BAB III

PELAKSANAAN KERJA

3.1 Kedudukan dan Koordinasi

Selama Subab ini membahas kedudukan yang ditempati selama magang serta alur koordinasi dengan pembimbing lapangan, berikut penjelasan rincinya.

3.1.1 Kedudukan

Selama program magang di Kompas Gramedia, peserta magang ditempatkan di Departemen Enterprise Technology yang berada di bawah naungan CITIS (Corporate IT & IS) dengan posisi Intern Junior Software Engineer. Sepanjang periode magang, bimbingan dan supervisi langsung diberikan oleh seorang Senior Software Engineer yang berperan sebagai mentor sekaligus pengawas terhadap setiap pekerjaan yang diberikan. Tugas-tugas tetap berada dalam jalur pengawasan, dan hasilnya dilaporkan oleh supervisor kepada manajer departemen.

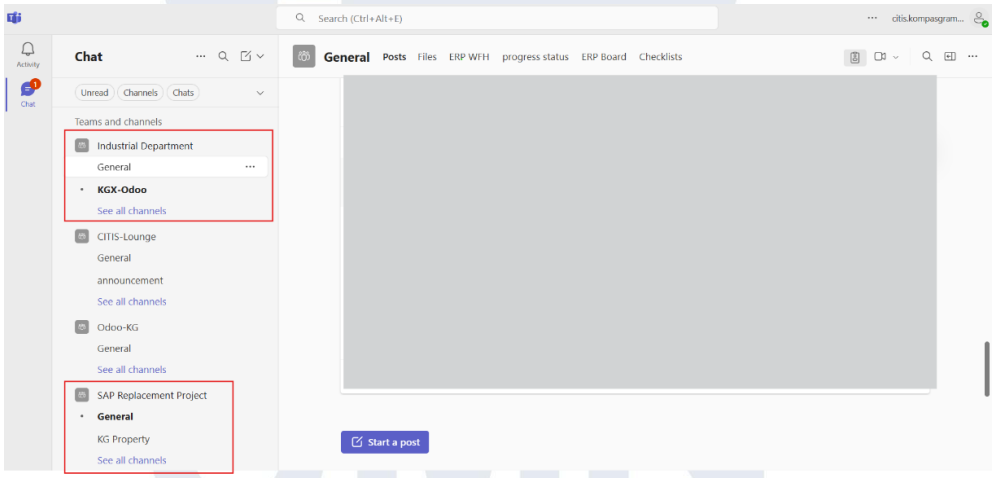
Dalam praktiknya, peran yang dijalankan adalah sebagai developer untuk mendukung kebutuhan Corporate Solution Department dan Industrial Solution Department. Kedua departemen tersebut berfungsi sebagai analis yang menyampaikan kebutuhan dan spesifikasi, sedangkan implementasi, penyesuaian, dan pemeliharaan sistem menjadi tanggung jawab developer sesuai permintaan.

3.1.2 Koordinasi

Koordinasi dalam pelaksanaan tugas dilakukan melalui dua metode utama. Pertama, melalui pertemuan langsung di kantor Kompas Gramedia Palmerah Selatan untuk membahas progres pekerjaan, mendiskusikan kendala yang muncul, serta memastikan setiap arahan dapat dipahami dengan jelas. Kedua,

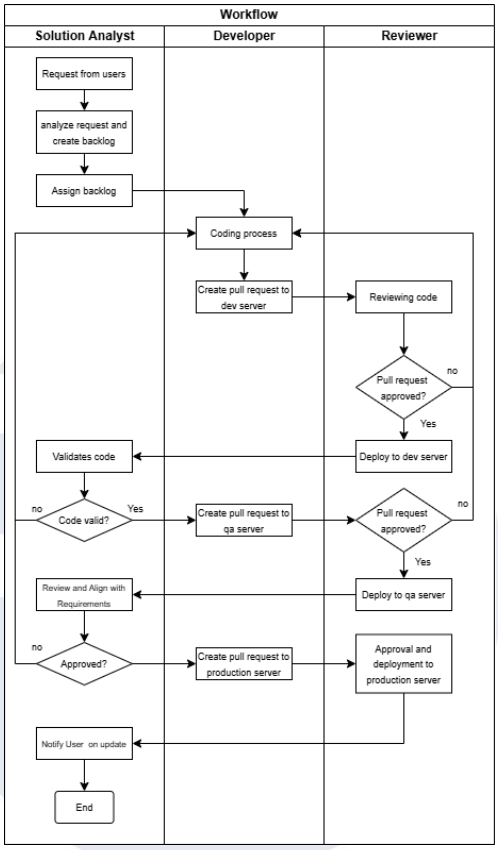
komunikasi juga dijalankan secara daring melalui platform Microsoft Teams, sehingga proses diskusi, pembaruan informasi, dan pelaporan hasil tetap berjalan efektif meskipun dilakukan secara jarak jauh.

Selain koordinasi harian tersebut, terdapat agenda rutin setiap dua minggu sekali pada hari Kamis berupa *sprint meeting industrial*. Pertemuan ini menjadi forum khusus untuk meninjau kembali capaian pekerjaan selama periode sprint sebelumnya sekaligus menyusun backlog atau rencana kerja untuk periode selanjutnya. Melalui forum ini, seluruh anggota tim dapat mengevaluasi hasil yang sudah dicapai, menyelaraskan kembali prioritas, dan memastikan arah pengembangan tetap sesuai dengan tujuan yang telah ditetapkan.



Gambar 3. 1 Tampilan Microsoft Teams

Melalui platform Microsoft Teams, koordinasi dilakukan bersama para Analyst yang berperan memberikan arahan mengenai kebutuhan sistem sekaligus memastikan setiap fitur yang dikembangkan tetap sesuai dengan spesifikasi yang sudah ditentukan. Untuk mendukung kelancaran komunikasi tersebut, perusahaan juga memberikan akses khusus agar peserta magang dapat terhubung langsung ke dalam ruang kerja Microsoft Teams. Tampilan akses tersebut dapat dilihat pada Gambar 3.1. Proses kerja selama magang dilaksanakan mengikuti alur sebagai berikut.



Gambar 3. 2 Alur Kerja dalam Corporate IT & IS Kompas Gramedia

Dapat filihat dari gambr 3.12 yang merupakan alur kerja yang mulai dari adanya permintaan dari user kemudian permintaan tersebut akan di diskusikan dengan analis, dan membuat *backlog* dan memberikan backlog tersebut kepada *developer.*

Setelah menerima tugas, *developer* akan mengerjakan permintaan tersebut sesuai dengan spesifikasi yang diberikan. Setelah proses pengembangan selesai, *developer* membuat *pull request* ke *development server*. Kode yang diajukan kemudian ditinjau oleh *reviewer*. Jika kode tersebut dinilai tidak aman atau belum memenuhi standar, *pull request* akan ditolak, sehingga *developer* harus melakukan perbaikan terlebih dahulu. Sebaliknya, jika kode telah dinyatakan aman dan sesuai standar, *reviewer* akan melanjutkan proses *deployment* ke *development server* atau biasanya disebut dev.

Setelah kode berhasil diunggah ke *development server*, kode tersebut akan diuji oleh analis untuk memastikan bahwa fungsionalitasnya sesuai dengan

permintaan pengguna (*user*). Apabila ditemukan ketidaksesuaian, analis akan memberikan umpan balik kepada *developer*, yang kemudian harus melakukan perbaikan. Sebaliknya, jika kode telah memenuhi kebutuhan, analis akan meminta *developer* untuk membuat *pull request* ke *QA server*.

Pull request ini kemudian ditinjau kembali oleh *reviewer*. Jika ditolak, proses akan kembali ke *developer* untuk melakukan perbaikan. Namun, jika disetujui, *reviewer* akan melanjutkan proses *deployment* ke *QA server*. Setelah itu, analis akan melakukan pengujian bersama *Financial System Developer* (*FSD*) untuk memastikan bahwa kode yang dikembangkan sesuai dengan standar etik dan kebijakan perusahaan.

Jika hasil pengujian masih belum sesuai, kode dikembalikan kepada *developer* untuk diperbaiki. Namun, jika kode telah memenuhi standar, *analis* akan meminta *developer* membuat *pull request* ke *production server*. Selanjutnya, *reviewer* akan meninjau dan melakukan *deployment* ke *production server*. Setelah semua tahap selesai, *analis* akan memberi konfirmasi kepada pengguna bahwa permintaan mereka telah berhasil diselesaikan

3.2 Tugas yang Dilakukan

Berisi tabel hal-hal yang dilakukan selama menjalankan program.

Tabel 3. 1 Detail Pekerjaan yang Dilakukan

No.	Minggu	Proyek	Keterangan
1	1-2	KG-ERP dan PMO	Instalasi Odoo, mempelajari alur bisnis, dan sharing session
2	3	KG-ERP	Membuat sub menu inventory dan berdiskusi dengan analis
3	4	KG-ERP	Memperbaiki text area di menu project
4	5	KG-ERP	Menutup akses button draft ketika statenya 'running'
5	6	KG-ERP	Melakukan perbaikan pada menu vendor bills
6	7	KG-ERP	Mengubah field destination project tidak dapat memanggil project yang sama
7	8	KG-ERP	Mengubah user access saat copy po di purchase order
8	9	KG-ERP	Menghilangkan create dari selection
9	10	KG-ERP dan PMO	Membuat group baru pada menu inventory viewer show price, memberikan penjagaan agar data kpi integration tidak dapat diedit
10	11	KG-ERP dan PMO	membuat Filter domain field return di stock picking type, dan juga menambahkan field reason di pr line
11	12	KG-ERP dan PMO	Menambahkan field draft pr qty di report stock request erp
12	13-15	PMO	Membuat menu KPI description
13	16	PMO	Menambah hak akses pada model pmo.kpi.integration
14	17	KG-ERP dan PMO	Menambahkan flag baru di update target/achivement pada PMO

3.3 Uraian Pelaksanaan Kerja

Subbab ini menguraikan secara rinci pelaksanaan kegiatan magang selama periode penugasan, mencakup proses pembelajaran (hard skill dan soft skill), kolaborasi tim, kontribusi pada pengembangan sistem, maintenance serta optimalisasi sistem, hingga kendala yang muncul beserta solusi yang diterapkan. Uraian dilengkapi peran dan tanggung jawab, metode atau alat yang digunakan, serta hasil yang dicapai pada tiap bagian. Berikut penjelasannya.

3.3.1 Instalasi Odoo, mempelajari alur bisnis, dan sharing session

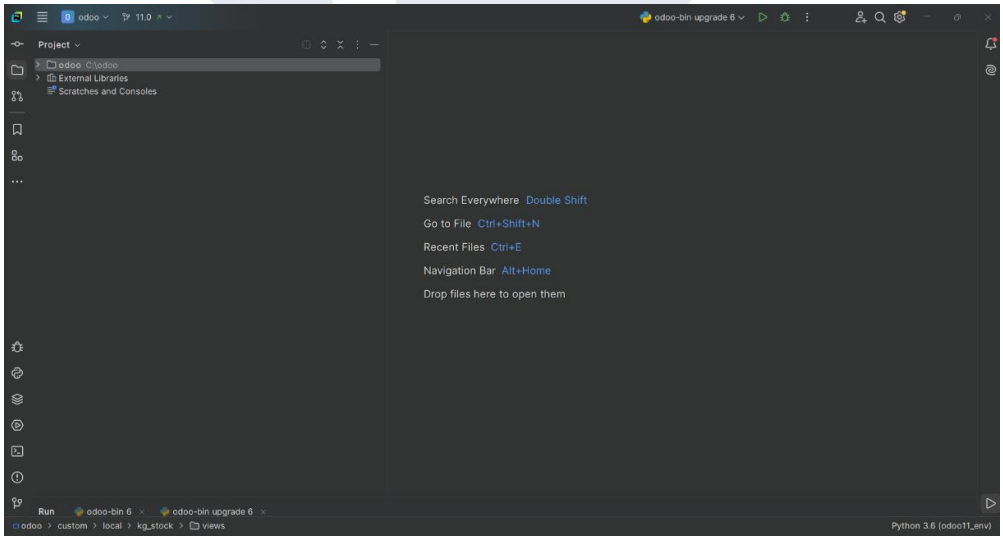
Pada minggu pertama dan kedua, dilakukan instalasi *Odoo 18 / 11*. Langkah pertama adalah menginstal *PyCharm* versi 2024.2.4 *Community Edition* sebagai *IDE* untuk menjalankan *Odoo* nantinya. Setelah itu, dibuat sebuah folder kosong berjudul odoo18 / 11 yang isinya ada 2 folder kosong yaitu *custom* dan *server* yang nantinya akan digunakan untuk menyimpan data *server* dan juga *local Odoo*. Selanjutnya, dilakukan instalasi *custom module* dari *Odoo 18* yang sebelumnya telah diberikan akses oleh perusahaan melalui *SSH key*. Setelah akses berhasil, *source code* asli *Odoo* disalin ke dalam folder *server* kosong yang telah disiapkan sebelumnya. Tahap berikutnya adalah melakukan pengaturan pada *file odoo.conf* untuk kebutuhan konfigurasi. Setelah itu, semua *requirements* diinstal agar sistem dapat berjalan dengan baik.

Setelah odoo sudah terinstall, dilakukan pembelajaran dan pengenalan alur bisnis. Proses pembelajaran mengenai alur sistem PMO dan KG-ERP sebagai langkah awal untuk memahami ruang lingkup pekerjaan. Proses ini dilaksanakan melalui beberapa cara, di antaranya mengikuti *sharing session* bersama *developer* lain untuk memperoleh gambaran umum mengenai alur kerja sistem, serta berdiskusi dengan senior untuk mendapatkan penjelasan lebih rinci terkait arsitektur dan integrasi sistem. Selain itu, dilakukan pula konsultasi dengan analis sistem guna memahami alur bisnis dan hubungan antar modul yang terdapat dalam sistem. Dokumentasi yang telah disusun

oleh senior juga dipelajari sebagai referensi tambahan. Melalui rangkaian kegiatan tersebut, diperoleh pemahaman menyeluruh mengenai cara kerja sistem PMO dan KG-ERP, baik dari sisi teknis maupun bisnis.

Selain mempelajari alur bisnis, dipelajari juga cara kerja tools tools yang digunakan. Ada berbagai macam tools utama digunakan untuk proses pengembangan, baik dari sisi teknis maupun kolaboratif. Untuk mendukung kegiatan kolaboratif, dimanfaatkan Microsoft Teams dan Azure DevOps. Sedangkan untuk kebutuhan pemrograman, digunakan PyCharm, DBeaver, dan Command Prompt.

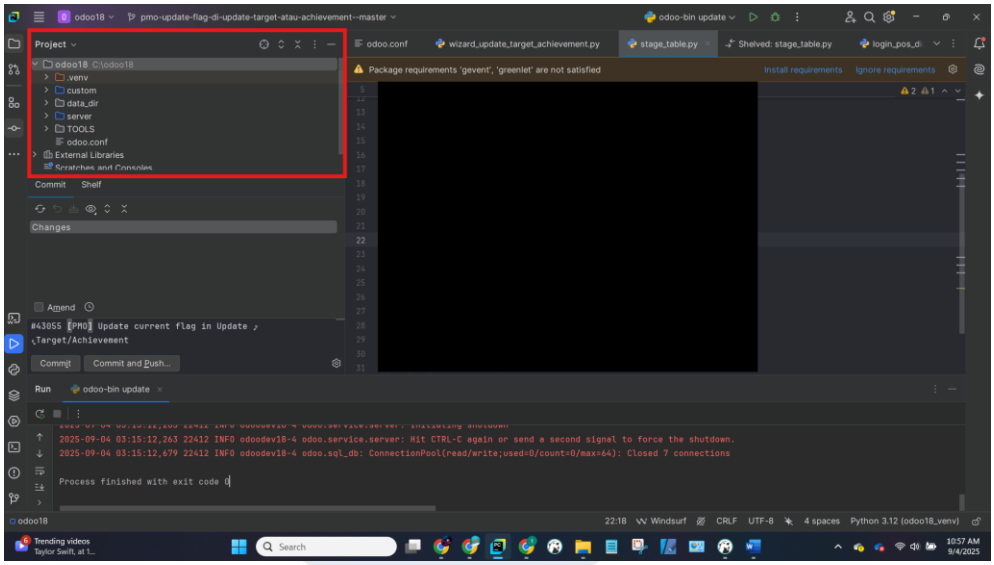
Tools yang pertama adalah PyCharm yang digunakan sebagai lingkungan pengembangan terintegrasi (IDE) yang mendukung bahasa Python, yang menjadi bahasa utama dalam pengembangan modul dan fitur pada framework Odoo. Gambar 3.3 adalah contoh tampilan antarmuka PyCharm.



