BAB 3
PELAKSANAAN KERJA MAGANG

3.1 Kedudukan dan Organisasi

Seperti yang terlihat pada Gambar 2.1, HashMicro membagi developernya ke dalam 2 (dua) bagian yaitu *Project* dan *Product*. *Project* Developer bertugas untuk mengerjakan *task* berupa kustomisasi modul sesuai dengan *project* perusahaan yang dikerjakan dari *Consultants*. Sedangkan, *Product* Developer mengerjakan *task* berupa pengembangan modul sesuai dengan tim modul masing-masing dari *System Analysts*. Pembagian developer untuk masuk ke dalam tim *project* ataupun *product* dilakukan dengan melihat hasil dari tugas akhir yang diberikan setelah selesai menjalankan *training* selama kurang lebih 2 (dua) bulan.

Setelah melalui *training* dan mengerjakan tugas akhir yang diberikan, penulis dinyatakan lulus *training* dan berkesempatan untuk masuk ke dalam salah satu tim developer tersebut yaitu tim *product* atau lebih tepatnya *Product Construction* sebagai *Software Programmer* (Developer). Dalam menjalankan pekerjaannya, penulis disupervisi oleh 2 (dua) *System Analysts* yaitu Kak Abby Subhansyah dan Kak Elwin Waruwu di tim *Product Construction* serta dimentori oleh Kak Muhamad Bilal. Dikarenakan pada tim *Product Construction* tidak memiliki *Product Owner* untuk saat ini sehingga *System Analysts* juga turut berperan sebagai *Product Owner* pada tim ini.

Komunikasi yang dilakukan selama magang berlangsung menggunakan aplikasi media sosial Whatsapp dan Skype. Whatsapp biasanya lebih digunakan untuk keperluan umum seperti pembagian informasi dari kampus merdeka, pengumuman dari perusahaan, *mentoring* secara berkala oleh mentor, dan segala sesuatu lainnya yang menyangkut kepentingan *intern* di perusahaan. Sedangkan, Skype digunakan sebagai media komunikasi untuk pembagian *task* atau pembahasan mengenai pekerjaan sesuai dengan timnya masing-masing.

Task yang diberikan oleh *System Analysts* dapat berbentuk secara lisan (melalui pesan/*chat* secara langsung) atau melalui dokumen (MS Word) yang sudah berisikan semua informasi dan keterangan yang dibutuhkan dalam mengerjakan *task* tersebut. Setelah selesai mengerjakan *task*, penulis dapat mengirimkan hasil pekerjaannya kepada *System Analysts* dengan 2 (dua) cara yaitu mengirimkan *file* yang telah dikerjakan secara langsung melalui pesan/*chat* (dikarenakan terdapat dua

orang atau lebih yang mengerjakan *file* yang sama sehingga dapat menyebabkan konflik nantinya) atau melakukan *push file* melalui Github Desktop.

3.2 Tugas yang Dilakukan

Tugas utama yang dilakukan penulis selama melakukan magang di HashMicro adalah menyelesaikan *task* yang diberikan oleh *System Analysts*, baik berupa *task* yang diberikan secara lisan maupun melalui dokumen. *Task* yang diberikan berhubungan dengan pengembangan salah satu modul *product* HashMicro yaitu *Construction*. Modul *Construction* saat ini terbagi menjadi 4 (empat) modul yaitu *Project Construction*, *Sales Construction*, *Purchase Construction*, dan *Accounting Construction*. Awalnya penulis hanya diberikan *task* yang berhubungan dengan modul *Project Construction*. Namun seiring dengan berjalannya waktu dan dikarenakan terdapat tuntutan *task* yang lebih prioritas, penulis juga ikut mengerjakan *task* modul lain di luar dari *Project Construction*.

Sesuai dengan arti dari nama bahasa inggrisnya, modul *Construction* merupakan modul yang digunakan untuk mengatur konstruksi atau pembangunan suatu proyek. Proyek yang akan dikerjakan merupakan titik utama (*highlight*) dari modul ini. Modul ini juga mengatur segala keperluan yang dibutuhkan dalam perencanaan dan perealisasian terhadap setiap proyek seperti penetapan *budget* terhadap bahan baku yang diperlukan atau tenaga kerja yang dibutuhkan, perencanaan daftar pekerjaan pembangunan yang akan dilakukan, daftar dan perbandingan vendor-vendor kontraktor yang akan digunakan, dan sebagainya.

Pengembangan modul *Construction* menggunakan *framework* Odoo 14 dengan bahasa pemrograman Python (*back-end*) dan XML (*front-end*). Python biasanya digunakan untuk menambahkan variabel baru, mengatur fungsi-fungsi (*function*) terhadap suatu variabel, serta memasukkan dan menarik data dari *database*. Sedangkan, XML digunakan untuk menampilkan variabel agar dapat dilihat oleh mata telanjang pengguna. Untuk melakukan modifikasi pada tampilan XML seperti mengubah ukuran dapat dibantu dengan menggunakan SCSS (*Sassy* CSS). Selain itu, Odoo menggunakan PostgreSQL sebagai *database*-nya secara *default* yang didapatkan ketika melakukan penginstalan Odoo untuk pertama kalinya. Hal ini dikarenakan Odoo akan memasukkan setiap data ke dalam *database* secara langsung dengan menggunakan *syntax* python yang telah disediakan.

Tugas-tugas yang dikerjakan oleh penulis selama berada di tim *Construction* yaitu melakukan pengembangan terhadap modul *Construction* seperti pengubahan

dan penambahan fungsi-fungsi (*function*) serta tampilan yang akan dijelaskan pada bagian 3.3.

3.3 Uraian Pelaksanaan Magang

Pelaksanaan kerja magang yang dijalankan selama 5 (lima) bulan di HashMicro mulai dari tanggal 01 Februari 2022 sampai dengan tanggal 30 Juni 2022 dibagi ke dalam 2 (dua) bagian yang dijelaskan sebagai berikut.

3.3.1 *Training*

Setiap *intern* yang bekerja di HashMicro wajib untuk mengikuti *training* terlebih dahulu sebelum bekerja di lapangan, khususnya untuk divisi *Software Programmer*. Intern yang berada di divisi *Software Programmer* wajib untuk mengikuti 2 (dua) rangkaian *training* yang diberikan oleh HashMicro yaitu *training* umum dan *training* divisi.

A. *Training* Umum

Training umum merupakan *training* yang dihadiri oleh seluruh *intern* yang bekerja di HashMicro. *Training* ini bertujuan untuk memperkenalkan alur dan penggunaan modul-modul yang dimiliki oleh HashMicro kepada *intern* agar memudahkan mereka dalam pelaksanaan dan pengerjaan magang untuk kedepannya. *Training* ini berlangsung selama 2 (dua) minggu dari tanggal 07 Februari 2022 sampai dengan tanggal 17 Februari 2022. Terdapat total 8 (delapan) modul yang diperkenalkan satu setiap harinya selama *training* ini. Modul-modul tersebut adalah *Purchase*, *Inventory*, *Sales*, *Accounting*, POS (*Point of Sales*), *Manufacture*, *Construction*, dan HRM (*Human Resource Management*). Selain itu, *training* umum ini juga membahas mengenai *basic knowledge and general features* yang terdapat pada seluruh modul sebagai pengetahuan dasar dalam pengembangan modul. *Training* ini diselenggarakan secara *online* dengan menggunakan Zoom Meeting. Dikarenakan *training* ini dilakukan secara *online*, media absensi yang digunakan adalah mengisi nama dan jabatan pada Google Spreadsheet yang disediakan serta mengerjakan kuis melalui tautan yang diberikan setelah sesi *training* berakhir. Kuis yang diberikan merupakan pertanyaan yang berhubungan dengan sesi *training* yang diselenggarakan.

B. *Training* Divisi

Training divisi merupakan *training* yang diberikan khusus untuk *intern* yang berada di divisi *Software Programmer*. *Training* ini bertujuan untuk memperkenalkan *framework* dan bahasa pemrograman yang akan digunakan dalam pelaksanaan kerja magang kedepannya. *Training* ini berlangsung selama 2 (dua) bulan mulai dari tanggal 04 Februari 2022 sampai dengan tanggal 30 Maret 2022 dengan libur di setiap hari Kamisnya oleh Bapak Wiku Hermawan. *Framework* yang dipelajari adalah Odoo 14 dan bahasa pemrogramannya adalah Python. Selain itu, *training* ini juga membahas mengenai Git dan Github yang berguna dalam memudahkan kolaborasi untuk pengembangan suatu proyek. *Training* ini juga diselenggarakan secara *online* dengan menggunakan Zoom Meeting. Puncak akhir dari *training* ini adalah pengerjaan tugas akhir *training* dimana hasil dari tugas ini nantinya akan digunakan sebagai acuan dalam penempatan tim selama pelaksanaan kerja magang di HashMicro berlangsung. Adapun uraian pelaksanaan *training* divisi dijelaskan pada Tabel 3.1.

Tabel 3.1. Pelaksanaan *Training*

Minggu Ke -	Pekerjaan yang dilakukan
1	- Mengikuti kegiatan *Onboarding* perusahaan - Mengerjakan HackerRank - Melakukan penginstalan *software-software* yang digunakan - Mengerjakan survei kebinekaan - Mengerjakan kuis melalui Google Form Quiz
2	- Mempelajari modul *Purchase, Inventory, Sales, Accounting,* dan POS (*Point of Sales*) - Mempelajari Python dan Git - Mengikuti kegiatan National *Onboarding*
3	- Mempelajari modul *Manufacture, Construction,* HRM (*Human Resource Management*), dan *Basic Knowledge and General Features* - Mempelajari Python, Git, dan Github
4	- Mempelajari Python, Git, Github, dan Odoo 14
5 - 9	- Mempelajari Odoo 14
8 - 9	- Mengerjakan tugas akhir *training*

Pada minggu pertama, diadakan kegiatan *onboarding* yang diselenggarakan oleh HashMicro secara *online* melalui Zoom Meeting pada tanggal 02 Februari 2022 sebagai acara penyambutan seluruh *intern* dan pengenalan perusahaan mulai dari penjelasan mengenai latar belakang perusahaan, peraturan-peraturan yang terdapat di perusahaan, sampai dengan pengenalan para mentor serta penjelasan mengenai *jobdesk* dari divisi yang diambil. Kemudian, mengerjakan tes pertama melalui HackerRank untuk menguji tingkat pemahaman dasar mengenai algoritma menggunakan bahasa pemrograman Python pada tanggal 03 Februari 2022. Lalu, pada tanggal 04 Februari 2022 diselenggarakan *training* divisi yang membahas mengenai pengenalan awal dan penginstalan *software-software* yang akan digunakan selama pelaksanaan magang di HashMicro yaitu Odoo 14, Python, dan PostgreSQL. Selain itu, juga mengerjakan survei kebinekaan dari Kemendikbudristek RI melalui aplikasi Pusmenjar yang wajib diunduh terlebih dahulu pada perangkat komputer masing-masing dan dikerjakan selama 30 menit. Minggu pertama ini diakhiri dengan pengerjaan kuis melalui Google Form Quiz yang diselenggarakan untuk menguji pemahaman mengenai bahasa pemrograman Python.

Pada minggu kedua, diselenggarakan *training* umum yang mempelajari 5 (lima) modul HashMicro yaitu *Purchase, Inventory, Sales, Accounting,* dan POS (*Point of Sales*) selama 5 (lima) hari dengan satu modul setiap harinya dari tanggal 07 Februari 2022 sampai tanggal 11 Februari 2022. Berdampingan dengan *training* umum yang dilaksanakan pada pagi hari, terdapat juga *training* divisi yang diselenggarakan pada siang hari. *Training* divisi pada minggu ini membahas mengenai bahasa pemrograman Python mencakup pengenalan awal Python, *function, output, class* (*object oriented*), *access modifier,* MRO (*Method Resolution Order*), *instant method, class method,* dan *static method* serta Git yang mencakup *add, commit, config, checkout, reset, branch, switch, merge,* dan konflik. Selain itu, pada tanggal 10 Februari 2022 diadakan kegiatan National *Onboarding* yang diselenggarakan oleh Kemendikbudristek RI secara *online* melalui platform Youtube. Kegiatan ini ditujukan sebagai acara penyambutan seluruh mahasiswa yang mengikuti kegiatan Kampus Merdeka angkatan kedua baik yang mengambil jalur magang ataupun SIB (Studi Independen Bersertifikat).

Pada minggu ketiga, diselenggarakan *training* umum yang mempelajari 3 (tiga) modul HashMicro yaitu *Manufacture, Construction,* dan HRM (*Human Resource Management*) selama 3 (tiga) hari dengan satu modul setiap harinya dari tanggal 14 Februari 2022 sampai tanggal 16 Februari 2022. Selain itu, diadakan

juga *training* umum yang membahas mengenai *Basic Knowledge and General Features* selama 2 (dua) hari pada tanggal 14 Februari 2022 dan 17 Februari 2022. Kemudian, *training* divisi pada minggu ini diadakan selama 2 (dua) hari pada tanggal 15 Februari 2022 dan 16 Februari 2022. *Training* divisi membahas mengenai Python (lanjutan pembahasan *class method* dan *instant method* serta *lambda*), Git (*log, clone, push*), dan penggunaan Github.

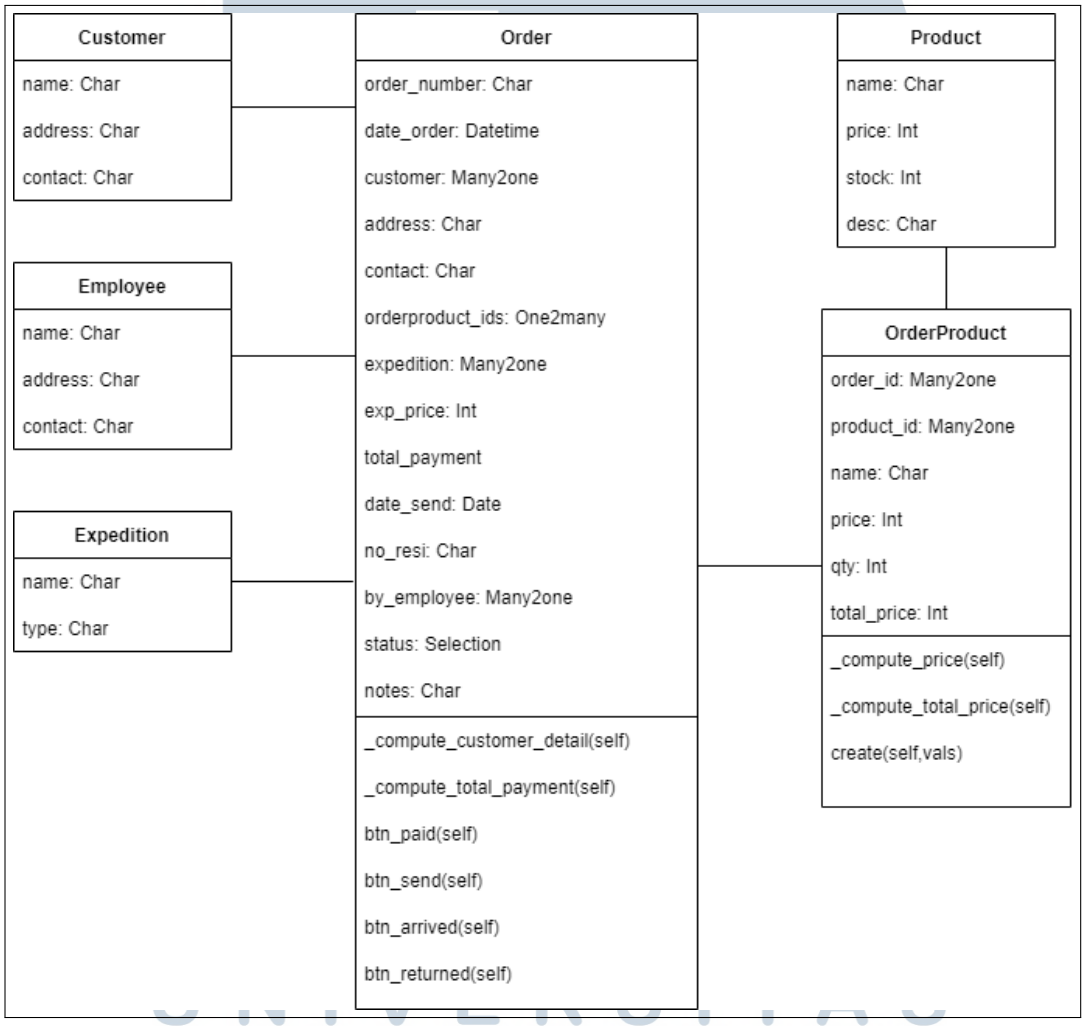
Pada minggu keempat, diselenggarakan *training* divisi dari tanggal 21 Februari 2022 sampai tanggal 25 Februari 2022. *Training* pada minggu ini membahas mengenai Python (*exceptions*), Git, Github, dan Odoo 14 yang mencakup pengenalan, penginstalan, dan konfigurasi awal serta pembuatan *custom* modul baru di Odoo 14. Mulai dari minggu ini, *intern* di divisi *Software Programmer* mulai berkenalan dengan *framework* Odoo dan mempelajari bagaimana caranya menambahkan modul baru, *object* (*class*) baru, *fields* baru, mengatur akses *security*, *init*, dan *manifest*, serta menampilkan *fields* yang terdapat di dalam *object* agar dapat dilihat dan dioperasikan (diinput).

