BAB III
PELAKSANAAN KERJA PROFESI

3.1 Bidang Kerja

Selama menjalani KP di PT Jaya Teknik Indonesia Praktikan ditempatkan pada divisi IT khususnya bidang IT Support, divisi IT ini bertanggung jawab terhadap kelancaran operasional teknologi informasi perusahaan, termasuk perancangan, pengembangan, pengelolaan, pemeliharaan, serta peningkatan sistem yang digunakan oleh seluruh unit perusahaan. Salah satu sistem utama yang telah diimplementasikan dan dikelola oleh PT Jaya Teknik Indonesia ini adalah Aplikasi E-TEKIND, Sebuah sistem ERP berbasis Odoo yang digunakan oleh perusaan untuk mengintegrasikan berbagai fungsi operasional untuk keseluruhan bisnis. Dalam bidang kerja yang dijalankan oleh Praktikan selama masa KP di PT Jaya Teknik Indonesia ini, Praktikan telah berkontribusi dalam beberapa aspek di antaranya:

1. Melakukan pengembangan pada modul report di Odoo versi 17
2. Melakukan pembaruan versi android untuk aplikasi *mobile* E-TEKIND dan CMMS Tiket
3. Membuat modul custom API untuk kebutuhan Odoo versi 17

Dengan bertugas di bidang IT Support, Praktikan tidak hanya membantu pengelolaan aplikasi E-TEKIND, tetapi juga berperan aktif dalam proses pengembangan dan optimalisasi sistem. Praktikan dituntut untuk memahami arsitektur dan mekanisme kerja Odoo, melakukan pengembangan API sesuai standar Odoo 17, menguasai dasar-dasar pengembangan aplikasi menggunakan Flutter, serta bagaimana cara *upgrade* android untk versi terbaru.

3.2 Pelaksanaan Kerja

Selama melaksanakan KP (Kerja Profesi) di PT Jaya Teknik Indonesia, Praktikan diberikan berbagai tugas secara bertahap selama periode magang oleh mentor tim IT. Praktikan ditempatkan dalam divisi IT bidang IT Support yang secara langsung terlibat dalam pengelolaan dan pengembangan sistem informasi perusahaan. Praktikan ditugaskan untuk

berkontribusi dalam pengembangan sistem internal perusahaan yaitu E-TEKIND yang berbasis Odoo, serta melakukan pembaruan aplikasi *mobile* E-TEKIND berbasis Odoo yang dibangun dengan *framework* Flutter

Odoo sendiri adalah platform ERP *open source* yang digunakan untuk memenuhi kebutuhan bisnis. Platform ini memiliki aplikasi atau modul modul yang terintegrasi dan memungkinkan semua jenis bisnis termasuk CRM, penjualan, stok, dan banyak lagi (Amalia & Syaifullah, 2024). Sedangkan Sistem *Enterprise Resources Planning* (ERP) atau sistem aplikasi terintegrasi merupakan sistem informasi yang digunakan untuk mengintegrasikan dan mengotomatisasikan proses bisnis dalam manajemen perusahaan secara transparasi dan memiliki akuntabilitas yang cukup tinggi (Indrayani, 2024). Dengan adannya sistem E-TEKIND berbasis Odoo yang diimplentasikan pada perusahaan, membuat fungsi operasional perusahaan dapat dijalankan secara lebih efisien dan terpusat dalam satu platform.

Sementara itu, pada sisi aplikasi mobile, PT Jaya Teknik Indonesia mengembangkan sistem E-TEKIND versi *mobile* menggunakan Flutter sebagai framework *Front end*, yang terintegrasi dengan sistem *Back End* berbasis Odoo. Flutter merupakan sebuah framework *open source* yang dibuat oleh Google dengan tujuan untuk membangun antarmuka pengguna (UI) yang konsisten dan menarik di berbagai platform, termasuk iOS, Android, dan web. Dengan Flutter, pengembang dapat membuat aplikasi untuk platform-platform tersebut menggunakan satu kode sumber (Ainah et al., 2024). Dengan penggunakan *framework* Flutter pada aplikasi *mobile* E-TEKIND yang mendukung *Progressive Web App* (PWA) memungkinkan integrasi antara sistem Odoo dan aplikasi *mobile* tetap selaras, serta mengikuti struktur sistem yang telah diterapkan di perusahaan.

Dengan tanggung jawab dan tugas yang diberikan kepada Praktikan, maka seluruh tugas dilaksanakan secara bertahap dan terjadwal selama magang berlangsung, yang dapat ditunjukkan pada Gantt Chart berikut:

Table 3. 1 *Gantt Chart* Kerja Profesi

No	Kegiatan	Januari				Februari				Maret				April				Mei			
		1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
1	Mentoring Odoo																				
2	*Install Depedencies* dan *Tools* Odoo 17																				
2	Pembuatan Modul *Reporting* Tagihan																				
3	Pembuatan Modul *Reporting Direct Purchase*																				
4	Pembuatan Modul *Reporting* Kuitansi																				
5	Pembuatan Modul *Reporting* Permohonan Fasilitas																				
6	Pembuatan Modul *Reporting* Penetapan Tugas & Mutasi																				
7	Pembuatan Modul *Reporting* Untuk Biaya Lembur																				
8	Pembaruan versi android untuk E-TEKIND & CMMS Tiket																				
9	Pembuatan Modul API untuk Purchase Order																				

3.2.1 Pengembangan Modul *Reporting* Pada Odoo versi 17

Pada tahap ini, sebelum Praktikan memulai fase pengkodean, Praktikan terlebih dahulu diberikan bimbingan oleh mentor terkait latihan hingga *tools* dan *dependencies* yang dibutuhkan untuk melakukan pengembangan pada Odoo 17. Berikut *tools* dan *dependencies* yang digunakan Praktikan untuk memulai tahap pengembangan pada modul *reporting* E-TEKIND berbasis Odoo 17 antara lain:

1. ***Visual Studio Code* (VSCode)**
   Praktikan menggunakan VSCode (*Visual Studio Code*) sebagai *code editor* untuk melakukan pengkodean dan pengembangan Odoo versi 17.
2. **DBeaver & PostgreSQL**
   Praktikan menggunakan DBeaver sebagai DBMS (*Database Management System*) dan PostgreSQL sebagai Database dari Odoo 17.
3. **Bahasa Pemrograman Python**
   Disini Praktikan menggunakan bahasa pemrograman Python versi 3.10.11 sebagai *requirement back end* yang dibutuhkan untuk menjalankan Odoo versi 17.
4. **XML**
   Praktikan menggunakan *eXtensible Markup Language* (XML) sebagai *requirement front end* yang digunakan oleh Odoo versi 17.
5. **Wkhtmltopdf**
   Praktikan menggunakan wkhtmltopdf yang merupakan sebuah alat *open source* yang dibutuhkan oleh Odoo versi 17, yang berfungsi untuk mengkonversi kode HTML menjadi PDF.
6. **XlsWritter**
   Praktikan menggunakan XlsWritter untuk mengkonversi data menjadi format Microsoft Excel untuk kebutuhan laporan.
7. **PDF *Print Preview***
   Praktikan juga mendownload plugin PDF *Print Preview* yang sudah disediakan Odoo, plugin ini berguna sebagai contoh tampilan dokumen sebelum mulai mencetak kedalam bentuk PDF.

Selama proses pengembangan modul reporting pada Odoo 17, Praktikan memanfaatkan tujuh *tools* dan *dependencies* utama seperti diatas. Praktikan berhasil mengembangkan enam modul laporan, yang meliputi Tagihan, *Direct Purchase*, Kuitansi, Surat Penetapan Tugas & Mutasi, Surat Permohonan Fasilitas, serta Daftar Biaya Lembur. Fokus pengembangan yang Praktikan lakukan berada pada sisi *front end* untuk template PDF dan Excel ditiap masing-masing laporan, sedangkan data yang digunakan pada tahap ini masih bersifat dummy yang sudah disediakan oleh Odoo guna mendukung simulasi dan uji tampilan.

3.2.1.1 Modul *Reporting* Tagihan

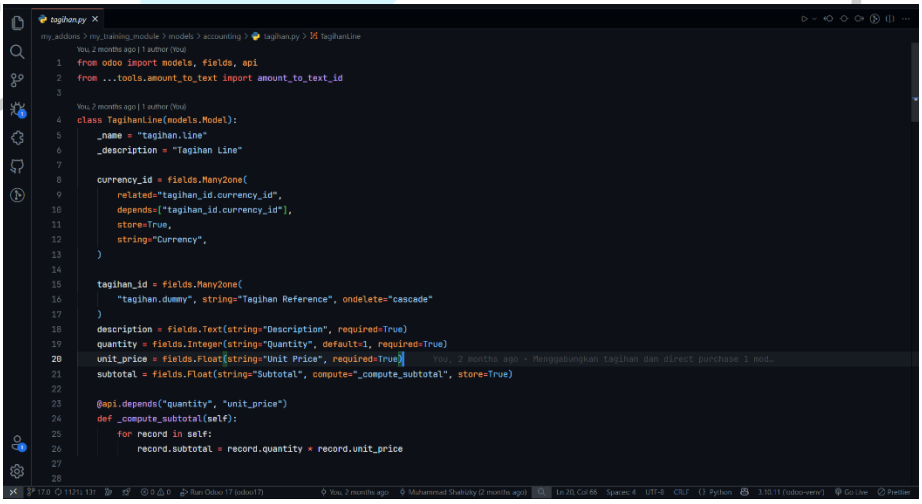
Modul *Reporting* Tagihan di Odoo 17 dirancang untuk menghasilkan laporan tagihan dalam format PDF yang terstruktur. Modul ini berfungsi untuk mencatat detail tagihan kepada pelanggan, termasuk nomor tagihan, nama pelanggan, item yang dibeli, subtotal, pajak, dan total tagihan. Proses pengembangan pada modul ini dapat mencangkup:

1. **Instalasi Modul *Accounting***

   Modul *Accounting* diinstal untuk mengambil data *dummy* yang diperlukan terkait tagihan.

2. **Pembuatan Model Python**

   Praktikan membuat dua model utama yaitu TagihanDummy untuk menyimpan informasi terkait tagihan. Dan TagihanLine untuk mencatat detail item pada setiap tagihan.



Gambar 3.1 Kode Model TagihanLine

```python
class TagihanDummy(models.Model):
    _name = "tagihan.dummy"
    _description = "Tagihan Dummy"

    # Tambahkan field currency_id
    currency_id = fields.Many2one(
        "res.currency",
        string="Currency",
        required=True,
        default=lambda self: self.env.user.company_id.currency_id,
    )

    amount_text = fields.Char(
        string="Amount in Text", compute="_compute_amount_text", store=True
    )

    # Informasi Tagihan
    tagihan_number = fields.Char(
        string="Nomor Kwitansi", required=True, readonly=True, default="/"
    )

    tagihan_date = fields.Date(string="Tagihan Date", required=True)
    due_date = fields.Date(string="Due Date")
    customer_name = fields.Char(string="Customer Name", required=True)
    customer_address = fields.Text(string="Customer Address")
    customer_tax_id = fields.Char(string="Customer Tax ID")

    # Relasi ke Detail Barang/Jasa
```

Gambar 3.2 Kode Model TagihanDummy

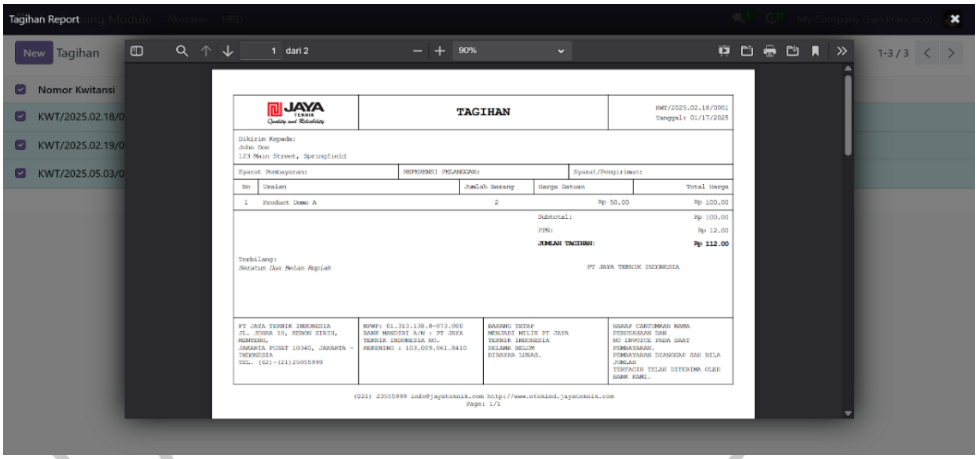
3. Pembuatan Laporan PDF dengan QWeb

Template QWeb digunakan untuk mendesain laporan dalam format HTML yang kemudian dikonversi menjadi PDF.

```xml
<odoo>
    <data>
        <template id="report_tagihan_document">
            <t t-call="web.basic_layout">
                <t t-foreach="docs" t-as="o">

                    <div class="page">
                        <!-- Header -->
                        <table class="header-table">
                            <tr>
                                <td class="logo-cell">
                                    <img t-if="res_company.logo" t-att-src="image_data_uri(res_company.logo)"/>
                                </td>
                                <td class="title-cell">
                                    TAGIHAN
                                </td>
                                <td class="info-cell">
                                    <div t-field="o.tagihan_number"/>
                                    <div>Tanggal: <span t-field="o.tagihan_date"/>
                                    </div>
                                </td>
                            </tr>
                        </table>

                        <!-- Customer Info -->
                        <table class="customer-info">
                            <tr>
                                <td>
                                    <span class="customer-label">Dikirim Kepada:</span>
                                    <br/>
```