Gambar 3. 3 Tampilan Antarmuka PyCharm

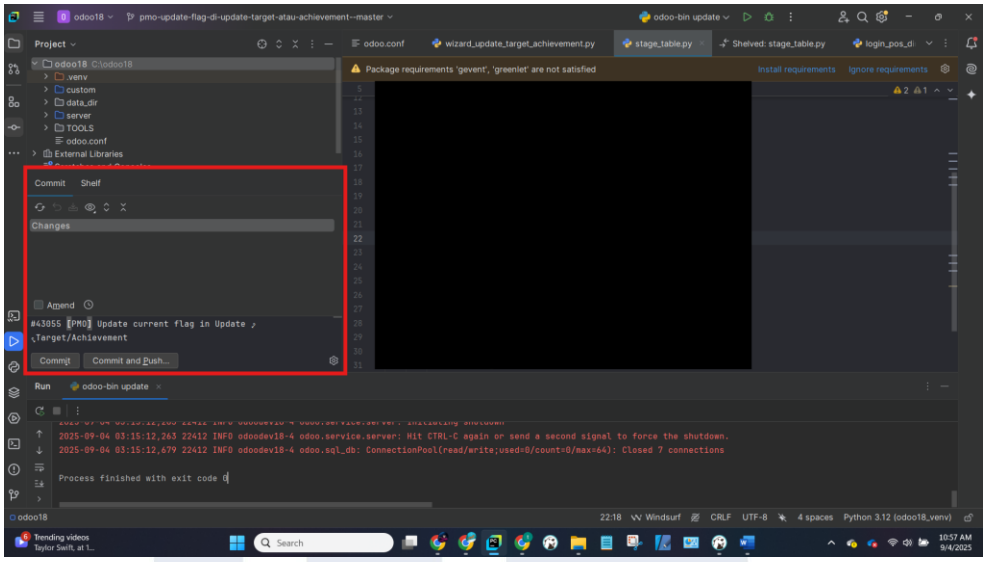
Untuk mendukung proses pengembangan, PyCharm menyediakan antarmuka yang terdiri dari beberapa bagian utama. Setiap bagian memiliki fungsi spesifik yang dirancang untuk mempermudah penulisan kode, pengelolaan proyek, serta proses *debugging*. Berikut adalah penjelasan mengenai bagian-bagian dari antarmuka PyCharm. Yang pertama ada lah bagian *Project Tool Window* seperti yang ditunjukkan pada gambar 3.4, bagian ini berfungsi untuk

menampilkan struktur proyek yang sedang dikerjakan. Melalui panel ini, file dan folder dapat diakses dengan mudah, sehingga memudahkan dalam pengelolaan maupun navigasi kode. Bagian ini biasanya terletak di sisi kiri antarmuka dan menjadi tempat utama untuk melihat keseluruhan isi proyek.



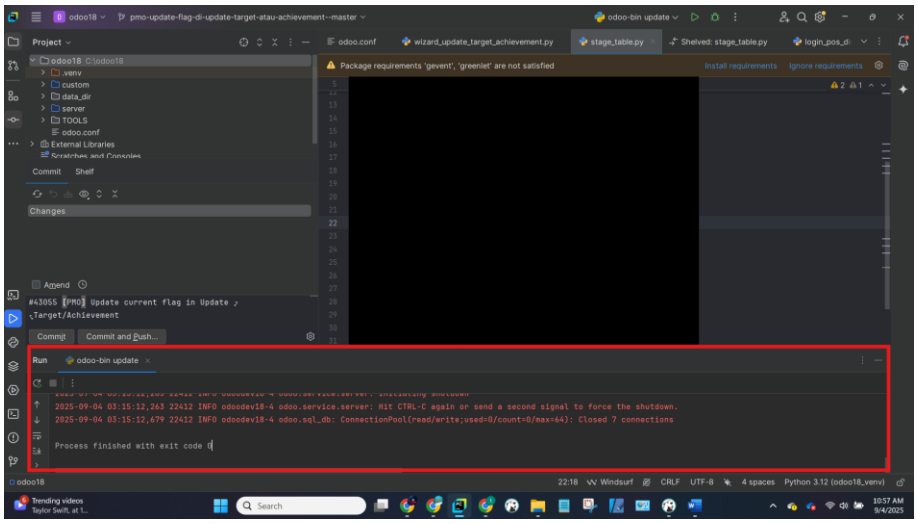
Gambar 3. 4 Tampilan Project Window pada PyCharm

Yang kedua adalah *Commit Tool Window* seperti yang ditunjukkan pada gambar 3.5 *Tool* ini digunakan untuk mengelola perubahan kode sebelum dikirim ke *repository*. Melalui panel ini, semua *file* yang telah diubah akan muncul, sehingga *developer* dapat mengecek kembali perubahan perubahan yang telah dilakukan, dan dapat membandingkannya dengan versi sebelum *code* diubah, dan dapat dipilih mana saja yang ingin di-*commit*. Selain itu, *developer* juga dapat menambahkan pesan *commit* agar setiap perubahan yang dilakukan lebih mudah dilacak.



Gambar 3. 5 Tampilan Commit Tool Window pada PyCharm

Yang ketiga adalah *Run Tool Window* seperti yang ditunjukkan pada gambar 3.6 . *Tools* ini berfungsi untuk menampilkan hasil eksekusi program yang dijalankan. Melalui panel ini, pengguna bisa melihat keluaran (*output*), pesan kesalahan (*error message*), maupun informasi proses yang sedang berlangsung. Bagian ini mempermudah pemantauan jalannya program sehingga jika terjadi kesalahan, dapat segera ditelusuri dan diperbaiki.

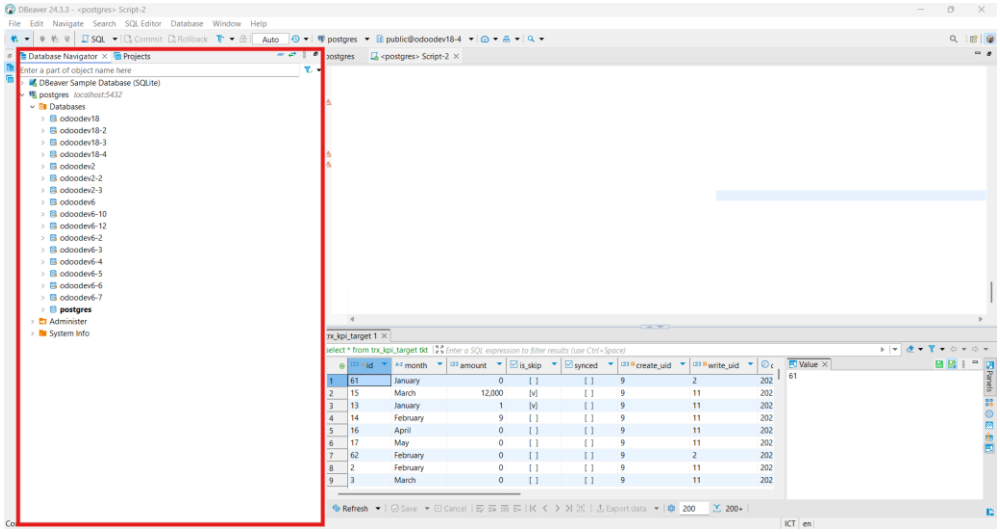


Gambar 3. 6 Tampilan Run Tool Window pada PyCharm

Tools kedua yang digunakan adalah Dbeaver. Aplikasi ini berfungsi sebagai *database management tool* yang memudahkan pengguna dalam mengelola,

mengeksekusi, serta memantau *query* pada basis data. Antarmuka DBeaver terdiri dari beberapa bagian utama yang saling mendukung dalam pengelolaan basis data. Bagian pertama adalah bagian *Database Navigator* seperti yang ditampilkan pada gambar 3.7. Bagian ini merupakan panel utama di *DBeaver* yang digunakan untuk melihat dan mengelola struktur *database*. Melalui bagian ini, pengguna dapat dengan mudah menjelajahi berbagai *schema*, tabel, *view*, prosedur, maupun objek lain yang ada dalam *database*. Tampilan hierarki yang ditawarkan membuat navigasi menjadi lebih sederhana, sehingga pengguna bisa langsung menemukan objek yang dibutuhkan tanpa harus mengetikkan perintah secara manual.

Selain itu, *Database Navigator* juga mempermudah interaksi dengan basis data. Misalnya, pengguna bisa melakukan *klik kanan* pada tabel untuk membuka data, mengeksekusi *query*, atau melihat detail struktur tabel tersebut. Dengan cara ini, proses pengelolaan *database* menjadi lebih cepat dan praktis, terutama saat harus bekerja dengan banyak tabel atau *schema* dalam satu proyek.

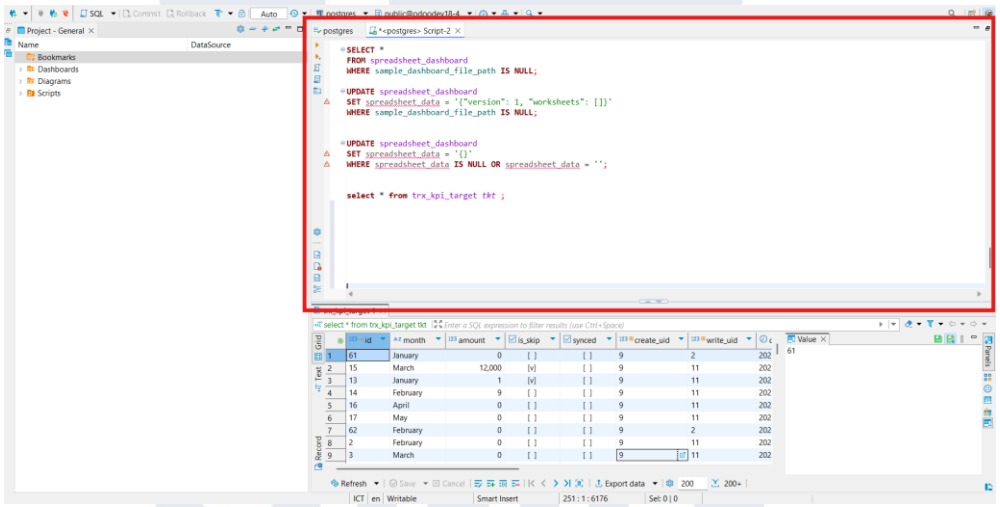


Gambar 3. 7 Tampilan Database Navigator pada Dbeaver

Bagian kedua adalah *SQL Editor* yang merupakan bagian di DBeaver yang digunakan untuk menulis dan menjalankan perintah SQL seperti yang

ditampilkan di gambar 3.8 . Melalui panel ini, pengguna bisa membuat, membaca, memperbarui, maupun menghapus data dengan menuliskan *query* sesuai kebutuhan. Tampilan editor yang interaktif dilengkapi dengan fitur *auto-complete*, pewarnaan sintaks, serta penanda kesalahan sehingga penulisan *query* menjadi lebih mudah dan minim kesalahan.

Tidak hanya itu, *SQL Editor* juga mendukung eksekusi *query* dalam skala kecil maupun besar. Hasil eksekusi akan langsung ditampilkan di panel bawah, sehingga pengguna dapat segera melihat output atau dampak dari perintah yang dijalankan. Fitur ini sangat membantu saat melakukan uji coba, analisis data, atau sekadar mengecek apakah *query* yang ditulis sudah sesuai dengan kebutuhan.

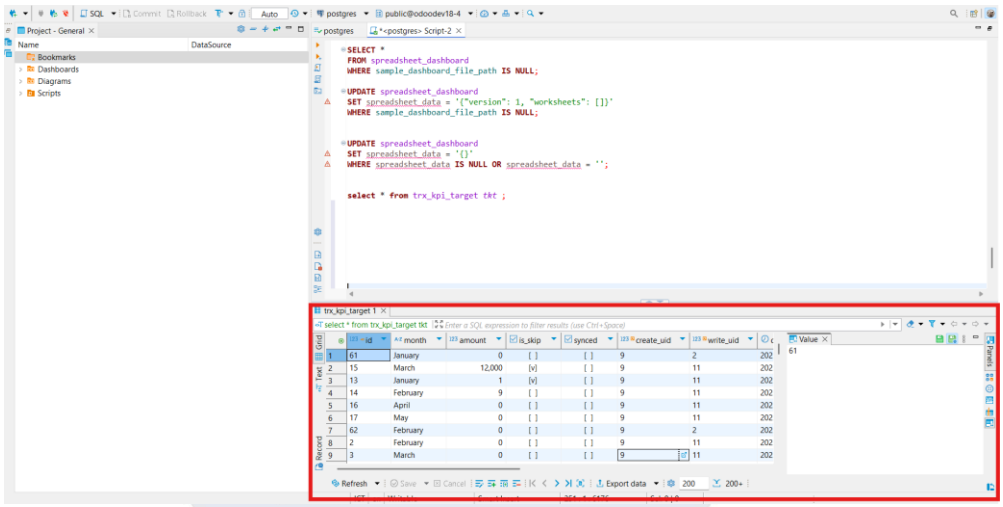


Gambar 3. 8 Tampilan SQL Editor pada Dbeaver

Bagian ketiga adalah *Results Panel* yang menampilkan hasil dari eksekusi *query* yang dijalankan di *SQL Editor*. Panel ini biasanya berada di bagian bawah layar, sehingga pengguna bisa langsung melihat data yang ditampilkan tanpa harus berpindah ke jendela lain. Informasi yang muncul bisa berupa tabel hasil seleksi, jumlah baris yang terpengaruh, hingga pesan kesalahan jika ada perintah yang tidak valid.

Selain hanya menampilkan hasil, *Results Panel* juga menyediakan berbagai opsi interaktif. Pengguna dapat melakukan penyaringan (*filter*),

pengurutan (*sort*), hingga mengekspor data ke berbagai format seperti CSV atau Excel. Dengan adanya fitur ini, analisis data menjadi lebih fleksibel karena hasil eksekusi *query* bisa langsung diproses atau disimpan untuk kebutuhan lebih lanjut. Tampilan *Result Panel* dapat dilihat pada gambar 3.9.



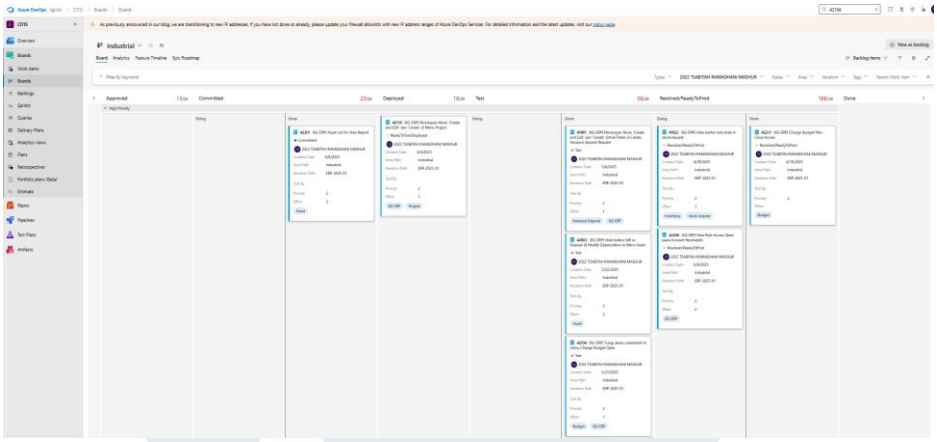
Gambar 3. 9 Tampilan Result Panel pada Dbeaver

Tools ketiga yaitu *Azure DevOps* tools yang digunakan untuk mengatur proyek sekaligus mendukung kerja tim. Di dalamnya ada beberapa fitur utama seperti *Azure Boards*, *Azure Repos*, dan *Azure Pipelines*.

Azure Boards digunakan sebagai sarana untuk mengelola *backlog* agar lebih teratur. Umumnya, analis menempatkan *backlog* yang berisi *task* pada kolom *Approved*. Setelah disetujui, *task* tersebut dapat langsung diambil oleh *developer* untuk dikerjakan. Jika sudah ditugaskan, *backlog* dipindahkan ke kolom *Committed (Doing)* sebagai penanda bahwa pekerjaan sedang berlangsung. Tampilan *Azure boards* dapat dilihat pada gambar 3.

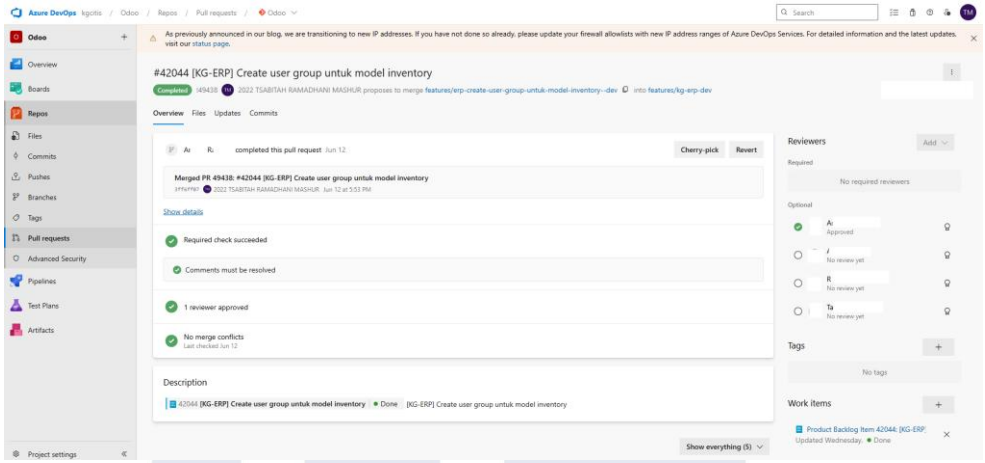
Apabila suatu *backlog* telah diselesaikan oleh *developer*, maka statusnya akan diperbarui dengan cara memindahkannya ke dalam kolom *Committed (Done)*. Tahap ini menandakan bahwa *task* yang bersangkutan sudah selesai dikerjakan dari sisi pengembangan. Selanjutnya, apabila *task* tersebut berhasil melalui proses *deployment* ke server, maka *backlog* akan dipindahkan kembali oleh *developer* menuju kolom *Deployed*. Perubahan status ini menunjukkan bahwa pekerjaan sudah berada pada tahap

implementasi di lingkungan server. Setelah mencapai tahap tersebut, seluruh proses selanjutnya akan menjadi tanggung jawab analis untuk melakukan pemeriksaan, validasi, maupun tindak lanjut yang diperlukan.



