22 Tampilan Halaman *List-Item* pada *Product Repair Item* (1)



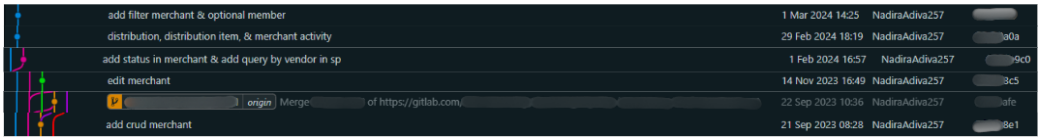

Gambar 3.23 Tampilan Halaman *List-Item* pada *Product Repair Item* (2)

**Pengujian**

Setelah semua fungsi pada fitur pengembalian EDC yang rusak berhasil direalisasikan melalui implementasi dari sisi *back-end* dan *front-end*, dilakukan pengujian menyeluruh tetapi masih sebatas pada lingkup *developer*. Pengujian dilakukan oleh penulis terlebih dahulu untuk kemudian diuji kembali oleh *supervisor*. Secara umum, hasil pengujian berjalan dengan baik dan sudah sesuai dengan kebutuhan yang ditentukan. Namun, pengujian untuk menghasilkan status *done* pada EDC yang rusak belum dilakukan oleh *supervisor*.

### 3.2.2 Pengembangan Fitur *Merchant*

Sebelum membahas pengembangan dari tahapan awal hingga akhir, berikut gabungan dari beberapa bukti pengerjaan yang terkait dengan fitur ini dan dapat dilihat pada Gambar 3.24.



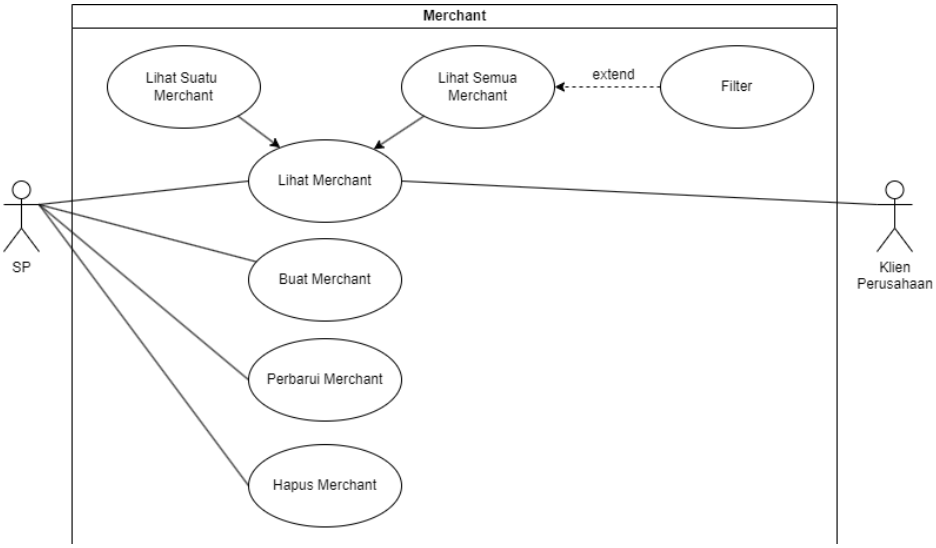

Gambar 3.24 Bukti Pengerjaan pada *Merchant*

**Analisis Kebutuhan**

Fitur *merchant* merupakan salah satu kebutuhan yang didapatkan melalui dokumen pertama yang disertakan oleh klien perusahaan kepada *developer*. Fitur ini mengelola entitas *merchant* yang akan digunakan sebagai tujuan distribusi EDC dari SP. Setelah dokumen tersebut dianalisis, dibutuhkan lima fungsi dasar berupa CRUD yang prosesnya dibuat oleh penulis.

**Desain Sistem**

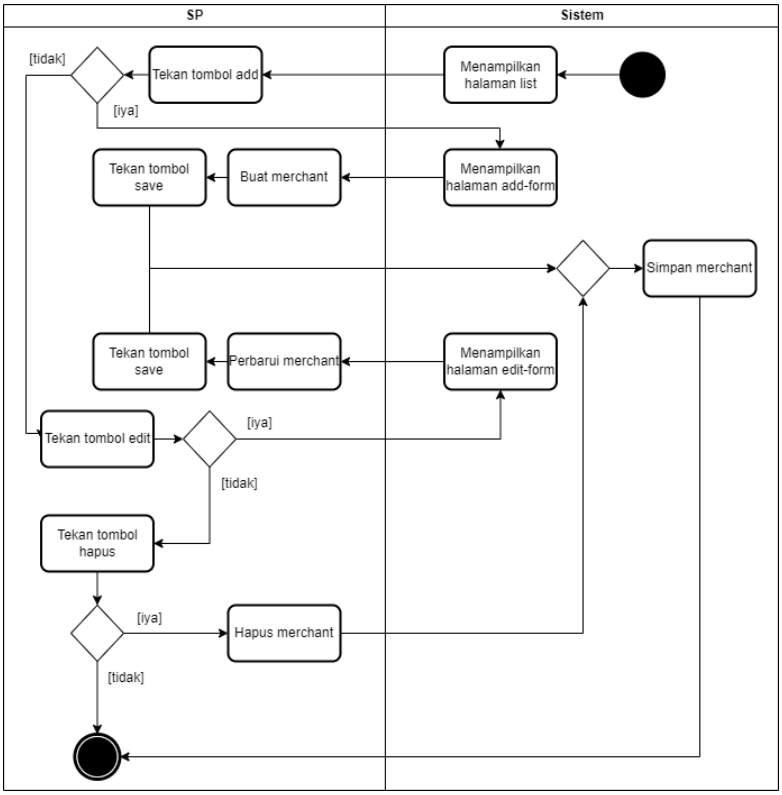
Terdapat empat rancangan dalam tahap desain sistem, yaitu *use case diagram*, *activity diagram*, *wireframe*, dan relasi antar tabel. Berikut rancangan *use case diagram* pada fitur ini yang dapat dilihat pada Gambar 3.25.




Gambar 3.25 *Use Case Diagram* pada *Merchant* (Sebelum *Review*)

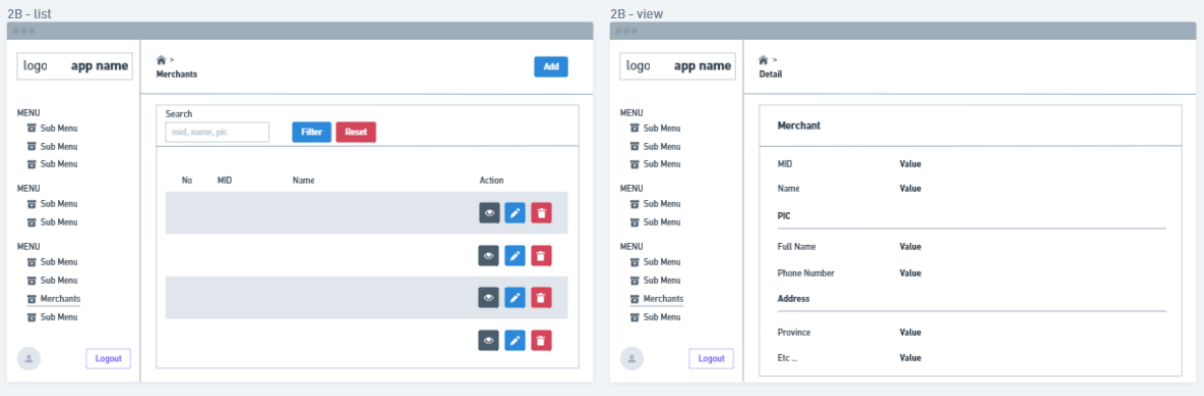
Berdasarkan Gambar 3.25, terdapat dua *user* yang dapat mengakses berbagai fungsi pada fitur ini, yaitu SP dan klien perusahaan. Semua *user* tersebut dapat melihat semua *merchant* dengan dilengkapi filter dari fungsi *read-list* dan melihat suatu *merchant* berdasarkan *id* dari fungsi *read-view*. Khusus untuk *user* SP, dapat melakukan tiga hal, yaitu membuat *merchant* dari fungsi *create*, memperbarui *merchant* berdasarkan *id* dari fungsi *update*, dan menghapus *merchant* berdasarkan *id* dengan cara *soft delete* dari fungsi *delete*.

Kemudian, rancangan *activity diagram* untuk alur proses pada fitur ini dapat dijelaskan secara singkat karena fokusnya hanya untuk pengelolaan entitas *merchant*. SP dapat membuat, memperbarui, atau menghapus suatu *merchant*. Setelah melakukan salah satu hal tersebut, *merchant* akan disimpan pada *database* dan alur proses akan berakhir. Rancangan tersebut dapat dilihat pada Gambar 3.26.

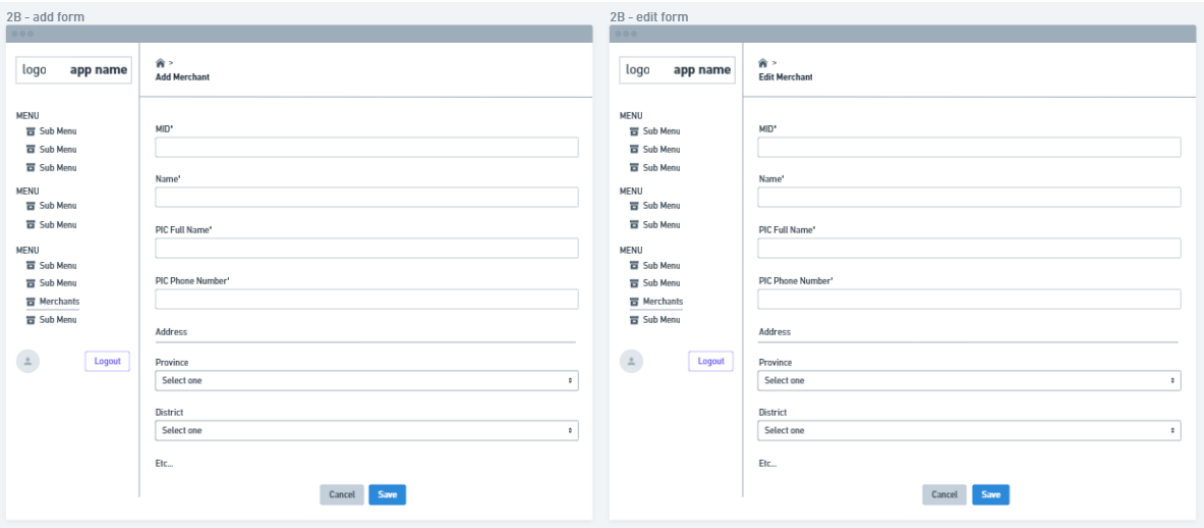



Gambar 3.26 *Activity Diagram* pada *Merchant* (Sebelum *Review*)

Kemudian, berikut rancangan *wireframe* pada fitur ini yang dapat dilihat pada Gambar 3.27 dan Gambar 3.28.




Gambar 3.27 *Wireframe* pada *Merchant* (Sebelum *Review*) (1)



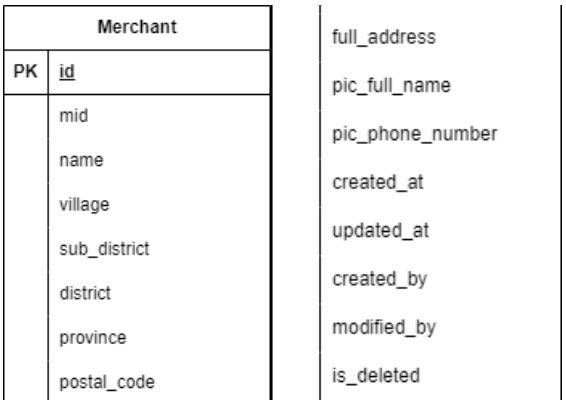

Gambar 3.28 *Wireframe* pada *Merchant* (Sebelum *Review*) (2)

Berdasarkan Gambar 3.27 dan Gambar 3.28, desain antarmuka pada fitur ini dapat dijelaskan secara singkat. Terdapat empat halaman, yaitu *list*, *view*, *add-form*, dan *edit-form*. Pada halaman *list* terdapat tabel untuk menampilkan semua data, tombol *add* ke halaman *add-form*, dan kolom *action* yang memiliki tiga tombol ke halaman *view*, ke halaman *edit-form*, serta untuk fungsi *delete*. Selain itu, pada halaman *list*, *add-form*, dan *edit-form* terdapat *dropdown* untuk menampilkan pilihan. Pembuatan tabel dan *dropdown* akan menggunakan teknologi yang sama dengan penjelasan pada Gambar 3.6, Gambar 3.7, dan Gambar 3.8.

Dapat disimpulkan juga bahwa setiap halaman mengerjakan fungsi tertentu melalui *endpoint* yang dimiliki. Pertama, halaman *list* memiliki *endpoint read-list* dengan HTTP GET dan *endpoint delete* dengan HTTP DELETE. Kedua, halaman *view* memiliki *endpoint read-view* dengan HTTP GET. Ketiga, halaman *add-form* memiliki *endpoint create* dengan HTTP

POST. Keempat, halaman *edit-form* memiliki *endpoint read-view* juga dan *endpoint update* dengan HTTP PUT.

Terakhir, rancangan relasi antar tabel pada fitur ini berupa tabel *merchant* pada *database* dengan beberapa atribut yang dimiliki, seperti *id* sebagai *primary key*, *merchant identification number* (MID), *name*, *person in charge* (PIC) *full name*, PIC *phone number*, dan *postal code*. Tabel *merchant* dikelompokkan ke dalam skema *management* dan tidak memiliki relasi apapun dengan tabel lain. Rancangan tersebut dapat dilihat pada Gambar 3.29.




Gambar 3.29 Relasi Antar Tabel pada *Merchant* (Sebelum *Review*)

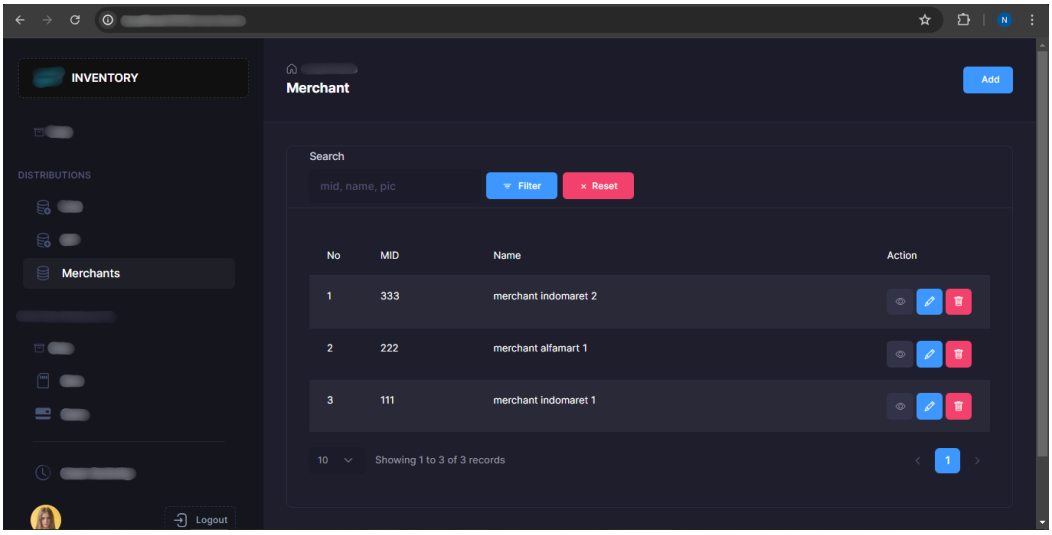
**Implementasi**

Implementasi kode dilakukan setelah mengetahui dan memahami kebutuhan fungsi, hak akses *user* melalui *use case diagram*, alur proses melalui *activity diagram*, desain antarmuka melalui *wireframe*, dan relasi antar tabel. Dengan begitu, setiap fungsi pada fitur *merchant* dapat terealisasikan. Sebagai tambahan informasi, fungsi *create* dan *update* membutuhkan VO, fungsi *read-view* membutuhkan DTO, dan fungsi *read-list* membutuhkan keduanya. Adanya kebutuhan VO pada fungsi *read-list* digunakan untuk proses filter. Filter untuk mencari *merchant* yang memenuhi satu atau beberapa kondisi dari empat kondisi yang disediakan, yaitu MID, *name*, PIC *full name*, dan PIC *phone number*.