Gambar 3.3 Kode *Front End Reporting* Tagihan

Hasil dari *Reporting* Tagihan yang dibuat oleh Praktikan pada Odoo 17 memungkinkan pembuatan laporan tagihan secara efisien dan terstruktur. Dengan menggunakan QWeb untuk desain laporan, sistem ini secara otomatis menghitung subtotal, pajak, dan total tagihan. Laporan yang dihasilkan disajikan dalam format PDF yang siap dicetak atau disimpan seperti yang dapat dilihat pada gambar 3.4



Gambar 3.4 Tampilan PDF *Report* Tagihan

3.2.1.2 Modul *Reporting Direct Purchase*

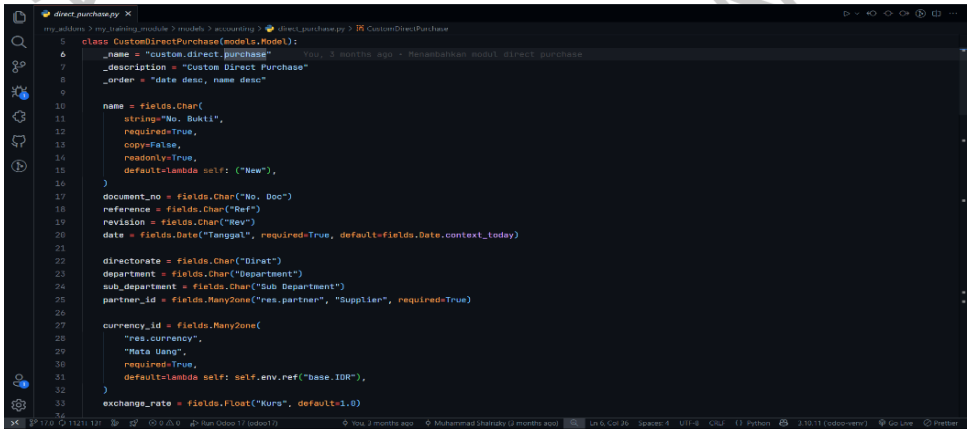
Pada tugas berikutnya, Praktikan diberikan tanggung jawab untuk mengembangkan Modul *Reporting Direct Purchase* pada Odoo 17. Modul ini dirancang untuk menghasilkan laporan pembelian langsung yang terstruktur, yang mencakup detail pembelian, informasi pemasok, item yang dibeli, serta total harga yang harus dibayar. Adapun proses pengembangan untuk modul *Reporting Direct Purchase* ini mencangkup:

1. **Instalasi Modul *Purchase***

   Instalasi Modul *Purchase* ini digunakan untuk mengambil data *dummy* yang akan dipergunakan untuk modul Direct Purchase.

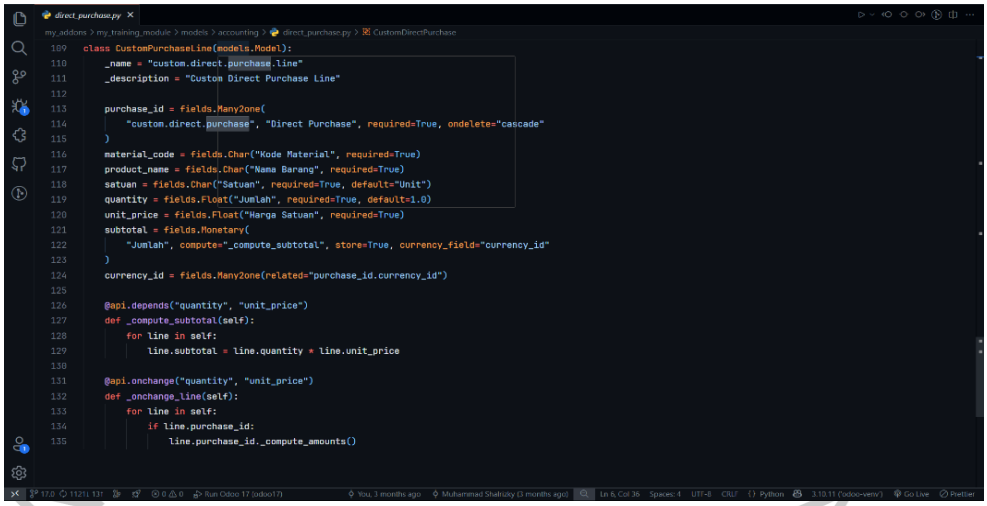
2. **Pembuatan Model Python**

   Praktikan membuat dua model utama yaitu CustomDirectPurchase untuk mencatat data pembelian dan CustomPurchaseLine untuk mencatat item pembelian.



```python
class CustomDirectPurchase(models.Model):
    _name = "custom.direct.purchase"
    _description = "Custom Direct Purchase"
    _order = "date desc, name desc"

    name = fields.Char(
        string="No. Bukti",
        required=True,
        copy=False,
        readonly=True,
        default=lambda self: ("New"),
    )
    document_no = fields.Char("No. Doc")
    reference = fields.Char("Ref")
    revision = fields.Char("Rev")
    date = fields.Date("Tanggal", required=True, default=fields.Date.context_today)

    directorate = fields.Char("Dirat")
    department = fields.Char("Department")
    sub_department = fields.Char("Sub Department")
    partner_id = fields.Many2one("res.partner", "Supplier", required=True)

    currency_id = fields.Many2one(
        "res.currency",
        "Mata Uang",
        required=True,
        default=lambda self: self.env.ref("base.IDR"),
    )
    exchange_rate = fields.Float("Kurs", default=1.0)
```

Gambar 3.5 Kode Model CustomDircetPurchase

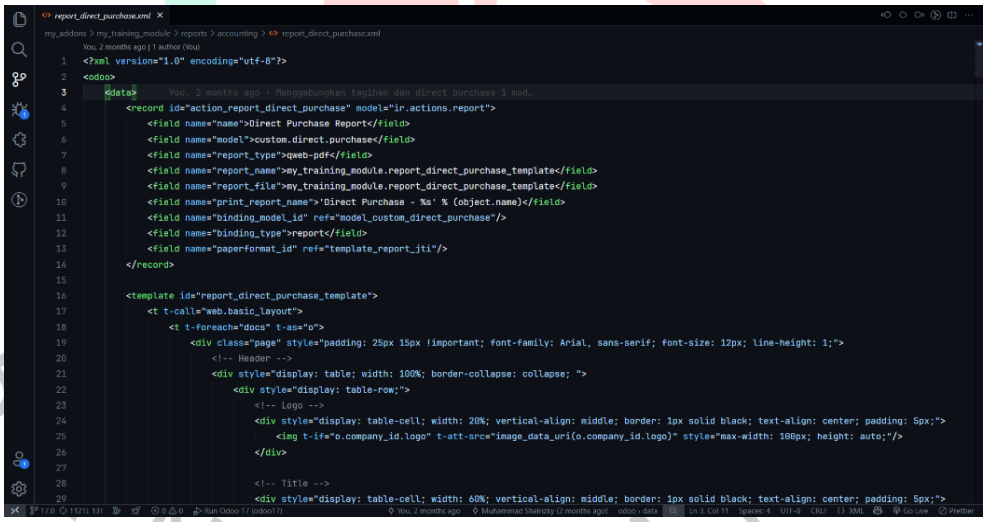


```python
class CustomPurchaseLine(models.Model):
    _name = "custom.direct.purchase.line"
    _description = "Custom Direct Purchase Line"

    purchase_id = fields.Many2one(
        "custom.direct.purchase", "Direct Purchase", required=True, ondelete="cascade"
    )
    material_code = fields.Char("Kode Material", required=True)
    product_name = fields.Char("Nama Barang", required=True)
    satuan = fields.Char("Satuan", required=True, default="Unit")
    quantity = fields.Float("Jumlah", required=True, default=1.0)
    unit_price = fields.Float("Harga Satuan", required=True)
    subtotal = fields.Monetary(
        "Jumlah", compute="_compute_subtotal", store=True, currency_field="currency_id"
    )
    currency_id = fields.Many2one(related="purchase_id.currency_id")

    @api.depends("quantity", "unit_price")
    def _compute_subtotal(self):
        for line in self:
            line.subtotal = line.quantity * line.unit_price

    @api.onchange("quantity", "unit_price")
    def _onchange_line(self):
        for line in self:
            if line.purchase_id:
                line.purchase_id._compute_amounts()
```

Gambar 3.6 Kode Model CustomPurchaseLine

3. Pembuatan Laporan PDF dengan QWeb

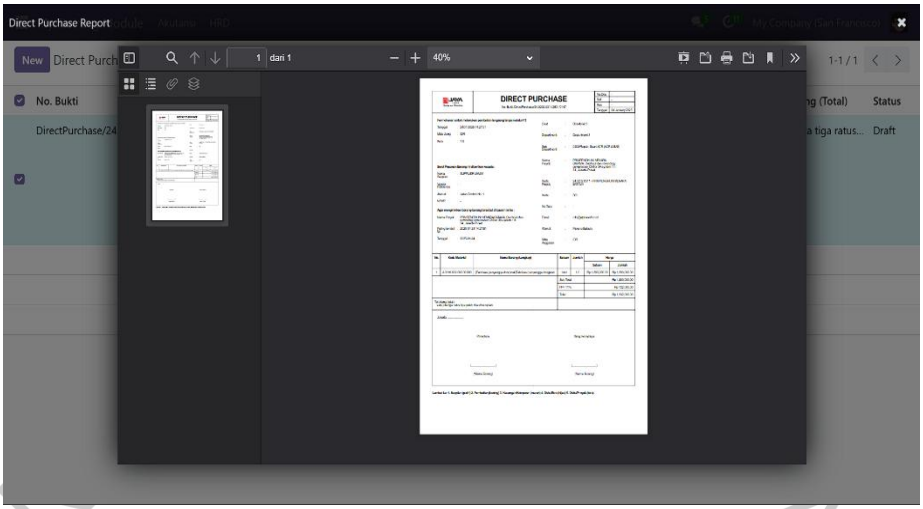
Praktikan membuat template Qweb dalam XML untuk menghasilkan laporan pembelian dalam format PDF yang mencakup informasi pembelian dan detail produk.



```xml
<?xml version="1.0" encoding="utf-8"?>
<odoo>
    <data>
        <record id="action_report_direct_purchase" model="ir.actions.report">
            <field name="name">Direct Purchase Report</field>
            <field name="model">custom.direct.purchase</field>
            <field name="report_type">qweb-pdf</field>
            <field name="report_name">my_training_module.report_direct_purchase_template</field>
            <field name="report_file">my_training_module.report_direct_purchase_template</field>
            <field name="print_report_name">'Direct Purchase - %s' % (object.name)</field>
            <field name="binding_model_id" ref="model_custom_direct_purchase"/>
            <field name="binding_type">report</field>
            <field name="paperformat_id" ref="template_report_jti"/>
        </record>

        <template id="report_direct_purchase_template">
            <t t-call="web.basic_layout">
                <t t-foreach="docs" t-as="o">
                    <div class="page" style="padding: 25px 15px !important; font-family: Arial, sans-serif; font-size: 12px; line-height: 1;">
                        <!-- Header -->
                        <div style="display: table; width: 100%; border-collapse: collapse; ">
                            <div style="display: table-row;">
                                <!-- Logo -->
                                <div style="display: table-cell; width: 20%; vertical-align: middle; border: 1px solid black; text-align: center; padding: 5px;">
                                    <img t-if="o.company_id.logo" t-att-src="image_data_uri(o.company_id.logo)" style="max-width: 100px; height: auto;"/>
                                </div>

                                <!-- Title -->
                                <div style="display: table-cell; width: 60%; vertical-align: middle; border: 1px solid black; text-align: center; padding: 5px;">
```

Gambar 3.7 Kode *Front End Reporting* Tagihan

Modul *Reporting Direct Purchase* pada Odoo 17 memungkinkan pembuatan laporan pembelian langsung yang efisien dan terstruktur dengan menggunakan QWeb. Sistem ini otomatis menghitung subtotal dan total pembelian. Laporan disajikan dalam format PDF yang mencakup informasi lengkap tentang pemasok, barang, jumlah, harga satuan, dan total harga yang harus dibayar seperti yang terlihat pada gambar 3.8