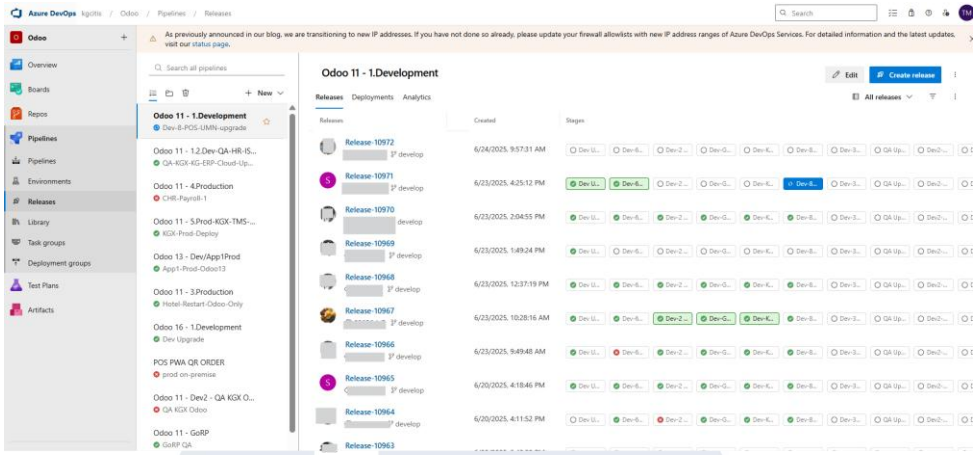
Gambar 3. 10 Tampilan Antarmuka Azure Boards

Azure Repos dimanfaatkan sebagai sarana utama untuk melakukan *pull request* dalam proses pengelolaan kode. Setelah seorang *developer* menyelesaikan pengerjaan suatu *backlog*, langkah berikutnya adalah membuat *pull request* melalui *Azure Repos*. Melalui mekanisme ini, setiap perubahan yang telah dilakukan pada kode tidak langsung digabungkan ke *branch* utama, melainkan terlebih dahulu ditinjau oleh pihak yang berwenang, biasanya seorang manajer atau *senior developer* yang berperan sebagai *reviewer*. Proses peninjauan tersebut mencakup pemeriksaan detail terhadap kualitas kode, kesesuaian dengan standar pengembangan, serta potensi dampaknya terhadap sistem secara keseluruhan. Dengan adanya tahap *review* ini, dapat dipastikan bahwa setiap pembaruan kode telah melalui proses verifikasi yang cermat sebelum akhirnya diintegrasikan ke *branch* utama. Tampilan *Azure Repos* dapat dilihat pada gambar 3.



Gambar 3. 11 Tampilan Antarmuka Azure Repos

Azure Pipelines digunakan sebagai sarana utama dalam mendukung proses *deployment*. Setelah perubahan kode yang diajukan melalui *pull request* memperoleh persetujuan, sistem *pipeline* akan berjalan secara otomatis untuk melakukan serangkaian proses, mulai dari *build* hingga *deployment* ke server. Dengan adanya mekanisme otomatis ini, setiap pembaruan kode tidak hanya dapat diintegrasikan dengan lebih cepat, tetapi juga dapat melalui tahap pengujian secara konsisten tanpa perlu dilakukan secara manual. Penggunaan *Azure Pipelines* memberikan manfaat signifikan dalam hal efisiensi, karena proses integrasi, pengujian, hingga penerapan hasil pekerjaan dari *backlog* dapat dilakukan dalam satu alur kerja yang terstruktur. Tampilan *Azure Pipeline*s dapat dilihat pada gambar 3.12



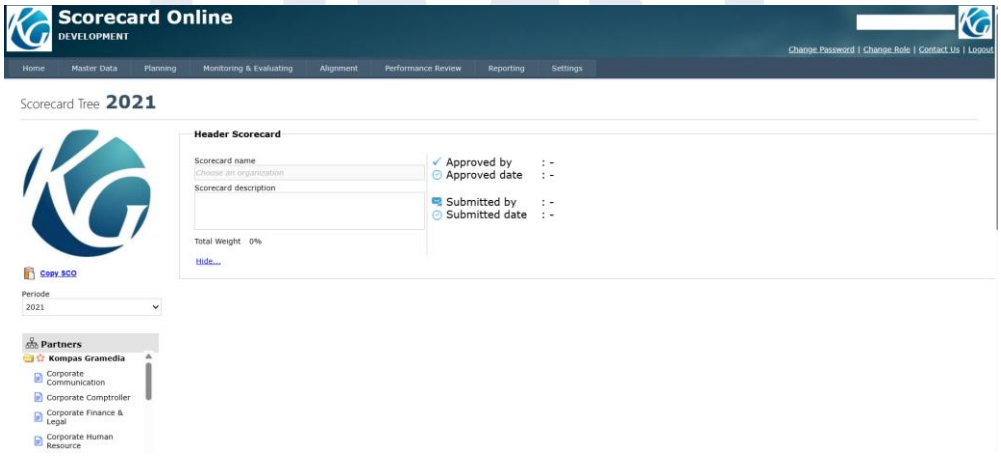
Gambar 3. 12 Tampilan Antarmuka Azure Pipelines

Dalam pembelajaran alur bisnis ERP dan PMO, dilakukan dengan cara berdiskusi dengan senior, dengan analis, developer lain, dan juga melihat dokumentasi perusahaan berikut adalah penjelasan dari masing masing sistem tersebut Sistem pertama adalah *Performance Management Online (PMO)*, yaitu sebuah sistem yang dirancang untuk memonitor serta mengevaluasi capaian kinerja perusahaan secara menyeluruh. Sistem ini berfungsi sebagai alat bantu dalam memantau pelaksanaan strategi organisasi, terutama melalui pengukuran terhadap pencapaian *Key Performance Indicator (KPI)* dari setiap unit bisnis yang berada di bawah naungan Kompas Gramedia. Dengan adanya *PMO*, manajemen perusahaan dapat memperoleh gambaran yang jelas mengenai tingkat keberhasilan setiap unit dalam mencapai target yang telah ditetapkan. Informasi yang tersaji tidak hanya bersifat administratif, tetapi juga mendukung proses evaluasi kinerja secara kuantitatif maupun kualitatif, sehingga hasil pemantauan menjadi lebih objektif dan dapat dipertanggungjawabkan.

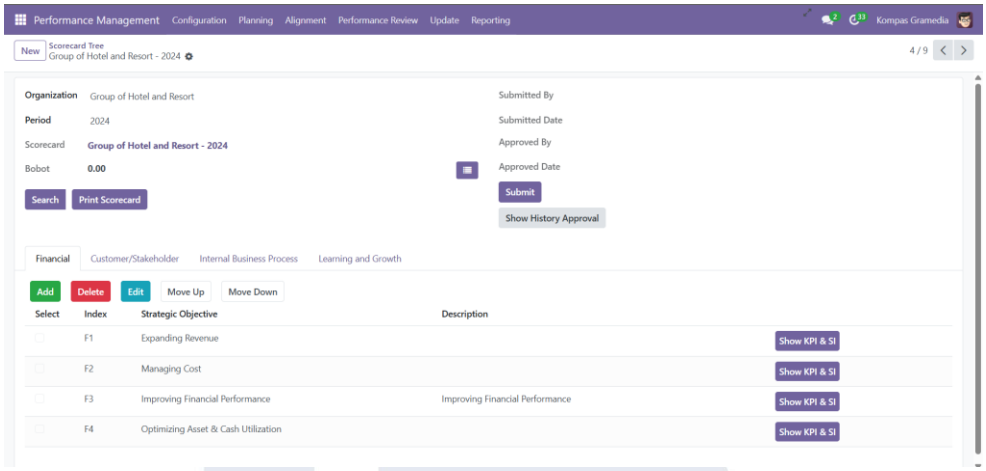
Selain itu, *PMO* dirancang agar mampu menyajikan data kinerja secara *real-time* dan terintegrasi, sehingga pihak manajemen dapat segera mengetahui perkembangan capaian yang sedang berlangsung. Hal ini memberikan nilai tambah dalam proses pengambilan keputusan, karena keputusan strategis dapat dilakukan berdasarkan data aktual yang akurat dan terkini. Lebih jauh

lagi, sistem ini juga membantu menciptakan transparansi dalam proses evaluasi kinerja, di mana setiap unit bisnis dapat melihat posisi capaian mereka masing-masing terhadap target yang telah ditentukan. Hasil dari *PMO* pada masing-masing divisi nantinya akan dipresentasikan kepada petinggi Kompas Gramedia sebagai bentuk laporan resmi untuk mengetahui perkembangan kinerja divisi tersebut. Dengan demikian, *PMO* tidak hanya menjadi alat pengawasan, tetapi juga sarana untuk mendorong peningkatan kinerja dan akuntabilitas di seluruh lini bisnis perusahaan.

Saat ini, sistem *Performance Management Online (PMO)* yang sedang dikembangkan merupakan *PMO* versi 4. Berbeda dengan versi sebelumnya yang tidak terintegrasi di dalam *Odoo* dan dibangun secara terpisah menggunakan platform *.NET*, versi terbaru ini diarahkan untuk berjalan langsung di dalam ekosistem *Odoo*, lebih tepatnya pada *Odoo 18*. Pada versi sebelumnya, sistem *PMO* menghadapi beberapa keterbatasan, terutama pada sisi antarmuka yang terkesan "tua" serta kinerja yang cukup lambat, sehingga kurang mendukung pengalaman pengguna secara optimal. Perbandingan Tampilan PMO versi 3 dan versi 4 ditunjukkan pada gambar 3.13 Dan 3.14



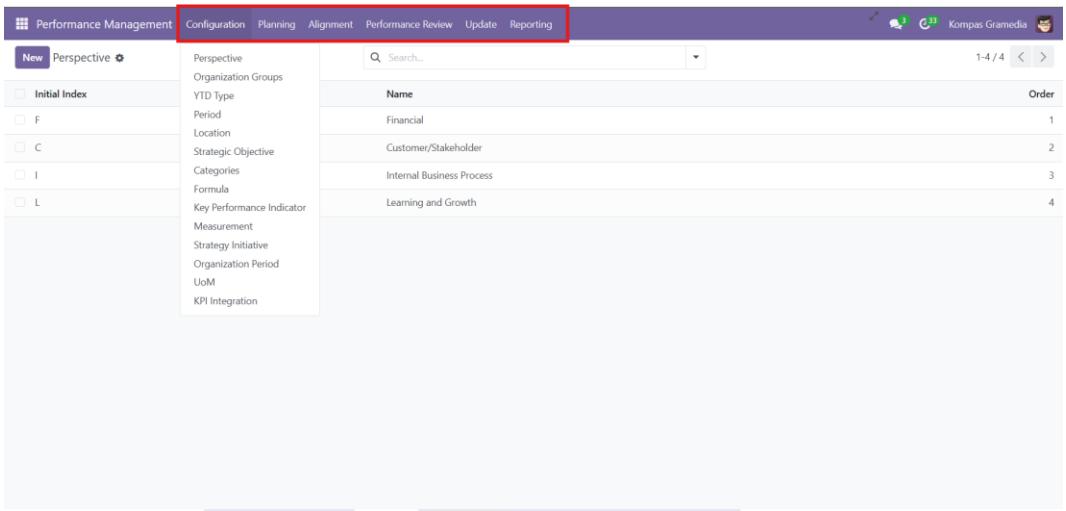
Gambar 3. 13 Tampilan Performance Management Online Versi 3



Gambar 3. 14 Tampilan Performance Management Online Versi 4

Pengembangan *PMO* versi 4 ini hadir sebagai upaya perbaikan sekaligus modernisasi, dengan tujuan menghadirkan tampilan antarmuka yang lebih segar, responsif, serta mendukung kinerja yang lebih cepat. Dengan integrasi langsung ke dalam *Odoo 18*, sistem tidak hanya lebih stabil, tetapi juga memiliki fleksibilitas lebih tinggi untuk dikembangkan dan diadaptasikan dengan kebutuhan pengguna di masa mendatang. Perubahan ini diharapkan dapat meningkatkan efisiensi penggunaan, mempercepat akses data kinerja, sekaligus memperbaiki kualitas pengalaman pengguna ketika berinteraksi dengan sistem.

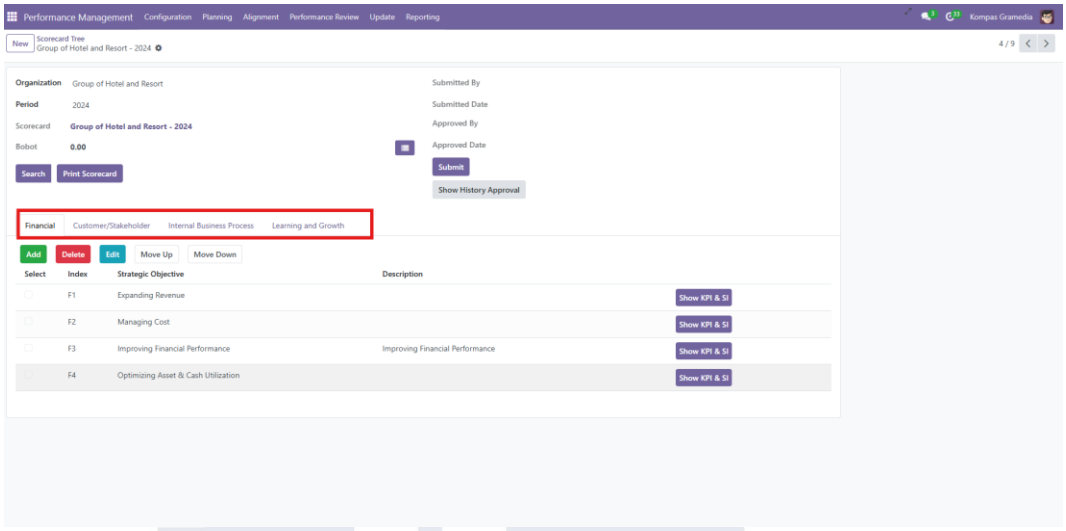
Pada Odoo 18, menu utama dari sistem *Performance Management Online (PMO)* diberi nama *Performance Management*. Di dalam menu ini tersedia enam *sub menu* utama, yaitu *Configuration*, *Planning*, *Alignment*, *Performance Review*, *Update*, dan *Reporting* seperti yang ditunjukkan pada gambar 3.15. Akses penuh terhadap keenam *sub menu* ini diberikan hanya kepada pengguna dengan peran sebagai *PMO Admin*. Meskipun kerangka enam *sub menu* ini telah terbentuk, pada saat ini sistem masih dalam proses pengembangan sehingga sebagian fitur belum tersedia secara lengkap.



Gambar 3. 15 Tampilan Sub Menu Utama dari PMO

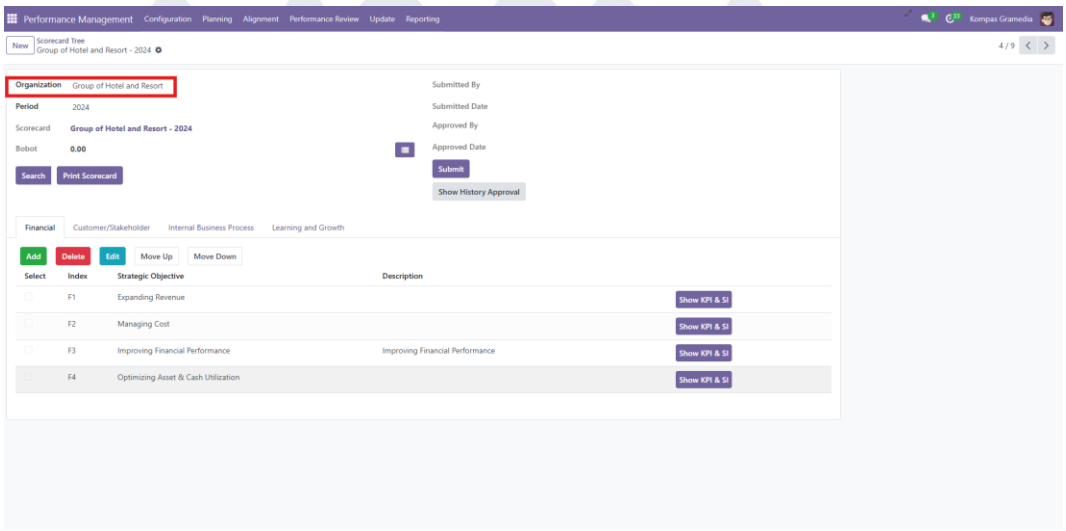
Sub menu pertama yang terdapat pada *Performance Management* adalah *Configuration*. Bagian ini berfungsi sebagai pusat pengaturan dasar yang nantinya akan digabungkan dan ditampilkan dalam bentuk *Scorecard Tree*. Salah satu *configuration* utama yang digunakan adalah *Perspective*, yaitu kerangka pengukuran kinerja yang menjadi dasar dalam menilai pencapaian setiap unit bisnis. Di Kompas Gramedia, terdapat empat perspektif utama yang dijadikan acuan, yaitu *Financial*, *Customer*, *Internal Business Process*, serta *Learning and Growth* seperti yang ditunjukkan pada gambar 3.16 .