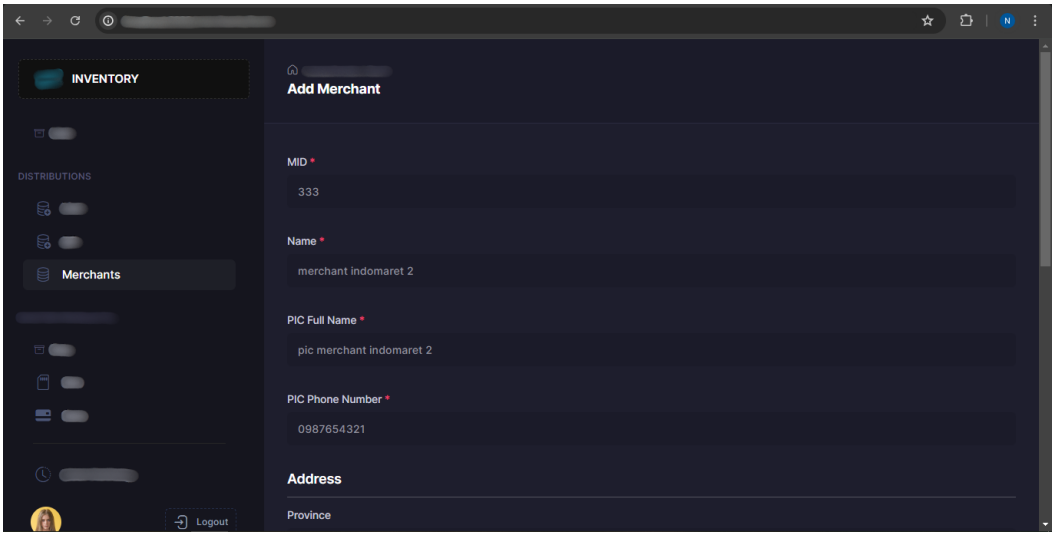
Jika implementasi setiap fungsi di sisi *back-end* sudah selesai, maka penulis membuat berbagai halaman di sisi *front-end*. Pembuatan halaman mengikuti rancangan desain antarmuka dengan melibatkan *endpoint* dari setiap fungsi. Penjelasan mengenai hal tersebut dapat dilihat pada Gambar 3.27 dan Gambar 3.28. Hasil implementasi dari kedua sisi tersebut pada fitur ini dapat dilihat mulai dari Gambar 3.30.

Pada Gambar 3.30 adalah halaman *list* yang menjalankan fungsi *read-list* sehingga menampilkan semua *merchant* dalam bentuk tabel pada *user* SP. Terdapat *merchant* dengan

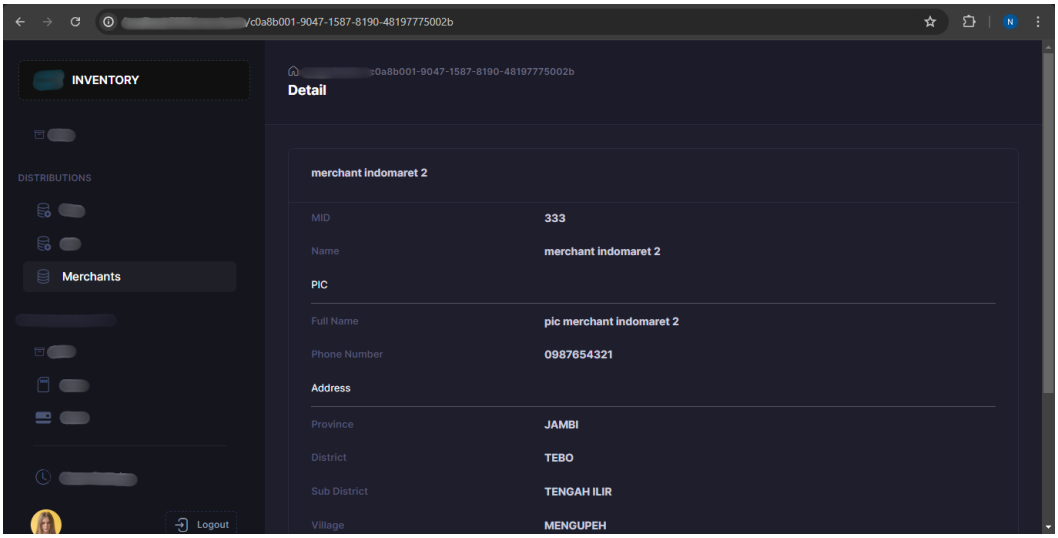
MID 333. *Merchant* tersebut didapatkan dari Gambar 3.31, yaitu halaman *add-form* yang menjalankan fungsi *create* sehingga *merchant* dengan MID 333 berhasil dibuat oleh *user* SP. Pada Gambar 3.32 adalah halaman *view* yang menjalankan fungsi *read-view* sehingga dapat menampilkan detail *merchant* berdasarkan *id* yang dimiliki oleh *merchant* dengan MID 333.




Gambar 3.30 Tampilan Halaman *List* pada *Merchant* (Sebelum *Review*)




Gambar 3.31 Tampilan Halaman *Add-Form* pada *Merchant* (1)



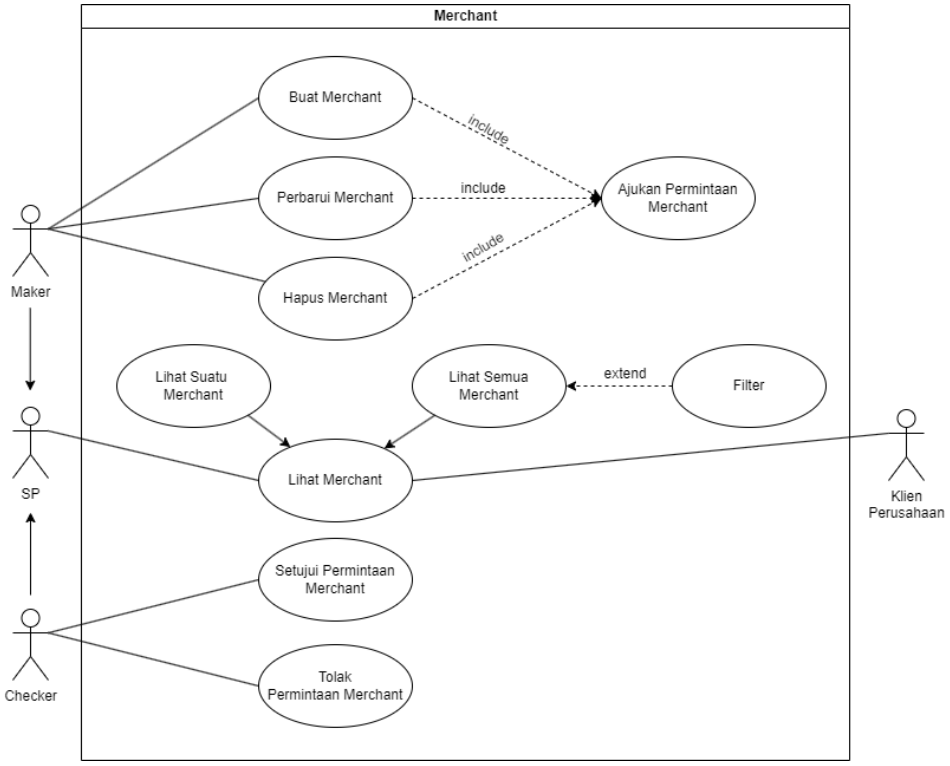

Gambar 3.32 Tampilan Halaman *View* pada *Merchant*

**Pengujian**

Setelah semua fungsi pada fitur *merchant* berhasil direalisasikan melalui implementasi dari sisi *back-end* dan *front-end*, dilakukan pengujian menyeluruh yang masih sebatas pada lingkup *developer*. Pengujian dilakukan oleh penulis terlebih dahulu untuk kemudian diuji kembali oleh *supervisor*. Secara umum, hasil pengujian berjalan dengan baik dan sudah sesuai dengan kebutuhan yang ditentukan.

Di tengah proses pengembangan fitur lain yang masih berjalan, dilakukan presentasi pengujian oleh rekan kerja kepada klien perusahaan dan fitur *merchant* termasuk fitur yang diujikan. Terdapat berbagai *review* untuk perbaikan dari hasil pengujian tersebut. Salah satu *review* yang berhubungan dengan fitur *merchant* adalah dibutuhkan validasi atau konfirmasi bahwa pembuatan, pembaruan, atau penghapusan suatu data entitas sudah tepat dan sesuai melalui *privilege checker maker* pada *user*. Kebutuhan tersebut mengharuskan proses berulang dari tahapan analisis kebutuhan hingga pengujian.

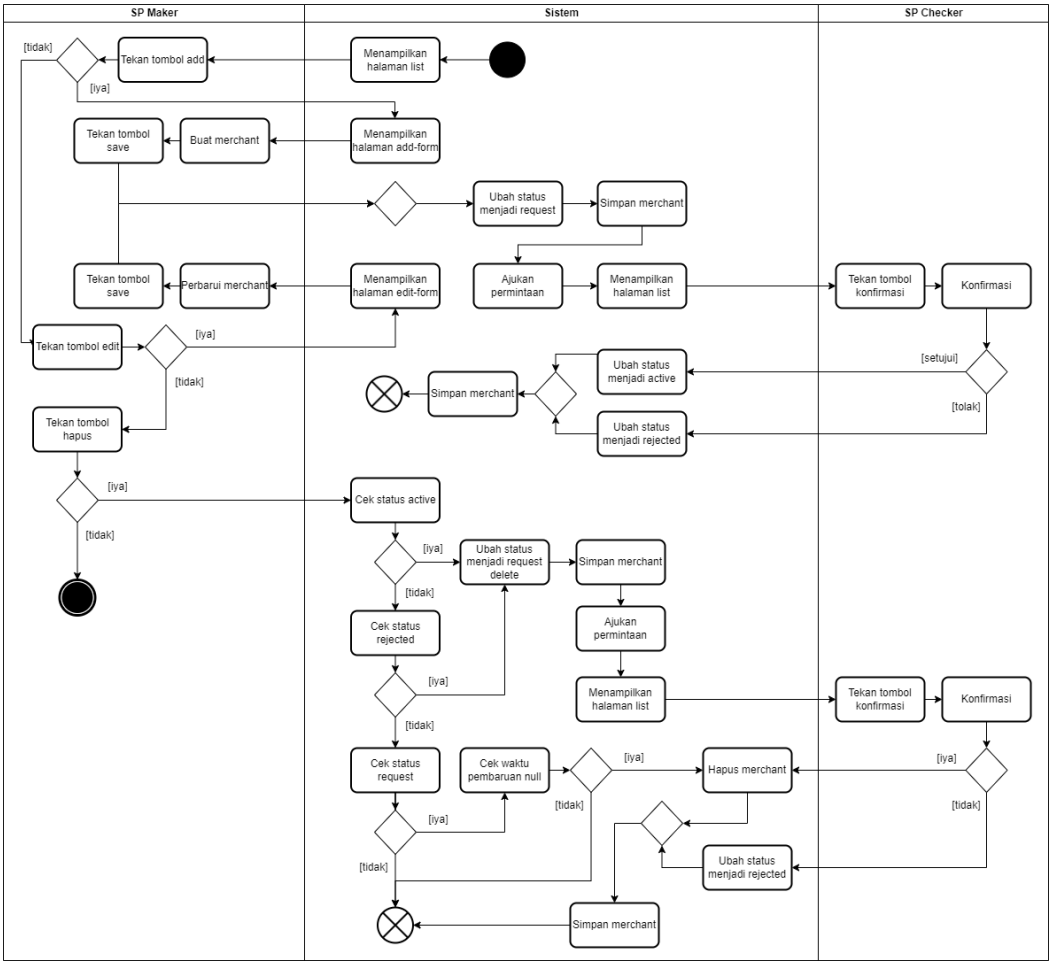
Terdapat dua kebutuhan fungsi terbaru, yaitu *approve* dan *reject*. Berikut rancangan *use case diagram* terbaru yang dapat dilihat pada Gambar 3.33.




Gambar 3.33 *Use Case Diagram* pada *Merchant* (Setelah *Review*)

Berdasarkan Gambar 3.33, pembaruan terjadi pada *user* SP yang memiliki *privilege checker maker*. Semua *user* tetap dapat melakukan fungsi *read-list* dan *read-view* seperti yang dijelaskan pada Gambar 3.25. Khusus untuk *user* SP *maker*, dapat melakukan fungsi *create*, *update*, dan *delete* seperti yang dijelaskan pada Gambar 3.25 untuk *user* SP. Akan tetapi, ketiga fungsi tersebut juga sekaligus mengajukan permintaan untuk pembuatan, pembaruan, atau penghapusan *merchant*. Khusus untuk *user* SP *checker*, dapat menyetujui permintaan *merchant* berdasarkan *id* dari fungsi *approve* dan menolak permintaan *merchant* berdasarkan *id* dari fungsi *reject*.

Kemudian, berikut rancangan *activity diagram* terbaru yang dapat dilihat pada Gambar 3.34.

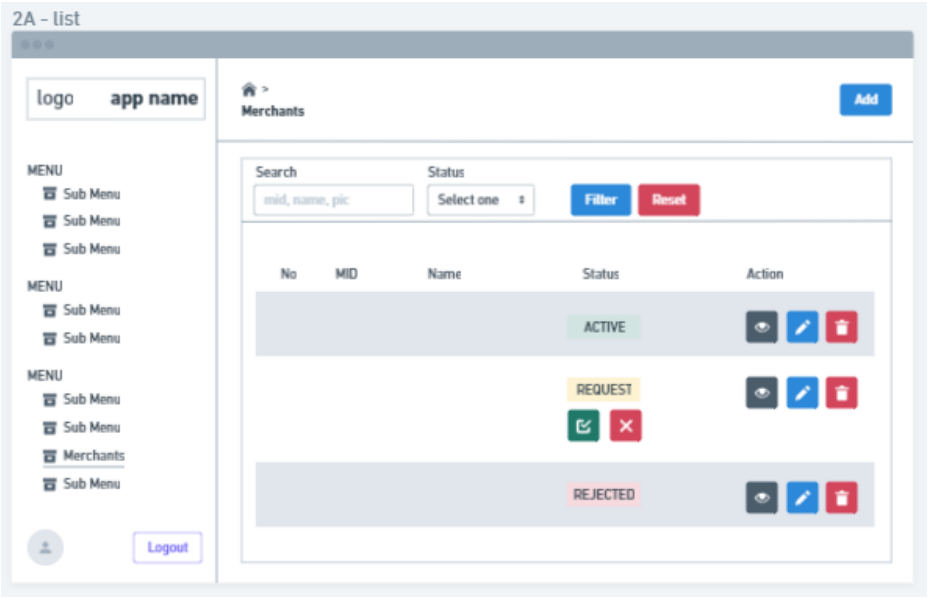



Gambar 3.34 *Activity Diagram* pada *Merchant* (Setelah *Review*)

Berdasarkan Gambar 3.34, pembaruan terjadi di setiap proses membuat, memperbarui, atau menghapus suatu *merchant*. Penjelasan alur proses terbaru dibagi menjadi tiga supaya lebih singkat. Pertama, jika SP *maker* membuat *merchant*, maka secara otomatis terbentuk pengajuan permintaan pembuatan dengan status berubah menjadi *request*. SP *checker* lalu akan mengonfirmasi apakah permintaan pembuatan disetujui atau ditolak. Jika disetujui, maka status berubah menjadi *active* dan dapat digunakan sebagai tujuan distribusi EDC dari SP. Jika ditolak, maka status berubah menjadi *rejected*. Kedua, alur prosesnya seperti yang pertama, tetapi dilakukan untuk memperbarui *merchant*. Ketiga, jika SP *maker* menghapus *merchant*, maka statusnya dicek terlebih dahulu. Jika status berupa *active* atau *rejected*, maka secara otomatis terbentuk pengajuan permintaan penghapusan dengan status berubah menjadi *request delete*. SP *checker* lalu akan mengonfirmasi apakah permintaan penghapusan disetujui atau ditolak. Jika disetujui, maka *merchant* dihapus dengan cara *soft delete*. Jika tidak disetujui,

maka status berubah menjadi *rejected*. Sementara itu, jika status berupa *request* dan nilai waktu pembaruan adalah *null*, maka *merchant* langsung dihapus dengan cara *soft delete*.

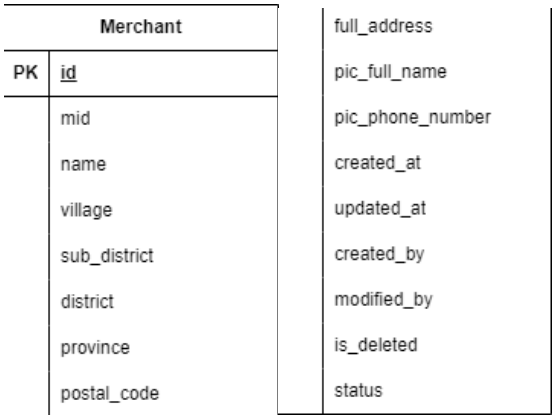
Kemudian, berikut rancangan *wireframe* terbaru yang dapat dilihat pada Gambar 3.35.




