# BAB 3
# PELAKSANAAN KERJA MAGANG

Selama kegiatan magang, saya menjalankan peran ganda sebagai Developer dan Project Manager untuk proyek migrasi ERP Odoo versi 15 ke versi 18 di PT Klik Semangat Indonesia. Peran ini membawa tanggung jawab teknis dan manajerial yang signifikan, mengingat sistem ERP menjadi tulang punggung operasional perusahaan yang mengelola berbagai proses bisnis seperti penjualan, pembelian, logistik, akuntansi, hingga sumber daya manusia.

## 3.1 Kedudukan dan Koordinasi

Selama kegiatan magang, saya sebagai Developer memiliki tanggung jawab untuk:

1. Mengembangkan software Odoo dengan kustomisasi sesuai kebutuhan pengguna, termasuk mengadaptasi modul yang sudah ada dan membuat modul baru.
2. Mengelola database dan server, serta memastikan sistem berjalan dengan optimal dan aman.
3. Menganalisis semua proses bisnis pada setiap divisi agar lebih efisien dan dapat berjalan lebih lancar melalui integrasi sistem ERP.

Sebagai Project Manager untuk proyek migrasi ke Odoo 18, saya memiliki tanggung jawab untuk:

1. Memastikan kelancaran proses migrasi dari persiapan hingga implementasi, termasuk pemetaan kebutuhan bisnis, penyesuaian kode, serta pengujian sistem.
2. Berkoordinasi dengan tim lain untuk memastikan keberhasilan migrasi, menyusun timeline proyek, dan mengelola sumber daya yang dibutuhkan untuk mendukung proyek.
3. Menyusun dan memastikan dokumentasi lengkap terkait migrasi sistem, serta memberikan dukungan kepada tim terkait untuk transisi yang mulus.

4. Mengelola risiko dan memecahkan masalah teknis yang mungkin timbul selama proses migrasi untuk memastikan proses berjalan sesuai dengan target dan anggaran yang telah ditetapkan.

Selain itu, pelatihan pengguna turut diikuti serta dukungan teknis terkait penggunaan sistem Odoo yang baru diberikan.

## 3.2 Tugas yang Dilakukan

- **Pengembangan Sistem:** Mengembangkan dan menyesuaikan sistem ERP Odoo agar sesuai dengan kebutuhan operasional perusahaan, termasuk membuat dan memodifikasi berbagai modul.
- **Peningkatan Efisiensi Operasional:** Membuat fitur dan sistem pendukung untuk mempercepat serta menyederhanakan proses kerja di berbagai divisi seperti pengiriman, pemesanan stok, dan penjualan.
- **Pengelolaan dan Pemeliharaan Infrastruktur:** Mengelola server dan database perusahaan serta memastikan sistem berjalan dengan stabil, aman, dan dapat diandalkan.
- **Migrasi Sistem ERP:** Merencanakan dan mengeksekusi proses migrasi dari Odoo versi 15 ke versi 18, termasuk persiapan teknis, pengujian, implementasi, dan pendampingan setelah go-live.
- **Penyesuaian Sistem Keuangan dan Pelaporan:** Menyesuaikan sistem pelaporan dan proses keuangan agar mendukung kebutuhan lintas perusahaan dan kebijakan internal.
- **Koordinasi dan Manajemen Proyek:** Berkoordinasi dengan berbagai tim/divisi dalam perusahaan untuk memastikan setiap proses migrasi dan pengembangan berjalan sesuai target dan kebutuhan.
- **Pelatihan dan Dukungan Pengguna:** Memberikan pelatihan, dokumentasi, dan dukungan teknis kepada pengguna sistem agar dapat beradaptasi dengan sistem baru secara efektif.

Enterprise Resource Planning (ERP) adalah sistem informasi yang digunakan untuk mengintegrasikan dan mengelola proses bisnis utama dalam sebuah organisasi secara real-time. ERP tidak hanya menghubungkan proses lintas

divisi, tetapi juga berfungsi sebagai pusat data operasional dan keuangan yang mendukung pengambilan keputusan berbasis data terkini.

Solusi open-source seperti Odoo fleksibel, modular, dan tepat untuk UKM/ritel, sebagaimana didukung oleh studi akademik. [3, 4, 5]. Di ranah akademik, integrasi ERP juga mendapatkan perhatian sebagai bagian dari pembelajaran berbasis industri; faktor seperti dukungan manajemen, pelatihan, dan infrastruktur sangat berpengaruh. [6, 7, 8].

Kajian implementasi ERP menunjukkan bahwa tantangan umum termasuk misalignment proses, kurangnya pelatihan pengguna, dan manajemen perubahan yang tidak optimal—sehingga dibutuhkan strategi pengetahuan dan pelatihan yang sesuai. [9]. Selain itu, kendala adaptabilitas dan interoperabilitas ERP bagi UKM/startup juga menjadi topik penting dalam pengembangan platform bisnis proses masa depan. [10].

**Odoo** sendiri merupakan salah satu platform ERP open-source yang menyediakan ekosistem aplikasi terintegrasi seperti Point of Sale (POS), Inventory, Purchase, Sales, dan Accounting. Selain itu, Odoo juga mendukung pengembangan modul kustom seperti Approval Workflow dan sistem Komisi Reseller. Keunggulan utama Odoo terletak pada kemampuannya untuk dikustomisasi sesuai proses bisnis spesifik serta didukung komunitas pengembang yang aktif. Dengan platform ini, perusahaan dapat menjalankan proses lintas divisi dalam satu sistem yang efisien dan scalable.

Berdasarkan pemahaman terhadap ekosistem Odoo tersebut, pelaksanaan kegiatan magang diarahkan untuk mendalami penerapan sistem ERP secara langsung pada konteks operasional bisnis ritel di PT Klik Semangat Indonesia.

## 3.3 Uraian Pelaksanaan Magang

Waktu dan Pelaksanaan kerja magang diuraikan seperti pada Tabel 3.1 dan untuk pekerjaan lengkap nya ada pada Tabel 3.2

Tabel 3.1. Tabel waktu pelaksanan magang

| No | Kegiatan | Waktu Pelaksanaan |
|---|---|---|
| 1 | Melakukan pengembangan dan kustomisasi pada Odoo 15 | 01 Februari 2025 – 27 Februari 2025 |
| 2 | Melakukan persiapan untuk migrasi batch 1 Odoo versi 15 ke Odoo 18 | 28 Februari 2025 – 04 April 2025 |
| 3 | Melakukan migrasi batch 1 Odoo 15 ke Odoo 18 | 05 April 2025 – 08 April 2025 |
| 4 | Memberi dukungan dan melakukan penyesuaian error / bugs pasca migrasi batch 1 di Odoo 18 | 05 April 2025 – 08 April 2025 |
| 5 | Melakukan evaluasi dan analisa masalah yang membuat gagal migrasi batch 1 | 09 April 2025 – 24 April 2025 |
| 6 | Melakukan migrasi batch 2 Odoo 15 ke Odoo 18 | 25 April 2025 – 31 Mei 2025 |
| 7 | Melakukan pengembangan dan pemeliharaan Odoo 18 | 25 April 2025 – 31 Mei 2025 |