Berdasarkan hasil wawancara dengan *senior* di perusahaan, diketahui bahwa perspektif ini bersifat sangat stabil dan jarang mengalami perubahan. Umumnya, perubahan perspektif hanya terjadi dalam kurun waktu yang cukup panjang, bahkan bisa mencapai sepuluh tahun sekali, tergantung pada kebijakan strategis perusahaan. Stabilitas ini menunjukkan bahwa kerangka perspektif yang digunakan telah sesuai dengan kebutuhan organisasi dalam jangka panjang.



Gambar 3. 16 Perspective pada Performance Management Online

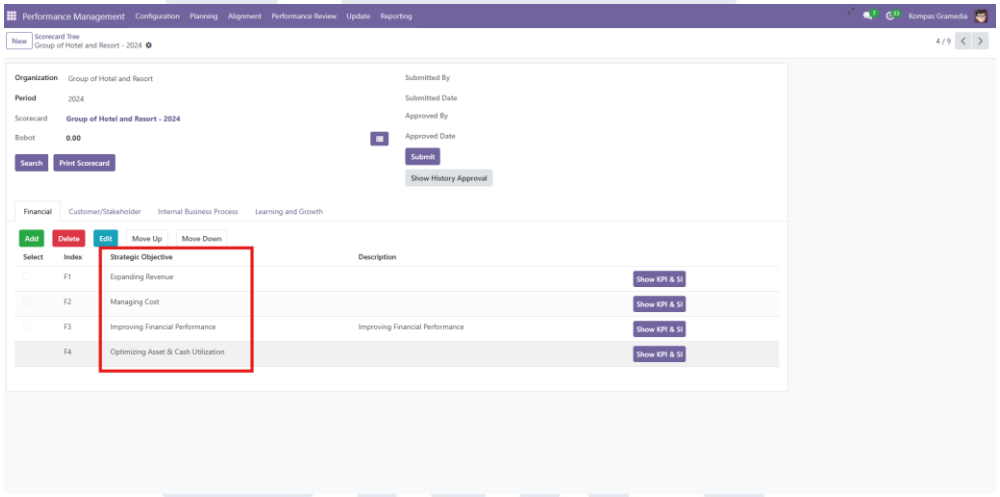
Kemudian, pada *Configuration* juga terdapat sub menu yang disebut *Organization Group* sebagaimana ditunjukkan pada Gambar 3.17 Bagian ini berperan sebagai cabang utama dalam struktur sistem, karena menjadi titik awal dalam penyusunan hierarki pengukuran kinerja. Melalui *Organization Group*, setiap unit organisasi dikelompokkan dan dipetakan sesuai dengan kedudukan serta fungsinya di dalam perusahaan.



Gambar 3. 17 Organization Group pada Performance Management Online

Selanjutnya, di dalam *Configuration* juga terdapat komponen *Strategic Objective (SO)* sebagaimana ditunjukkan pada Gambar 3.18 *Strategic Objective* merupakan tujuan besar atau utama yang ingin dicapai oleh perusahaan sesuai dengan masing-masing *perspective* yang telah ditetapkan. Dengan adanya *SO*, arah pencapaian kinerja perusahaan menjadi lebih jelas, karena setiap sasaran dirumuskan secara spesifik sesuai dengan kerangka perspektif yang berlaku.

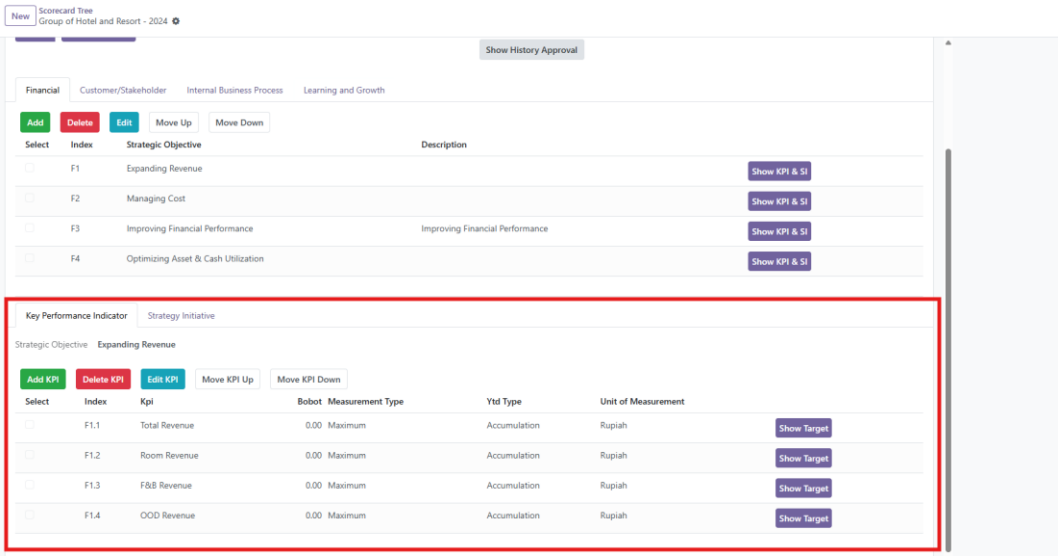
Strategic Objective ini juga berhubungan langsung dengan cabang pada *Organization Group*, sebab setiap unit organisasi memiliki tujuan strategisnya masing-masing. Dengan demikian, setiap *SO* yang ditetapkan mencerminkan target yang relevan dengan peran dan fungsi dari unit tersebut dalam mendukung visi perusahaan secara keseluruhan.



Gambar 3. 18 Strategic objective pada Performance Management Online

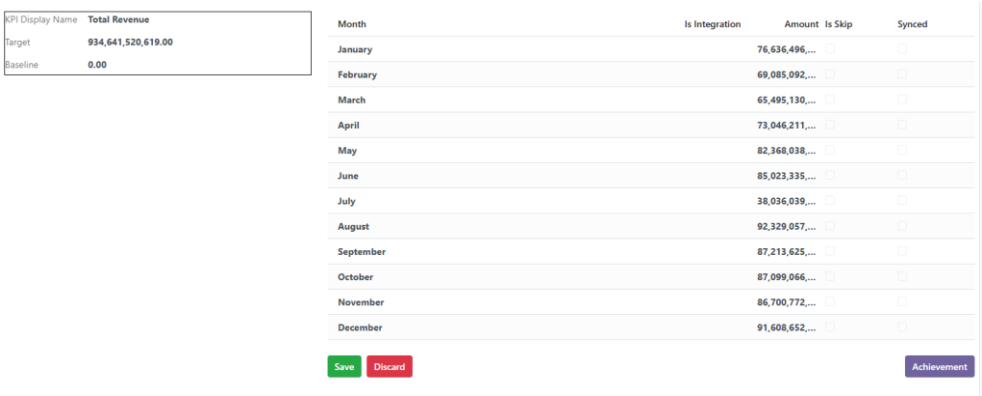
Selain *Strategic Objective*, pada *Configuration* juga terdapat komponen *Key Performance Indicator (KPI)* sebagaimana ditunjukkan pada Gambar 3.19 KPI merupakan ukuran spesifik yang digunakan untuk menilai sejauh mana suatu *Strategic Objective* telah tercapai. Dengan kata lain, apabila SO dipandang sebagai tujuan utama yang ingin diwujudkan oleh perusahaan, maka KPI berperan sebagai alat ukur untuk memastikan apakah tujuan tersebut sudah terlaksana sesuai target yang ditetapkan.

Melalui KPI, capaian kinerja dapat dievaluasi secara kuantitatif maupun kualitatif, tergantung pada indikator yang dipilih. Setiap SO umumnya memiliki satu atau lebih KPI sebagai penanda keberhasilan, sehingga perusahaan dapat memantau perkembangan pencapaian kinerja secara lebih detail dan terukur.



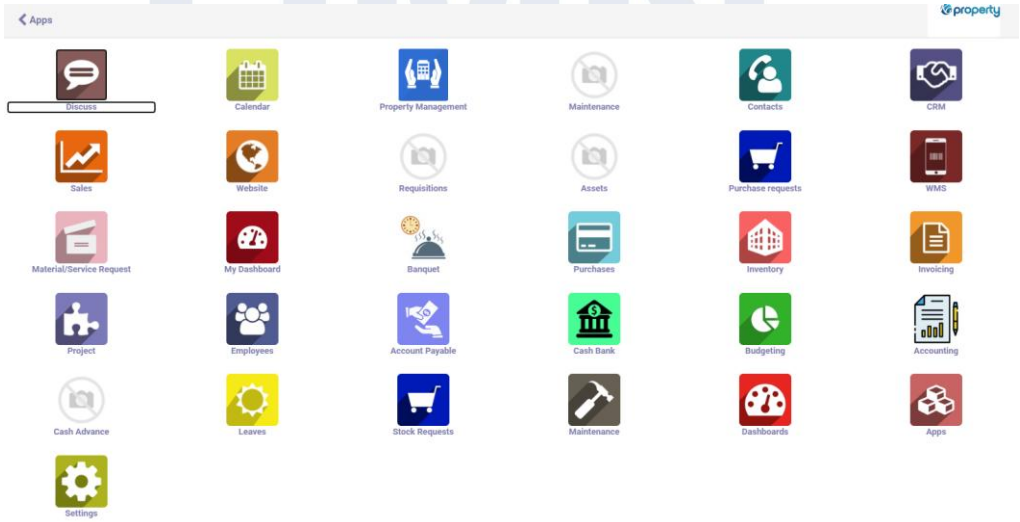
Gambar 3. 19 Key performance indicator pada Performance Management Online

Setiap KPI dalam sistem PMO dapat menampilkan target kinerja secara lebih rinci, baik per bulan maupun dalam bentuk akumulasi tahunan, karena pada dasarnya capaian KPI dihitung dalam periode satu tahun. Seperti yang ditunjukkan pada Gambar 3.20 , data KPI memiliki dua sumber utama, yaitu melalui proses *integration* dari sistem luar serta yang dibuat secara manual langsung di dalam aplikasi. Mekanisme ini memberikan fleksibilitas dalam pengelolaan indikator kinerja sesuai kebutuhan masing-masing unit bisnis.



Gambar 3. 20 Target KPI pada Performance Management Online

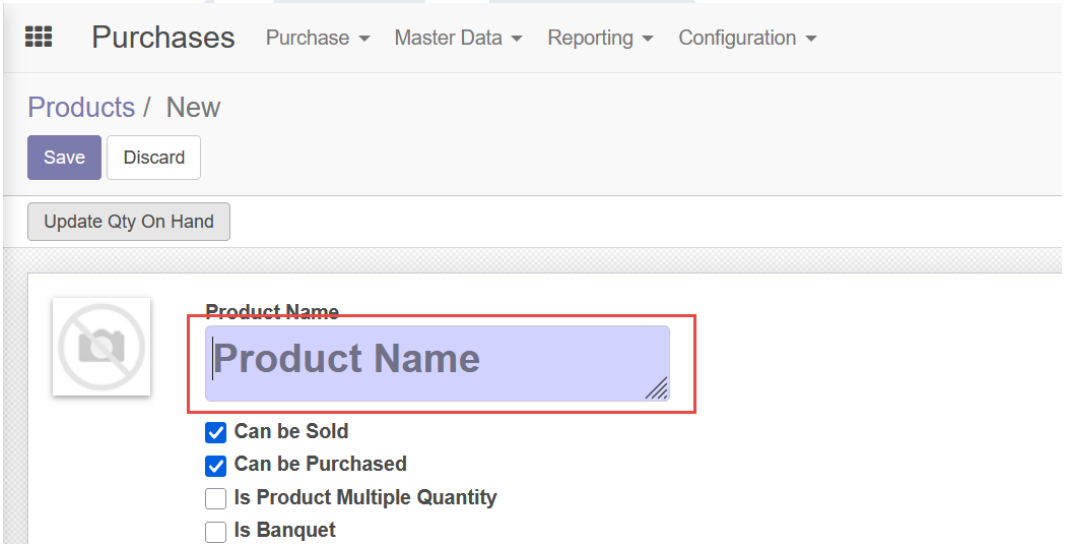
Kemudian sistem yang kedua adalah *KG-ERP (Enterprise Resource Planning)*, yaitu sistem terpadu yang dikembangkan secara khusus untuk memenuhi kebutuhan *user* dalam mengelola berbagai fungsi bisnis perusahaan, seperti keuangan, sumber daya manusia, manajemen operasional, dan lain-lain. Dalam proses pengembangannya, *KG-ERP* diarahkan untuk mempermudah integrasi data antarbagian sehingga mengurangi terjadinya duplikasi informasi serta meningkatkan efisiensi kerja. Selain itu, sistem ini juga dilengkapi dengan fitur yang mendukung otomatisasi proses, sehingga aktivitas bisnis yang sebelumnya dilakukan secara manual dapat diselesaikan lebih cepat, terstruktur, dan minim kesalahan tampilan dari menu menu yang tersedia pada sistem KG-ERP dapat dilihat pada gambar 3.21



Gambar 3. 21 Tampilan Mernu Menu KG-ERP

3.3.2 Memperbaiki text area di menu project

Pada menu product pengguna mengajukan permintaan untuk melakukan perbaikan pada *text box* judul produk. Menurut laporan pengguna, lebar kolom yang ada saat ini terlalu pendek seperti yang ditunjukkan pada gambar 3.22, sehingga perlu diperpanjang untuk dapat menampung input teks secara lengkap. Permintaan ini kemudian dipernbaiki dengan melakukan modifikasi pada lebar kolom tersebut.



Gambar 3. 22 Tampilan Text Box Produk

Implementasi perbaikan ini dilakukan pada antarmuka (*front-end*) dengan cara mengubah aturan *styling* (CSS) pada elemen *textbox* product. atribut width diatur ke nilai 60% untuk memperlebar bidang isian.

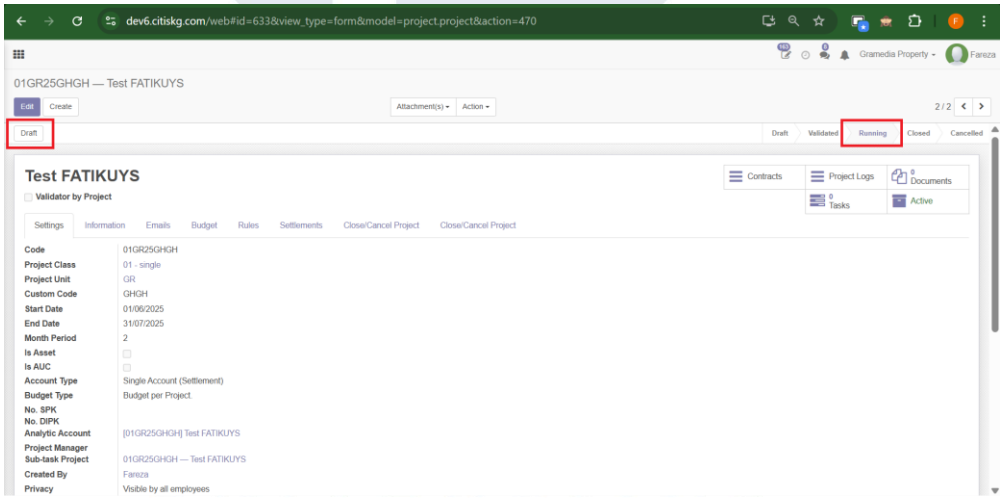
```
<xpath expr="//sheet" position="inside">
    <style>
        .oe_title {
            width: 60% !important;
            max-width: 60% !important;
        }
    </style>
</xpath>
```

Gambar 3. 23 Kode Pengaturan Lebar dari Text Box

Kode pengaturan lebar dari text box

3.3.3 Menutup akses button draft ketika statenya 'running'

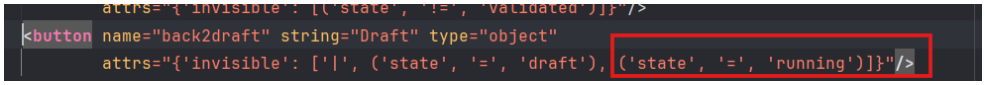
Pada menu project analis meminta perbaikan untuk menambah kondisi pada button draft di model project.project seperti yang ditunjukkan pada gambar 3.24



Gambar 3. 24 Tampilan Button Draft pada Menu Project

Pada awalnya button ini hanya muncul saat state dari projectnya bukan draft, karena asumsi bahwa seluruh state dapat diubah lagi menjadi draft kecuali draft itu sendiri. Namun ada kebutuhan khusus dari analis untuk pencegahan pengguna mengubah state project dari running menjadi draft lagi. Untuk membuat pencegahan ini dilakukan penyesuaian pada file XML

