OPTIMALISASI PENERAPAN MODUL ODOO 15 PADA PROYEK ENTERPRISE RESOURCE PLANNING (ERP)



Disusun Oleh:

N a m a	: Joaquin Brilliant P
NIM	: 22523190

PROGRAM STUDI INFORMATIKA – PROGRAM SARJANA
FAKULTAS TEKNOLOGI INDUSTRI
UNIVERSITAS ISLAM INDONESIA
2026

HALAMAN PENGESAHAN DOSEN PEMBIMBING

OPTIMALISASI PENERAPAN MODUL ODOO 15 PADA PROYEK ENTERPRISE RESOURCE PLANNING (ERP)

TUGAS AKHIR JALUR MAGANG

Disusun Oleh:

N a m a : Joaquin Brilliant Pramustya
NIM : 22523190

Yogyakarta, 14 Agustus 2026

Pembimbing,

(Andhik Budi Cahyono, S.T, M.T)

HALAMAN PENGESAHAN DOSEN PENGUJI

OPTIMALISASI PENERAPAN MODUL ODOO 15 PADA PROYEK ENTERPRISE RESOURCE PLANNING (ERP)

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, 14 Agustus 2026

Tim Penguji

Andhik Budi Cahyono, S.T, M.T _____

Anggota 1

Dr. Novi Setiani, S.T., M.T. _____

Anggota 2

Dr. Nur Wijayaning Rahayu, S.Kom., M.Cs. _____

Mengetahui,

Ketua Program Studi Informatika – Program Sarjana

Fakultas Teknologi Industri

Universitas Islam Indonesia

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

HALAMAN PERNYATAAN KEASLIAN TUGAS AKHIR

Yang bertanda tangan di bawah ini:

Nama	:	Joaquin Brilliant Pramustya
NIM	:	22523190

Tugas akhir dengan judul:

OPTIMALISASI PENERAPAN MODUL ODOO 15 PADA PROYEK ENTERPRISE RESOURCE PLANNING (ERP)

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, 14 Agustus 2026



(Joaquin Brilliant Pramustya)

HALAMAN PERSEMBAHAN

Laporan ini saya persembahkan untuk kedua orang tua saya, yang telah memberikan lebih dari sekadar kehidupan mereka menanamkan keyakinan bahwa belajar sungguh-sungguh dan bekerja dengan sepenuh hati adalah cara terbaik untuk menjalani hidup. Semoga setiap langkah dalam menimba ilmu ini menjadi ibadah dan bernilai di sisi Allah SWT.

HALAMAN MOTO

Ada hari-hari ketika langit terasa terlalu rendah dan dada terlalu sempit. Di saat seperti itu, yang tersisa hanyalah percaya bahwa Allah sedang menyusun sesuatu yang belum sanggup dipahami oleh mata yang terburu-buru.

KATA PENGANTAR

Puji syukur penulis panjatkan ke hadirat Allah SWT atas segala rahmat, hidayah, dan kemudahan yang diberikan sehingga laporan tugas akhir jalur magang dengan judul "Optimalisasi Penerapan Modul Odoo 15 pada Proyek Enterprise Resource Planning (ERP)" ini dapat diselesaikan dengan baik.

Laporan ini disusun sebagai salah satu syarat untuk memperoleh gelar Sarjana Komputer pada Program Studi Informatika, Fakultas Teknologi Industri, Universitas Islam Indonesia. Laporan ini mendokumentasikan kegiatan magang yang dilaksanakan di PT Gamatechno Indonesia selama enam bulan, mencakup proses optimalisasi modul-modul sistem ERP Ganesha Operations berbasis Odoo 15 serta keterlibatan penulis pada proyek pendukung lainnya.

Selama pelaksanaan magang dan penyusunan laporan ini, penulis menghadapi beberapa tantangan, terutama dalam memahami sistem ERP yang sudah berjalan dengan kompleksitas modul yang saling terintegrasi. Bekerja pada sistem aktif dengan pengguna nyata mengharuskan penulis untuk lebih berhati-hati dan teliti dalam setiap perubahan yang dilakukan. Namun dengan dukungan dari berbagai pihak, seluruh tantangan tersebut dapat dilewati.

Pada kesempatan ini, penulis ingin menyampaikan terima kasih yang sebesar-besarnya kepada:

1. Allah SWT atas segala rahmat, kemudahan, dan pertolongan yang diberikan di setiap langkah perjalanan ini.
2. Mama dan Ayah tercinta, yang telah memberikan segalanya — doa, dukungan, dan kepercayaan yang tidak pernah goyah. Setiap pencapaian ini adalah milik kalian juga.
3. Bapak Andhik Budi Cahyono, S.T., M.T. selaku dosen pembimbing yang telah meluangkan waktu untuk memberikan arahan, masukan, dan bimbingan selama proses penyusunan laporan ini.
4. Seluruh dosen Program Studi Informatika Universitas Islam Indonesia yang telah mendidik dan membimbing penulis selama menempuh pendidikan di kampus ini.
5. Seluruh tim PT Gamatechno Indonesia yang telah menerima penulis sebagai bagian dari tim dan memberikan kesempatan untuk belajar langsung di lingkungan pengembangan perangkat lunak profesional.

6. Teman-teman Program Studi Informatika UII yang telah menemani perjalanan akademik ini dari awal hingga akhir.
7. A Keysha Farah F.N, yang senantiasa menemani dan menyemangati di setiap proses penyusunan laporan ini. Terima kasih atas kesabaran dan dukungannya.

Penulis menyadari bahwa laporan ini masih jauh dari sempurna. Kritik dan saran yang membangun sangat diharapkan untuk perbaikan ke depannya. Semoga laporan ini dapat memberikan manfaat bagi pembaca, khususnya bagi yang ingin memahami proses optimalisasi sistem ERP berbasis Odoo dalam konteks proyek industri nyata.

Yogyakarta, 14 Agustus 2026



(Joaquin Brilliant Pramustya)

SARI

Pengelolaan organisasi berskala besar memerlukan sistem yang mampu mengintegrasikan seluruh fungsi bisnis secara terpadu. Ganesha Operations, lembaga bimbingan belajar berskala nasional, menggunakan sistem ERP berbasis Odoo 15 yang dikembangkan dan dipelihara oleh PT Gamatechno Indonesia. Sistem ini sudah berjalan aktif dengan pengguna nyata di berbagai cabang, namun tetap memerlukan penyesuaian seiring berkembangnya kebutuhan operasional inilah konteks utama pelaksanaan magang dalam laporan ini.

Magang dilaksanakan di PT Gamatechno Indonesia, Yogyakarta, selama enam bulan (September 2025 – Maret 2026) dengan sistem kerja hybrid. Penulis terlibat dalam tiga proyek: optimalisasi modul ERP Ganesha Operations (Odoo 15) sebagai proyek utama, bantuUMKM.com berbasis React.js, dan i-BOSS Nasmoco berbasis Laravel. Laporan ini berfokus pada proyek Odoo yang memiliki cakupan teknis dan relevansi paling tinggi.

Pengelolaan pekerjaan mengadopsi pendekatan *agile* dengan GitLab setiap *task* dikerjakan pada terisolasi, diajukan melalui *Merge Request*, dan ditinjau tim QA sebelum diintegrasikan ke utama. Pendekatan ini mencerminkan nilai inti Agile tanpa menerapkan ritual Scrum secara formal.

Delapan *task* optimalisasi berhasil dikerjakan pada lapisan *custom addons* melalui mekanisme *inheritance* tanpa menyentuh kode inti Odoo, meliputi perbaikan *error field* modul Jurnal Pengeluaran, validasi *mandatory* modul Karyawan dan Master Data Diskon, penyembunyian tombol pada modul Entri Jurnal, validasi nomor telepon modul Kontak, serta *filter* produk *expired* modul BIMKER. Dua *task* dengan kompleksitas integrasi tinggi telah di-*merge* dan dilanjutkan oleh anggota tim. Secara keseluruhan, optimalisasi yang dilakukan berkontribusi pada peningkatan integritas data, konsistensi antarmuka, dan keselarasan sistem dengan proses bisnis Ganesha Operations.

Kata kunci: *Enterprise Resource Planning*, Odoo 15, optimalisasi modul, *custom addon*, *inheritance*.

GLOSARIUM

Agile	metodologi pengembangan perangkat lunak yang berfokus pada iterasi pendek dan adaptif, mengutamakan kolaborasi tim dan respons terhadap perubahan kebutuhan.
Branch	cabang pengembangan pada Git yang dibuat terisolasi dari kode utama agar pengerjaan satu *task* tidak mengganggu kode yang sedang berjalan.
Compile	proses untuk mengubah berkas kode program dengan berkas lain yang terkait menjadi berkas yang siap untuk dieksekusi oleh sistem operasi secara langsung.
Custom Addon	modul tambahan yang dikembangkan di luar modul bawaan Odoo dan ditempatkan dalam direktori terpisah.
Debug	langkah untuk menelusuri kesalahan kode program.
Docker	platform kontainerisasi yang mengemas aplikasi beserta dependensinya ke dalam container agar dapat berjalan secara konsisten di berbagai lingkungan pengembangan.
ERP	(Enterprise Resource Planning) sistem informasi terintegrasi yang menyatukan fungsi-fungsi bisnis seperti keuangan, SDM, dan operasional ke dalam satu platform berbasis data terpusat.
Inheritance	mekanisme pada Odoo yang memungkinkan model atau tampilan yang sudah ada diperluas atau dimodifikasi tanpa menyentuh kode sumber aslinya.
Issue	unit pekerjaan pada GitLab yang memuat deskripsi *task* yang perlu diselesaikan beserta penugasan kepada pengembang yang bertanggung jawab.
Merge Request	permintaan formal di GitLab untuk menggabungkan perubahan kode ke *branch* utama setelah melalui peninjauan tim QA dan persetujuan lead proyek.
Misfit	ketidaksesuaian antara fungsionalitas bawaan sistem ERP dengan kebutuhan aktual organisasi pengguna.
MVC	(Model-View-Controller) pola arsitektur yang memisahkan aplikasi menjadi tiga lapisan: Model untuk data dan

	logika bisnis, View untuk antarmuka, dan Controller sebagai penghubung keduanya.
Odoo	platform ERP berbasis open-source yang dibangun menggunakan Python dan PostgreSQL, menyediakan ratusan modul bisnis yang dapat dikustomisasi sesuai kebutuhan organisasi.
ORM	(Object-Relational Mapping) lapisan abstraksi yang memetakan objek Python ke tabel basis data PostgreSQL secara otomatis tanpa penulisan perintah SQL secara langsung.
QA	(Quality Assurance) peran dalam tim pengembangan yang bertanggung jawab melakukan pengujian fungsional terhadap perubahan kode sebelum diintegrasikan ke sistem utama.
State	status suatu dokumen dalam alur kerja Odoo seperti draft atau done, yang digunakan sebagai dasar pengaturan perilaku antarmuka secara dinamis.

DAFTAR ISI

HALAMAN JUDUL i
HALAMAN PENGESAHAN DOSEN PEMBIMBING ii
HALAMAN PENGESAHAN DOSEN PENGUJI iii
HALAMAN PERNYATAAN KEASLIAN TUGAS AKHIR iv
HALAMAN PERSEMBAHAN v
HALAMAN MOTO vi
KATA PENGANTAR vii
SARI ix
GLOSARIUM x
DAFTAR ISI xii
DAFTAR TABEL xiv
DAFTAR GAMBAR xv
BAB I PENDAHULUAN 1
1.1 Latar Belakang 1
1.2 Ruang Lingkup 2
1.3 Batasan Aktivitas 3
1.4 Tujuan 4
1.5 Manfaat 4
1.6 Sistematika Penulisan 5
BAB II LANDASAN TEORI DAN TINJAUAN PUSTAKA 7
2.1 Enterprise Resource Planning (ERP) 7
2.2 Odoo 15 9
2.3 Docker dan Docker Compose 13
2.4 PostgreSQL 13 15
2.5 Metode Pengembangan Agile 15
2.6 Tinjauan Pustaka 17
BAB III PELAKSANAAN MAGANG 18
3.1 Manajemen Proyek 18
3.1.1 Penempatan Kerja 18
3.1.2 Alur Kerja Agile 19
3.1.3 Aktivitas Pendukung Selama Magang 21
3.1.4 Tools yang Digunakan 22

3.1.5 Daftar Proyek dan Hasil 22
3.1.6 Daftar Tugas dan Proyek 23
3.2 Proyek Utama: Optimalisasi ERP Ganesha Operations (Odoo 15) 25
3.2.1 Gambaran Sistem ERP Ganesha Operations 26
3.2.2 Arsitektur Dan Posisi Pengembangan Modul 27
3.2.3 Konfigurasi dan Pengoperasian *Environment* Docker 28
3.2.4 Perbaikan *Error Field* pada Modul Jurnal Pengeluaran 29
3.2.5 Validasi *Mandatory* pada Modul Karyawan dan Master Data Diskon 31
3.2.6 Penyembunyian Tombol pada Modul Entri Jurnal 34
3.2.7 Perbaikan Logic Nominal dan *Filter* Pembayaran Modul Penyetoran 36
3.2.8 Perbaikan Validasi Nomor Telepon Modul Kontak 40
3.2.9 *Filter* Produk *Expired* Modul BIMKER 42
3.3 Proyek Training: bantuUMKM.com 44
3.3.1 Gambaran Sistem dan Tech Stack 44
3.3.2 Pengembangan *Login* 45
3.3.3 Pengembangan *Dashboard* dan Fitur CRUD modul Sales, Reports, Employee 45
3.4 Proyek i-BOSS Nasmoco (Laravel) 46
3.4.1 Gambaran Sistem dan Tech Stack 46
3.4.2 Pembaruan Logic Modal Datatable dan *Filter* pada Menu User Request 47
3.4.3 Penambahan Fitur Lampiran pada Halaman Detail User Request 47
3.4.4 Perbaikan *Bug* Edit Vendor Group 48
BAB IV REFLEKSI PELAKSANAAN MAGANG 48
4.1 Relevansi Akademik 49
4.2 Pembelajaran Magang. 50
BAB V PENUTUP 52
5.1 Kesimpulan 52
5.2 Saran 53
DAFTAR PUSTAKA 55
LAMPIRAN 57

DAFTAR TABEL

Tabel 3.1 Daftar Proyek Saat Magang 22
Tabel 3.2 Daftar Tugas pada Proyek Optimalisasi ERP Ganesha Operations 23
Tabel 3.3 Daftar Tugas pada Proyek bantuUMKM.com 24
Tabel 3.4 Daftar Tugas pada Proyek i-BOSS Nasmoco 25
Tabel 3.5 Komponen Utama Arsitektur Client-Server Sistem 27
Tabel 3.6 Rincian *Task* yang dilakukan 28
Tabel 3.7 Konfigurasi Docker Compose 29

DAFTAR GAMBAR

Gambar 2.1 Gambar Konsep ERP 7
Gambar 2.2 Gambar Perbandingan Sistem Terpisah VS ERP Terintegrasi 8
Gambar 2.3 Gambar Multitier Architecture 10
Gambar 2.4 Alur MVC 11
Gambar 2.5 Mekanisme *Inheritance* 13
Gambar 2.6 Kontainer Odoo didalam Docker 14
Gambar 2.7 Alur Agile 16
Gambar 3.1 Alur Kerja Pemberian *Task* 20
Gambar 3.2 Halaman Odoo 27
Gambar 3.3 *Task Error Field* 30
Gambar 3.4 Perbaikan Kode Entri Jurnal 31
Gambar 3.5 *Mandatory Field* Karyawan 31
Gambar 3.6 Perbaikan Kode *Field* Karyawan 32
Gambar 3.7 *Mandatory Field* Diskon 33
Gambar 3.8 Perbaikan kode *Mandatory* Diskon 33
Gambar 3.9 Hide Button "kirim Surat Penyetoran" 34
Gambar 3.10 Perbaikan Kode Hide Button 35
Gambar 3.11 Hide Button surat instruksi setor 35
Gambar 3.12 Perbaikan Kode Hide Button 35
Gambar 3.13 Perbaikan Kode Hide Button 36
Gambar 3.14 Perbaikan Kode Nominal Belum Setor 38
Gambar 3.15 Perbaikan Kode *Filter* Tunai dan Non tunai 39
Gambar 3.16 Penambahaan *Field* Transfer 39
Gambar 3.17 *Error* Nomor telepon tidak dapat diinput 41
Gambar 3.18 Perbaikan Kode Nomor Telepon 42
Gambar 3.19 Perbaikan Kode *Expired* 43
Gambar 3.20 Perbaikan Kode *mandatory* 43

BAB I
PENDAHULUAN

1.1 Latar Belakang

Semakin kompleks operasional sebuah organisasi, semakin besar kebutuhannya terhadap sistem informasi yang mampu menyatukan seluruh fungsi bisnis dalam satu platform yang terintegrasi. Bagian keuangan, sumber daya manusia, pengadaan, persediaan, hingga penjualan dari sebuah perusahaan pada hakikatnya saling bergantung satu sama lain, Sehingga tidak bisa lagi dikelola secara terpisah. Kebutuhan inilah yang dijawab oleh sistem Enterprise Resource Planning (ERP), yaitu sistem informasi terintegrasi yang menyatukan berbagai fungsi bisnis di atas satu basis data terpusat, sehingga data yang dihasilkan oleh satu fungsi langsung tersedia bagi fungsi lainnya tanpa duplikasi pencatatan. Namun, implementasinya tidak sesederhana instalasi perangkat lunak: setiap organisasi memiliki proses bisnis dan alur kerja yang khas, sementara sistem ERP di pasar dirancang untuk kebutuhan industri secara umum. Ketidaksesuaian inilah yang menempatkan kemampuan adaptasi platform ERP sebagai pertimbangan utama sebuah perusahaan memilih platform tersebut.

Di antara berbagai platform ERP yang berkembang saat ini, Odoo merupakan salah satu solusi open-source yang paling banyak diadopsi industri pada satu dekade terakhir. Dibangun menggunakan Python pada lapisan backend, PostgreSQL sebagai basis data, dan XML untuk pendefinisian antarmuka, Odoo menawarkan kombinasi keunggulan yang sulit dipenuhi platform lain: modularitas yang memungkinkan adopsi fungsionalitas secara bertahap, fleksibilitas pengembangan melalui mekanisme *inheritance* yang aman terhadap kode inti, sifat open-source yang membebaskan organisasi dari ketergantungan lisensi berbayar, serta kemudahan pengembangan modul kustom yang menjadikannya adaptif terhadap kebutuhan bisnis sebuah organisasi. Karakteristik inilah yang menjadikan Odoo relevan bagi organisasi dengan proses bisnis yang dinamis.

Pertimbangan tersebut juga menjadi alasan strategis bagi PT Gamatechno Indonesia, perusahaan teknologi informasi yang berdiri sejak 2005 di Yogyakarta dan merupakan bagian dari ekosistem bisnis Universitas Gadjah Mada. Bergerak di bidang solusi digital terintegrasi yang mencakup pengembangan sistem informasi enterprise, aplikasi mobile, IoT, dan transformasi digital, perusahaan ini telah lebih dari dua dekade melayani klien dari sektor pendidikan, pemerintahan, manufaktur, kesehatan, keuangan, hingga logistik. Dalam

ekosistem layanan tersebut, Odoo menjadi salah satu platform andalan perusahaan dalam mengembangkan solusi ERP bagi kliennya salah satunya sistem ERP Ganesha Operations, sebuah sistem produksi aktif yang menjadi konteks utama laporan tugas akhir ini.

Perlu disadari bahwa implementasi ERP tidak berhenti pada saat sistem mulai digunakan. Sebaliknya, ERP yang sudah berjalan justru memasuki fase yang menantang secara teknis, yaitu fase optimalisasi berkelanjutan untuk menyesuaikan sistem dengan dinamika kebutuhan bisnis yang terus berkembang. Kebutuhan pengguna berubah, regulasi diperbarui, dan alur operasional mengalami penyempurnaan dari waktu ke waktu. Pada sistem berbasis Odoo, kebutuhan-kebutuhan tersebut umumnya bermuara pada penyempurnaan logika bisnis, penambahan validasi data, penyesuaian perilaku antarmuka berdasarkan status *workflow*, hingga penajaman mekanisme *filter*ing dan pengolahan data. Pekerjaan semacam ini memiliki kompleksitas tersendiri karena dilakukan di atas sistem aktif. Setiap perubahan harus mempertimbangkan dampaknya terhadap modul-modul lain yang saling terintegrasi melalui basis data terpusat.

