BAB 3
PELAKSANAAN KERJA MAGANG

Pelaksanaan kerja magang yang terorganisir dengan baik menjadi hal yang harus diperhatikan karena dapat menjadi faktor pendukung dalam setiap kontribusi yang diberikan. Pada bab 3 ini, akan dibahas mengenai kedudukan dan koordinasi, tugas yang dilakukan, uraian pelaksanaan magang, serta kendala dan solusi yang ditemukan.

3.1 Kedudukan dan Koordinasi

Dalam menjalani kegiatan magang ini, ditempatkan pada departemen Odoo dan berperan sebagai *Odoo Developer*. Tepat berada di bawah supervisi langsung dari Muhammad Hafidz sebagai *Lead Technical Team/Supervisor* dan Rifky Rifandis sebagai *Project Manager*. Dalam menjalankan tugas dan tanggung jawab, dilakukan koordinasi dengan:

- *Supervisor*, untuk mendapatkan arahan dalam penanganan dan pengerjaan sistem.
- *Project Manager*, untuk mendapatkan tugas harian serta mendiskusikan fitur-fitur yang ingin dibuat.

Sistematika pengerjaan diikuti sesuai dengan arahan dari *Supervisor* dan *Project Manager*, seperti menggunakan Github untuk kolaborasi *repository* serta prosedur lain yang telah ditetapkan.

3.2 Tugas yang Dilakukan

Selama menjalani kegiatan magang sebagai Odoo *Developer* pada PT Adiraja Integrasi, terdapat beberapa tugas yang berhubungan langsung dengan proses pengembangan dan penyesuaian sistem ERP Odoo yang dibutuhkan oleh klien perusahaan. Beberapa tugas yang dilakukan antara lain:

1. **Kustomisasi Modul Odoo**: Melakukan modifikasi modul pada Odoo seperti menambahkan *field*, modifikasi *button* dan tampilan, serta menambahkan proses dan fitur baru lainnya.

2. **Pembuatan Modul Baru**: Diberikan tanggung jawab untuk bisa mengembangkan modul baru lengkap dengan seluruh proses yang diminta yang terdiri dari pembuatan model (`models.py`), tampilan antarmuka (`views.xml`), hak akses (`ir.model.access.csv`), serta pengaturan menu dan *action*. Modul baru yang dibuat antara lain, modul *training* dan modul *consignment*.

3. **Pembuatan Dashboard**: Mengembangkan *dashboard* dengan menu *purchasing* dan *fleet* untuk klien dalam Odoo 14 dengan menggunakan (`.py`, `.js`, dan `.xml`). Dalam *dashboard* ini terdapat tabel dan juga *chart* yang berfungsi untuk analisa dan juga untuk *tracking* data yang diambil dari *Purchase Request, Product, Purchase Order, Supplier, Transaction, dan Product on Delivery*. Lalu, terdapat juga *filter-filter* data sesuai jangka waktu, yaitu *All, Today, This Week* dan *This Month*. *Filter all*, berfungsi untuk menampilkan keseluruhan data yang masuk. *Filter today*, berfungsi untuk menampilkan data yang masuk hingga pukul 09.00 pagi hari ini dari kemarin. *Filter this week*, berfungsi untuk menampilkan data yang masuk hingga pukul 09.00 pagi hari ini dari 7 hari ke belakang. Lalu, untuk *filter this month* berfungsi untuk menampilkan data yang masuk hingga pukul 09.00 pagi hari ini dari 30 hari ke belakang.

4. **Penggunaan Rest API**: Menggunakan Rest API untuk penarikan data yang bertujuan supaya data dapat ditampilkan ke *dashboard* yang telah dibuat, seperti data *Purchase Summary*, *Purchase Request*, *Purchase Order*, *Product*, dan *Transaction*. Selain itu juga, Rest API yang digunakan dalam Odoo bertujuan untuk mengetahui apakah data yang diinput masuk ke *record* Odoo.

5. **Mempersiapkan Server untuk Demo Implementation**: Melakukan konfigurasi dan *setup server* dengan menggunakan *server* internal Adiraja untuk kebutuhan *demo implementation*. Odoo yang digunakan adalah Odoo 18 dikarenakan kebutuhan fitur yang lebih lengkap dan fungsional. *Setup server* menggunakan WSL sehingga Windows dapat terhubung dengan Linux. Secara singkat, langkah-langkah dalam mempersiapkan *server* meliputi *login* ke dalam IP internal, membuat *directory* yang bisa diakses oleh *user* tertentu, *install dependency* yang dibutuhkan, *install* odoo dengan *command line*, konfigurasi *port* (contoh port :8070) dan juga odoo *service*, lalu *deploy*.

3.3 Uraian Pelaksanaan Magang

Selama menjalankan kegiatan magang sebagai Odoo *Developer*, diberikan berbagai tugas yang berkaitan dengan pengembangan sistem ERP berbasis Odoo. Tugas-tugas tersebut mencakup aktivitas pemrograman dan desain antarmuka. Rincian pekerjaan tiap minggu selama pelaksanaan kerja magang disajikan pada Tabel 3.1.

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

Minggu Ke -	Pekerjaan yang dilakukan
1	Mempelajari dan memahami tentang Odoo versi 14
2	Melakukan modifikasi modul pada Odoo
3	Menambahkan dan modifikasi fitur yang tersedia pada modul yang dipakai
4	Membuat modul baru yaitu modul consignment
5	Membuat list tree view dan form untuk menu consignment agreement
6	Membuat menu consignment pricelist dan consignment stock di module consignment
7	Membuat modul baru yaitu modul training
8	Membuat menu training module, training code, training tools, training task dalam modul training
9	Revisi dan membuat button baru pada menu training code dan training task
10	Mempelajari pembuatan custom dashboard pada Odoo 14 untuk klien
11	Memulai pembuatan dashboard menggunakan `.py`, `.xml`, dan `.js` serta membuat menu purchasing dan fleet untuk custom dashboard
12	Menambahkan filter waktu dan revisi tampilan custom dashboard
13	Maintenance dan mempersiapkan environment dashboard untuk go-live
14	Memulai konfigurasi Odoo 18 dan mempelajari fitur baru Odoo 18

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

Minggu Ke -	Pekerjaan yang dilakukan
15	Membuat modul baru untuk testing fungsi dan report yaitu modul Hospital Management System
16	Mempelajari setup server Odoo 18
17	Memulai setup server Odoo 18 di IP internal Adiraja dan instalasi modul serta addons tambahan

3.4 Perancangan Sistem

Pada bagian perancangan sistem, infrastruktur dan alur kerja sistem disusun secara menyeluruh dalam mendukung pengembangan modul-modul yang digunakan baik oleh klien maupun internal. Infrastruktur dan alur kerja sistem yang dibangun beserta dengan segala fiturnya akan dijelaskan melalui *flowchart* dan penjelasan secara detil. Langkah ini dilakukan untuk memastikan seluruh komponen teknologi yang dibangun dapat berjalan dengan optimal, efisien, serta fungsional dalam pengelolaan sumber daya pertambangan.

3.4.1 Platform dan Sistem yang Digunakan

Odoo merupakan *open source* ERP yang digunakan sebagai fondasi dalam pengembangan modul-modul untuk pengembangan pada ERP PT Mitra Indah Lestari dan PT Adiraja Integrasi. Sesuai dengan kebutuhan bisnis yang dibutuhkan oleh klien, maka Odoo yang digunakan adalah Odoo 14 *Community Edition*.

Keberhasilan pengembangan modul-modul Odoo ini didukung beberapa komponen teknologi, sebagai berikut:

A. **Database PostgreSQL**
PostgreSQL adalah *database* yang digunakan dalam pengembangan modul Odoo. Versi PostgreSQL yang digunakan oleh perusahaan adalah PostgreSQL versi 14 untuk Odoo 14 *Community Edition*.

B. **Operating System**
Operating System yang digunakan adalah Linux melalui WSL (*Windows Subsystem for Linux*). Sistem ini digunakan untuk memastikan operasional yang fleksibel dalam konfigurasi maupun pengembangan Odoo.

C. **Server**

Server yang dipakai merupakan *server* internal PT Adiraja Integrasi. *Server* ini sudah dibekali SSH (*Secure Shell*) oleh *hosting* sehingga memungkinkan keamanan yang lebih terjamin dalam pengembangan dan penggunaan Odoo bagi klien ataupun internal.

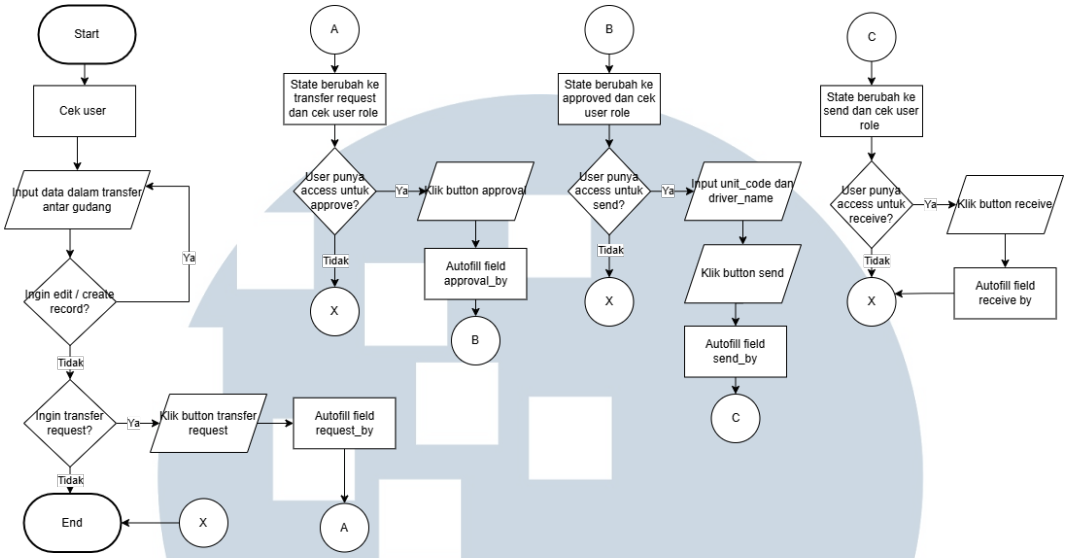
3.4.2 Flowchart Sistem

Sistem yang dirancang tentu saja harus dibuat sedemikian rupa sehingga proses dan tampilan yang dibangun akan sesuai dengan permintaan klien dan internal. *Flowchart* dibuat sehingga proses yang akan dibangun akan dapat mudah dipahami dan lebih sistematis dalam pembuatannya.