Tabel 3.2. Uraian pekerjaan tiap minggu selama magang

| Minggu Ke- | Pekerjaan yang Dilakukan |
|---|---|
| 1 | Mengembangkan sistem batch transfer dengan field driver dan jenis kendaraan. Menambahkan perhitungan insentif berdasarkan berat barang dan nilai insentif per produk. Menambahkan field attachment wajib untuk foto muatan sebelum validasi batch. |
| 2 | Menambahkan tombol cetak seluruh delivery order dalam batch. Mengustomisasi picking list agar tampil produk dengan stok 0. Melakukan uji dan validasi batch delivery. |
| 3 | Mengembangkan sistem komisi reseller pada POS. Menambahkan pemilihan salesperson di POS. Komisi dihitung otomatis dan disimpan. Membuat laporan dalam pivot dan tree, bisa difilter. |
| 4 | Menambahkan fitur extra supplier pada replenishment. Mengembangkan sistem replenishment untuk toko. Menambahkan wizard filter by warehouse untuk menghindari loading berat. |
| 5 | Identifikasi semua modul (default, custom, third-party). Pemetaan data yang akan dimigrasi. Pembuatan dokumen kebutuhan. Setup environment staging Odoo 18. |
| 6 | Konfigurasi awal dan instalasi Odoo 18. Analisis fitur Odoo 15 vs 18. Review kustomisasi lama. Menentukan modul utama. Mulai porting custom module. |

Lanjut ke halaman berikutnya

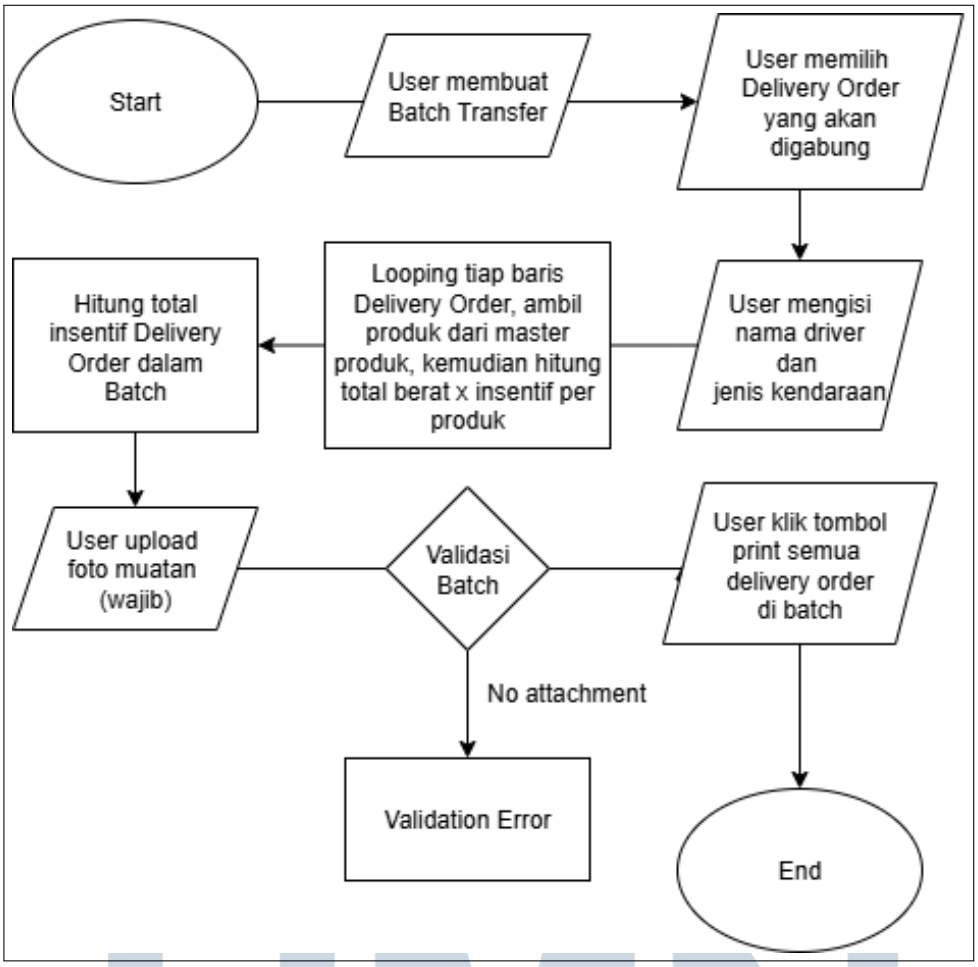
Tabel 3.2 Uraian pekerjaan tiap minggu selama magang (lanjutan)

| Minggu Ke- | Pekerjaan yang Dilakukan |
|---|---|
| 7 | Migrasi master data. Adaptasi dan testing custom module. Uji proses bisnis dasar (SO, PO, invoice). |
| 8 | End-to-end testing, simulasi transaksi, training user per divisi, dan koreksi dari feedback user. |
| 9 | Uji coba sistem ganda. POS pakai Odoo 18, sisanya masih Odoo 15. Ditemukan bug dan ketidaksesuaian. Evaluasi dan rollback ke Odoo 15. |
| 10 | Revisi modul error. Uji migrasi ulang di environment kecil. Persiapan cut-off dan rencana go-live baru. |
| 11 | Migrasi ulang dan validasi data. End-to-end testing dan simulasi. Training user per divisi. |
| 12 | Go live Odoo 18. Monitoring intensif. Migrasi bertahap. Transfer delivery dan receipt tertunda. |
| 13 | Sinkronisasi manual transaksi historis. Migrasi tagihan vendor dan invoice pelanggan dari Odoo 15. |
| 14 | Koreksi hak akses user. Perbaikan kendala closing POS akibat akun belum lengkap. |
| 15 | Mengembangkan fitur auto bill dan invoice untuk otomatisasi pembuatan tagihan vendor dan pelanggan berdasarkan transaksi pembelian dan penjualan yang telah tervalidasi. |
| 16 | Menambahkan fitur auto return bill dan invoice untuk mempercepat proses pengembalian barang, di mana sistem secara otomatis membuat return dan dokumen keuangan terkait saat dilakukan return picking. |

### 3.3.1 Pengembangan Delivery System berbasis Batch Transfer

Pada flowchart Gambar 3.1 menggambarkan alur kerja sistem Batch Transfer Delivery yang dimulai dari pembuatan batch baru, pemilihan beberapa delivery order, serta pengisian data driver dan jenis kendaraan. Sistem kemudian menghitung total berat dan insentif berdasarkan setiap produk dalam batch. Setelah itu, pengguna diwajibkan mengunggah foto muatan sebagai syarat validasi. Jika foto belum diunggah, sistem akan menampilkan pesan kesalahan dan proses berhenti; jika sudah, batch dapat divalidasi dan seluruh delivery order dalam batch dapat dicetak sekaligus. Alur ini memastikan kontrol pengiriman berjalan

sistematis, akurat, dan terdokumentasi.