Dari minggu kelima sampai minggu kesembilan, diselenggarakan *training* divisi dari tanggal 01 Maret 2022 sampai dengan 30 Maret 2022 yang membahas mengenai Odoo 14. *Training* yang diadakan selama 5 (lima) minggu ini membahas kelanjutan dari pembuatan *custom* modul baru yang telah dibuat pada minggu-minggu sebelumnya yaitu penjelasan mengenai *fields* baru yang tidak sering dijumpai seperti *many2one, one2many,* dan *many2many*, penggunaan bahasa, tanggal, dan waktu, cara melakukan *inherit* pada *object*, membuat *report* ke dalam bentuk PDF (membuat *button* yang apabila ditekan akan melakukan pengunduhan *file* berupa PDF, biasanya digunakan untuk mencetak faktur atau bon), dan sebagainya. Selain itu, juga mempelajari cara menggunakan *controller* untuk mendapatkan data dalam bentuk JSON yang dibantu dengan melakukan pengecekan melalui Postman.

Pada minggu kedelapan sampai minggu kesembilan, penulis mengerjakan tugas akhir *training* sebagai syarat kelulusan *training* dengan waktu pengerjaan selama seminggu dari tanggal 23 Maret 2022 sampai dengan 29 Maret 2022. Tugas akhir yang diberikan adalah pembuatan *custom* modul baru pada Odoo 14 dengan tema bebas. Selain menjadi syarat kelulusan *training*, tugas akhir ini juga diadakan guna untuk mengulang kembali seluruh pelajaran yang telah dipelajari selama *training* dan melihat sejauh mana para peserta magang paham dengan *framework* Odoo setelah menjalankan *training* selama 2 (dua) bulan. Tugas akhir *training* yang dikerjakan oleh penulis akan dijelaskan pada bagian B.1.

B.1 Tugas Akhir *Training*

Pembuatan *custom* modul baru atau yang biasanya disebut sebagai modul *addons* yang dikerjakan oleh penulis adalah modul "*Online Shop*". Modul ini digunakan untuk membantu sistem rekapan perjualbelian toko *online*. Meskipun modul yang dihasilkan cukup sederhana tapi modul ini telah memenuhi kriteria utama dari tujuan utamanya dibangun.



Gambar 3.1. *Class Diagram* Modul *Online Shop*

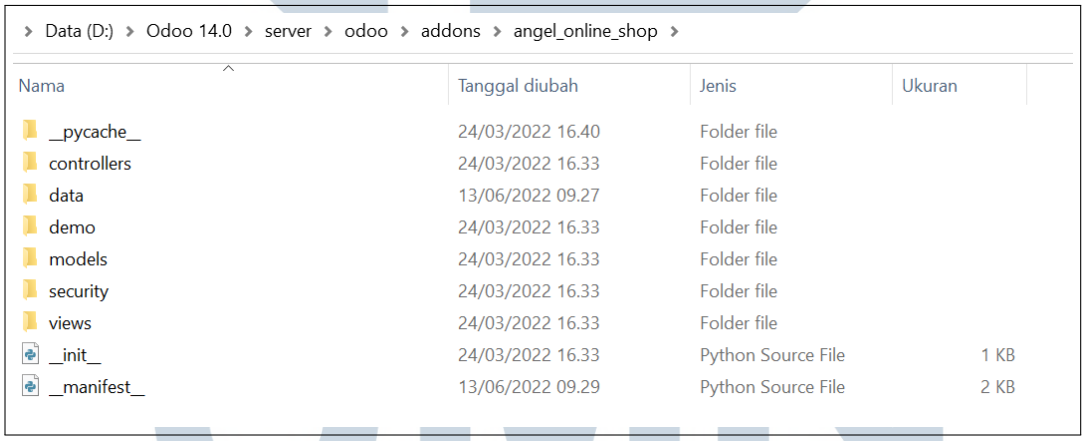
B.1.1 Pembuatan Modul

Untuk melakukan penambahan modul baru pada Odoo dapat dilakukan dengan menggunakan potongan kode pada Gambar 3.2.

```
D:\>"D:\Odoo 14.0\python\python.exe" "D:\Odoo 14.0\server\odoo-bin"
scaffold angel_booking_order "D:\Odoo 14.0\server\odoo\addons"
```

Gambar 3.2. Potongan Kode Menambahkan Modul Baru

Potongan kode pada Gambar 3.2 merupakan perintah untuk menambahkan modul *addons* baru ke dalam folder Odoo. Sesuai dengan urutannya, potongan kode tersebut dapat diterjemahkan menjadi `"dir Python" "dir odoo-bin" scaffold nama-modul-baru "dir addons"`. `dir` merupakan *directory* dari masing-masing tujuan yang dituju seperti `dir Python` yang berarti *directory* dari Python, `dir odoo-bin` berarti *directory* dari odoo-bin, dan `dir addons` berarti *directory* dimana modul *addons* baru tersebut akan diletakkan. Setelah `scaffold` berhasil dijalankan maka struktur (folder) modul *addons* baru tersebut berhasil dibuat seperti yang dapat dilihat pada Gambar 3.3.



Gambar 3.3. Struktur Modul *Addons* Baru

Folder *models* digunakan untuk menambahkan *object* (*class*) dengan ekstensi *file* berupa (.py). Folder *views* digunakan untuk menambahkan halaman tampilan dengan ekstensi *file* berupa (.xml). Folder *security* digunakan untuk mengatur akses atau *permission* dari setiap *object* yang ada. *File manifest* digunakan untuk mengatur *file* tampilan mana saja yang akan digunakan atau ditampilkan.

Sesuai dengan *class diagram* yang terdapat pada Gambar 3.1, berikut ini merupakan potongan kode Python dari setiap *object* yang dibuat pada modul ini.

customer.py 1

models > customer.py > ...

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


class Customer(models.Model):
    _name = 'os.customer'
    _description = 'New Description'

    name = fields.Char(string='Name')
    address = fields.Char(string='Address')
    contact = fields.Char(string='Contact')
```

Gambar 3.4. Potongan Kode Python Customer

Dari Gambar 3.4, variabel _name merupakan variabel yang menunjukkan nama *object* dari *class* tersebut yaitu os.customer. Sedangkan variabel name dibawahnya merupakan variabel biasa atau sebutannya di Odoo adalah *fields* yang berfungsi sebagai variabel untuk menyimpan data. *Fields* juga dapat digunakan untuk berbagai tipe data seperti *Char, Integer, Date, Boolean,* dan sebagainya. Selain itu, fungsi `string` yang terdapat pada *fields* digunakan untuk mengatur kata atau kalimat yang akan ditampilkan pada saat *fields* tersebut dipanggil.

employee.py 1

models > employee.py > ...

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


class Employee(models.Model):
    _name = 'os.employee'
    _description = 'New Description'

    name = fields.Char(string='Name')
    address = fields.Char(string='Address')
    contact = fields.Char(string='Contact')
```

Gambar 3.5. Potongan Kode Python Employee

product.py 1

models > product.py > ...

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


class Product(models.Model):
    _name = 'os.product'
    _description = 'New Description'

    name = fields.Char(string='Name', required=True)
    price = fields.Integer(string='Price', required=True)
    stock = fields.Integer(string='Stock', required=True)
    desc = fields.Char(string='Description')
```

Gambar 3.6. Potongan Kode Python Product

Dari Gambar 3.6, fungsi `required` pada *fields* digunakan untuk mengatur kewajiban pengisian *fields*. Jika fungsi `required` diberi nilai *True* maka *fields* tersebut wajib diisi pada saat penginputan data atau dengan kata lain *fields* tersebut tidak boleh kosong. Sebaliknya jika fungsi `required` pada *fields* tersebut diberi nilai *False* maka *fields* tersebut tidak wajib diisi pada saat penginputan data atau dengan kata lain *fields* tersebut diperbolehkan untuk kosong.

expedition.py 1

models > expedition.py > ...

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


class Expedition(models.Model):
    _name = 'os.expedition'
    _description = 'New Description'

    name = fields.Char(string='Name', required=True)
    type = fields.Char(string='Type')
```

Gambar 3.7. Potongan Kode Python Expedition

order.py 1, M

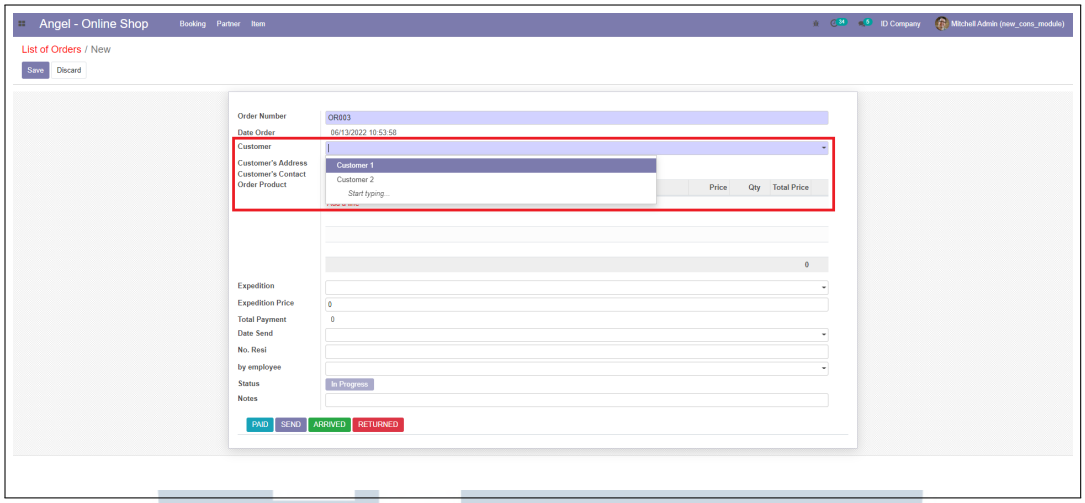
models > order.py > ...

```python
from asyncio.windows_events import NULL
from email.policy import default
from odoo import api, fields, models


class Order(models.Model):
    _name = 'os.order'
    _description = 'New Description'

    order_number = fields.Char(string='Order Number', required=True)
    date_order = fields.Datetime('Date Order',
                                 default=fields.Datetime.now(),
                                 readonly=True)

    customer = fields.Many2one(comodel_name='os.customer',
                               string='Customer',
                               required=True)
    address = fields.Char(string="Customer's Address",
                               required=True,
                               compute='_compute_customer_detail')
    contact = fields.Char(string="Customer's Contact",
                               required=True,
                               compute='_compute_customer_detail')
    @api.depends('customer')
    def _compute_customer_detail(self):
        for record in self:
            record.address = record.customer.address
            record.contact = record.customer.contact

    orderproduct_ids = fields.One2many(comodel_name='os.orderproduct',
                                       inverse_name='order_id',
                                       string='Order Product')

    expedition = fields.Many2one(comodel_name='os.expedition', string='Expedition')
    exp_price = fields.Integer(string='Expedition Price', default=0)
```

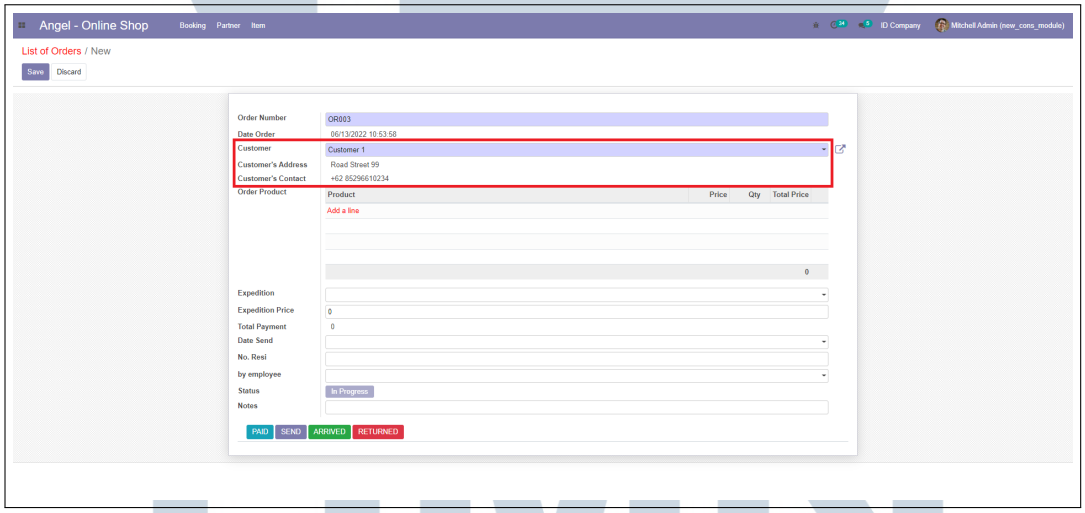
Gambar 3.8. Potongan Kode Python Order

Dari Gambar 3.8, fungsi `readonly` pada *fields* digunakan untuk membuat *fields* yang bersangkutan hanya dapat dibaca sehingga tidak dapat diubah apabila diberi nilai *True*. Fungsi `default` digunakan untuk menetapkan nilai atau data secara otomatis terhadap *fields* tersebut. Fungsi `compute` digunakan untuk memasukkan suatu *method* atau *function* ke dalam *fields*.

Tipe data *Many2one* digunakan untuk mengambil data dari banyak (*many*) ke satu (*one*). Contohnya pada *fields* customer:Many2one, melalui fungsi `comodel\underline{ }name` yang digunakan untuk menghubungkan relasi antara *fields* dengan target *object* yaitu os.customer, maka *fields* ini akan berisikan pilihan seluruh data yang ada di *object* os.customer seperti yang terlihat pada Gambar 3.9. Kemudian dari seluruh data tersebut dapat dipilih satu untuk dimasukkan ke dalam *fields* customer seperti yang terlihat pada Gambar 3.10.