Gambar 3.8 Tampilan PDF *Reporting Direct Purchase*

3.2.1.3 Modul *Reporting* Kuitansi

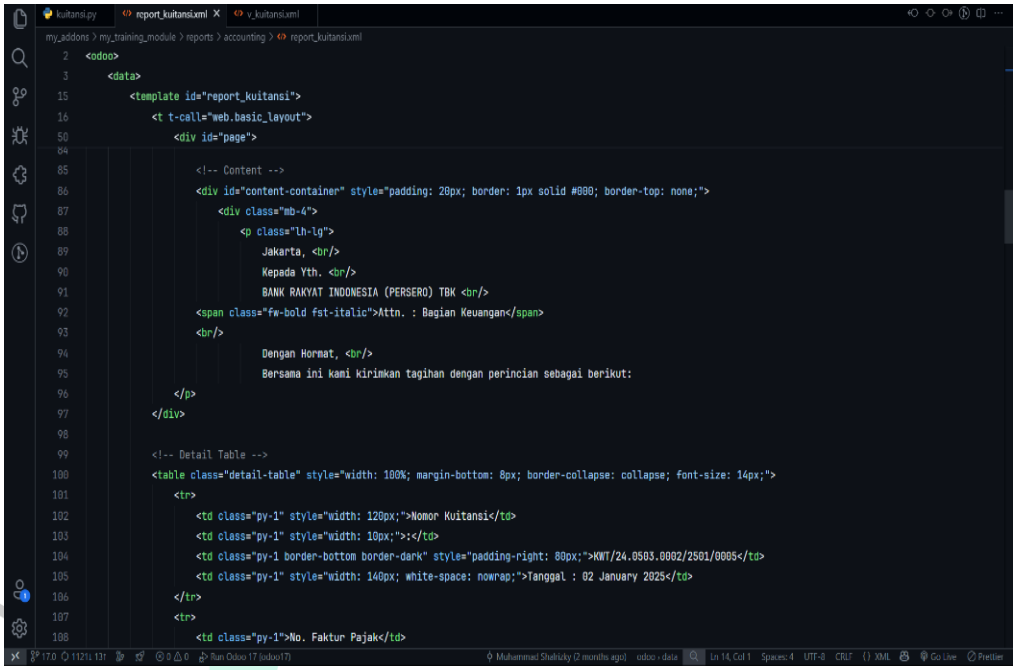
Berbeda dari kedua report sebelmunya, pada modul *Reporting Kuitansi* ini Praktikan membuat template PDF statis yang hanya menggunakan QWeb atau XML sebagai *front end* tanpa mengambil data *dummy* dari Odoo seperti sebelumnya. Modul ini dirancang untuk menghasilkan laporan kuitansi dalam format PDF, tujuannya agar dapat digunakan untuk transaksi yang melibatkan pembayaran atau penerimaan. Proses pengembangannya meliputi:

1. **Pembuatan Template QWeb untuk Laporan PDF**

   Praktikan membuat template QWeb berbasis XML seperti sebelumnya yang mendefinisikan bagaimana laporan kuitansi akan ditampilkan dalam format PDF. Template ini berisi informasi statis seperti nomor kuitansi, tanggal, jumlah pembayaran, dan informasi terkait lainnya.

2. **Pembuatan Data Statis Pada XML**

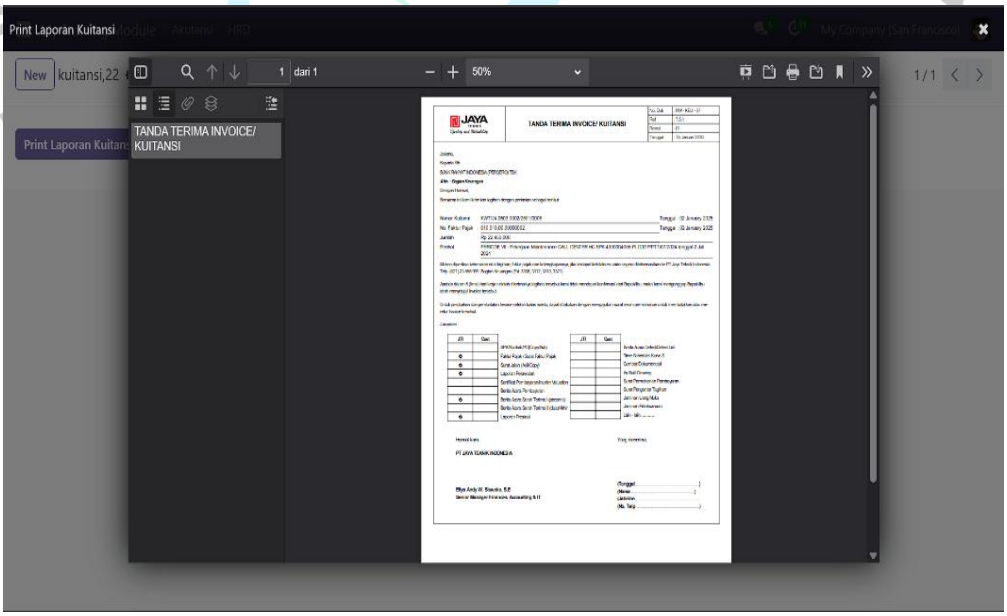
   Karena tidak menggunakan model Odoo atau data yang diambil dari sistem, semua data yang akan digunakan untuk laporan kuitansi didefinisikan langsung di dalam template XML. Ini berarti data seperti nomor kuitansi, tanggal, dan jumlah pembayaran diinput secara langsung ke dalam template seperti yang dapat terlihat pada gambar 3.9 dibawah:



```
<odoo>
    <data>
        <template id="report_kuitansi">
            <t t-call="web.basic_layout">
                <div id="page">

                    <!-- Content -->
                    <div id="content-container" style="padding: 20px; border: 1px solid #000; border-top: none;">
                        <div class="mb-4">
                            <p class="lh-lg">
                                Jakarta, <br/>
                                Kepada Yth. <br/>
                                BANK RAKYAT INDONESIA (PERSERO) TBK <br/>
                    <span class="fw-bold fst-italic">Attn. : Bagian Keuangan</span>
                    <br/>
                                Dengan Hormat, <br/>
                                Bersama ini kami kirimkan tagihan dengan perincian sebagai berikut:
                    </p>
                </div>

                <!-- Detail Table -->
                <table class="detail-table" style="width: 100%; margin-bottom: 8px; border-collapse: collapse; font-size: 14px;">
                    <tr>
                        <td class="py-1" style="width: 120px;">Nomor Kuitansi</td>
                        <td class="py-1" style="width: 10px;">:</td>
                        <td class="py-1 border-bottom border-dark" style="padding-right: 80px;">KWT/24.0503.0002/2501/0005</td>
                        <td class="py-1" style="width: 140px; white-space: nowrap;">Tanggal : 02 January 2025</td>
                    </tr>
                    <tr>
                        <td class="py-1">No. Faktur Pajak</td>
```

Gambar 3.9 Data Statis *Reporting* Kuitansi

Modul *Reporting* Kuitansi di Odoo 17 dirancang untuk menghasilkan laporan kuitansi dalam format PDF yang mencakup informasi seperti nomor kuitansi, jumlah pembayaran, dan perihal. Berbeda dengan dua modul sebelumnya, modul ini menggunakan data statis dalam template tanpa memerlukan model kompleks. Hasil dari laporan ini dapat dilihat pada gambar 3.10 dibawah:



Gambar 3.10 Tampilan PDF *Reporting* Kuitansi

3.2.1.4 Modul *Reporting* Permohonan Fasilitas

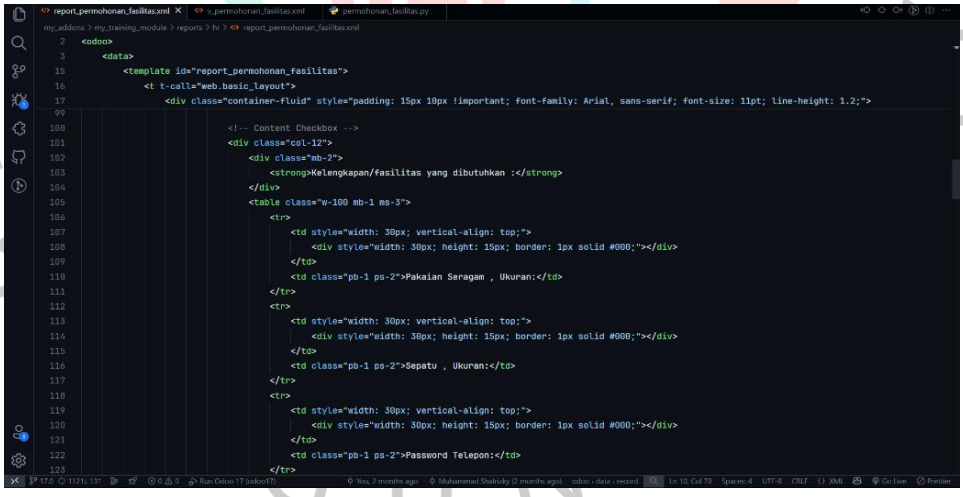
Sama seperti pada Modul *Reporting* Kuitansi, pada Modul *Reporting* Surat Permohonan Fasilita ini Praktikan hanya membuat *front end* untuk laporan dalam format PDF yang berisi informasi terkait permohonan fasilitas untuk karyawan baru menggunakan data statis. Pengembangan yang dilakukan juga mencangkup:

1. **Pembuatan Template QWeb untuk Laporan PDF**

   Praktikan membuat template QWeb berbasis XML untuk mendefinisikan bagaimana surat permohonan fasilitas akan ditampilkan dalam format PDF. Template ini mencakup informasi seperti nama karyawan, NRP, tanggal masuk, dan kelengkapan fasilitas yang diminta.

2. **Pembuatan Data Statis Pada XML**

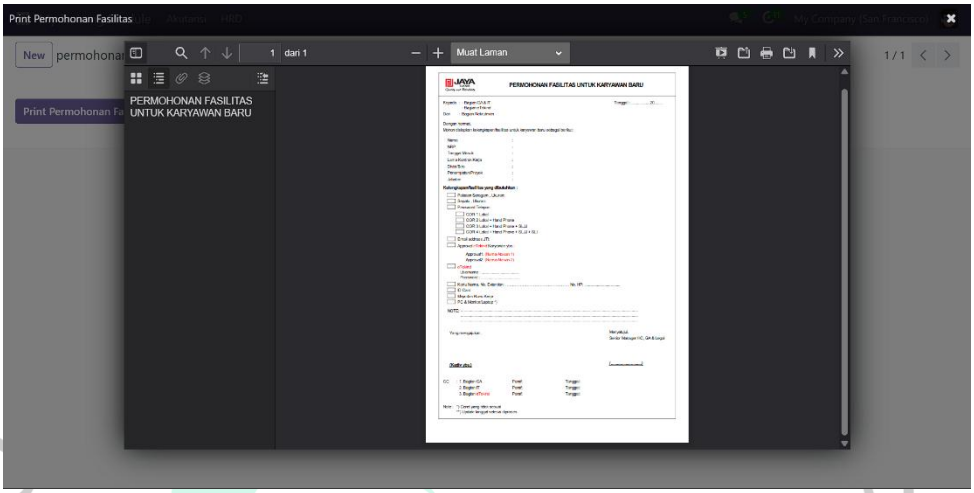
   Seperti pada modul kuitansi, laporan ini menggunakan data statis yang didefinisikan langsung di dalam template XML, tanpa mengambil data dari model Odoo. Berikut potongan kode yang menggambarkan potongan kode data static langsung pada XML.



Gambar 3.11 *Data Statis Reporting Permohonan Fasilitas*

Modul *Reporting* Permohonan Fasilitas di Odoo 17 dirancang untuk menghasilkan laporan surat permohonan fasilitas dalam format PDF yang mencakup informasi seperti nama karyawan, NRP, tanggal masuk, dan kelengkapan fasilitas yang diminta. Sama seperti pada modul *Reporting* Kuitansi, modul ini menggunakan data statis dalam template tanpa

memerlukan model kompleks. Hasil dari laporan ini dapat dilihat pada gambar 3.12 dibawah:



Gambar 3.12 Tampilan PDF *Reporting* Permohonan Fasilitas

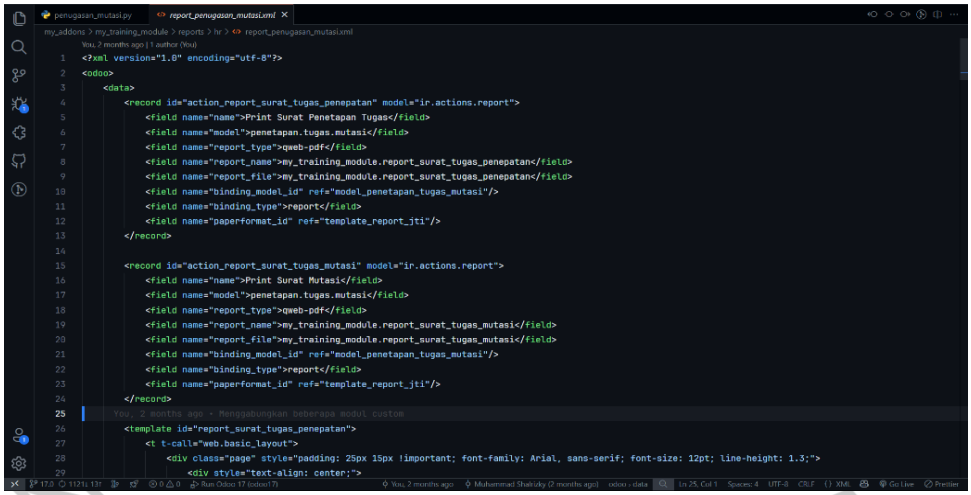
3.2.1.5 Modul *Reporting* Penetapan Tugas & Mutasi

Pada Modul *Reporting* Penetapan Tugas & Mutasi di Odoo versi 17, Praktikan bertanggung jawab untuk mengembangkan dua template PDF sekaligus yaitu template penetapan tugas dan template keputusan mutasi. Modul ini dirancang untuk menghasilkan dua jenis laporan yang memuat informasi terkait penetapan tugas dan mutasi karyawan dalam perusahaan. Proses pengembangan modul ini mencakup dua laporan yang terpisah namun saling terkait. Adapun proses pengembangan untuk modul ini dapat mencangkup:

1. **Pembuatan Template QWeb untuk Laporan PDF**

   Praktikan membuat dua template QWeb berbasis XML, masing-masing untuk Template Penetapan Tugas dan Template Keputusan Mutasi. Kedua template ini akan ditampilkan dalam format PDF.