Gambar 3.1. Flowchart cara kerja sistem delivery berbasis Batch Transfer

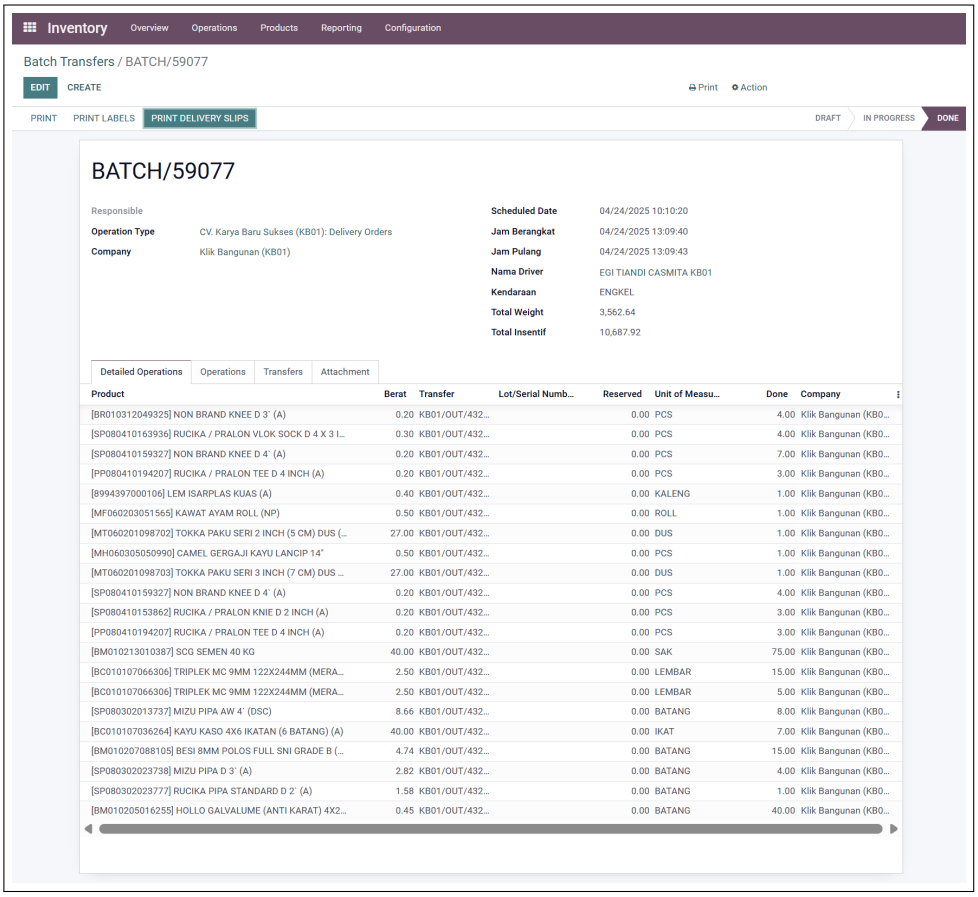
```python
from odoo import fields, models, api

class StockMoveLine(models.Model):
    _inherit = 'stock.move.line'

    x_custom_weight = fields.Float('Weight', compute='_compute_weight_from_product')
    x_total_weight = fields.Float('Total Weight', compute='_compute_custom_weight')

    x_custom_insentif = fields.Float('Insentif', compute='_compute_insentif_from_product')
    x_total_insentif = fields.Float('Total Insentif', compute='_compute_total_insentif')

```

```
    @api.depends('product_id')
    def _compute_weight_from_product(self):
        for line in self:
            line.x_custom_weight = line.product_id.weight

    @api.depends('x_custom_weight','qty_done')
    def _compute_custom_weight(self):
        for line in self:
            line.x_total_weight = line.x_custom_weight * line.
    qty_done

    @api.depends('product_id')
    def _compute_insentif_from_product(self):
        for line in self:
            line.x_custom_insentif = line.product_id.
    product_tmpl_id.product_insentif

    @api.depends('x_custom_insentif', 'qty_done','x_custom_weight'
    )
    def _compute_total_insentif(self):
        for line in self:
            line.x_total_insentif = line.x_custom_insentif * line.
    qty_done * line.x_custom_weight

from odoo import fields, models, api
from odoo.exceptions import UserError

class StockPickingBatch(models.Model):
    _inherit = 'stock.picking.batch'

    x_drivers_name = fields.Many2one('ksi.drivers', string='Nama
    Driver', required=True)
    x_cars = fields.Selection([
        ('Carry', 'CARRY'),
        ('Engkel','ENGKEL'),
        ('Traga', 'TRAGA'),
        ('Motor','MOTOR'),
        ('Truck','TRUCK'),
        ('-','-'),
    ], string='Kendaraan', required=True)

    x_sum_total_insentif = fields.Float(string="Total Insentif",
    compute="_compute_sum_total_insentif", store=True)
```

```
49      x_sum_total_weight = fields.Float(string="Total Weight",
    compute="_compute_sum_total_weight", store=True)
50
51      @api.depends('picking_ids.move_line_ids.x_total_insentif')
52      def _compute_sum_total_insentif(self):
53          for batch in self:
54              total = sum(batch.picking_ids.mapped('move_line_ids').
    mapped('x_total_insentif'))
55              batch.x_sum_total_insentif = total
56
57      @api.depends('picking_ids.move_line_ids.x_total_weight')
58      def _compute_sum_total_weight(self):
59          for batch in self:
60              total = sum(batch.picking_ids.mapped('move_line_ids').
    mapped('x_total_weight'))
61              batch.x_sum_total_weight = total
```

Kode 3.1: Listing implementasi delivery system

Potongan kode 3.1 merupakan *custom model inheritance* dari `stock.picking.batch` yang menambahkan beberapa field dan logika bisnis untuk mendukung sistem pengiriman berdasarkan driver dan insentif. Field `x_drivers_name` menyimpan data driver yang dipilih dari model `ksi.drivers`, sedangkan `x_cars` adalah pilihan jenis kendaraan yang digunakan. Field `x_sum_total_weight` dan `x_sum_total_insentif` merupakan field komputasi yang otomatis menghitung total berat barang dan total insentif berdasarkan nilai insentif per produk dari delivery order dalam batch. Logika ini memanfaatkan relasi dari `picking_ids` dan `move_line_ids`. Selain itu, sistem juga menambahkan validasi wajib berupa attachment foto sebelum batch dapat divalidasi untuk memastikan proses pengiriman terdokumentasi dengan baik.