Gambar 3.9. *Fields* Customer Sebelum Dipilih



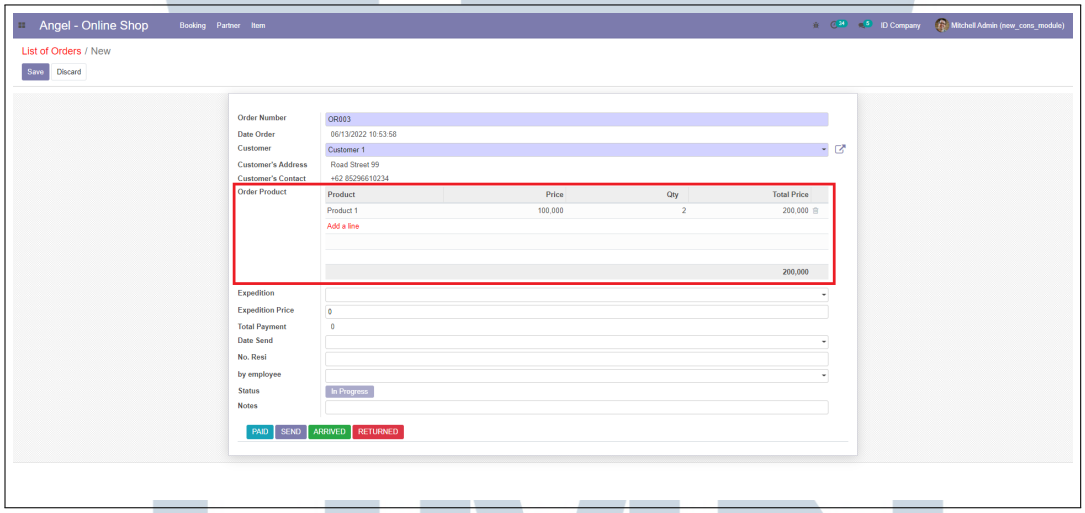
Gambar 3.10. *Fields* Customer Setelah Dipilih

Method _compute_customer_detail yang terdapat pada Gambar 3.8 merupakan *method* yang digunakan untuk mengisi *fields* address dan contact sesuai dengan address dan contact yang dimiliki oleh customer yang dipilih.

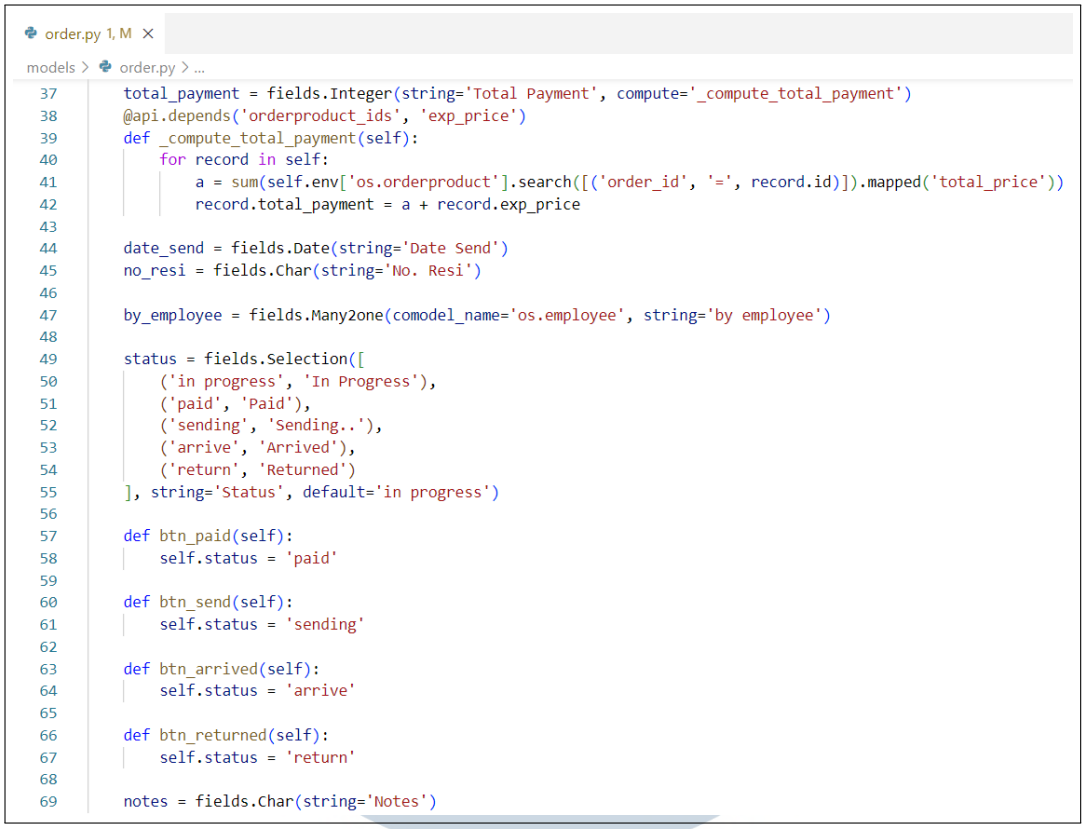
Tipe data *One2many* digunakan untuk memasukkan berbagai data (*many*) ke dalam satu *fields* (*one*). Berbeda dengan tipe data *Many2one* yang dapat langsung digunakan, tipe data *One2many* ini harus dibarengi dengan penggunaan *Many2one* dan pembuatan *object* baru. Hal ini dikarenakan *fields One2many* akan menampung sejumlah data untuk disimpan kedalamnya sehingga diperlukan pembuatan *object* baru sesuai dengan *fields* atau data yang diperlukan. Contoh penggunaan tipe data ini dapat

dilihat pada *fields* orderproduct_ids:One2many, *fields* ini dihubungkan dengan *object* os.orderproduct melalui fungsi `comodel\underline{ }name` dengan `inverse\underline{ }name` yaitu order_id. `inverse\underline{ }name` digunakan untuk menghubungkan relasi antara *fields* yang memiliki maksud yang sama namun kebalikannya (satunya *One2many* dan yang lainnya *Many2one*) pada *object* yang berbeda. Dengan kata lain, `inverse\underline{ }name` ini digunakan sebagai penghubung *id* dari satu *object* ke *object* lainnya.

Setelah *fields* orderproduct_ids terhubung dengan *object* os.orderproduct, maka setiap *fields* yang ada di *object* os.orderproduct seperti yang terlihat pada Gambar 3.13 akan dipanggil ke dalam *field* orderproduct_ids. Penginputan *fields* *One2many* ini dapat dilihat pada Gambar 3.11.



Gambar 3.11. *Fields* Order Product

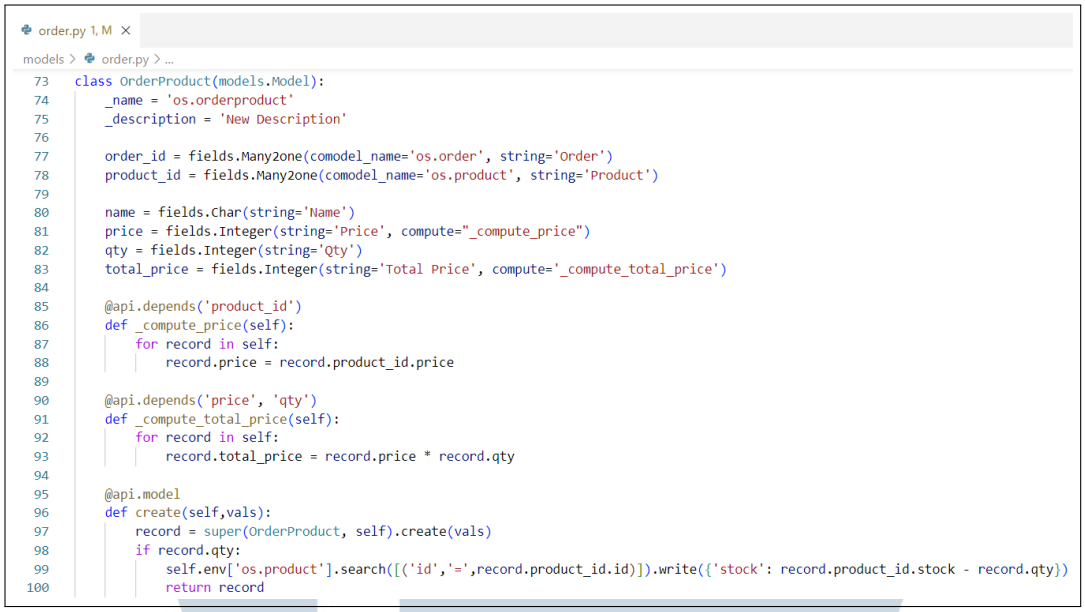


```
order.py 1, M
models > order.py > ...
37      total_payment = fields.Integer(string='Total Payment', compute='_compute_total_payment')
38      @api.depends('orderproduct_ids', 'exp_price')
39      def _compute_total_payment(self):
40          for record in self:
41              a = sum(self.env['os.orderproduct'].search([('order_id', '=', record.id)]).mapped('total_price'))
42              record.total_payment = a + record.exp_price
43
44      date_send = fields.Date(string='Date Send')
45      no_resi = fields.Char(string='No. Resi')
46
47      by_employee = fields.Many2one(comodel_name='os.employee', string='by employee')
48
49      status = fields.Selection([
50          ('in progress', 'In Progress'),
51          ('paid', 'Paid'),
52          ('sending', 'Sending..'),
53          ('arrive', 'Arrived'),
54          ('return', 'Returned')
55      ], string='Status', default='in progress')
56
57      def btn_paid(self):
58          self.status = 'paid'
59
60      def btn_send(self):
61          self.status = 'sending'
62
63      def btn_arrived(self):
64          self.status = 'arrive'
65
66      def btn_returned(self):
67          self.status = 'return'
68
69      notes = fields.Char(string='Notes')
```

Gambar 3.12. Lanjutan Potongan Kode Python Order

Dari Gambar 3.12, tipe data *selection* merupakan tipe data yang digunakan untuk membuat suatu *fields* memiliki beberapa pilihan data, biasanya digunakan pada *fields* yang mengatur status, tipe, ataupun kategori. Contoh penggunaan tipe data ini dapat dilihat pada *fields* status yang memiliki 5 (lima) pilihan data, dengan salah satu potongan kodenya yaitu `('in progress', 'In Progress')`. Sisi sebelah kiri `('in progress')` merupakan *value* dari pilihan tersebut yang dapat dipanggil pada *method*, sedangkan sisi sebelah kanan `('In Progress')` merupakan *string* yang akan ditampilkan.

Method _compute_total_payment adalah *method* untuk menambahkan total harga dari orderproduct_ids dengan harga dari ekspedisi. Sedangkan *method* btn_paid, btn_send, btn_arrived, dan btn_returned merupakan *method* pada masing-masing *button* yang berfungsi untuk mengubah *value* status sesuai dengan *button* yang dioperasikan.



```
order.py 1, M
models > order.py > ...
73  class OrderProduct(models.Model):
74      _name = 'os.orderproduct'
75      _description = 'New Description'
76
77      order_id = fields.Many2one(comodel_name='os.order', string='Order')
78      product_id = fields.Many2one(comodel_name='os.product', string='Product')
79
80      name = fields.Char(string='Name')
81      price = fields.Integer(string='Price', compute="_compute_price")
82      qty = fields.Integer(string='Qty')
83      total_price = fields.Integer(string='Total Price', compute='_compute_total_price')
84
85      @api.depends('product_id')
86      def _compute_price(self):
87          for record in self:
88              record.price = record.product_id.price
89
90      @api.depends('price', 'qty')
91      def _compute_total_price(self):
92          for record in self:
93              record.total_price = record.price * record.qty
94
95      @api.model
96      def create(self,vals):
97          record = super(OrderProduct, self).create(vals)
98          if record.qty:
99              self.env['os.product'].search([('id','=',record.product_id.id)]).write({'stock': record.product_id.stock - record.qty})
100             return record
```

Gambar 3.13. Potongan Kode Python OrderProduct

Dari Gambar 3.13, *method* _compute_price digunakan untuk menarik data harga sesuai dengan harga dari *fields* product yang dipilih. *Method* _compute_total_price digunakan untuk mengalikan harga product dengan kuantitas yang dimasukkan. *Method* yang terakhir merupakan *method* yang digunakan untuk mengurangi jumlah stok produk yang terdapat pada *object* os.product sesuai dengan kuantitas produk yang telah dimasukkan atau di *order*.

Kemudian untuk dapat menampilkan seluruh *object* yang telah dibuat sesuai dengan potongan kode diatas, maka diperlukan juga pembuatan *file* XML pada setiap *object* tersebut.

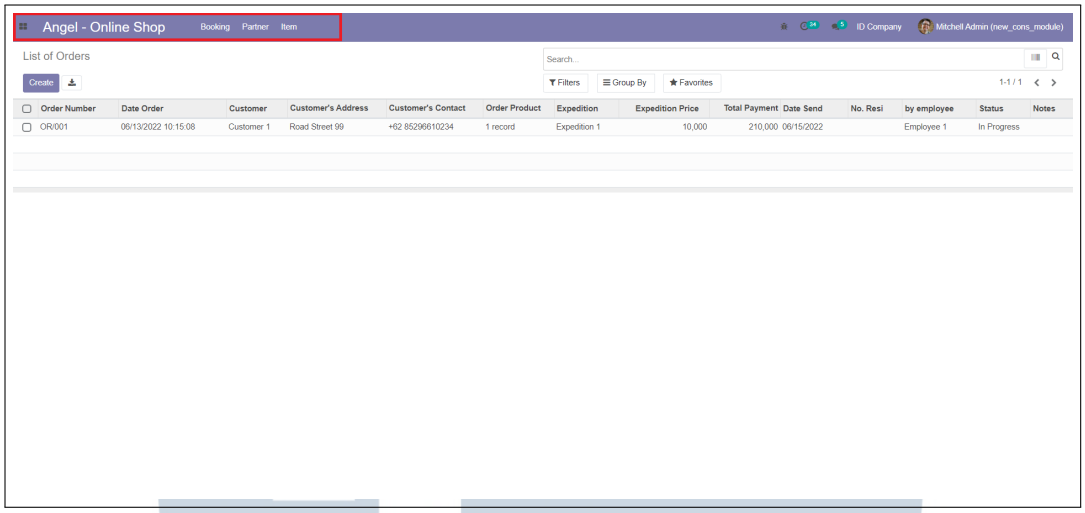
menu.xml M

views > menu.xml

```xml
<?xml version='1.0' encoding='utf-8'?>
<odoo>
    <data>
        <!-- This Menu Item will appear in the Upper bar, that's why It needs NO parent or action -->
        <menuitem
            id="os_menu_root"
            name="Angel - Online Shop"
            sequence="10"/>

        <!-- This Menu Item Must have a parent -->
        <menuitem
            id="os_item_menu_categ"
            name="Item"
            parent="os_menu_root"
            sequence="30"/>

        <!-- This Menu Item Must have a parent -->
        <menuitem
            id="os_partner_menu_categ"
            name="Partner"
            parent="os_menu_root"
            sequence="20"/>

        <!-- This Menu Item Must have a parent -->
        <menuitem
            id="os_booking_menu_categ"
            name="Booking"
            parent="os_menu_root"
            sequence="10"/>
    </data>
</odoo>
```

Gambar 3.14. Potongan Kode XML Menu

Potongan kode pada Gambar 3.14 merupakan potongan kode XML pada *file* menu.xml yang digunakan sebagai akar dari setiap tampilan yang akan dimunculkan. Pada *line* 5-8 merupakan potongan kode untuk mengatur *root* (akar) tampilan menu dasar. Kemudian *line* 11-15 merupakan potongan kode untuk membuat menu Item, line 18-22 merupakan potongan kode untuk membuat menu Partner dan *line* 25-29 merupakan potongan kode untuk membuat menu Booking. Ketiga *menuitem* tersebut (Item, Partner, dan Booking) berada dibawah menu *root* yang diatur melalui fungsi `parent` dan semakin kecil nilai dari `sequence` yang ada pada masing-masing *menuitem* maka semakin terdepan menu tersebut akan ditampilkan, baik berupa lebih atas ataupun lebih ke kiri. Gambar 3.15 berikut merupakan tampilan yang ditampilkan berdasarkan potongan kode tersebut.