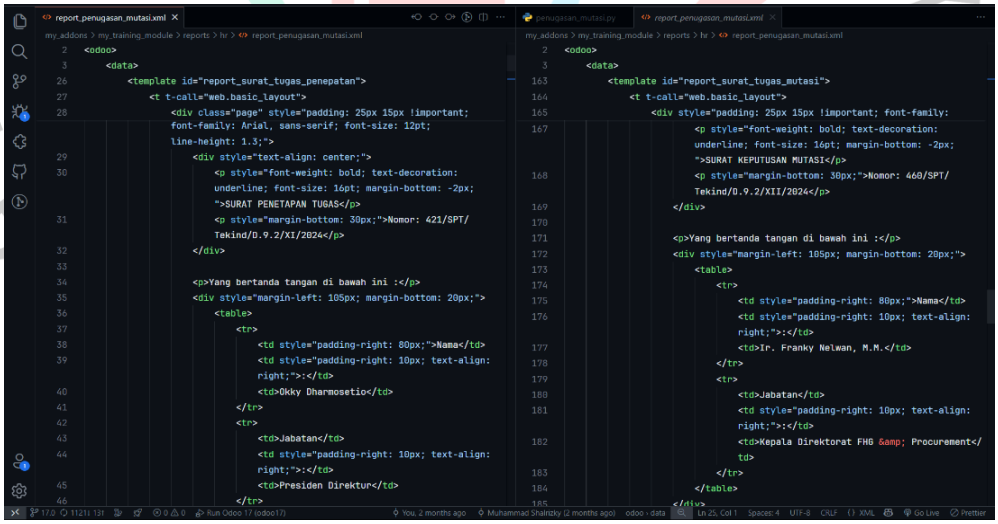
   - Template Penetapan Tugas mencakup informasi seperti nama karyawan, jabatan, divisi lama dan baru, serta tanggal berlaku.
   - Template Keputusan Mutasi berisi informasi terkait mutasi karyawan, termasuk nama, jabatan, divisi lama dan baru, serta tanggung jawab baru.



Gambar 3.13 Kode *Record Action Reporting* Penetapan Tugas & Mutasi

2. Pembuatan Data Statis Pada XML

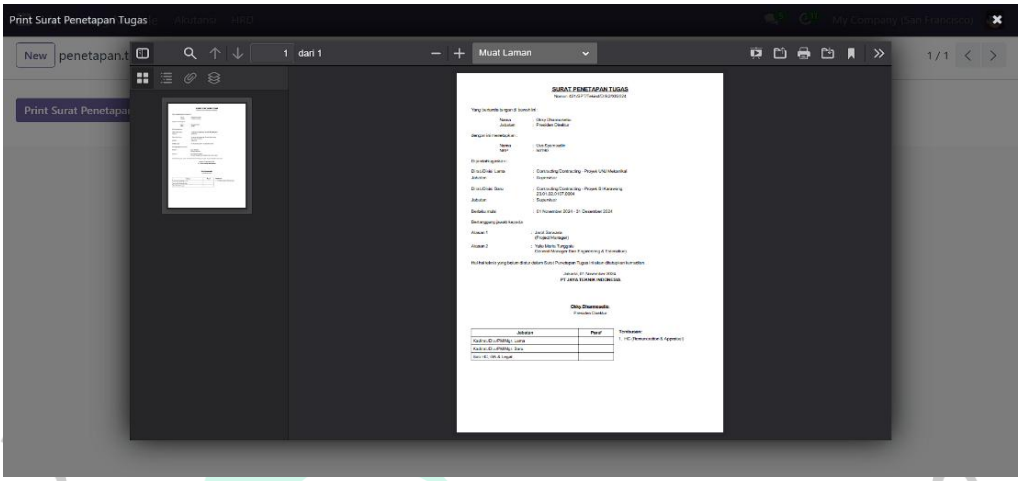
Seperti pada modul sebelumnya, laporan ini menggunakan data statis yang didefinisikan langsung di dalam template XML. Ini berarti informasi seperti nama karyawan, jabatan, divisi, dan tanggal ditulis langsung ke dalam template tanpa bergantung pada model Odoo atau data dinamis pada kedua template tersebut seperti gambar 3.14 diabwah:



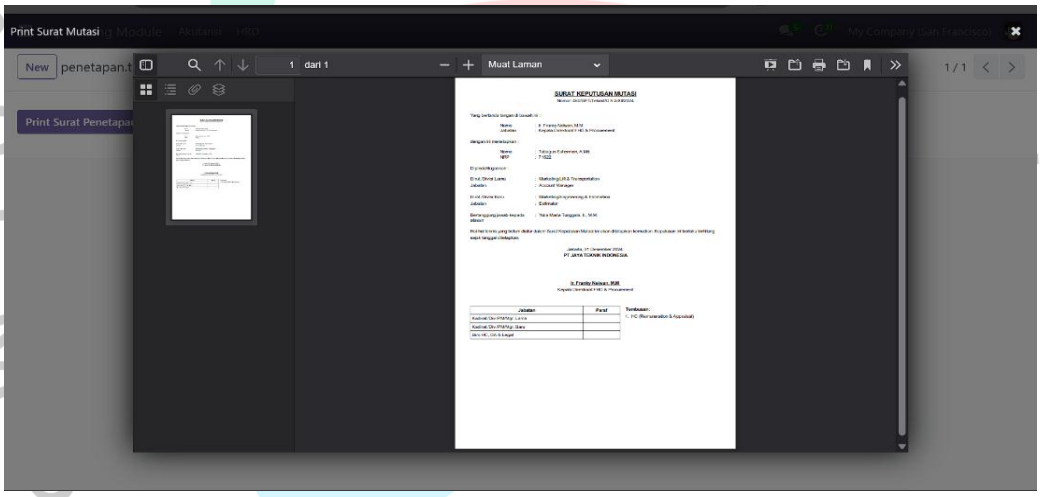
Gambar 3.14 Kode Template *Reporting* Penetapan Tugas & Mutasi

Modul Reporting Penetapan Tugas & Mutasi di Odoo dirancang untuk menghasilkan dua laporan PDF yaitu template penetapan tugas dan template keputusan mutasi. Praktikan membuat dua template QWeb berbasis XML untuk kedua template tersebut, masing-masing mencakup

informasi terkait penetapan tugas dan mutasi karyawan. Hasil dari pengembangan *reporting* ini dapat dilihat pada gambar dibawah ini:



Gambar 3.16 Tampilan PDF *Reporting* Penetapan Tugas



Gambar 3.15 Tampilan PDF *Reporting* Mutasi

3.2.1.6 Modul *Reporting* Biaya Lembur

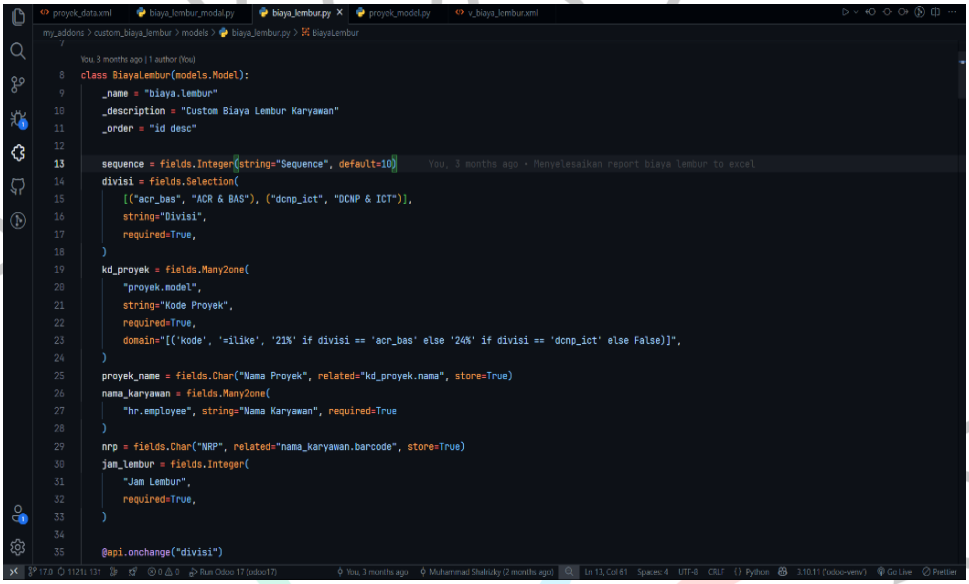
Modul Biaya Lembur di Odoo ini berbeda dengan modul sebelumnya karena laporan yang dihasilkan berupa format Excel. Modul ini memungkinkan pengguna untuk mengelola biaya lembur karyawan berdasarkan proyek dan divisi, serta Praktikan ditugaskan untuk membuat fitur *filter* agar mempermudah pencarian data sesuai kriteria yang diinginkan, seperti divisi, kode proyek, dan nama karyawan. Berikut proses pengembangan pada modul *Reporting* Biaya Lembur sebagai berikut:

1. **Instalasi Modul *Employees***

   Praktikan menginstall modul *employees* dari Odoo versi 17 untuk kebutuhan mengambil data dummy karyawan yang sudah tersedia.

2. **Pembuatan Model Python**

   Praktikan membuat model BiayaLembur untuk menyimpan informasi terkait biaya lembur, yang mencakup divisi, kode proyek, nama karyawan, NRP, dan jam lembur.

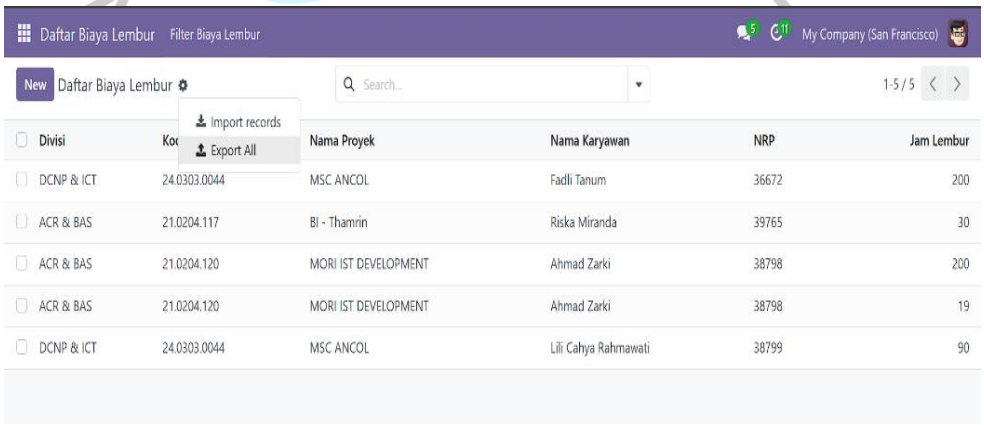


```python
class BiayaLembur(models.Model):
    _name = "biaya.lembur"
    _description = "Custom Biaya Lembur Karyawan"
    _order = "id desc"

    sequence = fields.Integer(string="Sequence", default=10)
    divisi = fields.Selection(
        [("acr_bas", "ACR & BAS"), ("dcnp_ict", "DCNP & ICT")],
        string="Divisi",
        required=True,
    )
    kd_proyek = fields.Many2one(
        "proyek.model",
        string="Kode Proyek",
        required=True,
        domain="[('kode', '=ilike', '21%' if divisi == 'acr_bas' else '24%' if divisi == 'dcnp_ict' else False)]",
    )
    proyek_name = fields.Char("Nama Proyek", related="kd_proyek.nama", store=True)
    nama_karyawan = fields.Many2one(
        "hr.employee", string="Nama Karyawan", required=True
    )
    nrp = fields.Char("NRP", related="nama_karyawan.barcode", store=True)
    jam_lembur = fields.Integer(
        "Jam Lembur",
        required=True,
    )

    @api.onchange("divisi")
```

Gambar 3.17 Kode Model BiayaLembur

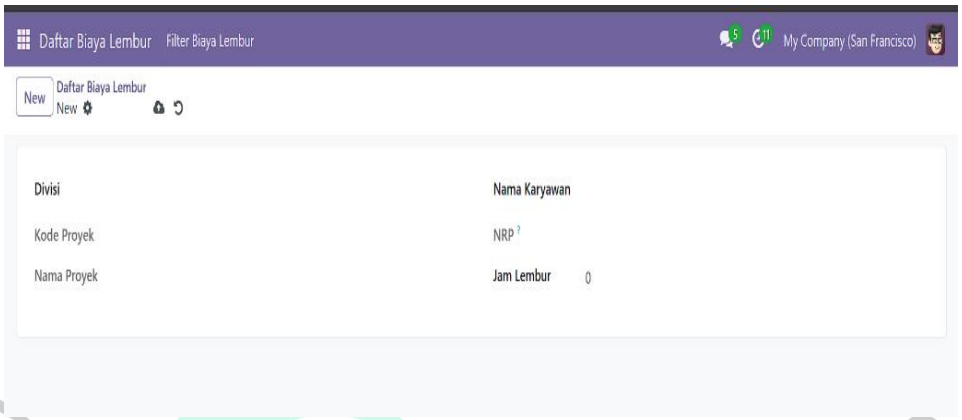
3. **Pembuatan Tampilan *Tree View* dan *Form***

   Praktikan membuat tampilan *Tree View* untuk menampilkan data biaya lembur yang sudah terinput, dengan tombol untuk mengekspor data ke Excel.