Konteks inilah yang melatarbelakangi laporan tugas akhir ini. Penulis berkesempatan terlibat langsung dalam proses optimalisasi modul-modul sistem ERP Ganesha Operations berbasis Odoo 15 selama pelaksanaan magang di PT Gamatechno Indonesia. Kontribusi yang diberikan tidak berorientasi pada pembangunan sistem baru, melainkan pada optimalisasi, perbaikan, dan penyesuaian modul-modul yang telah digunakan agar lebih akurat dan lebih selaras dengan kebutuhan operasional yang sesungguhnya. Laporan ini disusun untuk mendokumentasikan proses tersebut secara teknis dan akademik, sekaligus menjadi referensi bagi pengembangan dan optimalisasi sistem ERP berbasis Odoo di masa mendatang.

1.2 Ruang Lingkup

Laporan tugas akhir ini berpusat pada proyek optimalisasi sistem ERP Ganesha Operations berbasis Odoo 15. Proyek ini menjadi fokus utama pembahasan karena cakupan teknisnya yang paling luas, durasi keterlibatan yang paling panjang, serta relevansinya yang paling tinggi terhadap judul laporan. Seluruh analisis, pembahasan implementasi, dan refleksi teknis dalam laporan ini berpusat pada pengalaman penulis selama melakukan optimalisasi modul-modul sistem Ganesha Operations yang telah berjalan secara aktif di lingkungan produksi.

Ruang lingkup pekerjaan pada proyek Odoo yang dibahas dalam laporan mencakup berbagai bentuk optimalisasi modul yang dilakukan berdasarkan *task* pada repositori GitLab

perusahaan. Pekerjaan tersebut meliputi penyesuaian logika bisnis pada lapisan model, penambahan validasi data untuk menjaga integritas informasi, perbaikan proses transaksi yang tidak akurat, penyesuaian perilaku antarmuka pengguna berdasarkan status *workflow*, serta penajaman mekanisme *filter*ing dan pengolahan data. Seluruh pekerjaan ini diterapkan pada sistem ERP yang sudah memiliki pengguna dan data aktif, sehingga setiap optimalisasi dilakukan dengan mempertimbangkan integritas modul-modul lain yang saling terintegrasi melalui basis data terpusat.

Selain proyek Odoo, penulis juga terlibat dalam dua proyek pendamping yaitu bantuUMKM.com dan i-BOSS Nasmoco. Kedua proyek tersebut disebutkan dalam laporan ini sebagai bagian dari gambaran utuh pengalaman magang yang turut membentuk kompetensi teknis penulis, namun tidak menjadi fokus pembahasan utama. Porsi pembahasan kedua proyek tersebut dibahas secara singkat sesuai dengan porsinya, tanpa menyamai kedalaman pembahasan pada proyek Odoo.

Pembahasan laporan ini tidak mencakup perancangan arsitektur sistem secara menyeluruh, pembangunan modul ERP baru dari awal, maupun konfigurasi infrastruktur server produksi. Hal-hal tersebut berada di luar ranah pekerjaan penulis selama magang, sehingga tidak dijadikan bagian dari ruang lingkup pembahasan laporan.

1.3 Batasan Aktivitas

Dari ketiga proyek yang diikuti selama masa magang, proyek kustomisasi sistem ERP Odoo 15 untuk klien Ganesha Operations menjadi fokus utama pembahasan dalam laporan tugas akhir ini. Pertimbangan utamanya adalah cakupan teknis yang paling luas, durasi keterlibatan yang paling panjang, serta relevansi yang paling tinggi terhadap topik yang diangkat. Oleh karena itu, pembahasan mengenai implementasi, proses pengembangan, dan pembelajaran teknis dalam laporan ini berpusat pada pengalaman penulis selama mengerjakan kustomisasi modul-modul sistem Ganesha Operations.

Dalam pelaksanaannya, seluruh pekerjaan penulis pada proyek Odoo bersifat *task*-based dan mengikuti alur kerja yang sudah terdefinisi dalam repositori proyek perusahaan. Penulis tidak terlibat dalam pembuatan modul baru dari awal, perancangan arsitektur sistem, maupun konfigurasi infrastruktur server produksi. Keterlibatan penulis terbatas pada pengerjaan *task* kustomisasi yang diberikan, termasuk mengikuti meeting bersama klien dalam kapasitas memahami konteks kebutuhan, bukan sebagai pengambil keputusan teknis.

Dua proyek lainnya bantuUMKM.com dan i-BOSS Nasmoco tetap menjadi bagian dari gambaran utuh pengalaman magang yang membentuk kompetensi penulis. Pada proyek bantuUMKM.com, keterlibatan penulis terbatas pada sisi frontend di bawah pengawasan mentor. Pada proyek i-BOSS Nasmoco, penulis berperan sebagai *developer* dengan cakupan yang lebih terbatas dibandingkan proyek Odoo. Kedua proyek ini disertakan dalam laporan secara proporsional dan tidak menjadi fokus pembahasan teknis utama.

Pembatasan ruang lingkup ini menjadi landasan bagi penyusunan landasan teori pada bab berikutnya, yang mencakup teknologi dan konsep yang secara langsung mendukung pelaksanaan proyek Odoo selama masa magang.

1.4 Tujuan

Tujuan dari kegiatan magang yang dilaksanakan di PT Gamatechno Indonesia pada proyek sistem ERP Ganesha Operations berbasis Odoo 15 adalah sebagai berikut.

1. Mengoptimalkan modul-modul sistem ERP Ganesha Operations yang sudah berjalan, melalui perbaikan logika bisnis dan penyesuaian perilaku sistem agar lebih sesuai dengan kebutuhan operasional pengguna.
2. Meningkatkan keakuratan dan keandalan data pada sistem ERP Ganesha Operations dengan menambahkan mekanisme validasi pada lapisan model maupun antarmuka di modul-modul yang dioptimalkan.
3. Menyesuaikan tampilan antarmuka dan alur kerja sistem agar lebih selaras dengan proses bisnis yang berlaku, termasuk pembatasan akses elemen antarmuka berdasarkan status rekord dan penyesuaian alur input data.

1.5 Manfaat

Kegiatan magang ini diharapkan dapat memberikan manfaat bagi beberapa pihak yang terlibat.

Bagi Ganesha Operations selaku pengguna sistem, optimalisasi yang dilakukan diharapkan dapat meningkatkan kualitas sistem ERP yang digunakan sehari-hari. Perbaikan logika bisnis pada modul Penyetoran diharapkan dapat meningkatkan akurasi kalkulasi nilai transaksi keuangan. Penambahan validasi pada beberapa modul diharapkan dapat mengurangi potensi kesalahan input dan penyimpanan data yang tidak lengkap. Penyesuaian antarmuka

berbasis status rekord diharapkan dapat memperjelas alur kerja pengguna dan mengurangi potensi kesalahan pengisian data dalam operasional sehari-hari.

Bagi PT Gamatechno Indonesia selaku pengembang sistem, keterlibatan penulis dalam proses optimalisasi diharapkan dapat membantu kelangsungan pemeliharaan sistem ERP Ganesha Operations dan sedikit meringankan beban tim pengembang yang juga menangani proyek lain secara bersamaan.

Bagi penulis sendiri, kegiatan magang ini memberikan pengalaman teknis yang cukup berbeda dari pengerjaan proyek akademik biasa. Bekerja langsung pada sistem yang sudah aktif digunakan membuat penulis belajar menelusuri arsitektur modul yang sudah ada, memahami keterkaitan antar model data, dan lebih berhati-hati dalam membuat perubahan karena dampaknya bisa berpengaruh pada modul lain yang saling terhubung.

1.6 Sistematika Penulisan

Penyajian laporan tugas akhir ini menggunakan sistematika penulisan sebagai berikut.

BAB I PENDAHULUAN

Bab ini membahas latar belakang penulisan laporan tugas akhir, ruang lingkup, batasan aktivitas, tujuan, manfaat, dan sistematika penulisan laporan.

BAB II LANDASAN TEORI DAN TINJAUAN PUSTAKA

Bab ini membahas landasan teori yang berkaitan dengan Enterprise Resource Planning, *framework* pengembangan Odoo 15, Docker dan Docker Compose, PostgreSQL, serta metode pengembangan Agile, dilengkapi dengan tinjauan terhadap penelitian-penelitian terdahulu yang membahas implementasi sistem ERP berbasis Odoo.

BAB III PELAKSANAAN MAGANG

Bab ini membahas manajemen proyek selama magang, gambaran sistem ERP Ganesha Operations beserta arsitektur dan *environment* pengembangannya, serta uraian teknis atas setiap *task* optimalisasi modul yang dikerjakan. Bab ini juga membahas secara proporsional dua proyek pendamping, yaitu bantuUMKM.com dan i-BOSS Nasmoco.

BAB IV REFLEKSI PELAKSANAAN MAGANG

Bab ini membahas relevansi antara pembelajaran di perkuliahan dengan praktik di industri, serta pembelajaran, kendala, dan keterampilan yang berhasil dikembangkan selama pelaksanaan magang.

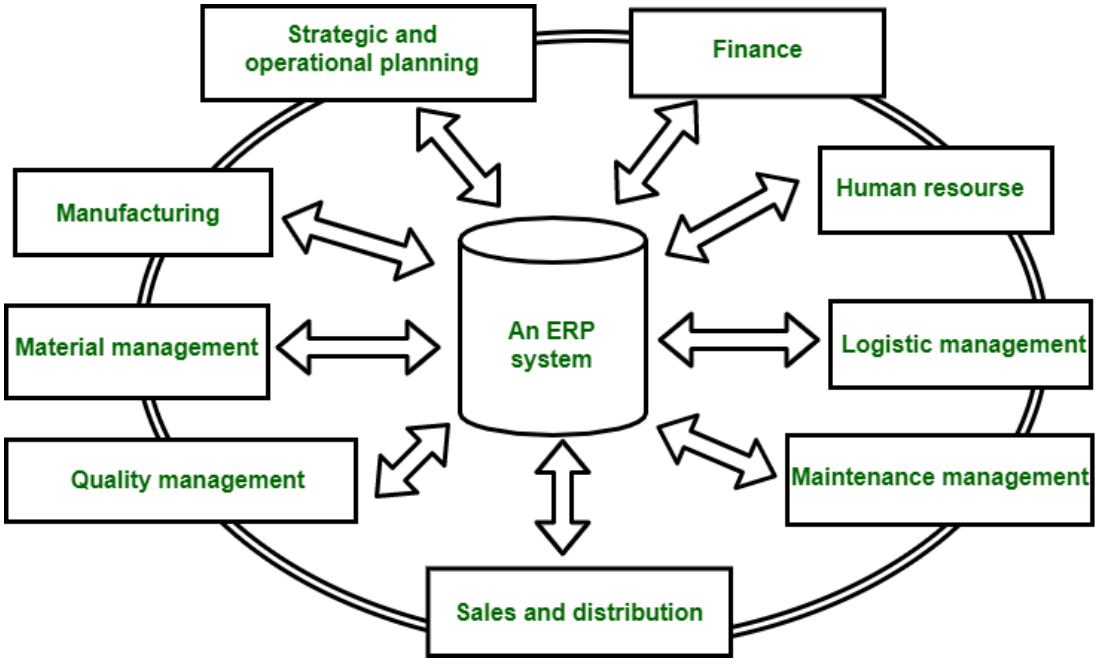
BAB V PENUTUP

Bab ini membahas kesimpulan atas pelaksanaan magang dan penulisan laporan, serta saran bagi pihak-pihak terkait untuk pengembangan selanjutnya.

BAB II
LANDASAN TEORI DAN TINJAUAN PUSTAKA

2.1 Enterprise Resource Planning (ERP)

Pengelolaan organisasi modern menuntut kemampuan untuk melihat seluruh proses bisnis secara terpadu dan dalam waktu nyata. Kebutuhan inilah yang melatarbelakangi lahirnya konsep Enterprise Resource Planning (ERP), merupakan sistem yang digunakan perusahaan untuk mengelola berbagai proses bisnis dengan menggunakan satu basis data yang terintegrasi untuk menghadirkan data yang konsisten di seluruh fungsi bisnis secara real time (Monk & Wagner 2013). Dalam konteks sistem informasi, ERP bukan sekadar perangkat lunak manajemen ERP merupakan infrastruktur digital yang mendefinisikan bagaimana sebuah organisasi merepresentasikan, menyimpan, dan memproses data operasionalnya. Sistem ini mengintegrasikan proses bisnis di bidang manufaktur dan produksi, keuangan dan akuntansi, penjualan dan pemasaran, serta sumber daya manusia ke dalam satu sistem perangkat lunak terpadu, sehingga informasi yang sebelumnya tersebar di berbagai sistem terpisah tersimpan dalam satu repositori data terpusat yang dapat diakses oleh seluruh bagian organisasi (Laudon & Laudon, 2018). Berikut adalah gambar tentang konsep ERP.



Gambar 2.1 Gambar Konsep ERP

Sumber: https://www.geeksforgeeks.org/

Konsekuensi dari arsitektur terpadu ini cukup signifikan: sebuah transaksi yang dicatat pada satu bagian sistem secara otomatis berdampak pada bagian lain yang berbagi basis data yang sama, tanpa memerlukan sinkronisasi manual. Keterpaduan ini memberikan visibilitas menyeluruh terhadap kondisi organisasi secara real time kondisi yang sulit dicapai apabila setiap fungsi bisnis dikelola oleh sistem yang terpisah. Namun keterpaduan ini juga membawa tanggung jawab tersendiri: setiap perubahan pada satu bagian sistem berpotensi berdampak melampaui batas modul yang bersangkutan, sehingga pengelolaan perubahan pada sistem ERP yang sudah berjalan menuntut kehati-hatian agar modifikasi yang dilakukan tidak mengganggu fungsi lain yang saling bergantung. Kondisi inilah yang secara nyata dihadapi pada organisasi yang sebelumnya mengandalkan pengelolaan data secara manual dan tidak terintegrasi, di mana ketiadaan sistem yang terpusat menyebabkan keterlambatan akses informasi dan meningkatkan risiko kesalahan manusia (Kurniawan, Yulia, & Irawan, 2021).Berikut adalah gambar perbandingan tentang sistem teripsah dan terintegrasi.



Gambar 2.2 Gambar Perbandingan Sistem Terpisah VS ERP Terintegrasi

Sumber: https://www.neuronworks.co.id/

Implementasi ERP di dunia nyata hampir selalu membutuhkan penyesuaian. Sistem ERP yang tersedia di pasaran baik yang bersifat privat maupun open-source dirancang untuk mencakup kebutuhan bisnis secara generik, sementara setiap organisasi memiliki proses, terminologi, dan alur kerja yang khas. Ketidaksesuaian antara fungsionalitas yang disediakan

paket perangkat lunak ERP dengan kebutuhan aktual organisasi inilah yang dalam literatur dikenal sebagai misfit (Soh, Siew Kien, & Joanne, 2000). Untuk mengatasi misfit, organisasi pada dasarnya memiliki beberapa pilihan yang berada dalam satu spektrum: mengadaptasi proses bisnis agar sesuai dengan sistem, menerima keterbatasan sistem sebagaimana adanya, mengembangkan workaround tanpa menyentuh kode inti, hingga melakukan kustomisasi untuk mencapai fungsionalitas yang dibutuhkan (Soh et al., 2000).

(Laudon & Laudon, 2018) membedakan dua cara intervensi teknis terhadap perangkat lunak ERP. Cara pertama adalah melalui configuration tables yang disediakan oleh vendor, yang memungkinkan organisasi menyesuaikan aspek tertentu dari sistem sesuai cara kerja bisnisnya tanpa menyentuh kode sumber. Cara kedua adalah dengan menulis ulang sebagian kode perangkat lunak apabila konfigurasi standar tidak mencukupi kebutuhan organisasi meskipun kustomisasi yang terlalu ekstensif berisiko menurunkan performa sistem dan mengorbankan manfaat integrasi yang menjadi keunggulan utama ERP. Oleh karena itu, setiap upaya penyempurnaan yang dilakukan terhadap sistem ERP yang sudah berjalan perlu dilaksanakan secara hati-hati agar optimalisasi dapat berlangsung tanpa mengorbankan stabilitas dan integritas sistem secara keseluruhan.

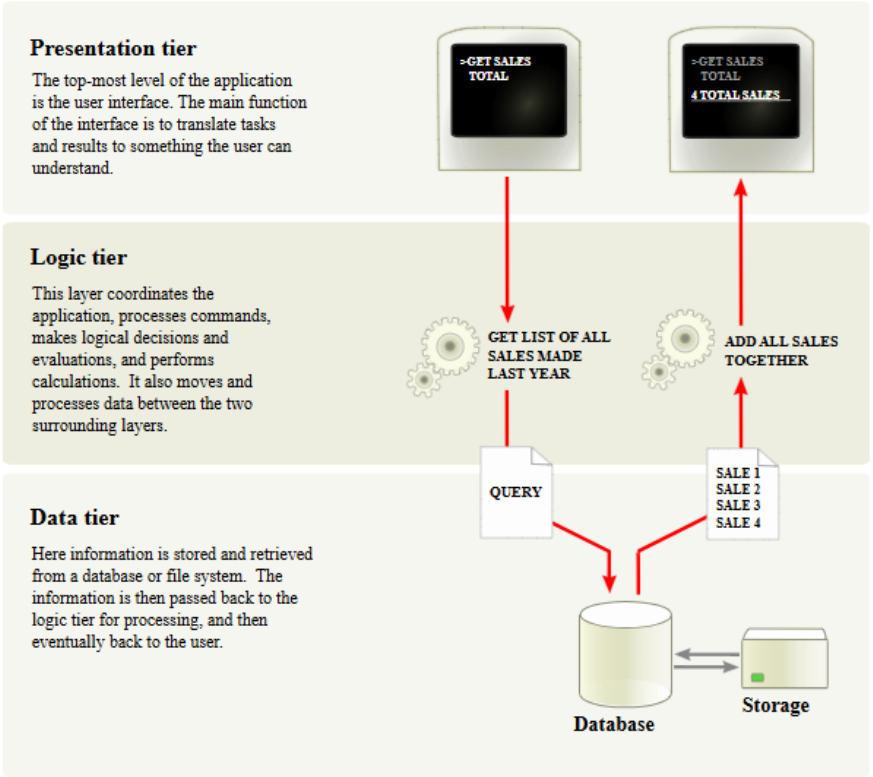
2.2 Odoo 15

Odoo merupakan platform Enterprise Resource Planning (ERP) berbasis open-source yang dibangun menggunakan bahasa pemrograman Python, XML, dan JavaScript dengan PostgreSQL sebagai sistem manajemen basis datanya (Arfikriyana & Suhendi, 2022). Di atas platform tersebut, ratusan aplikasi bisnis dibangun sebagai modul-modul mandiri yang saling terintegrasi mencakup CRM, akuntansi, penjualan, inventaris, eCommerce, HR, MRP, dan lain sebagainya sehingga seluruh fungsi operasional perusahaan dapat berjalan dalam satu ekosistem yang terhubung melalui satu basis data terpusat (Aziza & Rahayu, 2019). Karakteristik modular ini menjadikan Odoo sebagai salah satu platform ERP yang paling cepat berkembang di dunia, dengan sebagian besar modulnya tersedia secara gratis untuk diunduh dan digunakan (Yana, Sasmita, & Susila, 2022).

Keunggulan Odoo tidak hanya terletak pada keterbukaan kode sumbernya, tetapi juga pada fleksibilitas cakupan adopsinya. Odoo dapat diaplikasikan pada berbagai macam sektor usaha perdagangan, tekstil, agrikultural, dan lainnya serta tidak hanya digunakan oleh perusahaan-perusahaan besar, tetapi juga oleh perusahaan kecil dan independen (Arfikriyana & Suhendi, 2022). Fleksibilitas terbukti dalam praktik: (Yana, Sasmita, & Susila, 2022)

mendokumentasikan penerapan Odoo pada usaha garmen berskala UMKM menggunakan modul Sales, Purchase, Inventory, Manufacturing, Invoice, dan Website secara bersamaan, sementara (Terminanto & Ismail, 2022) menunjukkan bahwa Odoo mampu memenuhi kebutuhan pengelolaan keuangan organisasi jasa melalui modul accounting yang mengintegrasikan pembuatan invoice, pencatatan jurnal, hingga pelaporan keuangan secara real-time.