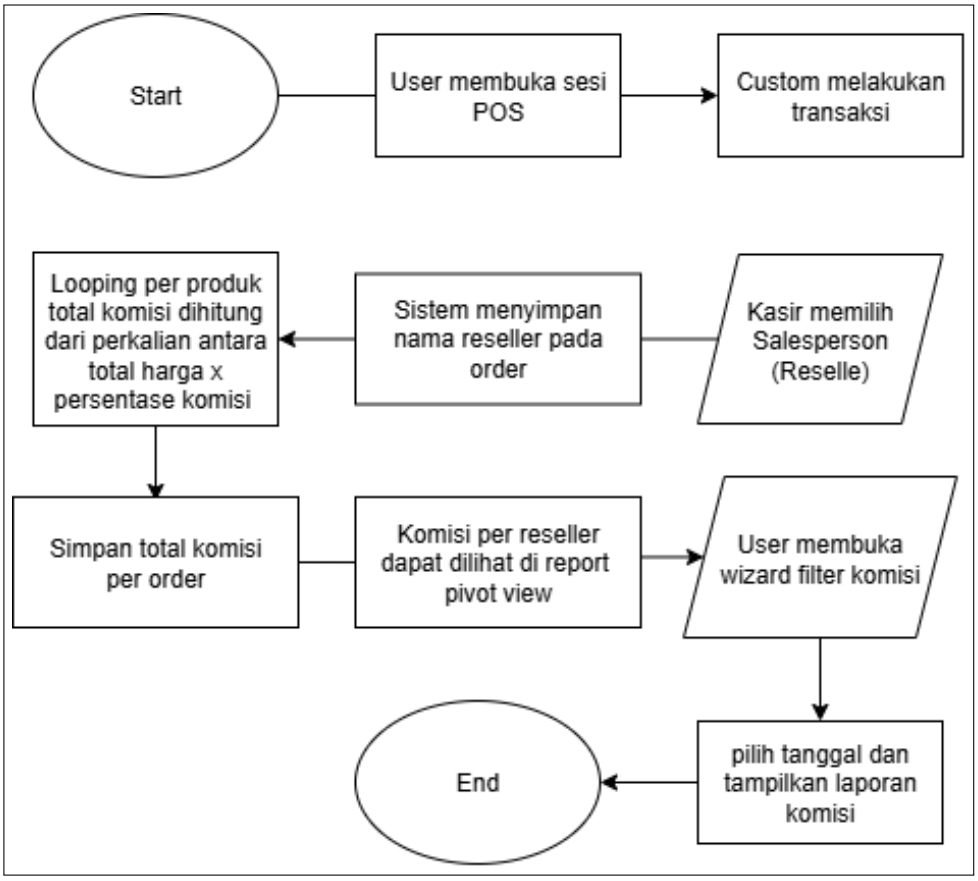



Gambar 3.2. Tampilan implementasi sistem delivery berbasis batch transfer

Tampilan Gambar 3.2 merupakan hasil dari pengembangan sistem batch delivery yang kamu buat. Ini mendukung pengelolaan pengiriman lebih efisien, memudahkan tracking insentif driver, serta memastikan dokumen pendukung (foto muatan) lengkap. Fitur ini sangat membantu logistik toko bangunan dengan pengiriman dalam volume besar dan variasi produk tinggi.

### 3.3.2 Pengembangan sistem komisi reseller pada POS

Pada flowchart Gambar 3.3 menggambarkan bagaimana komisi dihitung otomatis setiap kali terjadi transaksi POS dengan reseller yang dipilih. Setiap item dihitung berdasarkan persentase komisi yang ditentukan di produk, lalu dicatat per order. Data ini bisa ditarik sebagai laporan untuk pembayaran komisi reseller secara berkala.




Gambar 3.3. Flowchart cara kerja sistem Reseller Commission pada POS

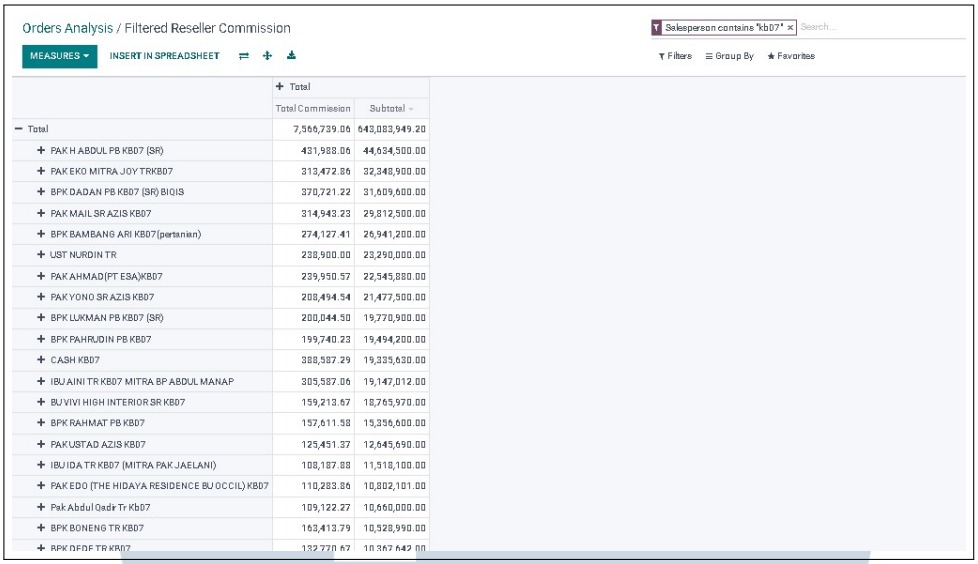
```python
from odoo import models, fields, api

class PosOrderLine(models.Model):
    _inherit = 'pos.order.line'

    product_commission = fields.Float(
        related='product_id.product_commission',
        readonly=True,
        store=True,
        string="Commission (%)"
    )

    partner_id = fields.Many2one(
        related='order_id.partner_id',
        readonly=True,
        store=True,
        string="Customer"
    )

```

```
20      total_commission = fields.Float(
21          string='Total Commission',
22          readonly=True,
23          compute='_compute_total_commission',
24          group_operator='sum',
25          store=True
26      )
27
28      @api.depends('price_subtotal', 'qty')
29      def _compute_unit_price(self):
30          """Menghitung harga per unit"""
31          for record in self:
32              record.unit_price = (record.price_subtotal or 0.0) / (
    record.qty or 1.0)
33
34      @api.depends('product_commission', 'price_subtotal')
35      def _compute_total_commission(self):
36          """Compute total commission based on subtotal and
    commission percentage"""
37          for record in self:
38              record.total_commission = record.product_commission *
    record.price_subtotal
39
40      @api.depends('product_commission', 'unit_price')
41      def _compute_commission(self):
42          """Compute commission per unit"""
43          for record in self:
44              record.commission = record.product_commission * record
    .unit_price
45
46
47  class ProductTemplate(models.Model):
48      _inherit = 'product.template'
49
50      product_commission = fields.Float(string="Product Commission
    (%)")
51
52
53  from odoo import models, fields, api
54
55  class ResellerCommissionWizard(models.TransientModel):
56      _name = 'reseller.commission.wizard'
57      _description = 'Filter Wizard for Reseller Commission'
```

```
58
59      start_date = fields.Date(string="Start Date", required=True)
60      end_date = fields.Date(string="End Date", required=True)
61      company_id = fields.Many2one(
62          'res.company', string="Company", required=True,
63          default=lambda self: self.env.company.id
64      )
65
66      def action_apply_filter(self):
67          """Apply filter based on selected date range and company
    """
68          return {
69              'type': 'ir.actions.act_window',
70              'name': 'Filtered Reseller Commission',
71              'view_mode': 'pivot,tree',
72              'res_model': 'pos.order.line',
73              'domain': [('create_date', '>=', self.start_date),
74                         ('create_date', '<=', self.end_date),
75                         ('company_id', '=', self.company_id.id)],
76              'target': 'current',
77          }
```

Kode 3.2: Listing implementasi Odoo commission dan wizard

Potongan kode 3.2 merupakan pengembangan sistem komisi reseller pada Odoo POS. Model `pos.order.line` di-extend untuk menyimpan persentase komisi produk melalui field `product_commission` (related ke `product.template`), serta menghitung nilai `total_commission` dari subtotal harga produk dikalikan komisi. Field `partner_id` ditambahkan untuk referensi customer. Pada model `product.template`, field `product_commission` ditambahkan untuk mengisi nilai komisi per produk. Terakhir, `reseller.commission.wizard` adalah transient model yang digunakan sebagai wizard untuk memfilter data berdasarkan tanggal dan perusahaan, dan menampilkan hasilnya dalam tampilan pivot dan tree untuk keperluan pelaporan.