Gambar 3.35 *Wireframe* pada *Merchant* (Setelah *Review*)

Berdasarkan Gambar 3.35, pembaruan desain antarmuka hanya dilakukan pada halaman *list* yang memiliki penambahan kolom pada tabel berupa status dan penambahan kondisi filter berupa *dropdown* status. Selain itu, kolom status memiliki tombol centang dan silang jika status berupa *request* atau *request delete* dan *user* yang *login* adalah SP *checker*. Tombol tersebut untuk mengerjakan fungsi tertentu melalui *endpoint*. Tombol centang untuk *endpoint approve* dengan HTTP PATCH, sedangkan tombol silang untuk *endpoint reject* dengan HTTP PATCH.

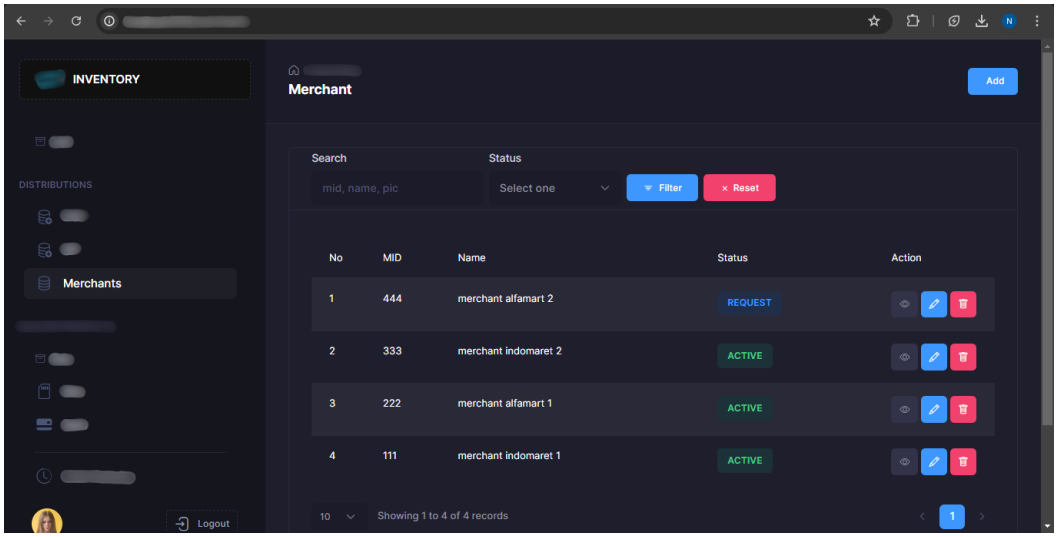
Terakhir, rancangan relasi antar tabel terbaru hanya memiliki pembaruan berupa adanya penambahan atribut status pada tabel *merchant*. Rancangan tersebut dapat dilihat pada Gambar 3.36.



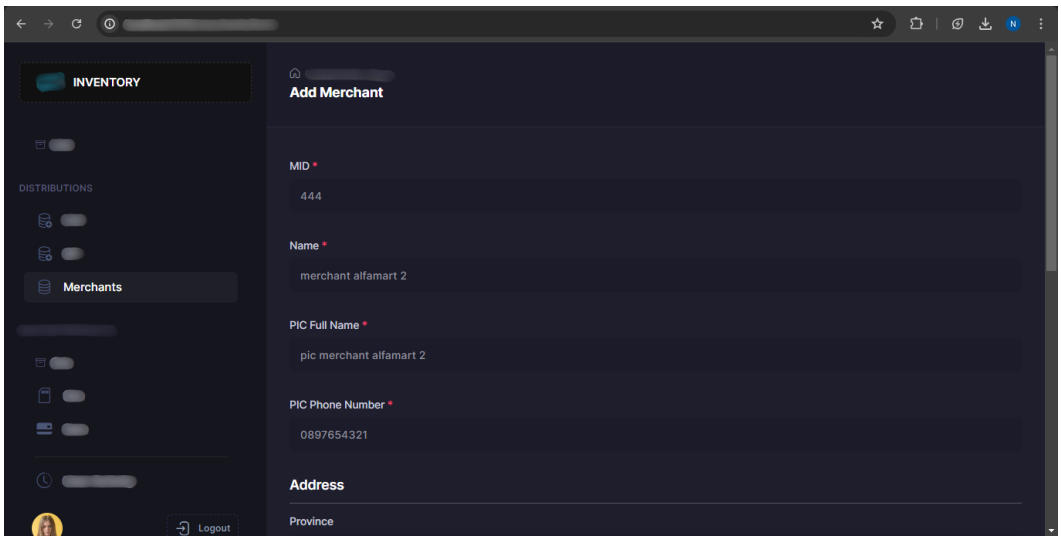

Gambar 3.36 Relasi Antar Tabel pada *Merchant* (Setelah *Review*)

Implementasi kode dilakukan setelah mengetahui dan memahami kembali berbagai hal yang terbaru, mulai dari kebutuhan fungsi hingga relasi antar tabel. Hasil implementasi dari sisi *back-end* dan *front-end* pada *review* fitur ini dapat dilihat mulai dari Gambar 3.37.

Pada Gambar 3.37 adalah halaman *list* yang menjalankan fungsi *read-list* sehingga menampilkan semua *merchant* dalam bentuk tabel untuk *user* SP *maker*. Terdapat *merchant* dengan MID 444 dan berstatus *request* yang didapatkan dari hasil pada Gambar 3.38. Gambar tersebut merupakan halaman *add-form* yang menjalankan fungsi *create* sehingga *merchant* dengan MID 444 berhasil dibuat oleh *user* SP *maker*. Pembuatan tersebut secara otomatis juga mengajukan permintaan pembuatan yang ditunjukkan dengan status *request* pada *merchant* tersebut.

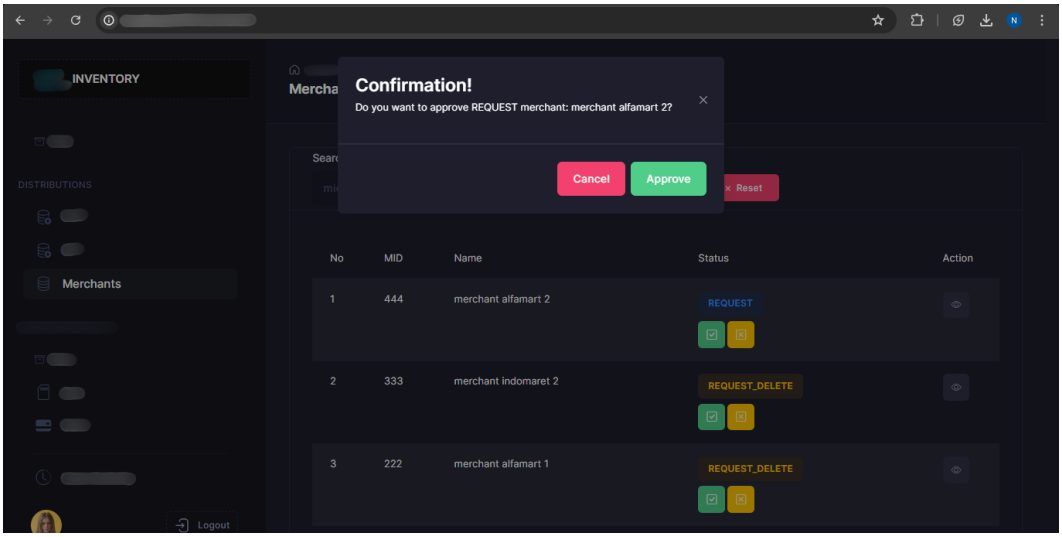



Gambar 3.37 Tampilan Halaman *List* pada *Merchant* (Setelah *Review*) (1)



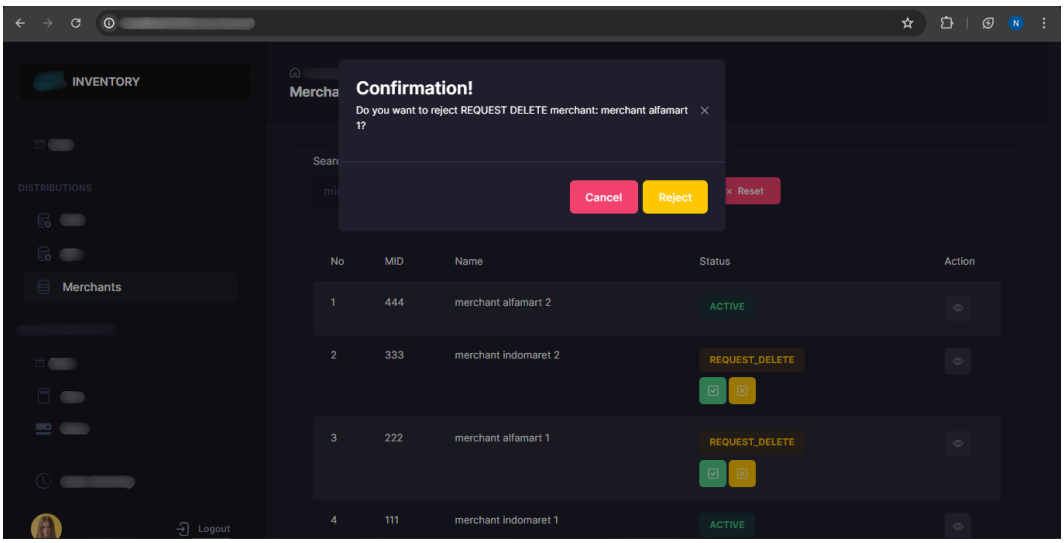

Gambar 3.38 Tampilan Halaman *Add-Form* pada *Merchant* (2)

Pada Gambar 3.39 adalah halaman *list* yang sama seperti pada Gambar 3.37. Akan tetapi, untuk *user* SP *checker*. Selain itu, terdapat *merchant* dengan MID 333 dan 222 yang statusnya sudah berubah. Dari yang sebelumnya pada Gambar 3.37 adalah *active* menjadi *request delete* dikarenakan *user* SP *maker* menekan tombol hapus untuk fungsi *delete* yang secara otomatis juga mengajukan permintaan penghapusan. Terdapat juga *pop up modal* yang muncul ketika tombol centang pada kolom status *merchant* dengan MID 444 dan berstatus *request* ditekan oleh *user* SP *checker*. Hal tersebut untuk menjalankan fungsi *approve* yang mengonfirmasi bahwa permintaan pembuatan akan disetujui.




Gambar 3.39 Tampilan Halaman *List* pada *Merchant* (Setelah *Review*) (2)

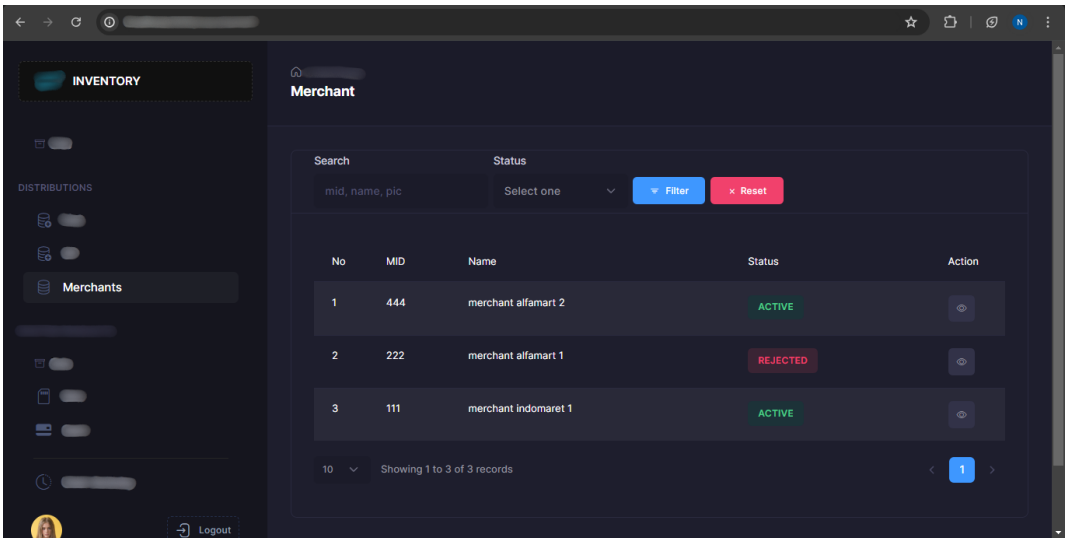
Pada Gambar 3.40 adalah lanjutan halaman pada Gambar 3.39 yang menampilkan bahwa *merchant* dengan MID 444 sudah berganti status menjadi *active*. *Merchant* tersebut adalah keberhasilan persetujuan pembuatan pada Gambar 3.39. Terdapat juga *pop up modal* yang muncul ketika tombol silang pada kolom status *merchant* dengan MID 222 dan berstatus *request delete* ditekan oleh *user* SP *checker*. Hal tersebut untuk menjalankan fungsi *reject* yang mengonfirmasi bahwa permintaan penghapusan akan ditolak.



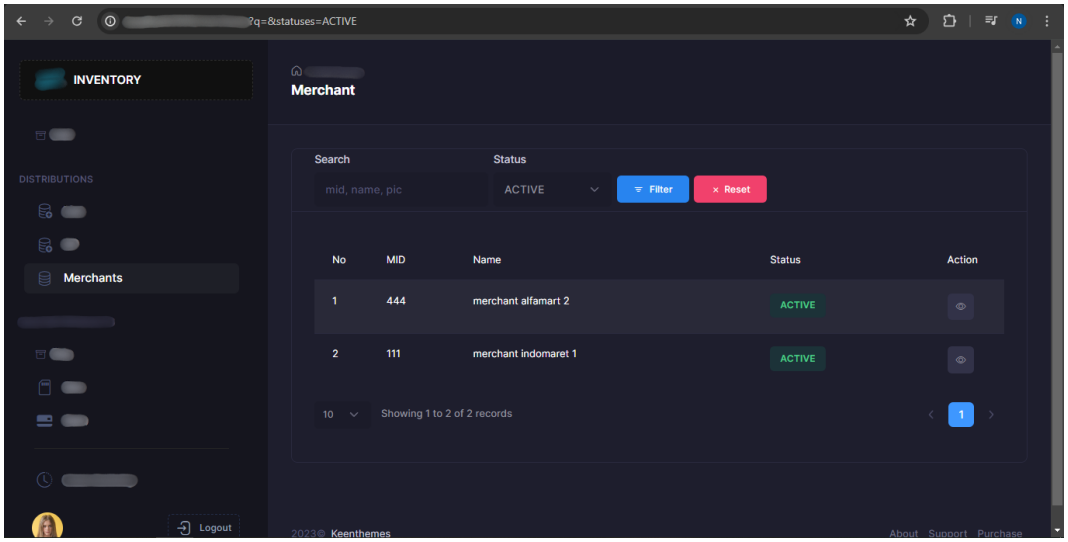

Gambar 3.40 Tampilan Halaman *List* pada *Merchant* (Setelah *Review*) (3)

Pada Gambar 3.41 adalah lanjutan halaman pada Gambar 3.40 yang menampilkan bahwa *merchant* dengan MID 222 sudah berganti status menjadi *rejected*. *Merchant* tersebut adalah keberhasilan penolakan penghapusan pada Gambar 3.40. Terdapat juga *merchant* dengan MID

333 sudah tidak lagi ditampilkan. Hal tersebut adalah keberhasilan persetujuan penghapusan pada Gambar 3.40 yang statusnya adalah *request delete*, tetapi proses *pop up modal*-nya tidak ditampilkan. Keberhasilan tersebut terjadi karena *user* SP *checker* menekan tombol centang pada kolom status *merchant* tersebut, dimana tombol centang menjalankan fungsi *approve* yang mengonfirmasi bahwa permintaan penghapusan akan disetujui. Kemudian, pada Gambar 3.42 adalah lanjutan halaman pada Gambar 3.41 yang menampilkan semua *merchant* dengan filter yang didasarkan pada kondisi status berupa *active*.