Berikut adalah penjelasan detil mengenai tiap *flowchart* yang telah dibuat untuk menggambarkan proses dalam sistem ini:

A. **Flowchart Transfer Antar Gudang**

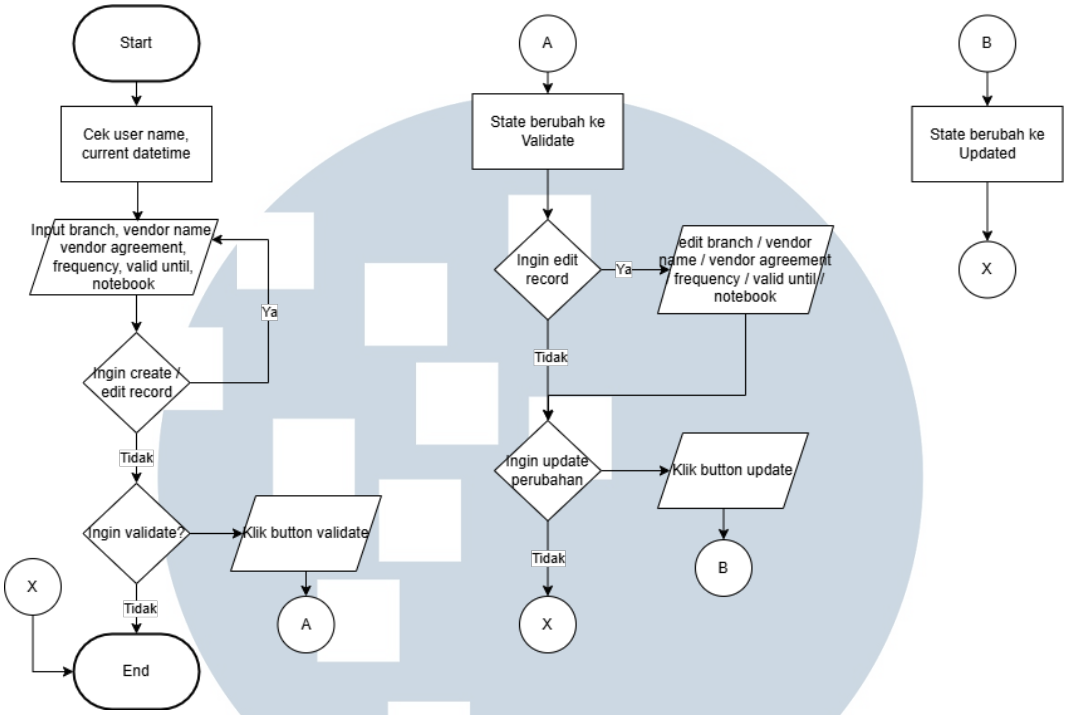
Dalam modul *inventory*, terdapat menu Transfer Antar Gudang yang diberlakukan beberapa modifikasi dalam fungsinya. Pada saat pertama kali membuka menu TAG, Odoo akan cek *user* yang sedang mengakses menu tersebut lalu jika memiliki hak akses, maka dapat membuat *record* TAG (Transfer Antar Gudang). Jika hak akses hanya sebatas membuat *record*, maka setelah membuat *record* dapat melakukan *transfer request* dengan menekan tombol *transfer request*. Setelah tombol *transfer request* ditekan, maka *state* berubah menjadi *transfer request*, lalu *user* yang memiliki hak akses untuk *approve* dapat memberikan *approve* dengan menekan tombol *approval*. Setelah tombol *approval* ditekan, *state* berpindah menjadi *approved*. *User* yang memiliki hak akses untuk *send* dapat *send record* tersebut dengan menekan tombol send, namun sebelum *send* diwajibkan untuk *input unit code* dan *driver name*. Lalu, *state* akan berpindah menjadi *send*, jika *user* memiliki hak akses untuk *receive*, maka dapat *receive* dengan menekan *button receive*. Setelah itu, *state* akan berpindah ke *done* dan *record* sudah tidak dapat dilakukan pengubahan lagi. *Flowchart* transfer antar gudang dapat dilihat pada Gambar 3.1.



Gambar 3.1. *Flowchart* Transfer Antar Gudang

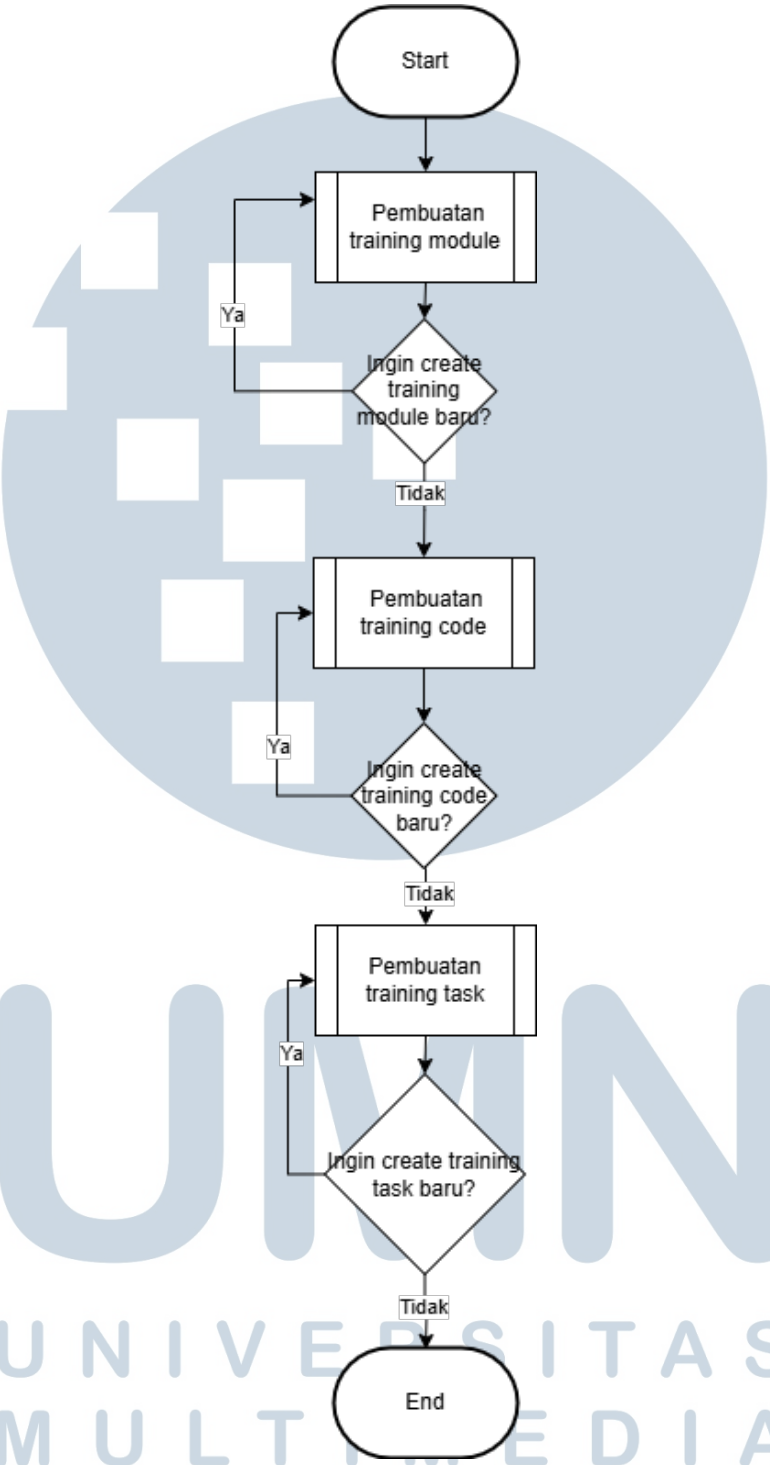
B. **Flowchart Consignment Agreement**

Dalam modul *consignment*, terdapat menu *consignment agreement* yang merupakan modul dan menu baru yang dibuat untuk kebutuhan *consignment* PT Mitra Indah Lestari. Saat pertama kali membuka menu ini, Odoo akan cek *user name* dan *datetime*. Lalu, *user* dapat melakukan *input* pada semua *required field* untuk membuat *record*, *record* yang dibuat dapat di-*edit* setelah di-*save*, lalu jika ingin *validate* maka tekan tombol *validate*. Setelah *validate* ditekan, maka *state* berubah ke *validate*. Jika ingin melakukan perubahan terakhir pada *record*, maka isi *form* dapat disesuaikan dengan kebutuhan. Jika *record* sudah sesuai, maka dapat tekan *button update* untuk memperbarui *record*, dan *state* akan berubah menjadi *updated*. *Flowchart consignment agreement* dapat dilihat pada Gambar 3.2.



Gambar 3.2. *Flowchart Consignment Agreement*

C. **Flowchart Module Training** *Module training* dibuat untuk kepentingan supaya para klien dapat memahami sistem dan juga alur yang dapat dilakukan untuk melakukan kustomisasi serta fungsionalitas dari fitur yang telah ada. *Module training* yang telah dibuat memiliki beberapa menu seperti *training module*, *training code*, dan *training task*. Fitur yang dimiliki dalam *module* ini baik dari tampilan, fungsi tombol, dan proses pembuatan data dapat menjadi gambaran supaya klien dapat dengan mudah memahami dan mengembangkannya. *Flowchart module training* tertera pada gambar 3.3.

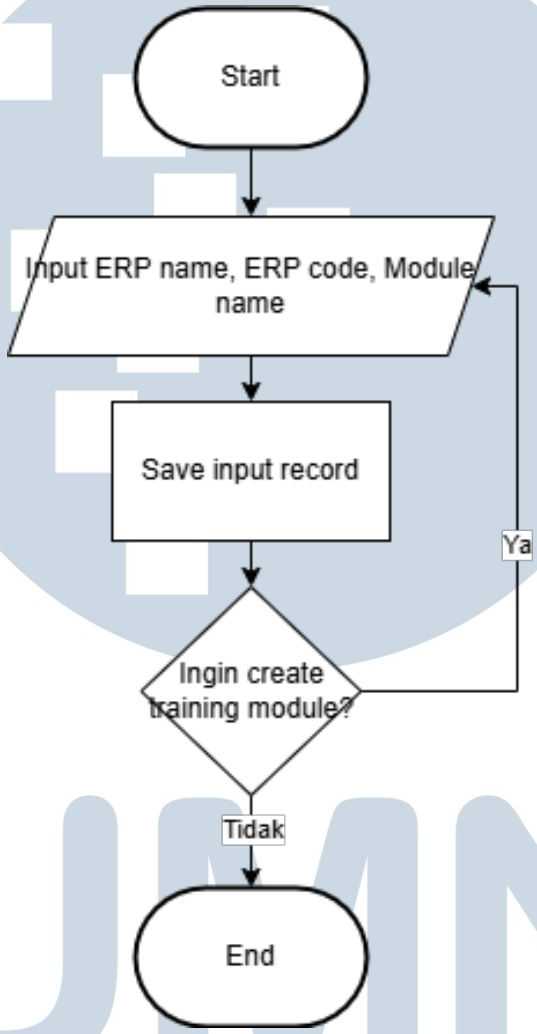


Gambar 3.3. *Flowchart Module Training*

D. **Flowchart Training Module**

Modul baru yang juga dikembangkan untuk kebutuhan *training* data PT Mitra Indah Lestari adalah modul *training*. Modul ini memiliki beberapa menu,