Gambar 3.15. Tampilan Menu

customer_views.xml M

views > customer_views.xml

```xml
<?xml version='1.0' encoding='utf-8'?>
<odoo>
    <!-- os.customer tree view -->
    <record id="os_customer_view_tree" model="ir.ui.view">
        <field name="name">Customer</field>
        <field name="model">os.customer</field>
        <field name="arch" type="xml">
            <tree>
                <!-- Add your fields here -->
                <field name="name"/>
                <field name="address"/>
                <field name="contact"/>
            </tree>
        </field>
    </record>

    <!-- os.customer form view -->
    <record id="os_customer_view_form" model="ir.ui.view">
        <field name="name">Form Customer</field>
        <field name="model">os.customer</field>
        <field name="arch" type="xml">
            <form string="">
                <sheet>
                    <group>
                        <!-- Add your fields here -->
                        <field name="name"/>
                        <field name="address"/>
                        <field name="contact"/>
                    </group>
                </sheet>
            </form>
        </field>
    </record>

    <!-- os.customer action window -->
    <record id="os_customer_action" model="ir.actions.act_window">
        <field name="name">List of Customers</field>
        <field name="type">ir.actions.act_window</field>
        <field name="res_model">os.customer</field>
        <field name="view_mode">tree,form</field>
    </record>

    <!-- This Menu Item must have a parent and an action -->
    <menuitem
        id="os_customer_menu_act"
        name="Customers"
        parent="os_partner_menu_categ"
        action="os_customer_action"
        sequence="10"/>
</odoo>
```

Gambar 3.16. Potongan Kode XML Customer

Dari Gambar 3.16, *line* 44-49 merupakan potongan kode untuk mengatur dimana letak *file* tampilan atau *views* (customer.xml) ini akan ditampilkan yaitu di bawah menu Partner dengan nama Customer. Kemudian *views* ini memiliki *action* berupa *tree* dan *form* yang diatur pada *line* 36-41. *Tree* merupakan tampilan tabel pada Odoo yang digunakan untuk menampilkan data-data dari *database*.

Sedangkan, *form* merupakan tampilan formulir pada Odoo yang digunakan untuk memasukkan data ke dalam *database*.

employee_views.xml M

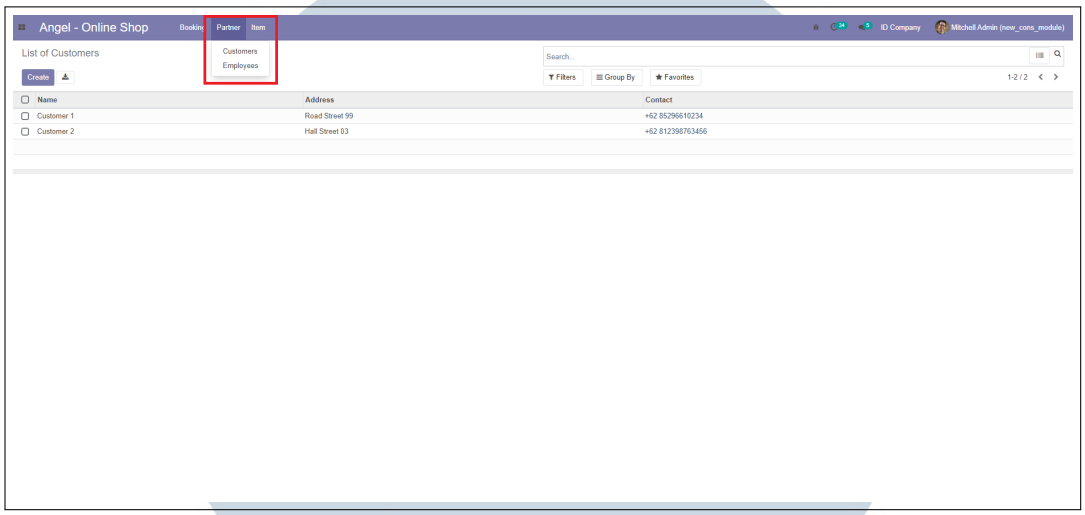
views > employee_views.xml

```xml
<?xml version='1.0' encoding='utf-8'?>
<odoo>
    <!-- os.employee tree view -->
    <record id="os_employee_view_tree" model="ir.ui.view">
        <field name="name">Employee</field>
        <field name="model">os.employee</field>
        <field name="arch" type="xml">
            <tree>
                <!-- Add your fields here -->
                <field name="name"/>
                <field name="address"/>
                <field name="contact"/>
            </tree>
        </field>
    </record>

    <!-- os.employee form view -->
    <record id="os_employee_view_form" model="ir.ui.view">
        <field name="name">Form Employee</field>
        <field name="model">os.employee</field>
        <field name="arch" type="xml">
            <form string="">
                <sheet>
                    <group>
                        <!-- Add your fields here -->
                        <field name="name"/>
                        <field name="address"/>
                        <field name="contact"/>
                    </group>
                </sheet>
            </form>
        </field>
    </record>

    <!-- os.employee action window -->
    <record id="os_employee_action" model="ir.actions.act_window">
        <field name="name">List of Employees</field>
        <field name="type">ir.actions.act_window</field>
        <field name="res_model">os.employee</field>
        <field name="view_mode">tree,form</field>
    </record>

    <!-- This Menu Item must have a parent and an action -->
    <menuitem
        id="os_employee_menu_act"
        name="Employees"
        parent="os_partner_menu_categ"
        action="os_employee_action"
        sequence="20"/>
</odoo>
```

Gambar 3.17. Potongan Kode XML Employee

Dari Gambar 3.17, *line* 44-49 menunjukkan bahwa *views* Employee ini juga berada dibawah menu Partner dengan nilai `sequence` yang lebih besar

dibandingkan dengan *views* Customer. Sehingga tampilan yang akan dihasilkan adalah menu Customer yang mendahului menu Employee seperti pada Gambar 3.18.



Gambar 3.18. Tampilan Menu Partner

product_views.xml M

views > product_views.xml

```xml
<?xml version='1.0' encoding='utf-8'?>
<odoo>
    <!-- os.product tree view -->
    <record id="os_product_view_tree" model="ir.ui.view">
        <field name="name">Products</field>
        <field name="model">os.product</field>
        <field name="arch" type="xml">
            <tree>
                <!-- Add your fields here -->
                <field name="name"/>
                <field name="price"/>
                <field name="stock"/>
                <field name="desc"/>
            </tree>
        </field>
    </record>

    <!-- os.product form view -->
    <record id="os_product_view_form" model="ir.ui.view">
        <field name="name">Form Product</field>
        <field name="model">os.product</field>
        <field name="arch" type="xml">
            <form string="">
                <sheet>
                    <group>
                        <!-- Add your fields here -->
                        <field name="name"/>
                        <field name="price"/>
                        <field name="stock"/>
                        <field name="desc"/>
                    </group>
                </sheet>
            </form>
        </field>
    </record>

    <!-- os.product action window -->
    <record id="os_product_action" model="ir.actions.act_window">
        <field name="name">List of Products</field>
        <field name="type">ir.actions.act_window</field>
        <field name="res_model">os.product</field>
        <field name="view_mode">tree,form</field>
    </record>

    <!-- This Menu Item must have a parent and an action -->
    <menuitem
        id="os_product_menu_act"
        name="Products"
        parent="os_item_menu_categ"
        action="os_product_action"
        sequence="10"/>
</odoo>
```

Gambar 3.19. Potongan Kode XML Product

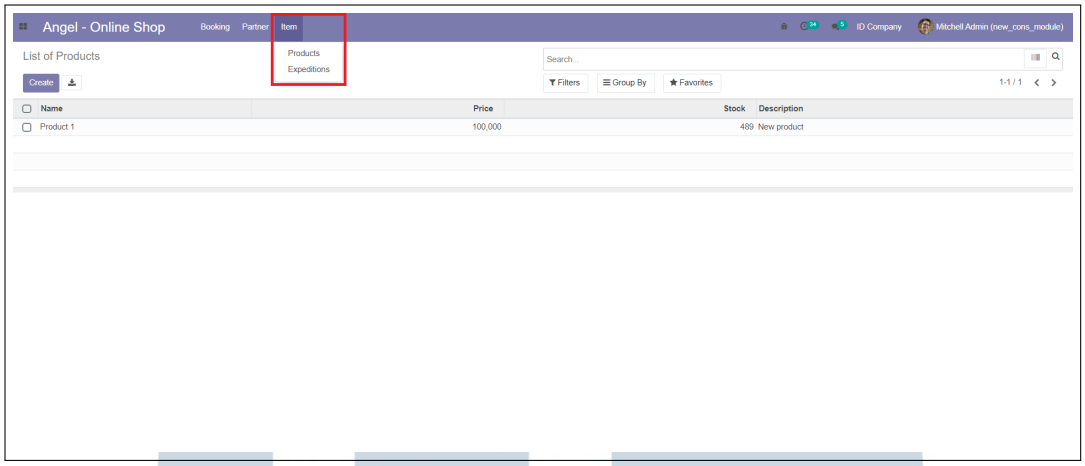
expedition_views.xml M

views > expedition_views.xml

```xml
<?xml version='1.0' encoding='utf-8'?>
<odoo>
    <!-- os.expedition tree view -->
    <record id="os_expedition_view_tree" model="ir.ui.view">
        <field name="name">Expedition</field>
        <field name="model">os.expedition</field>
        <field name="arch" type="xml">
            <tree>
                <!-- Add your fields here -->
                <field name="name"/>
                <field name="type"/>
            </tree>
        </field>
    </record>

    <!-- os.expedition form view -->
    <record id="os_expedition_view_form" model="ir.ui.view">
        <field name="name">Form Expedition</field>
        <field name="model">os.expedition</field>
        <field name="arch" type="xml">
            <form string="">
                <sheet>
                    <group>
                        <!-- Add your fields here -->
                        <field name="name"/>
                        <field name="type"/>
                    </group>
                </sheet>
            </form>
        </field>
    </record>

    <!-- os.expedition action window -->
    <record id="os_expedition_action" model="ir.actions.act_window">
        <field name="name">List of Expeditions</field>
        <field name="type">ir.actions.act_window</field>
        <field name="res_model">os.expedition</field>
        <field name="view_mode">tree,form</field>
    </record>

    <!-- This Menu Item must have a parent and an action -->
    <menuitem
        id="os_expedition_menu_act"
        name="Expeditions"
        parent="os_item_menu_categ"
        action="os_expedition_action"
        sequence="20"/>
</odoo>
```

Gambar 3.20. Potongan Kode XML Expedition

Dari Gambar 3.19 dan Gambar 3.20, *views* Product dan *views* Expedition memiliki menu `parent` yang sama yaitu menu Item. Dikarenakan *views* Product memiliki `sequence` yang lebih kecil dibandingkan dengan `sequence` pada *views* Expedition maka menu Product akan ditampilkan mendahului menu Expedition seperti pada Gambar 3.21.



Gambar 3.21. Tampilan Menu Item

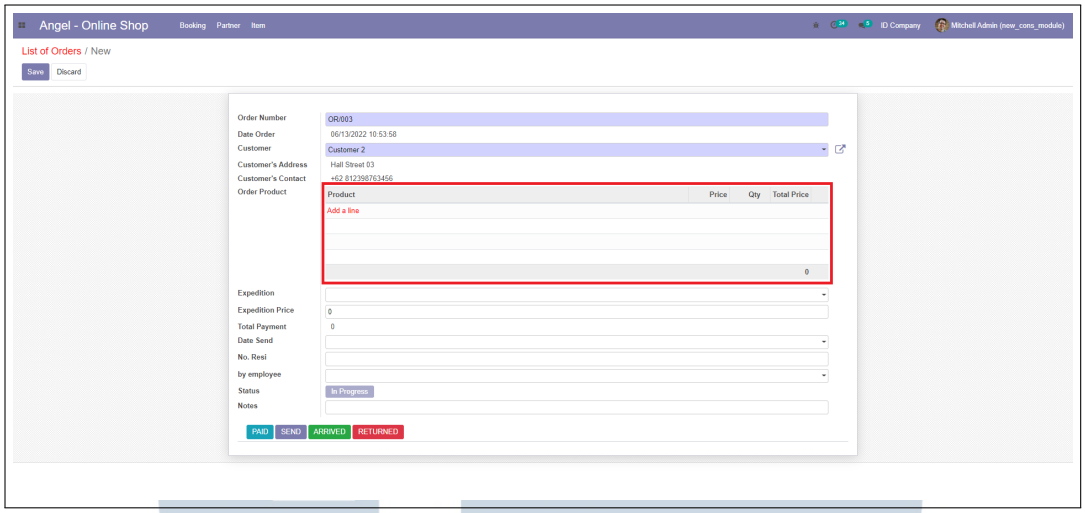
order_views.xml M

views > order_views.xml

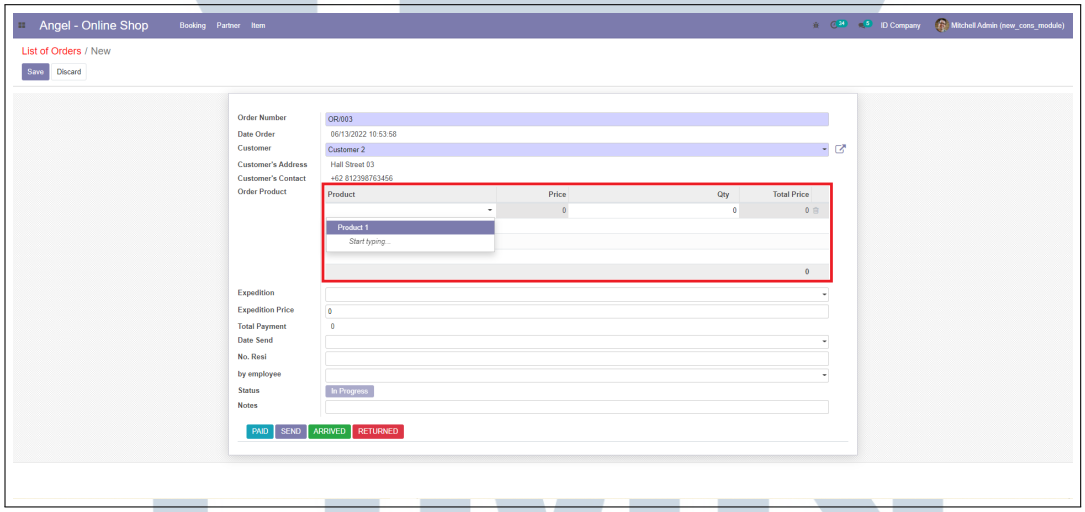
```xml
<?xml version='1.0' encoding='utf-8'?>
<odoo>
    <!-- os.order tree view -->
    <record id="os_order_view_tree" model="ir.ui.view">
        <field name="name">Order</field>
        <field name="model">os.order</field>
        <field name="arch" type="xml">
            <tree>
                <!-- Add your fields here -->
                <field name="order_number"/>
                <field name="date_order"/>
                <field name="customer"/>
                <field name="address"/>
                <field name="contact"/>
                <field name="orderproduct_ids"/>
                <field name="expedition"/>
                <field name="exp_price"/>
                <field name="total_payment"/>
                <field name="date_send"/>
                <field name="no_resi"/>
                <field name="by_employee"/>
                <field name="status"/>
                <field name="notes"/>
            </tree>
        </field>
    </record>

    <!-- os.order form view -->
    <record id="os_order_view_form" model="ir.ui.view">
        <field name="name">Form Order</field>
        <field name="model">os.order</field>
        <field name="arch" type="xml">
            <form string="">
                <sheet>
                    <group>
                        <!-- Add your fields here -->
                        <field name="order_number"/>
                        <field name="date_order"/>
                        <field name="customer"/>
                        <field name="address"/>
                        <field name="contact"/>

                        <field name="orderproduct_ids">
                            <tree editable='bottom'>
                                <field name="product_id"/>
                                <field name="price"/>
                                <field name="qty"/>
                                <field name="total_price" sum="Total"/>
                            </tree>
                        </field>
```

Gambar 3.22. Potongan Kode XML Order

Dari Gambar 3.22, *line* 43-50 merupakan potongan kode yang digunakan untuk melakukan penginputan data pada *fields One2many*. Fungsi `editable='bottom'` merupakan fungsi yang digunakan agar *tree* tersebut dapat diubah atau di *input* secara langsung di setiap barisnya, jika tidak maka penginputan *tree* akan dilakukan dengan memunculkan *pop-up wizard* (*modal*). Tampilan dari penggunaan fungsi ini dapat dilihat melalui Gambar 3.23 dan Gambar 3.24.