Gambar 3.41 Tampilan Halaman *List* pada *Merchant* (Setelah *Review*) (4)

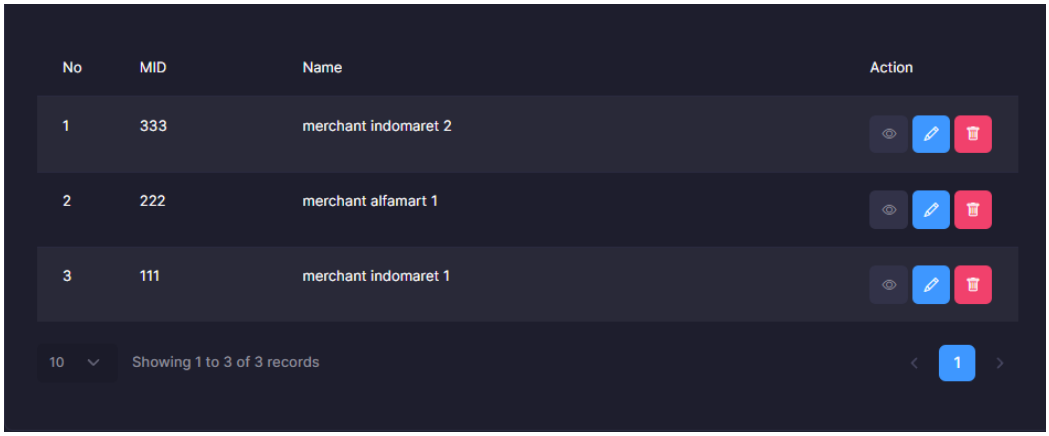



Gambar 3.42 Tampilan Halaman *List* pada *Merchant* (Setelah *Review*) (5)

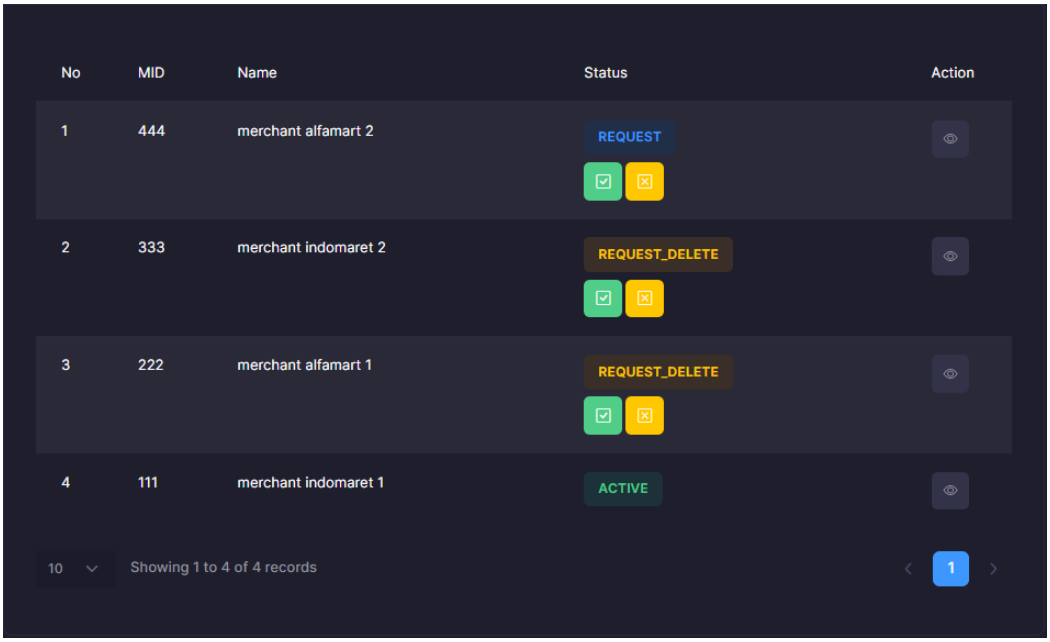
Setelah fungsi *approve* dan *reject* berhasil direalisasikan melalui implementasi ulang dari sisi *back-end* dan *front-end*, dilakukan pengujian ulang oleh penulis terlebih dahulu untuk

kemudian diuji kembali oleh *supervisor*. Hasil pengujian berjalan dengan baik dan sudah sesuai dengan *review* dari klien perusahaan. Salah satu perbandingan hasil sebelum *review* dan setelah *review* dapat dilihat pada Gambar 3.43 dengan *user* yang *login* adalah SP dan Gambar 3.44 dengan *user* yang *login* adalah SP *checker*.

Pada Gambar 3.43 terlihat bahwa setiap *merchant* tidak memiliki status apapun dan tidak memiliki kolom status pada tabel. Sementara itu, pada Gambar 3.44 terlihat bahwa setiap *merchant* memiliki status yang ditampilkan pada kolom status. Status didapatkan dari proses *checker maker* oleh *user* SP dan setiap status memiliki arti tersendiri seperti yang dijelaskan pada alur proses terbaru. Terdapat tombol centang dan silang pada kolom status jika status berupa *request* atau *request delete* dan *user* yang *login* adalah SP *checker* sehingga dapat mengerjakan *endpoint approve* dan *reject*.



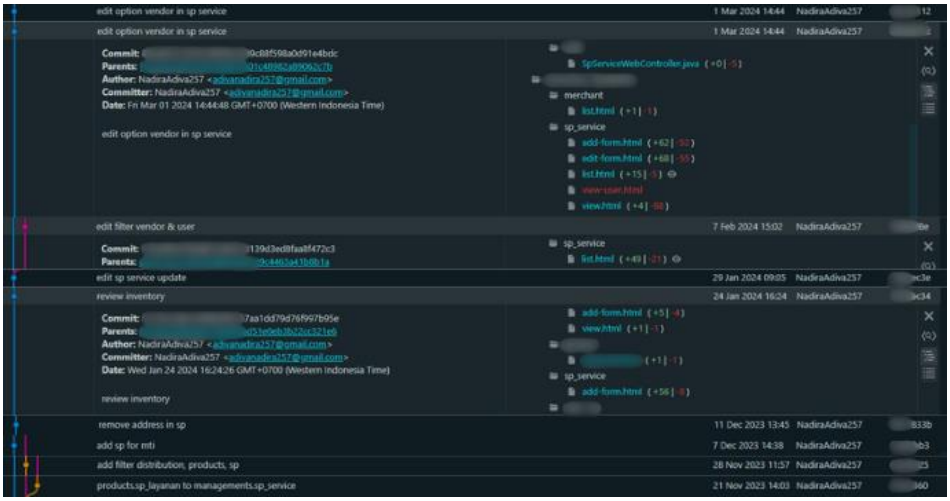

Gambar 3.43 Bukti Sebelum *Review* pada *Merchant*



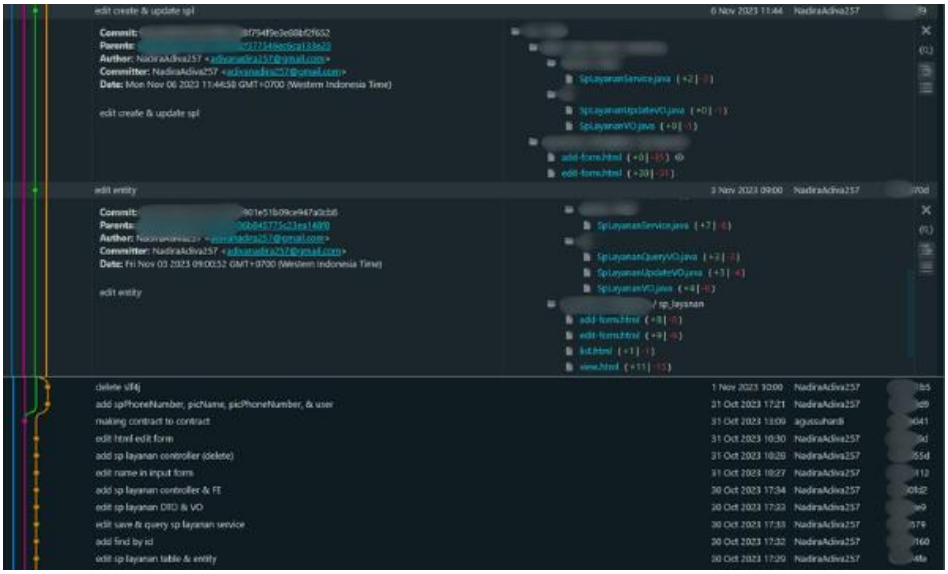

Gambar 3.44 Bukti Setelah *Review* pada *Merchant*

### 3.2.3 Pengembangan Fitur *Service Point* (SP)

Sebelum membahas pengembangan dari tahapan awal hingga akhir, berikut gabungan dari beberapa bukti pengerjaan yang terkait dengan fitur ini dan dapat dilihat pada Gambar 3.45 dan Gambar 3.46.



Gambar 3.45 Bukti Pengerjaan pada SP (1)



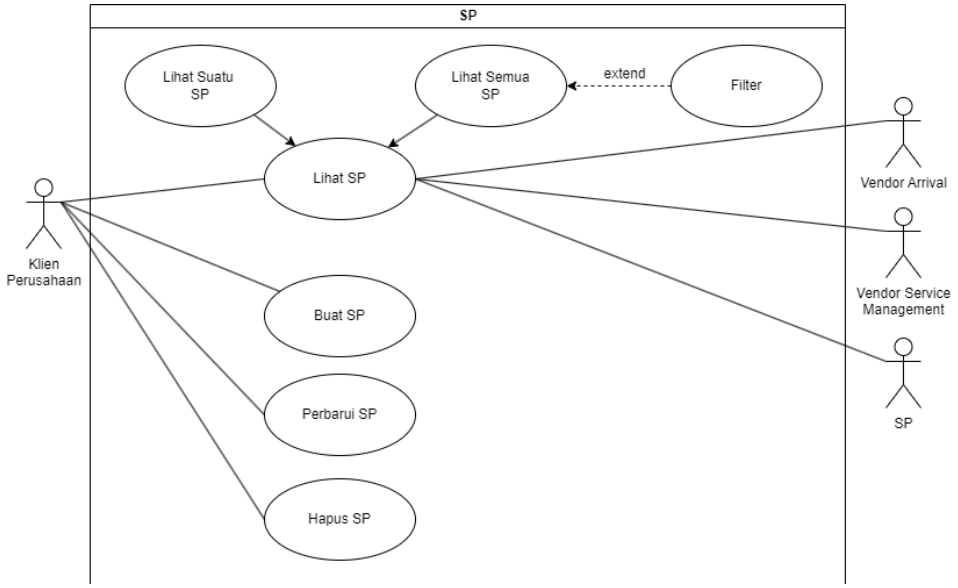
Gambar 3.46 Bukti Pengerjaan pada SP (2)

**Analisis Kebutuhan**

Fitur SP merupakan salah satu kebutuhan yang didapatkan melalui dokumen pertama yang disertakan oleh klien perusahaan kepada *developer*. Fitur ini mengelola entitas SP yang akan digunakan sebagai tujuan distribusi EDC dari SP dari vendor *arrival* atau vendor *service management* dan untuk membuat akun *user* SP. SP sendiri beroperasi di bawah vendor *arrival* atau vendor SP. Selanjutnya, setelah dokumen tersebut dianalisis, dibutuhkan lima fungsi dasar berupa CRUD. Proses dasar *back-end* pada fungsi CRUD tersebut, kecuali *read-list*, dibuatkan oleh rekan kerja untuk selanjutnya dikerjakan oleh penulis. Fungsi *read-list* pada penjelasan sebelumnya dikecualikan karena prosesnya dibuat oleh penulis.

**Desain Sistem**

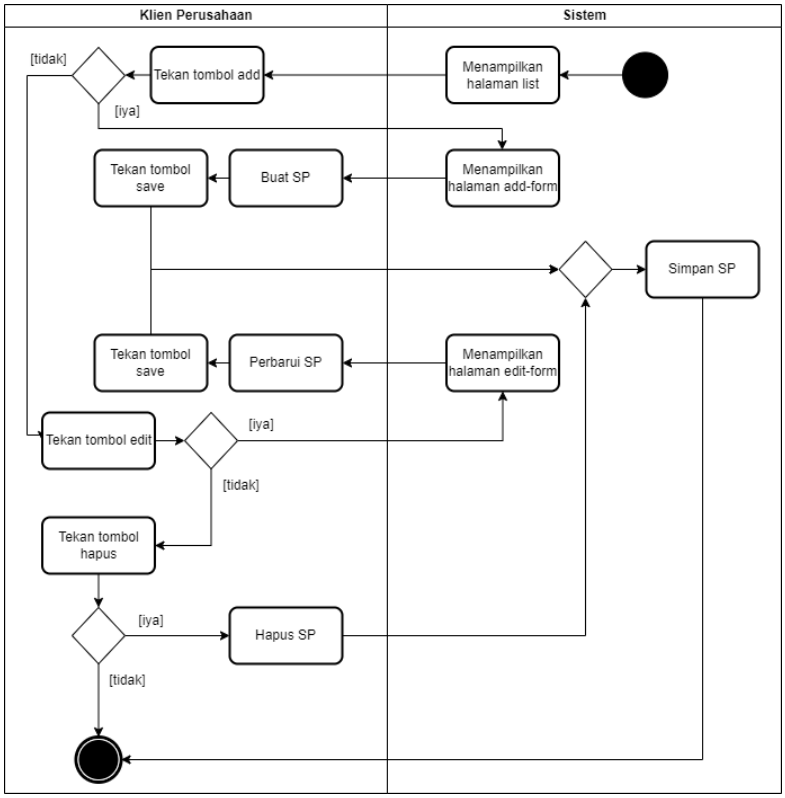
Terdapat empat rancangan dalam tahap desain sistem, yaitu *use case diagram*, *activity diagram*, *wireframe*, dan relasi antar tabel. Berikut rancangan *use case diagram* pada fitur ini yang dapat dilihat pada Gambar 3.47.




Gambar 3.47 *Use Case Diagram* pada SP (Sebelum *Review*)

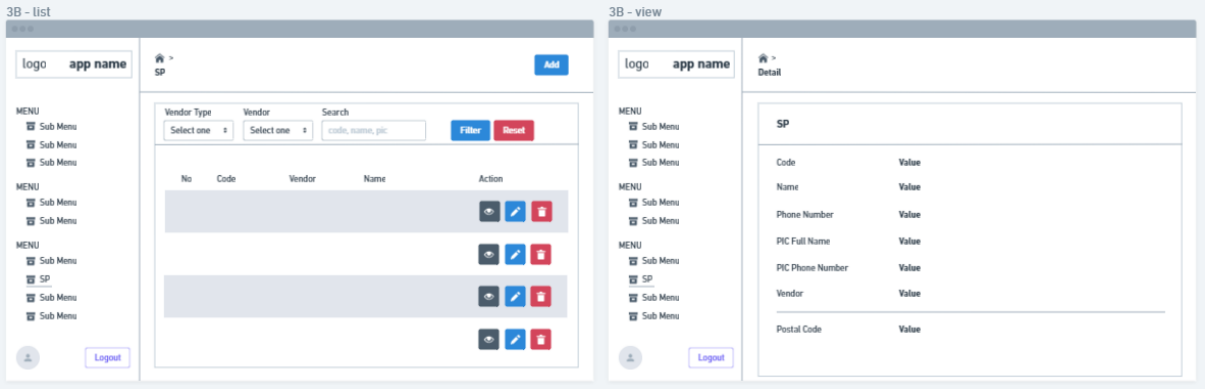
Berdasarkan Gambar 3.47, terdapat empat *user* yang dapat mengakses berbagai fungsi pada fitur ini, yaitu klien perusahaan, vendor *arrival*, vendor *service management*, dan SP. Semua *user* tersebut dapat melihat semua SP dengan dilengkapi filter dari fungsi *read-list* dan melihat suatu SP berdasarkan *id* dari fungsi *read-view*. Khusus untuk *user* klien perusahaan, dapat melakukan tiga hal, yaitu membuat SP dari fungsi *create*, memperbarui SP berdasarkan *id* dari fungsi *update*, dan menghapus SP berdasarkan *id* dengan cara *soft delete* dari fungsi *delete*.

Kemudian, rancangan *activity diagram* untuk alur proses pada fitur ini dapat dijelaskan secara singkat karena fokusnya hanya untuk pengelolaan entitas SP. Klien perusahaan dapat membuat, memperbarui, atau menghapus suatu SP. Setelah melakukan salah satu hal tersebut, SP akan disimpan pada *database* dan alur proses akan berakhir. Rancangan tersebut dapat dilihat pada Gambar 3.48.