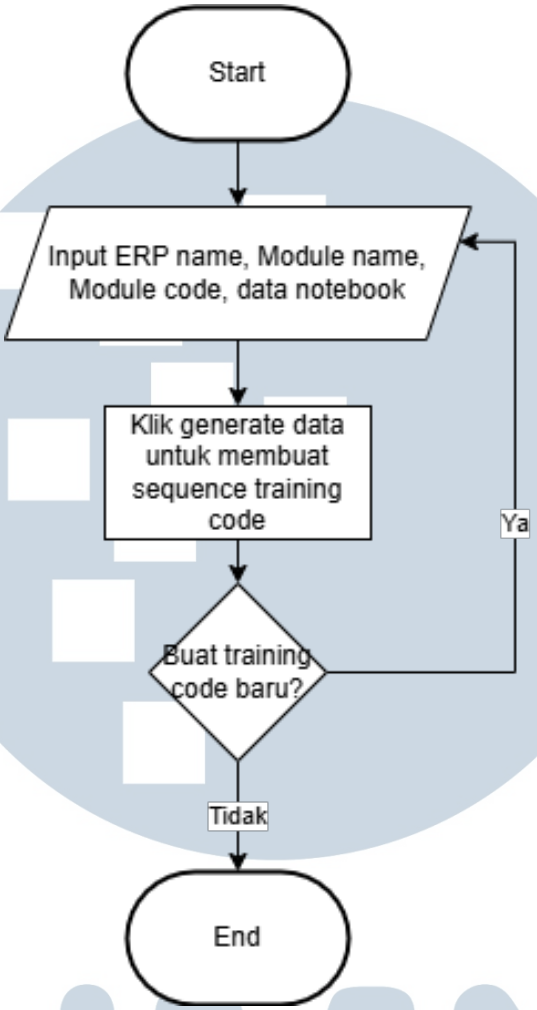
salah satunya *training module*. Saat pertama masuk menu ini, *user* dapat *input* semua *field* di *tree view*, yaitu *erp name*, *erp code*, dan *module name*. *Flowchart training module* dapat dilihat pada Gambar 3.4.



Gambar 3.4. *Flowchart* Pembuatan *Training Module*

E. **Flowchart Training Code**

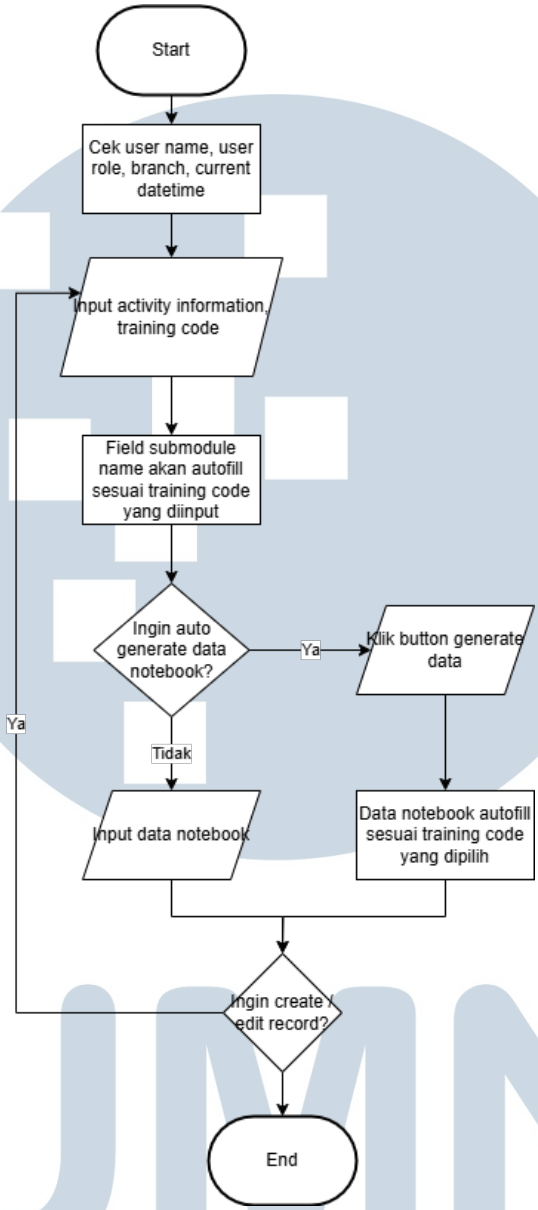
Menu lain dari modul *training* adalah *training code*. Pada menu ini, *user* dapat *input* semua *required field* yaitu *erp name*, *module name*, dan *module code*, serta *notebook* yang tertera. Lalu, setelah seluruh isi *notebook* terisi kecuali *training code*, maka dapat menekan tombol *generate training code* untuk mengisi *training code* secara otomatis pada *notebook*. Jika ingin membuat *training code* baru, maka *user* dapat membuat *record* baru. *Flowchart training code* dapat dilihat pada Gambar 3.5.



Gambar 3.5. *Flowchart* Pembuatan *Training Code*

F. **Flowchart Training Task**

Menu *training task* merupakan menu lain dalam modul *training*. Pada saat membuka menu ini, Odoo akan cek *user name*, *user role*, *branch*, dan *datetime*. Jika memiliki hak akses, *user* dapat *input activity information* dan *training code* (diambil dari menu *training code*), lalu *field submodule name* akan *autofill* sesuai *training code* yang diinput. Jika ingin *generate* data secara otomatis, maka dapat menekan tombol *generate* data dan data yang diambil sesuai dari *training code*, jika *training code* yang dipilih tidak memiliki data di *notebook* maka data tidak akan di-*generate*. *User* dapat melakukan *edit* dan membuat *record* baru, lalu jika sudah tidak ada perubahan maka dapat *save record*.



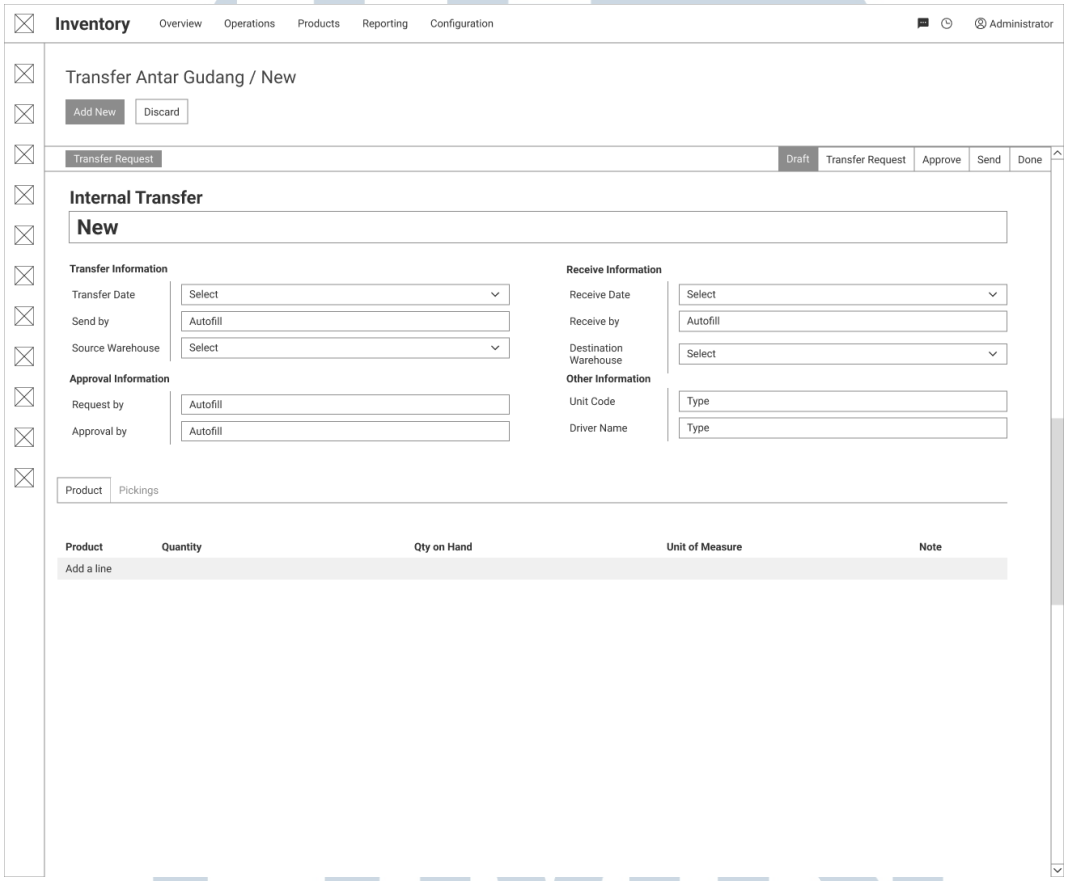
Gambar 3.6. *Flowchart* Pembuatan *Training Task*

3.5 Perancangan UI/UX

Perancangan UI/UX dibutuhkan sebagai fundamental dan juga pedoman dalam pembangunan modul ERP Odoo, sehingga dapat memenuhi kebutuhan klien. Tentu saja, dalam membangun tampilan sistem ini harus memiliki fungsionalitas yang baik tetapi juga mudah untuk dipahami dan digunakan. Kerangka-kerangka tampilan awal Odoo untuk klien akan dijelaskan sebagai berikut.

3.5.1 Wireframe Transfer Antar Gudang

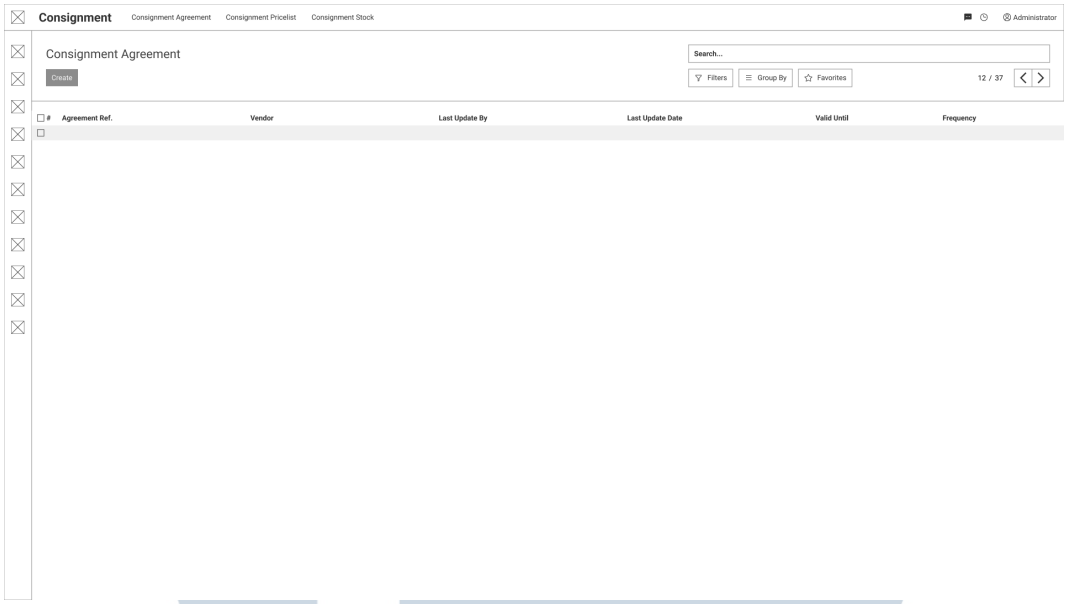
Pada bagian ini, dibuat kerangka tampilan supaya tepat di atas *form* terdapat tombol dan juga beberapa *state*. Kerangka tampilan untuk transfer antar gudang dapat dilihat pada Gambar 3.7.