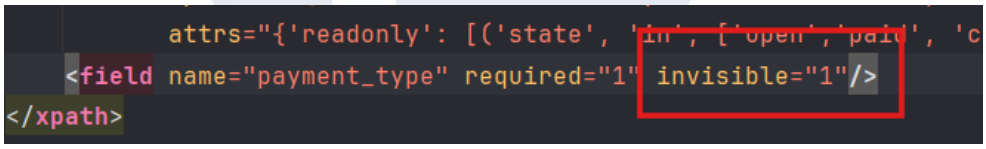
dengan menambahkan kondisi baru yaitu tombol ini disembunyikan saat statenya running seperti yang ditunjukkan pada gambar 3.25



Gambar 3. 25 Kode Penambahan Attribut Penjagaan State Running

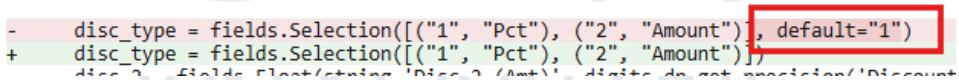
3.3.4 Melakukan perbaikan pada menu vendor bills

Pada menu invoicing lebih tepatnya sub menu vendor bills, analis meminta bebrapa perbaikan yaitu yang pertama menghilangkan atau menyembunyikan fields payment_type, karena fields ini tidak perlu ditunjukkan atau dilihat oleh pengguna. Kemudian yang kedua field disc_type diminta saat belum diisi user kolomnya kosong bukan memiliki isi. Dalam penyelesaiannya dilakukan beberpa penyesuaian pada XML nya yaitu menambahkan atrribut *invisible* agar field payment_type menjadi hilang pada tampilan, potongan kode ini dapat dilihat pada gambar 3.26



Gambar 3. 26 Kode Penambahan Attribut Invisible

Kemudian untuk domain disc_type dilakukan penyesuaian pada file Python nya, yaitu dengan menghilangkan attribut defaultnya, sehingga saat page diload fieldnya menjadi kosong, potongan kode ini dapat dilihat pada gambar 3.27

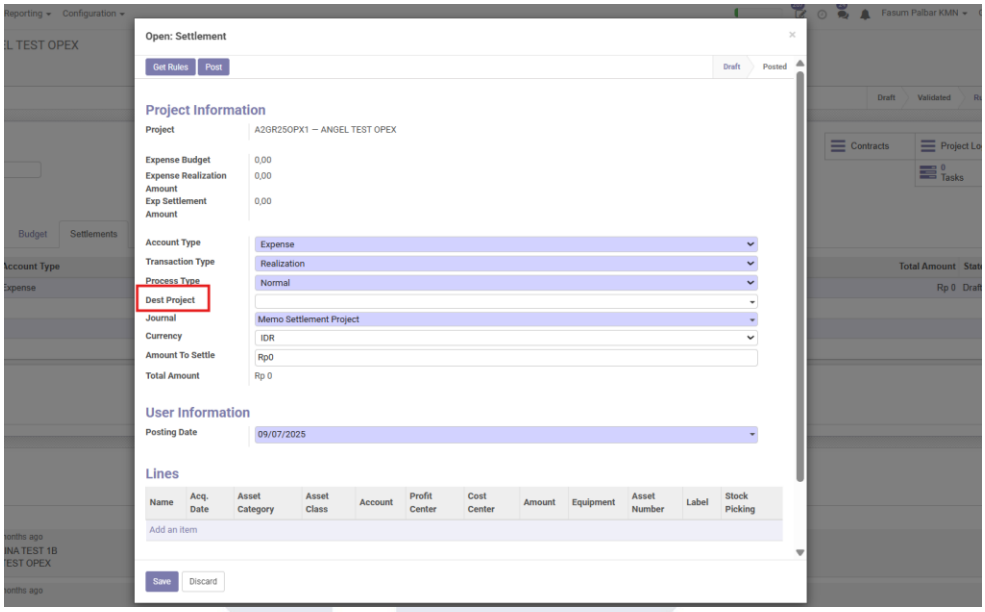


Gambar 3. 27 Kode penghapusan Attribut Default

3.3.5 Mengubah field destination project tidak dapat memanggil project yang sama

Pada menu project di tab settlements, analis meminta sedikit perbaikan pada domain fields dest project yang ditunjuk pada gambar 3.28 Sebelumnya domain dari field dest project tidak ada penyesuaian apa apa, jadi yang ditunjukkan seluruh domain dest project. Pengguna meminta agar domain

yang muncul tidak boleh memanggil project itu sendiri. karena dengan memanggil projectnya sendiri, dapat membuat looping tanpa henti, untuk mencegah hal ini perlu adanya penyesuaian.



Gambar 3. 28 Tampilan Fields Dest Project

Dalam penyelesaiannya dibuat penyesuaian pada bagian file XML nya. Penyesuaian ini berupa penambahkan syarat domain, dimana domain yang muncul dari dest project hanya yang id dari project.settlement nya tidak sama dengan project_id seperti yang ditunjukkan pada gambar 3.29

```
<field name="dest_project_id" force_save="1"
       attrs="{'invisible': [('account_type', 'not in', ('inc', 'exp'))], 'readonly': [('state', '=', 'posted')]}"
       options="{'no_create': True, 'no_open':True}"
       domain="[('id', '!=', project_id)]"/>
```

Gambar 3. 29 Kode Penyesuaian Domain Dest Project

3.3.6 Mengubah user access saat copy po di purchase order

Pada menu purchase, sub menu purchase order – spk. pengguna meminta agar tetap dapat menyalin PO yang ada sebagai dasar pembuatan PO baru, meskipun sistem memberlakukan aturan *'Not allow direct PO without PR'*. Karena sebelumnya group tersebut tidak dapat menyalin PO, dilakukan perubahan pada logika pemogramannya, dimana user group *'Not allow direct PO without PR'* masih dapat meng-copy po namun hanya dapat copy po dan membuat po dari purchase request, bukan membuat po baru tanpa purchase

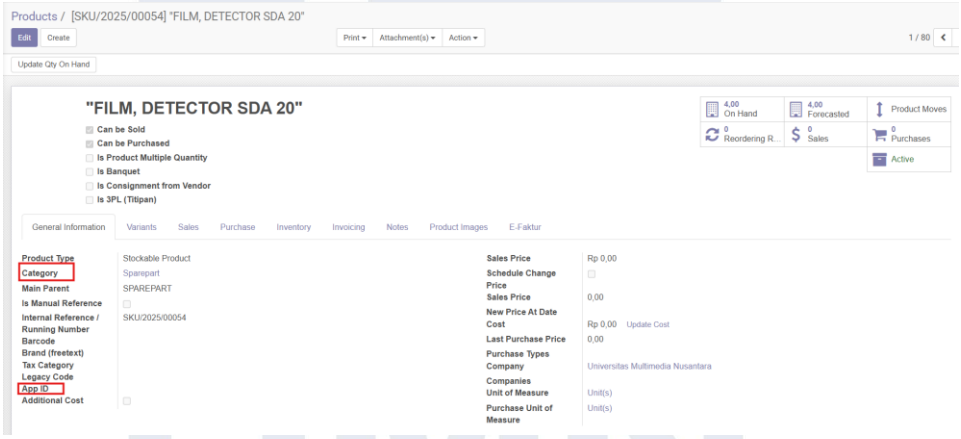
request.

```
def create(self, vals):
    if self.env.user.has_group('kg_purchase.group_not_allow_po_without_pr'):
        is_copy = vals.get('old_order_id')
        if not vals.get('from_purchase_request') and not is_copy:
            raise UserError("You can only create purchase order from purchase request.")
```

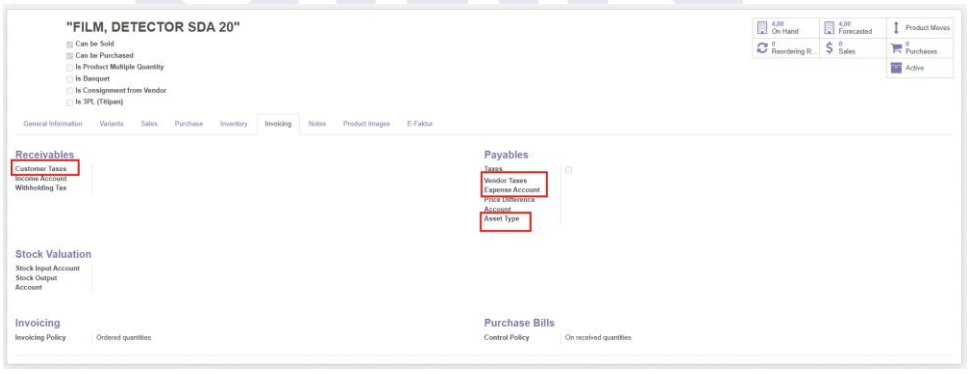
Gambar 3. 30 Kode Penyesuaian hak Group

3.3.7 Menghapus Create dari Selection

Pada menu project pengguna meminta beberapa penyesuaian yaitu menghilangkan akses '*create and edit'* pada beberapa fields yaitu fields *Category, APP ID* pada *tab general information* Dan fields *Customer, Vendor Taxes, Expense Account, Asset Type* pada tab *invoicing* seperti yang ditunjukkan pada gambar 3.31 Dan gambar 3.32



Gambar 3. 31 Fields Category dan App Id pada Product



Gambar 3. 32 Fields customer, vendor taxes,espense account, asset type pada product

Dengan menghapus akses '*create and edit'* ini pengguna tidak dapat menambah data baru dari fields fields tersebut, karena dengan adanya

penambahan data baru bisa saja merusak hierarki datanya, jadi pengguna cukupu menggunakan data yang sudah ada. Untuk menghilangkan akses ini dilakukan penyesuaian pada file xml nya, penyesuaian ini menambahkan attribut baru yaitu *no create edit*

```
<xpath expr="//field[@name='categ_id']" position="attributes">
    <attribute name="domain">categ_id_domain</attribute>
    <attribute name="options">{'no_create': True}</attribute>
    <attribute name="context">{'form_view_readonly': True}</attribute>
</xpath>
<xpath expr="//field[@name='asset_category_id']" position="attributes">
    <attribute name="options">{'no_create': True}</attribute>
</xpath>
<xpath expr="//field[@name='supplier_taxes_id']" position="attributes">
    <attribute name="options">{'no_create': True}</attribute>
</xpath>
<xpath expr="//field[@name='property_account_expense_id']" position="attributes">
    <attribute name="options">{'no_create': True}</attribute>
</xpath>
<xpath expr="//field[@name='taxes_id']" position="attributes">
    <attribute name="options">{'no_create': True}</attribute>
</xpath>
```

Gambar 3. 33 Kode Penambahan Attribut pada Fields

Setelah kode ini diterapkan, maka pengguna tidak dapat lagi membuat data baru, sehingga mencegah kebingungan sistem karena datanya yang kurang lengkap.

3.3.8 Membuat group baru pada menu inventory viewer show price

Salah satu modul yang terdapat pada sistem KG-ERP adalah *Inventory*, yang berfungsi untuk mengelola persediaan barang serta memastikan proses distribusi berjalan dengan lancar. Pada awalnya, modul *Inventory* hanya memiliki tiga jenis pengguna (*role*), yaitu *Manager*, *Logistic Supervisor*, dan *User*. Ketiga jenis pengguna ini secara default diberikan hak akses penuh berupa *Create, Read, Update, Delete (CRUD)* pada sebagian besar menu di dalam *Inventory*. Meskipun hak akses penuh ini mempermudah fleksibilitas dalam penggunaan, di sisi lain terdapat kebutuhan dari pihak pengguna yang menghendaki adanya pembatasan akses tertentu.

Berdasarkan permintaan analis tersebut, muncul kebutuhan untuk menambahkan satu jenis *role* baru, yaitu *Viewer*. Peran ini dirancang khusus untuk karyawan yang hanya memerlukan akses dalam bentuk *Read Only*, tanpa adanya kewenangan untuk melakukan *Create, Update*, maupun *Delete*. Dengan adanya *Viewer*, beberapa karyawan dapat tetap memantau data *Inventory* sesuai kebutuhan pekerjaan mereka, namun di saat yang sama risiko perubahan data yang tidak semestinya dapat diminimalisir.

Dalam proses penambahan *role* baru tersebut, langkah pertama yang dilakukan adalah melakukan inisialisasi *group* baru dengan nama *Viewer*. *Group* ini berfungsi sebagai wadah utama yang nantinya digunakan untuk mengatur hak akses pengguna dengan tipe *Read Only* pada modul *Inventory*. Dengan adanya *group Viewer*, sistem dapat secara jelas membedakan antara pengguna dengan hak akses penuh (*Manager, Logistic Supervisor, User*) dan pengguna yang hanya memiliki kewenangan untuk melakukan *viewing*.

Kode inisialisasi *group Viewer* ini dapat dilihat pada Gambar 3.34 Pada tahap ini, dipastikan bahwa *group Viewer* dapat dikenali sistem sebagai salah satu entitas baru di dalam daftar *user groups*.

```
<record id="group_stock_viewer" model="res.groups">
    <field name="name">Viewer</field>
    <field name="category_id" ref="base.module_category_warehouse_management"/>
    <field name="implied_ids" eval="[(4, ref('base.group_user'))]"/>
</record>
```

Gambar 3. 34 Kode Inialisasi Group Viewer

Setelah *group Viewer* berhasil dibuat, langkah berikutnya adalah mengatur hak akses pengguna di dalam modul *Inventory*. Karena peran *Viewer* memang hanya ditujukan untuk melihat data, maka hak aksesnya dibatasi hanya pada fungsi *Read* saja. Dengan begitu, pengguna dengan peran ini tidak bisa melakukan *Create, Update,* maupun *Delete*, sehingga interaksi mereka dengan data benar-benar sebatas pemantauan.

Pengaturan hak akses ini ditetapkan langsung pada level *model*, yang menjadi dasar dari pengelolaan data di sistem KG-ERP. Dengan cara ini, sistem bisa lebih presisi dalam membatasi apa yang boleh dan tidak boleh

dilakukan oleh setiap pengguna. Pada Gambar 3.35 ditunjukkan kode yang mengatur akses tersebut, di mana *Viewer* hanya diberi izin untuk membaca data pada model-model yang terkait dengan *Inventory*.

```
access_stock_picking_viewer,kg_stock.stock_picking_viewer,stock.model_stock_picking,kg_stock.group_stock_viewer,1,0,0,0
access_stock_move_viewer,kg_stock.stock_move_viewer,stock.model_stock_move,kg_stock.group_stock_viewer,1,0,0,0
access_stock_move_line_viewer,kg_stock.stock_move_line_viewer,stock.model_stock_move_line,kg_stock.group_stock_viewer,1,0,0,0
access_stock_inventory_viewer,stock_inventory_viewer,stock.model_stock_inventory,kg_stock.group_stock_viewer,1,0,0,0
access_stock_inventory_line_viewer,stock_inventory_line_viewer,stock.model_stock_inventory_line,kg_stock.group_stock_viewer,1,0,0,0
access_stock_picking_type_viewer,stock_picking_type_viewer,stock.model_stock_picking_type,kg_stock.group_stock_viewer,1,0,0,0
access_stock_production_lot_viewer,stock_production_lot_viewer,stock.model_stock_production_lot,kg_stock.group_stock_viewer,1,0,0,0
access_stock_warehouse_orderpoint_viewer,stock_warehouse_orderpoint_viewer,stock.model_stock_warehouse_orderpoint,kg_stock.group_stock_viewer,1,0,0,0
access_stock_quant_viewer,stock_quant_viewer,stock.model_stock_quant,kg_stock.group_stock_viewer,1,0,0,0
access_stock_quant_package_stock_viewer,stock_quant_package_stock_viewer,stock.model_stock_quant_package,kg_stock.group_stock_viewer,1,0,0,0
access_procurement_rule_viewer,procurement_rule_viewer,stock.model_procurement_rule,kg_stock.group_stock_viewer,1,0,0,0
access_stock_location_path_viewer,stock_location_path_viewer,stock.model_stock_location_path,kg_stock.group_stock_viewer,1,0,0,0
access_stock_fixed_putaway_viewer,stock_fixed_putaway_viewer,stock.model_stock_fixed_putaway_strat,kg_stock.group_stock_viewer,1,0,0,0
access_product_price_history_stock_viewer,product_price_history_stock_viewer,product.model_product_price_history,kg_stock.group_stock_viewer,1,0,0,0
access_barcode_nomenclature_stock_viewer,barcode_nomenclature_stock_viewer,barcodes.model_barcode_nomenclature,kg_stock.group_stock_viewer,1,0,0,0
access_barcode_rule_stock_viewer,barcode_rule_stock_viewer,barcodes.model_barcode_rule,kg_stock.group_stock_viewer,1,0,0,0
access_stock_forecast_viewer,stock_forecast_viewer,stock.model_report_stock_forecast,kg_stock.group_stock_viewer,1,0,0,0
access_stock_scrap_viewer,stock_scrap_viewer,stock.model_stock_scrap,kg_stock.group_stock_viewer,1,0,0,0
```

Gambar 3. 35 Kode Penetapan Hak Akses Model Kepada Group Viewer

Setelah pengaturan hak akses pada level *model* selesai dilakukan, langkah berikutnya adalah mengatur hak akses di level *menu item*. Pada tahap ini, pengguna dengan peran *Viewer* diberikan akses terbatas hanya pada beberapa menu tertentu di modul *Inventory*. Menu yang dapat diakses antara lain *Dashboard, Operation, Master Data,* serta sebagian menu pada *Reporting*.