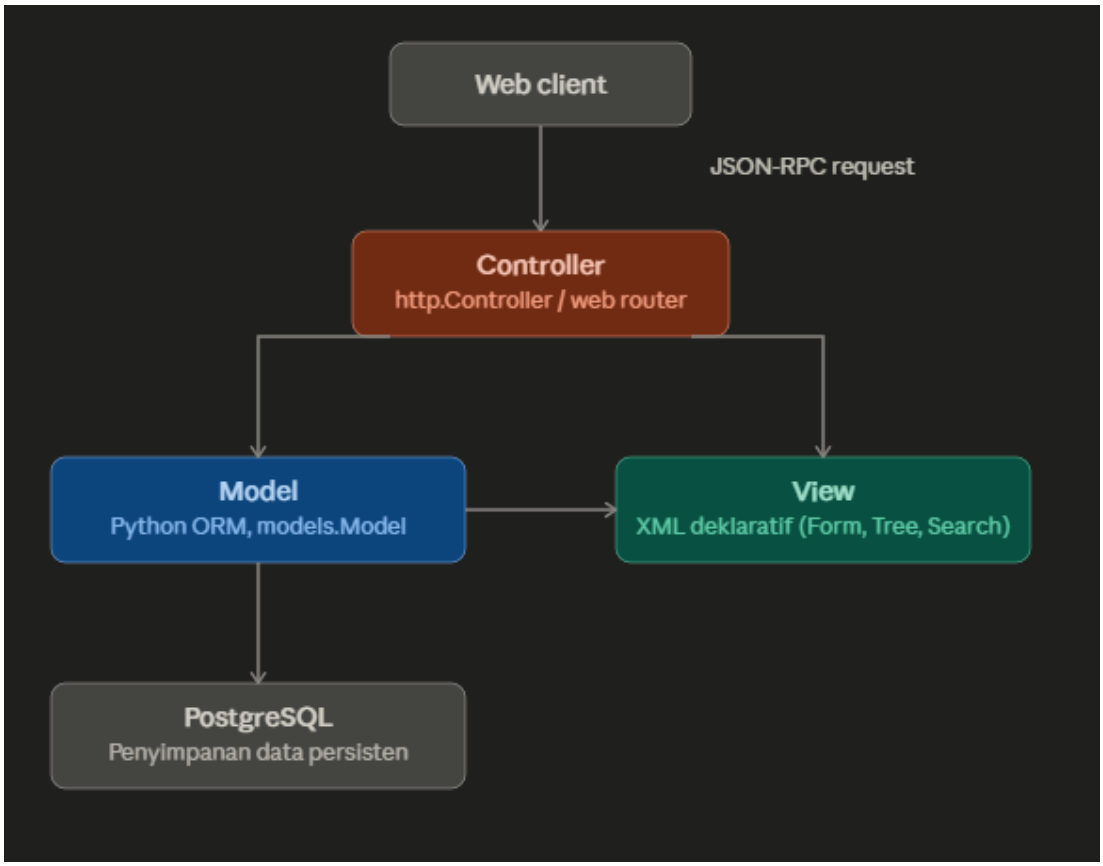
Odoo dibangun di atas arsitektur multitier yang memisahkan sistem menjadi tiga lapisan dengan tanggung jawab yang berbeda dan terdefinisi secara tegas. Lapisan pertama adalah lapisan presentasi (presentation tier) yang menangani seluruh interaksi antarmuka pengguna melalui web browser. Lapisan kedua adalah lapisan logika bisnis (business logic tier) yang memproses aturan, alur kerja, dan komputasi yang mendefinisikan perilaku sistem. Lapisan ketiga adalah lapisan data (data tier) yang bertanggung jawab atas penyimpanan dan pengambilan data secara persisten melalui PostgreSQL. Pemisahan ini memastikan bahwa perubahan pada satu lapisan tidak berdampak langsung pada lapisan lain, sehingga sistem dapat dikembangkan, diuji, dan dipelihara secara modular. Berikut adalah gambar tentang multitier yang terdapat pada Odoo.



Gambar 2.3 Gambar Multitier Architecture

Sumber : https://www.odoo.com/

Sebagai *framework* aplikasi bisnis, Odoo menegakkan prinsip separation of concerns melalui pola arsitektur Model-View-Controller (MVC) yang terwujud secara fisik dalam struktur direktori setiap custom addon. Lapisan Model bertanggung jawab atas representasi data dan logika bisnis inti, lapisan View menangani definisi antarmuka pengguna secara deklaratif, sedangkan lapisan Controller menjembatani komunikasi keduanya. Pemilihan Python sebagai bahasa pemrograman utama pada lapisan Model bertujuan agar pengembang dapat dengan mudah melakukan kustomisasi modul sendiri maupun melakukan konfigurasi modul yang sudah ada, sementara PostgreSQL menjadi fondasi penyimpanan seluruh data yang dihasilkan lintas modul (Kurniawan et al., 2021). Berikut gambar mengenai alur MVC yang terjadi.



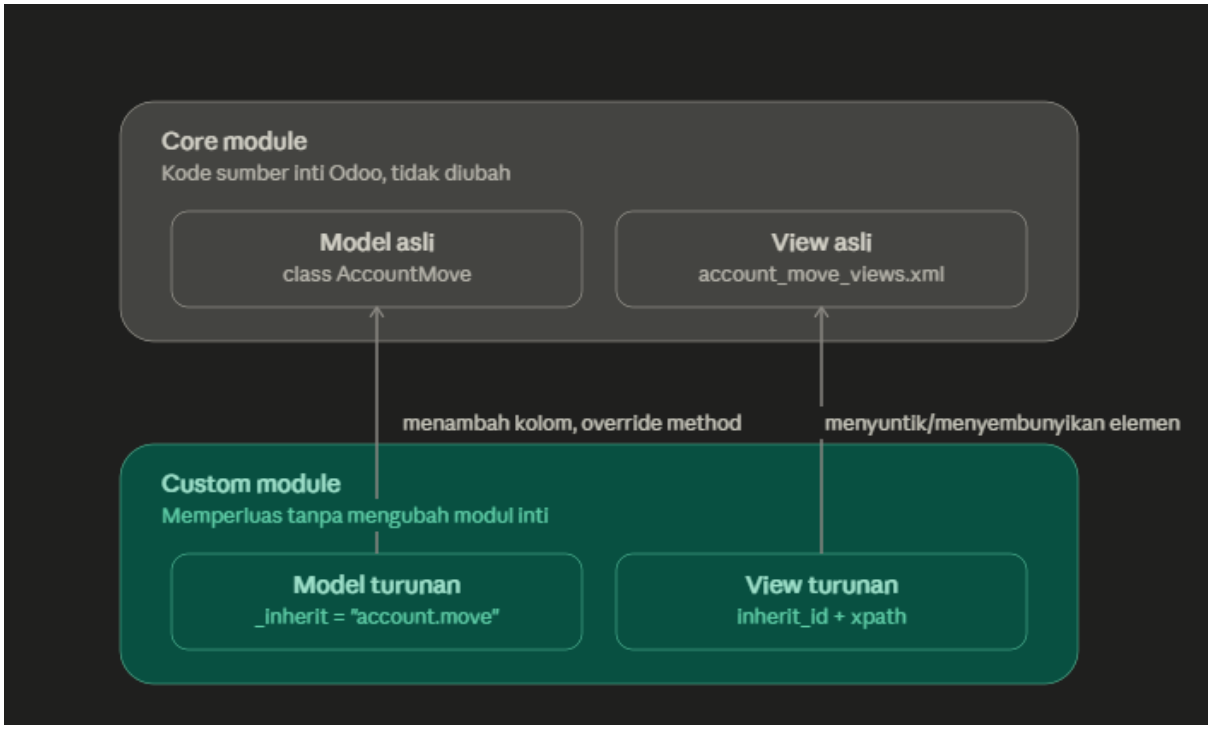
Gambar 2.4 Alur MVC

Lapisan Model dibangun menggunakan kelas Python yang mewarisi models.Model. Mekanisme Object-Relational Mapping (ORM) bawaan Odoo bertindak sebagai abstraksi yang memetakan objek Python menjadi tabel fisik pada PostgreSQL secara transparan tanpa penulisan query SQL secara manual. Selain menyediakan tipe data standar (Char, Integer, Many2one, One2many), ORM Odoo mendukung computed *field* yang dikalkulasi dinamis

melalui fungsi Python dengan dekorator @api.depends. Guna menjamin integritas data dari manipulasi sisi klien maupun penetrasi API programatik, dekorator @api.constrains diterapkan sebagai validasi berlapis di tingkat server sebelum data disimpan ke basis data. Keseluruhan mekanisme ini berjalan di atas arsitektur Python dan PostgreSQL yang menjadi pondasi teknis Odoo (Kurniawan et al., 2021).

Lapisan View mengelola penyajian data secara deklaratif melalui dokumen XML terpisah, sehingga penyesuaian antarmuka tidak memengaruhi logika bisnis di lapisan Model. Odoo menyediakan variasi tampilan sesuai skenario interaksi, meliputi Form View untuk manipulasi satu *record*, Tree/List View untuk representasi tabular, serta Search View untuk penyaringan dan pengelompokan data. Seluruh dokumen XML view didaftarkan dalam berkas manifest__.py agar dimuat secara sekuensial saat modul diinstalasi atau diperbarui. Pengendalian perilaku antarmuka secara dinamis dilakukan melalui atribut kondisional attrs atau ekspresi logika invisible, required, dan readonly yang mengevaluasi nilai suatu *field* secara real-time. Konfigurasi tampilan pada tingkat ini dapat dilakukan melalui *developer* mode bawaan Odoo untuk kebutuhan seperti pengubahan role pengguna dan penghilangan menu yang tidak relevan (Arfikriyana & Suhendi, 2022).

Kapasitas modular Odoo dioptimalkan lebih lanjut melalui mekanisme *inheritance* yang memungkinkan perluasan fungsionalitas tanpa mengubah kode sumber inti (core modules), sehingga kustomisasi tetap terisolasi dan sistem aktif tidak terganggu saat pembaruan versi dilakukan. Pada lapisan Model, ekstensi dilakukan melalui atribut inherit untuk memperluas kelas secara in-place, menambah kolom baru pada tabel basis data terkait, atau melakukan override method bawaan secara global. Pada lapisan View, modifikasi memanfaatkan ekspresi <xpath> untuk menavigasi hierarki dokumen XML induk secara selektif guna menyuntikkan atau menyembunyikan elemen antarmuka tanpa menduplikasi definisi view aslinya. Pendekatan kustomisasi berbasis kode menawarkan cakupan yang lebih luas karena Odoo dirancang sebagai platform yang fleksibel dengan modul-modul yang dapat dikustomisasi dan dikonfigurasi sesuai kebutuhan spesifik setiap perusahaan. Berikut gambar mengnai mekanisme inheritanc.



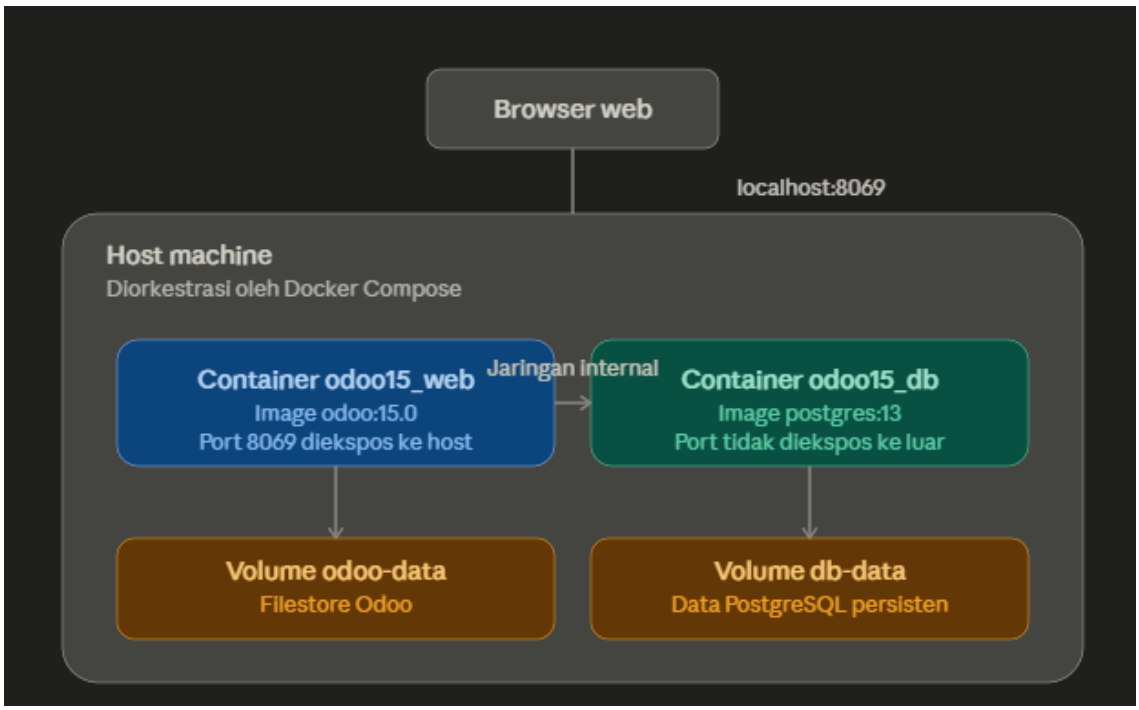
Gambar 2.5 Mekanisme *Inheritance*

2.3 Docker dan Docker Compose

Perbedaan lingkungan pengembangan akibat variasi konfigurasi sistem operasi, versi dependensi, dan pengaturan runtime antar-mesin merupakan hambatan signifikan dalam rekayasa perangkat lunak berbasis tim. Docker mengatasi permasalahan ini melalui teknologi kontainerisasi (containerization) yang mengemas aplikasi beserta seluruh dependensinya ke dalam satu unit terisolasi yang disebut container. Pendekatan ini menjamin aplikasi berjalan secara konsisten di atas mesin mana pun yang memiliki Docker Engine tanpa bergantung pada konfigurasi sistem host yang mendasarinya (Khalida et al., 2019). Berbeda dari virtualisasi tradisional yang mensyaratkan salinan penuh sistem operasi untuk setiap instansi sehingga konsumsi sumber daya menjadi sangat besar, container hanya berbagi kernel sistem operasi host sehingga jauh lebih ringan, proses boot berlangsung dalam hitungan detik, dan lebih efisien dalam penggunaan memori (Megantara, Alzami, Pramunendar, & Prabowo, 2022). Setiap container beroperasi dalam ruang proses dan *file*system yang sepenuhnya terisolasi satu sama lain, yang berarti dependensi, konfigurasi, maupun kegagalan pada satu container tidak akan merambat dan memengaruhi container lain yang berjalan dalam lingkungan yang sama (Khalida et al., 2019).

Ketika sebuah sistem terdiri dari beberapa layanan yang saling bergantung, menjalankan dan mengoordinasikan setiap container secara manual menjadi tidak praktis dan

rawan kesalahan konfigurasi. Docker Compose hadir untuk mengatasi permasalahan ini dengan menyediakan mekanisme orkestrasi multi-container melalui satu berkas konfigurasi deklaratif berbasis YAML bernama docker-compose.yml (Megantara et al., 2022). Melalui berkas tunggal ini, seluruh layanan yang dibutuhkan oleh sistem didefinisikan secara terpusat beserta jaringan virtual yang menghubungkannya dan urutan ketergantungan antar-layanan sehingga seluruh lingkungan dapat diaktifkan maupun dihentikan sekaligus dengan satu perintah tanpa harus mengelola setiap container secara terpisah (Alamsyah, Wahanani, & Idhom, 2021). Satu berkas definisi yang sama dapat didistribusikan ke seluruh anggota tim untuk memastikan setiap pengembang menjalankan lingkungan dengan konfigurasi yang benar-benar identik, sehingga perbedaan perilaku sistem akibat perbedaan konfigurasi lokal dapat dieliminasi sepenuhnya. Berikut adalah gambar dari kontainer docker



Gambar 2.6 Kontainer Odoo didalam Docker

Mekanisme volume yang disediakan Docker Compose memastikan data yang dihasilkan oleh layanan tetap tersimpan secara persisten di sistem host meskipun container dihentikan, di-restart, atau diganti dengan versi baru, sehingga tidak ada data yang hilang akibat siklus hidup container yang bersifat sementara (Alamsyah et al., 2021). Selain mekanisme penyimpanan, Docker Compose juga membentuk jaringan virtual internal yang memungkinkan seluruh container berada dalam satu jaringan dan dapat terhubung satu sama lain, sehingga komunikasi antar-layanan dapat berlangsung tanpa perlu dikonfigurasi secara manual oleh pengembang (Alamsyah et al., 2021).

2.4 PostgreSQL 13

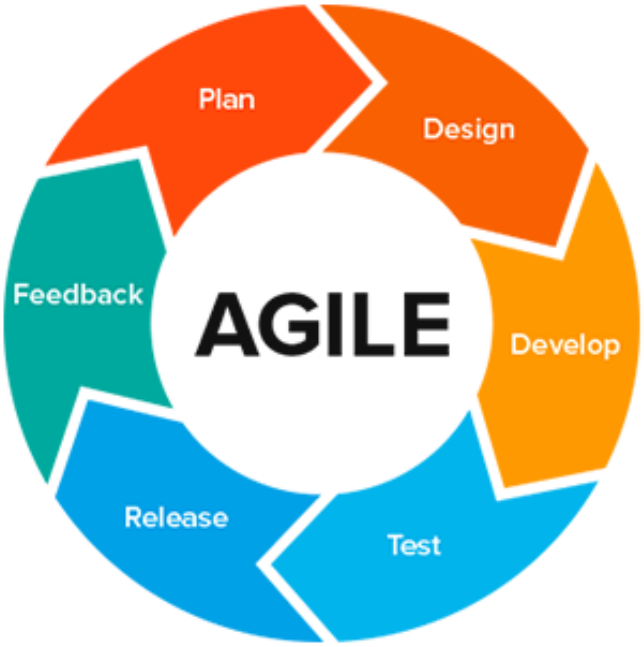
PostgreSQL merupakan sistem manajemen basis data (database management system) berbasis open-source yang digunakan oleh Odoo sebagai fondasi penyimpanan seluruh data operasionalnya (Arfikriyana & Suhendi, 2022). Pilihan Odoo untuk bertumpu pada PostgreSQL bukan sekadar preferensi teknis, melainkan keputusan arsitektural yang konsisten di seluruh implementasinya Odoo menggunakan Python sebagai bahasa pemrograman utama dan PostgreSQL sebagai basis datanya, sehingga keduanya membentuk satu tumpukan teknologi yang terpadu dan saling mendukung (Kurniawan et al., 2021). Posisi PostgreSQL sebagai basis data yang didukung oleh Odoo menjadikannya komponen infrastruktur yang tidak dapat digantikan dalam ekosistem pengembangan maupun operasional sistem ERP berbasis platform ini.

Dalam ekosistem pengembangan Odoo, PostgreSQL beroperasi sepenuhnya di balik lapisan abstraksi ORM sehingga pengembang modul tidak berinteraksi dengan basis data secara langsung. Setiap kelas model Python yang didefinisikan dalam modul Odoo dipetakan secara otomatis menjadi skema fisik di PostgreSQL pada saat modul diinstalasi atau diperbarui, tanpa memerlukan penulisan skrip DDL (Data Definition Language) secara manual. Pemisahan tanggung jawab ini memungkinkan pengembang modul berkonsentrasi sepenuhnya pada logika bisnis di lapisan Python, sementara kompleksitas pengelolaan skema relasional ditangani oleh lapisan *framework* secara transparan (Kurniawan et al., 2021).

2.5 Metode Pengembangan Agile

Agile merupakan metodologi pengembangan perangkat lunak yang berfokus pada pengiriman nilai secara bertahap melalui siklus iterasi yang pendek dan adaptif. Berbeda dari pendekatan waterfall yang mengharuskan penyelesaian seluruh fase perencanaan dan spesifikasi sebelum implementasi dimulai, Agile memandang perubahan kebutuhan sebagai bagian alami dari proses pengembangan dengan membangun mekanisme umpan balik ke dalam setiap siklus kerjanya (Sunardi & Fadli, 2018). Landasan filosofis metodologi ini dituangkan dalam Agile Manifesto yang menekankan empat prioritas utama: individu dan interaksi di atas proses dan alat, perangkat lunak yang berfungsi di atas dokumentasi yang komprehensif, kolaborasi dengan klien di atas negosiasi kontrak, serta respons terhadap perubahan di atas mengikuti rencana yang sudah ditetapkan (Sunardi & Fadli, 2018). Keempat prioritas ini tidak dimaksudkan untuk meniadakan proses, dokumentasi, kontrak, maupun perencanaan,

melainkan untuk menetapkan hierarki nilai ketika terjadi konflik antara keduanya dalam situasi pengembangan yang nyata. Berikut adalah gambar dari Alur metode Agile.