Gambar 3.23. Tampilan Fungsi *Editable* Sebelum Ditekan

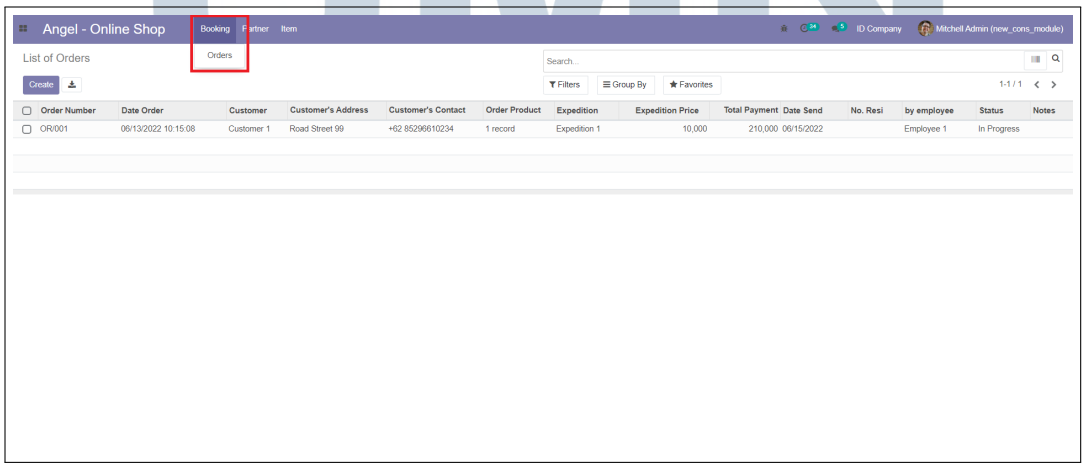


Gambar 3.24. Tampilan Fungsi *Editable* Setelah Ditekan

```
order_views.xml M
views > order_views.xml
52                          <field name="expedition"/>
53                          <field name="exp_price"/>
54                          <field name="total_payment"/>
55                          <field name="date_send"/>
56                          <field name="no_resi"/>
57                          <field name="by_employee"/>
58                          <field name="status" widget="statusbar" statusbar_visible=" " />
59                          <field name="notes"/>
60                      </group>
61                      <header>
62                          <button name="btn_paid" type="object" string="PAID" class="btn-info" />
63                          <button name="btn_send" type="object" string="SEND" class="btn-primary" />
64                          <button name="btn_arrived" type="object" string="ARRIVED" class="btn-success" />
65                          <button name="btn_returned" type="object" string="RETURNED" class="btn-danger" />
66                      </header>
67                  </sheet>
68              </form>
69          </field>
70      </record>
71
72      <!-- os.order action window -->
73      <record id="os_order_action" model="ir.actions.act_window">
74          <field name="name">List of Orders</field>
75          <field name="type">ir.actions.act_window</field>
76          <field name="res_model">os.order</field>
77          <field name="view_mode">tree,form</field>
78      </record>
79
80      <!-- This Menu Item must have a parent and an action -->
81      <menuitem
82          id="os_order_menu_act"
83          name="Orders"
84          parent="os_booking_menu_categ"
85          action="os_order_action"
86          sequence="10"/>
87  </odoo>
```

Gambar 3.25. Lanjutan Potongan Kode XML Order

Dari Gambar 3.25, *line* 81-86 menunjukkan bahwa *views* Order berada pada menu `parent` Booking seperti yang dapat dilihat pada Gambar 3.26.



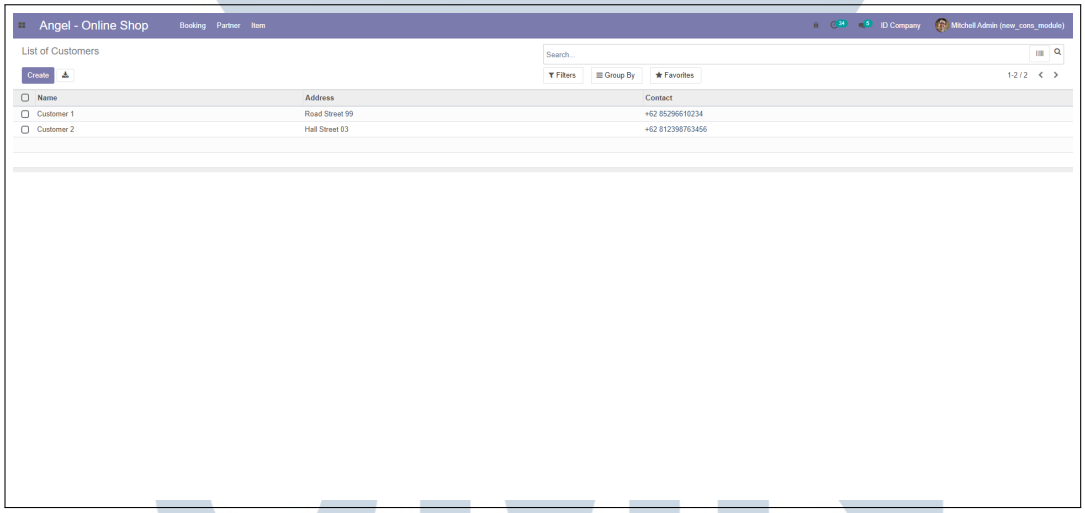
Gambar 3.26. Tampilan Menu Booking

B.1.2 Hasil Implementasi Modul

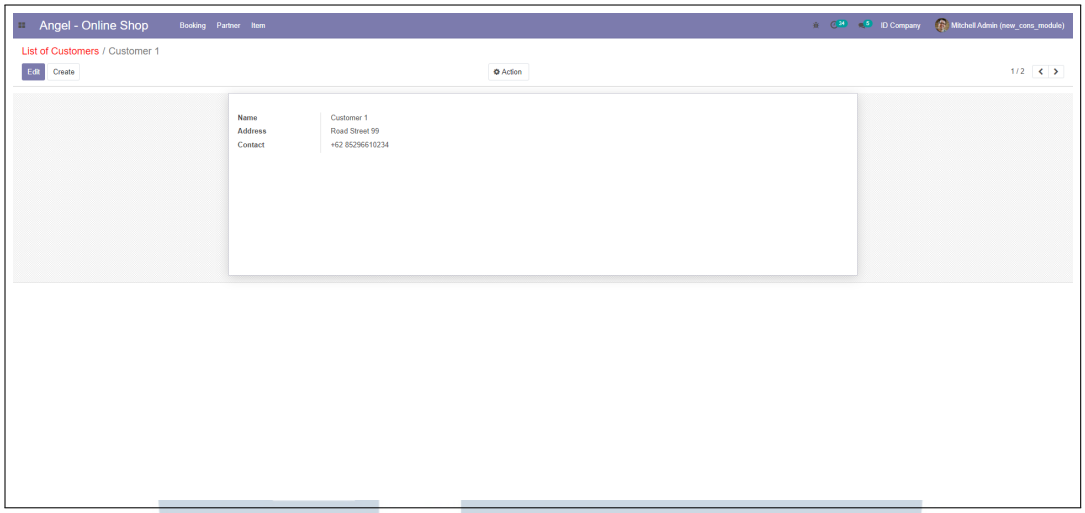
Berikut ini merupakan hasil implementasi dari pembuatan modul ”*Online Shop*” yang dikerjakan oleh penulis.

- Halaman Customer

  Halaman ini digunakan untuk membuat daftar pelanggan yang pernah melakukan transaksi perbelanjaan ataupun pelanggan baru yang akan melakukan transaksi pertamanya di toko *online* tersebut. Gambar 3.27 menunjukkan tampilan *tree* yang berisikan data berupa nama, alamat, dan nomor telepon dari setiap pelanggan. Sedangkan Gambar 3.28 menunjukkan tampilan *form* dari data pelanggan yang dipilih. Pada tampilan *form*, pengguna juga dapat melakukan pengubahan data dengan menekan tombol *edit*.



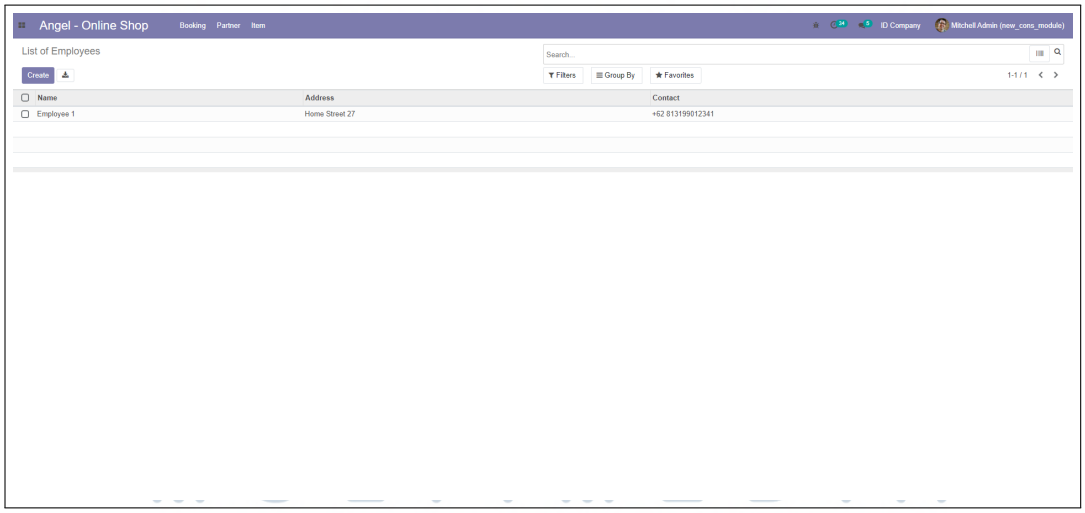
Gambar 3.27. Tampilan Tree Halaman Customer



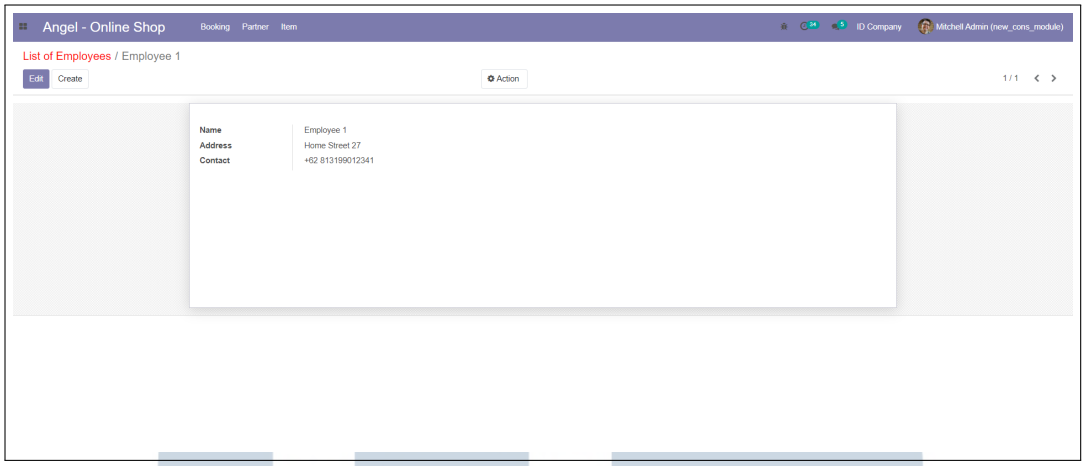
Gambar 3.28. Tampilan Form Halaman Customer

- Halaman Employee

  Halaman ini digunakan untuk membuat daftar pegawai yang berada di bawah naungan toko *online* tersebut. Gambar 3.29 menunjukkan tampilan *tree* yang berisikan data berupa nama, alamat, dan nomor telepon dari setiap pegawai. Sedangkan Gambar 3.30 menunjukkan tampilan *form* dari data pegawai yang dipilih.



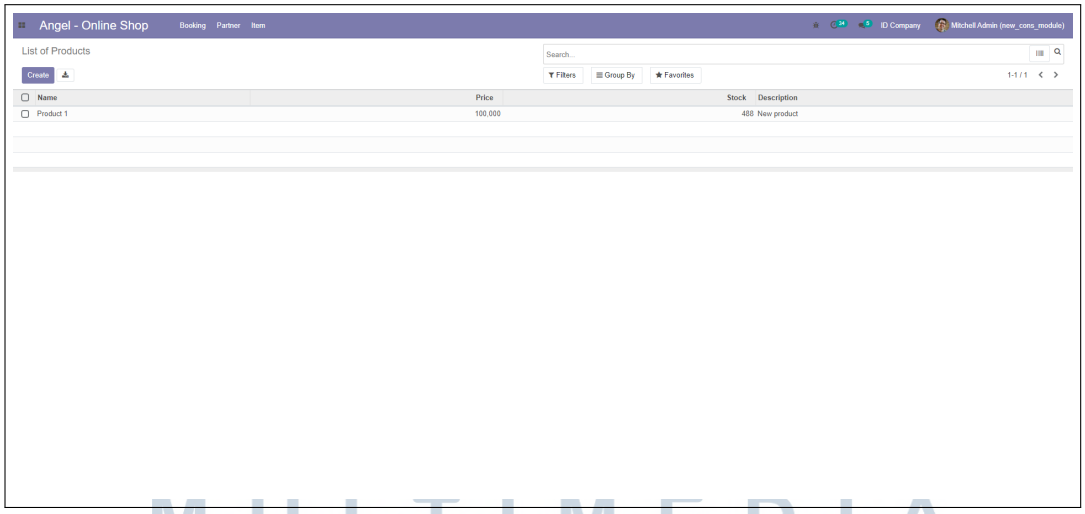
Gambar 3.29. Tampilan Tree Halaman Employee



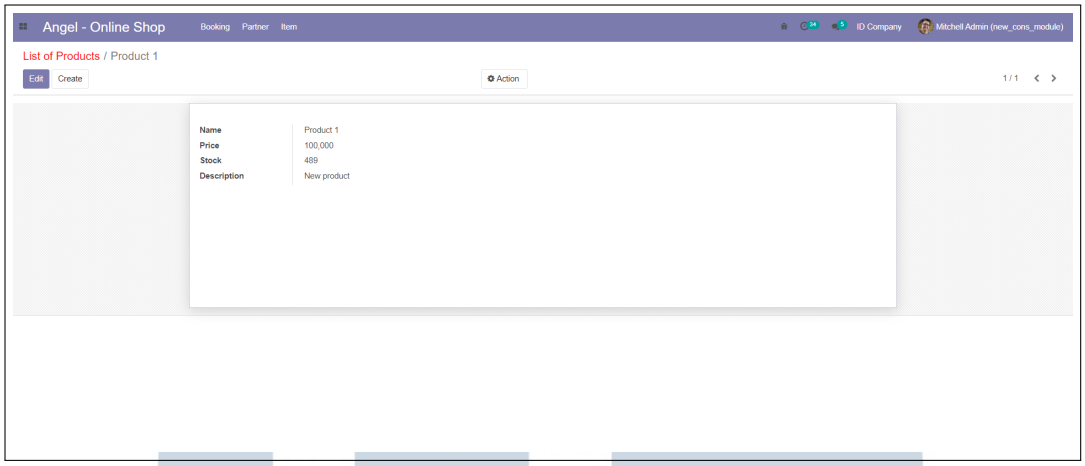
Gambar 3.30. Tampilan Form Halaman Employee

- Halaman Product

  Halaman ini digunakan untuk membuat daftar produk yang ditawarkan oleh toko *online* tersebut. Gambar 3.31 menunjukkan tampilan *tree* yang berisikan data berupa nama, harga per satuan, jumlah stok, dan deskripsi dari setiap produk yang ditawarkan. Sedangkan Gambar 3.32 menunjukkan tampilan *form* dari data produk yang dipilih.