Khusus untuk menu *Reporting*, tidak semua laporan dapat dilihat oleh pengguna *Viewer*. Beberapa laporan yang bersifat sensitif hanya dapat diakses oleh peran *Manager* atau *Supervisor* (*SPV*), sesuai dengan kebutuhan dan tanggung jawab masing-masing jabatan. Dengan pembatasan ini, sistem tetap bisa memberikan transparansi informasi kepada pengguna *Viewer*, namun tetap menjaga kerahasiaan data yang dianggap penting dan strategis.

Pada Gambar 3.36 ditunjukkan potongan kode yang digunakan untuk mengatur hak akses pada *menu item* ini. Melalui pengaturan tersebut, pengguna dengan peran *Viewer* hanya dapat membuka menu yang relevan dengan kebutuhan pemantauan, tanpa memiliki kemungkinan untuk masuk ke menu lain yang memang tidak diperuntukkan bagi mereka.

```
<menuitem action="action_stock_putaway_request"
          name="Putaway Requests"
          parent="stock.menu_stock_warehouse_mgmt"
          id="menu_stock_putaway_request"
          sequence="125"
          groups="stock.group_stock_user,kg_stock.group_stock_viewer"/>
```

Gambar 3. 36 Kode Penetapan Hak Akses Menu kepada Group Viewer

Namun, setelah sistem *Viewer* ini diterapkan, muncul kebutuhan baru dari pihak pengguna. Mereka menginginkan agar tidak semua pengguna dengan peran *Viewer* memiliki akses untuk melihat informasi terkait *cost* pada laporan. Hal ini dikarenakan data *cost* dianggap sensitif dan hanya boleh diakses oleh pengguna tertentu yang memang membutuhkan informasi tersebut untuk menunjang pekerjaannya.

Untuk menjawab kebutuhan tersebut, solusi yang diambil adalah membuat *additional access* berupa kelompok baru dengan nama *inventory (Show Price)*. Peran tambahan ini secara khusus memungkinkan pengguna untuk melihat data *cost* pada laporan, sementara *Viewer* biasa tetap tidak memiliki akses ke informasi tersebut. Dengan begitu, sistem dapat lebih fleksibel dalam membedakan kebutuhan pengguna, tanpa harus memberikan akses yang berlebihan kepada semua *Viewer*.

Pada Gambar 3.37 ditunjukkan kode inisialisasi untuk *group inventory (Show Price)*. Inisialisasi ini memastikan bahwa sistem dapat mengenali kelompok baru tersebut sebagai *user group* yang terpisah, sehingga hak akses dapat diatur lebih detail sesuai kebijakan.

```
<record id="group_stock_viewer_show_price" model="res.groups">
    <field name="name">Inventory show price</field>
    <field name="category_id" ref="module_additional_stock"/>
    <field name="comment">Jika true, maka user dapat melihat harga produk.</field>
</record>
```

Gambar 3. 37 Kode Inialisasi Group Inventory Show Price

Selain melakukan penambahan *role* baru, dilakukan juga penyesuaian pada hak akses di beberapa *menu item*. Hal ini karena tidak semua laporan (*report*) seharusnya dapat diakses oleh pengguna dengan peran *Viewer*. Mengingat struktur hierarki akses di dalam modul *Inventory* disusun mulai dari *Manager* > *Logistic Supervisor* > *User* > *Viewer*, maka pembatasan akses perlu dilakukan dengan cara membatasi hak akses tertentu hanya sampai level *User* saja.

Dengan penyesuaian tersebut, laporan-laporan yang bersifat lebih teknis atau strategis, seperti laporan terkait performa pergerakan barang, tidak dapat

diakses oleh *Viewer*. Hanya pengguna dengan peran minimal *User* yang berhak untuk membuka laporan tersebut. Tujuannya adalah agar data yang dianggap lebih sensitif tetap berada di bawah kendali pengguna yang memiliki tanggung jawab lebih besar, sementara *Viewer* hanya dapat mengakses informasi yang bersifat umum. Contoh dari penyesuaian ini ditunjukkan pada Gambar 3.38, yang memperlihatkan bagaimana sebuah *menu item* diatur agar hanya bisa diakses oleh kelompok pengguna tertentu.

```
<!-- Menu item to open Wizard  -->
<menuitem id="menu_wizard_kg_stock_over_period"
          name="Stock Over Period"
          sequence="13"
          parent="menu_wizard_kg_stock_report_group"
          action="action_wizard_kg_stock_over_period"
          groups="stock.group_stock_user"
/>
```

Gambar 3. 38 Kode Penyesuaian Pembatasan Akses Report

Untuk membedakan antara *Viewer* biasa yang tidak memiliki akses melihat *cost* dan *Viewer* dengan hak akses tambahan (*Viewer Show Price*), dilakukan pemisahan file laporan (*MRT file*) menjadi dua versi yang berbeda. Pemisahan ini bertujuan agar sistem dapat menampilkan laporan sesuai dengan hak akses yang dimiliki oleh masing-masing pengguna.

Selain itu, dibuat juga logika khusus pada bagian query untuk mengatur apakah nilai *cost* akan ditampilkan atau tidak. Dengan cara ini, ketika pengguna hanya tergabung dalam *group Viewer* biasa, maka data *cost* secara otomatis tidak muncul di laporan. Sebaliknya, jika pengguna termasuk dalam kelompok *Viewer Show Price* atau memiliki peran lebih tinggi seperti *Logistics Supervisor*, maka informasi *cost* akan ditampilkan secara lengkap. Implementasi logika tersebut ditunjukkan pada Gambar 3.39 Dan gambar 3.40

```
if self.env.user.has_group("kg_stock.group_stock_viewer_show_price") or self.env.user.has_group("kg_stock.group_stock_logistics_spv"):
    cost_fields = """
            COALESCE(sm.price_unit, 0) as cost,
            COALESCE(sm.value, 0) as total_amount,
        """
    debit_credit_select = """
            COALESCE(aml_se.debit, 0) as debit,
            COALESCE(aml_t.credit, 0) as credit,
    """
    debit_credit_group = """
            aml_se.debit,aml_t.credit,
    """
else:
    cost_fields = ""
    debit_credit_select = ""
    debit_credit_group = ""
```

Gambar 3. 39 Kode Menentukan User Dapat Melihat Cost atau Tidak

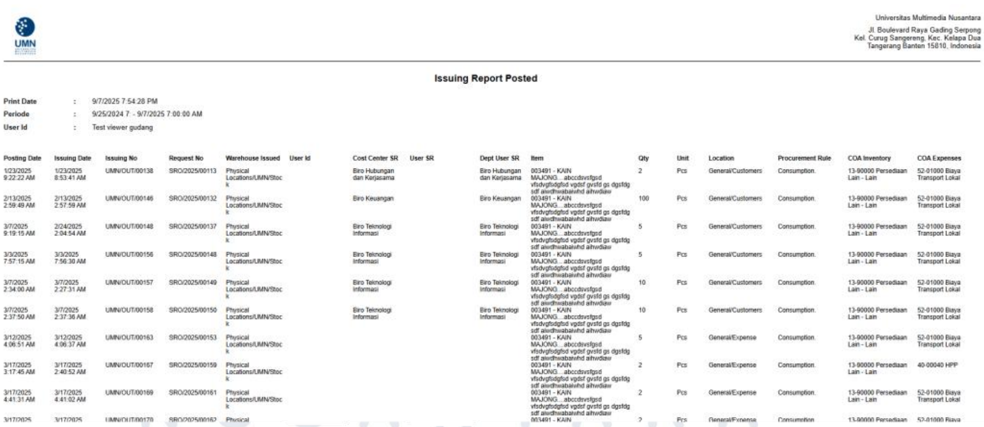
```
def _define_report_name(self): [redacted]

    is_show_cost = self.env.user.has_group("kg_stock.group_stock_viewer_show_price") or self.env.user.has_group("kg_stock.group_stock_logistics_spv")

    if self.type == "0":
        rpt = "/kg_stock_account/static/rpt/stock_request.mrt" if is_show_cost else "/kg_stock_account/static/rpt/stock_request_hide.mrt"
    elif self.type == "1":
        rpt = "/kg_stock_account/static/rpt/stock_request_return.mrt" if is_show_cost else "/kg_stock_account/static/rpt/stock_request_return_hide.mrt"
    elif self.type == "2":
        rpt = "/kg_stock_account/static/rpt/stock_request_category.mrt" if is_show_cost else "/kg_stock_account/static/rpt/stock_request_category_hide.mrt"

    return rpt
```

Pada laporan untuk *Viewer* biasa, informasi terkait nilai *cost* tidak ditampilkan. Laporan hanya berisi data umum, hal ini sesuai dengan tujuan awal, yaitu memberikan akses informasi yang cukup untuk kebutuhan pemantauan tanpa menyertakan data yang bersifat sensitif. Tampilan laporan untuk *Viewer* biasa dapat dilihat pada Gambar 3.40


UMN

Universitas Multimedia Nusantara
Jl. Boulevard Raya Gading Serpong
Kel. Curug Sangereng, Kec. Kelapa Dua
Tangerang Banten 15810, Indonesia

Issuing Report Posted

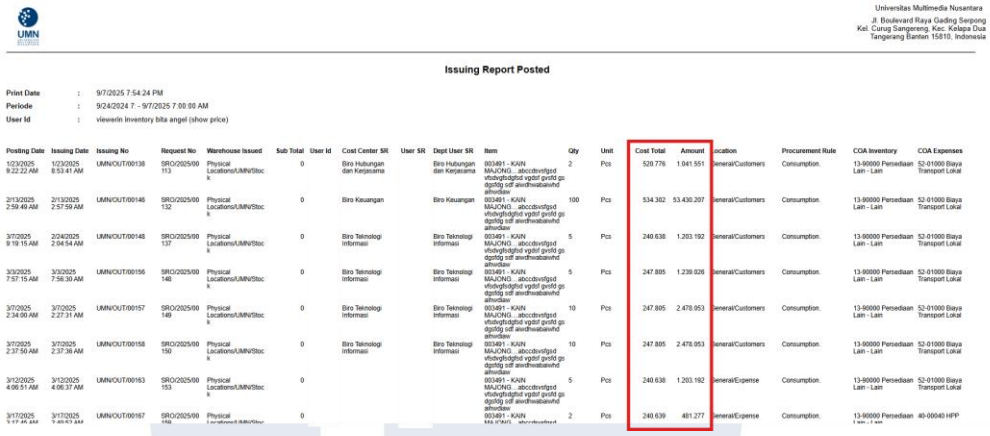
Print Date : 9/7/2025 7:54:28 PM
Periode : 9/25/2024 7: - 9/7/2025 7:00:00 AM
User Id : Test viewer gudang

Posting Date	Issuing Date	Issuing No	Request No	Warehouse Issued	User Id	Cost Center SR	User SR	Dept User SR	Item	Qty	Unit	Location	Procurement Rule	COA Inventory	COA Expenses
1/23/2025 9:22:22 AM	1/23/2025 8:53:41 AM	UMN/OUT/00138	SRO/2025/00113	Physical Locations/UMN/Stock		Biro Hubungan dan Kerjasama		Biro Hubungan dan Kerjasama	003491 - KAIN MAJONG....abccdsvsfgsd vfsdvgfsdgfsd vgdsf gvsfd gs dgsfdg sdf aiwdhwabaiwhd aihwdiaw	2	Pcs	General/Customers	Consumption.	13-90000 Persediaan Lain - Lain	52-01000 Biaya Transport Lokal
2/13/2025 2:59:49 AM	2/13/2025 2:57:59 AM	UMN/OUT/00146	SRO/2025/00132	Physical Locations/UMN/Stock		Biro Keuangan		Biro Keuangan	003491 - KAIN MAJONG....abccdsvsfgsd vfsdvgfsdgfsd vgdsf gvsfd gs dgsfdg sdf aiwdhwabaiwhd aihwdiaw	100	Pcs	General/Customers	Consumption.	13-90000 Persediaan Lain - Lain	52-01000 Biaya Transport Lokal
3/7/2025 9:19:15 AM	2/24/2025 2:04:54 AM	UMN/OUT/00148	SRO/2025/00137	Physical Locations/UMN/Stock		Biro Teknologi Informasi		Biro Teknologi Informasi	003491 - KAIN MAJONG....abccdsvsfgsd vfsdvgfsdgfsd vgdsf gvsfd gs dgsfdg sdf aiwdhwabaiwhd aihwdiaw	5	Pcs	General/Customers	Consumption.	13-90000 Persediaan Lain - Lain	52-01000 Biaya Transport Lokal
3/3/2025 7:57:15 AM	3/3/2025 7:56:30 AM	UMN/OUT/00156	SRO/2025/00148	Physical Locations/UMN/Stock		Biro Teknologi Informasi		Biro Teknologi Informasi	003491 - KAIN MAJONG....abccdsvsfgsd vfsdvgfsdgfsd vgdsf gvsfd gs dgsfdg sdf aiwdhwabaiwhd aihwdiaw	5	Pcs	General/Customers	Consumption.	13-90000 Persediaan Lain - Lain	52-01000 Biaya Transport Lokal
3/7/2025 2:34:00 AM	3/7/2025 2:27:31 AM	UMN/OUT/00157	SRO/2025/00149	Physical Locations/UMN/Stock		Biro Teknologi Informasi		Biro Teknologi Informasi	003491 - KAIN MAJONG....abccdsvsfgsd vfsdvgfsdgfsd vgdsf gvsfd gs dgsfdg sdf aiwdhwabaiwhd aihwdiaw	10	Pcs	General/Customers	Consumption.	13-90000 Persediaan Lain - Lain	52-01000 Biaya Transport Lokal
3/7/2025 2:37:50 AM	3/7/2025 2:37:36 AM	UMN/OUT/00158	SRO/2025/00150	Physical Locations/UMN/Stock		Biro Teknologi Informasi		Biro Teknologi Informasi	003491 - KAIN MAJONG....abccdsvsfgsd vfsdvgfsdgfsd vgdsf gvsfd gs dgsfdg sdf aiwdhwabaiwhd aihwdiaw	10	Pcs	General/Customers	Consumption.	13-90000 Persediaan Lain - Lain	52-01000 Biaya Transport Lokal
3/12/2025 4:06:51 AM	3/12/2025 4:06:37 AM	UMN/OUT/00163	SRO/2025/00153	Physical Locations/UMN/Stock					003491 - KAIN MAJONG....abccdsvsfgsd vfsdvgfsdgfsd vgdsf gvsfd gs dgsfdg sdf aiwdhwabaiwhd aihwdiaw	5	Pcs	General/Expense	Consumption.	13-90000 Persediaan Lain - Lain	52-01000 Biaya Transport Lokal
3/17/2025 3:17:45 AM	3/17/2025 2:40:52 AM	UMN/OUT/00167	SRO/2025/00159	Physical Locations/UMN/Stock					003491 - KAIN MAJONG....abccdsvsfgsd vfsdvgfsdgfsd vgdsf gvsfd gs dgsfdg sdf aiwdhwabaiwhd aihwdiaw	2	Pcs	General/Expense	Consumption.	13-90000 Persediaan Lain - Lain	40-00040 HPP
3/17/2025 4:41:31 AM	3/17/2025 4:41:02 AM	UMN/OUT/00169	SRO/2025/00161	Physical Locations/UMN/Stock					003491 - KAIN MAJONG....abccdsvsfgsd vfsdvgfsdgfsd vgdsf gvsfd gs dgsfdg sdf aiwdhwabaiwhd aihwdiaw	2	Pcs	General/Expense	Consumption.	13-90000 Persediaan Lain - Lain	52-01000 Biaya Transport Lokal
3/17/2025	3/17/2025	UMN/OUT/00170	SRO/2025/00162	Physical					003491 - KAIN	2	Pcs	General/Expense	Consumption	13-90000 Persediaan	52-01000 Biaya

Gambar 3. 40 tampilan Report untuk Group Viewer

Sementara itu, pada laporan untuk *Inventory (Show Price)*, informasi mengenai nilai *cost* beserta total jumlahnya ditampilkan secara lengkap. Laporan ini ditujukan untuk pengguna yang memang memiliki kebutuhan khusus dalam menganalisis data biaya, sehingga mereka dapat menggunakan

informasi tersebut untuk proses evaluasi maupun pengambilan keputusan yang lebih detail. Tampilan laporan untuk *Inventory (Show Price)* ditunjukkan pada Gambar 3.41



Gambar 3. 41 Tampilan Report untuk Group Inventory Show Price

Selain pada pengaturan hak akses untuk *Viewer*, penyesuaian juga dilakukan pada menu Inventory → Master Data → Product. Pada menu ini terdapat sejumlah *field* yang sebelumnya masih dapat dilakukan *create* maupun *edit*, padahal field-field tersebut sudah memiliki keterkaitan dengan modul lain sehingga tidak diperbolehkan untuk diubah secara sembarangan. Field yang dimaksud adalah Category (categ_id, product.template), APP ID (app_id, product.template), Customer Taxes (taxes_id, product.template), Vendor Taxes (supplier_taxes_id, product.template), Expense Account (property_account_expense_id, product.template), Asset Type (asset_category_id, product.template), serta Tag Category.