Gambar 2.7 Alur Agile

Sumber : https://aguayo.co

Dalam praktiknya, Agile diimplementasikan melalui siklus kerja iteratif di mana setiap iterasi menghasilkan perangkat lunak yang dapat diverifikasi dan dievaluasi secara langsung oleh pemangku kepentingan. Agile dipilih karena memiliki sifat adaptif dan responsif terhadap perubahan, dan terbukti memiliki tingkat keberhasilan proyek yang lebih tinggi dibandingkan metode waterfall studi yang dirilis oleh The Standish Group menunjukkan tingkat sukses Agile mencapai 42% sementara waterfall hanya 14% (Murdiani, Yudhana, & Sunardi, 2020). Mekanisme umpan balik yang pendek antara pengembang, penguji, dan pemangku kepentingan menjadi ciri khas yang membedakan Agile dari pendekatan lainnya, karena koreksi dan penyesuaian dapat dilakukan segera setelah evaluasi tanpa harus menunggu seluruh sistem selesai dibangun. Pendekatan iteratif ini terbukti efektif dalam berbagai konteks pengembangan perangkat lunak, termasuk pada proyek yang bersifat pemeliharaan dan penyempurnaan sistem yang sudah berjalan, di mana kebutuhan bersifat reaktif terhadap temuan operasional dan tidak selalu dapat direncanakan jauh di muka (Murdiani et al., 2020); (Sunardi & Fadli, 2018).

2.6 Tinjauan Pustaka

Kurniawan S, Yulia, dan Irawan (2021) dalam penelitiannya tentang kustomisasi dan implementasi Odoo ERP pada perusahaan manufaktur PT. X menguraikan proses penyesuaian modul Human Resources Odoo agar selaras dengan kebutuhan operasional dan peraturan ketenagakerjaan Indonesia. Penelitian tersebut menerapkan konfigurasi modul-modul bawaan Odoo untuk area penjualan, pembelian, dan persediaan, serta mengembangkan modul kustom untuk area HR yang mencakup pengelolaan absensi, lembur, penggajian, dan surat peringatan. Pengembangan modul kustom dilakukan menggunakan Python dan PostgreSQL sebagai fondasi teknis platform Odoo, dengan pendekatan yang memanfaatkan fleksibilitas open-source Odoo untuk mengakomodasi kebutuhan spesifik perusahaan.

Arfikriyana dan Suhendi (2022) dalam penelitiannya tentang analisis dan implementasi modul akuntansi pada aplikasi ERP berbasis open-source menguraikan penerapan modul Accounting Odoo pada perusahaan konsultan pajak PT Delta Solusi Konsultama. Penelitian tersebut mencakup konfigurasi fitur-fitur akuntansi inti seperti pembuatan invoice, pencatatan jurnal, rekonsiliasi akun, dan pelaporan keuangan. Penyesuaian tampilan sistem dilakukan melalui *developer* mode bawaan Odoo untuk menghilangkan menu yang tidak relevan dengan proses bisnis perusahaan dan mengatur hak akses pengguna sesuai peran masing-masing dalam organisasi.

Yanaa, Sasmita, dan Susila (2022) dalam penelitiannya tentang penerapan ERP menggunakan Odoo 14 pada usaha garmen Club Ink Bali mendokumentasikan implementasi Odoo secara menyeluruh pada skala UMKM dengan mengaktifkan tujuh modul sekaligus, yaitu Sales, Purchase, Inventory, Manufacturing, Invoice, Point of Sale, dan Website. Penelitian tersebut menunjukkan bahwa Odoo mampu mengintegrasikan seluruh fungsi operasional dari produksi hingga penjualan dalam satu platform, dan menegaskan bahwa platform ini tersedia sebagai solusi open-source yang dapat diunduh dan digunakan secara gratis oleh organisasi dari berbagai skala.

BAB III
PELAKSANAAN MAGANG

3.1 Manajemen Proyek

Subbab ini menjelaskan bagaimana pengelolaan pekerjaan selama kegiatan magang berlangsung, mulai dari informasi penempatan dan lingkungan kerja, alur kerja yang diterapkan dari penerimaan *task* hingga serah terima hasil pekerjaan, tools yang digunakan selama proses pengembangan, hingga ringkasan proyek yang dikerjakan selama periode magang

3.1.1 Penempatan Kerja

Kegiatan magang dilaksanakan di PT Gamatechno Indonesia yang berlokasi di Yogyakarta selama enam bulan, terhitung sejak September 2025 hingga Maret 2026. PT Gamatechno Indonesia menerapkan sistem kerja hybrid yang membagi hari kerja antara Work From Office (WFO) dan Work From Anywhere (WFA). Berdasarkan kebijakan yang berlaku selama pelaksanaan magang, hari kerja WFO ditetapkan pada Selasa dan Kamis, sementara hari Senin,Rabu dan Jumat dilaksanakan secara WFA. Jam kerja berlangsung mulai pukul 08.30 hingga 17.00 WIB pada seluruh hari kerja, baik saat WFO maupun WFA. Sistem kerja ini memberikan fleksibilitas yang cukup bagi penulis untuk tetap produktif dari luar kantor tanpa mengorbankan intensitas koordinasi dengan tim.

Selama masa magang, penulis ditempatkan di bawah koordinasi langsung seorang *Project* Manager yang bertanggung jawab atas pengelolaan proyek dan distribusi penugasan kepada seluruh anggota tim. Selain *Project* Manager, penulis juga berinteraksi secara rutin dengan *tester* yang berperan dalam proses pengujian hasil pekerjaan, serta mentor teknis yang memberikan arahan dan bimbingan pada tahap awal adaptasi. Koordinasi harian dilakukan melalui media komunikasi digital, sementara diskusi teknis yang membutuhkan penjelasan lebih mendalam diselenggarakan melalui video conference secara berkala. Pola komunikasi ini memungkinkan alur kerja tetap berjalan secara teratur meskipun tidak seluruh anggota tim hadir di kantor pada waktu yang sama.

Proses adaptasi pada awal masa magang mencakup pemahaman terhadap alur kerja internal perusahaan, standar pengelolaan pekerjaan yang berlaku, serta mekanisme kolaborasi tim yang sudah diterapkan sebelum penulis bergabung. Fase ini membutuhkan waktu beberapa

pekan sebelum penulis dapat bekerja secara mandiri dengan ritme yang lebih produktif. Penugasan diberikan secara bertahap sesuai dengan kebutuhan proyek yang sedang berjalan dan hasil koordinasi antara *Project* Manager dengan tim teknis. Selama periode magang, penulis terlibat dalam beberapa proyek yang berbeda secara bergantian, dengan tingkat keterlibatan dan durasi yang bervariasi tergantung pada prioritas proyek dan kebutuhan tim pada masing-masing periode.

3.1.2 Alur Kerja Agile

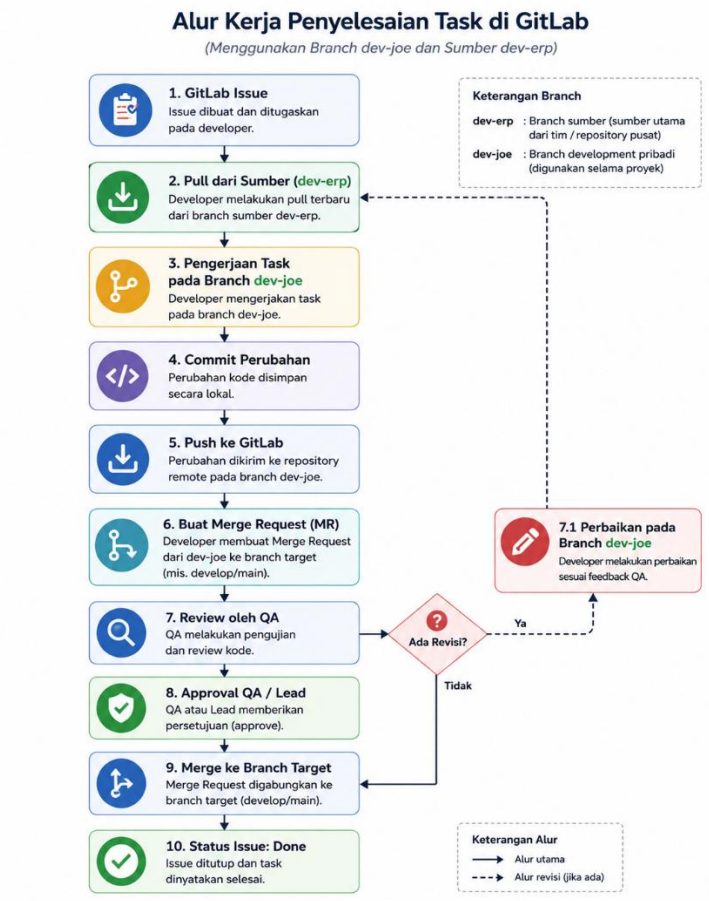
Pengelolaan seluruh pekerjaan selama masa magang mengadopsi pendekatan iterative issue-driven development yang terpusat pada platform GitLab sebagai sistem pelacak aktivitas tim. Setiap unit pekerjaan direpresentasikan sebagai sebuah Issue di GitLab yang dibuat oleh *Project* Manager atau anggota tim terkait, memuat deskripsi masalah atau spesifikasi kebutuhan yang perlu diselesaikan beserta penugasan kepada pengembang yang bertanggung jawab.

Setiap Issue merepresentasikan satu siklus iterasi yang bergerak melalui serangkaian status secara progresif: To Do saat Issue baru dibuat dan belum dikerjakan, In Progress saat pengembang mulai mengerjakan, *Review* saat hasil pekerjaan diajukan untuk diperiksa, Done Test setelah pengujian oleh QA dinyatakan berhasil, dan Done saat Issue dinyatakan selesai sepenuhnya. Pergerakan status ini mencerminkan posisi aktual pekerjaan dalam alur kerja tim dan membentuk visibilitas progres yang transparan tanpa memerlukan pelaporan manual terpisah.

Dalam praktiknya, setiap siklus dimulai dengan sinkronisasi kode dari *branch* utama repositori GitLab untuk memastikan basis kode lokal sudah sesuai dengan versi terbaru tim, dilanjutkan dengan pembuatan *branch* baru yang terisolasi dari *branch* utama. Isolasi ini memastikan bahwa proses pengembangan dalam satu iterasi tidak memengaruhi stabilitas sistem yang sedang berjalan. Tim QA/*Tester* berperan aktif tidak hanya sebagai penguji di akhir siklus, tetapi juga sebagai jembatan komunikasi yang menyampaikan konteks dan esensi *task* secara langsung kepada pengembang memastikan pemahaman terhadap kebutuhan yang sesungguhnya tersampaikan sejak awal iterasi. Setelah pengembangan selesai dilakukan secara lokal dan diverifikasi melalui pengujian mandiri di lingkungan Docker, perubahan di-commit dan di-push ke repositori, kemudian penulis membuka Merge Request sebagai penanda formal bahwa iterasi siap memasuki fase validasi.

Fase validasi dijalankan oleh tim QA melalui pengujian fungsional terhadap perubahan yang diajukan. Apabila ditemukan ketidaksesuaian, Issue dikembalikan kepada pengembang beserta catatan evaluasi yang spesifik untuk diperbaiki dalam sub-iterasi berikutnya. Proses ini berlangsung hingga seluruh aspek pengujian dinyatakan lolos. Setelah QA memberikan konfirmasi, lead proyek melakukan approval akhir dan mengeksekusi merge ke *branch* utama, menandai penutupan satu siklus iterasi secara penuh.

Sebelum perubahan diajukan melalui Merge Request, penulis juga menambahkan komentar () pada bagian-bagian kode yang mengalami perubahan, terutama pada logika bisnis yang memiliki keterkaitan dengan beberapa modul lain. Dokumentasi pada tingkat kode ini bertujuan untuk memudahkan proses peninjauan oleh tim QA, membantu pengembang lain memahami alasan implementasi yang dilakukan, serta mendukung proses pemeliharaan apabila perubahan tersebut perlu dikembangkan kembali pada iterasi berikutnya. Berikut adalah gambar dari alur kerja penyelesaian task.



Gambar 3.1 Alur Kerja Pemberian *Task*

3.1.3 Aktivitas Pendukung Selama Magang

Selain melaksanakan *task* yang diberikan, penulis juga melakukan proses adaptasi terhadap lingkungan kerja dan alur pengembangan perangkat lunak yang diterapkan di PT Gamatechno Indonesia. Pada masa awal magang, penulis mempelajari struktur tim, mekanisme komunikasi, serta alur kerja pengembangan yang menggunakan metodologi Agile. Adaptasi juga dilakukan dengan membiasakan diri menggunakan berbagai tools yang digunakan dalam proses pengembangan, seperti GitLab untuk manajemen repositori, Docker sebagai lingkungan pengembangan, dan Odoo sebagai platform utama yang dikembangkan. Ketika mengalami kendala dalam memahami struktur modul maupun logika bisnis yang diterapkan, penulis secara aktif berdiskusi dengan mentor dan anggota tim sehingga proses adaptasi terhadap lingkungan kerja dapat berlangsung dengan baik.

Selama pelaksanaan magang, penulis juga berinteraksi secara aktif dengan mentor maupun anggota tim pengembang, baik pada saat diskusi penyelesaian *task*, proses *code review*, rapat tim, maupun percakapan informal di lingkungan kerja. Melalui interaksi tersebut, penulis memperoleh berbagai wawasan yang tidak hanya berkaitan dengan aspek teknis pengembangan perangkat lunak, tetapi juga mengenai budaya kerja di industri, pentingnya komunikasi yang efektif dalam tim, cara menyampaikan kendala maupun usulan solusi dalam forum diskusi, serta bagaimana keputusan teknis dipertimbangkan berdasarkan kebutuhan bisnis dan dampaknya terhadap pengguna sistem. Selain itu, penulis juga menerapkan masukan yang diperoleh dari mentor dengan menambahkan code comments pada bagian-bagian kode yang memiliki logika bisnis atau alur proses yang kompleks sehingga perubahan yang dilakukan lebih mudah dipahami oleh pengembang lain maupun peserta magang berikutnya.

Ketika menunggu hasil pengujian dari tim *testing* maupun penugasan *task* berikutnya, penulis memanfaatkan waktu tersebut untuk melakukan pembelajaran mandiri (*self learning*) guna memperdalam pemahaman terhadap Odoo dan teknologi yang digunakan selama magang. Aktivitas tersebut dilakukan dengan mempelajari dokumentasi resmi Odoo melalui situs web Odoo Documentation, membaca referensi terkait fitur yang sedang dikembangkan, serta memahami mekanisme implementasi yang relevan dengan pekerjaan yang sedang dilakukan. Meskipun mentor sempat memberikan arahan mengenai beberapa modul yang dapat dipelajari, pembelajaran penulis lebih banyak difokuskan pada dokumentasi resmi Odoo dan materi yang mendukung penyelesaian *task* yang sedang dikerjakan. Melalui proses pembelajaran mandiri tersebut, penulis memperoleh pemahaman yang lebih baik mengenai konsep dasar Odoo, pola

pengembangan modul, serta praktik implementasi yang sesuai dengan kebutuhan sistem ERP Ganesha Operations.

3.1.4 Tools yang Digunakan

Selama pelaksanaan magang, beberapa alat digunakan secara rutin untuk mendukung proses pengembangan dan koordinasi tim. GitLab digunakan sebagai platform utama untuk manajemen repositori kode, pelacakan issue, dan pengelolaan Merge Request. Docker digunakan untuk menjalankan lingkungan pengembangan Odoo secara lokal agar konsisten dengan *environment* yang digunakan tim pada proyek ERP Ganesha Operations. XAMPP digunakan sebagai lingkungan pengembangan lokal untuk proyek bantuUMKM.com berbasis React.js, sementara Laragon digunakan untuk proyek i-BOSS Nasmoco berbasis Laravel. Visual Studio Code digunakan sebagai editor kode utama dalam pengerjaan seluruh proyek. Komunikasi harian dengan tim dilakukan melalui media komunikasi digital, sementara diskusi teknis yang lebih mendalam dilaksanakan melalui video conference secara berkala.

3.1.5 Daftar Proyek dan Hasil

Selama enam bulan magang, penulis terlibat dalam tiga proyek dengan tingkat keterlibatan yang berbeda-beda. Ringkasan proyek dan hasil yang dicapai disajikan pada tabel 3.1 berikut.

Tabel 3.1 Daftar Proyek Saat Magang

No	Proyek	Tech Stack	Keterlibatan	Hasil
1	Optimalisasi ERP Ganesha Operations	Odoo15, Python	Proyek utama	Menyelesaikan 8 *task* optimalisasi modul, dengan 6 *task* telah berhasil diverifikasi oleh tim QA dan 2 task telah di-*merge* ke repositori utama untuk dilanjutkan proses pengembangan dan verifikasi oleh tim pengembang
2	bantuUMKM.com	React.js	Proyek training	Mengembangkan fitur ***login***, ***dashboard***, serta operasi **CRUD** pada modul **Sales**, **Reports**, dan **Employees** sesuai kebutuhan proyek pelatihan.
3	i-BOSS Nasmoco	Laravel	Proyek pendukung	Melakukan pembaruan logika modal *datatable*, menambahkan fitur lampiran pada halaman **User Request**, serta memperbaiki *bug* pada fitur **Edit Vendor Group**.

3.1.6 Daftar Tugas dan Proyek

Proyek 1: Optimalisasi ERP Ganesha Operations (Odoo 15)

Proyek ini merupakan proyek utama yang dikerjakan penulis selama masa magang. Sistem ERP Ganesha Operations merupakan sistem informasi terintegrasi berbasis Odoo 15 yang dikembangkan dan dipelihara oleh PT Gamatechno Indonesia untuk klien Ganesha Operations, sebuah lembaga bimbingan belajar berskala nasional. Penulis terlibat dalam fase pemeliharaan dan optimalisasi berkelanjutan sistem yang sudah berjalan aktif, dengan mengerjakan perbaikan dan penyesuaian pada lapisan *custom addons* melalui mekanisme *inheritance* tanpa menyentuh kode inti Odoo. Berikut merupakan tabel 3.2 terkait rincian *task* yang dilakukan .

Tabel 3.2 Daftar Tugas pada Proyek Optimalisasi ERP Ganesha Operations

No	Nama Tugas	Deskripsi	Hasil
1	Perbaikan *Error Field* Modul Jurnal Pengeluaran	*Field* journal_id memiliki atribut required=True namun tidak ditampilkan di antarmuka, menyebabkan *error* saat pengguna mencoba menyimpan form	Atribut required dipindahkan ke lapisan *view* menggunakan attrs sehingga validasi hanya aktif saat *field* ditampilkan. Terselesaikan dan lolos QA
2	Validasi *Mandatory* Modul Karyawan (Promosi)	Formulir Promosi Karyawan tidak memiliki validasi *mandatory* pada *field* jabatan asal, hasil evaluasi, dan jabatan tujuan sehingga data bisa tersimpan tidak lengkap	Validasi *mandatory* ditambahkan melalui *view inheritance* pada ketiga *field*. Terselesaikan tanpa menyentuh kode Python
3	Validasi *Mandatory* Master Data Diskon	*Field-field* penting pada master data diskon tidak memiliki validasi *mandatory* sehingga data diskon bisa tersimpan dalam kondisi tidak lengkap	Validasi *mandatory* ditambahkan pada *field-field* yang diperlukan. Terselesaikan dan lolos QA
4	Penyembunyian Tombol Modul Entri Jurnal (Issue #538)	Opsi *Surat Instruksi Setor* masih muncul di menu *Print* padahal sudah tidak digunakan dalam proses bisnis	Tombol disembunyikan menggunakan *view inheritance* dan attrs. Terselesaikan dan lolos QA
5	Penyembunyian Tombol Modul Entri Jurnal (Issue #539)	Tombol *Kirim Surat Penyetoran* masih muncul pada state yang tidak seharusnya	Tombol disembunyikan berbasis kondisi *state* menggunakan attrs. Terselesaikan dan lolos QA
6	Penyesuaian Logic Nominal Belum Disetor & *Filter* Metode Pembayaran	Kalkulasi nominal belum disetor tidak akurat dan metode pembayaran tunai perlu di*filter* dari opsi yang tersedia	Implementasi selesai dan telah di-*merge* ke repositori utama. Pengujian menyeluruh dilanjutkan oleh tim karena

			kompleksitas integrasi yang tinggi
7	Perbaikan Validasi Nomor Telepon Modul Kontak	Fungsi validasi nomor telepon yang digabung dalam satu fungsi menyebabkan konflik validasi antar *field*	Fungsi validasi dipisah per *field* menggunakan @api.constrains. Terselesaikan dan lolos QA
8	*Filter* Produk *Expired* Modul BIMKER	Produk yang sudah *expired* masih muncul dalam daftar pilihan pada form BIMKER	Implementasi *filter* selesai dan telah di-*merge* ke repositori utama. Pengujian menyeluruh dilanjutkan oleh tim karena kompleksitas integrasi yang tinggi

Proyek 2: bantuUMKM.com

Proyek ini merupakan proyek pelatihan yang dikerjakan pada fase awal masa magang sebelum penulis ditempatkan pada proyek aktif. Sistem ini dirancang sebagai platform *dashboard* manajemen usaha berbasis web. Penulis bertanggung jawab penuh atas pengembangan sisi *frontend* menggunakan React.js, sementara sisi *backend* dikerjakan oleh rekan magang. Berikut merupakan Tabel 3.3 yang berisi daftar *task* yang dikerjakan dalam *task* ini.