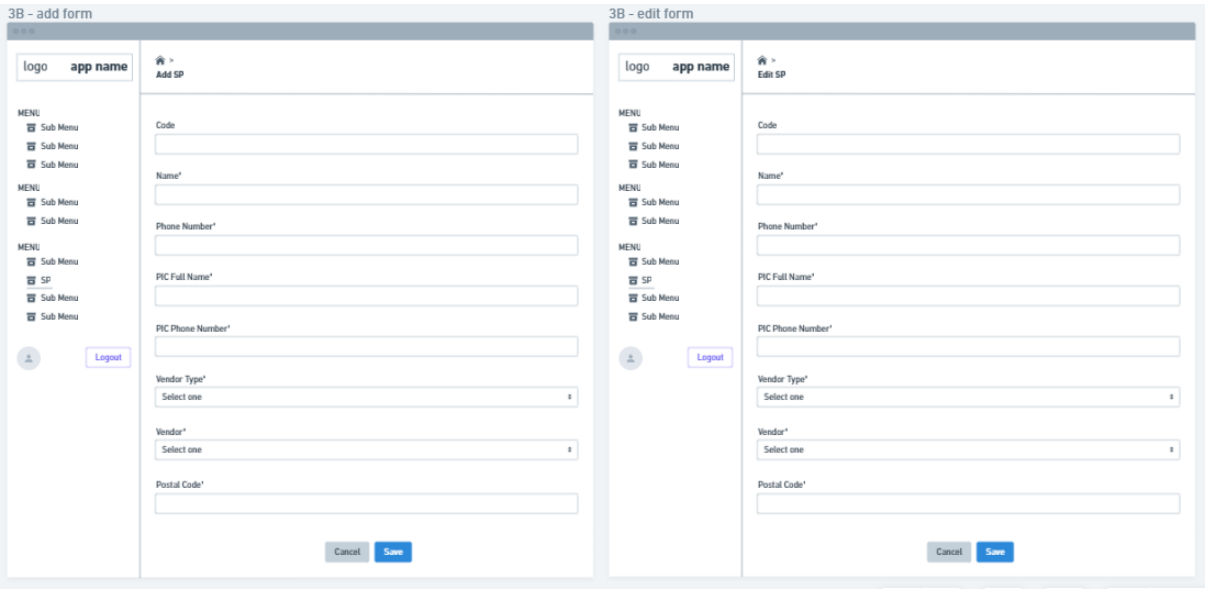



Gambar 3.48 *Activity Diagram* pada SP (Sebelum *Review*)

Kemudian, berikut rancangan *wireframe* pada fitur ini yang dapat dilihat pada Gambar 3.49 dan Gambar 3.50.




Gambar 3.49 *Wireframe* pada SP (Sebelum *Review*) (1)



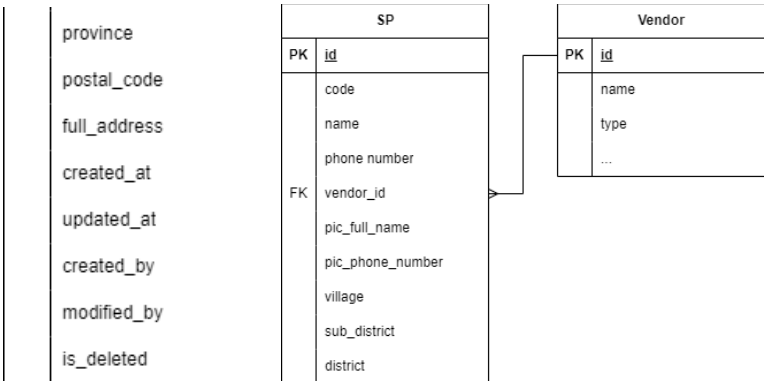

Gambar 3.50 *Wireframe* pada SP (Sebelum *Review*) (2)

Berdasarkan Gambar 3.49 dan Gambar 3.50, desain antarmuka pada fitur ini dapat dijelaskan secara singkat. Terdapat empat halaman, yaitu *list*, *view*, *add-form*, dan *edit-form*. Pada halaman *list* terdapat tabel untuk menampilkan semua data, tombol *add* ke halaman *add-form*, dan kolom *action* yang memiliki tiga tombol ke halaman *view*, ke halaman *edit-form*, serta untuk fungsi *delete*. Selain itu, pada halaman *list*, *add-form*, dan *edit-form* terdapat *dropdown* untuk menampilkan pilihan. Pembuatan tabel dan *dropdown* akan menggunakan teknologi yang sama dengan penjelasan pada Gambar 3.6, Gambar 3.7, dan Gambar 3.8.

Dapat disimpulkan juga bahwa setiap halaman mengerjakan fungsi tertentu melalui *endpoint* yang dimiliki. Pertama, halaman *list* memiliki *endpoint read-list* dengan HTTP GET dan *endpoint delete* dengan HTTP DELETE. Kedua, halaman *view* memiliki *endpoint read-view* dengan HTTP GET. Ketiga, halaman *add-form* memiliki *endpoint create* dengan HTTP

POST. Keempat, halaman *edit-form* memiliki *endpoint read-view* juga dan *endpoint update* dengan HTTP PUT.

Terakhir, berikut rancangan relasi antar tabel pada fitur ini yang dapat dilihat pada Gambar 3.51.




Gambar 3.51 Relasi Antar Tabel pada SP (Sebelum *Review*)

Berdasarkan Gambar 3.51, terdapat rancangan tabel SP pada *database* dengan beberapa atribut yang dimiliki, seperti *id* sebagai *primary key*, *code*, *name*, *phone number*, vendor *id*, PIC *full* name, PIC *phone number*, dan *postal code*. Tabel SP dikelompokkan ke dalam skema *management* dan memiliki relasi *many to one* dengan tabel vendor. Hal tersebut ditunjukkan dengan *foreign key* pada SP (vendor *id*) yang merujuk ke *primary key* pada vendor (*id*) sehingga setiap SP pasti memiliki satu vendor (baik bertipe *arrival* maupun bertipe SP) dan setiap vendor (baik bertipe *arrival* maupun bertipe SP) dapat memiliki satu atau banyak SP. Hal tersebut sesuai dengan pernyataan pada analisis kebutuhan bahwa SP beroperasi di bawah salah satu vendor tersebut.

**Implementasi**

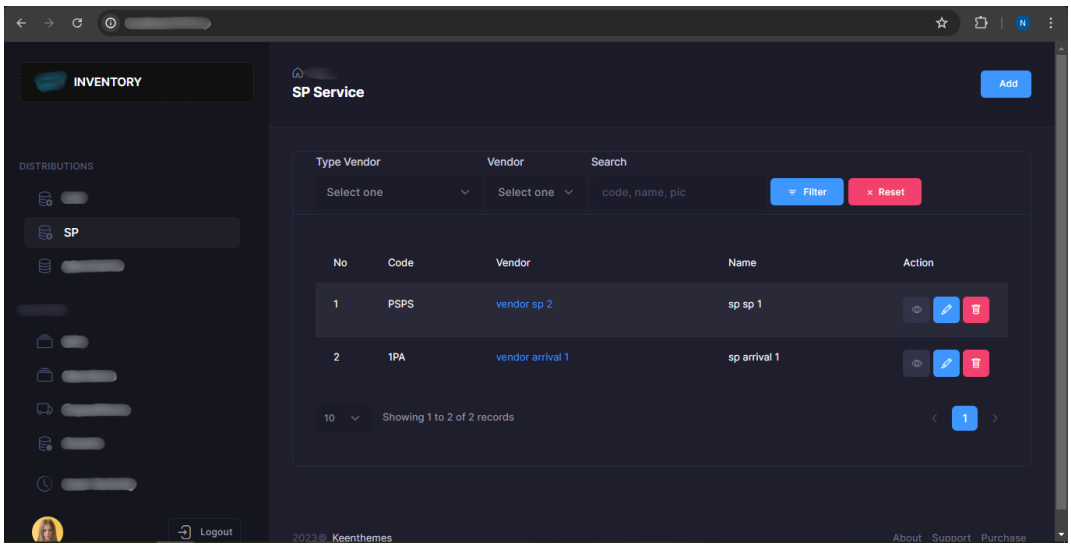
Implementasi kode dilakukan setelah mengetahui dan memahami kebutuhan fungsi, hak akses *user* melalui *use case diagram*, alur proses melalui *activity diagram*, desain antarmuka melalui *wireframe*, dan relasi antar tabel. Dengan begitu, setiap fungsi pada fitur SP dapat terealisasikan. Terdapat tiga tambahan informasi, yaitu fungsi *create* dan *update* membutuhkan VO, fungsi *read-view* membutuhkan DTO, dan fungsi *read-list* membutuhkan keduanya.

Kebutuhan VO pada fungsi *read-list* digunakan untuk proses filter. Filter untuk mencari SP yang memenuhi satu atau beberapa kondisi dari lima kondisi yang disediakan, yaitu *name*, *phone number*, PIC *full name*, PIC *phone number*, dan vendor. Adanya filter berdasarkan vendor bertipe *arrival* atau bertipe SP pada fitur SP juga digunakan untuk otomatisasi

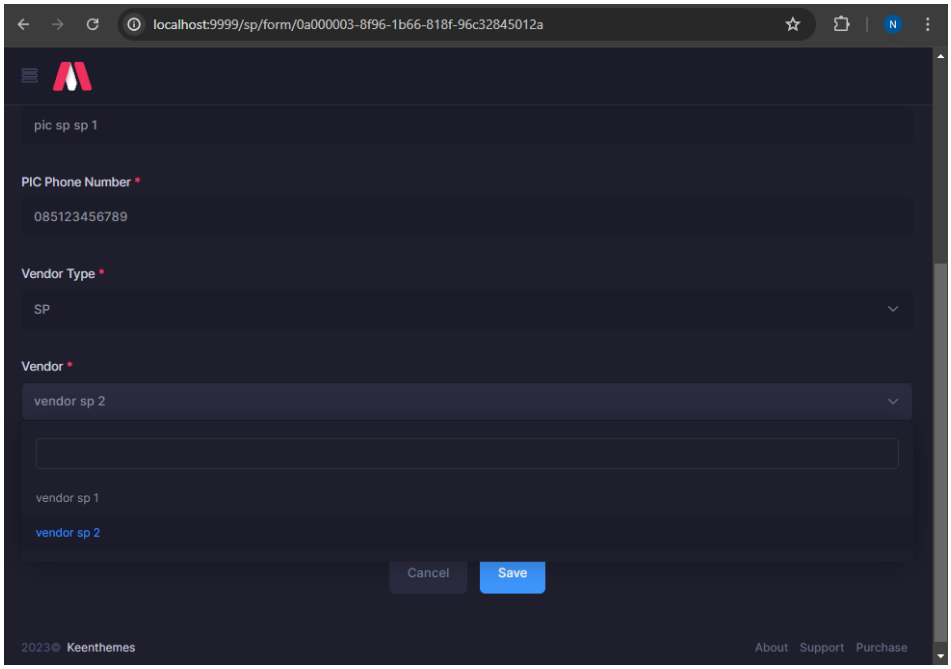
penyesuaian pengambilan SP berdasarkan *user* yang *login*. Jika *user* yang *login* adalah klien perusahaan atau vendor *service management*, maka dapat mengambil semua SP dari vendor manapun. Jika *user* yang *login* adalah vendor *arrival* atau SP, maka hanya dapat mengambil SP yang ditujukan kepada vendor mereka.

Jika implementasi setiap fungsi di sisi *back-end* sudah selesai, maka penulis membuat berbagai halaman di sisi *front-end*. Pembuatan halaman mengikuti rancangan desain antarmuka dengan melibatkan *endpoint* dari setiap fungsi. Penjelasan mengenai hal tersebut dapat dilihat pada Gambar 3.49 dan Gambar 3.50. Hasil implementasi dari kedua sisi tersebut pada fitur ini dapat dilihat mulai dari Gambar 3.52.

Pada Gambar 3.52 adalah halaman *list* yang menjalankan fungsi *read-list* sehingga menampilkan semua SP dalam bentuk tabel pada *user* klien perusahaan. Terdapat SP dengan nama sp sp 1 yang beroperasi di bawah vendor sp 2. Pada Gambar 3.53 adalah halaman *edit-form* yang menjalankan fungsi *read-view* sehingga menampilkan detail SP berdasarkan *id* yang dimiliki oleh SP dengan nama sp sp 1. Salah satu detailnya adalah bahwa SP tersebut beroperasi di bawah vendor sp 2 yang bertipe SP. Hal tersebut diketahui dari pilihan *default* pada *dropdown* tipe vendor dan *dropdown vendor* (nama vendor) yang disesuaikan dengan data *read-view* dari *database*. Kedua *dropdown* tersebut saling berhubungan, dimana setiap tipe vendor dapat memiliki satu atau banyak vendor. Halaman *edit-form* juga menjalankan fungsi *update* untuk semua atribut SP pada halaman tersebut, kecuali *code*.

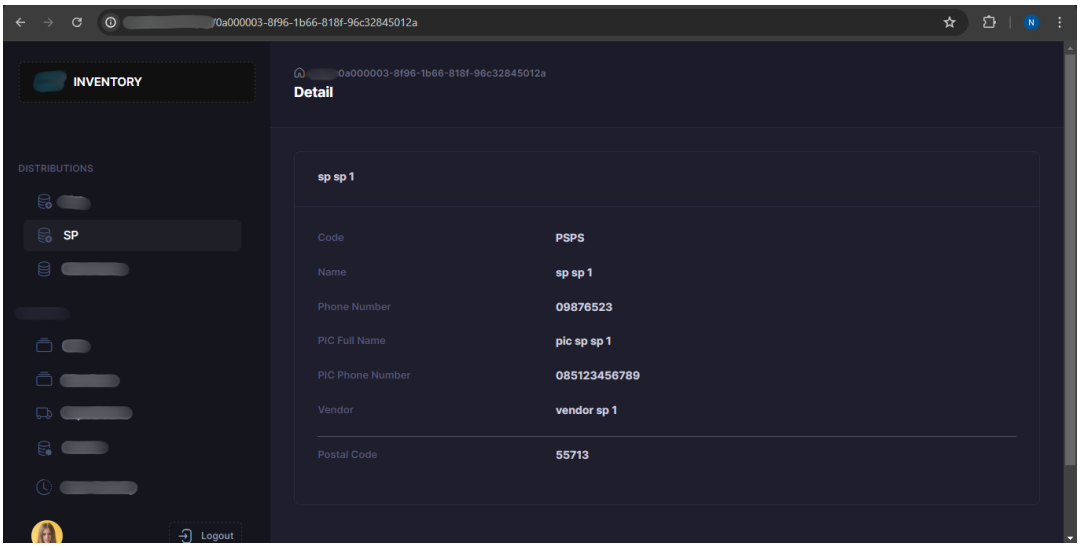



Gambar 3.52 Tampilan Halaman *List* pada SP (Sebelum *Review*)




Gambar 3.53 Tampilan Halaman *Edit-Form* pada SP (Sebelum *Review*)

Pada Gambar 3.54 adalah halaman *view* yang menjalankan fungsi *read-view* juga sehingga menampilkan detail SP berdasarkan *id* yang dimiliki oleh SP dengan nama sp sp 1. Terlihat bahwa SP tersebut kini beroperasi di bawah vendor sp 1. Hal tersebut menandakan bahwa pada Gambar 3.53, vendor berhasil diperbarui dari vendor sp 2 yang bertipe SP ke vendor sp 1 yang bertipe SP juga oleh *user* klien perusahaan melalui dua pilihan pada *dropdown* vendor yang bertipe SP di Gambar 3.53.



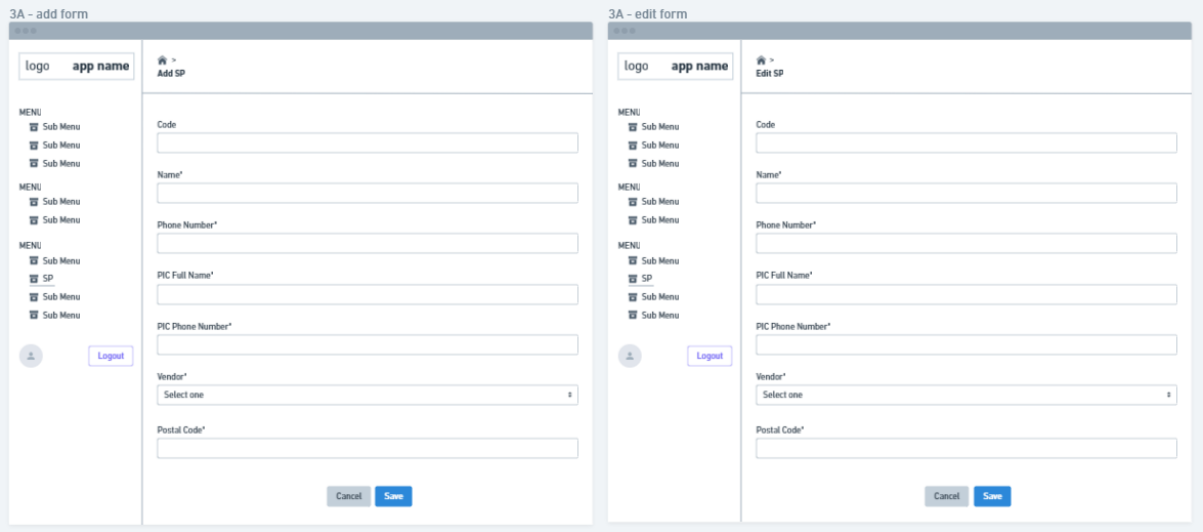

Gambar 3.54 Tampilan Halaman *View* pada SP

**Pengujian**

Setelah semua fungsi pada fitur SP berhasil direalisasikan melalui implementasi dari sisi *back-end* dan *front-end*, dilakukan pengujian menyeluruh yang masih sebatas pada lingkup *developer*. Pengujian dilakukan oleh penulis terlebih dahulu untuk kemudian diuji kembali oleh *supervisor*. Secara umum, hasil pengujian berjalan dengan baik dan sudah sesuai dengan kebutuhan yang ditentukan.