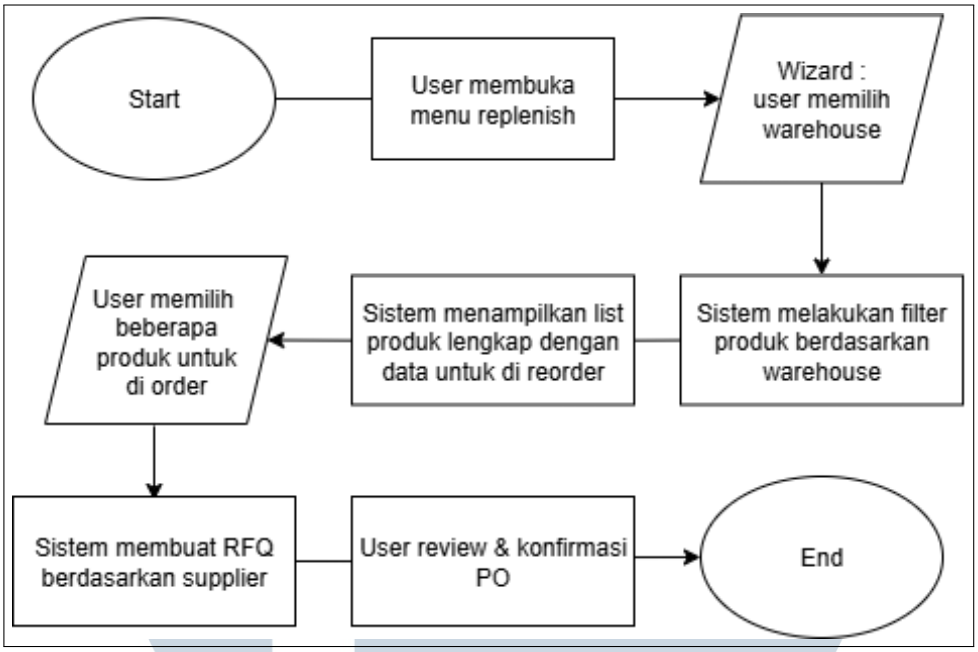



Gambar 3.4. Tampilan implementasi sistem reseller commission di POS

Tampilan ini menunjukkan hasil laporan komisi reseller POS dalam bentuk pivot table yang dihasilkan dari modul custom POS Reseller Commission. Laporan ini menyajikan total nilai penjualan (subtotal) dan total komisi yang diperoleh oleh masing-masing reseller (salesperson). Setiap baris merepresentasikan reseller tertentu, lengkap dengan subtotal transaksi dan perhitungan total komisi berdasarkan persentase komisi yang ditentukan per produk. Laporan ini sangat membantu manajemen dalam memantau program reseller dan menghitung komisi yang harus dibayarkan secara akurat dan efisien.

### 3.3.3 Pengembangan sistem Replenishment

Flowchart Gambar 3.5 menunjukkan bagaimana modul ini dirancang untuk memudahkan tim toko dalam memantau kebutuhan barang dan melakukan permintaan pembelian berdasarkan gudang masing-masing. Dengan adanya filter warehouse, proses replenishment menjadi lebih efisien, cepat, dan tidak membebani server.



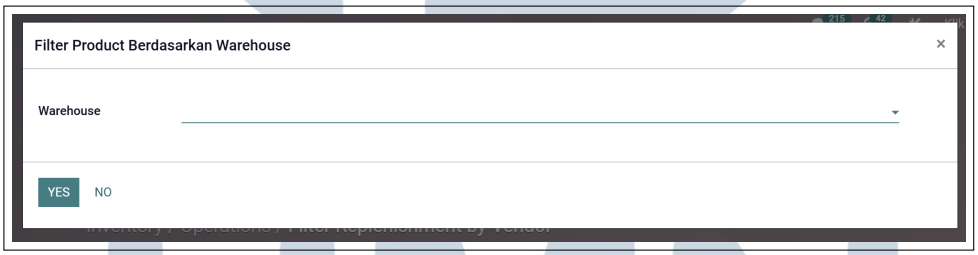

Gambar 3.5. Flowchart cara kerja sistem Replenishment Wizard

```python
from odoo import models, fields, api, _

class ReplenishmentFilterWizardWarehouse(models.TransientModel):
    _name = 'replenishment.filter.wizard.warehouse'
    _description = 'Wizard untuk Filter Product Berdasarkan Warehouse'

    warehouse_id = fields.Many2one('stock.warehouse', string="Warehouse", required=True)

    def action_apply_filter_warehouse(self):
        """
        Filter replenishment records berdasarkan Warehouse yang dipilih
        """
        self.ensure_one()
        return {
            'type': 'ir.actions.act_window',
            'name': 'Replenishment',
            'view_mode': 'tree',
            'res_model': 'stock.warehouse.orderpoint',
            'domain': [('warehouse_id', '=', self.warehouse_id.id)],
            'target': 'current',
```

```
21          }
```

Kode 3.3: Listing implementasi Odoo Replenishment Filter Wizard

Potongan kode 3.3 merupakan pengembangan pada sistem Replenishment di Odoo dengan menambahkan wizard bernama `replenishment.filter.wizard.warehouse`. Wizard ini memungkinkan pengguna untuk memfilter data kebutuhan restock berdasarkan gudang tertentu (`warehouse_id`). Setelah gudang dipilih dan tombol diklik, sistem akan menampilkan daftar produk yang perlu diisi ulang (`stock.warehouse.orderpoint`) dalam tampilan tree view yang sesuai dengan warehouse yang dipilih. Wizard ini diakses melalui menu *Inventory ¿ Warehouse Management*. Selain itu, dilakukan penyesuaian pada tampilan `stock.warehouse.orderpoint` dengan memberikan hak akses khusus hanya kepada user dengan grup `stock.group_stock_manager` agar dapat melihat atau mengedit field `supplier_id`, sebagai bagian dari pengelolaan extra supplier untuk setiap produk.




Gambar 3.6. Tampilan implementasi sistem wizard pada Replenishment

Gambar 3.6di atas menunjukkan tampilan wizard dengan judul *Filter Product Berdasarkan Warehouse*, yang merupakan bagian dari modul `replenishment.filter.wizard.warehouse`. Wizard ini memungkinkan pengguna untuk memilih salah satu gudang (`warehouse`) melalui menu dropdown yang tersedia. Setelah memilih gudang, pengguna dapat menekan tombol **YES** untuk menerapkan filter, sehingga sistem hanya akan menampilkan data produk yang perlu diisi ulang (*replenish*) sesuai dengan gudang yang dipilih. Tombol **NO** berfungsi untuk membatalkan dan menutup wizard tanpa melakukan tindakan. Wizard ini dirancang untuk membantu menghindari beban loading berat saat menampilkan seluruh data replenishment, serta mendukung proses permintaan stok yang lebih terfokus dan efisien berdasarkan lokasi gudang.

### 3.3.4 Proyek Migrasi Sistem Odoo 15 ke Odoo 18

Proyek migrasi sistem ERP Odoo dari versi 15 ke versi 18 merupakan langkah strategis perusahaan dalam meningkatkan efektivitas operasional dan memastikan keberlanjutan sistem informasi yang mendukung seluruh proses bisnis. Migrasi ini tidak hanya bertujuan memperbarui teknologi, tetapi juga menyelaraskan kustomisasi lama dengan fitur-fitur terbaru yang ditawarkan oleh Odoo 18. Proses migrasi mencakup serangkaian tahapan yang sistematis, mulai dari identifikasi kebutuhan dan persiapan lingkungan kerja, pengembangan ulang modul kustom, pengujian menyeluruh, hingga implementasi final di lingkungan produksi. Berikut penjelasan tahapan yang dilakukan secara terstruktur dalam proyek migrasi ini.