Gambar 3.18 Tampilan *Tree View*

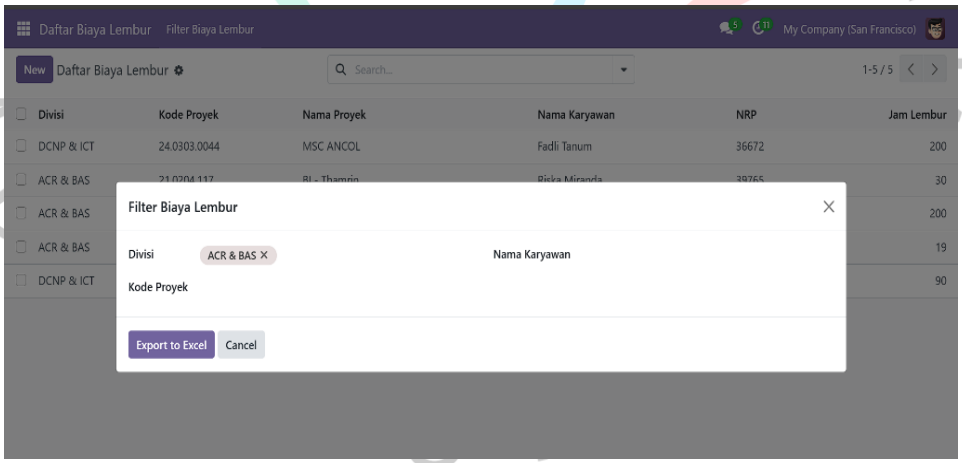
Praktikan juga membuat *Form View* untuk memasukkan data biaya lembur karyawan seperti yang dapat terlihat pada gambar 3.19 dibawah.



Gambar 3.19 Tampilan *Form View*

4. Pembuatan Modal *Filter* Biaya Lembur

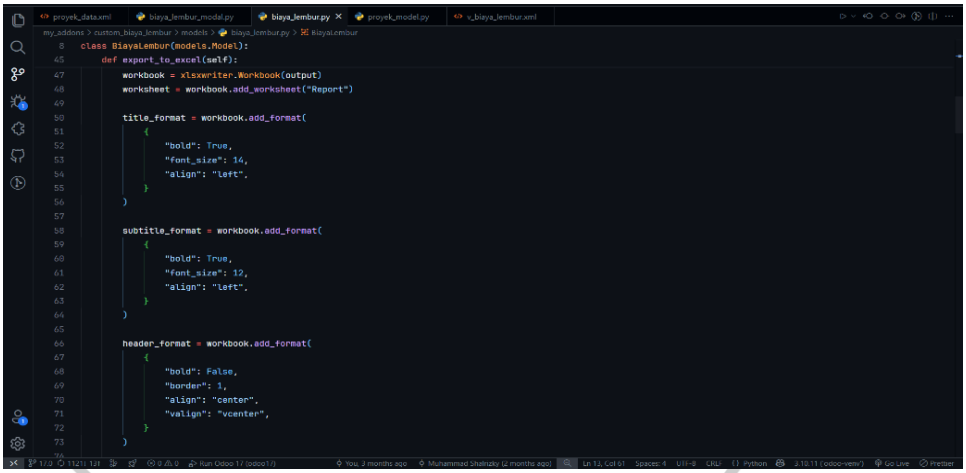
Pada modul ini Praktikan juga membuat fitur modal *filter* yang memungkinkan pengguna memilih kriteria *filter* seperti divisi, kode proyek, dan nama karyawan sebelum mengunduh laporan dalam format Excel. Contoh Tampilan Modal Filter di Odoo:



Gambar 3.20 Modal *Filter* Biaya Lembur

5. Ekspor Data ke Excel

Setelah memfilter data, laporan biaya lembur dapat diekspor ke dalam format Excel menggunakan xlsxwriter untuk mencatat rincian biaya lembur dan menghitung total jam lembur per divisi atau proyek.



```
class BiayaLembur(models.Model):
    def export_to_excel(self):
        workbook = xlsxwriter.Workbook(output)
        worksheet = workbook.add_worksheet("Report")

        title_format = workbook.add_format(
            {
                "bold": True,
                "font_size": 14,
                "align": "left",
            }
        )

        subtitle_format = workbook.add_format(
            {
                "bold": True,
                "font_size": 12,
                "align": "left",
            }
        )

        header_format = workbook.add_format(
            {
                "bold": False,
                "border": 1,
                "align": "center",
                "valign": "vcenter",
            }
        )
```

Gambar 3.21 Kode *Export To Excel* Biaya Lembur

Modul Biaya Lembur berhasil dikembangkan dengan data karyawan yang diambil dari modul *Employees* yang telah Praktikan instal sebelumnya. Modul ini mencakup *form view* untuk memasukkan data dan *tree view* untuk menampilkan data biaya lembur. Fitur *filtering* dan ekspor Excel menambah fleksibilitas dalam pengelolaan data lembur. Berikut hasil dari laporan Biaya Lembur Karyawan yang telah di export ke excel:

Biaya Lembur Beban Proyek Langsung

Bulan May 2025

No	Divisi/Kode Proyek	Nama Proyek	Nama Karyawan	NRP	Jam Lembur
	ACR & BAS				249
1	21.0204.117	BI - Thamrin	Riska Miranda	39765	30
2	21.0204.120	MORI IST DEVELOPMENT	Ahmad Zarki	38798	200
			Ahmad Zarki	38798	19
	DCNP & ICT				290
3	24.0303.0044	MSC ANCOL	Fadli Tanum	36672	200
			Lili Cahya Rahmawati	38799	90
	Grand Total				539

Gambar 3.22 Hasil *Reporting* Biaya Lembur

3.2.2 Pembaruan Versi Android Untuk Aplikasi *Mobile* E-TEKIND & CMMS Tiket

Pada tahap ini, Praktikan melakukan pembaruan versi Android untuk aplikasi *mobile* E-TEKIND dan CMMS Tiket, yang sebelumnya sudah dikembangkan menggunakan Flutter dan berbasis Odoo versi 11. Aplikasi E-TEKIND menggunakan konsep PWA (*Progressive Web App*) yang terhubung langsung dengan Odoo versi 11, sementara CMMS Tiket adalah aplikasi yang dikhususkan untuk pelanggan PT Jaya Teknik Indonesia. Sebelumnya, aplikasi ini hanya kompatibel dengan perangkat Android versi

lama dan tidak dapat dipublikasikan di Google Play Store akibat masalah kompatibilitas.

Praktikan diberikan tugas untuk memperbarui kedua aplikasi ini agar dapat berjalan dengan optimal di perangkat Android terbaru agar nantinya dapat dipublikasikan kembali di Google Play Store. Pembaruan ini melibatkan peningkatan penggunaan Flutter untuk memastikan antarmuka pengguna (UI) yang responsif dan aplikasi yang lebih stabil pada perangkat Android terbaru. Adapun *tools* yang Praktikan pakai dapat mencangkup seperti berikut:

1. **Visual Studio Code (VSCode)**

   Sama seperti sebelumnya, Praktikan menggunakan Visual Studio Code (VSCode) sebagai kode editor utama dalam pengembangan aplikasi. VSCode dipilih karena kemampuannya yang ringan, fleksibel, dan memiliki berbagai extensions yang mendukung pengembangan aplikasi dengan Flutter.

2. **Framework Flutter**

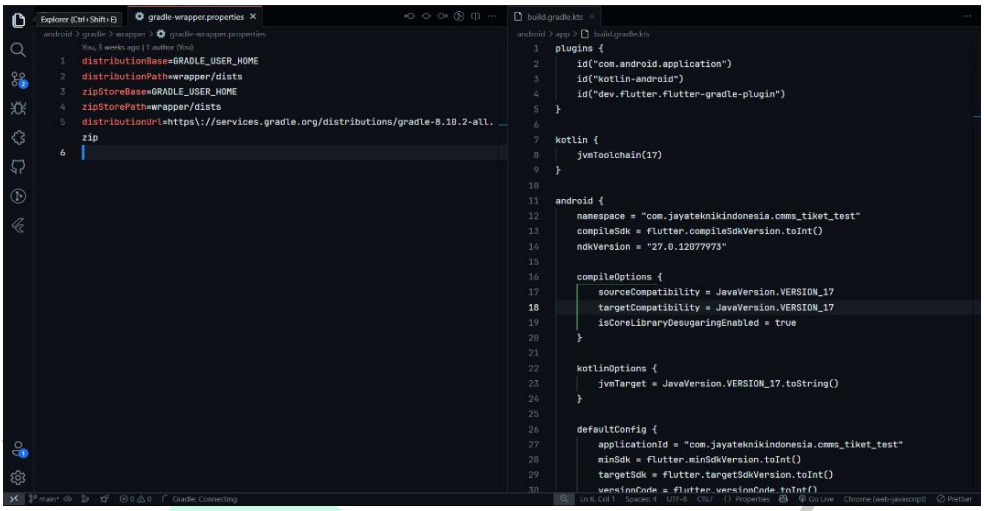
   Untuk pengembangan aplikasi mobile, Praktikan menggunakan Flutter versi terbaru, yaitu 3.29.2. Pembaruan ke versi terbaru dari Flutter membantu mengoptimalkan aplikasi agar lebih stabil, meningkatkan performanya, dan memastikan kompatibilitas dengan versi Android terbaru.

3. **Emulator Andorid**

   Untuk pengujian aplikasi, praktikan menginstal Emulator Android dengan SDK versi 34.0.0 dan menggunakan Android versi 14. Emulator Android digunakan untuk mensimulasikan perangkat Android pada komputer, sehingga Praktikan dapat menguji aplikasi tanpa memerlukan perangkat fisik.

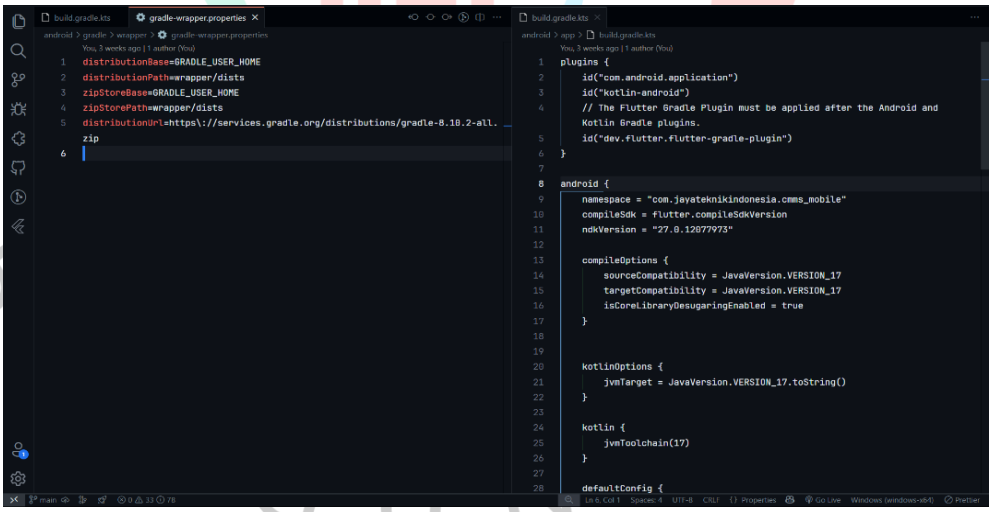
Ketiga *tools* ini memungkinkan Praktikan untuk melakukan pembaruan versi Android pada kedua aplikasi, memastikan aplikasi dapat berjalan optimal pada perangkat Android terbaru, serta nantinya siap untuk dipublikasikan kembali di Google Play Store. Adapun langkah-langkah dalam proses pembaruan dan pengembangan yang Praktikan lakukan untuk kedua aplikasi ini adalah:

1. ***Update* Gradle E-TEKIND dan CMMS Tiket**



Gambar 3.23 Kode *Update* Gradle CMMS Tiket

Pada tahap ini Praktikan melakukan pembaruan versi Gradle 8.10.2 dengan Java versi 17.0.0.1 untuk memastikan aplikasi mendapatkan *update* terbaru yang mendukung peningkatan stabilitas dan keamanan. Versi terbaru Gradle akan memudahkan pengelolaan *dependensi* dan mempercepat waktu *build* aplikasi.