Tabel 3.3 Daftar Tugas pada Proyek bantuUMKM.com

No	Nama Tugas	Deskripsi	Hasil
1	Pengembangan Halaman *Login*	Membangun halaman *login* dengan validasi input di sisi *frontend* dan autentikasi berbasis JWT	Halaman *login* selesai dengan validasi dan navigasi ke *dashboard* setelah berhasil masuk
2	Pengembangan *Dashboard*	Membangun halaman *dashboard* dengan empat kartu ringkasan bisnis dan *sidebar* navigasi yang konsisten di seluruh halaman	*Dashboard* selesai dengan tampilan empat kartu ringkasan dan navigasi antar halaman
3	Fitur CRUD Modul Sales	Membangun antarmuka pengelolaan data penjualan dengan tabel, pencarian, dan fitur impor/ekspor Excel	Fitur CRUD modul Sales selesai dan berjalan sesuai spesifikasi
4	Fitur CRUD Modul Reports	Membangun antarmuka pengelolaan data laporan dengan pola antarmuka yang seragam dengan modul lainnya	Fitur CRUD modul Reports selesai dan berjalan sesuai spesifikasi
5	Fitur CRUD Modul Employees	Membangun antarmuka pengelolaan data karyawan dengan formulir yang lebih kompleks termasuk fitur unggah foto	Fitur CRUD modul Employees selesai dengan delapan *field* dan pratinjau gambar

Proyek 3: i-BOSS Nasmoco (Laravel)

Proyek ini merupakan proyek pendukung yang dikerjakan setelah proyek pelatihan. i-BOSS Nasmoco adalah aplikasi *back office* berbasis Laravel yang dikelola PT Gamatechno Indonesia untuk klien Nasmoco. Keterlibatan penulis pada proyek ini lebih terbatas dibandingkan proyek Odoo, namun memberikan pengalaman teknis dalam pengembangan berbasis *framework* PHP.Berikut adalah tabel 3.4 yang berisi rincian *task* dalam *project* ini.

Tabel 3.4 Daftar Tugas pada Proyek i-BOSS Nasmoco

No	Nama Tugas	Deskripsi	Hasil
1	Pembaruan Logic Modal Datatable dan *Filter* Menu User Request	Empat kartu status didefinisikan secara terpisah dengan struktur HTML yang berulang, dan halaman belum memiliki modal detail maupun mekanisme *filter*	Kartu direfaktor menggunakan loop @foreach, modal detail dan *filter dropdown* dengan empat parameter ditambahkan. Terselesaikan
2	Penambahan Fitur Lampiran Halaman Detail User Request	Belum tersedia cara untuk melampirkan dokumen pendukung setelah permintaan dibuat	Tombol unggah dengan pengecekan hak akses dan endpoint /add-attachment/{id} berhasil ditambahkan. Implementasi pada halaman lain dilanjutkan oleh tim
3	Perbaikan *Bug* Edit Vendor Group	Sistem menampilkan *error* saat pengguna menyimpan data vendor tanpa mengisi *field* PIC karena kolom tidak dikonfigurasi sebagai *nullable*	*Bug* diperbaiki pada dua lapisan sekaligus. Terselesaikan dan lolos QA

3.2 Proyek Utama: Optimalisasi ERP Ganesha Operations (Odoo 15)

Proyek utama yang dikerjakan selama masa magang adalah optimalisasi sistem ERP Ganesha Operations, sebuah sistem informasi terintegrasi berbasis Odoo 15 yang dikembangkan dan dipelihara oleh PT Gamatechno Indonesia untuk klien Ganesha Operations. Sistem ini sudah berjalan aktif sebagai production system ketika penulis bergabung, sehingga seluruh pekerjaan dilakukan pada sistem yang sudah memiliki pengguna nyata, data transaksi yang terakumulasi, dan modul-modul yang saling terintegrasi. Pembahasan pada subbab-subbab berikut mencakup gambaran sistem, lingkungan pengembangan yang digunakan, serta *task-task* optimalisasi yang dikerjakan selama masa magang.

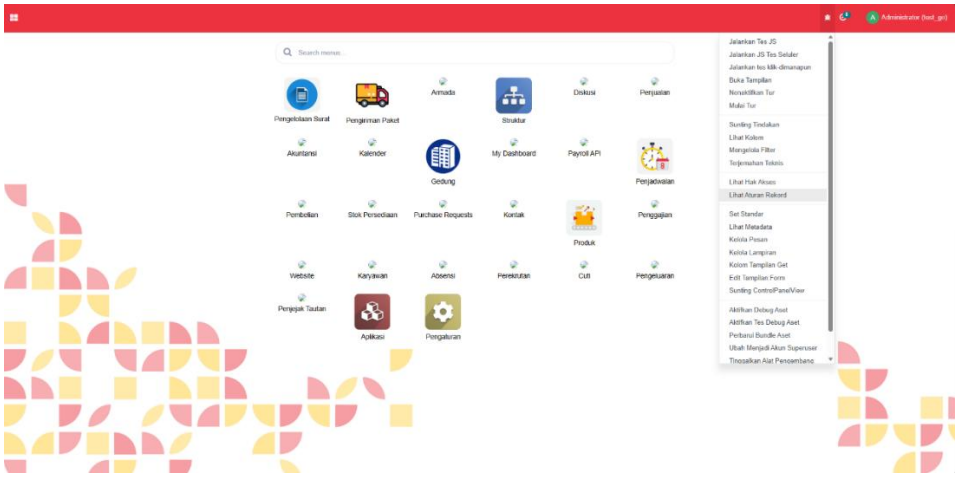
3.2.1 Gambaran Sistem ERP Ganesha Operations

Sistem ERP Ganesha Operations merupakan sistem informasi terintegrasi yang dikembangkan dan dipelihara oleh PT Gamatechno Indonesia untuk klien Ganesha Operations, sebuah lembaga pendidikan berskala nasional yang menjalankan operasional bimbingan belajar di berbagai cabang di Indonesia. Dalam relasi proyek ini, Ganesha Operations berperan sebagai pemilik sistem (client) yang menggunakan ERP untuk menjalankan proses bisnis operasionalnya sehari-hari, sementara PT Gamatechno Indonesia berperan sebagai vendor pengembang yang bertanggung jawab atas pemeliharaan dan optimalisasi sistem secara berkelanjutan.

Sistem ini sudah beroperasi penuh sebagai production system ketika penulis bergabung dalam tim pengembang pada bulan November 2025. Artinya, ERP tersebut sudah memiliki basis data transaksi yang terakumulasi selama beberapa periode operasional, sejumlah pengguna aktif dari berbagai cabang yang menjalankan sistem setiap hari, serta puluhan modul kustom yang sudah berjalan dan saling terintegrasi. Penulis tidak terlibat dalam pembangunan sistem dari awal, melainkan masuk dalam fase pemeliharaan dan optimalisasi berkelanjutan sebuah konteks pengembangan perangkat lunak yang secara fundamental berbeda dengan pengembangan green*field* yang umum ditemui di lingkungan akademik.

Sistem Ganesha Operations memanfaatkan berbagai modul Odoo, baik modul bawaan (core modules) seperti modul Account, HR, Contact, dan Inventory, maupun modul kustom yang dikembangkan secara spesifik untuk kebutuhan bisnis pendidikan seperti modul BIMKER (Bimbingan Belajar Kerjasama), modul Penyetoran, modul Expense Sheet, serta sejumlah modul pendukung lainnya. Seluruh modul kustom ditempatkan dalam satu direktori bernama ganesha-operation-app yang menjadi workspace utama bagi tim pengembang.

Selama masa magang, penulis melaksanakan aktivitas optimalisasi pada beberapa modul kustom tersebut berdasarkan *task* yang diberikan melalui repositori GitLab perusahaan. Optimalisasi yang dilakukan tidak menyentuh kode inti Odoo, basis data PostgreSQL, maupun infrastruktur Docker melainkan terfokus pada lapisan custom addons melalui mekanisme Python ORM dan XML view *inheritance*. Pemisahan tanggung jawab ini selaras dengan praktik rekayasa perangkat lunak yang baik pada sistem enterprise yang sudah aktif, di mana optimalisasi diisolasi dari komponen inti agar stabilitas sistem tetap terjaga selama proses pengembangan berlangsung. Berikut merupakan tampilan awal halaman Odoo dari *project* ini.



Gambar 3.2 Halaman Odoo

3.2.2 Arsitektur Dan Posisi Pengembangan Modul

Sistem Ganesha Operations dibangun di atas arsitektur client-server yang ditawarkan oleh Odoo 15, sebagaimana telah dijelaskan pada Subbab 2.2 dan 2.3. Tabel 3.5 berikut merangkum bagaimana komponen-komponen arsitektur tersebut terpetakan pada konteks sistem Ganesha Operations, sekaligus menunjukkan posisi pekerjaan penulis di antara komponen-komponen tersebut.

Tabel 3.5 Komponen Utama Arsitektur Client-Server Sistem

Komponen	Teknologi	Peran dalam Sistem
Client Layer	Browser Web	Antarmuka pengguna berbasis web odoo client; mengirim permintaan JSON-RPC ke server Odoo
Application Server	Odoo 15 (Python)	Memproses logika bisnis, ORM, routing, autentikasi, dan manajemen sesi
Persistence Layer	PostgreSQL 13	Menyimpan seluruh data operasional, struktur tabel dikelola oleh ORM Odoo
Containerization	Docker Compose	Mengorkestrasi container Odoo dan PostgreSQL dalam satu *environment* terisolasi
File Configuration	odoo.conf	Mendefinisikan parameter runtime Odoo: addons_path, koneksi DB, worker, port
Custom Modules	Python + XML	Implementasi logika bisnis kustom dan tampilan yang mewarisi modul inti Odoo

Berdasarkan pemetaan tersebut, seluruh aktivitas optimalisasi yang penulis lakukan selama magang berlangsung pada lapisan custom addons, yaitu direktori ganesha-operation-app yang berisi modul-modul kustom Ganesha Operations. Komponen lain seperti kode inti Odoo, server aplikasi, basis data PostgreSQL, dan infrastruktur Docker berada di luar lingkup pekerjaan penulis sehingga tidak dibahas lebih lanjut dalam laporan ini.

Struktur Custom Addons dan Ruang Kerja Penulis

Direktori ganesha-operation-app merupakan tempat seluruh modul kustom Ganesha Operations berada. Direktori ini disimpan dalam repositori Git yang terpisah dari kode inti Odoo, dan dimuat ke dalam container Odoo melalui mekanisme bind mount yang dijelaskan pada Subbab 3.2.3. Di dalamnya terdapat banyak subdirektori yang masing-masing merupakan modul kustom dengan struktur standar Odoo, seperti models/, views/, dan manifest__.py.

Selama magang, penulis melakukan optimalisasi pada beberapa modul dalam direktori tersebut. Modul-modul yang menjadi ranah pekerjaan penulis dirangkum pada tabel 3.6 berikut.

Tabel 3.6 Rincian *Task* yang dilakukan

Modul Kustom	Jenis Pekerjaan Utama	Lapisan Modifikasi
hr_expense_sheet	Perbaikan *error field* Jurnal Pengeluaran (Issue #439)	Python ORM
go_hr	Validasi *mandatory* Promosi & Demosi (Issue #518 #519)	XML View *Inheritance*
go_inventory	Validasi *mandatory* Master Data Diskon (Issue #527 #528)	XML View *Inheritance*
go_account	Penyembunyian tombol Entri Jurnal (Issue #538 #539)	XML View *Inheritance*
go_account & go_expense	Penyesuaian logic Nominal Belum Disetor (Issue #543 #553)	Python ORM + XML View
res.partner (extended)	Validasi format Nomor Telepon Modul Kontak	Python ORM
go_bimker	Validasi *field* master data berbasis state	XML View *Inheritance*

Pemilihan lapisan modifikasi pada setiap *task* disesuaikan dengan karakteristik masalah yang ditemukan. Perubahan yang berkaitan dengan logika bisnis dilakukan pada lapisan Python ORM, sedangkan penyesuaian validasi dan tampilan dilakukan melalui XML View atau XML View *Inheritance*, sebagaimana telah dijelaskan pada Subbab 2.2.4. Tabel di atas menjadi acuan modul yang akan dibahas lebih rinci pada Subbab 3.2.4 sampai 3.2.9.

3.2.3 Konfigurasi dan Pengoperasian *Environment* Docker

Pengembangan modul Odoo 15 selama magang dilakukan pada *environment* lokal berbasis Docker Compose yang konfigurasinya telah disiapkan oleh rekan satu tim proyek

Ganesha Operations. Penulis menggunakan *environment* ini secara langsung untuk pengembangan dan pengujian tanpa perlu menyusun konfigurasi infrastruktur dari awal.

Environment terdiri atas dua service utama, yaitu container basis data (postgres:13) dan container aplikasi (odoo:15.0), sebagaimana didefinisikan pada docker-compose.yml berikut.

Tabel 3.7 Konfigurasi Docker Compose

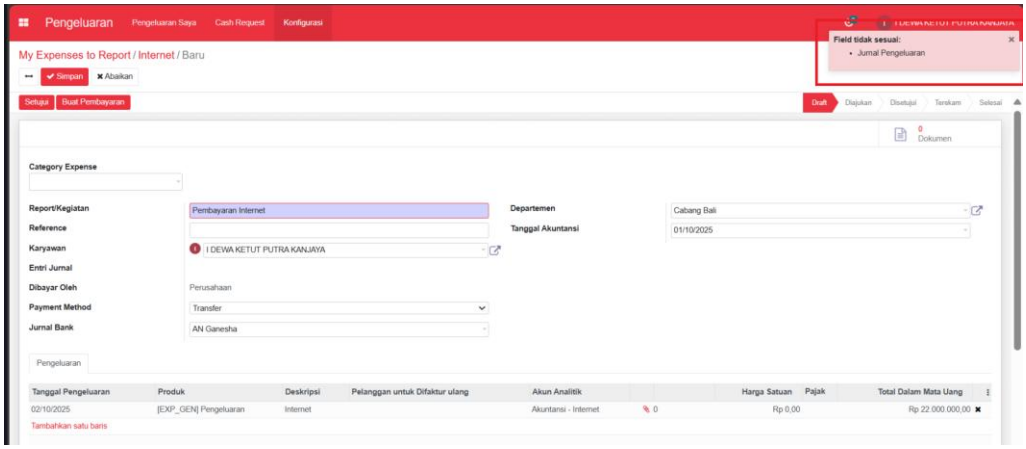
Service	Image	Fungsi Utama
db	postgres:13	Menyediakan layanan basis data PostgreSQL dengan penyimpanan data persisten menggunakan volume db-data.
odoo	odoo:15.0	Menjalankan aplikasi Odoo 15, memuat modul kustom melalui /mnt/extra-addons, serta menggunakan berkas konfigurasi odoo.conf.

Konfigurasi yang paling relevan dengan aktivitas pengembangan penulis adalah volume ./ganesha-operation-app:/mnt/extra-addons, yang memasang direktori custom addons ke dalam container melalui bind mount. Dengan konfigurasi ini, perubahan kode pada modul kustom yang dilakukan langsung dari laptop penulis akan tersinkronisasi otomatis ke dalam container tanpa perlu rebuild image, sehingga proses pengujian dapat dilakukan secara iteratif hanya dengan melakukan upgrade module atau restart service Odoo.

Dalam praktiknya, penulis memulai pekerjaan dengan menjalankan docker compose up -d, memastikan akses ke localhost:8069 berhasil, lalu memodifikasi modul kustom menggunakan Visual Studio Code dan menguji hasilnya melalui antarmuka Odoo pada basis data pengujian test_go_dev. Perubahan yang melibatkan logika bisnis atau struktur model memerlukan upgrade module agar diterapkan secara penuh, sedangkan perubahan pada *file* Python tertentu memerlukan restart service Odoo melalui docker compose restart odoo.

3.2.4 Perbaikan *Error Field* pada Modul Jurnal Pengeluaran

Permasalahan yang dilaporkan pada Issue #439 ditemukan pada form Jurnal Pengeluaran di sistem Ganesha Operations. Pengguna mengisi seluruh *field* yang tersedia pada form tersebut dan mencoba mengirimkan dokumen, namun sistem menampilkan notifikasi *error* yang menyatakan bahwa *field* Jurnal Pengeluaran belum diisi. Permasalahan ini membingungkan pengguna karena *field* yang dimaksud tidak muncul di antarmuka form sama sekali pengguna tidak dapat melihat *field* tersebut, sehingga tidak ada cara untuk mengisinya meskipun sudah mencoba berulang kali. Berikut adalah rincian letak *error task* tersebut.



Gambar 3.3 *Task Error Field*

Setelah ditelusuri, penyebabnya ditemukan pada definisi *field* journal_id di model HRExpenseSheet yang memiliki atribut required=True. Atribut ini menyebabkan ORM Odoo memberlakukan validasi wajib isi secara global pada saat penyimpanan, tanpa mempertimbangkan apakah *field* tersebut ditampilkan di antarmuka atau tidak. Karena *field* journal_id memang tidak dimunculkan pada tahap pengisian form, validasi tersebut selalu gagal dan menghalangi pengguna menyelesaikan pengajuan. Perbaikan dilakukan dengan mengubah atribut required=True menjadi required=False pada definisi *field* tersebut.

Setelah perubahan diterapkan dan modul diperbarui, pengguna dapat menyelesaikan pengisian form Jurnal Pengeluaran dan mengirimkan dokumen tanpa hambatan. Notifikasi *error* yang sebelumnya muncul tidak lagi ditampilkan oleh sistem. Perubahan ini tidak menghilangkan fungsionalitas *field* journal_id, *field* tersebut tetap berperan pada state lain yang memang memerlukannya melainkan hanya melepas restriksi yang tidak relevan pada tahap pengisian awal. Berikut rincian kode yang penulis tambahkan di *task* ini.

```
journal_id = fields.Many2one(
    'account.journal',
    string='Expense Journal',
    states={
        'done': [('readonly', True)],
        'post': [('readonly', True)]
    },
    check_company=True,
    domain="[('type', '=', 'purchase'), ('company_id', '=',
company_id), ('is_show_in_ecv', '=', True)]",
    default=_default_journal_id,
```

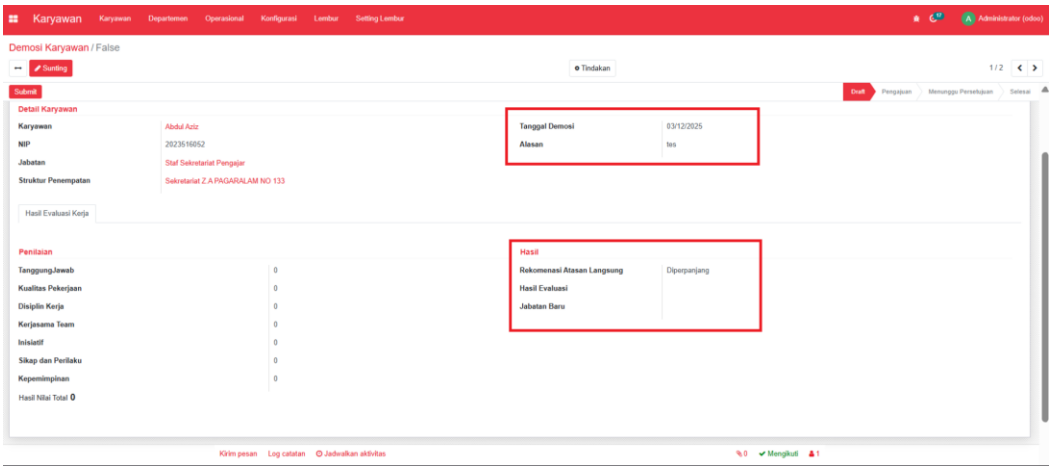
```
    required=False,
    help="The journal used when the expense is done."
)
```

Gambar 3.4 Perbaikan Kode Entri Jurnal

Pengerjaan *task* ini dilakukan melalui dua iterasi pengembangan. Pada iterasi pertama, penulis masih beradaptasi dengan alur penggunaan GitLab sehingga terjadi kesalahan saat melakukan push, yaitu perubahan kode dikirim ke *branch* yang tidak sesuai. Setelah memperoleh *feedback* dari mentor mengenai penggunaan *branch* yang benar, penulis mempelajari kembali alur kerja GitLab yang diterapkan pada proyek, kemudian mengulangi proses implementasi dengan menyesuaikan *branch* tujuan. Setelah dilakukan pengujian ulang, perubahan berhasil di-merge ke repositori utama.