#### A Pengembangan dan Adaptasi Sistem

Proyek migrasi sistem ERP dari Odoo 15 ke Odoo 18 diawali dengan proses identifikasi seluruh modul yang digunakan dalam sistem lama, baik modul default bawaan Odoo, modul kustom internal perusahaan, maupun modul pihak ketiga. Selain itu, dilakukan pemetaan data penting yang akan dimigrasikan serta penyusunan dokumen kebutuhan fungsional sebagai acuan pengembangan. Setelah kebutuhan diinventarisasi, dilakukan instalasi dan konfigurasi awal Odoo 18 pada environment *staging*, termasuk penyesuaian konfigurasi dasar dan analisis perbandingan fitur antara Odoo versi 15 dan 18.

Selanjutnya dilakukan review atas kustomisasi lama dan dipilih modul-modul utama yang akan dipertahankan atau diperbarui. Proses porting modul kustom dimulai bersamaan dengan migrasi data master seperti produk, pelanggan, dan vendor. Setelah modul berhasil diadaptasi, dilakukan pengujian awal terhadap proses bisnis dasar, termasuk penjualan (Sales Order), pembelian (Purchase Order), dan penagihan (Invoice).

#### B Uji Coba dan Simulasi

Tahap uji coba dilakukan secara menyeluruh untuk memastikan bahwa seluruh proses bisnis dapat berjalan dengan baik pada sistem Odoo 18. Pengujian dilakukan secara bertahap mulai dari transaksi harian hingga alur akuntansi dan pelaporan.

Pengujian dimulai dari proses penjualan di POS, di mana kasir melakukan transaksi langsung dengan pelanggan. Sistem diuji untuk memastikan transaksi POS menghasilkan order, pembayaran, dan jurnal akuntansi secara otomatis. Selanjutnya, dilakukan pengujian terhadap proses pengelolaan stok untuk memastikan bahwa penerimaan produk dari gudang pusat tercatat secara real-time dan mempengaruhi kuantitas stok secara akurat di lokasi toko.

Setelah itu, pengujian dilanjutkan ke proses penjualan dan pembelian konvensional melalui modul Sales Order (SO) dan Purchase Order (PO). Pada tahap ini, skenario diuji baik untuk transaksi normal maupun antar-perusahaan (intercompany transaction), di mana satu entitas melakukan penjualan yang secara otomatis menghasilkan purchase order di entitas lain.

Seluruh transaksi penjualan dan pembelian kemudian diverifikasi untuk memastikan bahwa dokumen penagihan seperti invoice dan vendor bill terbentuk dengan benar dan otomatis masuk ke sistem akuntansi. Validasi dilakukan dengan memastikan bahwa setiap dokumen tersebut menghasilkan jurnal sesuai dengan ketentuan akuntansi perusahaan.

Pengujian berlanjut ke proses pembayaran, pengiriman barang, penerimaan barang, hingga pelaporan keuangan akhir seperti neraca, laporan laba rugi, dan laporan arus kas. Pengujian juga mencakup rekonsiliasi otomatis, alokasi pembayaran parsial, serta konsistensi antar modul.

Setiap hasil pengujian dicatat dan dibandingkan dengan alur bisnis aktual di Odoo 15, guna memastikan bahwa sistem baru tidak hanya setara namun juga memberikan peningkatan efisiensi dan kecepatan dalam proses bisnis. Skenario uji juga mencakup pengujian hak akses antar divisi, tampilan mobile, hingga integrasi antar modul untuk kebutuhan audit dan pelaporan manajemen.

## C Go-Live dan Evaluasi Hasil

Setelah validasi selesai, dilakukan migrasi ulang penuh ke Odoo 18, disertai pengujian end-to-end, simulasi sistem, dan pelatihan ulang user untuk setiap divisi.

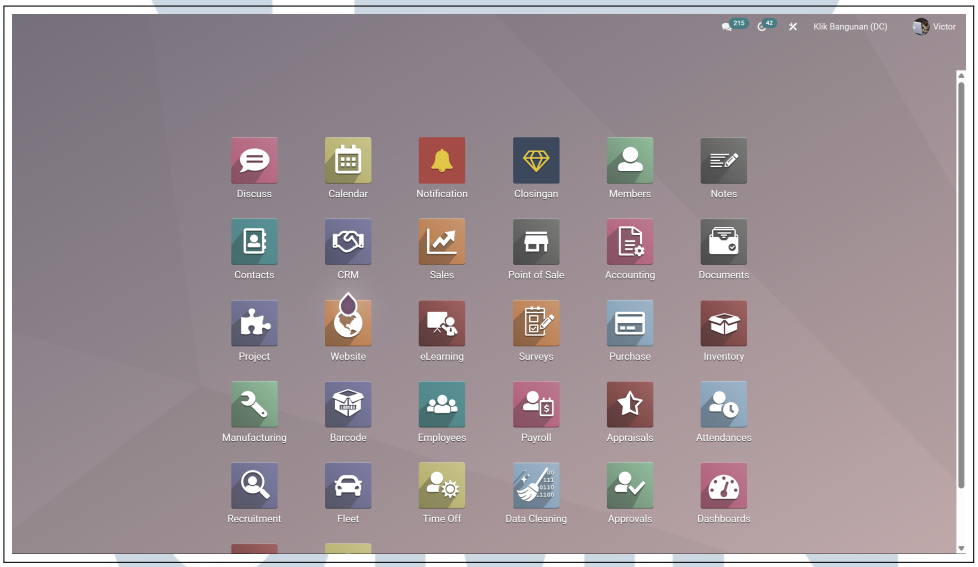
Tahap akhir adalah proses *go-live* Odoo 18 secara resmi, yang dilaksanakan secara bertahap untuk meminimalkan risiko. Proses ini didampingi dengan monitoring intensif dan pengelolaan manual untuk proses-proses tertentu seperti pengiriman (delivery) dan penerimaan barang (receipt) yang sempat tertunda selama transisi.

Sebagai bagian dari dokumentasi migrasi, berikut dilampirkan perbandingan

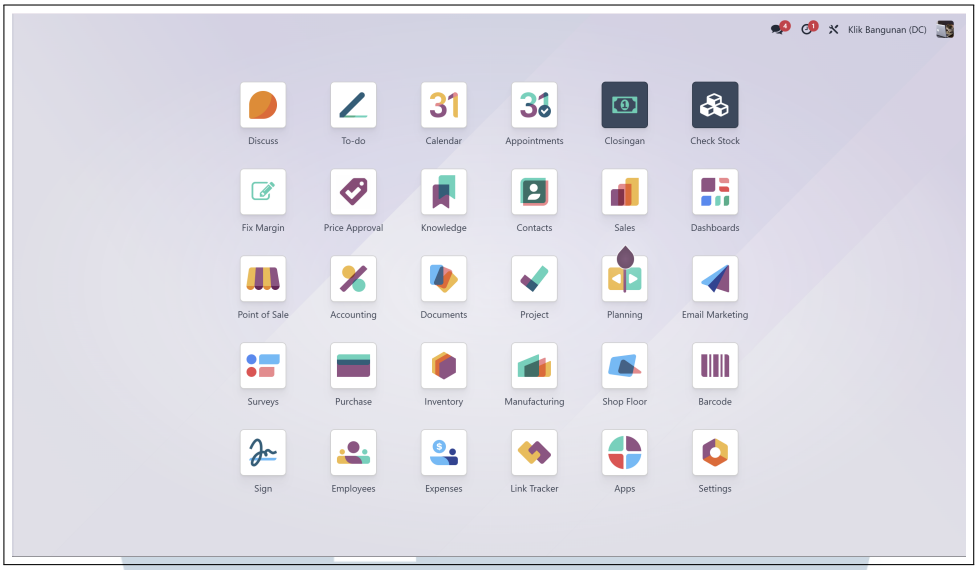
tampilan menu awal antara Odoo versi 15 Gambar 3.7 dan Odoo versi 18 Gambar 3.8. Tampilan antarmuka pengguna (user interface) Odoo 18 hadir dengan desain yang lebih modern dan bersih dibandingkan Odoo 15. Pada Odoo 18, elemen visual seperti ikon menu, ukuran font, dan struktur modul ditampilkan lebih responsif dan minimalis, sehingga memudahkan navigasi pengguna.