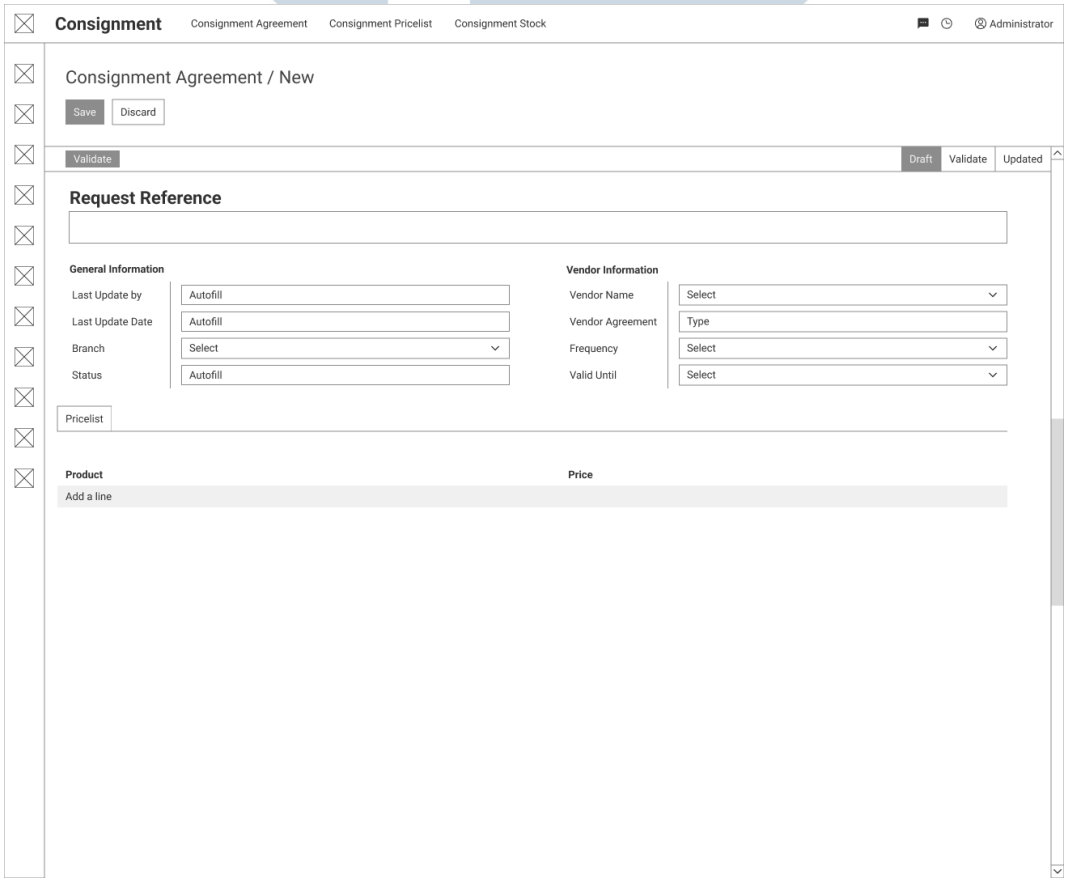
Gambar 3.7. *Wireframe* Transfer Antar Gudang

3.5.2 Wireframe Consignment

Pada bagian ini, terdapat kerangka tampilan untuk *Consignment Agreement* yang berupa *tree view* dan *form view*. Kerangka tampilan *tree view* dapat dilihat pada Gambar 3.8, sedangkan kerangka tampilan *form view* dapat dilihat pada Gambar 3.9.



Gambar 3.8. *Wireframe Tree View Consignment Agreement*



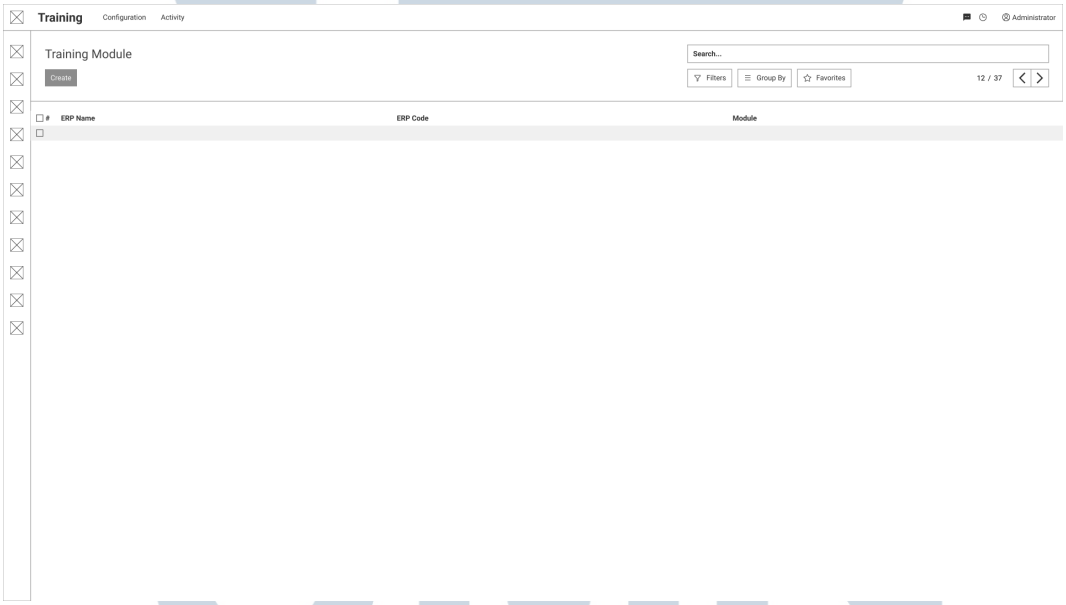
Gambar 3.9. *Wireframe Form Consignment Agreement*

3.5.3 Wireframe Training

Pada bagian ini, terdapat kerangka tampilan untuk 3 menu yaitu *training module*, *training code*, dan *training task*. *Training task* dan *training code* memiliki *tree view* dan *form view*, sedangkan *training module* hanya memiliki *editable tree view*. Beberapa kerangka tampilan tersebut akan dijelaskan sebagai berikut.

A Wireframe Training Module

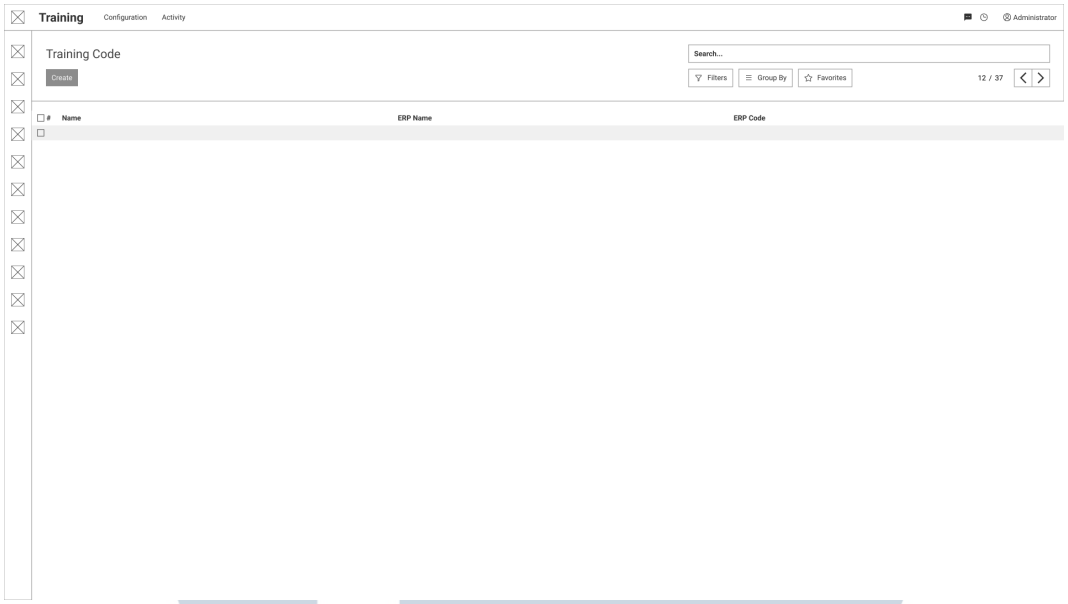
Pada menu ini, *view* dibuat sepraktis mungkin karena tidak memiliki proses yang rumit. Maka dari itu, menu ini hanya memiliki *editable tree view*. Kerangka tampilan *editable tree view training module* dapat dilihat pada Gambar 3.10.



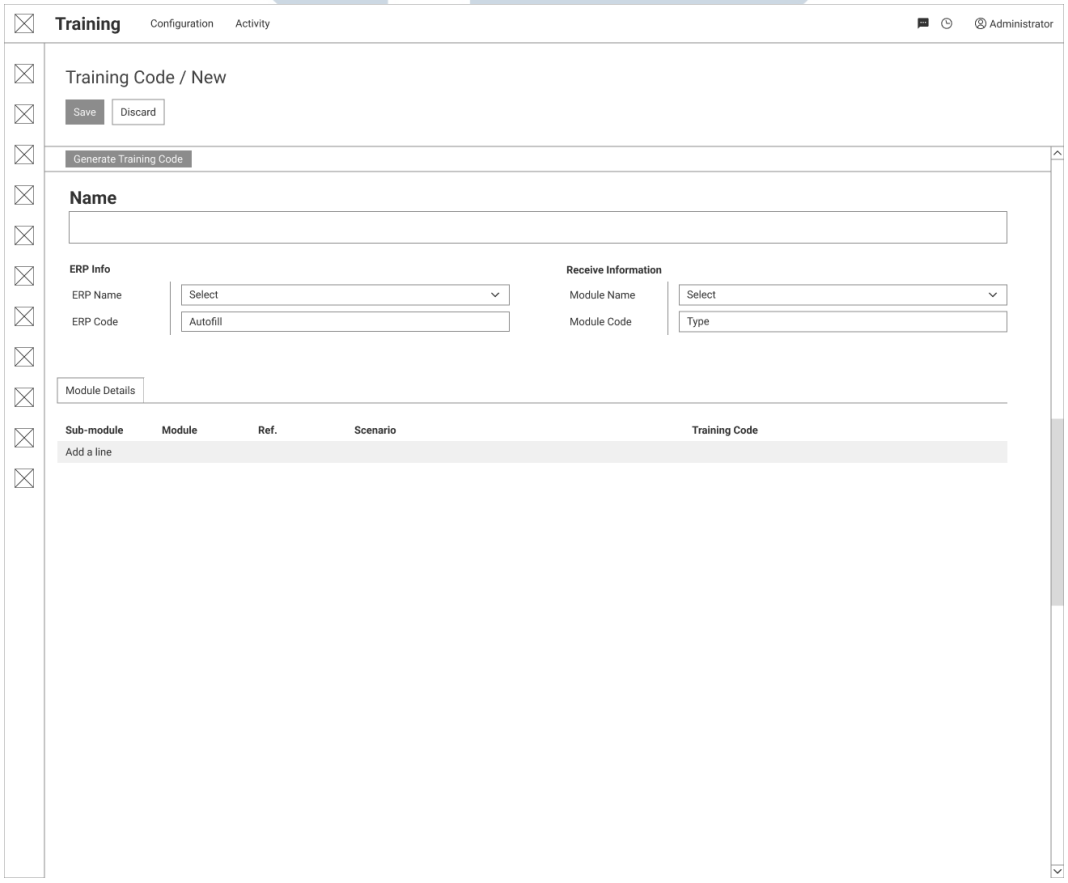
Gambar 3.10. *Wireframe Tree View Training Module*

B Wireframe Training Code

Pada menu ini, *tree view* tidak *editable* karena terdapat *form view* yang akan secara otomatis mengisi data pada *tree view*. Dalam *form view* hanya ada tambahan pada *field*, *notebook*, dan juga tombol *generate training code*. *Tree view* untuk *training code* dapat dilihat pada Gambar 3.11, sedangkan *form view* dapat dilihat pada Gambar 3.12.