3.2.5 Validasi *Mandatory* pada Modul Karyawan dan Master Data Diskon

Issue #518 dan #519 mengidentifikasi permasalahan pada formulir Promosi dan Demosi di modul Karyawan. Kedua formulir digunakan untuk mencatat proses mutasi jabatan karyawan secara formal, namun sejumlah *field* yang secara operasional wajib diisi seperti tanggal efektif, alasan mutasi, hasil evaluasi, dan jabatan tujuan tidak memiliki validasi *mandatory*. Dokumen mutasi akibatnya dapat dikirimkan dalam kondisi tidak lengkap dan tersimpan ke basis data tanpa data yang seharusnya wajib ada, berpotensi mengganggu integritas data kepegawaian yang menjadi acuan proses administrasi selanjutnya. Berikut adalah gambar terkait *error* dari rincian *task* di Gitlab.



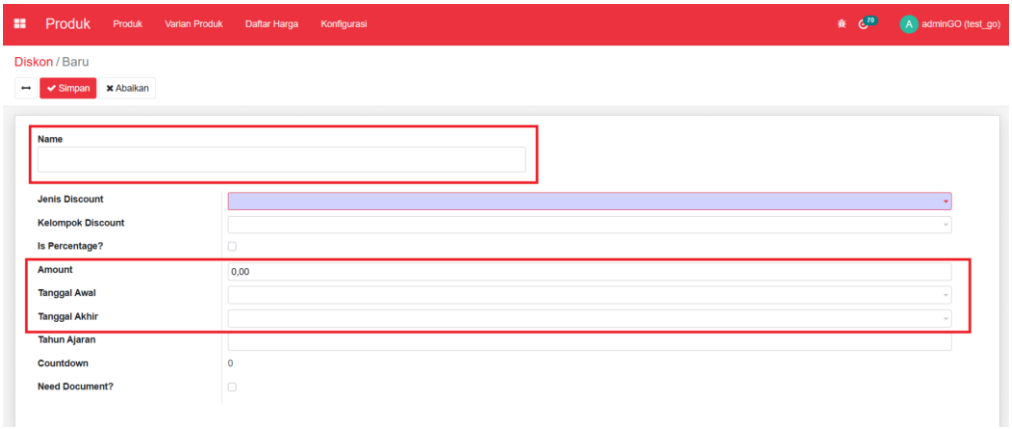
Gambar 3.5 *Mandatory Field* Karyawan

Karena kedua formulir merupakan view milik modul go_hr sendiri, perbaikan dilakukan langsung pada berkas XML view masing-masing dengan menambahkan atribut required="1" pada *field* yang bersangkutan. Mekanisme ini memanfaatkan validasi bawaan Odoo di lapisan antarmuka: sistem secara otomatis menampilkan indikator wajib isi dan memblokir pengiriman formulir apabila *field* yang ditandai masih kosong. Seluruh perbaikan selesai tanpa menyentuh kode Python maupun skema basis data. Berikut merupakan tambahan kode pada *field* karyawan.

```
<group>
    <field name="demotion_date" required="1"/>
    <field name="reason" required="1"/>
</group>
...
<group string="Hasil">
    <field   name="recommendation"   string="Rekomendasi   Atasan
Langsung" required="1"/>
    <field name="evaluation" required="1"/>
    <field name="new_job_id" required="1"/>
</group>
```

Gambar 3.6 Perbaikan Kode *Field* Karyawan

Issue #527 dan #528 menangani permasalahan serupa pada master data Diskon dan Jenis Diskon di modul Inventori, di mana *field-field* krusial seperti nama, nominal, dan periode berlaku tidak bersifat *mandatory*. Perbedaan mendasar dari dua issue sebelumnya terletak pada kepemilikan view: kedua formulir ini didefinisikan di luar modul yang dikerjakan, sehingga modifikasi langsung pada berkas XML induk tidak dapat dilakukan tanpa melanggar prinsip isolasi modul. Pendekatan yang digunakan adalah membuat berkas XML baru yang mewarisi view induk melalui mekanisme view *inheritance* dan menggunakan ekspresi XPath untuk menyuntikkan atribut required secara presisi pada *field* yang dituju. Hasilnya identik dari sisi pengguna, namun seluruh perubahan sepenuhnya terisolasi di berkas baru tanpa menyentuh definisi view asli mencerminkan praktik kustomisasi modul Odoo yang baik di mana sistem inti tetap utuh dan tidak terganggu. Berikut gambar dari rincian *task* di Gitlab terkait *error* yang terjadi.



Gambar 3.7 *Mandatory Field* Diskon

```
<record id="go_diskon_views_form_inherit" model="ir.ui.view">
    <field
name="name">go.diskon.views.form.inherit.mandatory</field>
    <field name="model">go.discount</field>
    <field name="inherit_id" ref="go_diskon_views_form"/>
    <field name="arch" type="xml">

        <xpath expr="//field[@name='name']" position="attributes">
            <attribute name="required">1</attribute>
        </xpath>

        <xpath           expr="//group/group/field[@name='amount']"
position="attributes">
            <attribute name="required">1</attribute>
        </xpath>

        <xpath                    expr="//field[@name='start_date']"
position="attributes">
            <attribute name="required">1</attribute>
        </xpath>

        <xpath                      expr="//field[@name='end_date']"
position="attributes">
            <attribute name="required">1</attribute>
        </xpath>

    </field>
</record>
```

Gambar 3.8 Perbaikan kode *Mandatory* Diskon

Rincian kode diatas adalah tambahan kode yang penulis tambahkan. Setelah seluruh perubahan pada keempat formulir diterapkan dan modul diperbarui, *field-field* yang sebelumnya dapat dikosongkan kini menampilkan indikator wajib isi. Sistem mencegah penyimpanan data yang tidak lengkap pada formulir mutasi karyawan maupun entri master

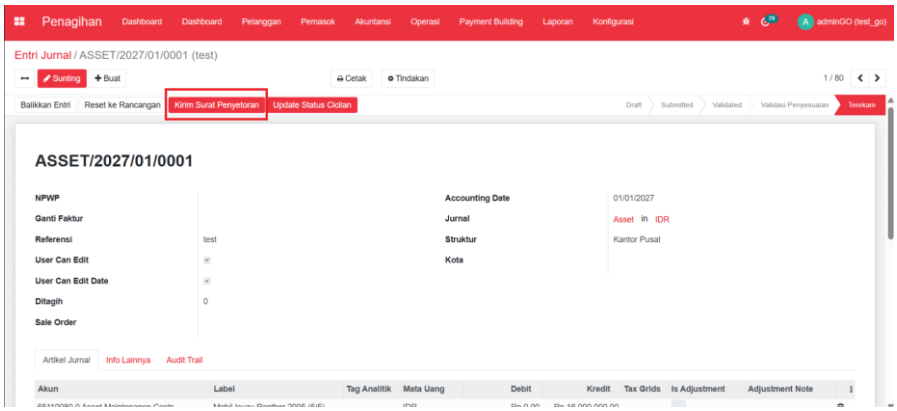
data diskon, memastikan setiap data yang masuk ke sistem sudah memenuhi kelengkapan yang dipersyaratkan.

Task ini dapat diselesaikan dalam satu iterasi pengembangan. Setelah implementasi selesai, penulis melakukan pengujian mandiri menggunakan lingkungan Docker sebelum diteruskan kepada tim *testing*. Berdasarkan *feedback* yang diberikan, validasi yang diimplementasikan telah berjalan sesuai kebutuhan sehingga tidak diperlukan perbaikan lanjutan dan perubahan berhasil di-merge ke repositori utama

3.2.6 Penyembunyian Tombol pada Modul Entri Jurnal

Issue #538 dan Issue #539 berkaitan dengan masih ditampilkannya beberapa aksi yang sudah tidak digunakan pada halaman Entri Jurnal (account.move). Aksi tersebut terdiri atas tombol Kirim Surat Penyetoran yang berada pada bagian header formulir serta opsi Surat Instruksi Setor yang muncul pada menu Print. Keberadaan kedua elemen tersebut berpotensi menimbulkan kebingungan bagi pengguna karena proses bisnis yang terkait sudah tidak lagi digunakan dalam operasional sistem.

Penanganan Issue #538 dilakukan menggunakan mekanisme view *inheritance* Odoo. Penulis membuat view turunan yang mewarisi go_account.go_account_move_views, kemudian menggunakan XPath untuk menambahkan atribut invisible="1" pada tombol action_send_laporan_surat_penyetoran. Dengan pendekatan tersebut, tombol Kirim Surat Penyetoran tidak lagi ditampilkan pada antarmuka tanpa perlu melakukan perubahan pada view asli. Gambar berikut mengenai letak tombol yang akan dihilangkan.



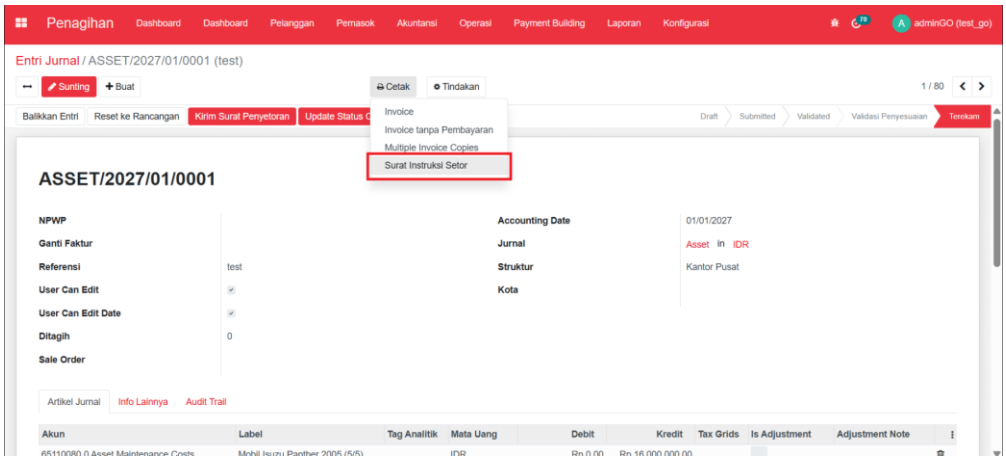
Gambar 3.9 Hide Button "kirim Surat Penyetoran"

Berikut merupakan potongan kode yang penulis tambahkan dalam menghilangkan button tersebut.

```
<xpath
expr="//button[@name='action_send_laporan_surat_penyetoran']"
       position="attributes">
    <attribute name="invisible">1</attribute>
</xpath>
```

Gambar 3.10 Perbaikan Kode Hide Button

Issue #539 memiliki karakteristik yang berbeda karena elemen yang ingin dihilangkan bukan berupa tombol pada view, melainkan opsi laporan yang muncul pada menu Print. Berdasarkan hasil penelusuran, opsi Surat Instruksi Setor berasal dari objek ir.actions.report yang masih terikat pada model account.move melalui atribut binding_model_id. Untuk menghilangkan opsi tersebut tanpa menghapus definisi laporan, nilai binding_model_id diubah dan menjadi false. Gambar tersebut merupakan tambahan button yang harus dihilangkan.



Gambar 3.11 Hide Button surat instruksi setor

```
<field name="binding_model_id" eval="False"/>
```

Gambar 3.12 Perbaikan Kode Hide Button

Perubahan dan tambahan kode tersebut menyebabkan laporan tidak lagi terdaftar pada menu Print, sehingga opsi Surat Instruksi Setor tidak dapat diakses melalui halaman Entri Jurnal.

Pada tahap pengujian lanjutan, ditemukan bahwa akses menuju fitur Surat Instruksi Setor masih tersedia melalui tombol action_surat_instruksi_penyetoran pada form modul Penyetoran. Oleh karena itu, dilakukan penyesuaian tambahan menggunakan view *inheritance*

dengan XPath position="replace" untuk menghapus tombol tersebut dari antarmuka. Langkah ini memastikan bahwa seluruh titik akses menuju fitur yang sudah tidak digunakan berhasil dihilangkan secara konsisten.

```
<xpath expr="//button[@name='action_surat_instruksi_penyetoran']"
       position="replace"/>
```

Gambar 3.13 Perbaikan Kode Hide Button

Melalui implementasi tersebut, kedua issue berhasil diselesaikan tanpa memerlukan perubahan pada model Python maupun struktur basis data. Seluruh penyesuaian dilakukan pada lapisan XML dan konfigurasi report Odoo sehingga risiko terhadap fungsi lain dalam sistem dapat diminimalkan.

Implementasi *task* ini memerlukan dua iterasi pengembangan. Pada iterasi pertama, tombol Kirim Surat Penyetoran telah berhasil disembunyikan, namun berdasarkan hasil pengujian dan *feedback* dari mentor masih ditemukan bahwa tombol Surat Instruksi Setor pada menu *dropdown* masih ditampilkan. Sebagai upaya penyelesaian permasalahan, penulis melakukan analisis kembali terhadap elemen tampilan yang terkait, kemudian menyesuaikan implementasi hingga kedua tombol berhasil disembunyikan sesuai kebutuhan sistem.

3.2.7 Perbaikan Logic Nominal dan *Filter* Pembayaran Modul Penyetoran

Modul Penyetoran pada sistem Ganesha Operations digunakan untuk mencatat proses penyerahan dana tunai dari unit kerja cabang ke kas pusat organisasi. Permasalahan yang diidentifikasi pada Issue #543 dan #553 berkaitan dengan ketidakakuratan nilai yang ditampilkan pada kolom Nominal Belum Disetor. Sistem menghitung nilai tersebut berdasarkan total tagihan pada faktur secara keseluruhan, tanpa mempertimbangkan berapa yang sudah benar-benar dibayarkan oleh siswa. Akibatnya, pada kasus pembayaran cicilan, nilai yang ditampilkan lebih besar dari dana yang sebenarnya diterima cabang. Selain itu, sistem tidak membedakan metode pembayaran dana yang masuk melalui transfer bank ikut terhitung bersama pembayaran tunai, padahal menu Penyetoran ini secara khusus hanya untuk mencatat pergerakan uang tunai fisik yang diserahkan ke pusat.

Salah satu faktor yang membuat *task* ini cukup kompleks adalah sifat modul Penyetoran yang tidak berdiri sendiri. Modul ini terintegrasi dengan beberapa modul lain secara bersamaan mulai dari modul akuntansi untuk data faktur dan pembayaran, modul jurnal untuk membedakan tipe transaksi tunai dan non-tunai, hingga modul struktur organisasi untuk

konteks multi-cabang. Perubahan pada satu titik berpotensi mempengaruhi bagian lain yang tidak langsung terlihat, sehingga proses penelusuran dan pengujian membutuhkan pemahaman terhadap alur data lintas modul yang cukup dalam.

Setelah menelusuri struktur kode yang ada, ditemukan bahwa model go.penyetoran.line menghitung nilai nominal_belum_setor dengan membaca langsung total nilai faktur, bukan dari riwayat pembayaran yang sudah terjadi. Untuk memperbaiki ini, penulis melakukan perubahan pada fungsi _get_nominal() di model tersebut. Logika kalkulasinya diubah agar membaca riwayat pembayaran aktual yang sudah terekonsiliasi dengan faktur, kemudian menyaring hanya pembayaran yang melalui jurnal bertipe kas (tunai). Dari total pembayaran tunai tersebut, dikurangi dengan nominal yang sudah pernah disetor sebelumnya pada dokumen penyetoran lain yang sudah berstatus selesai. Berikut perubahan kode yang penulis lakukan di *task* ini.

```
class GoPenyetoranLine(models.Model):
    _inherit = "go.penyetoran.line"

    @api.depends('invoice_id',          'invoice_id.payment_state',
'invoice_id.line_ids.matched_credit_ids')
    def _get_nominal(self):
        for line in self:
            if line.invoice_id:
                invoice = line.invoice_id.sudo()
                payments = invoice._get_reconciled_payments()

                total_cash_paid = sum(
                    p.amount for p in payments if p.journal_id.type
== 'cash'
                )

                domain = [
                    ('invoice_id', '=', invoice.id),
                    ('go_penyetoran_id.state', '=', 'done')
                ]
                if isinstance(line.id, int):
                    domain.append(('id', '!=', line.id))

                penyetoran_sebelumnya                                =
self.env['go.penyetoran.line'].sudo().search(domain)
                total_sudah_disetor                                  =
sum(penyetoran_sebelumnya.mapped('nominal_akan_disetor'))

                line.nominal_belum_setor   =   total_cash_paid   -
total_sudah_disetor
            else:
                line.nominal_belum_setor = 0
```

```
class GoPenyetoran(models.Model):
    _inherit = "go.penyetoran"

    is_cash_payment = fields.Boolean(
        string="Is Cash Payment",
        compute="_compute_is_cash_payment",
        store=False
    )

    @api.depends('line_ids.invoice_id')
    def _compute_is_cash_payment(self):
        for rec in self:
            invoices = rec.line_ids.mapped('invoice_id').sudo()
            if not invoices:
                rec.is_cash_payment = False
                continue

            all_invoices_are_cash = True
            for inv in invoices:
                payments = inv._get_reconciled_payments()
                if not payments or any(p.journal_id.type != 'cash'
for p in payments):
                    all_invoices_are_cash = False
                    break
            rec.is_cash_payment = all_invoices_are_cash
```

Gambar 3.14 Perbaikan Kode Nominal Belum Setor

Selain perubahan logika kalkulasi, penulis juga menambahkan *field* payment_method_type pada model account.move melalui mekanisme *inheritance*, serta melakukan override pada fungsi action_create_payments() agar setiap pembayaran yang dikonfirmasi secara otomatis ditandai sebagai cash atau non_cash berdasarkan tipe jurnal yang digunakan. Penandaan ini menjadi dasar bagi computed *field* is_cash_payment pada model go.penyetoran untuk menentukan apakah seluruh transaksi dalam satu dokumen penyetoran bersifat tunai atau tidak. Berikut tambahan kode di *file* lain.

```
class AccountMove(models.Model):
    _inherit = "account.move"

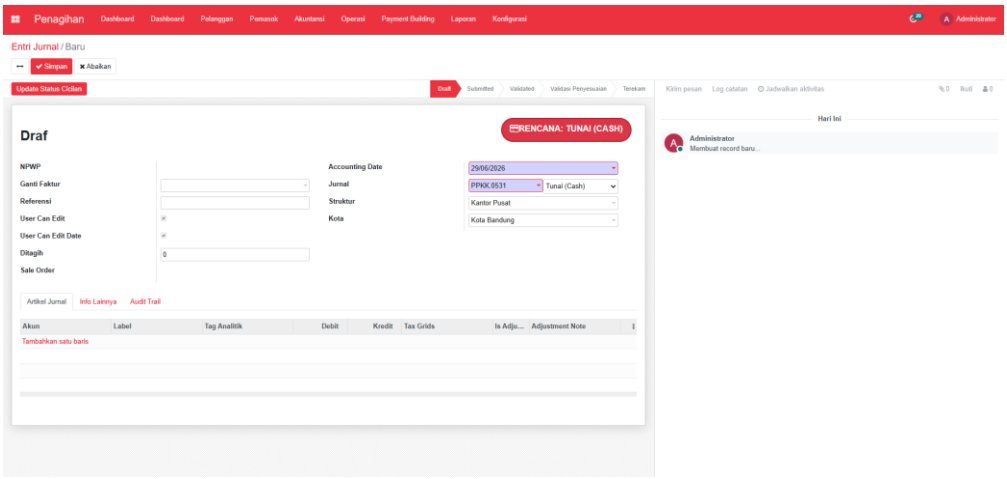
    metode_pembayaran_manual = fields.Selection([
        ('cash', 'Tunai (Cash)'),
        ('bank', 'Transfer (Bank)')
    ], string="Metode Pembayaran", tracking=True)

    cara_bayar_display = fields.Char(
        string="Info Pembayaran",
        compute="_compute_cara_bayar_display"
    )
```

```
    @api.depends('metode_pembayaran_manual', 'payment_state',
'line_ids.matched_credit_ids')
    def _compute_cara_bayar_display(self):
        for rec in self:
            payments = rec._get_reconciled_payments()
            if payments:
                journal = payments[0].journal_id
                rec.cara_bayar_display = "LUNAS: TUNAI" if
journal.type == 'cash' else "LUNAS: TRANSFER"
            elif rec.metode_pembayaran_manual:
                dict_metode =
dict(rec._fields['metode_pembayaran_manual'].selection or [])
                label =
dict_metode.get(rec.metode_pembayaran_manual)
                rec.cara_bayar_display = f"RENCANA:
{label.upper()}"
            else:
                rec.cara_bayar_display = "METODE BELUM DIPILIH"

    def action_register_payment(self):
        res = super(AccountMove, self).action_register_payment()
        if self.metode_pembayaran_manual:
            journal = self.env['account.journal'].search([
                ('type', '=', self.metode_pembayaran_manual),
                ('company_id', '=', self.company_id.id)
            ], limit=1)
            if journal:
                res['context']['default_journal_id'] = journal.id
        return res
```