Gambar 3. 24 Kode *Update* Gradle E-TEKIND

2. **Update Flutter *Framework***

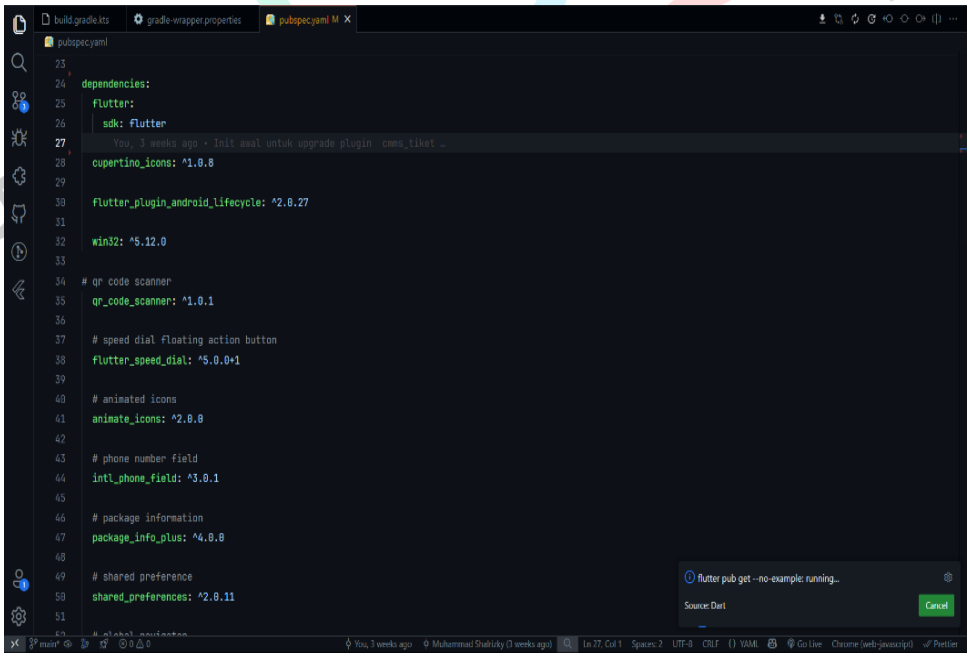
Praktikan memperbarui kedua aplikasi *mobile* dengan menggunakan Flutter versi terbaru, yakni versi 3.29.2, untuk memastikan aplikasi mendukung versi Android terbaru. Pembaruan ini juga mencakup *update* ke berbagai *plugin* dan *dependencies* yang digunakan dalam

aplikasi Flutter untuk mendukung fungsionalitas aplikasi di Android terbaru. Seperti yang dapat dilihat pada gambar 3.25 untuk spesifikasi *environment* Flutter dalam proses pembaruan kedua aplikasi tersbut.

Gambar 3.25 Spesifikasi Flutter

3. Update *Dependencies*

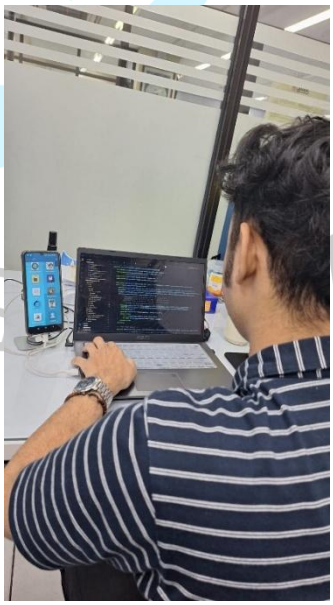
Untuk meningkatkan kompatibilitas dan kinerja aplikasi pada Android terbaru, Praktikan memperbarui *dependencies* yang digunakan dalam aplikasi mobile E-TEKIND dan CMMS Tiket. Pembaruan ini mencakup *plugin* yang berhubungan dengan tampilan UI, akses jaringan, penyimpanan lokal, dan lainnya.



Gambar 3.26 *Update Depedencies* CMMS Tiket

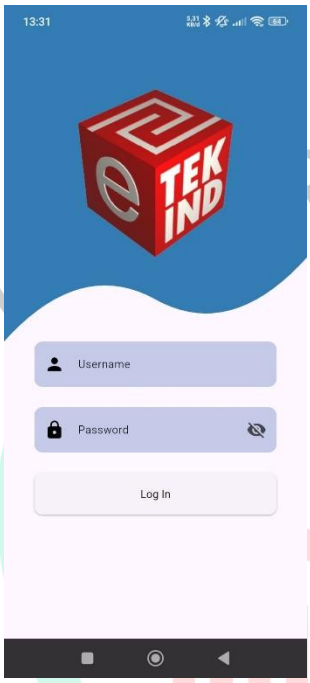
Gambar 3.27 *Update Dependencies* E-TEKIND

Pembaruan pada aplikasi E-TEKIND dan CMMS Tiket yang Praktikan kerjakan berhasil memastikan aplikasi berjalan optimal pada perangkat Android versi 7 hingga 14. Dengan memperbarui Gradle, Flutter Framework (versi 3.29.2), dan *dependencies*, aplikasi kini lebih stabil dan kompatibel dengan versi Android terbaru. Praktikan melakukan pengujian dengan menggunakan Emulator Android serta perangkat fisik untuk memastikan kinerja yang lancar. Dengan begitu Aplikasi juga siap dipublikasikan kembali di Google Play Store nantinya.

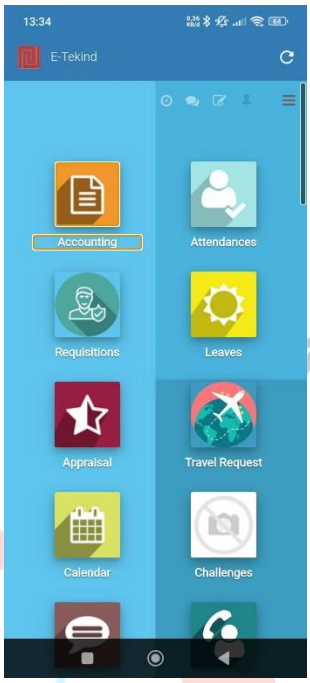


Gambar 3.28 Praktikan Memperbarui Aplikasi

- **Hasil pembaruan untuk aplikasi *mobile* E-TEKIND**



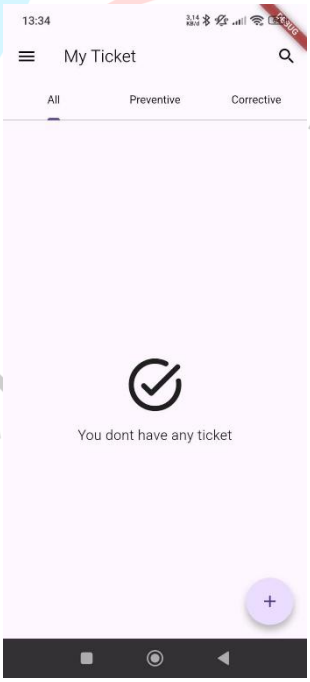
Gambar 3.29 *Login* E-TEKIND



Gambar 3.30 *Home* E-TEKIND

- **Hasil pembaruan untuk aplikasi *mobile* CMMS Tiket**

Gambar 3.31 *Login* CMMS Tiket



Gambar 3.2 *Home* CMMS Tiket

3.2.3 Modul *Custom* API Untuk Kebutuhan Odoo Versi 17

Praktikan juga diberikan tanggung jawab untuk mengembangkan modul *custom* API *Purchase Order* (PO) yang memungkinkan integrasi dengan sistem Odoo versi 17 menggunakan metode CRUD (*Create, Read, Update, Delete*). API ini dibuat untuk memudahkan pengelolaan data PO secara eksternal melalui *request* HTTP, yang didukung oleh data *dummy* untuk *user* dan *purchase order*. API yang Praktikan buat ini terdiri dari beberapa *endpoint* yang memungkinkan untuk:

- *Get Data Purchase Order*: Mengambil daftar PO dan detailnya.
- *Create Data Purchase Order*: Membuat PO baru berdasarkan data yang dikirimkan.
- *Update Data Purchase Order*: Memperbarui data PO yang sudah ada.
- *Delete Data Purchase Order*: Menghapus PO yang sudah dibatalkan.

Proses tersebut memungkinkan integrasi langsung dengan Odoo 17 melalui token *authentication* yang Praktikan buat untuk memastikan keamanan setiap request yang dilakukan. Setiap *request* API akan memverifikasi token untuk memastikan bahwa hanya pengguna yang sah dan sudah terdaftar pada database res.user yang dapat mengakses dan mengubah data. Adapun *tools* dan *dependencies* yang Praktikan gunakan dalam membuat API ini antara lain:

1. **Modul *Purchase* (Odoo)**
   Praktikan menggunakan modul *Purchase* di Odoo untuk mengakses data *dummy* dan mengelola data *Purchase Order* (PO), seperti mengambil, membuat, memperbarui, dan menghapus PO.
2. **Visual Studio Code (VSCode)**
   Praktikan menggunakan VSCode sebagai code editor untuk menulis model Python. VSCode menyediakan fitur-fitur yang memudahkan pengembangan dalam mengerjakan API PO pada Odoo 17.
3. **DBeaver & PostgreSQL**
   Praktikan menggunakan DBeaver sebagai DBMS (*Database Management System*) dan PostgreSQL sebagai Database dari Odoo 17. Kedua *tools* ini sangat berguna bagi Praktikan untuk mengembangkan modul API *Purchase Order.*

4. **Bahasa Pemrograman Python**

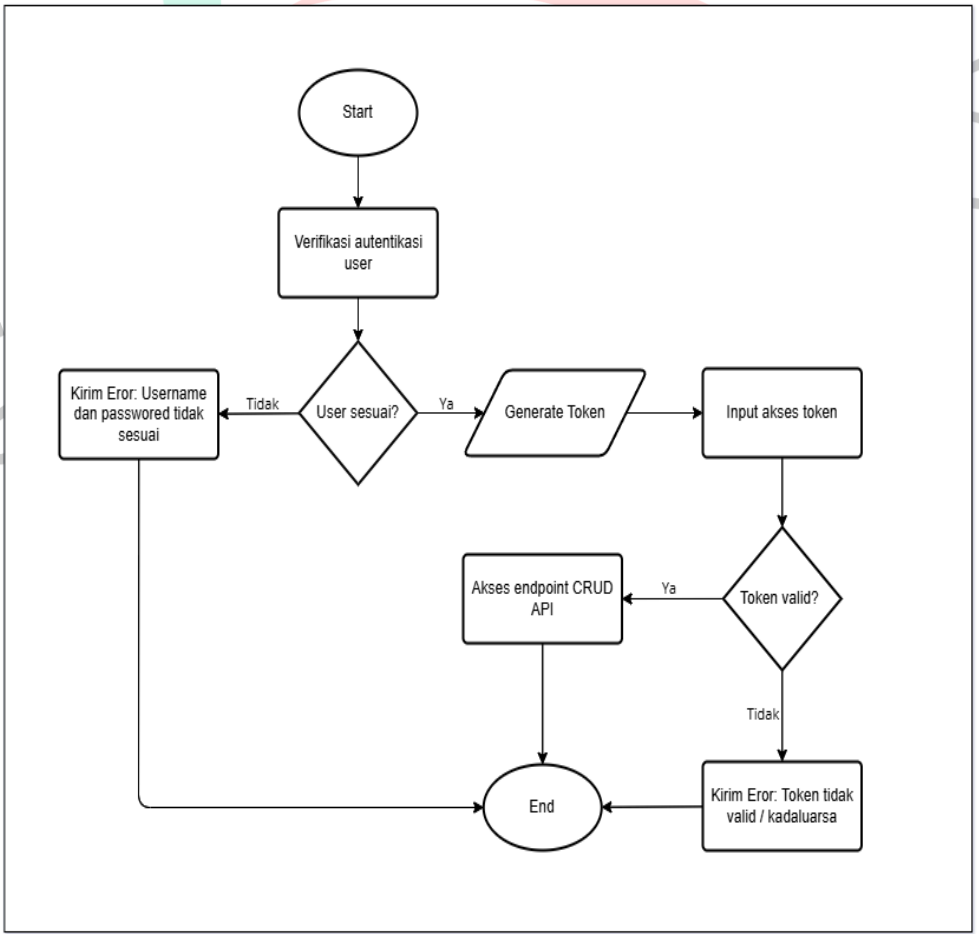
   Python digunakan untuk menulis logika API, membuat model data, dan mengelola operasi CRUD pada Purchase Order di Odoo 17.

5. **Postman**

   Praktikan menggunakan Postman untuk menguji API yang dikembangkan, termasuk memverifikasi *endpoint* API dan memastikan bahwa data dikirim dan diterima dengan benar.