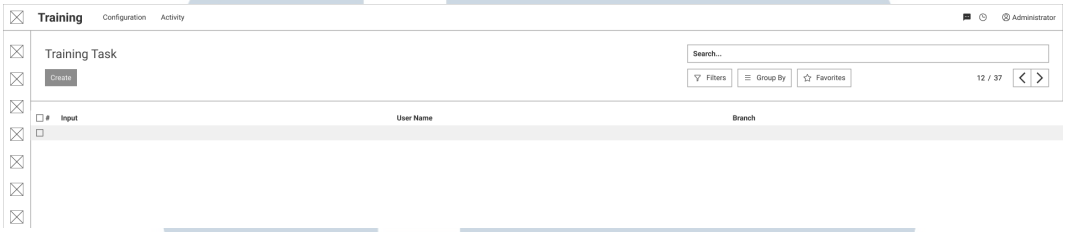
Gambar 3.11. *Wireframe Tree View Training Code*



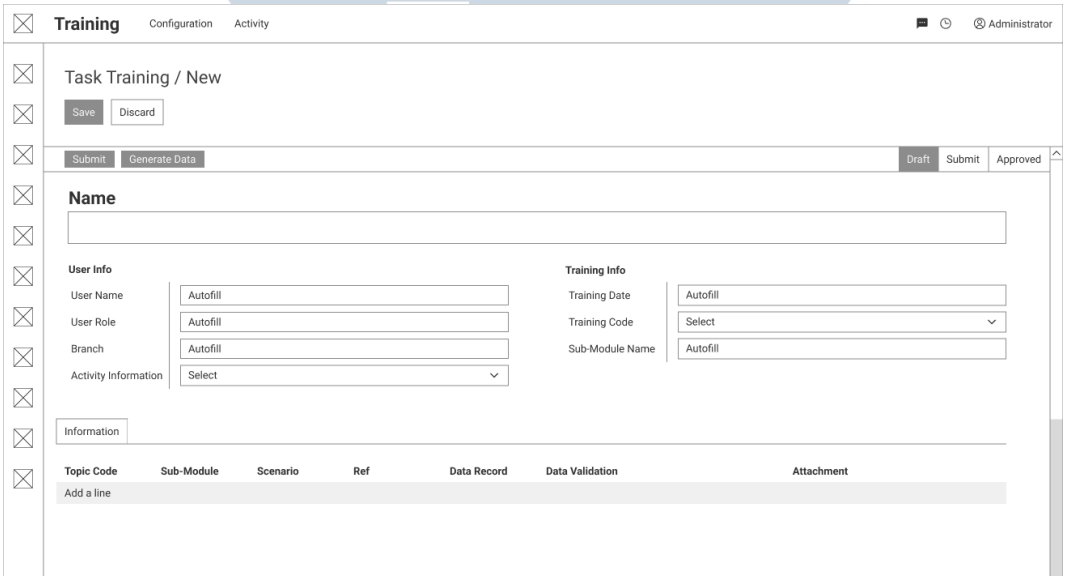
Gambar 3.12. *Wireframe Form Training Code*

C Wireframe Training Task

Pada menu ini, kerangka tampilannya tidak jauh berbeda dengan kerangka tampilan *training code*. Namun, pada *form view* menu ini terdapat penambahan *state* dan juga tombol. Kerangka tampilan *tree view training task* dapat dilihat pada Gambar 3.13, sedangkan *form view* dapat dilihat pada Gambar 3.14.



Gambar 3.13. *Wireframe Tree View Training task*



Gambar 3.14. *Wireframe Form Training Task*

3.6 Potongan Code

Pembangunan modul ERP Odoo tentu saja tidak berupa rancangan tampilan dan *flowchart* saja, namun dibutuhkan juga *code* sehingga seluruh rancangan tersebut dapat direalisasikan. Potongan *code* transfer antar gudang, *code consignment*, dan *code training* merupakan *code* penting dalam kasus

pengembangan Odoo PT Mitra Indah Lestari. Potongan *code* tersebut akan dijelaskan sebagai berikut.

3.6.1 Code Transfer Antar Gudang

Dalam menu transfer antar gudang, terdapat beberapa *code* yang menjadi dasar dan *function* dalam berjalannya sistem. Beberapa *code* yang menjadi *code* penting dalam sistem transfer gudang adalah sebagai berikut.

A Code Check User Role

Pada *code* ini, terlihat bahwa hak akses *user* terbagi menjadi 5 dan masing-masing memiliki fungsi yang berbeda, yaitu *allowsend*, *allowreceive*, *allowsendreceive*, *allowapproval*, dan *allaccess*. Tiap hak akses tersebut juga diperuntukkan pada *state* tertentu. Potongan *code check user role* dapat terlihat pada Gambar 3.15.

```
@api.model  William Purba
def _get_user_role(self):
    return self.env.user.role

def _check_user_role(self, action):  2 usages  William Purba
    user_role = self.env.user.role
    role_actions = {
        'allowsend': ['action_transfer_request', 'action_send'],
        'allowreceive': ['action_transfer_request', 'action_receive'],
        'allowsendreceive': ['action_transfer_request', 'action_send', 'action_receive'],
        'allowapproval': ['action_approve'],
        'allaccess': ['action_transfer_request', 'action_approve','action_send', 'action_receive']
    }
    return action in role_actions.get(user_role, [])
```

Gambar 3.15. *Code Check User Role*

B Code Sequence

Dalam *code* ini, setiap kali *record* dibuat maka akan muncul. *Trigger* dalam pembuatan *sequence code* ini adalah ketika *field name*nya adalah "*New*" dan dengan *rule ir.sequence*. Potongan *code sequence* dapat dilihat pada Gambar 3.16.

```
@api.model  William Purba +1
def create(self, vals):
    if vals.get('name', 'New') == 'New':
        vals['name'] = self.env['ir.sequence'].next_by_code('stock.internal.transfer') or 'New'
    return super(StockInternalTransfer, self).create(vals)
```

Gambar 3.16. *Code Sequence*

C Code Action Transfer Request

Method pada *code* ini berfungsi untuk *trigger* tombol *transfer request*. Proses yang berjalan dalam *method* ini yaitu cek apakah *user* memiliki akses untuk melakukan *request*. Jika memiliki akses *request*, maka *user name* beserta *datetime* melakukan *request* akan tampil, selain itu *state* juga akan berpindah. Pada Gambar 3.17 merupakan potongan *code action transfer request*.

```
def action_transfer_request(self):  William Purba +1 *
    if not self._check_user_role('action_transfer_request'):
        raise UserError("You do not have permission to perform this action")

    for record in self:
        user_role = False
        warehouse = self.env['stock.warehouse'].search([
            '|',
            ('branch_id', '=', False),
            ('branch_id', 'in', self.env.user.branch_ids.ids)
        ])
        if warehouse:
            user_role = self.env.user.role
        user_name = self.env.user.name
        formatted_name = f"<span style='color:#00A09D;'>{user_name}</span>"
        formatted_date = datetime.now().strftime('%Y-%m-%d %H:%M:%S')
        next_state = 'transfer_request'

        record.write({
            'request_by': f"{formatted_name}  {formatted_date}",
            'state': next_state,
            'user_role': user_role
        })
    return True
```

Gambar 3.17. *Code Action Transfer Request*

D Code Action Approve

Method action approve ini tidak jauh berbeda dalam hal fungsinya seperti *method action transfer request*. *Method* ini akan melakukan pengecekan hak akses *user*, lalu jika *user* dapat melakukan *approve*, maka *approver name* dan *approval date* akan tampil. Saat tombol *approve* ditekan, *state* juga akan berpindah. Potongan *code action approve* dapat terlihat pada Gambar 3.18.

```
def action_approve(self):  William Purba +1
    if not self._check_user_role('action_approve'):
        raise UserError("You do not have permission to perform this action")

    for rec in self:
        if rec.state != 'transfer_request':
            raise UserError(f"Invalid transfer state: {rec.state}. Cannot approve.")

    for rec in self.line_ids:
        if rec.on_hand <= 0 :
            raise UserError('Qty %s Tidak tersedia pada source warehouse'%rec.product_id.name)

    approver_name = self.env.user.name
    formatted_approver_name = f"<span style='color:#00A09D;'>{approver_name}</span>"
    formatted_approval_date = datetime.now().strftime('%Y-%m-%d %H:%M:%S')
    self.write({
        'state': 'approve',
        'approve_by': f"{formatted_approver_name} {formatted_approval_date}",
    })
    return True
```

Gambar 3.18. *Code Action Approve*

E Code Driver Name dan Unit Code

Method ini berfungsi untuk menampilkan teks *Not yet selected* pada *field unit code* dan *driver name*. Teks *Not yet selected* hanya akan muncul ketika *state* berada di *draft* dan *transfer request*. Potongan *code driver name* dan *unit code* dapat dilihat pada Gambar 3.19.

```
@api.depends("state", "unit_code")  William Purba
def _compute_unit_code_display(self):
    for record in self:
        if record.state in ['draft', 'transfer_request']:
            record.unit_code_html = "<span style='color: gray;'>Not yet selected</span>"
        else:
            record.unit_code_html = record.unit_code.equipment_id if record.unit_code else ""

@api.depends("state", "driver_name")  William Purba
def _compute_driver_name_display(self):
    for record in self:
        if record.state in ['draft', 'transfer_request']:
            record.driver_name_html = "<span style='color: gray;'>Not yet selected</span>"
        else:
            record.driver_name_html = record.driver_name.name if record.driver_name else ""
```

Gambar 3.19. *Code Driver Name* dan *Unit Code*

F Code Not Yet Received

Pada potongan *code* ini, berfungsi untuk menampilkan *Not yet received* pada *field receive by*. Teks *Not yet received* ini akan muncul pada tiap *state* kecuali *state send* dan *done*. Potongan *code not yet received* dapat dilihat pada Gambar 3.20.

```
@api.depends('state')  William Purba
def _compute_receive_by(self):
    for rec in self:
        if rec.state not in ['send', 'done']:
            rec.receive_by = "Not yet received"
```

Gambar 3.20. *Code Not Yet Received*

3.6.2 Code Consignment

Pada modul *consignment*, terdapat beberapa potongan *code* yang menjadi dasar dalam berjalannya proses dalam modul *consignment* ini. Proses yang berjalan pada modul *consignment* sebagian besar berada pada menu *consignment agreement*. Beberapa *code* yang menjadi *code* penting dalam sistem *consignment* adalah sebagai berikut.