Akan tetapi, terdapat beberapa *review* untuk perbaikan yang didapatkan dari hasil pengujian kepada klien perusahaan seperti halnya pada fitur *merchant*. *Review* untuk perbaikan dari klien perusahaan juga sama seperti pada fitur *merchant*, yaitu dibutuhkan validasi atau konfirmasi bahwa pembuatan, pembaruan, atau penghapusan suatu data entitas sudah tepat dan sesuai melalui *privilege checker maker* pada *user*. Kebutuhan tersebut mengharuskan proses berulang dari tahapan analisis hingga pengujian. Keseluruhan prosesnya sama seperti pada fitur *merchant*, tetapi beroperasinya untuk entitas SP.

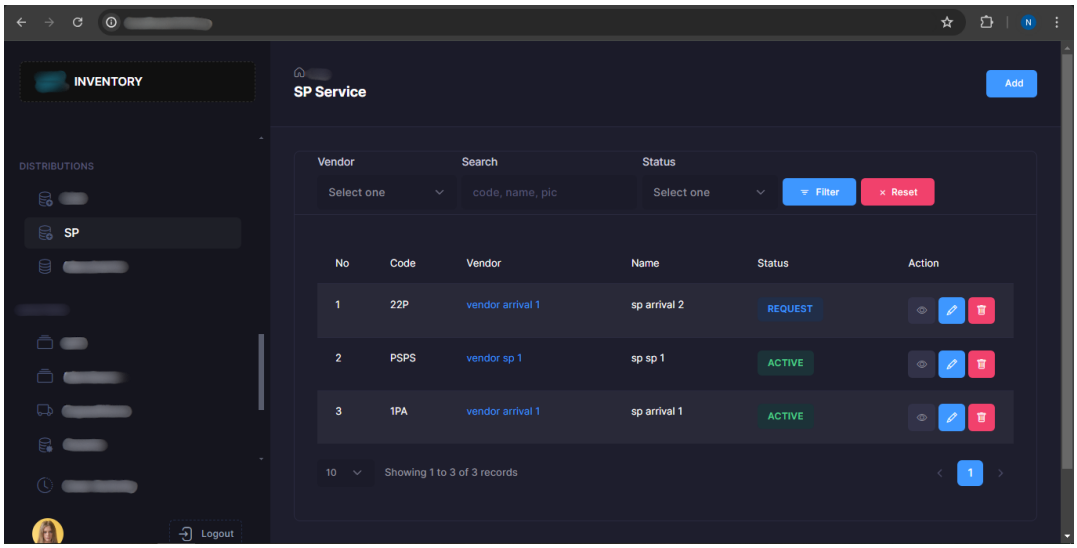
Selain itu, dari *supervisor* juga memberikan beberapa *review* untuk perbaikan. Salah satunya adalah penghapusan *dropdown* tipe vendor dan pembaruan *text* pada pilihan *dropdown* vendor untuk menyederhanakan tampilan yang dapat meningkatkan efisiensi *user*. Hal tersebut dibuktikan melalui pilihan *dropdown* vendor yang akan diambil dari gabungan tipe vendor (baik *arrival* maupun SP) sehingga penulisan *text* berubah dari yang awalnya hanya nama vendor saja menjadi tipe vendor dan nama vendor. Oleh karena itu, rancangan *wireframe* diperbarui pada halaman *list*, *add-form*, dan *edit-form* dengan menghapus *dropdown* tipe vendor sehingga hanya menyisakan *dropdown* vendor. Rancangan untuk halaman *add-form* dan *edit-form* tersebut dapat dilihat pada Gambar 3.55.



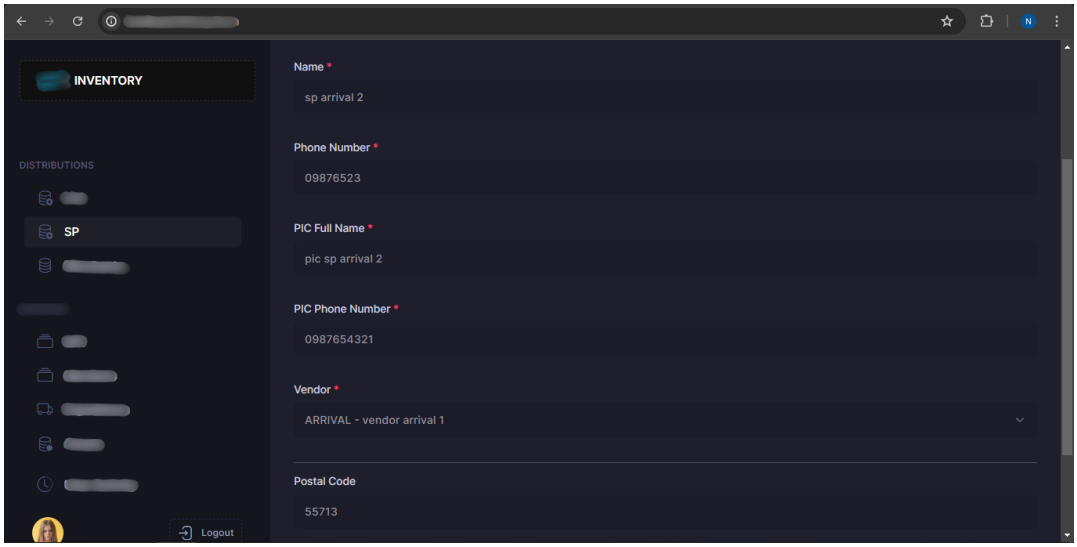

Gambar 3.55 *Wireframe* pada SP (Setelah *Review*)

Implementasi kode dilakukan setelah mengetahui dan memahami kembali berbagai hal yang terbaru terkait perbaikan dari *review supervisor* dengan segala prosesnya. Kemudian, hasil implementasi dari sisi *back-end* dan *front-end* pada gabungan kedua *review* tersebut dapat dilihat mulai dari Gambar 3.56.

Pada Gambar 3.56 adalah halaman *list* yang menjalankan fungsi *read-list* sehingga menampilkan semua SP dari vendor manapun ke dalam bentuk tabel karena *user* yang *login* adalah klien perusahaan *maker*. Terdapat SP dengan nama sp *arrival* 2 yang beroperasi di bawah vendor *arrival* 1 dan berstatus *request* yang didapatkan dari hasil pada Gambar 3.57. Gambar tersebut merupakan halaman *add-form* yang menjalankan fungsi *create* sehingga SP dengan nama sp *arrival* 2 yang beroperasi di bawah vendor *arrival* 1 berhasil dibuat oleh *user* klien perusahaan *maker*. Pembuatan tersebut secara otomatis juga mengajukan permintaan pembuatan yang ditunjukkan dengan status *request* pada SP tersebut.

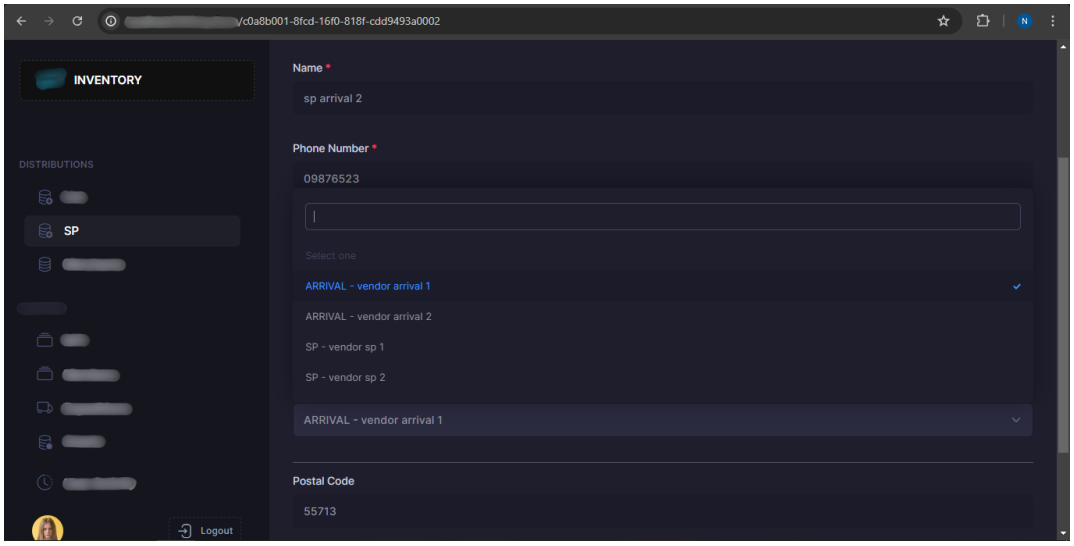



Gambar 3.56 Tampilan Halaman *List* pada SP (Setelah *Review*) (1)



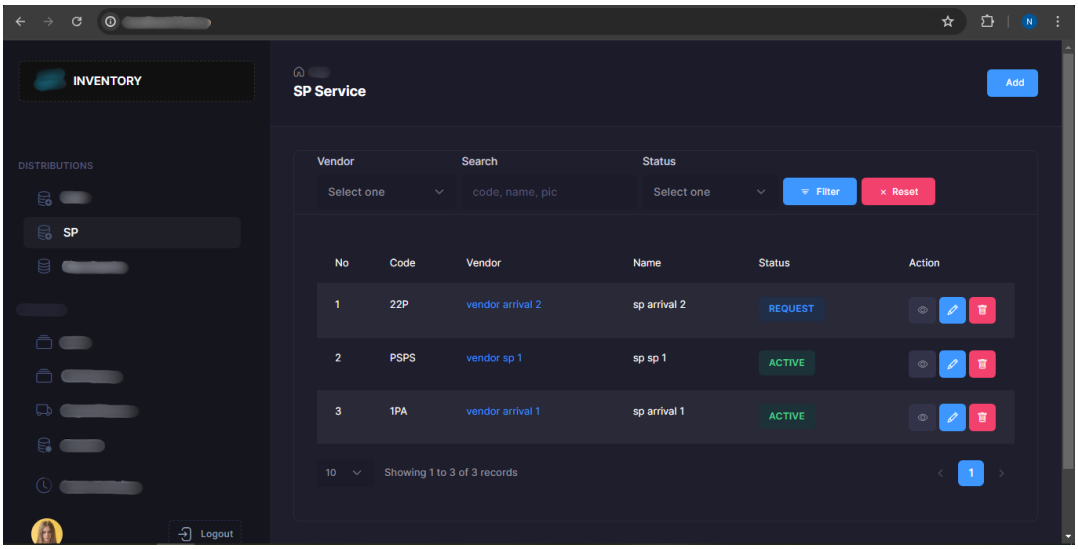

Gambar 3.57 Tampilan Halaman *Add-Form* pada SP (Setelah *Review*)

Pada Gambar 3.58 adalah halaman *edit-form* yang menjalankan fungsi *read-view* sehingga menampilkan detail SP berdasarkan *id* yang dimiliki oleh SP dengan nama sp *arrival* 2. Salah satu detailnya adalah bahwa SP tersebut beroperasi di bawah vendor *arrival* 1 yang bertipe *arrival*. Hal tersebut diketahui dari pilihan *default* pada *dropdown vendor* yang disesuaikan dengan data *read-view* dari *database*. Pilihan pada *dropdown* vendor tersebut diambil dari gabungan tipe vendor (baik *arrival* maupun SP) dengan *text* yang bertuliskan tipe vendor dan nama vendor. Halaman *edit-form* juga menjalankan fungsi *update* untuk semua atribut SP pada halaman tersebut, kecuali *code*.



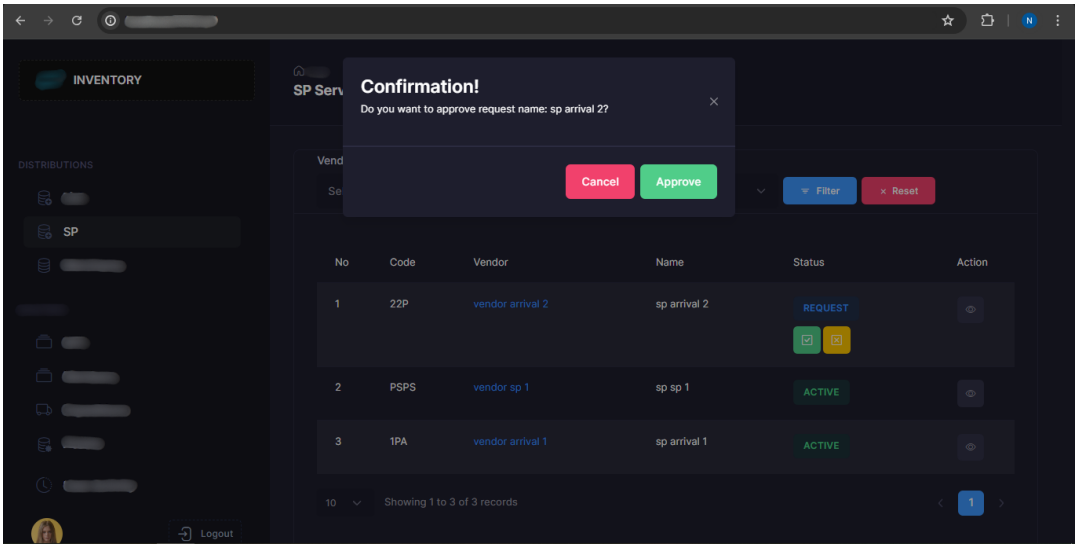

Gambar 3.58 Tampilan Halaman *Edit-Form* pada SP (Setelah *Review*)

Pada Gambar 3.59 adalah halaman *list* yang sama seperti pada Gambar 3.56. Akan tetapi, terlihat bahwa SP dengan nama sp *arrival* 2 kini beroperasi di bawah vendor *arrival* 2. Hal tersebut menandakan bahwa pada Gambar 3.58, vendor berhasil diperbarui dari vendor *arrival* 1 yang bertipe *arrival* ke vendor *arrival* 2 yang bertipe *arrival* juga oleh *user* klien perusahaan *maker* melalui empat pilihan pada *dropdown vendor* di Gambar 3.58. Pembaruan tersebut secara otomatis mengajukan permintaan pembaruan yang ditunjukkan dengan status *request* pada SP tersebut.



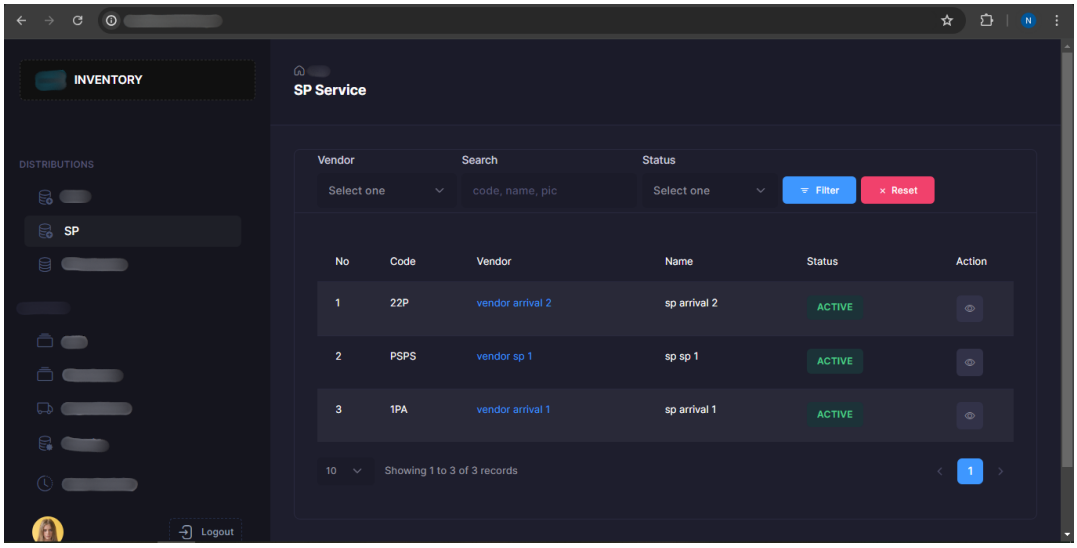

Gambar 3.59 Tampilan Halaman *List* pada SP (Setelah *Review*) (2)

Pada Gambar 3.60 adalah halaman *list* yang sama seperti pada Gambar 3.59. Akan tetapi, *user* yang *login* adalah klien perusahaan *checker*. Terdapat *pop up modal* yang muncul ketika tombol centang pada kolom status SP dengan nama sp *arrival* 2 yang beroperasi di bawah vendor *arrival* 2 dan berstatus *request* ditekan oleh *user* klien perusahaan *checker*. Hal tersebut untuk menjalankan fungsi *approve* yang mengonfirmasi bahwa permintaan pembaruan akan disetujui.