Perubahan ini juga memperlihatkan peningkatan pada sisi performa dan pengalaman pengguna, di mana waktu pemuatan halaman lebih cepat dan dukungan tampilan mobile lebih optimal. Selain itu, sistem pencarian dan akses cepat ke menu pada Odoo 18 lebih intuitif, sangat membantu dalam efisiensi kerja sehari-hari, terutama bagi pengguna dari berbagai divisi.

Perbandingan visual ini memberikan gambaran jelas atas peningkatan kualitas UI/UX sebagai salah satu nilai tambah dari proses migrasi ke versi Odoo terbaru.



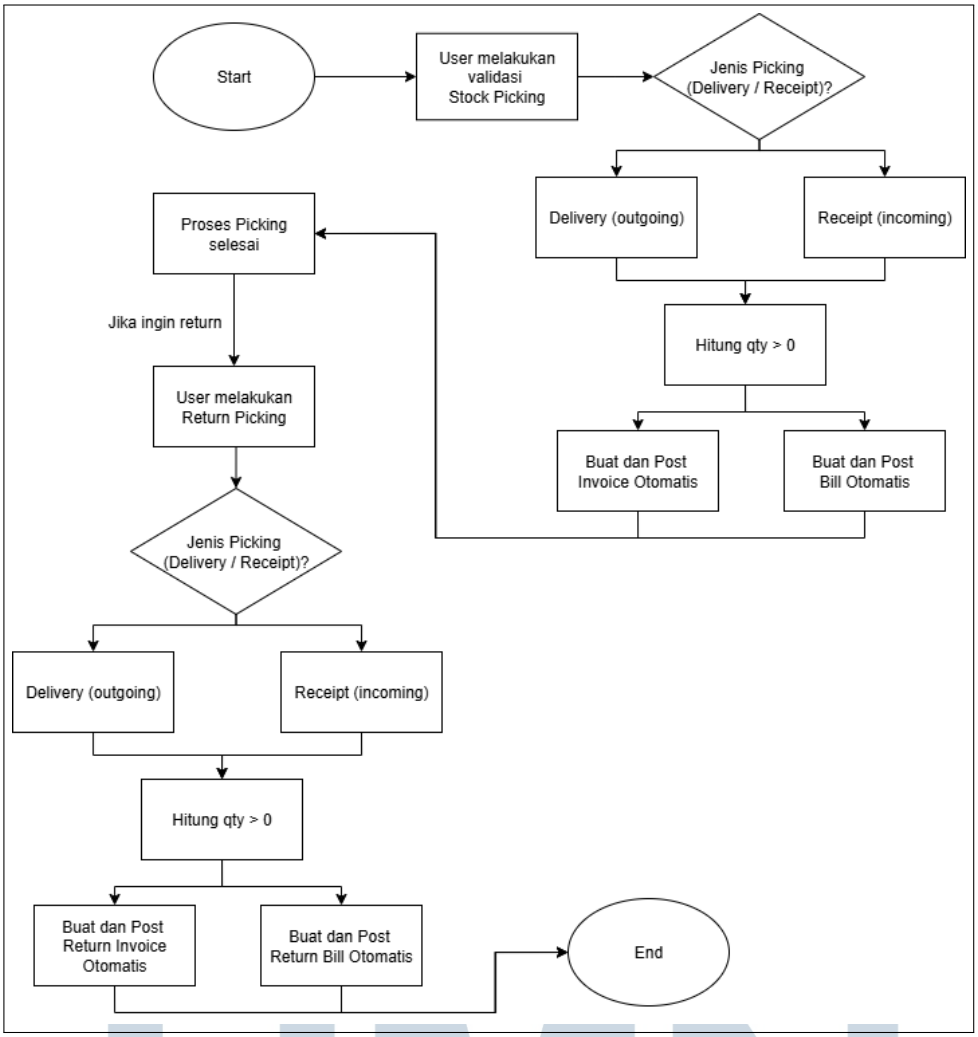

Gambar 3.7. Tampilan Odoo 15




Gambar 3.8. Tampilan Odoo 18

### 3.3.5 Pengembangan Sistem Auto Bill Invoice dan Auto Return Bill Invoice

Flowchart pada Gambar 3.9 menggambarkan bagaimana proses validasi pengiriman atau penerimaan dapat secara otomatis membuat invoice atau bill, dan bagaimana proses return juga ditangani dengan membuat dokumen kredit atau debit secara terintegrasi. Fitur ini mempercepat proses akuntansi dan mengurangi potensi human error dalam pembuatan dokumen finansial.




Gambar 3.9. Flowchart cara kerja Sistem Auto Bill Invoice dan Auto Return Bill Invoice

```python
from odoo import models
from datetime import date


class StockPicking(models.Model):
    _inherit = 'stock.picking'

    def button_validate(self):
        res = super().button_validate()

        auto_validate_invoice = self.env['ir.config_parameter'].sudo().get_param(
            'automatic_invoice_and_post.is_create_invoice_delivery_validate'
        )
```

```
14
15          if auto_validate_invoice:
16              for picking in self:
17                  if picking.picking_type_code == 'outgoing' and
    picking.sale_id:
18                      sale = picking.sale_id
19                      if any(move.quantity > 0 for move in picking.
    move_ids_without_package):
20                          try:
21                              invoice = sale._create_invoices()
22                              if invoice:
23                                  invoice.action_post()
24                          except Exception as e:
25                              pass
26
27                  if picking.picking_type_code == 'incoming' and
    picking.purchase_id:
28                      purchase = picking.purchase_id
29                      if any(move.quantity > 0 for move in picking.
    move_ids_without_package):
30                          try:
31                              bill_vals = purchase.
    action_create_invoice()
32                              if bill_vals:
33                                  bill = self.env['account.move'].
    browse(bill_vals.get('res_id'))
34                                  bill.invoice_date = date.today()
35                                  bill.date = date.today()
36                                  bill.action_post()
37                          except Exception as e:
38                              pass
39          return res
40
41
42  from odoo import api, fields, models
43
44
45  class StockReturnPicking(models.TransientModel):
46      """Inherited this class to display the view, to add the cancel
        reason."""
47      _inherit = 'stock.return.picking'
48
49      picking_type_name = fields.Char(string='Picking Type Name',
```

```
                                    help='Name of the picking type
')

    picking_type_code = fields.Selection([
        ('incoming', 'Receipts'),
        ('outgoing', 'Delivery Orders'),
        ('internal', 'Internal Transfers'),
    ], string="Picking Type Code", help="Technical type of picking
")

    @api.model
    def default_get(self, fields):
        result = super(StockReturnPicking, self).default_get(
fields)
        active_model = self.env.context.get('active_model')
        active_id = self.env.context.get('active_id')
        if active_model == 'stock.picking' and active_id:
            stock_picking = self.env[active_model].browse(
active_id)
            picking_type = stock_picking.picking_type_id
            result['picking_type_name'] = picking_type.name
            result['picking_type_code'] = picking_type.code
        return result

    def _update_stock_picking(self):
        """Update the 'is_paid' field of the active stock picking.
"""
        active_model = self.env.context.get('active_model')
        active_id = self.env.context.get('active_id')

        if active_model == 'stock.picking' and active_id:
            stock_picking = self.env[active_model].browse(
active_id)
            stock_picking.is_paid = True

    def _get_return_action(self):
        """
        Retrieve the action for returning a move, specifically for
 credit notes.
        """
        return self.env["ir.actions.actions"]._for_xml_id("
return_invoice_bill.return_move_action")