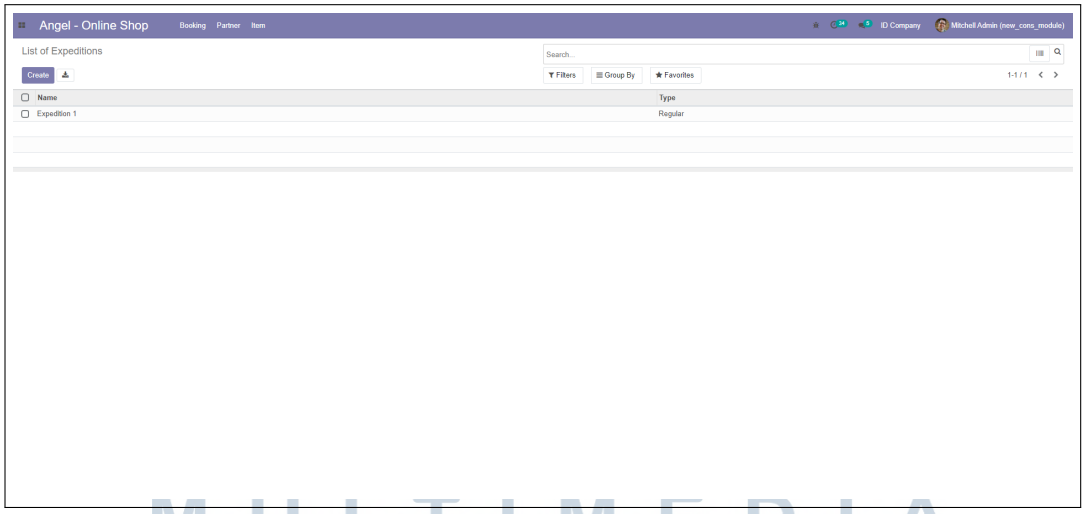
Gambar 3.31. Tampilan Tree Halaman Product



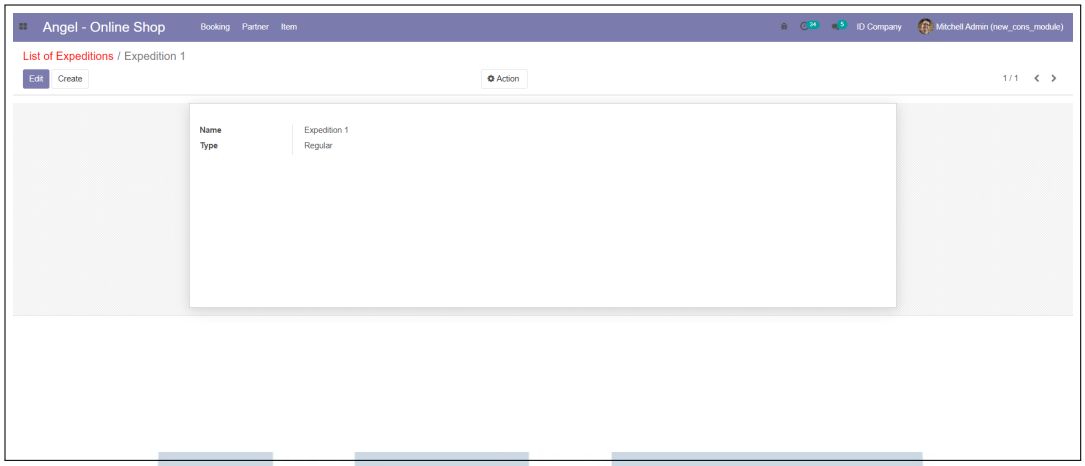
Gambar 3.32. Tampilan Form Halaman Product

- Halaman Expedition

  Halaman ini digunakan untuk membuat daftar ekspedisi pengiriman barang yang digunakan oleh toko *online* tersebut. Gambar 3.33 menunjukkan tampilan *tree* yang berisikan data berupa nama dan tipe dari setiap ekspedisi yang digunakan. Sedangkan Gambar 3.34 menunjukkan tampilan *form* dari data ekspedisi yang dipilih.



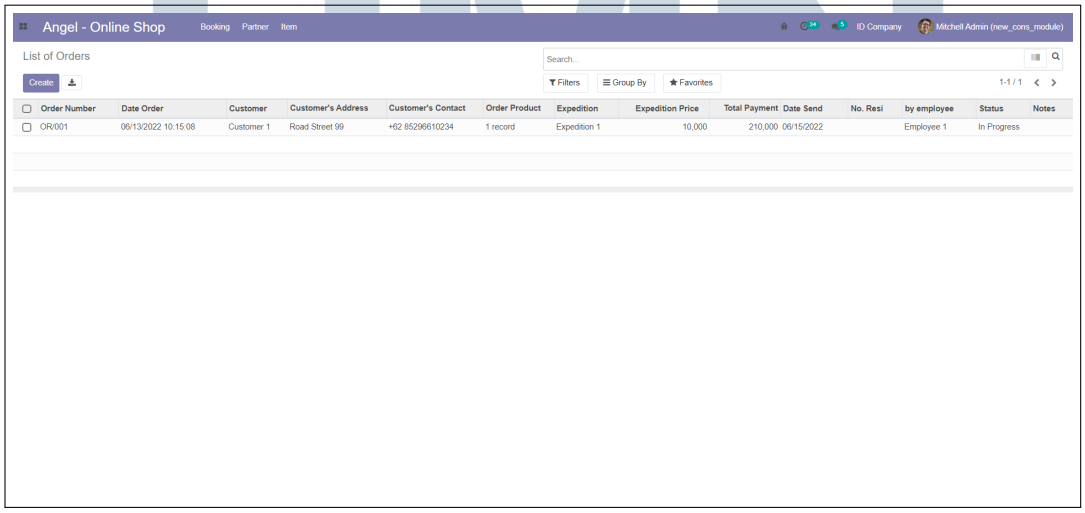
Gambar 3.33. Tampilan Tree Halaman Expedition



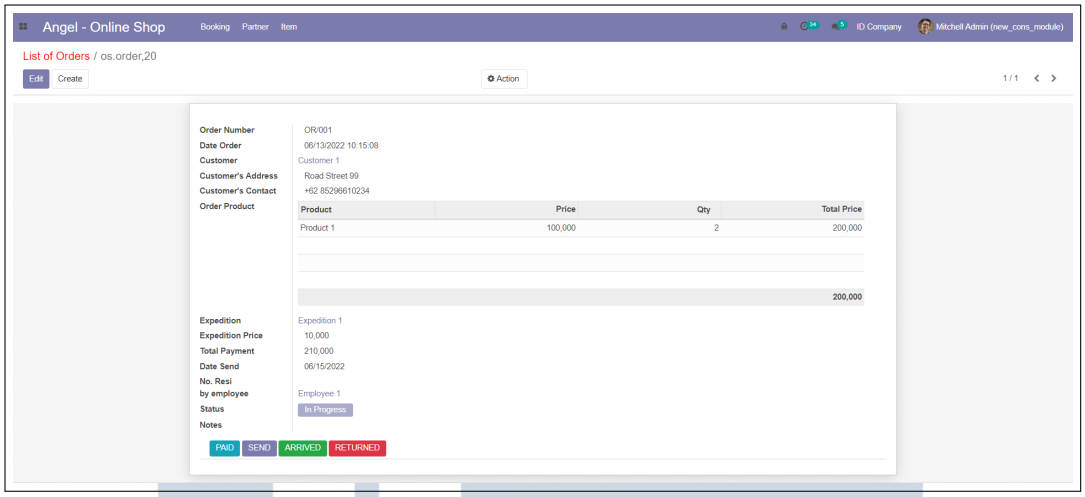
Gambar 3.34. Tampilan Form Halaman Expedition

- Halaman Order

  Halaman ini digunakan untuk membuat daftar pesanan yang dimiliki oleh toko *online* tersebut. Gambar 3.35 menunjukkan tampilan *tree* yang berisikan data berupa nomor pesanan, tanggal pesanan dipesan, nama, alamat, dan nomor telepon dari pelanggan yang memesan, daftar produk yang dipesan, ekspedisi yang digunakan, harga kirim ekspedisi, total harga pesanan, tanggal pesanan dikirim, nomor resi, pegawai yang memesan, status pesanan, dan catatan pesanan dari setiap pesanan yang dibuat. Sedangkan Gambar 3.36 menunjukkan tampilan *form* dari data pesanan yang dipilih.



Gambar 3.35. Tampilan Tree Halaman Order



Gambar 3.36. Tampilan Form Halaman Order

3.3.2 *Product Construction*

Setelah menjalani *training* selama 2 (dua) bulan dan mengerjakan tugas akhir *training*, penulis ditugaskan untuk melakukan pengembangan terhadap modul-modul *Product Construction* yang dimulai dari tanggal 04 April 2022 sampai dengan tanggal 30 Juni 2022. Adapun pelaksanaan realisasi kerja magang penulis selama berada di tim *Construction* akan diuraikan pada Tabel 3.2 dan Tabel 3.3.

Tabel 3.2. Pelaksanaan Realisasi Kerja Magang di Tim *Construction*

Minggu Ke -	Pekerjaan yang dilakukan
10	- Pengenalan tim *Construction* - Persiapan *local* - Penambahan *button* ”Confirm”, ”Confirm Project Budget”, dan ”Approve” (Project Budget) - Penambahan *domain* pada *fields* Cost Sheet (*Project Budget*)
11	- Penambahan *fields* is_subcon:boolean dan *attributes invisible* pada *fields* tertentu (*Work Orders*) - Penambahan *attributes readonly* pada *fields* tertentu (*Purchase Agreement*) - Pengubahan *value string* pada *widget statinfo* (*Work Orders*) - Penambahan *fields* cost_sheet:Many2one dan project_budget:Many2one (*Work Orders*) - Pembuatan format *page* di dalam *page* (*Work Orders*) - Penambahan *fields* consumed_history:One2many untuk setiap *fields* consumed (*Work Order*) - Penambahan *fields* subcon_id:Many2one dan *attributes invisible* pada *fields* tertentu di *page* Timesheets (*Work Order*)
12	- Pengubahan *value* pada *fields* state (*Budget Period*) - Penambahan *attributes invisible* pada *fields* tertentu (*Budget Period*) - Penambahan *button* ”Reset to Draft” (*Budget Period*) - Penambahan *view* ks_gantt (*Work Order*) - Melakukan *testing* kode (*Product Usage*)
13	- Penambahan SCSS pada *fields* tertentu (*Project Budget*)
14	- Melakukan *fixing error* pada *view* ks_gantt (*Work Order*) - Mempelajari fungsi-fungsi yang dimiliki *view* ks_gantt (*Work Order*) - Penambahan *fields* gantt_chart_color:Char dan dihubungkan ke fungsi `ks_task_color` pada *views* ks_gantt (*Work Order*) - Penambahan *attributes readonly* pada *fields* tertentu (*Work Order*) - Penambahan *page* Progress History beserta *fields* yang dibutuhkan (*Work Order*)

Tabel 3.3. Lanjutan Pelaksanaan Realisasi Kerja Magang di Tim *Construction*

Minggu Ke -	Pekerjaan yang dilakukan
15	- Penambahan *button* ”Create Revision” (Job Estimate) - Penambahan *smart button* ”Revision” (*Job Estimate*) - Penambahan *views tree* ”Revision History” (*Job Estimate*)
16 - 17	- Melakukan *fixing* pada *logic button* ”Cancel” (*Product Usage*) - Penambahan SCSS pada *page* Progress History (*Work Order*) - Penghapusan *fields* material_subcon (*Work Order*)
18	- Pengubahan *logic function* pada *fields* total, discounted_total, dan total_all (*Request for Quotation*) - Penambahan *fields* project:Many2one pada 4 (empat) menu di Approval Matrix Configuration, dikerjakan dengan melakukan *inherit* (*Purchase*) - Penambahan menu (*object*) Project Budget Approval Matrix di Configuration (*Construction*) - Penambahan menu (*object*) *Budget Period Approval Matrix* di Configuration (*Construction*) - Penambahan *fields* construction_project:Boolean, is_project_budget_approval_matrix:Boolean, dan is_budget_period_approval_matrix:Boolean serta *attributes invisible* (*Implementor General Settings*) - Penambahan menu Variable dan Subcon Estimate di Configuration (*Construction*) - Penambahan *default boolean* pada menu Variable dan Subcon Estimate di Configuration (*Purchase*) - Pengubahan dan penambahan *fields* pada *page* Costing serta mengatur CSS pada *fields* tersebut (*Project*) - Penambahan beberapa *fields* pada *views tree wizard*, dikerjakan dengan melakukan *inherit*, di Material Purchase Request Wizard
19	- Penambahan SCSS pada *page* Product (*Group of Product*) - Penambahan *tree* pada S-Curve (*Project*)

A. Modul *Project Construction*

A.1 Menu *Project Budget*

Terdapat 3 (tiga) buah *task* yang dikerjakan pada menu *Project Budget* ini sebagai berikut.

1. Penambahan 3 (tiga) *button* beserta *logic*-nya

   Button baru yang dibutuhkan pada menu ini adalah *button* ”Confirm Project Budget”, ”Confirm”, dan ”Approve”. *Button* ”Confirm Project Budget” digunakan sebagai *button* yang akan mengganti nilai state menjadi ”approve” dan hanya akan dimunculkan pada saat state bernilai ”draft”. Sedangkan *button* ”Confirm” digunakan sebagai *button* untuk mengganti state menjadi ”in_progress” dan hanya dimunculkan pada saat state bernilai ”approved”. Untuk *button* ”Approve” belum diberi *logic* khusus sehingga *button* ini hanya dibuat tanpa ditampilkan. Penambahan *button* ini dilakukan dengan membuat *tag* `<button>` pada *views* (XML) beserta *method*-nya pada *models* (Python). Contoh penggunaan *tag* `<button>` dapat dilihat pada Kode 3.1.

   Kode 3.1: Contoh penggunaan *tag button*

   ```
   <button name="nama_method" string="Nama Button" type="
       object"/>
   ```

   Berdasarkan Kode 3.1, fungsi `name` digunakan untuk menghubungkan *button* dengan *method* atau *logic*-nya. Sehingga jika `name` yang digunakan adalah nama_method maka Kode 3.2 merupakan contoh penerapannya pada *file* Python.