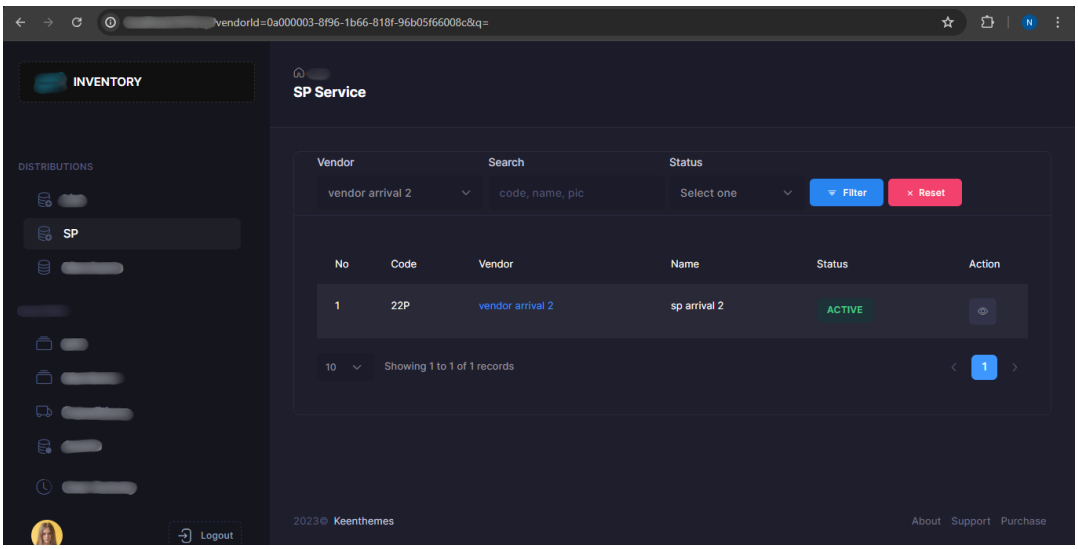



Gambar 3.60 Tampilan Halaman *List* pada SP (Setelah *Review*) (3)

Pada Gambar 3.61 adalah lanjutan halaman pada Gambar 3.60 yang menampilkan bahwa SP dengan nama sp *arrival* 2 yang beroperasi di bawah vendor *arrival* 2 sudah berganti status menjadi *active*. SP tersebut adalah keberhasilan persetujuan pembaruan pada Gambar 3.60. Pada Gambar 3.62 adalah lanjutan halaman pada Gambar 3.61 yang menampilkan semua SP dengan filter yang didasarkan pada vendor berupa vendor *arrival* 2.




Gambar 3.61 Tampilan Halaman *List* pada SP (Setelah *Review*) (4)

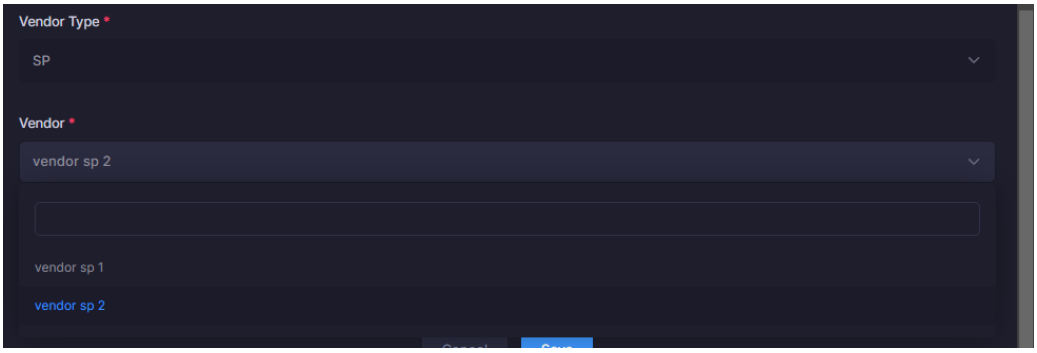



Gambar 3.62 Tampilan Halaman *List* pada SP (Setelah *Review*) (5)

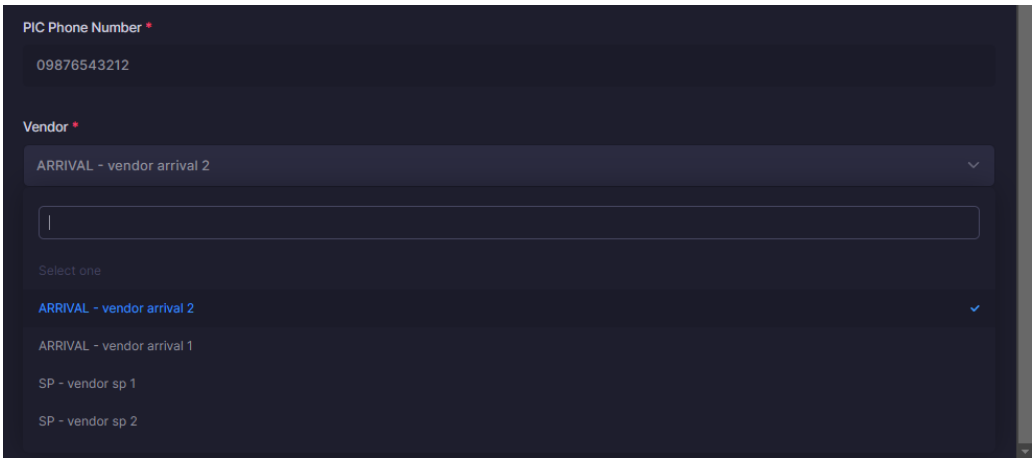
Setelah penghapusan *dropdown* tipe vendor dan pembaruan *text* pada pilihan *dropdown* vendor berhasil direalisasikan melalui segala proses implementasi, dilakukan pengujian ulang oleh penulis terlebih dahulu untuk kemudian diuji kembali oleh *supervisor*. Hasil pengujian berjalan dengan baik dan sudah sesuai dengan *review* dari *supervisor*. Perbandingan hasil sebelum *review* dan setelah *review* dapat dilihat pada Gambar 3.63 dan Gambar 3.64 dengan *user* yang *login* adalah klien perusahaan *maker*.

Pada Gambar 3.63 terlihat bahwa masih ada *dropdown* tipe vendor dan *dropdown* vendor (nama vendor). Keduanya saling berhubungan, dimana setiap tipe vendor dapat memiliki satu atau banyak vendor. Terdapat dua pilihan pada *dropdown* vendor yang bertipe SP. Sementara

itu, pada Gambar 3.64 terlihat bahwa sudah tidak ada *dropdown* tipe vendor sehingga hanya menyisakan *dropdown* vendor. Pilihan pada *dropdown* vendor tersebut diambil dari gabungan tipe vendor (baik *arrival* maupun SP) dengan *text* yang bertuliskan tipe vendor dan nama vendor. Terdapat empat pilihan pada *dropdown* vendor tersebut.




Gambar 3.63 Bukti Sebelum *Review* pada SP




Gambar 3.64 Bukti Setelah *Review* pada SP

## 3.3 Status Proyek

Status proyek ini ketika penulis selesai menjalankan kegiatan magang masih berada di tahapan implementasi dan pengujian untuk beberapa fitur yang masih perlu dikerjakan oleh *developer*, seperti fitur *member*. Semua fitur nantinya juga perlu diuji oleh *quality engineer* yang dapat menerapkan suatu metode pengujian, seperti *black-box*. Maka dari itu, proyek ini belum dapat digunakan secara langsung oleh berbagai *user* entitas yang terlibat dan belum dapat mencapai tahap akhir berupa evaluasi dan pemeliharaan.

# BAB IV
# REFLEKSI PELAKSANAAN MAGANG

## 4.1 Relevansi Akademik

Berdasarkan hasil teori-teori yang telah dijelaskan pada bab ke-2 (landasan teori dan tinjauan pustaka) menunjukkan adanya kesenjangan atau ketidaksesuaian dengan hasil yang didapatkan selama pelaksanaan di tempat magang, yaitu pada pengembangan metode *waterfall*. Terdapat dua teori yang bertolak belakang dengan pelaksanaan magang sehingga membuat kesenjangan. Pertama, setiap tahapan pada *waterfall* yang telah dilewati seharusnya tidak dapat diulang (Wahid, 2020). Kedua, metode *waterfall* cocok digunakan untuk pengembangan sistem yang berpeluang kecil mengalami perubahan (Solehudin et al., 2023).

Kesenjangan dengan teori yang dinyatakan oleh Wahid tersebut dibuktikan melalui salah satu *review* untuk perbaikan dari klien perusahaan ketika tahap pengujian pada fitur *merchant* dan SP dilakukan, yaitu dibutuhkan validasi atau konfirmasi bahwa pembuatan, pembaruan, atau penghapusan suatu data entitas sudah tepat dan sesuai melalui *privilege checker maker* pada *user*. Penambahan kebutuhan tersebut mengharuskan pengulangan proses dari tahap analisis kebutuhan hingga pengujian. Lalu, kesenjangan dengan teori yang dinyatakan oleh Solehudin et al., tersebut dibuktikan melalui dokumen kedua (pembaruan) yang disertakan kembali oleh klien perusahaan kepada *developer* ketika tahap implementasi pada fitur lain dilakukan. Dokumen tersebut berisikan pembaruan alur proses dari dokumen pertama dan kebutuhan beberapa fitur terbaru, salah satunya adalah fitur pengembalian EDC yang rusak. Proses penambahan kebutuhan fitur tersebut juga mengharuskan dimulai dari tahap analisis kebutuhan hingga pengujian.

Akibat dari kedua kesenjangan tersebut membuat target waktu proyek menjadi terlambat karena pengulangan dari tahap analisis kebutuhan hingga pengujian memakan waktu. Selain itu, membuat manajemen proyek menjadi lebih kompleks karena menyebabkan kebingungan untuk memprioritaskan antara fitur lain yang sudah masuk tahap implementasi dan fitur baru yang dimulai dari tahap analisis kebutuhan.

Selain itu, terdapat satu teori lagi yang berbeda dengan pelaksanaan magang, yaitu bahwa pada tahapan desain sistem biasanya merancang *use case diagram* (UCD), *activity diagram, entity relationship diagram* (ERD), hingga *prototype user interface* (Hermansyah et al., 2023). Sebelumnya, semua rancangan tersebut telah tersedia pada bab ke-3 (pelaksanaan magang) untuk kebutuhan laporan akhir. Namun, hal tersebut berbeda dengan realita penulis selama

melaksanakan magang. Perbedaan tersebut akhirnya membuat kesenjangan dengan teori yang dinyatakan oleh Hermansyah et al., yang dapat dibuktikan melalui tiga hal.

Pertama, tidak ada UCD, tetapi interaksi antara aktor (*user*) dan *use case* (fungsi spesifik pada fitur dalam sistem) tetap ada yang didapatkan dari analisis *developer* melalui gabungan fitur utama, peran umum *user*, dan alur proses. Kedua, tidak ada *activity diagram* dan ERD, tetapi alur proses dan rancangan *database* tersebut tetap ada dalam bentuk model tersendiri dan didapatkan dari kedua dokumen yang ikut disertakan oleh klien perusahaan kepada *developer*. Ketiga, tidak ada *prototype user interface* karena desain untuk tampilan halaman web langsung didapatkan dari *template* web desain Metronic.

Secara umum, akibat dari ketiga kesenjangan tersebut adalah pengembangan menjadi sedikit terlambat. Dari kesenjangan yang pertama karena *developer* harus saling memastikan kembali ketepatan interaksi antara user dan fungsi supaya dapat meminimalkan kesalahan pembuatan. Dari kesenjangan yang kedua karena membutuhkan waktu lebih untuk memahami alur proses dan rancangan *database* dalam bentuk model tersendiri. Dari kesenjangan yang ketiga karena *developer* harus mendesain sendiri secara mendadak ketika tampilan yang dibutuhkan tidak disediakan oleh *template* web desain Metronic.

Kesenjangan juga terjadi lagi pada bab ke-3, yaitu pada manajemen proyek. Manajemen pada proyek pengembangan inventori EDC berbasis web tidak menggunakan suatu metode yang sudah tersedia, seperti Scrum. Selain itu, tidak menggunakan alat yang mendukung manajemen proyek, seperti Trello dan Jira. Hal tersebut berakibat pada tiga hal.

Pertama, kurang jelasnya identifikasi terkait kebutuhan yang belum terpenuhi sehingga terkadang *supervisor* kebingungan memberikan pekerjaan selanjutnya kepada penulis. Kedua, tidak ada target waktu yang spesifik untuk penyelesaian setiap pekerjaan yang diberikan, tetapi tetap diusahakan untuk segera diselesaikan sehingga dapat beralih ke pekerjaan selanjutnya. Ketiga, tidak tentunya pembahasan yang dilakukan saat *daily meeting* dengan tim pada proyek ini, yaitu penulis dan rekan kerja sekaligus *supervisor*. Alasannya karena pembahasan dapat meliputi pelaporan *progress* pekerjaan, pemberian pekerjaan selanjutnya oleh *supervisor*, diskusi masalah atau kebingungan terkait pekerjaan, maupun pemberian umpan balik oleh *supervisor*.

*Daily meeting* dengan semua tim proyek pada bidang *software application development* dan *information technology* atau yang biasa disebut *huddle* juga sempat ditiadakan mulai bulan Oktober 2023 hingga Januari 2024. Hal tersebut menyebabkan berkurangnya umpan balik yang didapatkan dari anggota tim proyek lain atau bahkan dari *leader* bidang. Padahal umpan balik

dapat menjadi solusi untuk masalah proyek yang sedang dialami atau meningkatkan kualitas proyek supaya lebih baik lagi. Penulis sendiri juga mengalami dampaknya berupa kurangnya informasi penting pada divisi dan kurangnya sosialisasi dengan anggota tim proyek lain.

## 4.2 Pembelajaran Magang

### 4.2.1 Manfaat

Terdapat berbagai manfaat yang didapatkan oleh penulis selama menjalani 6 bulan magang di PT TSMakmur. Salah satu manfaatnya dari aspek keilmuan, yaitu ilmu tentang pengembangan sistem berbasis web dari sisi *back-end* menggunakan *framework* Spring Boot dan dari sisi *front-end* menggunakan jQuery dan AJAX dengan dukungan *template* Thymeleaf. Manfaat tersebut dapat dipecah menjadi tiga bagian.

Pertama, dari sisi *back-end* yang menggunakan *framework* Spring Boot. Penulis akhimya dapat mengetahui struktur sistem (hierarki folder dan *file*) yang memiliki hubungan erat pada empat lapisan *back-end* (*entity*, *repository*, *service*, dan *controller*) dengan prosesnya yang menggunakan berbagai keunggulan pada Spring Boot. Sebagai contohnya adalah penggunaan Spring Data pada lapisan *repository* yang berada pada folder *repository*. Ilmu tersebut akan membuat penulis lebih siap menghadapi pengembangan *back-end* menggunakan *framework* lain pada berbagai proyek mendatang.

Kedua, dari sisi *front-end* yang menggunakan jQuery dan AJAX. Penulis akhirnya dapat mengetahui proses pengiriman HTTP *request* dan penerimaan HTTP *response* antara *client side* dan *server side* secara asinkron dengan AJAX. Proses AJAX tersebut efisien dan mudah dipahami karena dituliskan dengan kode yang disederhanakan oleh jQuery. Selain itu, juga mengetahui proses pembuatan tabel dan *dropdown* dengan *plugin* jQuery (DataTables dan select2) yang datanya dapat diambil melalui AJAX. Ilmu tersebut dapat menjadi bekal untuk penulis dalam mengembangkan *front-end* pada proyek lain yang menggunakan jQuery dan AJAX.

Ketiga, dukungan oleh *template* Thymeleaf. Penulis dapat mengetahui bahwa *template* tersebut berintegrasi dengan *framework* Spring Boot yang tetap mendukung proses HTML dengan tampilannya yang dapat menjadi lebih dinamis. Hal tersebut karena dapat menyisipkan data dan logika dari *server side* ke elemen HTML di *client side*. Ilmu tersebut dapat digunakan kembali oleh penulis ketika di masa mendatang mengerjakan web secara *full-stack* dengan Spring Boot sebagai teknologi di sisi *back-end*.

### 4.2.2 Kendala, Hambatan, dan Tantangan

Selama menjalani 6 bulan magang di PT TSMakmur, tentu tidak selalu berjalan mulus. Kesulitan pasti muncul dan menjadi bagian yang tidak dapat dihindari dari praktik kerja nyata. Kesulitan tersebut dapat terbagi menjadi tiga, yaitu kendala, hambatan, dan tantangan.