Gambar 3.15 Perbaikan Kode *Filter* Tunai dan Non tunai



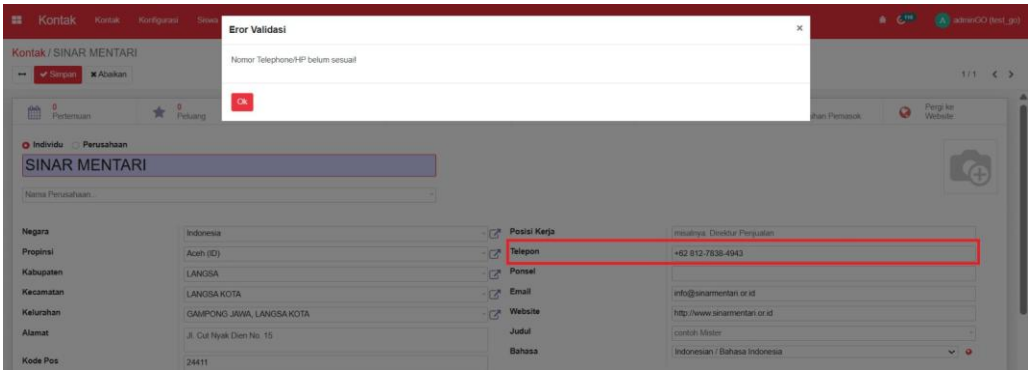
Gambar 3.16 Penambahaan *Field* Transfer

Hasil implementasi pada gambar diatas yang dilakukan penulis mencakup seluruh perubahan logika di atas dan telah di-merge ke repositori utama. Namun karena kompleksitas integrasi antar modul yang cukup tinggi, pada tahap pengujian fitur belum menampilkan hasil yang diharapkan pada antarmuka. Kondisi ini merupakan hal yang wajar pada pengembangan sistem enterprise berskala besar, di mana penyesuaian pada satu modul yang terhubung dengan banyak komponen lain seringkali membutuhkan lebih dari satu siklus iterasi untuk dapat berjalan dengan sempurna. Oleh karena itu, pengembangan dilanjutkan oleh anggota tim lain untuk penyempurnaan lebih lanjut. Kontribusi penulis pada *task* ini berfokus pada perancangan dan implementasi awal logika kalkulasi dan penandaan metode pembayaran sebagai fondasi dari fitur yang dikembangkan.

Task ini merupakan salah satu implementasi dengan tingkat kompleksitas yang tinggi sehingga memerlukan tiga iterasi pengembangan. Pada setiap iterasi, penulis memperoleh *feedback* dari mentor berdasarkan hasil pengujian terhadap logika yang telah diimplementasikan. Sebagai upaya penyelesaian permasalahan, penulis melakukan analisis ulang terhadap alur bisnis, menambahkan beberapa mekanisme *inheritance*, serta melakukan penyesuaian kode pada beberapa berkas dan modul yang saling terintegrasi. Meskipun perubahan telah berhasil di-merge ke repositori utama, hasil verifikasi menunjukkan bahwa implementasi masih memerlukan pengembangan lanjutan oleh tim pengembang.

3.2.8 Perbaikan Validasi Nomor Telepon Modul Kontak

Permasalahan yang dilaporkan pada modul Kontak berkaitan dengan validasi nomor telepon yang tidak berjalan sesuai kebutuhan. Sebelum perbaikan, sistem menggunakan satu fungsi validasi bernama _check_numeric_phone yang hanya memeriksa apakah nilai yang dimasukkan berupa angka murni menggunakan val.isdigit(). Pendekatan ini menimbulkan dua masalah sekaligus: di satu sisi menolak nomor dengan format internasional seperti +6281234567890 karena karakter + dianggap bukan angka, padahal format ini umum digunakan — di sisi lain juga terlalu longgar karena menerima kombinasi angka acak seperti 123 yang jelas bukan nomor telepon yang valid. Berikut error yang terjadi saat memasukan nomor dengan format +62.



Gambar 3.17 *Error* Nomor telepon tidak dapat diinput

Perbaikan dilakukan dengan memisahkan satu fungsi validasi tersebut menjadi dua fungsi terpisah, masing-masing menangani *field* yang berbeda. Fungsi _check_mobile_format khusus untuk *field* mobile menggunakan pola regex ^(\+62|0)8[0-9]{8,11}$ yang hanya menerima nomor HP Indonesia dengan awalan 0 atau +62. Fungsi _check_phone_format untuk *field* phone menggunakan regex yang lebih longgar ^[0-9+]{6,15}$ karena nomor telepon kantor atau rumah memang lebih bervariasi formatnya. Sebelum validasi dijalankan, karakter spasi dan tanda hubung dibersihkan terlebih dahulu sehingga pengguna tetap bisa mengetik nomor dengan format apapun tanpa langsung ditolak sistem. Berikut rincian kode yang penulis tambahkan.

```
# go_base/models/res_partner.py
# Implementation rules check: Logic consistency secured, clean
structure maintained.

from odoo import fields, models, api, _
from odoo.exceptions import ValidationError


class ResPartner(models.Model):
    _inherit = 'res.partner'

    @api.constrains('mobile')
    def _check_mobile_format(self):
        """
        Validasi format nomor HP standar Indonesia (prefix 0 atau
+62, diikuti 8, panjang 8-11 digit)
        """
        pattern = re.compile(r'^(\+62|0)8[0-9]{8,11}$')
        for rec in self:
            if rec.mobile:
                val = rec.mobile.replace(' ', '').replace('-', '')
                if not pattern.match(val):
```

```
                    raise ValidationError(_('Nomor HP harus format
08xxxxxxxx atau +628xxxxxxxx'))

    @api.constrains('phone')
    def _check_phone_format(self):
        """
        Validasi format nomor telepon rumah/kantor (mengizinkan
angka dan '+', panjang 6-15 karakter)
        """
        for rec in self:
            if rec.phone:
                val = rec.phone.replace(' ', '').replace('-', '')
                if not re.match(r'^[0-9+]{6,15}$', val):
                    raise ValidationError(_('Nomor Telepon tidak
valid'))
```

Gambar 3.18 Perbaikan Kode Nomor Telepon

Setelah perubahan diterapkan, pengguna bisa menyimpan nomor HP dengan format 08 maupun +62 tanpa mendapat *error*. Nomor yang tidak sesuai format tetap ditolak dengan pesan yang jelas, sehingga data kontak yang tersimpan di sistem lebih konsisten dan terjaga kualitasnya.

Task ini diselesaikan melalui dua iterasi pengembangan. Pada implementasi awal, validasi hanya diterapkan pada nomor telepon seluler. Setelah dilakukan evaluasi dan memperoleh *feedback* dari mentor, penulis mempertimbangkan bahwa sistem juga memungkinkan penggunaan nomor telepon rumah maupun nomor fax. Oleh karena itu, pada iterasi kedua penulis memperluas logika validasi agar dapat mengakomodasi seluruh jenis nomor telepon yang digunakan pada sistem, kemudian melakukan pengujian ulang hingga implementasi berjalan sesuai kebutuhan.

3.2.9 *Filter* Produk *Expired* Modul BIMKER

Modul BIMKER menyediakan fitur penambahan produk bimbingan belajar yang dapat diakses setelah form BIMKER di-*submit* dan pengguna menekan tombol "Input Product". Pada kondisi sebelum perbaikan, daftar produk yang muncul di tab tersebut menampilkan seluruh produk tanpa mem*filter* berdasarkan masa berlakunya. Akibatnya, produk yang sudah melewati Tanggal Akhir yang dinyatakan sebagai produk *expired* tetap muncul sebagai pilihan dan dapat dipilih oleh pengguna. Kondisi ini berpotensi menyebabkan kesalahan input karena pengguna tidak memiliki cara untuk membedakan produk yang masih aktif dari yang sudah tidak berlaku. Perbaikan dilakukan pada dua titik sekaligus karena definisi domain produk berada di dua tempat yang berbeda. Pada berkas go_product_bimker.py, domain *field* product_id diperbarui

dengan menambahkan kondisi *filter* tanggal akhir produk hanya ditampilkan apabila *field* end_date kosong atau bernilai lebih besar dari atau sama dengan tanggal hari ini. Bersamaan dengan itu, fungsi onchange_product_id juga diperbaiki dengan menambahkan pengecekan if self.product_id sebelum mengisi *field* name untuk menghindari potensi *error* ketika *field* produk dikosongkan. Pada berkas go_bimker_view.xml, *filter* yang sama ditambahkan pada domain *field* pemilihan produk di dalam form BIMKER agar hasilnya konsisten antara lapisan model dan lapisan tampilan. Berikut kode yang penulis tambahkan terkait *filter expired.*

```
product_id = fields.Many2one(
    'product.product',
    string="Produk",
    domain=[
        ('is_produk_go', '=', True),
        ('is_bimker', '=', 'bimker'),
        '|',
        ('product_tmpl_id.end_date', '=', False),
        ('product_tmpl_id.end_date', '>=', 'today'),
    ]
)

@api.onchange('product_id')
def onchange_product_id(self):
    if self.product_id:
        self.name = self.product_id.name
```

Gambar 3.19 Perbaikan Kode *Expired*

Bersamaan dengan perbaikan *filter*, penulis juga membuat berkas go_bimker_view_inherit.xml yang menambahkan validasi pada enam *field* utama form BIMKER. Melalui mekanisme view *inheritance* dan ekspresi XPath, atribut required dikonfigurasi aktif ketika state berada pada draft sehingga keenam *field* tersebut wajib diisi sebelum dokumen dapat dikirimkan. Atribut readonly dikonfigurasi aktif ketika state sudah meninggalkan draft sehingga data yang sudah terverifikasi terlindungi dari perubahan yang tidak disengaja. Berikut tambahan kode terkait *mandatory*.

```
<xpath expr="//field[@name='go_kota_id']" position="attributes">
    <attribute name="attrs">
        {'readonly': [('state', 'not in', ['draft'])],
         'required': [('state', '=', 'draft')]}
    </attribute>
</xpath>
```

Gambar 3.20 Perbaikan Kode *mandatory*

Implementasi yang dilakukan telah di-merge ke repositori utama, namun pengujian menyeluruh belum dapat diselesaikan pada masa magang penulis karena fitur ini memerlukan penyesuaian dengan komponen lain di modul berbeda yang terhubung dengan modul BIMKER. Ditambah dengan perpindahan penulis ke proyek lain pada periode yang bersamaan, pengujian akhir dan penyempurnaan dilanjutkan oleh anggota tim yang tetap menangani modul ini.

Implementasi *task* ini dilakukan melalui dua iterasi pengembangan. Pada iterasi pertama, logika penyaringan produk *expired* telah berhasil diimplementasikan, namun hasil yang diharapkan belum ditampilkan pada modul BIMKER. Berdasarkan *feedback* dari mentor dan hasil pengujian, penulis melakukan analisis ulang terhadap implementasi serta penyesuaian kode pada iterasi berikutnya. Meskipun perubahan telah berhasil di-merge ke repositori utama, hasil verifikasi menunjukkan bahwa implementasi masih memerlukan pengembangan lebih lanjut oleh tim pengembang.

3.3 Proyek Training: bantuUMKM.com

Selain proyek utama, penulis juga terlibat dalam proyek pelatihan bernama bantuUMKM.com yang dikerjakan pada fase awal masa magang. Proyek ini merupakan bagian dari proses orientasi yang diberikan perusahaan sebelum penulis ditempatkan pada proyek aktif, dan dikerjakan bersama satu rekan magang di bawah bimbingan Bapak Toto selaku *Project* Manager. Pembahasan pada subbab berikut mencakup gambaran sistem, pengembangan halaman *Login*, serta pengembangan *Dashboard* dan fitur CRUD.

3.3.1 Gambaran Sistem dan Tech Stack

Proyek bantuUMKM.com merupakan proyek pelatihan yang dikerjakan pada fase awal masa magang di bawah bimbingan *Project* Manager. Sistem ini dirancang sebagai platform *dashboard* manajemen usaha berbasis web yang mencakup pencatatan transaksi penjualan, pengelolaan laporan, dan manajemen data karyawan. Proyek dikerjakan bersama satu rekan magang rekan mengerjakan sisi backend, sementara penulis bertanggung jawab penuh atas pengembangan sisi frontend.

Technology stack yang digunakan pada sisi frontend mencakup React 18 sebagai *framework* utama, React Router DOM v7 untuk navigasi antar halaman, SweetAlert2 untuk

dialog interaktif, react-data-table-component untuk tampilan tabel data, dan SheetJS untuk fitur impor dan ekspor Excel. Komunikasi dengan backend dilakukan melalui REST API menggunakan Fetch API, dengan autentikasi berbasis JWT yang dikirimkan pada setiap request. Sisi backend dibangun rekan menggunakan Node.js, Express.js, dan MySQL seluruhnya di luar cakupan pekerjaan penulis.

3.3.2 Pengembangan *Login*

Halaman *Login* dibangun menggunakan React dengan tampilan dua kolom kolom kiri berisi ilustrasi dan teks sambutan, kolom kanan berisi formulir *login*. Pengelolaan nilai input menggunakan useState yang memperbarui nilai secara reaktif setiap kali pengguna mengetik. Sebelum formulir dikirim, sistem memeriksa apakah kedua *field* sudah terisi jika belum, notifikasi peringatan ditampilkan menggunakan react-toastify tanpa perlu mengirim request ke backend. Setelah *login* berhasil, pengguna diarahkan ke halaman *Dashboard* menggunakan useNavigate dari React Router DOM.

3.3.3 Pengembangan *Dashboard* dan Fitur CRUD modul Sales, Reports, Employees

Halaman *Dashboard* dirancang dengan tata letak dua bagian: sidebar navigasi di sisi kiri dan area konten utama di sisi kanan. Area konten menampilkan empat kartu ringkasan bisnis total penjualan, pendapatan bulanan, jumlah pelanggan aktif, dan pesanan tertunda masing-masing dengan warna latar berbeda untuk memudahkan pembacaan. Komponen sidebar digunakan secara konsisten di seluruh halaman agar pengguna bisa berpindah halaman kapan saja.

Tiga modul CRUD yang dikembangkan adalah Sales, Reports, dan Employees. Ketiganya memiliki pola antarmuka yang seragam: bilah aksi di bagian atas yang memuat tombol tambah data, kolom pencarian, dan tombol impor/ekspor Excel, diikuti tabel data menggunakan react-data-table-component dengan paginasi dan pengurutan kolom.

Formulir tambah dan ubah data menggunakan dialog modal SweetAlert2 dengan validasi input yang dilakukan sebelum data dikirim ke backend, sehingga *field* yang kosong langsung ditolak di sisi frontend. Modul Employees memiliki formulir yang lebih kompleks dibanding dua modul lainnya karena terdiri dari delapan *field* termasuk foto karyawan dengan fitur pratinjau gambar saat formulir edit dibuka.

Fitur pencarian pada ketiga modul bekerja langsung pada data yang sudah dimuat tanpa perlu request ulang ke backend, sehingga hasil muncul secara instan. Fitur ekspor Excel mengubah data yang tampil menjadi *file* Excel menggunakan SheetJS dan langsung diunduh melalui browser. Fitur impor membaca *file* Excel yang diunggah, mengubahnya menjadi array data, lalu mengirimkannya ke backend untuk disimpan.

3.4 Proyek i-BOSS Nasmoco (Laravel)

Proyek ketiga yang dikerjakan selama masa magang adalah pengembangan fitur pada sistem i-BOSS Nasmoco, sebuah aplikasi back office berbasis Laravel yang dikelola PT Gamatechno Indonesia untuk klien Nasmoco. Keterlibatan penulis pada proyek ini lebih terbatas dibandingkan proyek Odoo, namun tetap memberikan pengalaman teknis yang relevan terutama dalam pengembangan berbasis *framework* PHP dan pengelolaan kode melalui GitLab. Pembahasan pada subbab berikut mencakup gambaran sistem dan tiga *task* yang dikerjakan selama masa magang.

3.4.1 Gambaran Sistem dan Tech Stack

Sistem i-BOSS (Back Office Support System) merupakan aplikasi web internal yang dikembangkan PT Gamatechno Indonesia untuk klien Nasmoco, sebuah jaringan dealer otomotif berskala nasional. Sistem ini digunakan untuk mendukung operasional back office Nasmoco secara digital, mencakup pencatatan permintaan pengadaan (User Request), proses persetujuan bertahap, pengelolaan data vendor, hingga pembuatan Purchase Order. Nasmoco berperan sebagai pemilik sistem, sementara PT Gamatechno Indonesia berperan sebagai vendor pengembang.

Sistem dibangun menggunakan *framework* Laravel dengan PHP sebagai bahasa pemrograman utama dan Blade untuk membangun tampilan antarmuka pengguna. PostgreSQL digunakan sebagai basis data. Untuk kebutuhan pengembangan lokal, penulis menggunakan Laragon sebagai server lokal dan Visual Studio Code sebagai editor kode. Pengelolaan kode mengikuti alur GitLab yang sama dengan proyek Odoo berbasis Merge Request dengan peninjauan tim sebelum diintegrasikan ke *branch* utama.

Penulis terlibat sebagai *developer* yang mengerjakan *task* berbasis issue GitLab, mencakup implementasi pada lapisan antarmuka Blade maupun logika aplikasi di controller

dan migration basis data. Penulis tidak terlibat dalam perancangan arsitektur sistem, desain basis data, maupun penentuan kebutuhan bisnis.

3.4.2 Pembaruan Logic Modal Datatable dan *Filter* pada Menu User Request

Issue #239 mengidentifikasi dua permasalahan pada halaman daftar User Request. Pertama, keempat kartu ringkasan status Draft, Submitted, Requested, dan Closed masing-masing didefinisikan secara terpisah dengan struktur HTML yang identik namun nilai hard-coded yang berbeda. Pendekatan ini tidak efisien karena setiap perubahan tampilan harus dilakukan berulang pada keempat kartu secara manual. Kedua, halaman tidak memiliki modal untuk melihat detail item permintaan maupun mekanisme *filter* data.

Untuk permasalahan pertama, keempat kartu diubah menggunakan struktur @php $stats yang mendefinisikan data semua kartu dalam satu array, kemudian dirender dengan satu loop @foreach. Perubahan ini membuat modifikasi tampilan cukup dilakukan di satu tempat. Untuk permasalahan kedua, penulis menambahkan modal ShowRequest yang menampilkan tabel detail item menggunakan DataTables, terhubung ke endpoint baru get-item-table yang didaftarkan di routes/web.php. Selain itu ditambahkan komponen *filter dropdown* dengan empat parameter: Date Range, *Branch*, Job Type, dan Keyword, dilengkapi tombol Apply *Filter*s dan Reset All.

Penulis juga melakukan penyesuaian tampilan dengan mengganti warna aksen ungu bawaan tema menjadi hijau dan menyesuaikan komponen Select2 agar tampilannya konsisten dengan elemen form lain di halaman.font-family dan border-radius yang konsisten dengan elemen form lainnya pada halaman.

3.4.3 Penambahan Fitur Lampiran pada Halaman Detail User Request

Issue #257 menambahkan fitur unggah lampiran pada halaman detail User Request. Sebelumnya tidak ada cara untuk melampirkan dokumen pendukung seperti spesifikasi teknis atau surat persetujuan setelah permintaan dibuat, padahal dalam proses pengadaan nyata dokumen pendukung sering kali diperlukan untuk melengkapi permintaan yang sudah diajukan.