   Kode 3.2: Contoh *method* pada *button*

   ```
   def nama_method(self):
       pass
   ```

   Sedangkan untuk dapat menampilkan *button* hanya pada kondisi tertentu dapat menggunakan fungsi *attributes* (`attrs`) di dalam *tag* `<button>` seperti contoh pada Kode 3.3.

   Kode 3.3: Contoh penggunaan fungsi *attributes*

   ```
   attrs="{'invisible': [('nama_field', '=', 'value_field')
       ]}"
   ```

Kode 3.3 memberikan perintah untuk melakukan `invisible` (fungsi untuk menyembunyikan) pada saat memenuhi kondisi dimana suatu *fields* memiliki *value* tertentu. Sedangkan untuk menyembunyikan *button* "Approve" tanpa syarat tertentu dapat dilakukan dengan menggunakan fungsi `hidden="true"` atau `invisible=1`.

2. Penambahan fungsi `domain`

   Fungsi `domain` merupakan fungsi yang digunakan untuk melakukan *filtering* pada *fields* tertentu. Fungsi ini digunakan di dalam *tag fields* pada *file* XML (*views*). Hasil yang ingin dicapai dari *task* ini adalah *fields* cost_sheet akan mengisi *value*-nya secara otomatis berdasarkan *fields* project_id yang dipilih. Contoh penggunaan fungsi `domain` ini dapat dilihat pada Kode 3.4.

   Kode 3.4: Contoh penggunaan fungsi *domain*

   ```
   domain="[('id_field', '=', nama_field)]"
   ```

   Dari Kode 3.4, sisi sebelah kiri `('id_field')` menunjukkan *id* dari *fields* bersangkutan yang akan dihubungkan dan sisi sebelah kanan `(nama_field)` menunjukkan *fields* mana yang akan dihubungkan dengan sisi sebelah kiri.

3. Penambahan *file* SCSS

   Untuk mengatur *width* dari setiap kolom (*fields*) yang terdapat di dalam *tree* dapat dilakukan dengan menambahkan *class* dan membuat *file* SCSS dengan nama *class* tersebut.

A.2 Menu *Work Orders*

Task yang dikerjakan pada menu *Work Orders* akan dijelaskan sebagai berikut.

1. Penambahan *fields Boolean*

   Tujuan dari *task* ini adalah membuat sebuah *fields boolean* dengan nama is_subcon yang memiliki `default` bernilai *False*. *Value* dari *fields* ini ditentukan dari *fields* lain yang bernama sub_contractor. Apabila *fields* sub_contractor diisi maka *value* dari is_subcon adalah *True* dan begitu pula sebaliknya. Selain itu, apabila *value* is_subcon adalah *True* maka akan membuat beberapa *fields* lainnya menjadi *invisible* menggunakan *attributes* `invisible`.

2. Pengubahan *page* Timesheets

Task ini bertujuan untuk menambahkan *fields* baru ke dalam *tree* pada *page* Timesheets yaitu *fields* subcon_id yang memiliki tipe data *Many2one*. Kemudian *fields* ini hanya akan dimunculkan apabila *fields* is_subcon bernilai *True* dan memiliki *value* yang diambil dari *fields* sub_contractor. Sebaliknya jika *fields* is_subcon bernilai *False*, maka *fields* subcon_id tidak akan dimunculkan. Selain itu, *fields* employee_id juga akan disembunyikan pada saat situasi yang berkebalikan dengan subcon_id yaitu hanya akan dimunculkan pada saat is_subcon bernilai *False*. Untuk melakukan penyembunyian *fields* pada *tree* digunakan *atrributes* `column_invisible` yang dapat dilihat seperti pada contoh Kode 3.5.

Kode 3.5: Contoh penggunaan fungsi *column_invisible*

```
attrs="{'column_invisible':[('parent.nama_field', '==',
    value_field)]}"
```

Dari Kode 3.5, dapat dilihat bahwa untuk melakukan *invisible* pada sebuah kolom dapat menggunakan fungsi `column_invisible` dan memanggil *fields* tersebut menggunakan *parent* didepannya. Jika kita menggunakan `invisible` biasa untuk melakukan penyembunyian *fields* pada kolom, maka yang akan disembunyikan hanyalah data yang ada dibawahnya tetapi *header* atau judul dari *fields* tersebut akan tetap ada. Oleh karena itu, untuk menyembunyikan keseluruhan kolom digunakan fungsi `column_invisible`.

Dikarenakan *page* Timesheets ini berada pada *object* yang berbeda maka untuk melakukan pengubahan pada *page* ini diperlukan penggunaan *inherit object* pada *file models*-nya dan menggunakan *tag* `<xpath>` untuk mengubah tampilan *views*-nya. Kode 3.6 adalah contoh penggunaan *tag* `<xpath>`.

Kode 3.6: Contoh penggunaan *tag* xpath

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

Selain itu, untuk mengambil *value* suatu *fields* dari *object* yang berbeda dilakukan dengan cara menghubungkan ke *object* lain tersebut menggunakan `self.env['nama.object']`. *Method* tersebut digunakan

untuk menghubungkan suatu variabel dengan *object* yang dituju. Kemudian dilakukan pencarian *fields* yang terhubung antar *object* yang berbeda tersebut menggunakan `search`. Contoh penggunaan *method* ini ditunjukkan pada Kode 3.7.

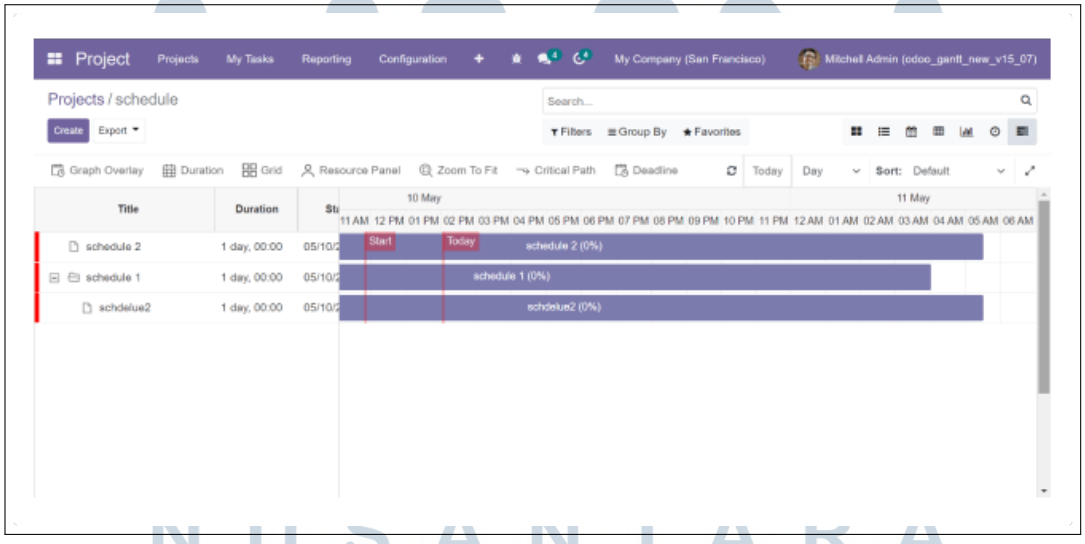
Kode 3.7: Contoh *method* untuk menghubungkan ke *object* lain

```
var = self.env['nama.object'].search([(
            'id_field', '=', self.field)])
```

Setelah mendapatkan suatu variabel yang menyimpan data dari *object* lain, maka langkah selanjutnya yang dilakukan adalah menyimpan *value* dari *fields* yang ada di *object* lain tersebut ke dalam *fields* yang diinginkan ataupun sebaliknya menggunakan `write`.

3. Penambahan *view* ks_gantt

   Tampilan grafik ks_gantt merupakan singkatan dari Ksolves Gantt View yang digunakan untuk membuat Gantt *Chart*. Penambahan *view* ini cukup dilakukan dengan menambahkan *record* dengan *tag* `<ks_gantt>` pada *views* dan memasukkan *value* dari setiap fungsi atau fitur di dalam *tag* ini sesuai dengan data yang ada. Tampilan Gantt *Chart* yang akan dihasilkan dapat dilihat pada Gambar 3.37.



Gambar 3.37. Tampilan Gantt *Chart* (ks_gantt)

Sumber: [9]

4. Penambahan *attributes* `readonly` pada *fields* tertentu

   Penggunaan *attributes* `readonly` di beberapa *fields* tertentu digunakan untuk membuat *fields* yang bersangkutan hanya dapat dibaca sehingga tidak dapat diinput atau dimasukkan data baru. Sama halnya dengan penggunaan *attributes* `invisible`, *attributes* `readonly` digunakan dengan cara memasukkannya ke dalam *tag fields* yang diinginkan pada *file* XML (*views*). Contoh penggunaan *attributes* `readonly` dapat dilihat pada Kode 3.8.

   Kode 3.8: Contoh penggunaan fungsi *readonly*

```
attrs="{'readonly': [('nama_field', '=', 'value_field')
    ]}"
```

   Berdasarkan Kode 3.8, maka *fields* yang bersangkutan akan menjadi *readonly* apabila memenuhi kondisi dimana *fields* tertentu memiliki *value* yang ditentukan.

5. Penambahan *page* Progress History

   Penambahan *page* ini diawali dengan menambahkan *fields One2many*. Kemudian setelah *fields One2many* dibuat, langkah selanjutnya adalah pembuatan *object* baru berdasarkan *fields One2many* tersebut. Lalu, di dalam *object* baru tersebut dimasukkan seluruh *fields* yang dibutuhkan. Setelah itu, dilanjutkan dengan pemanggilan *fields* pada *file* XML (*views*) agar *fields* tersebut dapat ditampilkan. Untuk dapat menampilkan *fields* ke dalam bentuk *page* dapat dilakukan dengan menggunakan *tag* `<page>` seperti contoh pada Kode 3.9.

   Kode 3.9: Contoh penggunaan *tag page*

```
<page name="nama" string="Nama Page">
    <field name="nama_field_o2m" class="nama_class">
        <tree>
            <field name="nama_field"/>
        </tree>
    </field>
</page>
```

   Tag `<page>` digunakan untuk membuat tampilan *tab*, biasanya *tag* ini diawali dengan *tag* `<notebook>`. Kemudian didalam *tag* `<page>` dilakukan

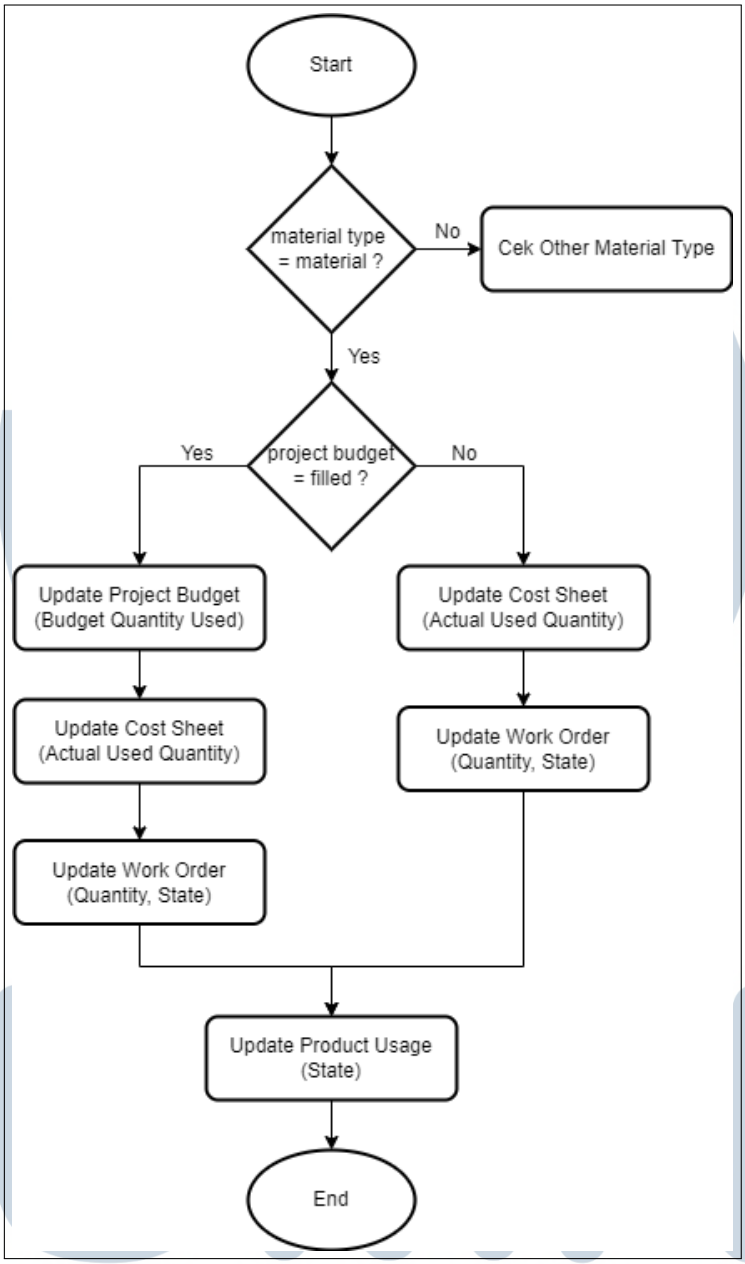
pemanggilan terhadap *fields One2many* yang terdiri dari beberapa *fields* yang akan ditampilkan menggunakan *tree*. Selain itu, *task* ini juga diakhiri dengan pembuatan SCSS untuk mengatur *width* dari setiap kolom yang terdapat pada *tree* tersebut.

A.3 Menu *Budget Period*

Task yang dilakukan pada menu ini adalah mengubah *value* pada *fields* state yang terdapat di dalam *tree* pada *page* Monthly Periods sesuai dengan *logic* pada *button* yang dioperasikan, menggunakan *attributes* `invisible` pada beberapa *fields*, dan menambahkan *button* baru yaitu ”Reset to Draft” beserta dengan *logic*-nya seperti mengubah *value* state, hanya dimunculkan pada saat state memiliki *value* tertentu, dan sebagainya.

A.4 Menu *Product Usage*

Task yang dikerjakan pada menu ini adalah melakukan *fixing* pada *logic button* ”Cancel”. Adapun *logic button* yang ingin dicapai digambarkan melalui *flowchart* pada Gambar 3.38 berikut.



Gambar 3.38. *Flowchart Logic Button* "Cancel"

Berdasarkan *flowchart* tersebut, *logic button* "Cancel" diawali dengan melakukan pengecekan terhadap tipe material yang digunakan. Tipe material yang terdapat pada Product Usage sendiri terdiri dari 4 (empat) yaitu Material, *Equipment*, *Labour*, dan *Overhead*. Namun untuk *flowchart* yang ditampilkan pada Gambar 3.38 hanya menunjukkan kondisi dimana tipe material adalah material atau tidak. Hal ini dikarenakan meskipun tipe material yang digunakan bukan material, urutan *flow* yang akan digunakan setelah pengecekan tipe material akan tetap sama.

Setelah pengecekan tipe material selesai dilakukan, maka langkah selanjutnya adalah melakukan pengecekan terhadap *fields* Project Budget. Jika *fields* tersebut memiliki isi atau *value* maka akan proses yang akan dijalankan adalah proses yang berada di sebelah kiri yaitu melakukan *update* pada Project Budget (mengganti *value* dengan melakukan perhitungan pada Budget Quantity Used), Cost Sheet (mengganti *value* dengan melakukan perhitungan pada Actual Used Quantity), dan Work Order (mengganti *value* dengan melakukan perhitungan pada Quantity dan State). Sedangkan jika *fields* tersebut tidak memiliki isi atau *value* maka proses yang akan dijalankan adalah sisi sebelah kanan yaitu melakukan *update* pada Cost Sheet (mengganti *value* dengan melakukan perhitungan pada Actual Used Quantity) dan Work Order (mengganti *value* dengan melakukan perhitungan pada Quantity dan State). Kemudian *flowchart* ini diakhiri dengan melakukan proses terakhir yaitu melakukan *update* pada Product Usage berupa mengganti *value* State.