A Code Consignment Agreement

Pada potongan *code* ini, berfungsi sebagai dasar untuk membuat menu *consignment agreement*. *Code* ini berisi *field* yang dibutuhkan dalam proses *consignment agreement*. Pada Gambar 3.21 dan Gambar 3.22 adalah tampilan dari potongan *code consignment agreement*.

```
class ConsignmentAgreement(models.Model):  1 usage   William Purba +2
    _name = 'consignment.agreement'
    _description = 'Consignment Agreement'
    _rec_name = "agreement_ref"

    name = fields.Char(string="Name", store=True)
    status = fields.Html(string="Status", store=True)
    agreement_ref = fields.Char(string="Agreement Ref.", required=True)
    vendor_name = fields.Many2one(
        'res.partner',
        string="Vendor",
        required=True,
        domain="[('is_company', '=', True)]"
    )
    last_update_by = fields.Many2one("res.users", string="Last Update by", default=lambda self: self.env.user, readonly=True)
    last_update_date = fields.Datetime(string="Last Update Date", default=fields.Datetime.now, readonly=True)
    valid_until = fields.Date(string="Valid Until", required=True)
    branch_name = fields.Many2one('res.branch', string="Branch")
    frequency = fields.Selection([
        ('1_months', '1 MONTHS'),
        ('3_months', '3 MONTHS'),
        ('6_months', '6 MONTHS'),
        ('9_months', '9 MONTHS'),
        ('12_months', '12 MONTHS'),
    ], string="Frequency", required=True)
```

Gambar 3.21. *Code Consignment Agreement* 1

```
state = fields.Selection([
    ('draft', 'Draft'),
    ('validate', 'Validate'),
    ('update', 'Updated'),
    # ('done', 'Done'),
], string="State", default='draft', tracking=True)
pricelist_ids = fields.One2many(
    'consignment.pricelist', 'agreement_id', string="Pricelist"
)
```

Gambar 3.22. *Code Consignment Agreement* 2

B Code Draft dan Compute Vendor

Dalam *code* ini, *method action draft* berfungsi untuk mengatur sehingga *state* dimulai dari *draft*. *Method compute vendor selection* berfungsi untuk mencari nama *partner* dalam *res.partner* dalam *domain company name*. Lalu, dilanjutkan dalam *method fields get* untuk menampilkan dropdown pada *field vendor name* di *form* dari komputasi yang telah dijalankan. Potongan *code draft* dan *compute vendor* dapat dilihat pada Gambar 3.23.

```
def action_draft(self):  William Purba
    self.write({'state': 'draft'})
    return True

@api.model  1 usage  William Purba
def _compute_vendor_selection(self):
    partners = self.env['res.partner'].search([('company_name', '!=', False)])
    return [(str(partner.id), partner.company_name or partner.name) for partner in partners]

@api.model  William Purba
def fields_get(self, allfields=None, attributes=None):
    res = super(ConsignmentAgreement, self).fields_get(allfields, attributes)
    if 'vendor_name' in res:
        res['vendor_name']['selection'] = self._compute_vendor_selection()
    return res
```

Gambar 3.23. *Code Draft* dan *Compute Vendor*

C Code Validate Consignment Agreement

Method ini hanya berfungsi untuk mengubah *state* dari *draft* menjadi *validate* ketika tombol *validate* ditekan. Potongan *code validate consignment agreement* dapat dilihat pada Gambar 3.24.

```
def action_validate(self):  William Purba *
    self.write({
        'state': 'validate',
    })
    return True
```

Gambar 3.24. *Code Validate Consignment Agreement*

D Code Update Agreement

Pada *method* ini, berfungsi untuk menampilkan teks *updated* pada *field status update* ketika tombol *update* ditekan. Lalu, ketika tombol ditekan maka *state* akan berubah. Potongan *code update agreement* dapat dilihat pada Gambar 3.25

```
def action_update(self):  William Purba
    status_update = f"<span style='color:green; font-weight:bold;'>Updated</span>"
    self.write({
        'status': f"{status_update}",
        'state': 'update',
    })
    return True
```

Gambar 3.25. *Code Update Agreement*

3.6.3 Code Modul Training

Dalam modul *training*, terdapat beberapa *code* yang menjadi dasar berjalannya sistem baik untuk keseluruhan modul *training* maupun masing-masing menu dalam modul *training*. Beberapa *code* yang menjadi *code* penting dalam modul *training* adalah sebagai berikut.

A Code Training Module

Pada potongan *code* ini, merupakan dasar untuk membuat menu *training module* yang berisi *editable tree view*. Pada Gambar 3.26 adalah tampilan dari potongan *code training module*

```
class TrainingModule(models.Model):  William Purba
    _name = 'training.module'
    _description = 'Training Module'
    _rec_name = 'erp_name'

    erp_name = fields.Char(string="ERP Name", store=True)
    erp_code = fields.Char(string='ERP Code', store=True, size=3)
    module = fields.Char(string='Module', store=True)
```

Gambar 3.26. *Code Training Module*

B Code Training Code dan Training Code Line

Pada *class TrainingCode* berfungsi sebagai isi dari *form* sehingga *record* dapat dibuat sesuai data yang diinginkan. Lalu, pada *class TrainingCodeLine* berfungsi supaya *form* dapat memiliki data yang bisa di-*edit* pada *notebook*. Potongan *code TrainingCode* dapat dilihat pada Gambar 3.27, sedangkan *code TrainingCodeLine* dapat dilihat pada Gambar 3.28.

```
class TrainingCode(models.Model):  2 usages  William Purba *
    _name = 'training.code'
    _description = 'Training Code'

    name = fields.Char(string="Name", readonly=True, store=True)
    erp_name = fields.Many2one('training.module', string="ERP Name",store=True)
    erp_code = fields.Char(string="ERP Code",store=True, readonly=True)
    module_name = fields.Selection(selection=lambda self: self.get_module_selection(),
                                   string="Module Name", store=True)
    module_code = fields.Char(string="Module Code",required=True, store=True)
    train_code = fields.Char(string="Training Code", store=True, readonly=True)

    training_code_line_ids = fields.One2many('training.code.line', 'training_code_id',
                                             ondelete="cascade")
```

Gambar 3.27. *Code Training Code*

```
class TrainingCodeLine(models.Model):  1 usage  William Purba
    _name = 'training.code.line'
    _description = 'Training Code Line'

    training_code_id = fields.Many2one('training.code')
    sub_module = fields.Char(string="Sub-module", store=True)
    scenario = fields.Char(string="Scenario", store=True)
    train_code = fields.Char(string="Training Code", store=True)
    models = fields.Selection(selection="_get_module", string="Module", required=True)
    reference = fields.Many2one('training.tools',string="Ref.")
```

Gambar 3.28. *Code Training Code Line*

C Code Training Task dan Training Task Line

Code Training Task dan *Training Task Line* secara struktur tidak jauh berbeda dengan *Training Code* dan *Training Code Line*, yang menjadi pembeda hanyalah *field* yang ditampilkan beserta fungsinya. Gambar 3.29 merupakan

potongan *code TrainingInput*, sedangkan Gambar 3.30 merupakan potongan *code TrainingInputLine*.

```
class TrainingInput(models.Model):  3 usages  William Purba +1 *
    _name = 'training.input'
    _description = 'Input Training'

    name = fields.Char(string="Input Code", readonly=True, copy=False, default="New")
    user_name = fields.Many2one('res.users',string="User Name",
                                default=lambda self: self.env.user,store=True,
                                readonly=True)
    user_role = fields.Char(string="User Role" ,compute="_compute_user_role",
                            store=True, readonly=True)
    branch_code = fields.Many2one('res.branch', string="Branch", store=True, readonly=True,
                                  default=lambda self: self.env.user.branch_id)
    train_code = fields.Selection(selection=lambda self: self._get_train_code_name(),
                                  string="Training Code", store=True)
    training_date = fields.Datetime(compute="_compute_training_date", string="Training Date"
    module_name = fields.Many2one('training.code', string="Module Name")
    submodule_name = fields.Char(string="Sub-Module Name")
    training_scenario = fields.Char(string="Training Scenario")
    state = fields.Selection(
        [('draft', 'Draft'), ('submit', 'Submit'), ('approved', 'Approved'), ('cancel', 'Can
        string="Status", track_visibility="onchange", default="draft")
    activity = fields.Selection([('trial', "Trial"),
                                 ('training', "Training"),
                                 ('testing', "Testing")], string="Activity Information"

    training_input_line_ids = fields.One2many('training.input.line', 'training_input_id',
                                              ondelete="cascade")
```

Gambar 3.29. *Code Training Task*

```python
class TrainingInputLine(models.Model):  2 usages  William Purba +1 *
    _name = 'training.input.line'
    _description = 'Input Training Line'

    topic_input = fields.Char(string="Topic Code")
    submodule_input = fields.Char(string="Sub-Module")
    scenario_input = fields.Char(string='Scenario')
    reference_input = fields.Char(string="Ref")
    data_input = fields.Char('Data Record')
    data_validation_input = fields.Char(compute='_onchange_data_input',
                                        string="Data Validation")
    attachment_id = fields.Binary(string="Attachment", attachment=True)
    training_input_id = fields.Many2one('training.input', readonly=True)
```

Gambar 3.30. *Code Training Task Line*

D Code Generate Data pada Training Task

Dalam *training task*, terdapat *code* yang dapat membantu dalam proses *training*. *Method* yang dipakai yaitu untuk mencari data pada *notebook* di *Training Code Line* sesuai dengan *train code* yang dipilih di menu *training code*. Lalu, ketika tombol *generate data* ditekan maka *training task line* akan mengambil seluruh data dari *training code line* sesuai dengan *train code* yang telah dipilih. Potongan *code generate data* dapat terlihat pada Gambar 3.31 dan Gambar 3.32.

```python
def action_generate_train_code(self):  William Purba
    self = self.sudo()
    """Mengambil data dari TrainingCodeLine dan memasukkan ke TrainingInputLine"""
    for record in self:
        if not record.train_code:
            raise UserError("Silakan pilih Training Code terlebih dahulu!")

        if not record.id:
            record.flush()
            if not record.id:
                raise UserError("Record belum tersimpan di database!")
```

Gambar 3.31. *Code Generate Data* 1

```
# Buat TrainingInputLine berdasarkan data TrainingCodeLine
for line in training_lines:
    self.env['training.input.line'].create({
        'training_input_id': record.id,
        'topic_input': line.train_code or 'Unknown',
        'submodule_input': line.sub_module or 'Unknown',
        'scenario_input': line.scenario or 'Unknown',
        'reference_input': line.reference.tools_name if line.reference else 'Unknown',
    })

# created_lines = self.env['training.input.line'].search([('training_input_id', '=', reco
self.env.cr.commit()
```