Implementasi dilakukan dengan menambahkan dua elemen baru pada halaman detail di berkas detail.blade.php. Pertama, tombol unggah yang hanya muncul bagi pengguna dengan hak akses update.procurement.user-request pengecekan hak akses ini dilakukan menggunakan

direktif @if(Auth::user()->can(...)) sehingga tombol tidak terlihat oleh pengguna yang tidak berwenang. Kedua, elemen <input type="*file*"> yang disembunyikan dari tampilan menggunakan class="d-none" dan baru aktif ketika tombol unggah diklik. Saat pengguna memilih *file*, fungsi JavaScript uploadAttachment() dipanggil secara otomatis dengan meneruskan ID permintaan sebagai parameter untuk proses pengiriman ke server.

Di sisi routing, endpoint baru POST /add-attachment/{id} didaftarkan di routes/web.php dan diarahkan ke method addAttachment pada UserRequestController, sehingga proses unggah lampiran ditangani oleh controller yang sama dengan operasi User Request lainnya. Implementasi pada halaman-halaman pengadaan lainnya yang termasuk dalam lingkup *task* yang sama dilanjutkan oleh anggota tim pengembang lain.

3.4.4 Perbaikan *Bug* Edit Vendor Group

Issue #234 mengidentifikasi *bug* pada fitur Edit Vendor sistem menampilkan *error* saat pengguna menyimpan data vendor tanpa mengisi *field* PIC (Person In Charge). Penyebabnya ada di dua lapisan sekaligus: kolom pic di tabel ref_vendor tidak dikonfigurasi sebagai nullable, dan di sisi controller nilai kosong dari form diteruskan langsung ke basis data tanpa penanganan khusus.

Perbaikan dilakukan melalui dua langkah. Pertama, dibuat migration untuk mengubah kolom pic menjadi nullable. Setelah dijalankan, ternyata tipe data aktual kolom di basis data adalah bigint bukan integer, sehingga migration pertama tidak berhasil. Langkah kedua membuat migration baru yang secara eksplisit menggunakan bigInteger dengan atribut nullable. Bersamaan dengan itu, baris kode di VendorController.php diubah agar nilai kosong dari form diteruskan sebagai null secara eksplisit, bukan sebagai empty string.

Setelah kedua perbaikan diterapkan, pengguna dapat menyimpan data vendor tanpa mengisi PIC tanpa mendapatkan *error*. Kasus dua migration untuk satu kolom ini terjadi karena tipe data aktual di basis data tidak selalu sama dengan yang tertulis di kode lama kondisi yang cukup umum ditemui saat bekerja pada sistem yang sudah berjalan cukup lama.

BAB IV
REFLEKSI PELAKSANAAN MAGANG

4.1 Relevansi Akademik

Pelaksanaan magang di PT Gamatechno Indonesia memberikan kesempatan untuk mempertemukan pemahaman yang dibangun melalui perkuliahan dengan realitas yang ditemui di lingkungan industri. Bagian ini menganalisis kesenjangan antara teori yang dipelajari di bangku kuliah dengan praktik yang ditemui di lapangan, sekaligus menarik pembelajaran akademik dari setiap kesenjangan yang ditemukan.

Dalam perkuliahan, sistem ERP diperkenalkan sebagai solusi yang diimplementasikan sekali lalu digunakan. Permasalahan teknis diasumsikan paling banyak muncul di fase awal, dan setelah sistem stabil pekerjaan pengembang pada dasarnya selesai. Kenyataan yang ditemui selama magang berbeda: sistem yang sudah berjalan bertahun-tahun tetap membutuhkan penyesuaian terus-menerus seiring berkembangnya kebutuhan bisnis. Bekerja pada sistem aktif ternyata lebih menantang dibandingkan membangun sistem baru karena setiap perubahan harus mempertimbangkan data yang sudah ada, pengguna yang sedang menggunakan sistem, dan modul-modul lain yang saling bergantung. Perbedaan ini menunjukkan bahwa kurikulum yang membahas ERP perlu memberikan porsi lebih pada fase pasca-implementasi yaitu pemeliharaan dan optimalisasi sistem yang sudah berjalan karena justru di sinilah sebagian besar pekerjaan pengembang di industri berlangsung.

Pola arsitektur MVC yang diajarkan di kelas memberikan kesan bahwa pengembangan fitur selalu menyentuh ketiga lapisan secara proporsional. Di lapangan, proporsi ini tidak berlaku. Dari seluruh *task* yang dikerjakan, sebagian besar hanya menyentuh satu atau dua lapisan saja. Yang lebih penting dari sekadar memahami ketiga lapisan adalah kemampuan mengidentifikasi lapisan mana yang harus disentuh untuk setiap permasalahan karena menyentuh lapisan yang salah pada sistem aktif bisa menimbulkan dampak yang melampaui batas *task* yang sedang dikerjakan. Perbedaan ini menghasilkan sudut pandang bahwa MVC dalam konteks sistem enterprise bukan sekadar pola struktural, melainkan alat analisis untuk membatasi risiko perubahan. Pemahaman ini sulit diperoleh dari proyek akademik yang tidak memiliki konsekuensi nyata ketika terjadi kesalahan.

Konsep *inheritance* yang diajarkan di perkuliahan berfokus pada membangun hierarki kelas dalam sistem yang dibuat sendiri dari awal. Di lingkungan Odoo, *inheritance* digunakan untuk mengintervensi sistem yang sudah ada. Perbedaannya mendasar: di kelas, kesalahan dalam membuat subkelas hanya berarti kode tidak berjalan dan bisa diperbaiki tanpa dampak lain. Di lapangan, perubahan melalui mekanisme inherit berdampak secara global pada seluruh bagian sistem yang menggunakan komponen tersebut. Pelajaran yang diperoleh adalah bahwa

konsep *inheritance* perlu diajarkan tidak hanya dalam konteks pembangunan sistem baru, tetapi juga dalam konteks modifikasi sistem yang sudah aktif karena karakteristik risikonya berbeda secara fundamental dan menuntut pendekatan yang berbeda pula.

Docker dalam perkuliahan diperkenalkan sebagai alat untuk menyebarkan aplikasi ke server secara konsisten. Manfaat yang paling terasa selama magang justru berbeda: Docker membuat seluruh anggota tim dapat bekerja pada kondisi lingkungan yang identik setiap harinya tanpa perlu konfigurasi ulang. Basis data pengujian yang disiapkan tim langsung bisa dijalankan di mesin lokal dengan konfigurasi yang sama persis seperti di mesin *developer* lain. Perbedaan ini menghasilkan pemahaman bahwa nilai containerization dalam fase pengembangan tidak kalah pentingnya dari nilai *deployment* bahkan dalam praktik sehari-hari justru lebih sering dirasakan. Aspek ini layak mendapat perhatian lebih dalam kurikulum yang membahas teknologi container.

Agile yang diajarkan di perkuliahan membentuk kesan bahwa menjalankan metodologi ini berarti mengikuti serangkaian ritual formal secara lengkap. Di lapangan, tim bekerja tanpa satu pun ritual formal tersebut namun tetap produktif dan responsif. Alur kerjanya sederhana: *task* diberikan melalui GitLab Issue, dikerjakan di *branch* terisolasi, diajukan melalui Merge Request, ditinjau tim, lalu digabungkan ke *branch* utama. Meski begitu, nilai-nilai inti Agile unit kerja yang kecil, umpan balik yang cepat, dan kolaborasi yang natural justru terasa lebih hidup dibandingkan saat simulasi Agile di kelas. Pembelajaran yang diperoleh adalah bahwa efektivitas Agile tidak bergantung pada kelengkapan ritualnya, melainkan pada seberapa konsisten nilai-nilai intinya diterapkan dalam keseharian tim. Pembelajaran ini sulit diperoleh dari simulasi akademik yang cenderung berfokus pada prosedur, bukan pada esensi kolaborasi yang sesungguhnya.

4.2 Pembelajaran Magang.

Pelaksanaan magang selama enam bulan di PT Gamatechno Indonesia memberikan banyak hal yang tidak diperoleh dari perkuliahan semata. Pembelajaran yang dimaksud bukan sekadar penguasaan teknologi baru, melainkan pergeseran cara berpikir dalam menghadapi permasalahan nyata di lingkungan pengembangan perangkat lunak profesional. Bagian ini memaparkan manfaat, kendala, dan tantangan yang diperoleh selama magang, khususnya yang berkaitan langsung dengan konsep-konsep yang telah dianalisis pada subbab sebelumnya.

Dari analisa yang telah dilakukan, pembelajaran paling mendasar yang diperoleh berkaitan dengan cara memahami sistem ERP yang sudah berjalan. Selama perkuliahan, cara

berpikir yang terbentuk adalah bahwa pengembangan perangkat lunak dimulai dari nol dan bergerak maju secara linear. Ketika pertama kali masuk ke proyek Ganesha Operations, pola pikir itu tidak bisa langsung dipakai. Sistem sudah punya ratusan data transaksi, pengguna aktif di berbagai cabang, dan modul-modul yang saling bergantung satu sama lain. Hal pertama yang harus dipelajari bukan cara menulis kode baru, melainkan cara membaca dan memahami kode yang sudah ada. Kemampuan ini menelusuri alur logika kode orang lain, memahami relasi antar model, dan menemukan titik yang tepat untuk melakukan perubahan berkembang perlahan selama magang dan terasa sebagai kompetensi yang paling berguna sekaligus paling jarang dilatih di bangku kuliah.

Dari sisi manfaat keilmuan, pemahaman terhadap arsitektur MVC berkembang melampaui definisi akademisnya. Di kelas, MVC dipahami sebagai cara mengorganisasi kode. Di lapangan, MVC justru menjadi cara berpikir sebelum menulis satu baris kode pun apakah permasalahan ini ada di lapisan logika bisnis atau di lapisan tampilan? Pertanyaan sederhana ini ternyata sangat menentukan karena salah memilih lapisan berarti perubahan bisa berdampak lebih luas dari yang diharapkan. Pemahaman tentang *inheritance* juga berkembang serupa. Konsep yang di kelas terasa teknis dan abstrak, di lapangan menjadi sangat konkret karena setiap perubahan melalui mekanisme *inheritance* berdampak global ke seluruh sistem. Ini membentuk kebiasaan untuk selalu berpikir dua kali sebelum mengubah sesuatu yang digunakan bersama oleh banyak bagian sistem.

Kendala terbesar yang ditemui selama magang adalah minimnya dokumentasi pada modul-modul kustom yang sudah ada. Berbeda dari proyek kuliah yang spesifikasinya jelas dan kodenya ditulis sendiri, modul pada sistem Ganesha Operations dibangun oleh banyak *developer* berbeda selama bertahun-tahun. Tidak ada dokumentasi yang menjelaskan mengapa suatu logika ditulis dengan cara tertentu. Cara mengatasinya adalah dengan menjadikan kode yang sudah ada sebagai dokumentasi itu sendiri membaca pola yang digunakan di modul lain, menggunakan mode *debug* Odoo untuk memeriksa nilai *field* secara langsung, dan menelusuri alur data dari antarmuka ke basis data sebelum mulai mengubah apapun. Proses ini mengajarkan bahwa kemampuan membaca dan memahami kode orang lain adalah kompetensi teknis yang sama pentingnya dengan kemampuan menulis kode baru.

Kendala lain yang juga cukup terasa adalah sifat sistem ERP yang sangat saling terhubung. Beberapa *task* yang terlihat sederhana dari deskripsinya ternyata menyentuh banyak bagian sistem sekaligus. Pengalaman mengerjakan *task* pada modul Penyetoran menjadi pelajaran konkret soal ini perubahan yang awalnya terlihat seperti perbaikan satu nilai ternyata

melibatkan penelusuran melalui beberapa model yang saling berelasi. Ini membentuk kebiasaan baru: sebelum menulis kode, selalu petakan dulu komponen mana saja yang terdampak. Kebiasaan ini tidak terbentuk dari perkuliahan, melainkan dari konsekuensi nyata ketika mengabaikannya.

Tantangan non-teknis yang juga memberikan pembelajaran bermakna adalah bekerja dalam tim dengan alur kerja profesional. Di perkuliahan, proyek kelompok tidak memiliki konsekuensi nyata jika kode yang digabungkan menghasilkan masalah. Di industri, setiap perubahan yang masuk melalui Merge Request bisa langsung berdampak pada pengguna aktif. Interaksi dengan tim QA secara konsisten membentuk perspektif baru: pekerjaan belum selesai hanya karena fitur berjalan sesuai yang diharapkan perlu juga dipastikan bahwa tidak ada bagian lain yang ikut terdampak. Kebiasaan memikirkan kemungkinan efek samping sebelum mengajukan Merge Request terbentuk bukan dari aturan formal, melainkan dari ritme kerja tim yang sudah terbiasa melakukannya.

Secara keseluruhan, magang ini mempertegas bahwa ada jarak yang cukup nyata antara pemahaman akademik dan tuntutan industri bukan berarti perkuliahan tidak memberikan bekal yang cukup, tetapi fondasi teoritis yang dibangun di kelas baru benar-benar teruji dan bermakna ketika dihadapkan pada sistem nyata dengan pengguna nyata. Justru di situlah nilai magang yang paling sulit digantikan oleh simulasi di lingkungan akademik.

BAB V
PENUTUP

5.1 Kesimpulan

Berdasarkan pelaksanaan kegiatan magang di PT Gamatechno Indonesia dan pembahasan pada BAB III dan BAB IV, seluruh tujuan yang ditetapkan dapat dinyatakan tercapai. Meskipun demikian, terdapat dua *task* yang telah di-merge ke repositori utama dan masih memerlukan proses verifikasi serta pengembangan lebih lanjut oleh tim pengembang.

Tujuan pertama terkait optimalisasi modul-modul sistem ERP Ganesha Operations telah terpenuhi melalui delapan *task* yang dikerjakan selama masa magang. Perbaikan dilakukan pada modul Penyetoran, Karyawan, Diskon, Entri Jurnal, Kontak, dan BIMKER dengan memanfaatkan mekanisme *inheritance* pada lapisan custom addons, sehingga perubahan yang

dilakukan tetap terisolasi dan tidak mengganggu kode inti Odoo maupun modul-modul lain yang sedang berjalan.

Tujuan kedua terkait peningkatan keakuratan dan keandalan data juga telah terpenuhi. Validasi ditambahkan pada beberapa modul, mulai dari validasi format data menggunakan @api.constrains pada modul Kontak, hingga validasi *mandatory* pada modul Karyawan dan Diskon yang mencegah data tersimpan dalam kondisi tidak lengkap.

Tujuan ketiga terkait penyesuaian tampilan antarmuka dan alur kerja turut terpenuhi. Beberapa tombol yang sudah tidak sesuai dengan proses bisnis disembunyikan, dan akses terhadap elemen antarmuka dibatasi berdasarkan status rekord yang sedang aktif, sehingga alur input pengguna menjadi lebih terarah dan potensi kesalahan pengisian data dalam operasional sehari-hari dapat dikurangi.

5.2 Saran

Meskipun seluruh tujuan magang telah tercapai sesuai dengan ruang lingkup pekerjaan yang diberikan, terdapat dua *task* yang setelah proses merge masih memerlukan proses verifikasi dan pengembangan lanjutan oleh tim pengembang. Oleh karena itu, pengembangan selanjutnya diharapkan dapat melanjutkan proses implementasi dan pengujian kedua *task* tersebut hingga siap digunakan secara optimal.

Meskipun tujuan pertama telah tercapai sesuai dengan ruang lingkup pekerjaan yang diberikan kepada penulis, terdapat dua *task*, yaitu penyesuaian logika Nominal Belum Disetor pada modul Penyetoran dan *filter* produk *expired* pada modul BIMKER, yang setelah di-merge ke repositori utama masih memerlukan proses verifikasi dan pengembangan lanjutan oleh tim pengembang karena kompleksitas integrasi antar modul. Untuk iterasi berikutnya, disarankan agar pengerjaan *task* serupa didahului dengan pemetaan menyeluruh terhadap seluruh modul yang saling terhubung sehingga dampak perubahan dapat diprediksi lebih awal dan siklus pengujian berjalan lebih efisien.

Dari sisi manfaat bagi Ganesha Operations, dua fitur yang belum berfungsi penuh berarti sebagian peningkatan akurasi data yang diharapkan belum dapat dirasakan pengguna. Disarankan agar tim pengembang yang melanjutkan kedua *task* tersebut memprioritaskan penyelesaiannya, mengingat modul Penyetoran berkaitan langsung dengan akurasi pelaporan keuangan cabang yang berdampak pada operasional sehari-hari.

Dari sisi dokumentasi teknis, sistem ERP Ganesha Operations memiliki banyak modul kustom yang tidak disertai dokumentasi yang memadai. Kondisi ini menyulitkan proses

analisis sebelum melakukan modifikasi dan memperlambat onboarding *developer* baru. Disarankan agar PT Gamatechno Indonesia menetapkan standar dokumentasi minimal untuk setiap modul kustom yang dikembangkan, mencakup penjelasan singkat tentang tujuan modul, relasi antar model yang terlibat, dan logika bisnis utama yang diimplementasikan.

DAFTAR PUSTAKA

Alamsyah, M. R., Wahanani, H. I., & Idhom. (2021). PENERAPAN DOCKER UNTUK MEMBANGUN INFRASTRUKTUR PRIVATE CLOUD STORAGE BERBASIS IAAS. In *Jurnal Informatika dan Sistem Informasi (JIFoSI)* (Vol. 2).

Arfikriyana, F., & Suhendi. (2022). Jurnal Informatika Terpadu BERBASIS OPEN SOURCE. *Jurnal Informatika Terpadu*, *8*(1), 13–20. Retrieved from https://journal.nurulfikri.ac.id/index.php/JIT

Aziza, & Rahayu. (2019). IMPLEMENTASI SISTEM ENTERPRISE RESOURCE PLANNING BERBASIS ODOO MODUL SALES DENGAN METODE RAD PADA PT XYZ. In *Journal Industrial Services* (Vol. 5).

Khalida, R., Muhajirin, A., Setiawati, S., Raya, J., Raya, J., & Utara, P. B. (2019). *Teknis Kerja Docker Container untuk Optimalisasi Penyebaran Aplikasi*.

Kurniawan, N., Yulia, K. S., & Irawan, E. C. (2021). *Kustomisasi dan Implementasi Odoo ERP: Studi Kasus Perusahaan Manufaktur PT. X*.

Laudon, K. C. ., & Laudon, J. P. . (2018). *Management information systems : managing the digital firm*. Pearson.

Megantara, R. A., Alzami, F., Pramunendar, R. A., & Prabowo, D. P. (2022). PENGEMBANGAN DAN IMPLEMENTASI DOCKER UNTUK MEMAKSIMALKAN UTILITAS SERVER UNIVERSITAS PADA MASA COVID-19. *Transmisi*, *24*(2), 48–54. https://doi.org/10.14710/transmisi.24.2.48-54

Monk, & Wagner. (2013). *Concepts in Enterprise Resource Planning*.

Murdiani, D., Yudhana, A., & Sunardi. (2020). *IMPLEMENTASI AGILE METHOD DALAM PENGEMBANGAN JURNAL ELEKTRONIK DI LEMBAGA PENELITIAN NON PEMERINTAHAN (NGO)*. *7*(4). https://doi.org/10.25126/jtiik.202071839

Soh, Siew Kien, & Joanne. (2000). *CULTURAL FITS AND MISFITS: IS ERP A UNIVERSAL SOLUTION?*

Sunardi, & Fadli, S. (2018). IDENTIFIKASI MASALAH PENERAPAN METODE AGILE (SCRUM) PADA PENGEMBANGAN PERANGKAT LUNAK DI PERGURUAN TINGGI (Studi Kasus Universitas Nahdlatul Ulama Nusa Tenggara Barat). In *Online) Jurnal Manajemen Informatika & Sistem Informasi* (Vol. 1). Cetak. Retrieved from Cetak website: http://e-journal.stmiklombok.ac.id/index.php/misi

Terminanto, A., & Ismail, M. F. (2022). IMPLEMENTASI MODUL ACCOUNTING BERBASIS ERP MENGGUNAKAN ODOO PADA LSP MUI. In *Jurnal Ilmiah Teknik Industri* (Vol. 10).

Yana, G., Sasmita, A., & Susila. (2022). Penerapan Enterprise Resource Planning (ERP) Menggunakan Odoo 14 (Studi Kasus: Usaha Garmen Pada Club Ink Bali. In *JITTER-Jurnal Ilmiah Teknologi dan Komputer* (Vol. 3).

LAMPIRAN

Lampiran A