Button "Cancel" ini digunakan untuk mengembalikan data yang sebelumnya telah dimasukkan ke dalam data atau dengan kata lain melakukan penghapusan data. Contoh kasus dari penggunaan *button* ini dapat digambarkan sebagai berikut.

"Budi memesan sebuah menu makanan menggunakan aplikasi pemesanan *online*. Awalnya, ia memasukkan menu nasi goreng sebanyak 1 (satu) porsi ke dalam keranjang. Kemudian ia teringat kepada adiknya yang juga belum makan sehingga ia memesan 1 (satu) porsi nasi goreng lagi untuk adiknya. Sekarang jumlah nasi goreng di keranjang Budi terdapat sebanyak 2 (dua) porsi. Namun adiknya tidak ingin memakan nasi goreng tetapi mie goreng sehingga Budi mengurangi seporsi nasi goreng tersebut dari keranjangnya. Sekarang porsi nasi goreng pada keranjang Budi adalah 1 (satu) porsi. Lalu, pada saat ingin melanjutkan ke pembayaran, Budi tiba-tiba berubah pikiran dan ikut memesan menu mie goreng seperti adiknya sehingga menu nasi goreng tidak lagi berada di keranjang Budi sekarang."

Logic button "Cancel" di menu ini mirip dengan kasus Budi diatas pada saat Budi ingin mengurangi porsi pada nasi goreng yang dipesannya. Pada saat pertama kali Budi mengurangi porsi nasi goreng-nya, hanya dilakukan pengurangan terhadap jumlah kuantitas pada menu nasi goreng tersebut. Namun pada saat Budi ingin mengurangi lagi jumlah porsi nasi goreng-nya, maka menu nasi goreng tersebut sekarang tidak ada lagi berada di keranjang atau dihapus.

Dalam pengerjaan untuk menjalankan *logic button* ini dibantu dengan menggunakan Odoo *Special Commands* yaitu `[(1, id, {})]` untuk melakukan

modifikasi *record* (data) pada *value fields* tertentu dan `[(2, id, 0)]` untuk melakukan penghapusan *record* (data) tertentu. Contoh penggunaannya dapat dilihat pada Kode 3.10.

Kode 3.10: Contoh penggunaan Odoo *Special Commands*

```
# Modified
field_ids = [(1, field.id, {
        'nama_field': value,
    })]

# Deleted
field_ids = [(2, field.id, 0)]
```

A.5 Menu *Projects*

Adapun *task* yang dikerjakan pada menu ini sebagai berikut.

1. Pengubahan dan penambahan *fields* pada *page* Costing

   Task ini bertujuan untuk melakukan pengubahan terhadap *page* Costing dengan mengubah nama `string` dari *page* tersebut menjadi Cost & Revenue serta menyembunyikan *fields* yang dulu dimilikinya dan menggantikannya dengan *fields* yang baru. *Task* ini dilakukan dengan melakukan *inherit* pada *file* Python dan XML-nya. Untuk melakukan pengubahan dan penambahan tersebut pada *file* XML dapat menggunakan *tag* `<xpath>` seperti contoh pada Kode 3.11.

   Kode 3.11: Contoh kode pengubahan dan penambahan pada *tag* xpath

```
<!-- hide fields -->
<xpath expr="//field[@name='nama_field']" position="
    attributes">
     <attribute name="invisible">1</attribute>
</xpath>

<!-- rename -->
<xpath expr="//field[@name='nama_field']" position="
    attributes">
     <attribute name="string">Nama Baru</attribute>
```

```
</xpath>

<!-- new fields -->
<xpath expr="//field[@name='nama_field']" position="
   inside">
    <field name="nama_field" />
</xpath>
```

Dari Kode 3.11, terdapat fungsi `expr` yang digunakan sebagai jalur lokasi dari *item* yang terdapat pada XML *parent* yang akan ditambahkan atau diubah. Fungsi `position` digunakan untuk menetapkan posisi dari letak *item* yang terdapat dibawahnya (selanjutnya hanya akan disebut sebagai *items*) terhadap *fields* yang terdapat pada `expr` (selanjutnya akan disebut sebagai *fields parent*). `position` memiliki beberapa macam *value* seperti *inside, before, after,* dan *attributes*. Secara *default*, xpath akan mengatur `position` menjadi *inside* apabila tidak diatur.

Inside digunakan untuk menambahkan *items* ke dalam *fields parent*. *Before* digunakan untuk menambahkan *items* di belakang *fields parent*. *After* digunakan untuk menambahkan *items* di depan *fields parent*. *Attributes* digunakan untuk mengubah *attributes* pada *fields parent* yang mencakup *invisible, string, readonly,* dan sebagainya.

Sedangkan untuk melakukan *inherit* pada *file* Python dapat menggunakan Kode 3.12.

Kode 3.12: Contoh penggunaan *inherit* pada Python

```
class NamaModul(models.Model):
    _inherit = 'nama.modul.parent'

    nama_fields_baru = fields.Char(string='')
```

2. Penambahan *tree* pada S-Curve

   Task ini bertujuan untuk menambahkan *tree* dibawah grafik S-Curve. Penambahan *tree* tersebut dilakukan dengan pembuatan *fields One2many* dan *object* baru yang dilengkapi dengan penambahan *fields* yang dibutuhkan. Penambahan *fields One2many* ini dilakukan pada *file* Python dan XML.

A.6 Menu *Construction*

Task yang dikerjakan pada menu ini adalah penambahan menu baru. Penambahan menu baru ini dilakukan di bawah menu *parent* Configuration sebanyak 4 (empat) menu yaitu Project Budget Approval Matrix, Budget Period Approval Matrix, Variable, dan Subcon Estimate. Menu Project Budget Approval Matrix dan Budget Period Approval Matrix merupakan menu baru yang dibuat berdasarkan menu Approval Matrix lain sehingga *task* ini dikerjakan dengan melakukan penambahan *object* baru pada *file* Python dan pemanggilan pada *file* XML berdasarkan menu lain yang telah ada sebelumnya. Sedangkan menu Variable dan Subcon Estimate merupakan menu baru yang dibuat hanya dengan melakukan penambahan terhadap *file* XML-nya saja karena menu ini telah ada pada modul master sebelumnya.

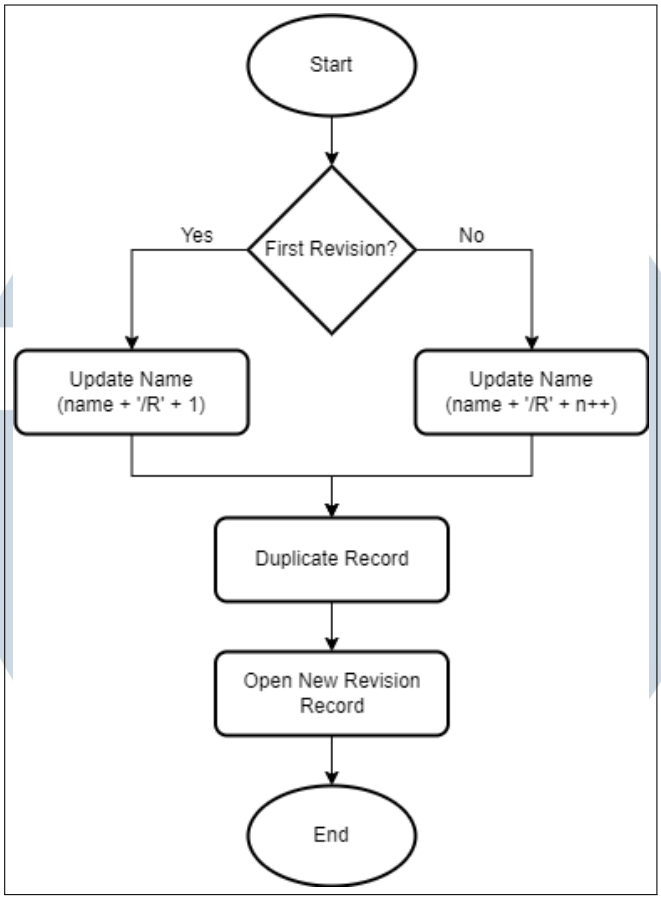
B. Modul *Sales Construction*

B.1 Menu *Job Estimates*

Task yang dikerjakan pada menu ini terdiri dari 3 (tiga) bagian yang berhubungan seperti yang akan dijelaskan sebagai berikut.

1. Penambahan *button*

   Task ini diawali dengan melakukan penambahan *button* ”Create Revision”. *Button* ini akan dimunculkan pada *record* (data) saat *record* tersebut memiliki state yang bernilai rejected. State rejected menunjukkan bahwa *record* (*record* disini anggap saja sebagai *record* pengajuan perencanaan pekerjaan) tersebut tidak disetujui sehingga perlu untuk diajukan ulang. Sehingga dibutuhkan sebuah *button* yang dapat melakukan *duplicate record* untuk mempermudah perevisian data yaitu *button* ”Create Revision”. Adapun *logic* pada *button* ini akan digambarkan pada Gambar 3.39.



Gambar 3.39. *Flowchart Logic Button* "Create Revision"

Berdasarkan *flowchart* tersebut, *logic button* ini dimulai dengan melakukan pengecekan kondisi apakah *record* yang akan direvisi merupakan *record* awal (belum memiliki revisian sama sekali) atau *record* lanjutan (telah memiliki revisi sebelumnya). Apabila *record* tersebut belum memiliki revisian sebelumnya, maka akan dilakukan pengubahan nama pada *record* baru (revisi) dengan menambahkan `'/R1'` dibelakang nama *record* awal. Sedangkan jika *record* tersebut merupakan *record* lanjutan yang telah memiliki revisi sebelumnya, maka akan dilakukan pengubahan nama pada *record* revisi dengan menambahkan `'/R' + n++` yang artinya menambahkan `n++` dibelakang `/R`, misalnya *record* lanjutan sebelumnya adalah `nama/R1` maka *record* revisi selanjutnya akan diberi nama `nama/R2` dan begitu pula selanjutnya. Setelah selesai mengganti nama, proses akan dilanjutkan dengan melakukan *duplicate record* dari *record* sebelumnya. *Duplicate record* ini tidak hanya melakukan *duplicate* data saja tetapi juga melakukan *duplicate* fungsi. Sehingga meskipun *record* awal telah dilakukan revisi, tetapi *record*

revisi selanjutnya juga dapat melakukan revisi untuk kedepannya. Setelah proses *duplicate* selesai, maka *logic* pada *button* ini akan diakhiri dengan menjalankan proses membuka halaman *record* revisi baru.

Untuk melakukan proses pengubahan nama pada *record* revisi menggunakan `split`, `append`, dan `join`. Pertama, nama *record* sebelumnya akan dipisahkan terlebih dahulu ke dalam bentuk *array* menggunakan `split`. Setelah itu, dilakukan penambahan *value* baru ke dalam *array* tersebut menggunakan `append`. Terakhir, *array* tersebut digabungkan kembali menggunakan `join`. Sedangkan untuk melakukan *duplicate* data menggunakan `copy`. Kode 3.13 adalah contoh potongan kode menggunakan `copy`.

Kode 3.13: Contoh penggunaan *copy*

```
field_ids.copy({
        'nama_field': value })
```

2. Penambahan *smart button* dan *views tree*

   Task ini merupakan kelanjutan dari *task* yang ada di poin pertama yaitu pembuatan *smart button* ”Revision” dan *views tree* ”Revision History”. *Smart button* ini hanya akan ditampilkan pada *record* lama yang telah dilakukan revisi (artinya *record* sebelumnya yang telah ditekan *button* ”Create Revision”) sehingga jika *record* tersebut masih baru, dapat dalam artian *record* awal yang belum pernah direvisi atau *record* revisi yang belum direvisi, maka *smart button* ini tidak akan dimunculkan. Setelah pembuatan *smart button*, *task* dilanjutkan dengan pembuatan *views tree* baru. *Views* ini akan ditampilkan pada saat *smart button* tersebut ditekan yang berisi *tree* dari *record* revisi berdasarkan *parent*-nya (*record* sebelumnya). Penambahan *smart button* dapat dilakukan dengan menggunakan Kode 3.14.

Kode 3.14: Contoh kode penambahan *smart button*

```
<div name="button_box" position="inside">
    <button name="nm_method" type="object" class="
        oe_stat_button" >
        <field string="Field" name="nm_field" widget="
            statinfo" />
    </button>
</div>
```

C. Modul *Purchase Construction*

C.1 Menu *Request for Quotation*

Task yang terdapat pada menu ini adalah pengubahan *logic function* pada *fields* total, discounted total, dan total all. Pengerjaan *task* ini dilakukan dengan menggunakan `compute` untuk setiap *fields* tersebut yang kemudian dilanjutkan dengan pembuatan *method* perhitungan. Tujuan dari *task* ini adalah membuat *fields* total dapat menghitung total harga dari keseluruhan *record* pada *tree* diatasnya, *fields* discounted total yang mengurangi total harga pada *fields* total dengan diskon yang ada, dan *fields* total all yang menambahkan harga setelah diskon pada discounted total dengan harga pajak.

3.4 Kendala dan Solusi yang Ditemukan

Adapun kendala dan solusi yang ditemukan selama pelaksanaan kerja magang di Hashmicro dijelaskan sebagai berikut.

3.4.1 Kendala

1. Dikarenakan *framework* Odoo baru dipelajari oleh penulis selama kegiatan magang berlangsung, sehingga masih ada beberapa fitur atau fungsi dari Odoo yang tidak diketahui oleh penulis meskipun telah menjalankan *training* selama kurang lebih 2 (dua) bulan. Hal ini menyebabkan penulis kesusahan menyelesaikan *task* yang diberikan.

2. Modul *Construction* merupakan modul *product* dari HashMicro yang sudah ada dan dikembangkan jauh sebelum penulis melaksanakan kegiatan magang. Sehingga dalam pengembangan modul ini, kadang kala penulis kebingungan dalam pencarian modul (*object*) ataupun *file* mana yang dipakai untuk menyelesaikan *task*.

3. Pelaksanaan kerja magang dilakukan secara WFH (*Work From Home*) sehingga komunikasi antara penulis dan *System Analyst* (SA) hanya dapat dilakukan melalui aplikasi *chat* (Skype). Hal ini menyebabkan penulis sering mengalami kebingungan dalam memahami *task* yang diberikan.

4. Dikarenakan komunikasi hanya dapat dilakukan secara *online* dan tidak dapat

bertemu secara langsung, sering kali penulis menunggu balasan *chat* dari SA untuk menunggu kelanjutan kabar dari *task* yang ada.

3.4.2 Solusi

1. Melakukan *research* melalui internet seperti membaca dokumentasi, mencari permasalahan *task* yang mirip di forum Odoo, atau menonton tutorial di Youtube serta menanyakannya kepada SA, mentor, atau teman sesama magang di HashMicro.
2. Berusaha untuk mencari *object* atau *file* yang tepat terlebih dahulu, jika dirasa sudah sangat kebingungan maka menanyakannya kepada SA.
3. Berdiskusi dengan SA kembali melalui *chat* atau melakukan *video call* dan *screen sharing*.
4. Menunggu balasan *chat* dari SA atau mencoba untuk mengeksplorasi sendiri terlebih dahulu terhadap *task* tersebut.