Pertama, kendala yang dihadapi oleh penulis sebenarnya sudah dijelaskan pada subbab 4.1, yaitu adanya pengulangan tahapan pada metode pengembangan *waterfall*, dari analisis kebutuhan hingga pengujian, sebagai bentuk perbaikan dari *review* pengujian dan sebagai proses penambahan kebutuhan fitur dari dokumen kedua (pembaruan). Selain itu, *developer* harus saling memastikan kembali ketepatan interaksi antara *user* dan fungsi supaya dapat meminimalkan kesalahan pembuatan. Kedua hal tersebut mengakibatkan tujuan atau target waktu proyek terhalang dan terbatasi. Maka, solusi untuk kendala pertama adalah *developer* melakukan pengulangan tahapan dengan cepat tetapi tepat sesuai perbaikan dan kebutuhan. Solusi untuk kendala kedua adalah *developer* mencatat hasil analisisnya yang menjelaskan tentang interaksi antara *user* dan fungsi.

Kedua, hambatan yang dihadapi oleh penulis adalah minimnya ilmu dan pengalaman tentang pengembangan web dari sisi *front-end*, terlebih lagi dengan teknologi Thyemelaf, jQuery, dan AJAX. Hal tersebut dikarenakan penulis lebih berpengalaman dari sisi *back-end* ketika mengembangkan web dibandingkan dari sisi *front-end* yang biasanya hanya dengan teknologi HTML, CSS, dan JavaScript. Walaupun begitu, penulis juga menghadapi minimnya ilmu tentang sisi *back-end* dengan *framework* Spring Boot. Kedua hal tersebut sempat menjadi hambatan untuk pengembangan proyek pada awal magang karena penulis membutuhkan waktu lebih untuk memahami berbagai teknologi tersebut. Maka dari itu, penulis terus belajar dengan cepat melalui dokumen yang dimiliki setiap teknologi tersebut maupun melalui sumber lain di internet.

Ketiga, tantangan yang dihadapi oleh penulis adalah saat diberikan pekerjaan mengenai fitur pengembalian EDC yang rusak. Fitur tersebut harus dapat diselesaikan dalam waktu kurang dua minggu sebelum masa magang penulis berakhir. Hal tersebut menjadi tantangan tersendiri bagi penulis karena fitur tersebut cukup kompleks dan diberikan kepada penulis di tengah beberapa fitur lain yang belum selesai. Dengan begitu, penulis berusaha semaksimal mungkin untuk fokus menyelesaikan fitur tersebut, termasuk segera bertanya ke *supervisor* ketika mengalami kebingungan, supaya fitur lain juga dapat segera diselesaikan.

# BAB V
# PENUTUP

## 5.1 Kesimpulan

Proyek pengembangan inventori EDC berbasis web pada fitur pengembalian EDC yang rusak, *merchant*, dan *service point* (SP) telah berhasil dilakukan dengan baik dan sesuai dengan tujuan. Proyek ini menggunakan metode pengembangan *waterfall*, teknologi *back-end* berupa bahasa pemrograman Java, *framework* Spring Boot, dan *database* PostgreSQL, serta teknologi *front-end* berupa *template* Thymeleaf, *library* jQuery, dan AJAX.

Pada fitur pengembalian EDC yang rusak, akses utamanya dilakukan oleh *user* klien perusahaan *maker* dan *user* vendor *arrival checker* untuk melakukan berbagai fungsi pada berbagai halaman yang sesuai. Fungsi tersebut antara lain menambahkan EDC yang rusak sekaligus menentukan pembuatan pengembalian, memperbarui pengembalian berdasarkan *id*, mengajukan permintaan pengembalian berdasarkan *id*, menyetujui atau menolak permintaan pengembalian berdasarkan *id*, menerima pengembalian berdasarkan *id*, menampilkan semua pengembalian dan EDC yang rusak dengan dilengkapi filter, dan menampilkan pengembalian berdasarkan *id*. Dengan begitu, fitur tersebut sudah sesuai dengan kebutuhan yang ditentukan dan mencapai tujuan yang diharapkan karena pengembalian EDC yang mengalami kerusakan dapat dikelola dan dipantau dengan jelas dan akurat oleh klien perusahaan dan vendor *arrival*.

Pada fitur *merchant*, akses utamanya dilakukan oleh *user* SP *checker maker* untuk melakukan berbagai fungsi pada berbagai halaman yang sesuai. Fungsi tersebut yaitu membuat, memperbarui, atau menghapus *merchant*, dan mengajukannya sebagai permintaan berdasarkan *id*, menyetujui atau menolak permintaan *merchant* berdasarkan *id*, menampilkan semua *merchant* dengan dilengkapi filter, dan menampilkan *merchant* berdasarkan *id*. Dengan begitu, fitur tersebut sudah sesuai dengan kebutuhan yang ditentukan dan mencapai tujuan yang diharapkan karena entitas *merchant* dapat dikelola dengan terstruktur dan akurat oleh SP sebagai tujuan distribusi EDC dari SP. Hal tersebut berlaku juga untuk fitur SP, tetapi akses utamanya dilakukan oleh *user* klien perusahaan *checker maker*, fokus operasinya untuk entitas SP, serta kegunaannya sebagai tujuan distribusi EDC dari vendor *arrival* atau vendor *service management* dan untuk membuat akun *user* SP *checker maker*.

## 5.2 Saran

Proyek pengembangan inventori EDC berbasis web tentunya juga memiliki beberapa saran yang diharapkan dapat meningkatkan kualitas dan keberhasilan proyek ke depannya. Pertama, seharusnya kebutuhan fitur dan fungsi serta alur proses pada sistem ditentukan secara jelas dan rinci sejak awal tahapan metode pengembangan *waterfall*. Hal tersebut sangat penting untuk mencegah terjadinya pengulangan tahapan sehingga target waktu proyek dapat dicapai dengan tepat. Kedua, sistem idealnya memiliki rancangan untuk hak akses *user* pada fitur dan fungsi, alur proses, desain antarmuka, dan relasi antar tabel, dengan metode, jenis, atau bentuk yang lazim digunakan dalam pengembangan sistem. Hal tersebut akan sangat memudahkan *developer* dalam memahami dan memastikan kebutuhan sehingga implementasi kode lebih efektif, efisien, dan minim kesalahan. Ketiga, sistem sebaiknya menggunakan manajemen proyek beserta alatnya yang sudah tersedia dan lazim digunakan dalam pengembangan sistem. Hal tersebut akan sangat membantu pengelolaan proyek yang terstruktur, efektif, dan efisien.

# DAFTAR PUSTAKA


Abdulloh, R. (2017). *Membuat Toko Online dengan Teknik OOP, MVC, dan AJAX*. Jakarta: PT Elex Media Komputindo. Retrieved from https://books.google.co.id/books?hl=id&lr=&id=0C9IDwAAQBAJ&oi=fnd&pg=PP1&ots=NuvIFtt0lq&sig=BnVnbgfO0Ze_5G3ASXJCUUBb-4Q&redir_esc=y#v=onepage&q&f=false

Adji, L. S., & Mailoa, E. (2024). Pembuatan REST API Manajemen Data Karyawan Berbasis Website Menggunakan Spring Boot. *Jurnal Indonesia : Manajemen Informatika Dan Komunikasi, 5(2)*, 1543-1552. doi:https://doi.org/10.35870/jimik.v5i2.713

Albuquerque, T. (2023, October 12). *REST API with Spring Boot 3 — Part 4*. Retrieved June 19, 2024, from Medium: https://tiagoamp.medium.com/rest-api-with-spring-boot-3-part-4-a09e5591b942

Amrin, A., & Aldiansyah, M. R. (2021). Model Waterfall Untuk Rancang Bangun Sistem Informasi Pengadaan Mesin EDC Pada E-Channel Operations Perbankan. *INSANtek, 2(2)*, 51-56. doi:https://doi.org/10.31294/instk.v2i2.668

Azizah, D. N., & Nurgiyatna, N. (2021). Pengembangan Sistem Inventory Barang Perusahaan Dagang berbasis Website (Studi Kasus: CV. Agung Nugraha). *Emitor: Jurnal Teknik Elektro, 21(01)*, 42-48. Retrieved from https://journals.ums.ac.id/index.php/emitor/article/view/13418

Brühwiler, M. P. (2023). *Stock management system for small and medium-sized companies*. Skripsi, Haaga-Helia University of Applied Sciences, Helsinki. Retrieved from https://www.theseus.fi/handle/10024/797687

Buak, S. T. (2023). *PEMBANGUNAN SISTEM MANAJEMEN STOK PADA BUNYU MAJU BERSAMA BERBASIS WEBSITE*. Skripsi, Universitas Atma Jaya Yogyakarta, Yogyakarta. Retrieved from https://e-journal.uajy.ac.id/30700/

C, A. (2023, January 18). *Apa Itu AJAX dan Bagaimana Cara Kerjanya?* Retrieved May 12, 2024, from Hostinger: https://www.hostinger.co.id/tutorial/apa-itu-ajax

Dagha, W. C., & Susetyo, Y. A. (2021). Web Event, Spring Boot, Java Pembangunan Aplikasi Web Event menggunakan Framework Spring Boot di PT XYZ. *JATISI (Jurnal Teknik Informatika dan Sistem Informasi), 8(3)*, 1457-1469. doi:https://doi.org/10.35957/jatisi.v8i3.1052

Desmayani, N. M., Wardani, N., Nugraha, P. G., Indrawan, I. Y., & Mahendra, G. S. (2022). Sistem Informasi Inventory pada PT. Djaya Buah Bersinar Denpasar Berbasis Web. *INSERT: Information System and Emerging Technology Journal, 3(2)*, 82-93. Retrieved from https://ejournal.undiksha.ac.id/index.php/insert/article/view/54696

Dirosa, D., & Kurniawan, T. A. (2021). Pengembangan Sistem Point of Sale berbasis Web (Studi Kasus: Bos Duren Malang). *Jurnal Pengembangan Teknologi Informasi Dan Ilmu Komputer, 5(4)*, 1615-1622. Retrieved from https://j-ptiik.ub.ac.id/index.php/j-ptiik/article/view/8952

Eriksson, J. (2022). *MIGRATION OF THE USER INTERFACE OF A WEB APPLICATION - From Thymeleaf to Angular.* Skripsi, Högskolan på Åland, Mariehamn. Retrieved from https://www.theseus.fi/bitstream/handle/10024/732037/Eriksson_Jennie.pdf

Ganiardi, M. A., Salamah, I., & Kusumanto, R. (2016). JQUERY SEBAGAI KOMPONEN USABILITAS ANTARMUKA APLIKASI WEB. *Jurnal Poli-Teknologi, 14(2)*. doi:https://doi.org/10.32722/pt.v14i2.743

GeeksforGeeks. (2022, March 11). *Spring Boot – Architecture*. Retrieved June 19, 2024, from GeeksforGeeks: https://www.geeksforgeeks.org/spring-boot-architecture/

Hermansyah, Wijaya, R. F., & Utomo, R. B. (2023). Metode Waterfall Dalam Rancang Bangun Sistem Informasi Manajemen Kegiatan Masjid Berbasis Web. *KLIK: Kajian Ilmiah Informatika dan Komputer, 3(5)*, 563-571. Retrieved from https://djournals.com/klik/article/view/756

Jayanto, D. P. (2017). *RANCANG BANGUN BACK-END "SIAP": SISTEM INFORMASI ASPIRASI DAN PENGADUAN MASYARAKAT BERBASIS WEB DENGAN MENGGUNAKAN METODE MICROSERVICE SPRINGBOOT.* Skripsi, Institut Teknologi Sepuluh November, Jawa Timur. Retrieved from https://repository.its.ac.id/42440/

Khaqiqi, M. T., & Harani, N. H. (2023). *PENERAPAN METODE GAMIFIKASI PADA REST API SPRING BOOT.* Bandung: Penerbit Buku Pedia. Retrieved from https://books.google.co.id/books?hl=id&lr=&id=t76-EAAAQBAJ&oi=fnd&pg=PA1&dq=apa+itu+spring+boot&ots=VR3X5drFkK&sig=zc6XgauamVze_glLo-W5REv5PcI&redir_esc=y#v=onepage&q=apa%20itu%20spring%20boot&f=false

Ma'ruf, A. A., Widyaputri, F. A., Nafisah, S., & Gunawan, D. (2023). Website Rumah Sakit Pelayanan Kesehatan Umum Muhammadiyah Sragen. *Jurnal Komputer dan Teknik*

*Informatika (KONTAK), 1(1),* 21-30. Retrieved from https://karya.brin.go.id/id/eprint/21561/

Majdina, M. Y., Praptono, B., & I, M. D. (2020). Perancangan Aplikasi Manajemen Persediaan Gudang Berbasis Website Pada Umkm Batik Sinuwun Dengan Agile Scrum Development Method. *e-Proceeding of Engineering, 7(2),* 5630-5637. Retrieved from https://openlibrarypublications.telkomuniversity.ac.id/index.php/engineering/article/view/12680

Maulana, N. (2022). Rancangan Sistem Informasi Persediaan Barang Berbasis Web pada Perusahaan Perdagangan. *Jutisi : Jurnal Ilmiah Teknik Informatika dan Sistem Informasi, 11(1),* 189-198. doi:http://dx.doi.org/10.35889/jutisi.v11i1.816

Munawaroh, S. (2005). Mengeksplorasi Database PostgreSQL dengan PgAdmin III. *Dinamik, 10(2),* 103-107. Retrieved from https://www.unisbank.ac.id/ojs/index.php/fti1/article/view/17

Oracle. (2020, November 24). *What Is a Database?* Retrieved May 10, 2024, from Oracle: https://www.oracle.com/id/database/what-is-database/

PostgreSQL. (2024). *About.* Retrieved May 10, 2024, from PostgreSQL: https://www.postgresql.org/about/

Sianipar, R. H. (2016). *JQuery: Belajar Dari Studi Kasus.* Yogyakarta: Penerbit Andi. Retrieved from https://books.google.co.id/books?hl=id&lr=&id=Uml2DwAAQBAJ&oi=fnd&pg=PA1&dq=contoh+jquery&ots=PyevEhSYzu&sig=47VZU6-HY8S83GWschgEs5hdTLA&redir_esc=y#v=onepage&q=contoh%20jquery&f=false

Sofi, S. M., Ahmad, S. N., Elias, K. A., & Ismail, A. Y. (2015). Penggunaan Object-Relational Database Management System (ORDBMS) dalam Pembangunan Sistem Permohonan Kod Penyelidikan dan Penulisan (SISKOD). *International Conference on Information Technology & Society,* 317-324. Retrieved from http://fstm.kuis.edu.my/icits/proceeding/fullpapers/IC-ITS%202015%20-%20IT%20107.pdf

Solehudin, A.-A., Fariz, N., Wahyu, N., Fauzi Permana, R., & Saifudin, A. (2023). Rancang Bangun Digitalisasi Persediaan Barang Berbasis Web Menggunakan Metode Waterfall. *LOGIC: Jurnal Ilmu Komputer Dan Pendidikan, 1(4),* 1000-1005. Retrieved from https://www.journal.mediapublikasi.id/index.php/logic/article/view/3163

Tahir, M. A. (2018). Implementasi Ajax pada Aplikasi Indeks Artikel Berbasis Web. *Jurnal Ilmiah Sistem Informasi Dan Teknik Informatika (JISTI), 1(2)*, 60-68. Retrieved from https://journal.jisti.unipol.ac.id/index.php/jisti/article/view/12

Thymeleaf. (2023, July 30). *Tutorial: Using Thymeleaf*. Retrieved May 10, 2024, from Thymeleaf: https://www.thymeleaf.org/doc/tutorials/3.1/usingthymeleaf.html#what-is-thymeleaf

Vicky, V. O., & Syaripudin, A. (2022). PERANCANGAN SISTEM INFORMASI ABSENSI PEGAWAI BERBASIS WEB DENGAN METODE WATERFALL (STUDI KASUS : KANTOR DBPR TANGERANG SELATAN). *OKTAL: Jurnal Ilmu Komputer Dan Sains, 1(01)*, 17-26. Retrieved from https://www.journal.mediapublikasi.id/index.php/oktal/article/view/2

Wahid, A. A. (2020). Analisis Metode Waterfall Untuk Pengembangan Sistem Informasi. *Jurnal Imu-ilmu Informatika dan Manajemen STMIK*, 1-5. Retrieved from https://www.researchgate.net/publication/346397070_Analisis_Metode_Waterfall_Untuk_Pengembangan_Sistem_Informasi

Yudhanto, Y., & Prasetyo, H. A. (2019). *Mudah Menguasai Framework Laravel.* Jakarta: PT Elex Media Komputindo. Retrieved from https://books.google.co.id/books?hl=id&lr=&id=8tKdDwAAQBAJ&oi=fnd&pg=PP1&dq=mudah+menguasai+framework+laravel&ots=lv241q-VVM&sig=uvh1PRdZqEaeY7rXfxijlswCYGg&redir_esc=y#v=onepage&q=mudah%20menguasai%20framework%20laravel&f=false