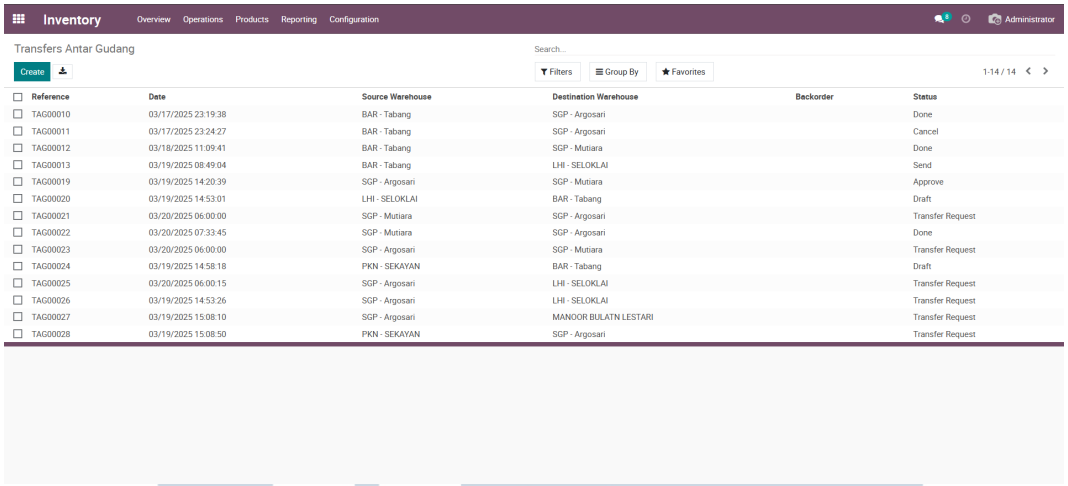
Gambar 3.32. *Code Generate Data* 2

3.7 Hasil Tampilan Sistem Odoo

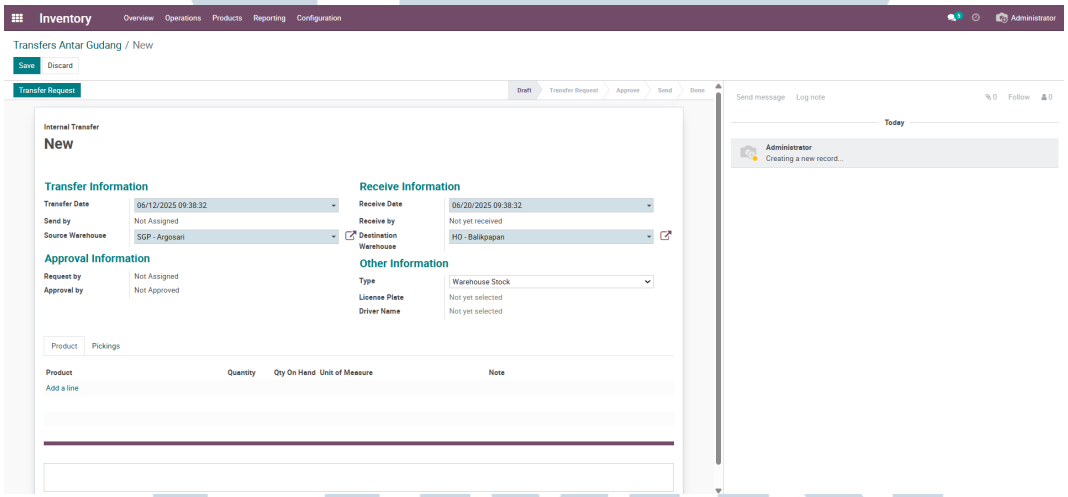
Tampilan sistem ini adalah hasil dari perancangan sistem yang telah dibangun sedemikian rupa untuk memenuhi kebutuhan klien. Hasil ini juga menjadi gambaran jelas dari *flowchart* yang telah disusun. Tampilan modul dan menu dengan segala fitur-fitur utama pada modul *training*, modul *consignment*, dan menu transfer antar gudang mencerminkan keberhasilan pengembangan sesuai dengan alur yang telah ditentukan.

3.7.1 Tampilan Transfer Antar Gudang

Tampilan awal transfer antar gudang yang merupakan *tree view* dapat dilihat pada Gambar 3.33. Sedangkan, *form view* dan fungsi menu transfer antar gudang dapat dilihat pada Gambar 3.34. Seluruh proses dan tampilan dibangun dengan *interface* yang *simple* dan mudah untuk dipakai sehingga dapat meningkatkan efisiensi penggunaan dan fungsionalitas.



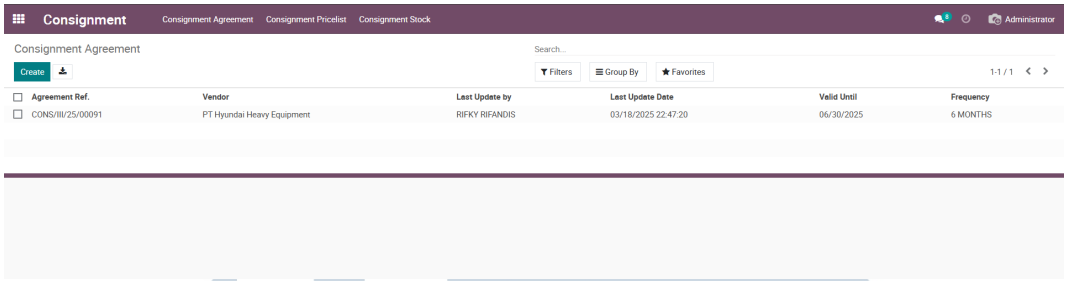
Gambar 3.33. Tampilan Awal Transfer Antar Gudang



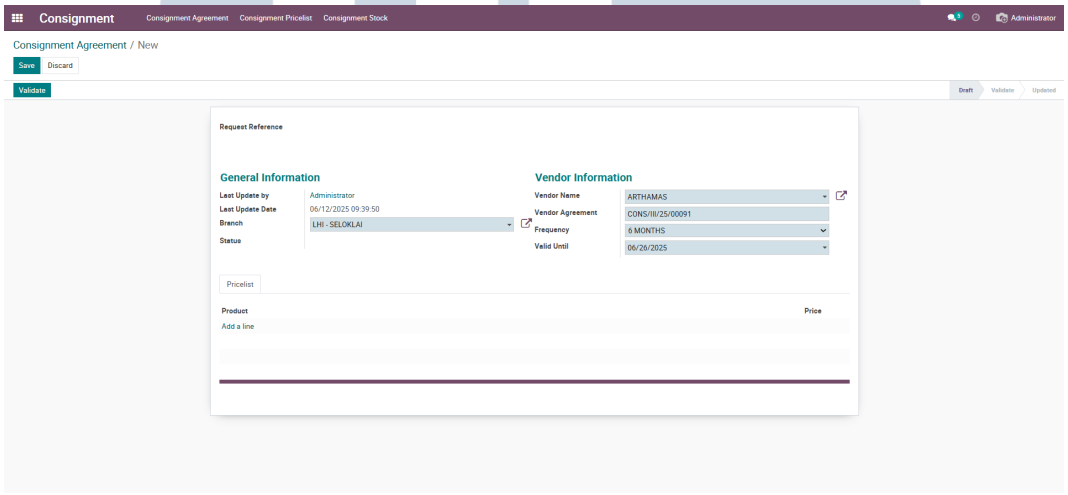
Gambar 3.34. *Form* Transfer Antar Gudang

3.7.2 Consignment Agreement

Tampilan *consignment agreement* terdapat tampilan awal (*tree view*) dan tampilan *form*. Pada Gambar 3.35 dapat terlihat ada tombol ”*Create*” sehingga *user* dapat dengan mudah membuat *record*. Lalu, pada Gambar 3.36 dapat terlihat *form consignment agreement* setelah tombol ”*Create*” ditekan.



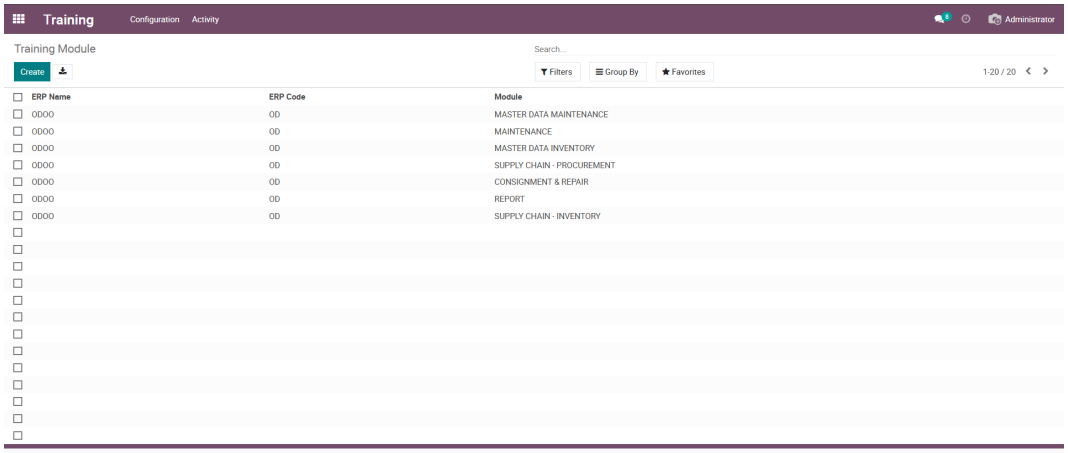
Gambar 3.35. Tampilan Awal *Consignment Agreement*



Gambar 3.36. *Form Consignment Agreement*

3.7.3 Training Module

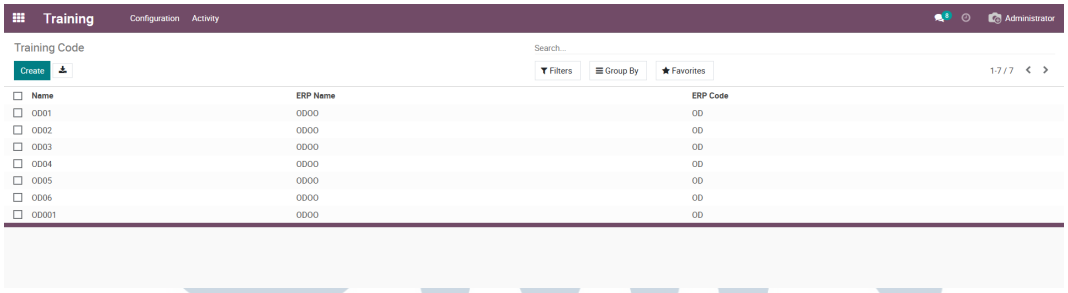
Tampilan *training module* berupa *editable tree view* sehingga dapat memudahkan dalam pembuatan *training module* seperti yang tertera pada Gambar 3.37.