3.3.9 Memberikan penjagaan agar data KPI integration tidak dapat di edit

Pada versi sebelumnya, baik data yang bersumber dari *integration* maupun yang dibuat secara manual sama-sama dapat diedit secara langsung, termasuk target per bulannya. Kondisi tersebut menimbulkan potensi ketidaksesuaian data, khususnya pada KPI yang berasal dari sistem integrasi, karena nilai yang

sudah otomatis terhubung dari sumber data seharusnya tidak boleh diubah kembali. Oleh karena itu, dilakukan perbaikan pada sistem dengan memberikan pembatasan, di mana KPI yang bersumber dari *integration* tidak dapat diedit, sementara KPI manual tetap dapat diubah sesuai kebutuhan. Perubahan ini bertujuan untuk menjaga konsistensi, keakuratan, dan validitas data yang digunakan dalam proses pemantauan kinerja.

Untuk mengatasi permasalahan tersebut, dilakukan pengembangan dengan menambahkan logika pemrograman yang berfungsi membedakan antara KPI yang bersumber dari integrasi dan KPI yang dibuat secara manual. Logika ini bekerja dengan cara menandai apakah suatu KPI termasuk ke dalam kategori *integration* atau bukan. Jika KPI teridentifikasi berasal dari tabel *PMO.KPI.integration*, maka sistem secara otomatis mengenalinya sebagai data yang sudah terintegrasi.

Selanjutnya, pada bagian antarmuka aplikasi, indikator tersebut dimanfaatkan untuk mengatur hak akses terhadap nilai target. Dengan demikian, apabila sebuah KPI memiliki status *integration*, maka field target yang ditampilkan akan bersifat *readonly* sehingga tidak dapat diubah lagi oleh pengguna. Sebaliknya, untuk KPI yang bersifat manual, pengguna tetap memiliki fleksibilitas untuk melakukan pengeditan sesuai kebutuhan. Kode lengkap dari pengembangan logika ini dapat dilihat pada Gambar 3.42 dan Gambar 3.43 yang menampilkan implementasi field, fungsi pemeriksaan sumber data KPI, serta pengaturan *readonly* pada antarmuka pengguna.

```python
    is_integration = fields.Boolean(
        string="Is Integration",
        compute="_compute_is_integration",
        store=False
    )

    @api.depends('kpi_line_id.kpi_id')  bita
    def _compute_is_integration(self):
        for rec in self:
            if rec.kpi_line_id and rec.kpi_line_id.kpi_id:
                exists = self.env['pmo.kpi.integration'].search_count([
                    ('kpi_id', '=', rec.kpi_line_id.kpi_id.id)
                ])
                rec.is_integration = bool(exists)
            else:
                rec.is_integration = False
```

Gambar 3. 42 Kode Inisialisasi is_integration Beserta Fungsinya

```xml
<list create="false" delete="false" editable="bottom">
    <field name="month" readonly="1"/>
    <field name="is_integration" invisible="1"/>
    <field name="amount" readonly="is_skip or no_crud or is_integration"/>
    <field name="is_skip" readonly="no_crud"/>
    <field name="synced" readonly="1"/>
</list>
```

Gambar 3. 43 Kode Pengaturan Readonly pada Antarmuka Pengguna

3.3.10 Membuat Filter domain field return di stock picking type

Pada bagian menu operation type, pengguna meminta perbaikan pada domain untuk field return_picking_type_id. Pengguna menginginkan ada filter pada domain tersebut dengan ketentuan tertentu . dalam penyelesaian, filter untuk kolom return_picking_type_id dibuat secara dinamis berdasarkan nilai code dari *Operation Type* itu sendiri. Logika ini memastikan bahwa tipe 'incoming' akan dipasangkan dengan 'outgoing', dan sebaliknya. Untuk tipe operasi 'internal', tipe baliknya juga harus 'internal', namun dengan validasi untuk mencegah pemilihan data yang sama (dirinya sendiri). Di luar ketiga kondisi tersebut, fungsionalitas filter tetap sama seperti sebelumnya tanpa ada perubahan. Penyesuaian ini ditunjukkan pada kode di gambar 3.44

```python
@api.onchange("code")  bita
def _onchange_code_set_return_domain(self):
    domain = []
    if self.code == "incoming":
        domain = [("code", "=", "outgoing")]
    elif self.code == "outgoing":
        domain = [("code", "=", "incoming")]
    elif self.code == "internal":
        current_id = self._origin.id or 0
        domain = [
            ("code", "=", "internal"),
            ("id", "!=", current_id),
        ]
    return {"domain": {"return_picking_type_id": domain}}
```

Gambar 3. 44 Kode Filter Domain Return

3.3.11 Menambahkan Field Reason pada Purchase Request Line

Pada modul Purchase Request pengguna meminta penambahan field baru bernama reason pada bagian header maupun line, dengan ketentuan ketika dilakukan proses copy pada Purchase Request Line, nilai reason dari line asal tidak boleh sama dengan nilai reason pada line hasil copy. Dalam penyelesaiannya, field *reason* di tambahkan ke kode XML agar muncul di *header* dan juga di *line*. Selain itu, logika pemrograman juga diubah sehingga ketika *Purchase Request* dicopy, seluruh data pada *line* akan diduplikasi, kecuali beberapa field tertentu, salah satunya *reason* yang akan diatur menjadi kosong/*False*. Kode penyelesaian permintaan ini dapat dilihat pada gambar 3.45 Dan 3.46

```
<field name="purchaser_ids" invisible="1"/>
<field name="is_po_header_draft_sent" invisible="1"/>
<field name="purchaser_id" force_save="1"
       attrs="{'readonly': [('is_po_header_draft_sent', '=', False),
                            '|', ('state','!=','approved', 'and', 'user_can_assign_purchaser','=',False)],
<field name="reason" readonly="1" string="Reason Done"/>
</field>
```

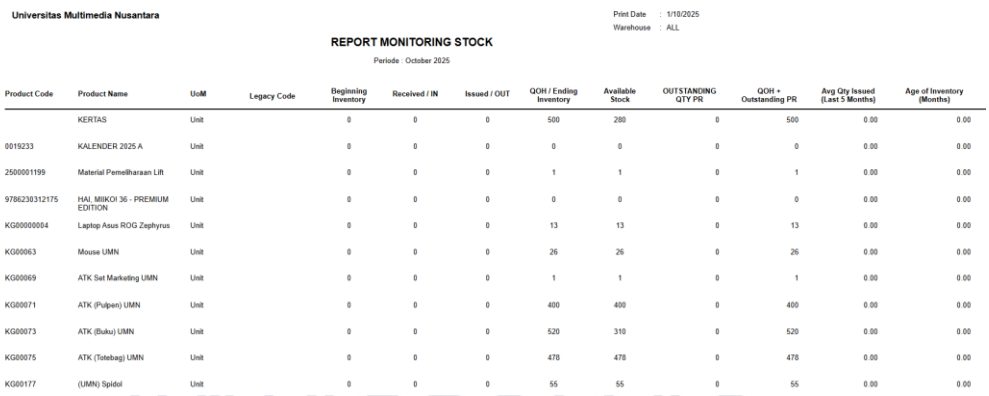
Gambar 3. 45 Kode Penambahan Field Reason pada Header dan Line

```
default.update({
    "requested_by": user_id,
    "reason": False,
    'name': self.env['ir.sequence'].with_context(
        ir_sequence_date=fields.Date.context_today(self),
        force_company=self.env.user.company_id.id
    ).next_by_code('purchase.request')
})
res =  super(KGPurchaseRequest, self).copy(default)
for line in res.line_ids:
    line.reason = False
return res
```

Gambar 3. 46 Kode Pencegahan Duplikasi Reason saat Copy Purchase Request

3.3.12 Menambahkan field draft pr qty di report stock request erp

permintaan lain adalah menambahkah kolom baru pada report stock request erp. Sebelumnya bentuk report stock request belum memiliki kolom draft pr qty seperti yang ditunjukkan pada gambar 3.47



Universitas Multimedia Nusantara

Print Date : 1/10/2025
Warehouse : ALL

REPORT MONITORING STOCK

Periode : October 2025

Product Code	Product Name	UoM	Legacy Code	Beginning Inventory	Received / IN	Issued / OUT	QOH / Ending Inventory	Available Stock	OUTSTANDING QTY PR	QOH + Outstanding PR	Avg Qty Issued (Last 5 Months)	Age of Inventory (Months)
	KERTAS	Unit		0	0	0	500	280	0	500	0.00	0.00
0019233	KALENDER 2025 A	Unit		0	0	0	0	0	0	0	0.00	0.00
2500001199	Material Pemeliharaan Lift	Unit		0	0	0	1	1	0	1	0.00	0.00
9786230312175	HAI, MIIKO! 36 - PREMIUM EDITION	Unit		0	0	0	0	0	0	0	0.00	0.00
KG00000004	Laptop Asus ROG Zephyrus	Unit		0	0	0	13	13	0	13	0.00	0.00
KG00063	Mouse UMN	Unit		0	0	0	26	26	0	26	0.00	0.00
KG00069	ATK Set Marketing UMN	Unit		0	0	0	1	1	0	1	0.00	0.00
KG00071	ATK (Pulpen) UMN	Unit		0	0	0	400	400	0	400	0.00	0.00
KG00073	ATK (Buku) UMN	Unit		0	0	0	520	310	0	520	0.00	0.00
KG00075	ATK (Totebag) UMN	Unit		0	0	0	478	478	0	478	0.00	0.00
KG00177	(UMN) Spidol	Unit		0	0	0	55	55	0	55	0.00	0.00

Gambar 3. 47 Report Stock Request sebelum Perubahan

Pada pengerjaan backlog ini dilakukan penambahan kolom baru pada menu *Inventory* di laporan *Stock Inventory* pada ERP menggunakan CTE (*Common table exspression*) dalam querynya. Bentuk querynya dapat dilihat pada gambar 3.48 Kolom yang dibuat dinamakan Draft PR Qty, yang berfungsi untuk menampilkan hasil penjumlahan dari nilai *qty deviation* dengan menerapkan beberapa ketentuan. Ketentuan yang digunakan antara

lain adalah data yang dihitung hanya berasal dari *purchase request line* yang memiliki status *draft*, serta memastikan bahwa jenis permintaan tersebut dikategorikan sebagai *stock* dan bukan *expense*. Data yang dihasilkan kemudian ditampilkan dalam bentuk agregasi dengan metode *group by* berdasarkan *perusahaan* dan produk.

```
z_total_draft_pr AS (
    SELECT prl.product_id,  pr.company_id, SUM(prl.qty_deviation) AS draft_pr_qty
    FROM purchase_request_line prl
    JOIN purchase_request pr ON pr.id = prl.request_id
    left join product_product pp on prl.product_id = pp.id
    left join product_template t on t.id = pp.product_tmpl_id
    LEFT JOIN stock_picking_type spt ON spt.id = prl.picking_type_id
    LEFT JOIN stock_location sl ON sl.id = spt.default_location_dest_id
    WHERE prl.line_state = 'draft' AND pr.company_id = {self.company_id.id}
    {where_product}
    AND prl.date_start::date <= '{curr_end_date}'
    AND sl.usage = 'internal'
    AND spt.is_direct_expense = false
    GROUP BY prl.product_id, pr.company_id
),
```

Gambar 3. 48 Kode Query Kolom Draft PR Qty

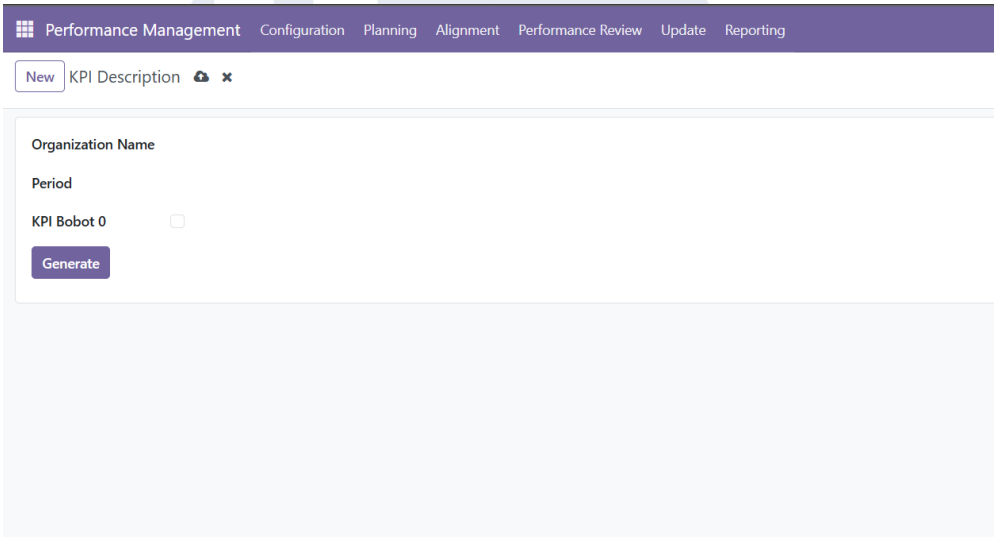
3.3.13 Membuat menu KPI Description

Pengembangan *PMO* versi 4 ini hadir sebagai upaya perbaikan sekaligus modernisasi, dengan tujuan menghadirkan tampilan antarmuka yang lebih segar, responsif, serta mendukung kinerja yang lebih cepat. Dengan integrasi langsung ke dalam *Odoo 18*, sistem tidak hanya lebih stabil, tetapi juga memiliki fleksibilitas lebih tinggi untuk dikembangkan dan diadaptasikan dengan kebutuhan pengguna di masa mendatang. Dalam perubahan ini dikembangkan beberapa menu yaitu KPI description dan juga Si description.

Pengembangan backlog pada menu KPI Description difokuskan untuk menghadirkan laporan deskriptif yang lebih rinci mengenai setiap *Key Performance Indicator*. KPI atau *Key Performance Indicator* merupakan indikator utama yang digunakan untuk mengukur kinerja perusahaan dalam mencapai tujuan tertentu. Menu ini dirancang untuk menampilkan laporan berupa deskripsi detail dari KPI yang dipilih. Dengan adanya menu ini,

pengguna dapat memahami konteks serta latar belakang dari setiap indikator kinerja yang digunakan dalam proses evaluasi.

Selain itu, dapat dilihat pada gambar 3.49 menu KPI Description juga dilengkapi dengan fitur pemilihan perusahaan dan tahun data KPI. Sehingga pengguna dapat menyesuaikan laporan sesuai kebutuhan analisis, baik berdasarkan periode waktu tertentu maupun entitas perusahaan yang diinginkan.

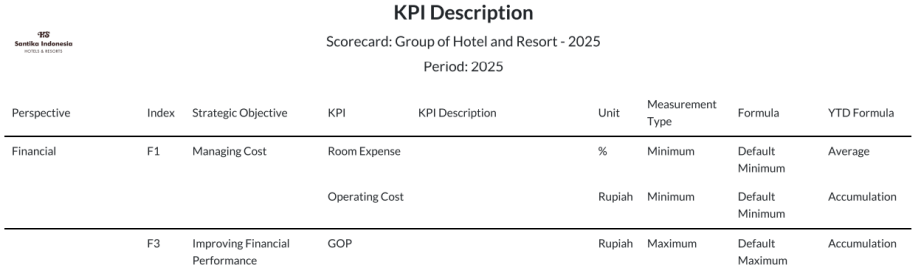


Gambar 3. 49 Menu KPI Description

Menu kpi description

Isi dari report KPI Description menampilkan informasi yang lebih detail terkait setiap *Key Performance Indicator* yang dipilih, tampilan report kpi description dapat dilihat pada gambar 3.50 . Informasi yang ditampilkan report tersebut adalah perspektif, index, serta strategic objective yang menjadi dasar penentuan indikator tersebut. Selain itu, laporan juga menampilkan informasi mengenai nama KPI serta deskripsi KPI yang menjelaskan makna dan tujuan indikator tersebut. Informasi tambahan seperti unit atau mata uang yang digunakan, measurement type, serta formula perhitungan turut ditampilkan untuk memperjelas metode evaluasi yang digunakan. Pada laporan KPI Description juga menyediakan YTD Formula (Year-to-Date)

yang berfungsi untuk menampilkan metode perhitungan akumulatif sepanjang periode berjalan.



KPI Description

Scorecard: Group of Hotel and Resort - 2025

Period: 2025

Perspective	Index	Strategic Objective	KPI	KPI Description	Unit	Measurement Type	Formula	YTD Formula
Financial	F1	Managing Cost	Room Expense		%	Minimum	Default Minimum	Average
			Operating Cost		Rupiah	Minimum	Default Minimum	Accumulation
	F3	Improving Financial Performance	GOP		Rupiah	Maximum	Default Maximum	Accumulation

Gambar 3. 50 Tampilan Report KPI Description

Tahap awal dalam pengembangan menu ini adalah pembuatan menu item baru agar pengguna dapat mengakses menu laporan. Seperti yang ditunjukkan pada Gambar 3.51, sebuah *menu item* baru dengan label "KPI descriptions Report" ditambahkan ke dalam sistem. *Menu item* ini dirancang untuk berfungsi sebagai titik masuk utama menuju halaman wizard laporan deskripsi KPI.