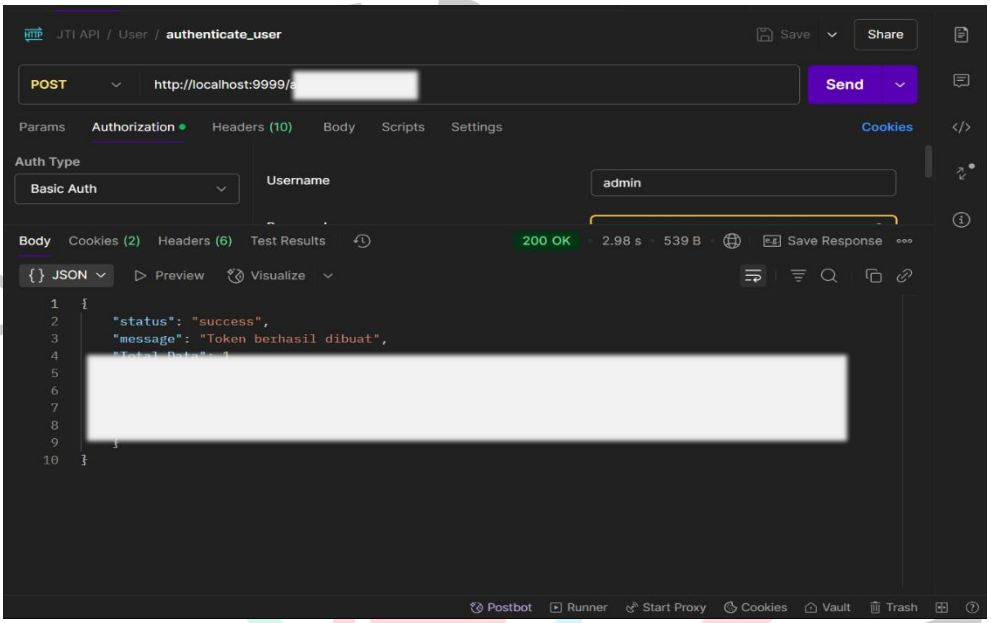
Untuk menggambarkan alur kerja API *Purchase Order*, Praktikan menggunakan *flowchart* untuk menjelaskan langkah-langkah dari setiap operasi API, mulai dari verifikasi token hingga eksekusi setiap operasi CRUD. Berikut adalah langkah-langkah yang terlibat dalam setiap operasi API yang Praktikan kembangkan:

1. **Autentikasi Token API**



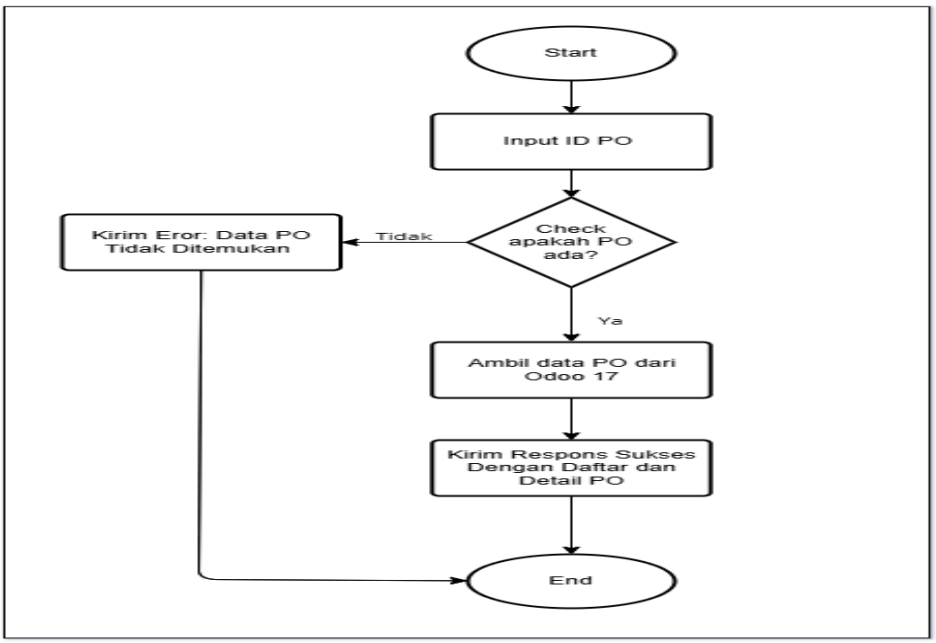
Gambar 3.31 *Flowchart Autentikasi* Token API

Seperti yang terlihat pada diagram *flowchart* 3.33 diatas, *Flowchart* ini berfungsi untuk memverifikasi token Bearer dalam *header Authorization*. Jika tidak valid, respons 401 *Unauthorized* dikembalikan, jika valid, lanjutkan *user* untuk mengaskes ke *endpoint*. Berikut hasil dari pengembangan Autentikasi API token:



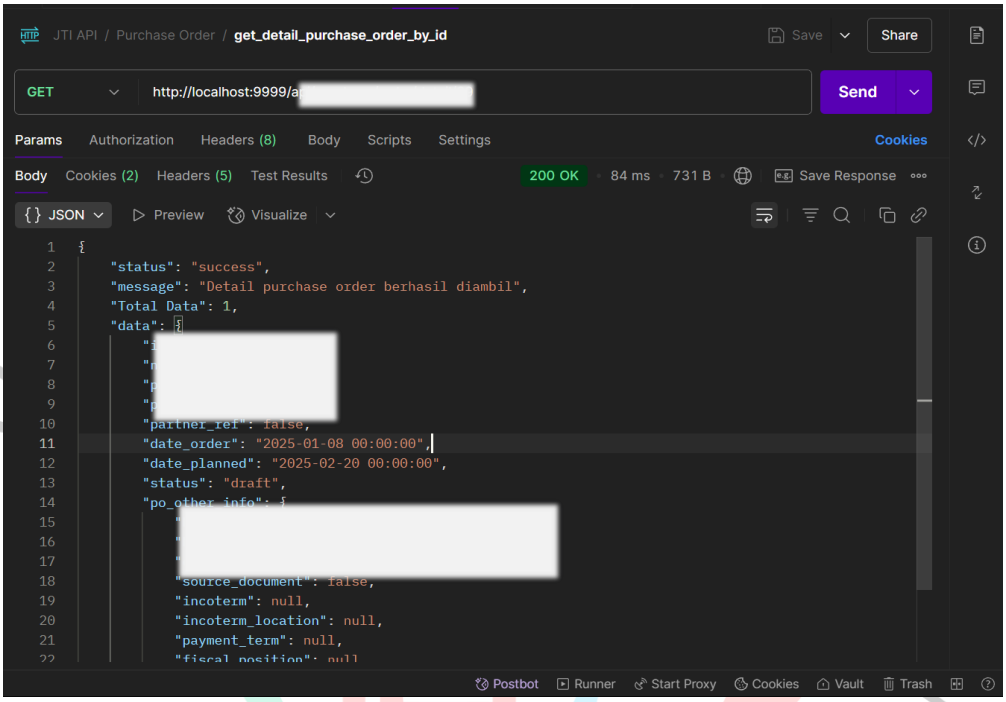
Gambar 3.32 Pengujian API Autentikasi Token

2. *Get Data Purchase Order*



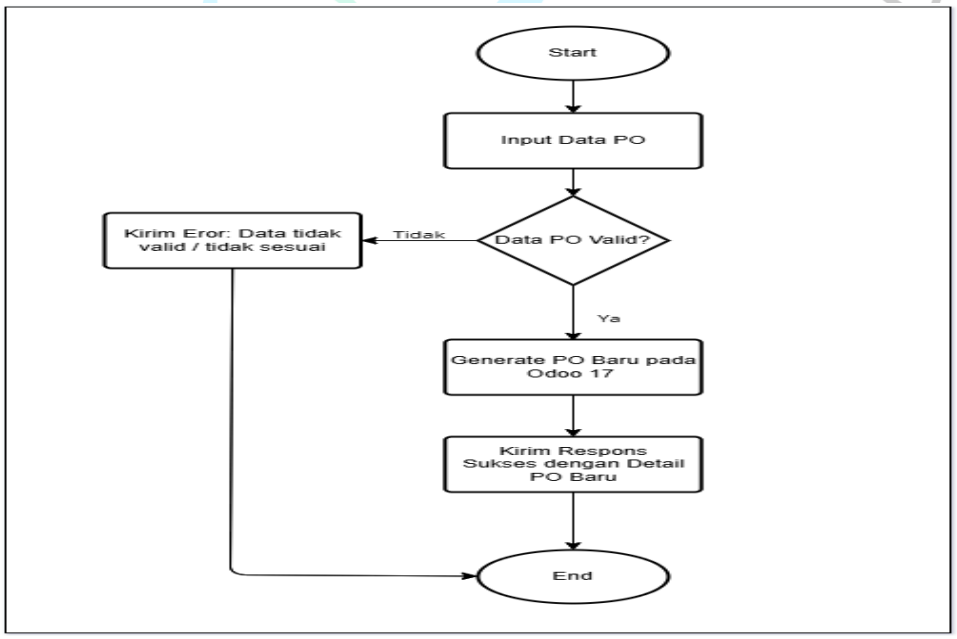
Gambar 3.33 *Flowchart Get Purchase Order* API

Setelah verifikasi token berhasil, *user* dapat meminta daftar PO dan detail PO dengan menginput ID PO pada API, kemudian sistem akan mengambil daftar PO dari Odoo dan mengembalikannya dalam format JSON. Berikut hasil dari pengembangan API *Get data Purchase Order*:



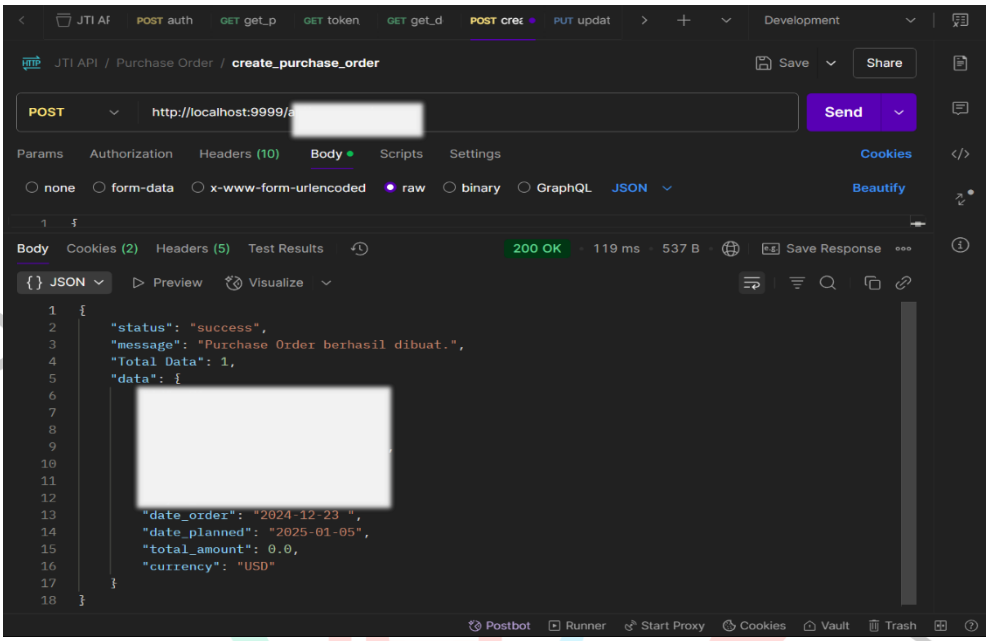
Gambar 3.34 Pengujian API *Get Purchase Order*

3. *Create Data Purchase Order*



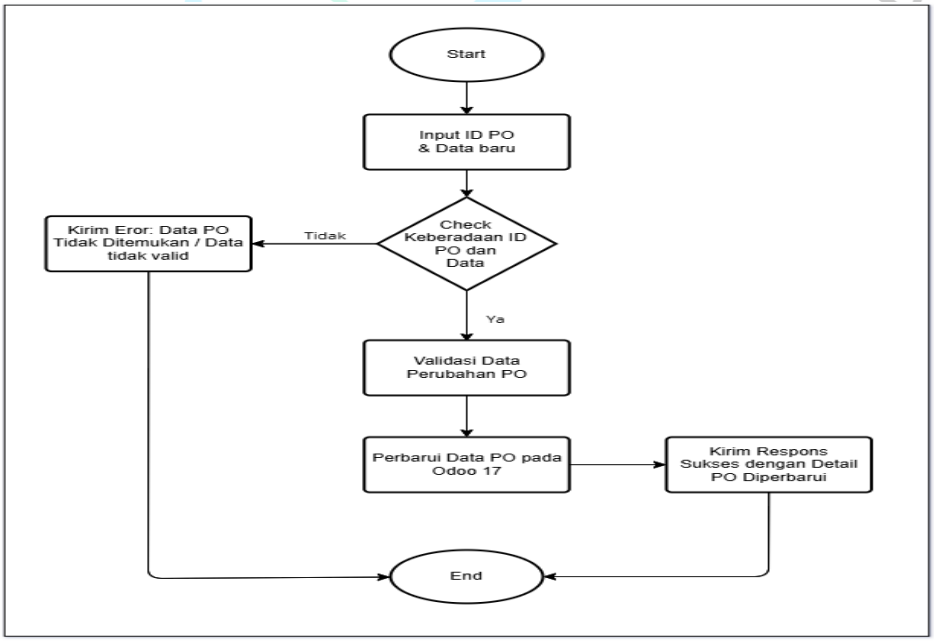
Gambar 3.35 Flowchart *Create Purchase Order*