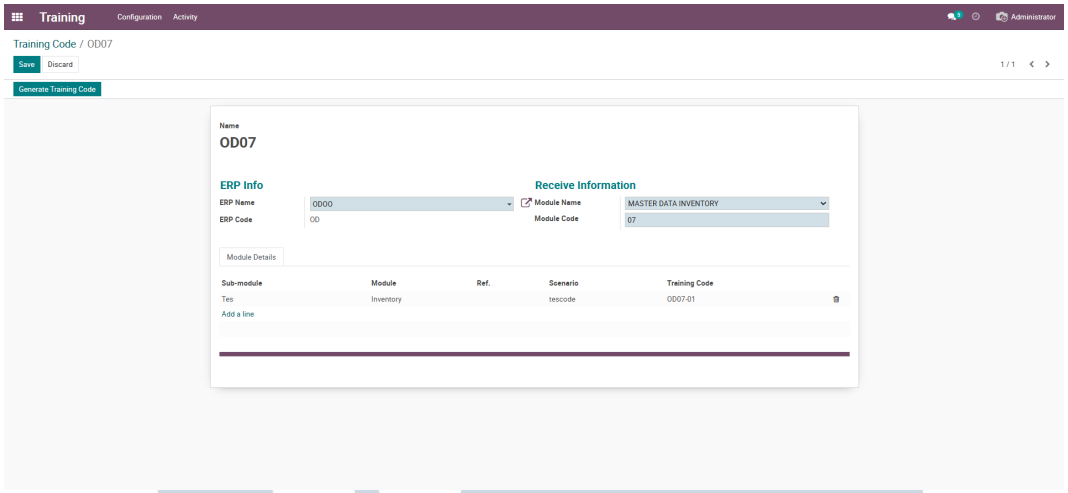
Gambar 3.37. Tampilan Menu *Training Module*

3.7.4 Training Code

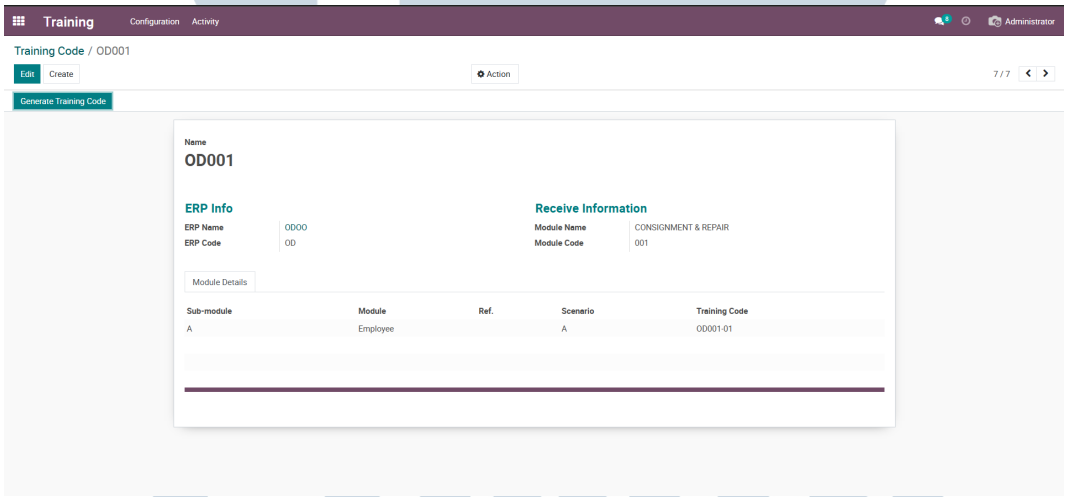
Pada *training code*, dibutuhkan *training module* dalam peng*input*an *field form*. *Tree view training code* dapat dilihat pada Gambar 3.38, sedangkan *Form view training code* dapat dilihat pada Gambar 3.39. Lalu, *Training module* yang dibuat yang bertujuan sebagai struktur dalam *sequence training code* dapat dilihat pada Gambar 3.40.



Gambar 3.38. Tampilan Awal *Training Code*



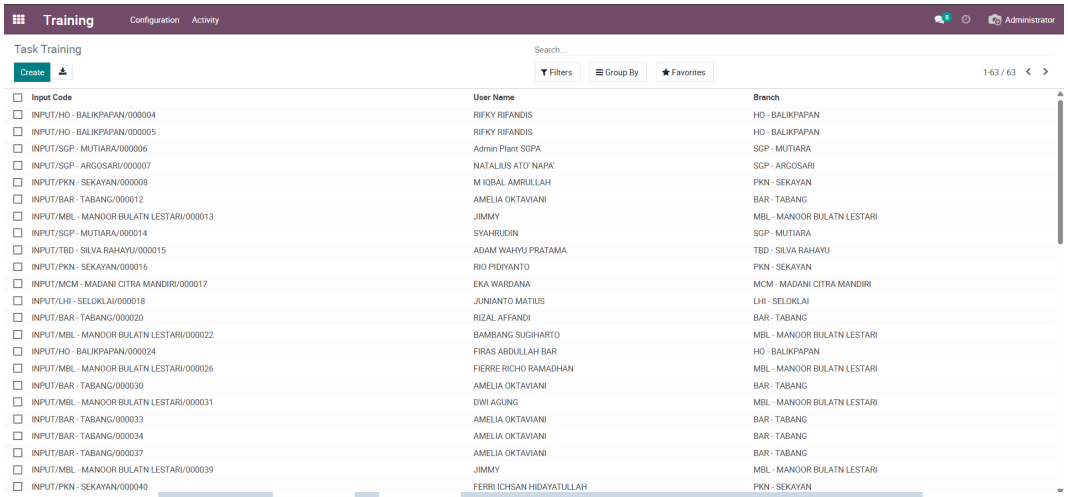
Gambar 3.39. *Form Training Code*



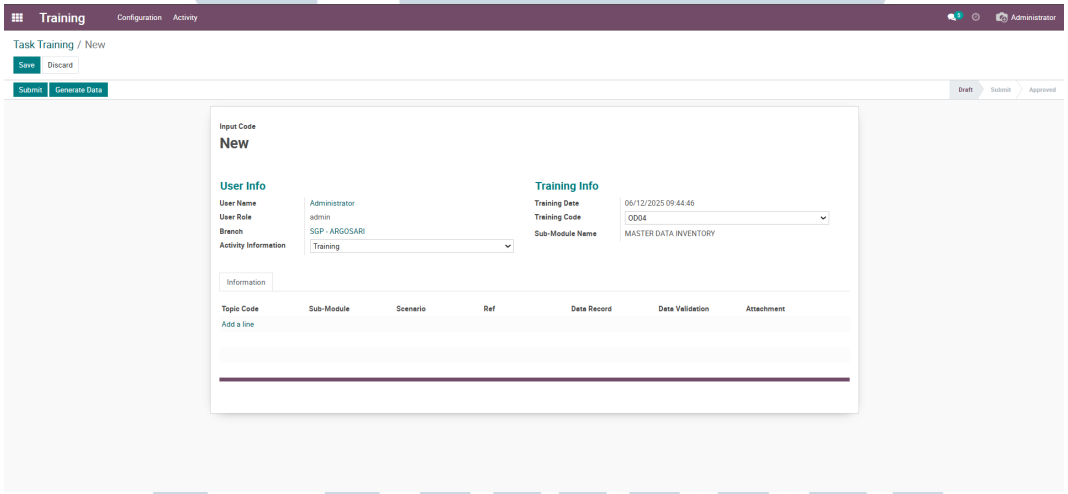
Gambar 3.40. *Form* Hasil *Training Code*

3.7.5 Training Task

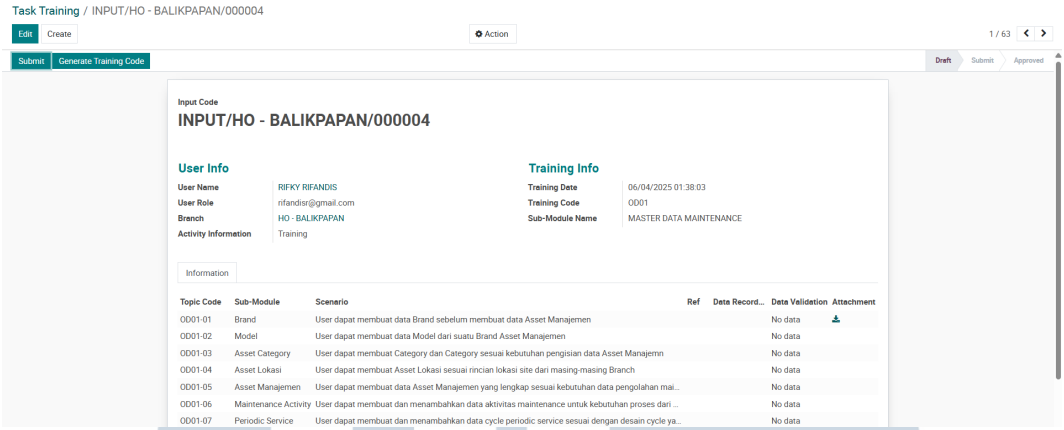
Dalam *training task*, terdapat *tree view* dengan tombol "*Create*" untuk membuat *record*, lalu ketika tombol tersebut ditekan maka *form view* akan terbuka. Dari seluruh *record* yang dibuat di *training module* dan *training code* akan berfungsi sebagai otomatisasi pembuatan data dalam *notebook* dalam *training task*. *Tree view training task* dapat dilihat pada Gambar 3.41. Sedangkan, proses *training task* ini akan terlihat pada Gambar 3.42 dan Gambar 3.43.



Gambar 3.41. Tampilan Awal *Training Task*



Gambar 3.42. *Form Training Task*



Gambar 3.43. *Form* Hasil *Training Task*

3.8 Kendala dan Solusi yang Ditemukan

Dalam menjalani magang sebagai Odoo *Developer*, terdapat 3 kendala utama yang ditemukan dari awal masa magang. Namun, seiring berjalannya waktu dan adaptasi yang dilakukan, maka terdapat juga solusi yang digunakan untuk bisa menyelesaikan kendala tersebut. Beberapa kendala yang dihadapi selama masa magang, yaitu:

- **Pemahaman Struktur Code**: mengalami kendala pada saat memulai pengerjaan proyek klien karena sudah terdapat struktur *code* yang sangat kompleks yang telah dibuat. Namun, ditemukan solusi dengan bertanya kepada *supervisor* dan anggota tim *technical* yang lain.
- **Konfigurasi dan Pemahaman Odoo**: terkendala pada saat belajar memahami tentang Odoo dan pada saat melakukan konfigurasi Odoo di minggu pertama magang karena berperan sebagai Odoo *Developer* merupakan hal yang sangat baru. Solusinya adalah untuk tetap berkomunikasi dengan *supervisor* dan anggota tim lain untuk bisa mengarahkan dalam proses konfigurasi Odoo.
- **Arahan Backlog yang Kurang Detil**: ditemukan *backlog* yang kurang detil dari *Project Manager*, tetapi langsung mendiskusikan tentang *backlog* yang diberikan oleh PM supaya tidak ada kesalahpahaman dalam pembuatan fitur dan proses yang harus dibuat.