Selanjutnya, untuk mendefinisikan perilaku yang terjadi ketika menu tersebut diakses, sebuah *window action* dikonfigurasikan. Berdasarkan Gambar 3.51, *action* ini terhubung dengan model wizard.reporting.kpi.description yang menjadi dasar logika dari fitur laporan ini. Tampilan antarmuka yang muncul diatur untuk secara spesifik menggunakan *view form* dengan ID eksternal kg_pmo.view_pmo_kpi_description_form. Pengaturan penting dalam *action* ini adalah penggunaan target="inline". Atribut ini memastikan bahwa saat pengguna mengklik menu, tampilan form wizard akan terbuka langsung di area konten utama halaman, bukan sebagai jendela *pop-up* yang terpisah.

```
<record id="action_wizard_reporting_kpi_description" model="ir.actions.act_window">
    <field name="name">KPI description</field>
    <field name="type">ir.actions.act_window</field>
    <field name="res_model">wizard.reporting.kpi.description</field>
    <field name="view_mode">form</field>
    <field name="view_id" eval="ref('kg_pmo.view_pmo_kpi_description_form')"/>
    <field name="target">inline</field>
</record>

<menuitem
    id="menu_reporting_kpi_description"
    name="KPI descriptions Report"
    action="action_wizard_reporting_kpi_description"
    parent="menu_pmo_reporting"
/>
```

Gambar 3. 51 Kode Menuitem dan Action KPI Description

Setelah *window action* berhasil dibuat, langkah selanjutnya adalah membuat sebuah *view* dengan tipe form, seperti yang dapat dilihat pada referensi Gambar 3.52. *View* ini terhubung dengan model wizard.reporting.kpi.description dan berfungsi sebagai halaman interaktif bagi pengguna untuk memasukkan parameter sebelum laporan dibuat. Judul form diatur sebagai "PMO KPI Description" untuk memberikan konteks yang jelas kepada pengguna.

Di dalam struktur form. Terdapat tiga komponen utama yang disediakan untuk input pengguna. Pertama, kolom company_id dan period_id yang memungkinkan pengguna untuk memilih perusahaan dan periode waktu yang spesifik untuk laporan. Pada kedua kolom ini, diterapkan opsi {'no_create': True, 'no_create_edit': True} yang bertujuan untuk membatasi pengguna agar hanya dapat memilih dari data yang sudah ada, bukan membuat data baru. Hal ini penting untuk menjaga konsistensi dan validitas data. Komponen ketiga adalah is_kpi_bobot.

kemudian elemen terakhir dari form ini, sebuah tombol dengan label "Generate". Tombol ini dikonfigurasi dengan type="object" yang akan memicu eksekusi metode action_generate yang ada pada model wizard saat diklik. Fungsi dari tombol ini adalah sebagai pemicu utama untuk memulai

proses pengolahan data dan pembuatan laporan berdasarkan parameter yang telah diisi oleh pengguna pada kolom-kolom sebelumnya.

```xml
<record model="ir.ui.view" id="view_pmo_kpi_description_form">
    <field name="name">pmo.kpi.description.wizard.form</field>
    <field name="model">wizard.reporting.kpi.description</field>
    <field name="arch" type="xml">
        <form string="PMO KPI Description">
            <sheet>
                <group>
                    <field name="company_id" options="{'no_create': True, 'no_create_edit': True}"/>
                    <field name="period_id" options="{'no_create': True, 'no_create_edit': True}"/>
                    <field name="is_kpi_bobot"/>
                    <button name="action_generate" type="object" class="btn btn-primary" string="Generate"/>
                </group>
            </sheet>
        </form>
    </field>
</record>
```

Gambar 3. 52 Kode View KPI Description

Kemudian untuk logika bisnis dan data input dari pengguna, dibuat sebuah model baru bernama wizard.reporting.kpi.description seperti yang ditunjuk pada Gambar 3.53 Model ini merupakan model *TransientModel*, yang artinya data yang tersimpan di dalamnya hanya bersifat sementara selama sesi interaksi pengguna. Model ini berfungsi sebagai "otak" yang menghubungkan data dari antarmuka pengguna dengan eksekusi metode action_generate saat tombol "Generate" ditekan.

```python
class WizardReportingKPIDescription(models.TransientModel):  bita
    _name = 'wizard.reporting.kpi.description'
    _description = 'Wizard Reporting KPI Description'

    name = fields.Char(compute='_compute_name')
    company_id = fields.Many2one( comodel_name: 'res.company', string='Organization Name', required=True)
    period_id = fields.Many2one( comodel_name: 'period', string='Period', required=True)
    is_kpi_bobot = fields.Boolean(string="KPI Bobot 0")
```

Gambar 3. 53 Kode Model Utama KPI Description

Kemudian untuk bentuk laporannya dihasilkan menggunakan Qweb yang merupakan mesin template bawaan dari odoo sendiri, dan untuk sintaksnya menggunakan XML, seperti potongan kode yang ditunjukkan pada gambar 3.54

Gambar 3. 54 Potongan Kode Laporan KPI Description

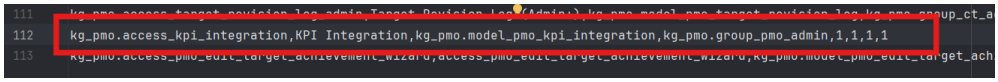
Dengan demikian, melalui tahapan-tahapan pengembangan yang telah dijabarkan mulai dari Integrasi antara *menu item*, *window action*, *view* antarmuka, model *transient*, dan templat laporan Qweb telah menghasilkan sebuah fitur baru. Hasil dari keseluruhan proses ini adalah menu "KPI descriptions Report".

3.3.14 Menambah hak akses pada model pmo.kpi.integration

Karena model *PMO.KPI.integration* masih tergolong baru, pada awalnya belum terdapat pengaturan hak akses yang jelas bagi pengguna. Kondisi ini berpotensi menimbulkan permasalahan, karena tanpa adanya hak akses, seluruh pengguna tidak dapat mengakses model tersebut. Untuk mengatasi hal tersebut, dilakukan penyesuaian dengan memberikan hak akses khusus pada model ini.

Dari 4 user yang tersedia untuk *performance management* Hak akses hanya diberikan pada user *CT Admin*. Dengan cara ini, hanya pihak yang memiliki otorisasi penuh yang dapat melakukan pengelolaan terhadap data *integration* pada KPI. Sementara itu, pengguna dengan peran lainnya tidak diberikan izin untuk mengubah data tersebut, sehingga konsistensi dan validitas data tetap terjaga. Kode lengkap dari pengaturan hak akses ini ditunjukkan pada

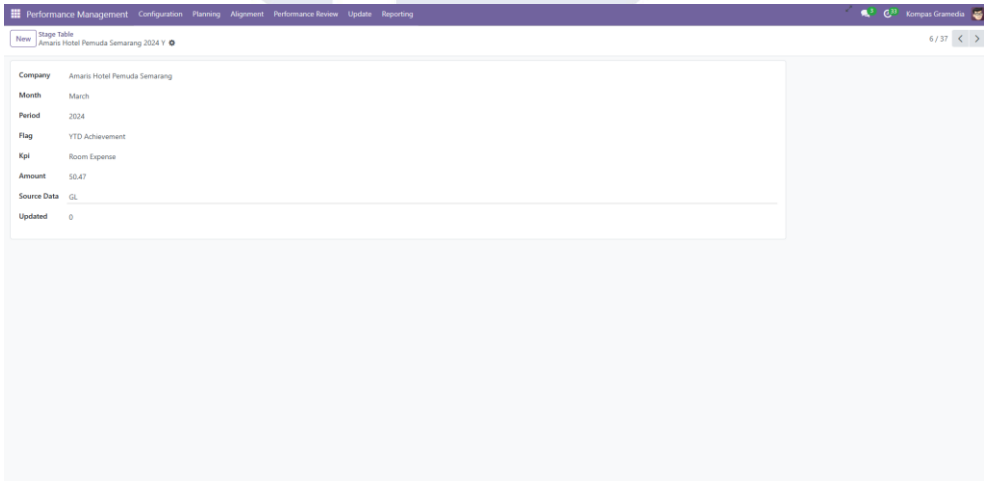
Gambar 3.55, yang memperlihatkan implementasi pembatasan akses hanya kepada pengguna dengan peran *CT Admin*.



Gambar 3. 55 Kode Penetapan Hak Akses Kepada CT Admin

3.3.15 Menambahkan flag baru di update target/achivement pada PMO

Pada *sub menu Update* dalam modul *Performance Management*, terdapat sebuah fitur penting yang disebut *Stage Table* seperti yang ditampilkan pada gambar 3.56 . Fitur ini berfungsi sebagai tahap awal penyimpanan data sebelum masuk ke dalam *Table Achievement*. Dengan adanya *Stage Table*, sistem dapat melakukan validasi maupun pemrosesan awal terhadap data capaian kinerja sehingga lebih terstruktur dan terkontrol sebelum hasil akhirnya disimpan sebagai data resmi pada tabel utama.



Gambar 3. 56 Sub Menu Stage Table pada Performance Management

Sebelumnya, struktur *Stage Table* hanya memiliki empat jenis *flag* sebagai penanda data, yaitu T (*Target*), A (*Achievement*), Y (*YTD Achievement*), dan Z (*YTD Target*). Namun, seiring dengan kebutuhan pengelolaan data yang lebih rinci, ditambahkan beberapa jenis *flag* baru. Penambahan ini mencakup B untuk *Amount Original Achievement*, C untuk *Divisor Achievement*, D untuk *Amount Original Target*, serta E untuk *Divisor Target*. Kode

implementasi dari pengembangan *flag* baru pada *Stage Table* ini ditunjukkan pada Gambar 3.37.

```
flag = fields.Selection([('T', 'Target'),
                         ('A', 'Achievement'),
                         ('Y', 'YTD Achievement'),
                         ('Z', 'YTD Target'),
                         ('B', 'Amount Original Achievement'),
                         ('C', 'Divisor Achievement'),
                         ('D', 'Amount Original Target'),
                         ('E', 'Divisor Target')])
```

Gambar 3. 57 Kode Pengembangan Flag Baru pada Stage Table

3.3.16 Kendala yang Ditemukan

Dalam pelaksanaan kegiatan magang, terdapat sejumlah kendala serta tantangan yang dihadapi, baik yang bersifat teknis maupun non-teknis. Berikut ini beberapa kendala yang ditemukan selama masa magang:

a. Ketidaksesuaian Data pada Sistem PMO

Salah satu kendala adalah karena sistem *Performance Management Online (PMO)* masih tergolong baru, data yang digunakan belum sepenuhnya sesuai dengan standar yang seharusnya. Ketidaksesuaian ini menimbulkan berbagai kendala teknis ketika sistem dijalankan di *local environment*. Salah satu dampak yang paling sering ditemui adalah munculnya error pada saat proses pengujian. Error tersebut tidak hanya menghambat jalannya sistem secara umum, tetapi juga membuat beberapa menu tidak dapat diakses atau bahkan gagal ditampilkan sama sekali.

Kondisi ini terjadi karena data yang belum konsisten menyebabkan sistem kesulitan membaca maupun memproses informasi yang seharusnya ditampilkan. Akibatnya, banyak menu yang seharusnya berfungsi normal justru tidak dapat dijalankan sesuai kebutuhan. Permasalahan ini memperlambat proses pengembangan, karena pengembang harus melakukan pengecekan manual terhadap data dan melakukan penyesuaian tambahan agar sistem bisa dijalankan tanpa hambatan

b. Kurangnya pengetahuan tentang PMO

Karena sistem masih berada pada tahap awal pengembangan, pemahaman terhadap Performance Management Online (PMO) menjadi salah satu kendala yang cukup signifikan. Sebelum terlibat dalam proses pengembangan, interaksi maupun penggunaan PMO belum pernah dilakukan, sehingga pengetahuan mengenai konsep dasar, fungsi utama, dan alur kerja sistem ini masih sangat terbatas. Ketidaktahuan awal ini membuat proses adaptasi memerlukan waktu lebih lama, karena setiap fitur maupun istilah yang ada pada PMO harus dipelajari dari awal.

c. Permintaan *user* yang tidak selalu realistis

Dalam proses pengembangan sistem, salah satu kendala yang cukup sering ditemui adalah adanya permintaan dari pengguna yang tidak selalu realistis untuk diimplementasikan. Hal ini biasanya muncul karena perbedaan sudut pandang antara pengguna dan pengembang. Sebagian pengguna, khususnya yang sudah berusia lanjut, cenderung lebih nyaman dengan tampilan antarmuka sistem yang sederhana dan terkesan "jadul", karena dianggap lebih mudah dipahami. Mereka lebih mengutamakan aspek keterbiasaan dibandingkan mengikuti standar desain modern yang telah disediakan oleh *Odoo*.

Permasalahan ini menjadi tantangan bagi pengembang karena mengubah antarmuka *Odoo* agar menyesuaikan dengan preferensi pengguna tersebut bukanlah hal yang sederhana. Justru, upaya tersebut berpotensi mengurangi fungsionalitas maupun fleksibilitas sistem. Bahkan, jika tampilan *Odoo* terlalu dipaksakan untuk mengikuti permintaan yang tidak sesuai dengan desain bawaan, maka nilai kegunaan sistem dapat menurun. *Odoo* yang seharusnya menawarkan antarmuka modern dan terintegrasi justru menjadi kurang optimal karena lebih menekankan pada pemenuhan preferensi individual yang tidak sejalan dengan tujuan utama pengembangan.

d. Keterbatasan akses terhadap database

Akses langsung ke *database* hanya dimiliki oleh senior developer, sehingga tidak dapat melakukan pengecekan atau pengujian data secara menyeluruh.

3.3.17 Solusi atas Kendala yang Ditemukan

Untuk mengatasi berbagai kendala yang telah diuraikan sebelumnya, Berikut adalah beberapa solusi yang diterapkan untuk mengatasi kendala-kendala tersebut

a. Menghapus *field* yang menyebabkan error

Untuk mengatasi permasalahan ketidaksesuaian data pada sistem PMO, diambil langkah sederhana namun efektif, yaitu dengan menghapus *field* yang menyebabkan error baik pada kode maupun pada *database* lokal. Dengan cara ini, sistem dapat berjalan kembali secara normal pada lingkungan pengembangan (*local*), sehingga fitur lain dapat diuji tanpa terganggu oleh data yang tidak valid. Implementasi dari solusi ini dapat dilihat pada *Gambar 3.58* yang menampilkan potongan kode setelah dilakukan penghapusan *field* yang bermasalah.

```
<div class="col-12 col-lg-6 o_setting_box" id="recaptcha">
        <div class="o_setting_left_pane">-->
            <field name="show_chat_bot"/>-->
        </div>-->
```

Gambar 3. 58 Penghapusan Field yang Bermasalah

b. Bertanya dan mengikuti *sharing session* dengan senior dan rekan kerja

Pada tahap awal pengembangan, kurangnya pemahaman mengenai sistem *Performance Management Online (PMO)* menjadi salah satu kendala yang cukup signifikan. Hal ini disebabkan karena sebelumnya belum pernah terlibat langsung dalam pengelolaan maupun pengembangan PMO, sehingga pemahaman terkait konsep, alur kerja, serta logika bisnis yang terdapat di

dalam sistem masih sangat terbatas. Kondisi tersebut menyebabkan proses adaptasi menjadi lebih lama dan berpotensi memperlambat pengembangan.

Untuk mengatasi kendala tersebut, dilakukan beberapa langkah, di antaranya dengan mengikuti sesi berbagi (*sharing session*) bersama senior maupun rekan kerja yang telah berpengalaman dalam menangani PMO. Melalui kegiatan ini, pengetahuan mengenai fungsi, alur kerja, serta tujuan dari setiap komponen dalam sistem dapat lebih dipahami. Selain itu, diskusi aktif dengan senior dan kolega juga menjadi sarana penting dalam mempercepat pemahaman, terutama ketika menghadapi masalah teknis maupun logis dalam sistem.

c. Berdiskusi dengan analis dan developer senior

Salah satu kendala yang ditemukan dalam proses pengembangan adalah adanya permintaan dari sebagian pengguna yang tidak selalu realistis untuk diimplementasikan. Jika dipaksakan, permintaan tersebut justru berpotensi mengurangi fungsionalitas sistem dan menghambat tujuan awal.

Solusi dari permasalahan ini adalah melakukan diskusi secara langsung dengan analis sistem serta *developer* senior. Melalui diskusi tersebut, dapat dicari jalan tengah atau alternatif terbaik yang tidak hanya tetap mempertahankan kemudahan bagi pengguna, tetapi juga sesuai dengan standar teknis pengembangan sistem

d. Menjalin Komunikasi Intensif dan Kolaboratif dengan Analis

Keterbatasan akses terhadap *database* menjadi salah satu kendala dalam proses pengembangan. Hanya senior tertentu yang memiliki hak akses penuh, sehingga pengembang lain tidak dapat secara langsung melakukan perubahan atau pengujian data di lingkungan *database* utama.

Solusi untuk mengatasi hal ini adalah dengan meminta penarikan *database* baru kepada pihak yang memiliki akses. Dengan adanya salinan *database*

tersebut, dapat dilakukan pengujian secara mandiri tanpa harus selalu bergantung pada senior.