Saat *user* ingin menginput data PO melalui *create*, maka sistem akan mencoba untuk memverifiikasi token user, validasi data PO yang dikirimkan, Jika berhasil maka sistem akan membuat PO baru dalam sistem Odoo 17. PO yang baru dibuat akan dikembalikan sebagai *respons* JSON kepada user. Berikut hasil pengujian API *create Purchase Order*:



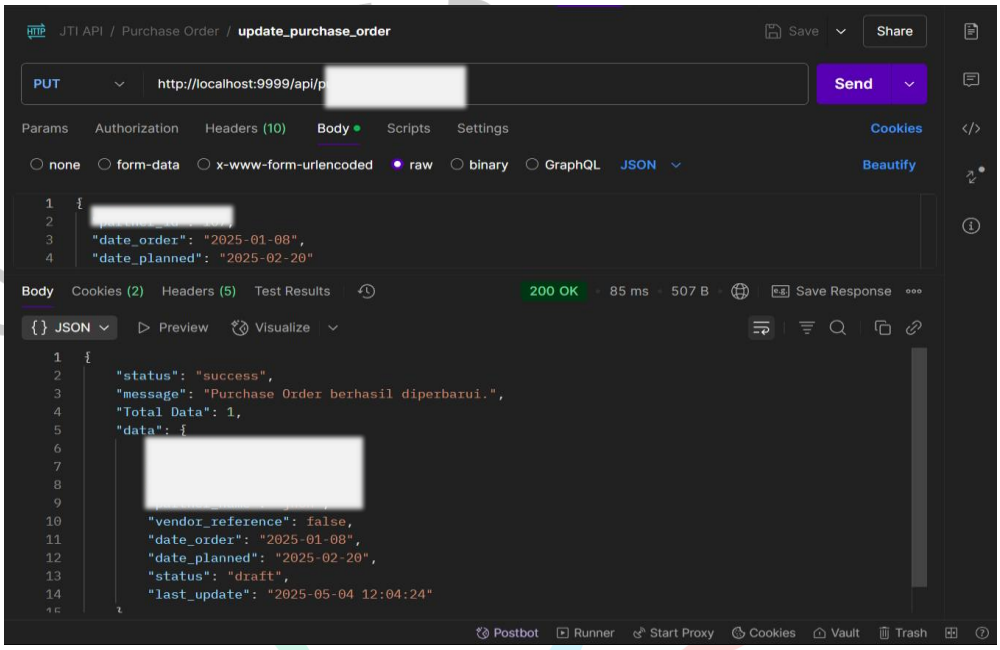
Gambar 3.36 Pengujian API *Create Purchase Order*

4. *Update Data Purchase Order*



Gambar 3.37 Flowchart *Update Purchase Order*

Untuk melakukan *update* pada *endpoint* API, *user* terlebih dahulu harus memasukan ID PO pada *header* HTTP, kemudian user memasukkan data PO yang ingin di *update*. Setelah itu sistem akan memverifikasi token dan periksa keberadaan PO. Jika PO ada, data akan diperbarui sesuai dengan *input* yang diterima. Kemudian dikembalikan kepada *user* berupa data JSON. Berikut hasil dari pengembangan *API update Purchase Order*:

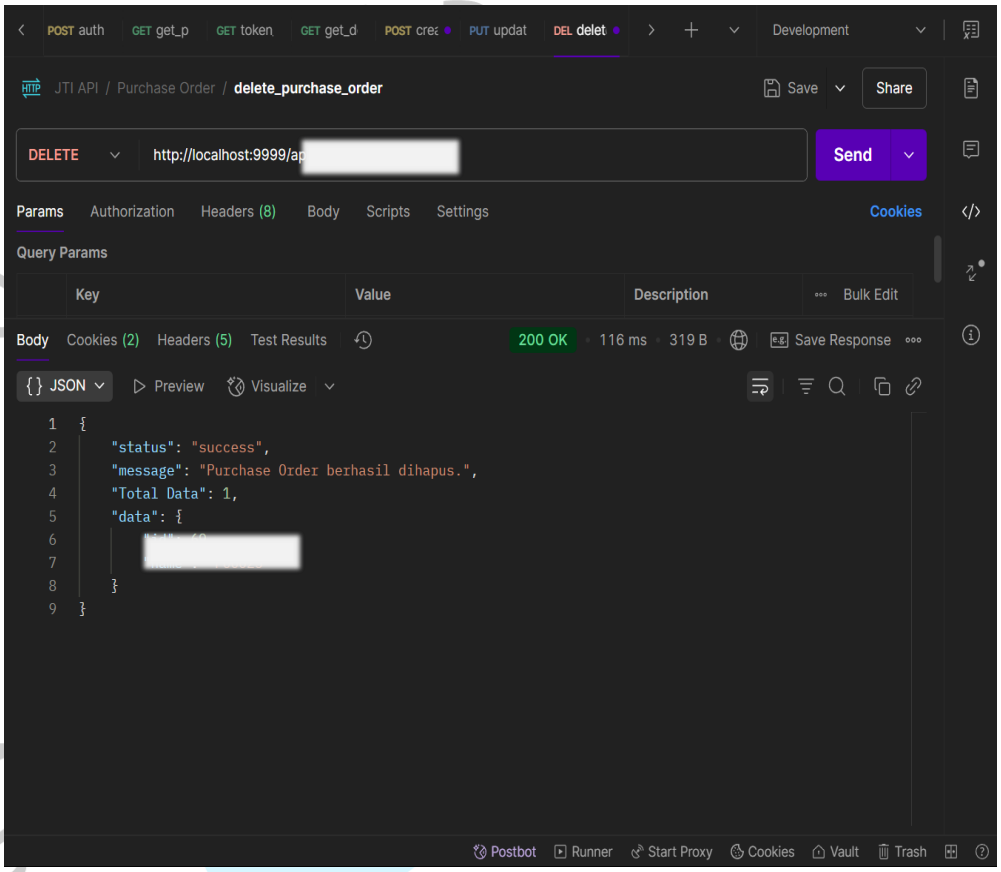


Gambar 3.38 Pengujian API *Update Purchase Order*

5. *Delete Data Purchase Order*

Gambar 3.39 Flowchart *Delete Purchase Order*

Sama seperti sebelumnya, *user* diharuskan memasukan ID PO melalui *header* HTTP untuk melakukan *delete* data. Kemudian sistem akan memverifikasi token dan pastikan PO yang akan dihapus sudah memenuhi syarat. Jika valid, PO akan dihapus dari sistem dan akan mengembalikan format JSON terkait data yang dihapus. Berikut hasil dari pengembangan API *delete purchase order*:



Gambar 3.40 Pengujian API *Delete Purchase Order*

Praktikan berhasil mengembangkan modul *custom* API untuk *Purchase Order* (PO) yang terintegrasi dengan Odoo versi 17 menggunakan metode CRUD (*Create, Read, Update, Delete*). API ini memungkinkan pengelolaan data PO secara eksternal dengan memastikan keamanan melalui token *authentication*. Setiap *endpoint* API berfungsi untuk mengambil, membuat, memperbarui, dan menghapus data PO, serta telah diuji menggunakan Postman untuk memastikan fungsionalitasnya. API yang dikembangkan oleh Praktikan ini siap untuk diterapkan untuk sistem E-TEKIND dengan Odoo versi 17.

3.3 Kendala Yang Dihadapi

Selama masa Kerja Profesi (KP) sebagai IT Support di PT Jaya Teknik Indonesia, Praktikan harus menghadapi beberapa kendala dan tantangan, seperti mempelajari penggunaan sistem Odoo dan *framework* Flutter, yang mana kedua teknologi tersebut masih awam bagi Praktikan. Proses pembelajaran yang Praktikan lakukan untuk kedua teknologi ini membutuhkan waktu yang lebih agar dapat optimal dalam menguasai kedua teknologi tersebut.

Selain itu, pembuatan API menggunakan Postman juga menjadi tantangan bagi Praktikan, dikarenakan *tools* tersebut belum benar-benar Praktikan kuasai secara lebih mendalam sehingga Praktikan perlu mempelajari bagaimana alur dan fungsionalitas Postman dalam menguji dan mengelola API.

Proses pembelajaran ini menjadi semakin menantang dengan adanya waktu yang terbatas, karena Praktikan juga harus menyelesaikan berbagai tugas yang diberikan dalam waktu yang sudah ditentukan. Hal ini menuntut Praktikan untuk terus belajar sekaligus beradaptasi dengan kecepatan yang diperlukan dalam pekerjaan sehari-hari.

3.4 Cara Mengatasi Kendala

Untuk mengatasi kendala dan tantangan yang dihadapi, Praktikan melakukan pendekatan *learning on the job,* pendekatan ini memungkinkan Praktikan dalam mempelajari teknologi dan *tools* yang digunakan sembari mengerjakan proyek yang diberikan. Praktikan memanfaatkan setiap kesempatan untuk belajar langsung melalui pengalaman yang diperoleh saat menerjakan tugas.

Tidak hanya itu, Praktikan juga memanfaatkan waktu istirahat untuk mempelajari dokumentasi dari teknologi dan *tools* yang digunakan, seperti Odoo, Flutter, dan Postman. Dengan pendekatan ini, Praktikan dapat memperdalam pemahaman terhadap teknologi tersebut secara bertahap, tanpa mengabaikan tanggung jawab terhadap proyek yang sedang dikerjakan.

Dalam mengatasi kendala yang dialami oleh Praktikan tidak terlepas oleh peran pembimbing kerja yang selalu memberikan arahan dan

solusi ketika Praktikan menghadapi kendala. Pembimbing mengajarkan Praktikan tentang Odoo, Flutter, dan pengembangan API secara teknis, sekaligus memberikan arahan yang jelas terkait mekanisme pengerjaan berbasis industri. Bantuan pembimbing ini membantu Praktikan memahami bagaimana cara menjalani dunia pekerjaan, khususnya di bidang teknologi, sehingga Praktikan lebih siap menghadapi tantangan di industri IT. Dengan bimbingan tersebut, Praktikan mampu melewati berbagai kendala serta terus termotivasi dalam proses belajar.

3.5 Pembelajaran Yang Diperoleh dari Kerja Profesi

Selama menjalani Kerja Profesi (KP) di PT Jaya Teknik Indonesia, Banyak pengalaman dan pembelajaran berharga yang Praktikan peroleh. Praktikan menjadi lebih mengerti tentang cara menjalankan dan mengembangkan modul-modul pada sistem Odoo, serta memahami cara kerja sistem Odoo secara lebih mendalam. Praktikan juga mempelajari cara menggunakan Flutter untuk pengembangan aplikasi mobile, termasuk cara melakukan pembaruan pada versi Android agar aplikasi dapat berjalan dengan optimal di perangkat terbaru.

Pembelajaran lainnya adalah Praktikan mendapatkan pemahaman lebih dalam mengenai penggunaan Postman, dimana Praktikan kini dapat menguji dan mengelola API dengan lebih efisien. Selain itu, Praktikan juga menjadi lebih mampu beradaptasi dengan jadwal kerja yang ada, di mana Praktikan dapat belajar sambil menyesuaikan waktu untuk menyelesaikan tugas-tugas yang diberikan. Kemampuan ini memungkinkan Praktikan untuk terus berkembang dan menyelesaikan proyek-proyek berbasis teknologi informasi dengan lebih efisien di dunia industri.

Selama pelaksanaan KP, Praktikan berhasil menerapkan beberapa mata kuliah yang telah Praktikan pelajari selama menempuh perkuliahan di UPJ khususnya pada program studi Sistem Informasi, di antaranya:

1. ***Enterprise Resource Planning***

   Mata kuliah ini mendasari pemahaman Praktikan tentang sistem Odoo sebagai platform ERP yang digunakan perusahaan.

2. **Sistem Basis Data**

   Mata kuliah ini membantu Praktikan dalam pengelolaan dan manipulasi data pada modul Odoo dan pembuatan *Custom* API *Purchase Order*.

3. **Perancangan dan Pemrograman Web**

   Mata kuliah ini juga berperan penting dalam pembuatan template QWeb untuk laporan PDF, Excel, hingga tampilan sistem Odoo perushaan.

4. **Mobile Device (Programming) Technology**

   Mata kuliah ini diaplikasikan oleh Praktikan dalam pengembangan dan pembaruan aplikasi mobile menggunakan Flutter.

5. **Pengolahan Informasi Berbasis Bahasa Pemrograman Script**

   Mata kuliah ini memberikan Praktikan pendalaman yang kuat khususnya untuk pemrograman Python yang sangat dibutuhkan untuk pengembangan backend Odoo dan API *Purchase Order*.

6. **Pengembangan Aplikasi Perangkat Lunak**

   Mata kuliah ini mendukung kemampuan Praktikan dalam merancang, mengembangkan, dan menguji modul serta aplikasi E-TEKIND.

Dengan keenam kombinasi mata kuliah yang telah dipelajari selama perkuliahan telah terbukti memberikan dasar yang kuat bagi Praktikan untuk mengaplikasikan pengetahuan secara langsung dan efektif selama pelaksanaan Kerja Profesi di PT Jaya Teknik Indonesia.