```

```
85      def action_returns_with_credit_note(self):
86          """
87          Perform the action of returning moves with credit notes
    and update picking.
88          """
89          self._update_stock_picking()
90          return self._get_return_action()
91
92      def action_returns_with_debit_note(self):
93          """
94          Perform the action of returning moves with debit notes and
     update picking.
95          """
96          self._update_stock_picking()
97          return self._get_return_action()
```

Kode 3.4: Listing implementasi Odoo Automatic Invoice dan Return Picking

Potongan kode 3.4merupakan pengembangan pada model `stock.picking` dan `stock.return.picking` untuk mendukung proses otomatisasi pembuatan invoice dan bill saat validasi picking, serta pembuatan dokumen return berupa credit note dan debit note. Fitur `button_validate()` pada `stock.picking` diubah agar saat pengguna memvalidasi pengiriman (delivery) atau penerimaan (receipt), sistem secara otomatis memeriksa apakah parameter konfigurasi `automatic_invoice_and_post.is_create_invoice_delivery_validate` diaktifkan. Jika ya, maka sistem akan secara otomatis membuat invoice dari sales order (untuk outgoing) atau bill dari purchase order (untuk incoming), kemudian langsung melakukan posting (konfirmasi) dokumen tersebut.

Selain itu, pengembangan juga dilakukan pada proses return picking. Pada model `stock.return.picking`, ditambahkan field tambahan untuk mencatat nama dan kode jenis picking yang sedang diproses, serta fungsi-fungsi baru seperti `action_returns_with_credit_note()` dan `action_returns_with_debit_note()`. Fungsi ini akan meng-update status `is_paid` pada record picking dan mengarahkan pengguna ke action pembuatan credit note atau debit note yang sesuai. Sistem ini dirancang untuk mempercepat proses penyesuaian keuangan dan menciptakan integrasi yang lebih otomatis antara logistik dan akuntansi dalam proses pengembalian barang.

## 3.4 Kendala dan Solusi yang Ditemukan

Selama proses migrasi dan implementasi sistem ERP Odoo, berbagai kendala muncul baik dari sisi teknis maupun non-teknis. Kendala-kendala ini tidak hanya berkaitan dengan aspek konfigurasi sistem, tetapi juga terkait kesiapan infrastruktur, adaptasi pengguna, serta kompatibilitas modul antar versi Odoo. Untuk memastikan kelancaran operasional perusahaan, setiap masalah tersebut diidentifikasi secara detail dan dicari solusi yang tepat melalui proses debugging, evaluasi, perencanaan ulang, dan pendampingan pengguna. Berikut ini adalah rangkuman kendala yang ditemui beserta solusi yang telah dilakukan untuk mengatasinya:

- **Perbedaan Perilaku Sistem antara Staging dan Production (Go Live)**
  Modul yang berjalan baik di environment staging menghasilkan error saat go-live, terutama pada modul kustom seperti replenishment dan POS. *Solusi:* Dilakukan debug ulang di environment production dengan memeriksa log error, membandingkan konfigurasi staging dan production (parameter sistem, hak akses, data), dan melakukan penyesuaian script kustom agar kompatibel.

- **Masalah Dependensi Modul saat Pengembangan di Odoo 15**
  Beberapa modul tidak dapat diinstal karena bergantung pada modul core atau pihak ketiga yang tidak lengkap atau strukturnya berubah di Odoo 18.
  *Solusi:* Dilakukan riset mendalam melalui komunitas Odoo, dokumentasi resmi, dan repositori GitHub untuk memahami dependensi modul, kemudian dilakukan refactoring kode agar sesuai versi terbaru.

- **Migrasi Batch 1 Tidak Menyeluruh**
  Data yang masuk belum lengkap dan banyak proses bisnis belum sesuai alur perusahaan sehingga mengganggu operasional.
  *Solusi:* Disusun perencanaan ulang migrasi batch 2, dengan pemetaan ulang data, pengujian lebih ketat (data testing flow), dan pelibatan pengguna kunci untuk validasi sebelum go-live.

- **Keterbatasan Akses dan Hak Pengguna**
  Pengguna tidak dapat melihat data penting seperti partner di Purchase Order (PO) atau Delivery Order (DO) akibat pengaturan hak akses yang belum optimal.
  *Solusi:* Implementasi kontrol akses dinamis berbasis grup dan checkbox

pada model `res.users`, sehingga pengaturan hak akses lebih fleksibel sesuai kebutuhan divisi.

- **Performa Lambat Akibat Jumlah Data Besar**
  Modul price approval dan vendor pricelist memproses puluhan ribu data sehingga sistem lambat ketika filter atau membuka form.
  *Solusi:* Optimasi dilakukan dengan query SQL untuk mengambil data penting secara langsung, serta pembuatan wizard filter yang lebih ringan dan cepat daripada ORM standar.

- **Infrastruktur Belum Stabil saat Implementasi**
  Server tidak stabil, backup belum otomatis, dan proses restore database memakan waktu lama.
  *Solusi:* Dilakukan penjadwalan backup otomatis harian, penyediaan prosedur restore cepat, dan penggunaan cloud atau virtual machine redundancy sebagai mitigasi downtime.

- **Ketidaksesuaian Antara Flow Bisnis dan Fitur Odoo Default**
  Transaksi antar-company, approval harga jual, dan penggunaan cost manual refund di POS tidak didukung fitur standar Odoo.
  *Solusi:* Dibuat penyesuaian kustom modul agar mendukung kebutuhan bisnis, termasuk pengaturan cost refund manual dan proses persetujuan harga jual multi-level.

- **Adaptasi Pengguna terhadap Sistem Baru**
  Pengguna mengalami kesulitan memahami tampilan dan alur baru, yang berdampak pada kesalahan input dan proses kerja yang lambat.
  *Solusi:* Disediakan sesi pelatihan interaktif, SOP berbentuk video dan PDF, dokumentasi FAQ, serta tim support internal untuk mendampingi pengguna pada masa transisi.

- **Kebutuhan Sistem Cadangan**
  Saat terjadi masalah kritis pada sistem baru, operasional tidak boleh berhenti total.
  *Solusi:* Odoo 15 tetap dijadikan fallback sementara jika sistem baru belum stabil, sehingga transaksi tetap dapat berjalan.

- **Monitoring Aktivitas Sistem dan Pengguna**
  Ketika masalah muncul, seringkali sulit menelusuri penyebab tanpa data

aktivitas yang detail.

*Solusi:* Aktivasi `ir.logging`, audit trail, dan log kustom untuk memantau aktivitas pengguna maupun sistem secara real-time sehingga investigasi lebih cepat.